



AUGMENTED GENERATION OF REACTIVE AMBIENTS ON SURFACES



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Towards educational
places for action,
discussion and
reflection to support
creative learning on
interactive surfaces

SUPERVISED BY
DR. JAVIER JAÉN MARTÍNEZ

ALEJANDRO CATALÁ BOLÓS
PHD THESIS · JULY 2012

AGORAS: AUGMENTED GENERATION OF REACTIVE AMBIENTS ON SURFACES

**Towards educational places for action, discussion and reflection to
support creative learning on interactive surfaces**

Alejandro Catalá Bolós

DEPARTAMENTO DE SISTEMAS INFORMÁTICOS
Y COMPUTACIÓN

*A thesis submitted for the degree of
Doctor of Philosophy in Computer Science*

Supervised by

Dr. Francisco Javier Jaén Martínez

July 2012



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

PhD. Thesis

© Alejandro Catalá Bolós. Printed in Valencia, Spain 2012.

This work is subjected to copyright. All rights are reserved. Some content of this document has been partially published in scientific papers. A list of the corresponding publications can be found in the “Conclusions” chapter for further acknowledgement.

This thesis has been partially supported by the Spanish Ministry of Education under the National Strategic Program of Research and Project TSII2010-20488. A. Catalá has been supported by a FPU fellowship from the Ministry of Education with the reference AP2006-00181.

Credits of the cover: Abel Gómez Llana and Nacho Hurtado Bosch

Supervisor

Dr. Francisco Javier Jaén Martínez

ISSI-Futurelab, Departamento de Sistemas Informáticos y Computación, Universitat Politècnica de València (Spain)

External Reviewers

Prof. Dr. Anton Nijholt

Universiteit Twente (The Netherlands)

Dr. Marcos Roberto da Silva Borges

Universidade Federal do Rio de Janeiro
(Brazil)

Dr. Diego López de Ipiña González de
Artaza

Universidad de Deusto (Spain)

Thesis Defense Committee Members

Dr. José Hilario Canós Cerdá

Universitat Politècnica de València
(Spain)

Dr. Elena María Navarro Martínez

Universidad de Castilla-La Mancha
(Spain)

Prof. Dr. Anton Nijholt

Universiteit Twente (The Netherlands)

Dr. Diego López de Ipiña González de
Artaza

Universidad de Deusto (Spain)

Dr. Begoña Gros Salvat

Universitat de Barcelona (Spain)

Resum

La creativitat és una habilitat d'especial interès per al desenvolupament humà donat que és una de les dimensions que permet a l'individu i a la societat en definitiva afrontar nous problemes i reptes de manera satisfactòria. A més d'entendre la creativitat com una sèrie de factors relatius a l'individu creatiu, s'ha de tindre en compte el grau de motivació intrínseca, l'entorn i altres factors socials que poden tindre un efecte rellevant en el desenvolupament d'aquesta important habilitat. Per la qual cosa resulta d'interès explorar-la en el context d'utilització de les tecnologies d'informació.

En particular, donat que els processos comunicatius, l'intercanvi de idees, i la interacció col·laborativa entre individus són característiques que es troben darrere dels processos creatius i que a més a més són facilitats en gran mesura per les taules interactives, una de les principals contribucions d'aquesta tesi és precisament l'exploració de la adequació de les superfícies interactives en tasques de construcció col·laborativa amb estudiants adolescents. Partint d'aquest estudi, que aporta evidència empírica sobre la capacitat potencial de la tecnologia de superfícies interactives per a la estimulació de la creativitat, aquesta tesi presenta AGORAS. Es tracta d'un *middleware* per a la construcció de ecosistemes de joc 2D per a taules interactives, on la idea última és que el entorns d'aprenentatge creatiu del futur consideren activitats d'aprenentatge més atractives i recompensant com seria jugar al joc de creació jocs.

En el context d'aquesta tesi, un model d'ecosistema basat en entitats ha sigut desenvolupat, el qual permet la seua animació en termes de simulació física. A més, una aproximació basada en regles però millorada per a permetre l'edició de paràmetres d'accions has sigut proposat per a suportar l'expressió de comportament lògic. Les regles utilitzen expressions visuals de flux de dades per a suportar la seua edició en el editor proposat per a superfícies interactives. Totes aquestes components en conjunció als editors i simuladors han sigut construïts sobre un Toolkit d'interacció per a superfícies interactives desenvolupat també en aquesta tesi. El model i les ferramentes han estat provades per a la creació d'alguns prototips de jocs bàsic però funcionals.

Resumen

La creatividad es una habilidad de especial interés para el desarrollo humano dado que es una de las dimensiones que permite al individuo y en última instancia a la sociedad enfrentarse a nuevos problemas y retos de forma satisfactoria. Además de entender la creatividad como una serie de factores relativos al individuo creativo, debe tenerse en cuenta que el grado de motivación intrínseca, el entorno y otros factores sociales pueden tener un efecto relevante sobre el desarrollo de esta importante habilidad, por lo que resulta de interés explorarla en el contexto de utilización de tecnologías de la información.

En particular, dado que los procesos comunicativos, el intercambio de ideas y la interacción colaborativa entre individuos son un pilar fundamental en los procesos creativos, y también que en gran medida todas ellas son características mayormente facilitadas por las mesas interactivas, una de las principales contribuciones de esta tesis consiste precisamente en la exploración de la idoneidad de las superficies interactivas en tareas creativas colaborativas de construcción en estudiantes adolescentes. Partiendo del estudio realizado, que aporta evidencia empírica acerca de la adecuación de las superficies interactivas como tecnología de potencial para el fomento de la creatividad, esta tesis presenta AGORAS: un middleware para la construcción de ecosistemas de juegos 2D para mesas interactivas, y cuya idea final es entender actividades de aprendizaje más enriquecedoras como aquellas que permiten la propia creación de juegos y su posterior consumo.

En el contexto de esta tesis también se ha desarrollado un toolkit básico para construcción de interfaces de usuario para superficies interactivas, se ha desarrollado un modelo de ecosistema basado en entidades que son simulables de acuerdo a leyes físicas; y se ha dotado al modelo de aproximación basada en reglas de comportamiento enriquecidas con expresiones dataflows y de su correspondiente editor para superficies. Se ha demostrado que este middleware junto con las herramientas de edición y simulación construidas permite la construcción de ecosistemas de juegos funcionales, que serán el núcleo principal sobre el que desarrollar los futuros entornos sobre superficies para el aprendizaje creativo a través del juego.

Abstract

Creativity is a skill of special interest for human development since it is a dimension that allows individuals and, eventually, society to face new problems and challenges successfully. Besides understanding creativity as a set of factors related to the creative individual, it must be considered the intrinsic motivation, the environment as well as other social factors that ultimately can have an effect on the development of such a relevant skill. Thus, it is interesting to explore it in the context of using new information technology.

Specifically, because communicative processes, ideas exchange and collaborative interaction between individuals are the characteristics behind creative processes which, moreover, are to a great extent facilitated by interactive tabletops, one of the main contributions of this dissertation is precisely exploring the suitability of interactive surfaces in collaborative creative tasks with teenager students. Departing from this study, which provides empirical evidence on the potential for tabletop technology to foster creativity, this thesis presents AGORAS, a middleware for the construction of 2D game ecosystems for tabletops, whose main contribution for the future creative learning environments is the consideration of more rewarding and engaging learning activities such as those focused on playing to create games and their subsequent play.

In the context of this dissertation, an ecosystem model based on entities to be enacted according to simulated physics has been developed. In addition, a rule approach has been adopted but tailored to support enhanced expressiveness in terms of consequent action parameters to provide logic behavior in the ecosystems. Rules rely on visual dataflow expressions being supported by a suitable editing tool on the surface. All these middleware components along editing and simulation tools have been built on top of a specific interaction toolkit for surfaces developed in this thesis. The model and tools have shown their capability to support the creation of basic and functional game prototypes.

Keywords

Creativity, Learning Technologies, Tabletop, Interactive Surface, Surface, Tangible User Interaction, Game model, Ecosystem Simulation, Physics Simulation, Empirical study, Rule, Rule Behavior, Dataflows, Dataflow Comprehension, Ambient Intelligence.

Dedicatoria

Dedicado a los que siempre han estado a mi lado.

A Rafael y M^a Dolores, mis padres, por su esfuerzo y sacrificio. Gracias por haberme siempre ofrecido su experiencia y mostrado el camino a seguir pese a las dificultades.

A mis hermanos y sus respectivas familias. M^a Luisa y Edgar, Jorge y M^a del Mar, Elvira y Rafa han sido más que hermanos, como “segundos padres” para mí.

A los más pequeños, Nerea, Ricardo Iván, Lorena y Álvaro, y todos los que están por llegar. Su felicidad y alegría son como una inagotable fuente de energía.

A Nelly, cuyo cariño me ha enseñado a valorar aquello que tengo y a apreciar más si cabe a las personas que quiero. Gracias por enseñarme a ver las muchas cosas que hay en la vida.

Todos ellos han estado siempre ahí, aunque no siempre he podido estar a su lado como hubiera querido.

También quisiera acordarme de los verdaderos amigos, que son verdaderos por el hecho de acordarse de ti en cualquiera de las circunstancias. Gracias por vuestra necesaria amistad.

Gracias a todos.

Acknowledgements

I would like to acknowledge all the people who, being aware or not, have helped me and contributed to the development of this work in many different ways, either direct or indirectly. Now I remember many friends, many colleagues and many researchers who came across in conferences, informal meetings, coffee breaks, etc. Thanks to all of them for their time, their knowledge sharing, their support, and their friendship.

First at all, being aware that the available space here is very limited and it therefore makes even more difficult to express all the feelings in the right deserved words, I want to express my sincere gratitude to Dr. Javier Jaén, for his support since the beginning, supervising this thesis and his encouragement in the course. He has been much more than my supervisor and has put the person first. I appreciate his effort too and I am very grateful for that. There are many things to learn from him.

I would also like to acknowledge to those who sometime worked in the FutureLab team. Their collaboration has been very important. Special thanks must go to who has worked closely to me: Jose Antonio Mocholí, Jose Miguel Esteve, Paula Carrasco, Fernando García, Patricia Pons, Adrià Martínez and Jose Azorín. Their enthusiasm and effort on their specific tasks and projects have contributed very positively on the development of this research.

In addition, I would like to thank everyone in the *Software Engineering and Information Systems* research group, for providing a nice working environment and supporting me during all this time. I encourage them to go on with their respective research lines, and specially start up new ones.

I especially remember to the colleagues with whom I enjoyed many good moments at lunch time and outside of the office: Abel, Cristóbal, Jose Antonio, Manolo, Nelly, Nour, Carlos, Elena, Javier, Jennifer, Isabel, Gonzalo, Jose Miguel, Raquel, Víctor, David, Rogelio, Felipe, M^a Eugenia, and Artur. They must know that this has been possible thanks to them.

Finally, I would also like to thank to all the kind and wonderful people of the *Human Media Interaction* department at the University of Twente. I am especially grateful to Dr. Betsy van Dijk for her time and support during my stay in Enschede. Her comments were really useful to round off this work. *Dank u wel.*

Table of Contents

1	Introduction	1
1.1	Motivation.....	1
1.2	Creativity in Education	3
1.3	IT in the Educational Context and its Suitability for Creative Learning.....	5
1.4	Creation of Games, Creative Learning and Tangible User Interfaces	10
1.5	Problem Statement	13
1.6	Overall Aim and Objectives.....	15
1.7	Research Methodology	16
1.8	Research Hypothesis	17
1.9	Outline of the Thesis	18
2	State of the Art.....	21
2.1	Introduction	21
2.2	Related Work.....	22
2.2.1	Supporting Programming and Performing in a Pre-established World	22
2.2.2	Creating Characters for an Ecosystem World	27
2.2.3	Advanced Platforms to Create Game based Ecosystems by Programmers.....	30
2.3	Feature Based Comparison	33
3	A Model Supporting Physically-Based 2D Ecosystems to Foster Creativity	39
3.1	Introduction	39
3.2	Model Requirements: Editing and Simulating.....	42
3.3	Ecosystem Model	43

3.4	Entity Model	44
3.5	Physically-based Entities	46
3.6	Exploring Creativity in Collaborative Construction Tasks	55
3.6.1	Creativity Assessment Model	55
3.6.2	Creativity Tangible Simulator Component	60
3.6.3	Experimental Evaluation	63
3.6.4	Overall Discussion.....	83
3.7	Conclusions	86
4	A Model of Reactive Entities Based on DataFlow Enhanced Rules.....	87
4.1	Introduction	87
4.2	Rule-based Behavior	89
4.3	A Generic DataFlow Model	93
4.4	User based Study on DataFlow Comprehension	98
4.4.1	Participants.....	98
4.4.2	Materials	100
4.4.3	Method and Procedure	103
4.4.4	Task.....	103
4.4.5	Results and Discussion.....	103
4.5	A Tabletop Based Rule Editor	110
4.5.1	Tabletop Design.....	110
4.5.2	Implementation.....	112
4.6	Conclusions	113
5	Surface Interaction Toolkit Support.....	115
5.1	Tangible Surface Interfaces.....	115
5.2	Touch Technologies	118
5.2.1	Microsoft Surface.....	122
5.3	User Interface Controls	124

5.3.1	Basic controls.....	125
5.3.2	Input Management.....	129
5.3.3	Complex controls.....	131
5.4	TangiWheel: A Hybrid Design for Collection Exploration	134
5.4.1	A Survey on Widgets for Collection Manipulation.....	135
5.4.2	Design Discussion.....	141
5.5	Experimental Evaluation of TangiWheel.....	149
5.5.1	Participants.....	151
5.5.2	Equipment.....	151
5.5.3	Method.....	151
5.5.4	Experiment 1: Acquisition and Basic Manipulation.....	154
5.5.5	Experiment 2: Match to Sample (MTS).....	156
5.5.6	Experiment 3: Series Edition.....	160
5.5.7	Questionnaire Results.....	166
5.6	Conclusions	171
6	Simulation Middleware.....	175
6.1	Introduction	175
6.2	Meta-Object Instantiation	176
6.2.1	Type System.....	176
6.2.2	Event System	179
6.2.3	Property Definitions	179
6.2.4	Actions Library.....	181
6.2.5	Data Processors.....	182
6.3	Ecosystem Enactment	183
6.4	Meta-Model Performance Verification.....	188
6.5	Physics Simulation Stress Test.....	192
6.6	Case Studies.....	193
6.6.1	A Pong-like Ecosystem	193

6.6.2	An Asteroids Ecosystem.....	201
6.7	Conclusion.....	207
7	Conclusions.....	209
7.1	Summary.....	209
7.2	Results of the Thesis.....	213
7.3	Acknowledgements.....	214
7.4	Further Research.....	215
	Appendix A. Experiments on Creativity	217
	Appendix B. DataFlow Comprehension	227
	Appendix C. Pong AGORAS Specification	253
	Appendix D. Asteroids AGORAS Specification	263
8	References.....	281

Chapter 1

Introduction

This chapter provides an overview of the research contained in this dissertation. It presents the motivation of the investigated problem as well as the aim and primary objectives of this research. Furthermore, it establishes the methodology research and states the hypothesis. The chapter concludes with the outline of the dissertation.

1.1 Motivation

According to a report of the Spanish Ministry of Education entitled “Las cifras de la Educación en España. Estadísticas e Indicadores”, nearly 30% of Spanish secondary school students do not manage to terminate this stage of their education successfully (*Ministerio, 2007*). In the words of Fernando Reimers, Professor of Education at the University of Harvard (*Pérez, 2008*),

“... almost 20 years after the law proposed this educational reform, it is worrying. And it is especially so because one of the results of globalization is precisely to give greater opportunities to the best qualified people.”

Although this educational failure may be due to multiple reasons and can be analyzed from many points of view, including the social, family, cultural and even economic areas, we are convinced that one of the factors that has contributed to this situation is the under-use of active educational strategies that would allow Spanish schoolchildren to develop their creative abilities and increase their motivation to learn. As Fernando Reimers himself recognizes, “Educating is not informing, but developing people’s talent and character”. One indication which supports the theory that we do not effectively apply teaching strategies that encourage pupils to develop creative problem-solving skills and abilities can be found in the 2006 PISA report from the OECD (*Pisa, 2007*), which does not measure pupils’ knowledge as such but, rather, their

capacity for understanding and solving real problems by applying knowledge from each of the principal PISA areas. In fact, this report classifies Spain as below the OECD average for all the assessed indicators and in the last place regarding pupil performance.

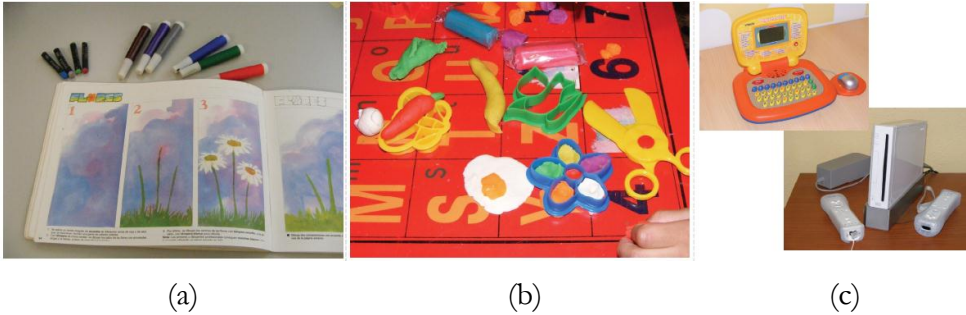


Figure 1. Traditional approaches regarding play and learning.

On the role of creativity in human development, when something new is created, we are not just consuming knowledge from other pre-existing sources. Instead, we are actually forming, reproducing and improving ideas and/or concepts, which requires heavier cognitive effort. Consequently, we may become a new source of knowledge. In this sense, creating is an active way towards positive learning reinforcement. In Figure 1, several usual approaches to games for learning are illustrated, some of them used from childhood. First, the Figure 1 (a) shows creation based on “paper and pencil”, which supports effectively the creative process. This may support playing by using the creations for narrative purposes. Second, another highly attractive approach capable of keeping motivation in desirable levels is the one in Figure 1 (b), based on model clay and modeling tangible tools. This approach is basically focused on the creation phase and the subsequent play, both in a tangible way. However, in computer-based technological approaches, the effective possibilities that these traditional approaches to play have offered seem to be not naturally supported anymore. Moreover, computer-based educational alternatives provide other advantages such as introducing players in a process of digital literacy (*Gros, 2008*) and the development of computational thinking skills (*Wing, 2006*), which are especially important and valuable for the creative and information society formation (*Resnick, 2002*). Figure 1 (c) shows a typical electronic toy supporting a wide variety of learning activities, and a modern video game console. These two examples do not support creation in general, although recently some effort is being made in the videogame industry to produce videogame titles addressing

creativity¹. Moreover, interaction is not as natural as in non-technological settings because it is confined to the possibilities that controllers offer, and foundations and assumptions of active learning theories are not being considered in many of these technological approaches.

Hence, developing a suitable technological platform with a natural way of interaction to foster creativity by means of involving users in active and social tasks would be challenging, but also interesting as this exploration on how technology can address such a difficult problem may open new perspectives in research. This is the main motivation leading the present dissertation.

The rest of this chapter is organized as follows. Some background on creativity and the suitability of some technologies for creative learning is provided. The motivating key ideas related to the use of tangible user interfaces and tabletops in our proposal are then discussed. Finally, the problem, objectives and research methodology and hypothesis will be described.

1.2 Creativity in Education

Education with non-technological support has mostly focused on passive strategies, which are primarily instantiated in the way of “master classes” relying exclusively on behaviorism principles. However, active strategies focused on the active role that learners must adopt, are spreading and getting more usual nowadays. These are founded on cognitive learning theories such as constructivism (*Dewey, 1963*), experiential learning (*Kolb, 1984*), and situated learning (*Brown, 1989*). In a way or another, all these theories rely on the idea that “to learn is to do”, and at the same time consider reflection and discussion processes necessary in several ways. Similarly, creative learning is also considered an active strategy, because it takes common assumptions from the aforementioned theories and because creative thinking specially requires active discussion and exchange of ideas between peers.

A serious approach to creative learning requires a definition of the concept of creativity and a discussion of its importance in the preparation of the present pupils for the future challenges they will face in society and in the labor market. In order to define the concept of creativity, we can go back to ancient Greek and Roman texts for the first known references to terms such as creativity and

¹ Several examples are “Spore” (<http://www.spore.com>) and “Little Big Planet” (<http://www.littlebigplanet.com/>).

imagination. More recently, studies such as those carried by Treffinger - who cites over 100 definitions found in the literature (*Treffinger, 1996*) - or Aleinikov, Kackmeister and Koenig - who discuss 101 definitions provided by both adults and children (*Aleinikov, 2000*) - demonstrate the enormous complexity of the concept and the different perspectives from which it can be approached.

In spite of the wide variety of opinions and the fact that the study of theoretical frameworks for defining creativity is a line of research that is still in its early development, the report "Assessing Creativity: A Guide for Educators" from the National Research Center on the Gifted and Talented provides detailed categories of the different approaches (*Treffinger, 2002*). Among these, the proposal by Teresa Amabile stands out for its simplicity and capacity to join together aspects that had been suggested separately in other approaches (*Amabile, 1983*). According to Amabile, creativity arises as a result of the combination of three factors: knowledge, creative thought and motivation.

Knowledge consists of all the information possessed by individuals to solve a problem. Howard Gardner identifies two types: knowledge related to a profound understanding of a certain domain (in-depth understanding) and a more superficial understanding of multiple areas (in-breadth understanding) (*Gardner, 1993*). A balanced combination of both types is necessary to maximize individual creative potential, as Frans Johansson explains in "The Medici Effect" (*Johansson, 2004*). Johansson goes even further and suggests that in-depth knowledge improves when subjects interact in a group with other individuals with other points of view and knowledge levels. As we shall see later, this is one of the important factors in our research project.

Creative thinking can be summed up according to the model proposed by Amabile as the presence of the following individual abilities to a greater or lesser degree: the ability to disagree with others and experiment with different solutions to proposals chosen by the majority; the ability to persevere in difficult or problematic situations, and, finally, the ability to gestate ideas during periods in which we alternately forget the problem and return to it with a new perspective. We can therefore conclude that facilitating processes of collective discussion, reflection and experimentation with new ideas, favors the development of creative thinking. In terms of Sternberg's theory (*Sternberg, 1999*), these would be the synthetic, analytic and practical aspects of intelligence, which, he maintains are the keys to creativity. These aspects also play a crucial part in our proposed work.

Finally, a great number of researchers consider motivation as the most important factor in the development of creativity. Among others, Amabile,

Gardner and Sternberg stress the prevalence of intrinsic motivation, which is directed by interest, satisfaction and the challenges presented in order to solve a problem, over “external pressures” or extrinsic motivation. With all these in mind, information technology must be taken into account because some other creativity theories consider also external social and environmental factors as relevant and supportive for creativity (*Sanyer, 2006*).

1.3 IT in the Educational Context and its Suitability for Creative Learning

Authors such as Scott, Leritz and Mumford (*Scott, 2004*), have analyzed more than 100 educational programs and initiatives designed to foment creativity, dividing them into four categories:

- Those oriented towards entertaining and generating ideas;
- Those whose aim is to give training in the use of images to describe ideas or situations;
- Those oriented towards training in cognitive processes by combining concepts and critical analysis; and finally,
- Those oriented towards developing divergent and convergent thought processes.

While many educational curricula do incorporate activities designed to generate new ideas, they do not tend to induce any debate and their effectiveness is widely questioned, especially when compared to other methods that combine different concepts and encourage divergent thinking.

Authors such as Amabile, Sternberg and Nickerson (*Sternberg, 1999*) support the idea that new pedagogical techniques for the development and evaluation of creative thought should consider the following aspects:

- reduce the number of rules that limit or control thought processes (many restrictions limit intrinsic motivation);
- offer maximum opportunities for choice in both defining the nature of the problem to be solved and formulating and experimenting with solutions;
- involve pupils in evaluating their own work and not make them fear failure, so that they understand that when their ideas fail, this can often be a source of new knowledge;
- promote creative collaboration between different individuals;

- seek stimulating learning environments; facilitate the explicit expression of the others' points of view and solutions; and finally, provide mechanisms for combining ideas, criticism and divergent thinking.

While technological revolutions tend to improve on existing systems, the introduction of new technology often involves profound changes in the processes they affect, especially in production methods. If this is paradigmatic in the appearance and evolution of computer science, sadly and surprisingly, expected changes have not had yet a significant impact in the education sector. Great efforts are being made to supply schools with computers (which in some cases are under-used), but the fact is that nothing seems to have changed drastically in the education methods and the class organization models for the last two centuries since the chalkboard was introduced in schools in the early 19th century. How can that be?

The individual capacity to comply exactly with the instructions given, with discipline and in silence, was in line with the needs of an industrial society and systems of mass production. The capabilities to take part in collaborative processes and to direct divergent thinking as the motor of creativity and innovation are essential in our current information society, which is organized into a network, and where information occupies the center of the productive processes. Educators should therefore not forget that our objective is the education of the complete person and not only the training of a work force, and that creativity is the resource necessary for tackling the social, cultural and environmental challenges that today's global society faces. As pointed in (*Resnick, 2002*), in order to give way to the so-called creative society, we do need some substantial advances in the role technology is playing in education.

Several taxonomies have been developed for classifying and better understanding the potential of IT in an educational context. (*Kindley, 2002*), for example, classifies e-learning systems according to the type of activity they convey, distinguishing between scenarios, simulation, and finally games. We find that this classification is not specific enough for our purposes; an alternative taxonomy more aligned with learning theories and their underlying teaching strategies, as the one done by (*Kebritchi, 2008*) in the field of computer games, would seem more appropriate. In the lack of an appropriate classification, we will review the different technological supports currently used in both traditional and modern learning applications, and analyze how they support to a greater or lesser degree, the pedagogical proposals for promoting creativity considered in the previous section.

1. Offline multimedia educational environments, typically in the form of CDs or DVDs, are interactive and allow learning methods to be defined which the user must comply with. Users can see how they are progressing, because there is immediate feedback, and activities include not only receiving information but also exercises, problem solving, scenarios, games, etc.

Cons: Although these systems focus on learning through creating artifacts, which is an important element in creativity, interaction is limited to one user, or perhaps two sharing the same application. This impedes in the terms of Sternberg and Johansson, what we have described as establishing in-breadth knowledge and, in the terms of Amabile, Sternberg and Nickerson, the creative collaboration between individuals and the explicit expression of the different points of view and solutions provided by others. This limits the divergent and convergent thought processes, which, as we have seen, Scott, Leritz and Mumford considered important.

2. Web-based educational environments, on their side, tend to bring learning management systems a little further and may allow tutors to follow students' progress. Pupils can share impressions and opinions with classmates and discuss the activities, thus encouraging critical and divergent thinking.

Cons: Activities are in general based on classical instruction and when oriented towards creativity they are usually in the form of individual activities. There are exceptions, such as collaborative web applications, but as the participants are not physically present, this means that methods of discussion, criticism and sharing solutions are generally primitive and based on textual information in the form of chat rooms, blogs, social networks, wikis, communities and forums. As with offline applications, we still have problems from the point of view of creative teaching strategies (although slightly mitigated by the existence of certain levels of communication between the participants).

3. By the fact that Virtual world based educational environments can come both on physical supports or be available on-line, they could be considered as specific cases of the two previous categories. However, we will treat them separately because of the specific properties they exhibit. Although the 3-D virtual world concept and the romantic idea of "Metaverse" which Neal Stephenson described in his science fiction book "Snow Crash", in which humans used avatars for interaction with software agents in a 3-D world as a metaphor of reality, has been in existence for many years, it was not until 2000 that Second Life caused a great impact all over the world (*SecondLife*, 2012). Second Life, defined as a virtual 3-D world created by its residents, offers a high degree of social interaction and creative possibilities of many different types in the user's

virtual world. The user can also make virtual interactive objects, games and activities available to other users. Similar types of virtual world programs include OpenSimulator and OpenLife Grid (*OpenSim, 2012*)(*Osgrid, 2012*) based on the same simulation idea. Although these types of platforms were not explicitly conceived for learning, their wide range of possibilities gives them a great potential in education. In fact, they are already being used in the educational world for providing meeting places, with both synchronized and non-synchronized discussions, sharing educational resources, etc. within a virtual world.

Cons: The main drawbacks of these environments are that they isolate participants from the real world in which the discussions take place and, secondly, that they hamper the modeling of entities by non-expert users because of its 3-dimensional nature. New advances are being made in this direction, such as the game for the PlayStation 3 platform known as the “Little Big Planet”, in which users can concentrate on a limited number of previously defined behavior patterns and do not have to create new ones. Although these proposals do provide advances in terms of creative learning, they are still limited to individual activities with little participation by multiple agents, which would encourage processes of criticism, divergence and later convergence. Also, in this and other similar games, the line of argument is pre-defined in the form of a sequence of problems to be solved and the solution, although it may be achieved creatively, is also pre-defined.

4. With the arrival of mobile devices in the form of PDAs and UMPCs², the possibility of using some educational applications on these devices arises. The possibilities of mobile educational environments involve much more than simply adapting resources and applications to their small-size interface. Activity design can incorporate learning based on available contextual information. Social networks can be used in search of cooperation, using mobile attributes in a literal sense, and they can be provided with “just in time” information systems. A leading example of this type learning environment is Savannah (*Facer, 2004*), in which participants use mobile devices to study the behavior of wild animals and have to move and act within an actual physical space as if they lived in the forest. In this way, solutions emerge naturally, which are designed, discussed and tested by a group following the principles of creative learning in which pupils generate and discuss ideas collectively and dynamically without the barriers encountered in web-based or virtual systems.

² PDA stands for Personal Digital Assistant and UMPC for Ultra Mobile Personal Computer.

Cons: The limitations of this type of environment are those imposed by the interaction limitations of the devices themselves and by the fact that virtual information cannot be easily handled collectively, since it is presented in the individual phones of each member. This greatly limits the type of creative action that can be performed with this technology.

5. Augmented reality based educational environments can go beyond the purely virtual world, and combine the characteristics of mobility for creating a real space augmented with digital objects. This technology seeks to provide more authentic and physical interaction in both physical and virtual entities, and therefore introduces interfaces and physical objects to provide a more realistic feedback. Also, the fact of interacting with tangible physical objects allows more natural mechanisms to be activated for learning, which reinforces, as suggested by Vygotsky (*Vygotsky, 1978*), the learning of the participants. They also incorporate all the characteristics developed for virtual reality and, as far as possible, offer the advantage of mobility when mobile devices transmit the augmented environments.

Cons: Two approaches are used for augmented reality based on (i) immersive visualization devices and on (ii) uncoupled visualization devices. In the former, the biggest problem lies in the high cost of the devices and the difficulties involved in their use for educational purposes, because the visualization panels are close to the user's eyes. In contrast, the latter technology is both inexpensive and easy to handle but not so immersive.

6. Educational environments based on robots and mechanical elements, are programmable electronic devices with sensors and actuators. The concept of programmable blocks in combination with the Logo language and the Lego construction kit are especially important (*Resnick, 1988*)(*Resnick, 1996*). These systems inspired the construction kits PicoCricket (*Picocricket, 2012*), which permits artistic creations with lights, sound, music and movements, and Lego MindStorms NXT (*Mindstorms, 2012*) designed for the construction of robots. RoboCup (*Robocup, 2012*) is another educational initiative in the use of robots, involving diverse competitions in various categories. Here again, creative learning is fostered by the construction of physical objects.

Cons: Currently, the main disadvantages of these systems are their high cost, low degree of interaction (the robots have a limited number of pre-defined movements), and for inexperienced users they are still difficult to program.

1.4 Creation of Games, Creative Learning and Tangible User Interfaces

After an overview of how information technologies are currently applied in the context of education, we can affirm that although they all bring some elements that can boost at least several of the three pillars on which creative learning is built (knowledge, creative thought and motivation), these systems do hardly achieve the cooperation and group participation that is so often present in many non-technological activities, such as gaming, which, since infancy, constitute an essential part of our initial learning stage. These more physical and social activities, whose objectives deal with group participation, cooperation, competition, respect for the established rules, or achieving common as opposed to individual goals, do often aim to inculcate socio-cultural values and encourage confidence, esteem, contact, communication, coexistence and discussion among those who take part, and can indeed constitute excellent promoters of creative learning.

Although changes in technology have gradually headed at involving subjects more actively, and have introduced the possibility for users to create artifacts as a way for learning reinforcement, these approaches are still far away from obtaining the high motivation levels needed to optimal learning experiences.

To overcome this issue, videogames are being considered as an effective tool. These games which are not only for fun but especially for learning are known as “serious games”. They allow for active participation and high task engagement. Indeed these are also important features from the point of view of constructivist and experiential learning theories. As described in (*Michael, 2005*) some general benefits of using serious games in education are the ability to model more complex systems, a higher engagement with learning materials, interactivity and quick testing and evaluation of answers, proximity to learning strategies founded on constructivism, and cost savings by reducing training times and using virtual environments rather than in expensive real settings.

But digital serious games are not always advantageous. They are usually focused on a very specific range of knowledge, for concrete purposes, and non-oriented to the masses resulting in non cost-effective developments. An alternative to them is the use of specific commercial games, in combination with traditional learning activities. Some studies using such an approach have been conducted in the context of formal learning settings (*De Freitas, 2007*)(*Ellis, 2006*)(*Gros, 2004*)(*Gros, 2007*)(*McFarlane, 2002*) but learning activities are normally focused on traditional tasks on paper, based completely on predefined

videogame content. These studies have concluded that videogames traditionally support three types of learning. The first one is learning from the tasks required from the games themselves. The second one is learning from the content of the game, although this does not usually correspond closely with the educational curriculum. The third one is learning by practicing the skills and abilities required by the game. These works have normally introduced a cyclical process considered important to learning, which consists of several phases including experimentation, reflection, activity and discussion. These processes are not usually supported entirely by digital games.

However, the use of games as a way of learning as suggested by Clark Abt, even before the digital age in the 60's, is one of the most effective strategies in fostering creativity (*Abt, 1970*). Abt proposed in his book on serious games that the game-creation process should be considered an important learning activity. This is a perspective that digital serious games do not seem to have been taking into consideration very much. He points out that the first learning phase, namely design and preparation, can be divided into two parts: a) the relatively passive preparation activity and b) the actual design of the game itself. The first one involves studying the background of the rules, roles, concepts, etc. The second one is more satisfying and involves inventing a simulation model of the process to be re-created, during which the different important variables involved with their interrelationships and dynamic interactions are controlled. If this is to be done satisfactorily, pupils must understand the process involved in the simulation, and in this way, increase not only their factual knowledge, but also of the interactions and processes involved. As all this activity is to be performed in group, it is also important from the point of view of social learning, which considers knowledge emerging from interaction and communication between individuals who pursuit shared objectives.

Our proposal is quite more ambitious and is not about building a creative problem-solving application in a specific domain, with a predefined behavior and pre-established reactions at the users' disposal. Following the ideas of Abt in the pre-digital age, we aim at "the construction of videogames" as the creative activity carried out. The interest, thus, does not lie in the game itself, but in the design activities, which will allow the group to acquire in-depth and in-breadth knowledge, put critical, convergent and divergent thinking into practice, providing high doses of intrinsic motivation or flow while these activities are being carried out. Consequently, what we propose is the creation of a platform that will profit as much as possible from traditional non-technological gaming activities, allowing pupils to create their own games according to the rules they themselves will lay down, and which, by means of digital technolo-

gies, will provide stimulating environments in which pupils will be able to experience interactively the results of their design decisions.

While play activities already naturally stimulate learning, videogames are also important components in the lives of modern teenagers. Bearing in mind contemporary social behavior, video games are an essential part of their culture and of the imaginary relationship they establish with the world, and to a certain extent, they affect their thinking and behavior. Therefore, the observation of the virtual entities they will have defined and created, in visually attractive ecosystems will foment the pupils' intrinsic motivations and their capacity to critically analyze through experiments the results of the decisions they have taken.

Tangible User Interfaces (TUI) combine control and representation in a single physical device (*Ullmer, 2001*). This concept both amplifies and simplifies the direct manipulation paradigm (*Shneiderman, 1983*) on which conventional graphic user interfaces (GUI) are based. In direct manipulation with GUI type graphic interfaces, users interact with digital information by selecting graphic representations (icons, windows, etc.) with pointing devices (e.g. a mouse). In the case of tangible interfaces such as interactive tables, users interact with the system with their hands and their fingers and also by manipulating specially configured physical objects. The concept goes thus far beyond the simple idea of multi-touch. This type of tables means much more than flat screens that can detect many fingers simultaneously. Interacting with the fingers still belongs to the idea of pointing devices, while interacting with physical objects can take us much farther. Such objects can represent abstract concepts or real entities; they can relate to other objects on the surface; they can be moved and turned around on the table surface. All these spatial changes can affect their internal properties and relationships with neighboring objects. These interfaces strengthen concepts such as "social interaction" and "collaboration" (*Hornecker, 2006*)(*Marshall, 2007*) or "game interaction" (*Gaver, 2004*). If they are compared to conventional WIMP interaction (windows, icons, menus and pointers) they provide a form of interaction much closer to the real world. While in WIMP interaction the user input is restricted to an ordered sequence of events (click, double click, etc.), which permanently limits the workflow, with interactive tables, any action is possible by any user, at any time and place, and in any order. Also, the seamless integration of physical control and visual output means that interacting with this type of systems takes place in the most natural and direct way possible. It is not necessary to know where to click to change a given parameter; whatever has to be changed can be touched and manipulated directly.

Collaboration, natural processes and clarity are properties that show future promise in educational applications. However, in order to analyze the prospects for a new type of technological infrastructure in creative learning, we should call to mind the three pillars that according to Amabile support this type of learning: knowledge, creative thinking and motivation. In the first place, it seems evident that interactive tables, with a group of participants, enhances the transmission of in-breadth knowledge thanks to direct communication among individuals with different knowledge levels and characteristics, as Johnson suggested in his book “The Medici Effect”. Secondly, as there is a tangible, touchable and shareable common space interaction, critical and divergent thinking is encouraged, alternative suggestions are offered by the rest of the group and, finally, consensus or divergent thinking is established. Such infrastructures put into practice the synthetic, analytic and practical aspects of Sternberg’s intelligence.

Finally, it has also been shown that this type of infrastructure is ideal for exploratory and creative activities and allows users to create constructions that would be impossible by other means (*Marshall, 2007*). They encourage intrinsic motivation and provide optimal learning experiences in which users are motivated to learn the effects of an action on the behavior of the interactive world by manipulating tangible elements. This has been recently showed in different areas such as music creation and performance (*Jordà, 2005*)(*Jordà, 2008*) or Logo programming (*Gallardo, 2008*). We are convinced that this type of exploratory interfaces will constitute the ideal platform for the non-trivial tasks involved in game definition and programming, such as the definition of the reactive behavior, choreographies of the game involved entities, and the relations and interactions between them.

1.5 Problem Statement

In short, the real problem that motivates the present dissertation is the lack of effective technological instruments and tools for fostering the development of creative skills. This lack is thought to be the result of neither involving engaging enough creative tasks, nor properly addressing processes such as action, reflection and collective discussion. Hence, we consider that a good approach to explore would be one using tangible surfaces, as they could easily facilitate these processes in co-location and face-to-face communication conditions. Consequently, our proposal will be aimed at the construction of a learning environment in which pupils can play to create games, facilitating in this way creative learning, applying the learning-by-doing philosophy and emphasizing the

three dimensions identified by Amabile: knowledge, creative thinking, and motivation.

In this sense, the relevance of our proposal relies on several factors: a) the suitability of interactive surface tabletops to support social learning since several individuals share a common physical space. As happens with non-digital games, communication during the creation, experimentation, and reflection is direct, non-computer mediated but computer-enhanced in our case; b) the simultaneous interaction naturally supported by interactive surfaces, being able to facilitate the construction of interfaces for conducting tasks in parallel and in cooperation; c) the support by the given software infrastructure for the construction of 2D ecosystems, making skills development not only be a consequence of consuming already pre-produced videogames with predefined contents, but instead a result of the creation and experimentation by active participants. Hence, cyclical processes such as reflection, creation, experimentation and collective discussion may be strengthened in the support of creative game-based activities on shared physical spaces based on surfaces; d) the new possibilities offered to teachers who may take this infrastructure as a tool in which they can prepare scenarios or case studies. In these, different entities and behaviors could be devised to try to foster reflection and discussion between students in a controlled and engaging way.

Our proposal is ambitious and it is not about building a creative problem-solving application in a specific domain, with a predefined behavior and pre-established reactions at the users' disposal. We aim at "the construction of videogames" as the creative activity to be carried out. The interest thus does not lie in the game itself, but in the design activities, which will allow the group to acquire in-depth and in-breadth knowledge, put critical, convergent and divergent thinking into practice, providing high doses of intrinsic motivation or flow while these activities are being carried out. What we thus propose is the creation of a platform that will profit as much as possible from traditional non-technological gaming activities, allowing pupils to create their own games according to the rules they themselves will lay down, and which, by means of digital technologies, will provide in the future stimulating environments in which pupils will be able to experience interactively the results of their design decisions.

1.6 Overall Aim and Objectives

The overall aim of this work is to advance in the development of an information system towards the learning environment of the future which properly supports social, constructivist and creative learning. With this objective in mind, according to what has been described before, the broad requirements to be satisfied are: 1) provide a common physical space which facilitates communication, fosters reflection, discussion and development of social skills; 2) allow more natural interaction by means of direct manipulation and even the usage of tangible elements; 3) situate learning in an interactive ecosystem, according to experiential and situated learning; 4) consider the creation of the ecosystem itself as an important part of play, giving support to the creation of different ecosystems to give way to different learning scenarios. These scenarios will entail the design of entities, actions, events, behaviors, etc.; 5) allow running the simulation and then interacting with the ecosystem just designed, to experiment and observe the consequences of entities' reactions.

In our proposal, we will use a tangible surface as a technological platform facilitating the previous requirements. These surfaces are horizontal display interfaces enabled with both multi-touch and tangible input. This kind of interfaces will allow us to address more easily the provision of collaborative discussion, co-location of participants and sharing of virtual and physical elements in the interaction.

The overall objective of this work is two-fold. On the one hand, a technological dimension related to the development of an infrastructure based on tangible surface interfaces for the cooperative creation of games, considering these as interactive and reactive ecosystems, as an expression for creative learning. On the other hand, a more educational dimension related to Human-Computer Interaction issues focused on exploring and obtaining empirical evidence that creativity on teenagers may be positively impacted by the technology being developed, and therefore to be considered in future learning settings based on the ideas presented in this dissertation.

The previously described dimensions result in an ambitious general goal of developing and evaluating a complete platform to support creative learning based on ecosystem creation and simulation on interactive surfaces. However in this thesis we will reduce the scope of this general approach and address the following specific objectives: 1) establish basic background for the future development of such a learning environment; 2) development of a software infrastructure to build user interfaces for tangible surfaces with co-located users; 3)

definition and development of a framework to support 2D reactive ecosystem edition and simulation; 4) Development of a simulator that allows to load, run, display and interact with ecosystems; 5) Build a basic creativity model to evaluate constructive tasks on surfaces similar to the processes involved in the construction of ecosystems; 6) Explore the process of building parts of ecosystems with teenagers in terms of creativity and collaborative human-computer interaction.

1.7 Research Methodology

The humankind has developed over the centuries by means of research. Research has provided us with answers about a wide variety of natural phenomena, increasing our knowledge about our environment, about the world, and introducing relevant artifacts and inventions improving our lives. Although it concerns about the search for answers, theories, or solutions to original problems, what really characterizes research is how the search is performed. In fact, the definitions that describe the research process all agree and emphasize that the search is an *organized* and *systematic* way of finding answers to questions. However, how research methods are properly involved in a specific research, to achieve such an organized and systematic search, depends on the field, questions and the phenomena under study.

In our case, this dissertation attempts to provide a solution based on tabletops to support the creation and simulation of ecosystems with creative learning purposes. To reach this goal, a synthetic view rather than an analytic one must be provided, since the issue is about making and inventing new or improved artifacts. This makes it suitable to follow the *design science research* methodology, which is defined as the design and validation of solutions proposals to practical problems (Hevner, 2004). In particular, we adopted the methodological proposal described by (Wieringa, 2009), which considers design science as a nested problem solving approach. Thus, the research can be structured into a nested set of regulative cycles which solve all the inner questions raised in the course, and eventually providing the answer to the original practical problem. The regulative cycle (van Strien, 1997) follows a structure consisting of several stages, including practical problem investigation; solution design; solution validation; implementation; and evaluation. The idea is that at every stage, if the knowledge or artifacts needed to address it are missing, it could raise other inner problems. In this way, the original problem would be decomposed into knowledge sub-problems and practical sub-problems, which should be addressed by means of new cycles.

The application of this research methodology has resulted in a set of key questions and sub-problems addressed that are summarized in Figure 2. On the right side of the chart, there are the deliverables related to the main questions that actually required either to design something or to make some research to obtain more knowledge about the designed artifacts. These have allowed fulfilling the specific objectives described above. Those parts that required a user evaluation, by means of experiments or user studies, rather than just an engineering validation of the software or system components have been emphasized by means of a strong borderline.

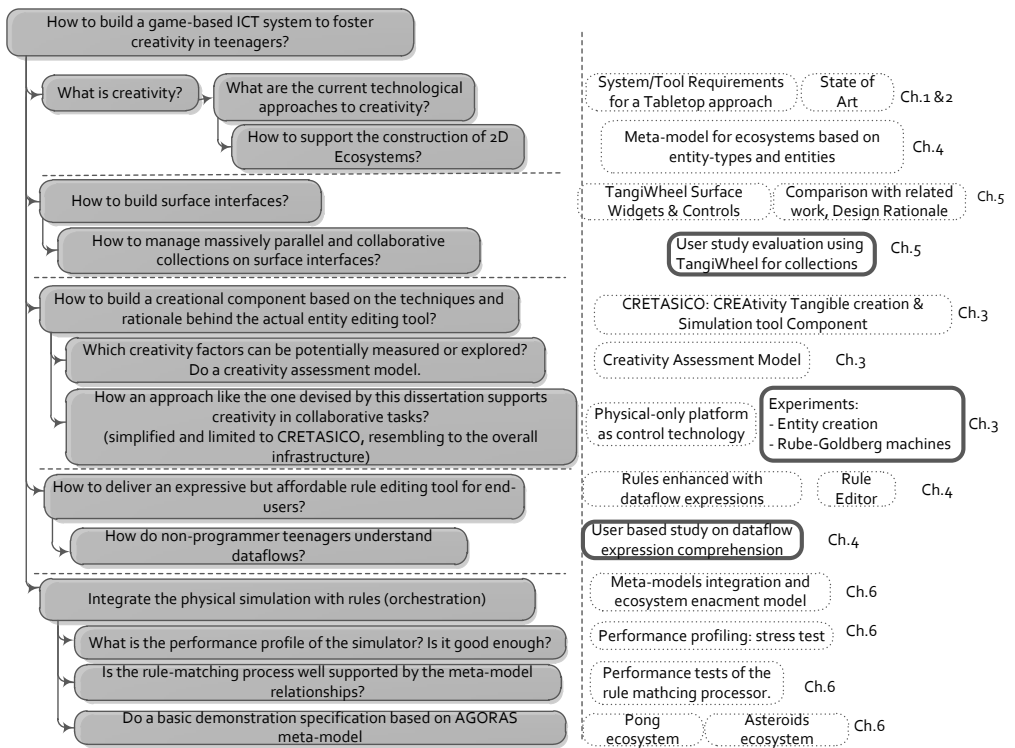


Figure 2. Problem decomposition and produced deliverables: designed artifacts and user evaluations.

1.8 Research Hypothesis

In the context of what has been told until now, we could think of formulating a hypothesis exclusively focused on technological aspects which would state on

the possibility to build a learning environment supporting the technical requirements by using interactive surfaces. However, as the background problem is more about how ICTs³ are being used in learning environments, and the need of orienting them towards creative learning, the hypothesis to be raised must be as follows:

“By using an infrastructure based on a surface interface and an environment for the creation and simulation of ecosystems, more effective cognitive collective processes can be triggered in terms of creativity and other traits relevant to collaborative interaction”.

1.9 Outline of the Thesis

The relevant questions that have been formulated and addressed were presented on purpose in that way with the intention of facilitating the logical relationship between chapters. It is therefore encouraged to take a glance at the Figure 2 as a reference of which content is presented in each chapter. The remainder of this thesis is structured as follows:

Chapter 2: State of the Art. This chapter describes related work concerning technology for some sort of learning support by means of some kind of ecosystem construction or creation of artifacts.

Chapter 3: A Model Supporting Physically-Based 2D Ecosystems to Foster Creativity. This chapter describes the model and the rationale of the main concepts to be involved in AGORAS ecosystems. With the aim of exploring creativity, this chapter also describes a basic prototype focused on the construction of structures, which follows the rationale and interaction techniques that would be expected in a final prototype of the AGORAS environment. It has successfully been used in several experiments on creativity with teenagers confronting a physical-only platform.

Chapter 4: A Model of Reactive Entities Based on DataFlow Enhanced Rules. This chapter presents the rule model enhanced with dataflows that has been adopted to represent the behavior of the entities. A user study on comprehension of dataflow expression is presented, and the rule editing tool prototype implemented is described.

³ ICT stands for Information and Communications Technology.

Chapter 5: Surface Interaction Toolkit Support. This chapter describes the interaction toolkit designed and implemented in the context of this thesis to support the construction of user interfaces for surfaces. This chapter also describes the experiments conducted on using collections implemented with Tangiwheel widgets since one of the most relevant interaction requirements requested in the construction of editors for AGORAS consists of exploring collections.

Chapter 6: Simulation Middleware. This chapter provides the description of the simulator and middleware details to put the AGORAS model to work. Some tests are presented on the rule matching processor and the ability of the middleware to manage the physically-based simulation. Finally, a sample specification of two vintage video-games like Pong and Asteroids are presented

Chapter 7: Conclusions. This chapter summarizes the contribution of the present dissertation, and discusses the most immediate future work.

Chapter 2

State of the Art

In the previous chapter we outlined the content of this dissertation and discussed that different technological aspects have had to be addressed. Some of these parts can even be understood as independent from each other, and therefore can be studied separately. Hence, in order to avoid confusion in the introduction of concepts and related work, the present section is exclusively focused on works related to the overall objective of this thesis, which is exploring technology for some sort of learning support by means of some kind of ecosystem construction or creation of artifacts. In those chapters that deal with specific objectives and report empirical studies, a specific section of related work will be included then.

2.1 Introduction

There is no single term or topic that can easily be used to describe the overall aim of this dissertation. In fact, the general goal leading this work towards creative learning-support based on the creation and simulation of 2D video-game ecosystems can be described by a combination of topics: digital tabletops with tangible input, 2D video games, creative learning, storytelling, computational thinking, etc. The more distant topics are involved, the lesser possibility to find a high coincident set of related works there is. The disparity of the topics covering this work gives an idea on how difficult it can be to find related works covering the aspects treated here. For this reason, the related works section presented in this chapter will present and discuss the range of works that are of interest to us for several reasons, although normally with a ground idea related to either creation of entities, behavior specification, play, simulation of ecosystems, storytelling, or even any combination of all of them. The differences among those works can be to some extent significant. In fact the section is going to briefly review several systems with different technological ap-

proaches to support in one way or another development of computational thinking skills or learning programming languages by children. There are some of them that, beyond the creation of programs, also support the creation of artifacts to be integrated in some sort of world or ecosystem. Others, however, allow the creation of figures or characters but storytelling is not actually programmed but based on human performance. The next subsection will show all this in a more structured way, and at the end of this chapter a comparison table will be presented to clarify the differences among all these related works.

2.2 Related Work

This section reviews a collection of selected works which, to some extent, are interesting for the background of this dissertation. For convenience, these are grouped in three sub-sections as follows. The first one deals with those proposals that have been focused on supporting performances of pre-established characters or entities in a world ecosystem in a broad sense, by either programming or just performing with them to tell a story. Hence, the common feature for the works in this group is that users cannot create the characters or entities themselves but these are already pre-existing and then users are allowed to perform with them or specify their behavior by encoding a program to tell a story.

The second subsection is about those proposals that have considered the creation of characters/entities as a central task. Once characters are created, most of the works in this subsection rely on a computational model to specify the behavior of these entities to build the simulation or the story.

Finally, the last subsection describes those works presenting advanced platforms that allow the creation of game-based ecosystems. The proposals in this category are complex, presenting typically one or more sample applications of the middleware and targeted at professionals with high skills in programming.

2.2.1 Supporting Programming and Performing in a Pre-established World

Since the programming language Logo and Graphics Turtle (*Papert, 1985*) was devised as a way to show and teach computational concepts, many proposals have been inspired in a subset of the features of Logo, many different technologies have been used to lowering even more the cognitive effort (*Kelleher, 2005*), and have taken advantage of more appropriated interaction metaphors or any other human factor such as social and collaboration skills. Two good examples

are the following proposals based on the idea of using Logo in non-conventional ways to support computational thinking and learning.

The work by Suzuki and Kato describes *AlgoBlock* (*Suzuki, 1995*), which is an educational tool where users use physical block-like pieces that can be arranged all together to program the movement of a submarine within a labyrinth. Each physical block represents an instruction for the submarine (e.g. go forward, turn left, turn right, etc.). The result of the program execution is shown on a CRT monitor by means of an animated submarine moving on a map. The system is primarily aimed at programming languages learning by K-12 students. Moreover, it allows students to improve their skills in problem-solving by means of some sort of collaborative programming tasks. By working with tangible tools, which can be shared in a collaborative workspace, *AlgoBlock* provides physical interaction and collaboration.

The second example is the work by Cockburn and Bryant (*Cockburn, 1998*). *Cleogo* is a programming environment for groups based on the Logo programming language. It allows several users to collaborate in real time in the development of programs and check their execution. The users work with different personal computers that are interconnected through a network. *Cleogo* uses a graphical user interface based on WIMP to program the movement of the turtle in Logo. The aim is to encourage children to solve problems collaboratively. Each user has a screen equipped with a keyboard and mouse, but the input controllers provide access to all the functionality of a shared graphical interface displayed in the different screens. Although the system is limited in terms of numbers of users, a realistic limit is about four, to avoid degradation in system response and therefore user collaboration. Users can stay in the same room or in distant locations. In the latter, an audio channel of communication is needed to support voice interaction. The system does not provide any policy to avoid contradictory or conflicting actions by different users. This is left to social protocols rather than software mechanisms. Related to this, one aspect to be considered is the awareness of the actions among users. *Telepointers* are used to facilitate awareness. They are pointer representations in the screen that shadow the pointers of the other users. They play an important role in common communicational expressions such as “this” or “put it here”, which normally requires gestural expressions to clarify the context of the statement.

Cleogo provides three different views following different programming paradigms. Users can use any of them as they prefer at any time. One is based on programming by demonstration following direct manipulation of the turtle. Another is an iconic language in which programs consist of a chain of instruc-

tions. The third is a textual language. Each language fits better for developing different user programming skills. The three views are kept consistently along the interaction.

Both previous works are relevant because one provides a tangible interface for the language and the other provides a multi-user networked approach. Although they support programming (i.e. creation of programs), the virtual objects and the world in the simulation are completely pre-established.

A third outstanding example using Logo is the work by Gallardo *et. al* (Gallardo, 2008). TurTan is a tangible programming language that uses a tangible surface interface in which tangible pieces represent virtual instructions of programs. As already mentioned, this is inspired by Logo, and therefore, it was designed to generate geometries with a turtle. As in the original Logo language, one of the design objectives of TurTan is learning programming concepts. TurTan starts with a black screen with the image of a little turtle in the middle. When a tangible (i.e. instruction) is put down on the surface, a visual response is provided and the instruction is executed, applying the result on the turtle. The programs consist of a sequence of tangible instructions that have been put down over the time. The parameters of instructions can be set by rotating the tangibles. Touch input is integrated seamlessly with the use of tangibles for the real-time visualization of the program results as users collaborate in the program construction. The work does not report on user evaluation on either learning or creativity, although the authors do mention the necessity to explore these dimensions as the system is oriented towards young children.

Another relevant work is the exploration on tangibles carried out by Fernaeus and Tholander in (Fernaeus, 2006). They discussed a system that allows children the creation, edition and simulation of a 2D virtual environment in a collaborative way. The TangibleSpaces system consists of a large carpet with an array, a set of plastic cards with RFID tags representing entities and operations, and a screen which shows the state of the system under construction. The cards comprise a compositional tangible mechanism as input to the system to express entity behavior. Specialized card operators for the creation of entities are available. They allow children to collectively create the virtual world by inserting the creational card along with a card representing an entity in the physical array. Additional behaviors or property changes can then be performed by stacking other cards on the entity. These compositional constructions in the physical world have a representation in the virtual simulation displayed on the screen.

Several issues that authors addressed are related with this duality representation in the physical setting and the virtual representation where the simulation is eventually carried out. These worlds could be seen from a mirror metaphor, so that actions in one world would affect to the other in both ways. However, the carpet remains more like an input method since changes in the virtual simulation are not easily transferred to the physical setting (e.g. entity cards cannot be autonomously moved as a consequence of the evolution in the virtual simulation). This disparity is not intended as a limitation by the authors but they try to complement each representation as much as possible. For instance, it is possible to create a forest as a bunch of trees by using only a tree card along with a modifier card, instead of using many tree cards. In this way the physical representation can be simple, suitable and advantageous to several situations.

In contrast to the work by Fernaeus and Tholander, which was focused on tangible interaction to make up a 2D interactive world, the work by Horn and Jacob (*Horn, 2007*) is more focused on designing tangible programming languages to specify the behavior of robotic ecosystems. Two tangible programming languages were presented, Quetzal and Tern. They use physical objects (plastic or wooden pieces) with no electronic device but with visual tags. Each piece represents a specific instruction and they can be connected to each other to build a program. These languages were designed to teach basic programming to children in a classroom setting in primary and secondary school. The main advantages are that they are made of durable low-cost components, with no-connection required and fostering collaboration among children.

Both languages are compiled using a portable scanning station by using computer vision techniques, so that the program is captured and translated to intermediate languages that finally are compiled to code for the targeted platforms. In the case of Quetzal, LEGO Mindstorms robots can be controlled, whereas virtual robots moving in a 2D virtual world are involved in the case of Tern. In case of syntax errors, the systems show the image of the program along with an arrow pointing towards the problematic piece. The robots can interact in the same world, and several student groups can collaborate to solve problems to pick up objects or navigate through a labyrinth. The teacher can easily add a projector to show the world array in a projection screen, allowing students to participate in a shared activity.

The work by Leitner et. al represents a great effort on joining tangible and interactive digital interfaces to foster creativity (*Leitner, 2008*)(*Leitner, 2009*). In-

creTable is a game based on *The Incredible Machine*⁴ that uses an advanced high technology setting focusing on mixed reality tabletops. The system is composed of an interactive tabletop equipped with digital pens, robots and a depth camera that allows advanced interaction with actual 3D objects. The game consists of several puzzle exercises that require the construction of Rube-Goldberg machines (*Rube-Goldberg, 2012*), involving virtual as well as actual domino pieces and other physical objects such as portals or robots. Levels are supposed to encourage user creativity to solve the puzzle in a complex way. Although the system allows users to freely create a specific arrangement of virtual and tangible elements to achieve the goal, these elements and the levels themselves are completely pre-established. A subsequent evaluation of the platform has explored the relationship of certain interaction aspects with this advanced technology with flow.

Finally, Kelleher and Pausch presented StoryTelling Alice (*Kelleher, 2007*). It is a programming environment aimed at girl teenagers (11 through 15 years old) to encourage them to learn programming skills. This goal is motivated given the low rate of female students enrolled in computer science courses in the United States of America. This system allows novel programmers to create programs that control the movement of objects in a virtual 3D world. The girls can tell their stories in the virtual environment by means of programs, and so they require programming scenarios as well as the character behaviors to appear in the animations. The 3D world has a list of objects, with properties and methods, and functions, which the users select from a gallery of objects. A set of pre-established animations (e.g. move, turn, resize...) can be applied to all these objects. Users can code their own procedures to specify behavior by selecting, dragging and dropping the methods and the objects in the parameter gaps accordingly in a WIMP based interface. This way Alice facilitates the construction of programs free of syntax errors by simply *drag&drop* interaction techniques and, moreover, the tool provides a pre-visualization mode so that users can see the resulting animation in advance to check whether the instructions they encoded do what programmers wanted.

In this context, their work in (*Kelleher, 2007b*) reported a user-based study. It was carried out to compare learning, behavior, and the attitude of girls who start programming with Alice. A total of 88 girls, 12.6 years old on average, were involved in workshops of four hours long. Forty-five used the generic version of Alice as a control group, and forty-three used the StoryTelling Alice.

⁴ http://en.wikipedia.org/wiki/The_Incredible_Machine_%28series%29

The task consisted in completing a tutorial of the software and creating a program within 2 hours and fifteen minutes with the version of Alice they were assigned. After that, they tried the other version of Alice for 30 minutes. From the programs produced and the answers to questionnaires, the study concluded that girls were equally entertained and were successful in learning programming concepts using both versions of Alice. However, the girls using StoryTelling Alice showed more engagement with programming, spending more time with the software; they were more likely to use the software during some extra time, and showed a higher interest in using it in the future.

2.2.2 Creating Characters for an Ecosystem World

If the works in the previous subsection were primarily characterized by the use of pre-established entities to carry out a simulation or a performance in a variety of ways, the works in this subsection are mainly distinguished as they also give relevance to the creation of the entities.

LogoBlocks is a graphical programming language to support programming for the LEGO *programmable brick* (Begel, 1996). The brick is a small computer that can be embedded and used in LEGO creations by reading from sensors and controlling engine activations. LogoBlocks is a graphical alternative to the former language BrickLogo. Instead of writing a program in text, users can now put several graphical blocks in the workspace so that they represent instructions of a program. Syntax errors are avoided by representing every instruction in the program with a block, and providing visual cues such as specific shapes matching with other blocks, and easily supporting block property exploration by means of double clicking techniques.

Although visual programming as in LogoBlocks is advantageous for beginners, since it allows them to avoid syntax issues and see the program at a glance, several drawbacks are already mentioned in this work. For instance, advanced programmers can feel frustration since the textual language could represent more concisely some basic statements or common behaviors. In addition, the number of visual primitives present in the screen is usually more limited since icons and graphical representations require more space. Another problem is the difficulty to support extensibility of the languages, which usually are designed as sealed domain specific languages.

LogoBlocks follows a drag&drop metaphor in a WIMP user interface. A palette on the left of the screen contains the different blocks available. They have different shapes and colors for the available block categories: the action blocks allow controlling engines and perform “wait” and “repeat” operations; the

sensor blocks obtain information from the real world; variable blocks represent variables in the program that can be connected to other blocks requiring numbers; and procedural blocks, which provide an abstraction mechanism for the implementation of procedures. Although LogoBlocks itself is simply the graphical programming language, it is targeted at the programmable brick and therefore the constructions of the robots are creations that can be also used as part of an ecosystem.

Maloney et. al present Scratch (*Maloney, 2004*). It is a graphical programming environment that allows children to program interactive stories, games, animations and simulations. All these based on 2D stages composed of a background and a set of sprite-based objects. The language used to specify behavior is based on LogoBlocks, so that users build programs by just dragging and dropping blocks representing instructions that match in shape to each other avoiding syntax errors. The main screen of the tool has a panel where the stage is shown, allowing program debugging, and testing new ideas increasingly and iteratively. Although Scratch is a mono-user application based on WIMP interaction, there exists a web-based online community supporting the Resnick's Spiral (*Resnick, 2007*), which aims to foster discussion and creativity between users, relying on collaboration, mutual discussion and distributed contribution. However, all this has to be done outside Scratch itself. In (*Aragon, 2009*), an empirical study is conducted, which explores the use of communications in distributed users using Scratch for effective collaboration in creative work. The authors concluded that socio-emotional communication is important for successful creative work and emphasized that systems supporting social creativity must facilitate sharing and play.

Another relevant work with the idea of creating 2D entity-based virtual ecosystems in full is the one by Repenning (*Repenning, 2000*). AgentSheets is a tool based on agents that allows users to create simulations and 2D interactive games. These creations can be published as Java applets on the web. Users can create simulations of sprite-based agents in a 2D world based on a rectangular array. A cell in the array representing the world can contain any number of agents stacked which can directly be manipulated. The users are responsible for designing the visual aspect of agents by drawing icons. The behavior of agents is based on event-based rules. The rule editor follows a rewriting rule paradigm. The user expresses the conditions of the rule by selecting the visual state and the visual icon representing the event. The action to be performed is typically expressed as a post-condition, showing how the situation expressed in the pre-condition must be changed.

An innovative tangible approach is the one presented by Raffle, Parkes and Ishii in (*Raffle, 2004*)(*Parkes, 2008*). Topobo is a 3D constructive assembly system that allows the creation of biomorphic forms like animals and skeletons. This is achieved by means of pieces embedded with kinetic memory, and the ability to record and playback physical motion. Topobo is designed to be a user interface that encourages creativity, discovery and learning through active experimentation with the system. Studies with children and early adolescents are reported. In the case of teenagers, the study explores how the system supports design, concluding that Topobo can help students to learn about several educational concepts on physics such as balance, center of mass, coordination and relative motion. Later, Topobo has been more extensively used in a range of different contexts for further evaluation. For example, it has been used in an extramural course for teenager students for three months for free activities; with children and young teenagers for 8 months targeted at object-oriented tasks in the context of sciences; or even in architecture courses with adult students, using Topobo for the design of their final project. In all these trials, the teachers considered Topobo as a useful or interesting tool, although they stated that training with the system is needed to be confident with it and show reliability in teaching.

Lu et. al present ShadowStory (*Lu, 2011*), a storytelling system inspired in Chinese traditional shadow puppetry. Children use a Tablet PC (a laptop with touch input) to create digital animated characters and other accessories or props, and then they are allowed to perform stories on a back-illuminated screen, controlling the characters with simple movements by means of orientation handheld sensors. Thus, ShadowStory includes two interaction modes. In the “design” mode, the elements for the story are created whereas in the “performance” mode the story is told to the public like in the traditional Chinese puppetries.

In the design phase, children use a TabletPC and its pen-based input to create three types of elements for the story: characters, props and backdrops. To create a character, the system provides an articulated template consisting of head, chest, arms and legs. These parts can be created individually using the knife and brush tools to cut and paint the parts. In addition, props or non-articulated accessories to be used as well as the curtains can be created similarly with a range of digital tools. Besides creating all these elements, it is possible to pick pre-defined elements from an existing library.

Once all the elements have been designed, the story is almost ready to be performed by the children. First, the stage should be arranged according to the

story along with the participation of the characters. Automatically, each character added to the stage enables a pair of wireless 3D orientation handheld sensors. Once all this is done, the performance can be activated and children can tell the story projecting the characters on a wall screen.

2.2.3 Advanced Platforms to Create Game based Ecosystems by Programmers

The main characteristic of the works in this subsection is that their ecosystems are prepared by developers or programmers, and only then consumed by the players in game-based activities.

The Teaching Table (*Khandelwal, 2007*) is an interactive touch-enabled tabletop developed with the aim of teaching basic mathematics to children 3 to 5 years old. The child has to put tagged physical objects to solve the problems (e.g. blocks with geometric form, numbered blocks, etc.). The system is able to provide audio feedback about the success or failure of the activities. The system was designed to be integrated in the classroom, providing learning activities for children, and evaluation tools for the teachers in the kindergarten. However, it is mono-user and hence not collaborative. The specific learning activity can be controlled from a PC, which allows monitoring the progress of the child. Several skills and learning abilities are developed by carrying out different activities. For example, to develop comprehension on numbers, the devised activity consists of putting numbers 1 to 10 in order on the surface. This activity is useful to improve skills such as counting with memory, number identification, etc. Another sample activity would be the creation and replication of simple patterns, according to colors or shapes of the displayed information. Children would improve skills such as completion with the blocks of partial patterns, replicating already existing sample patterns, identifying shapes, etc.

In the end, this is a flexible system that supports activities already available in separate commercial kindergarten toys, but with the advantage of providing more control to the teacher to monitor the progress. Therefore, its power lies in its generic nature, which allows activities for the developing of a variety of skills. Although the current research has been focused on mathematics, it seems to be feasible to cover other topics such as languages, literature, science and creativity.

In (*Lampe, 2007*), “The Augmented Knight’s Castle” is described. It is an environment of augmented toys including buildings and Playmobil® figures from the Middle Ages that enriches gameplay with background music, sound effects, verbal comments and different visual responses. In the game, smart toys can

be used, which are equipped with sensors and RFID technology to improve the gameplay and increase children interaction. This augmentation can provide richer learning experiences, facilitating the development of children's social skills.

From a technological point of view, a sensor network and a set of RFID devices are embedded in the toys. All these elements send data to a computer in the background, which holds the rules that coordinate the activation of visual (LEDs) and audio responses of the toys. Whereas fixed RFID antennas are located in the building and the ground to detect proximity, movable antennas are also put inside movable toys like carriages. The other toys, which are typically grasped by children and play a character role, are even embedded with several RFID tags to facilitate finer orientation and distinguish between the back and the front sides if necessary. It is also possible to use specific mobile devices to interact with toys, trigger specific responses and obtain multimedia information. In this way, toys and mobile devices can touch each other to progress in the plot of the storytelling and even produce "fights" between figures. The proximity sensors allow interaction based on the metaphor of "point and touch" to trigger responses, making the game play more attractive. The microphones enable the detection of high noisy levels usually related to high children excitement; accelerometers can be used to recognize gestures such as shaking a magic bottle, or detect circles of a magic wand, etc. In response to all these different events, the game play can be adapted according to specific pre-programmed responses. For instance, a LED in the magic bottle could be turned on to indicate that the magic potion is ready, or force feedback could be supplied once the circle gesture on the magic wand is detected, etc. The integration of devices with toys is seamlessly, so that children are apparently acting traditional toys but with improved capabilities. The logic behavior of the toy ecosystem is implemented by state machines. These are encoded by programmers and they allow the modeling and specification of processes in terms of conditions that trigger transitions between states performing actions on arrival or during the stay in a state. Hence, the plot for the game emerges from the collection of all state machines.

Another platform is the TTVViews (*Mazalek, 2008*) presenting a Role-Playing Game (RPG) based on an implementation of "Dungeons and Dragons" on a touch-enabled tabletop. In the game, players adopt the role of a character created by them and play together to tell a story within a set of established rules. As it occurs in traditional role games, the game-master guides the activity, helping to conduct and develop the plot, and controlling the progress of the game play. This plot construction is improvised as participants take decisions and

perform actions within the game. To coordinate the development of the plot, the game-master has an additional computer available besides the TTViews table. This is the GameMaster Interface, which provides statistics on the game and the possibility to handle the behavior of non-player characters or even props.

This game was implemented to support three players, who control the tangibles representing their characters (e.g. character pawns for fighter, wizard or rogue). The pawn can be moved to several places in the map to trigger the desired actions. The menus can be manipulated in turns by means of other tangibles, in such a way that these physical items manage the control of the game.

Magerkurth et al. present STARS (*Magerkurth, 2003*), which is a platform to develop digitally augmented tabletop games, integrating mobile devices with an interactive tabletop. The aim of this platform is to augment traditional tabletop games with computing functions but without dropping the human-centric dynamics from traditional games. The range of games that can be developed on top of STARS includes tabletop games that use cards and chips such as Monopoly, or others such as role or strategy games. However, this computational approach relieves players from routines (e.g. counting), provides rules and constraints commitment, and delivers complex simulations. To support this, the main technological elements available are: the digital tabletop with audio and video augmentation, with touch enabled technology and tracking of tangible pucks and chips (*Streitz, 2001*); PDAs, providing each player with personal information and a private space of interaction to control the game or send messages to the other players.

However STARS is not a platform for end-users to create games but it actually consists of a set of hardware and software layers offered as APIs to develop games on top. This is broadly focused on a game engine supporting different specification languages for the description of objects, players and rules for the gameplay. In this way, enriched tabletop games can be developed more easily, with less effort and cost, although all the functionality not offered by the platform has to be implemented from scratch anyway.

Finally, the work by Smith and Graham presents Raptor (*Smith, 2010*). It is a tool that allows users to sketch videogames (*Smith, 2009*). Designers can create and experience much more quickly their ideas for future videogames, and explore whether the game will be entertaining and therefore worthy to be developed. In this context of early design, sketches can be really useful given the high cost of developing videogames in industry. The system can be used to

generate and experience videogame sketches in a range of game categories: role, shooters, racing, etc. The designers use a tabletop interface to collaboratively create the world and the conditions of the videogame prototype, offering a 2D aerial view in which they can add, remove or change characters, props or even terrain by means of natural touch gestures and specific tangibles. The testers play the generated videogame on a PC with 3D graphics, playing with a regular console controller. The changes produced by the designer will be transferred to the game being played in such a way that new situations can be explored, obtaining comments directly from the testers. During the testing process, the interaction of the tester with the characters is partially based on *Wizard-of-Oz* techniques. This means that the behavior of the characters, props or the world itself must not necessarily be fully implemented, and the responses are given by actions performed by the designer on the tabletop in “real time”. This is useful to test new ideas when the behavior is not completely defined yet.

2.3 Feature Based Comparison

With the aim of providing a view of the variety of related works briefly described here, several features are presented next. We have considered this set of features relevant because they can be useful to compare proposals with regard to several characteristics related to the ideas and requirements associated to the work in this thesis.

Firstly, we consider some general features. The “Primary aim” feature indicates the primary function and aim of the proposal or system. For instance, a system could be created to support the learning of computational concepts or for social learning purposes, or simply for entertainment, etc. To simplify the classification, avoiding unnecessary complexity, only three categories have been considered: Learning purposes (L), Entertainment (E), and Prototype Sketching (S).

The “Target users” feature refers to the users that the system is aimed at. The “Study” feature indicates whether the proposal reported some kind of user experience, or any user-based evaluation, study or experiment. In addition to developing a proposal according to cognitive and/or social theories, it is highly interesting to evaluate them and validate that the main assumptions are achieved by the built prototype.

“Social interaction” indicates how users interact with each other within the system. Typically, systems supporting some kind of social interaction achieve this by putting users in a co-located setting or in networked different places.

Co-location allows for face-to-face communication whereas networking allows chatting or audio-video communication. Social interaction could be focused on competition or collaboration. As most of the proposals are about learning purposes, they normally focus on collaboration processes. A system can be used alone, not supporting any social interaction; although by putting users to discuss ideas in front of a shared single computer could provide some sort of social interaction anyway.

Additionally, we have considered some attributes that describe how the simulation or performance is carried out. Firstly, the “Ecosystem Type” indicates the type of ecosystem involved in the proposal. Normally, the ecosystems consist of a set of entities represented in a range of ways across the systems. It has been shown in the description of the related work how they can be represented by 2D single sprites, 2D virtual complex shapes, single or complex tangibles, robots, or 3D digital complex objects, etc.

Secondly, the “Behavior specification” informs about the inherent model, computational or not, behind the performance or simulation. Thirdly, “Tech. Support” in the simulation/performance group reports on the ground technological components being used by the system.

Another group of features describes the authorship facilities that the proposals offer. In the case that authorship tools are missing, the proposal would be more oriented to the consumption of contents although some sort of programming is still present. This is an important aspect under the perspective of Abt (*Abt, 1970*), since the special relevant task is more about playing to create the artifact rather than consuming the game. Of course, consumption of pre-established contents is useful as a means to convey knowledge and skills. Moreover, the existence of authorship tools can suit to a wider range of activities as reclaimed by McFarlane et al (*McFarlane, 2002*) and Gros (*Gros, 2007*) in their studies on pre-defined software.

The “World construction” feature simply indicates whether the system allows users the construction of a world ecosystem or not. This means that at least some pre-established components or entities can be arranged arbitrarily to make up a world. Similarly, the “Entity/Component construction” feature indicates whether an editing tool is provided to create the entities or components to populate the world ecosystem. The “Entity Creation Tech. Support” feature reports on the ground technological components being used by the system to support this. Finally the “Behavior construction” and “Behavior Tech. Support” features are similar to the previous ones but focused on the behavior specification by users.

		Simulation / Performance					Authorship				
Work	Aim	Target users	Study Social Interaction	Ecosystem Type	Behavior spec.	Technological support	World construction	Entity/Component	Entity tech. support	Behavior con-support	Structure Behavior Tech. Support
AlgoBlock	L	children(12)	y co-located	2D virtual	procedure	Tangibles + PC	n	n	-	y	Tangible blocks
Cleogo	L	children	n network	2D virtual - turtle	procedure	PC	n	n	-	y	textual code / drag&drop
TurTan	L		n co-located	2d virtual turtle	procedure	Tabletop	n	n	-	y	Tangibles
TangibleSpaces	L	children (6-12)	y co-located	2D virtual/tangible entities	procedure	projection screen + tangible cards and carpet	y	n	-	y	Tangible cards
Incretable	E	-	y co-located	Tangible and virtual parts	Physics simulation	Mixed reality (tabletop + simulation robots)	+n	n	-	y	tangibles
Quetzal/Tern	L	First/Second grade children	y co-located	virtual/actual entity robots	procedure + physics constraints	simulation robots	y	n	-	y	tangible
StoryTelling Alice	L	Novice programmers /girls	y -	3D virtual	procedure	PC	y	n	-	y	drag&drop metaphor in PC

SUPPORTING PROGRAMMING AND PERFORMING IN A PRE-ESTABLISHED WORLD

Table 1. Related work comparison: Programming and performing in a pre-established world.

		Simulation / Performance		Authorship								
Work	Aim	Target users	Study	Social Interaction	Ecosystem Type	Behavior spec.	Technological support	World construction	Entity/Component construction	Entity tech. support	Behavior construction	Behavior Tech. Support
Logoblocks	L	Lego brick programmers	n co-located	Tangible Robots	procedure (virtual blocks)	Robots	y	y	PC	y	Drag&drop (in PC)	
Scratch	L	children and teenagers	y -	2d virtual entities	procedure (virtual blocks)	PC	y	y	PC	y	drag & drop (in PC)	
Agentsheets	L	any	n -	2d virtual entities	Rules	PC	y	y	PC	y	drag & drop in GUI	
Topobo	L	children	y -	3D tangible entities	procedure	Tangible Robots	y	y	tangible-manual	y	kinetic robotic memory	
ShadowStory	L	y	y co-located	2d virtual performance orienteering sensors	PC-proj. + screen	y	y	tangible-manual	n	-		

CREATION OF CHARACTERS FOR AN ECO-SYSTEM WORLD

Table 2. Related work comparison: Creation of characters for an ecosystem world.

Work	Aim	Target users	Study	Social Interaction	Ecosystem Type	Behavior spec.	Simulation / Performance			Authorship		
							Technological support	World construction	Entity/Component construction	Entity tech. support	behavior construction	Behavior Tech. Support
TeachingTable	L	teachers for children (3-5)	n	co-located	-	procedures	tabletop	n	-	-	-	-
Augmented Knight	E	children	n	co-located	real toy figures	state-machine	WSNs	y/n	-	-	-	-
TTVRPG	E	players since 14	n	co-located	2d virtual	pre-established procedure	tabletop	n	-	-	-	-
STARS	E	Game developers for players	n	co-located	-	-	tabletop + n PDA	n	-	-	-	-
Raptor	S	Game designers and developers for players	n	co-located	3d virtual worlds	procedures	Tabletop+ y PC	y	-	-	-	tabletop

ADVANCED PLATFORMS TO CREATE GAME,-BASED ECOSYSTEMS

Table 3. Related work comparison: Advanced Platforms to create game-based ecosystems.

					Simulation / Performance			Authorship				
	Aim	Target users	Study	Social Interaction	Ecosystem Type	Behavior spec.	Technological support	World construction	Entity/Component construction	Entity tech. support	Behavior construction	Behavior Tech. Support
AGORAS	L	teenagers	y	co-located	2D virtual entities	physics + rules	tabletop	y	y		y	tabletop: dataflows in rules

Table 4. AGORAS proposal within the related work comparison.

Table 1, Table 2, and Table 3 show how there are a range of different technologies being used to support the simulation of world ecosystems or the performance by users based on games but mainly with learning purposes. Clearly there are two groups of proposals. Those that support the creation of the main elements or entities to be involved in the world ecosystem and those do not. Most systems support some sort of behavior specification given by the end-user in terms of instructions or programs. The programming tools are mostly based on *drag&drop* metaphors using WIMP interaction to facilitate the construction of programs by non-programmers or children. There is also a third group of work, but more focused on middleware and new technology capabilities rather than targeted at non-programmers or children.

Within this feature based comparison shown in the previous tables, the proposal of this dissertation, AGORAS, would be framed within the second group since it is related to the capability to create the entities participating in the world ecosystem. Table 4 illustrates the values for the features under consideration for our proposal. AGORAS would be targeted at teenagers working on the creation of entities and scenarios for learning purposes. The works in this category do not usually consider custom behavior specification in co-located collaboration. Thus, the crucial difference is such a co-location and collaborative support enabled by the tabletop platform, with all the benefits that it should introduce in a proper thinking-acting-reflecting process involving direct communication between peers and direct manipulation of virtual artifacts. In addition, another important difference is the combination of physically-based behavior with rules without need to write code or rely on *drag&drop* interfaces oriented to a single-user.

Chapter 3

A Model Supporting Physically-Based 2D Ecosystems to Foster Creativity

This chapter describes the model and the rationale of the main concepts to be involved in AGORAS ecosystems concerning stages, entity types, entities, etc. They support the specification of physically based behavior for the entities in the virtual ecosystems. A general tabletop platform like AGORAS, whose primary activity focuses on playing to create game ecosystems, is complex to design and implement. For our objective of exploring creativity, instead of trying to explore it on such a complex whole platform, some simplifications would be desirable in order to perform a more meaningful and interesting evaluation. This chapter also describes a prototype for experimental evaluation purposes intended as a tool for creating physically-based structures. It basically includes the core collaborative processes and interactions to be found in the eventual implementation of an AGORAS editor supporting the creation of physically-based entities for a given ecosystem. This prototype has successfully been used in several experiments on creativity with teenagers confronting a physical-only platform. This is important because it can provide empirical evidence on how this kind of technology and interaction techniques can foster creativity over traditional physical-only settings, and establishes the foundation of the AGORAS creation support.

3.1 Introduction

As pointed out in Chapter 1, games may be played either casually or seriously. When played casually people look for having fun or just having a good time. This has been the most common primary objective and motivation for playing games. However, the concept of Serious Games is strongly emerging in recent years. A game being played seriously is not being played primarily for amusement but rather for training, learning, education, etc. Serious games try to benefit from the power of fun provided by regular games in order to engage users in the activities that must be performed, to convey new knowledge or develop new specific skills.

Although many different learning styles or theories have been traditionally used in games in the past, it seems clear that several flavors of constructivism have over the years widely influenced game design and the subsequent social and

psychological studies. The basic idea of constructivism is “learning by doing” because individuals are required to act in order to know. This is a consequence of knowledge not being a mere copy of reality but an internalization of the idea transformed when understood. Therefore, essential aspects to reinforce and guide the learning processes are the experimental practice and the reflection on such a practice.

McFarlane et. al report on the educational use of games at school (*McFarlane, 2002*). Authors made an experimental evaluation of commercial video games at school settings according to an evaluation framework which they developed. They divided the learning type supported by games into three categories. The first category is characterized by learning as a result of tasks stimulated by the content of the games; the second one consists of knowledge developed through the content of the game, although in general content does not fit to curriculum content very well; the third one is about learning that develops skills as a result of playing the game. Some of these skills are essential to the context of the autonomous learner such as problem solving, sequencing, deductive reasoning and memorization, and others arise as a consequence of working in groups on tasks such as peer tutoring, co-operation and collaboration, co-learning, negotiating skills and group decision.

(*Gros, 2007*) also discusses the use of video games as educational material in such a way that teachers create learning environments that permit the possibility of dealing with a complex system is multidimensional, multimedia and interactive. The inclusion of the game in the classroom allows the entire group of students to work cooperatively and in discussion groups that should provide space for analysis and critical reflection of the medium itself. The author focuses on similar issues to the ones reported in (*McFarlane, 2002*), and use a four-staged methodology composed of: experimentation, reflection, activity, and discussion. In the stage of experimentation, students are asked to take notes of the decisions and results while playing. In the reflection, at the end of the sessions, the results and strategies are compared among participants. In the stage of activity, specific curricular activities are designed for the game, and in the stage of discussion the reflection on the actual process of learning and the joint discussion related to the proposed activities are performed.

The most extended conclusion is that games are effective teaching and training devices because they are highly motivating and may communicate very efficiently the concepts and facts of many subjects. However, some difficulties are also found: in general, commercial video games are used in these studies, resulting in game content hardly fitting to the curriculum content and without

any chance to be modified because software is sold as is. The gaming activity must be strictly planned to accommodate the limited available time and to focus on the part of the game that best fits for the purpose of the gaming session. Evaluated games are often for two players, and in spite of availability of online communications for the multiplayer ones, there is a lack of direct communication and so the discussion and reflection is postponed to the end of the session. Additionally, the exercises proposed are not related to the game itself but with ordinary tasks of practice with the exception that in this time they are based on the content of the game. An obvious alternative would be to design specific video games that fit to curriculum content, but it is very expensive and not feasible. Similarly, ordinary educational software could be a solution but this type of applications usually falls into a clearly instruction based design, leaving out the natural good features of game design. In our opinion, future gaming platforms supporting learning activities should provide pre-set scenarios easily editable by teachers to adapt them to the curriculum content, and a space for experimentation, discussion and reflection for both teachers and learners. Therefore, future research on serious games for learning should not be focused on producing games whose content fits to a wide range of educational topics and learning styles but instead on giving teachers and learners the tools to produce their own games.

In fact, this is not a revolutionary idea. In the late 60's, Abt already thought of considering the process of building a game as an important learning activity (*Abt, 1970*). He pointed out that the first phase of game learning, the design and preparation stage, may be divided into two kinds of activities: "a relatively passive preparation for active game, and the actual design of the game to be played". The former is more common and implies learning the background of the material to be simulated in the game, its rules, roles, concepts, etc. The latter is probably a more rewarding way of game preparation. The game designer is actually inventing a simulation model of the process to be played. In the course of doing so, the student must identify the significant variables involved, the relationships among them, and the dynamics of the interaction. To do this successfully, it requires understanding the process to be simulated. Therefore, involving the students in this process expands their knowledge, learning not just factual content but also the processes and interactions involved. A direct consequence from Abt's work is that the focus is not on what can be learned by playing games but what is learned by playing to build games, and at last playing them.

Consequently, a first step towards supporting these future learning environments is to meet, to a large extent, the pedagogical principles on games and,

accordingly, to give a tool that, on one hand, allows teachers to design basic learning scenarios according to curriculum objectives and skills in which they are interested and, on the other hand, both teachers and learners have the ability to collaboratively build common playing worlds consisting of interactive entities supporting actions and being able to exhibit reactive behavior. Because a reactive and event based conception is a general framework that people easily understand in terms of action-reaction and cause-effect relationships, such a tool would provide a suitable space for experimenting, reflecting, and discussing in both game creation and playing processes if properly used on a digital tabletop.

Hence, with these needs, this chapter proposes a conceptual model to enable a tool supporting the creation and further enactment of interactive environments. In the remaining of the chapter, we present the basic model for authoring entities and an experimental evaluation of its potential to foster creativity in collaborative construction tasks.

3.2 Model Requirements: Editing and Simulating

Our proposal is ambitious since it is not about building a creative problem-solving application in a specific domain, with a predefined behavior and pre-established reactions put at the users' disposal. If it were intended to provide pre-established entities with some sort of variability based for instance on an attribute selection technique then a model with these specific concepts and a few configuration attributes on them would be enough. However, as more complete creation and simulation processes must be supported, a model with several advanced capabilities should be proposed. In particular, the model must support the description of the concepts to be created in the authoring process, as well as the concepts representing instances since they must be instantiated and enacted by the simulation middleware.

All this means that the model must contain meta-concepts to express descriptions, and also reflection capabilities to allow the middleware to modify the definitions of the types on which entities are defined. So the model is not about the entities themselves but about the information required to describe any entity. Moreover, meta-modeling and its advanced features are also needed (*Atkinson, 2003*) to properly support the simulation process. Consequently, in the next sections, *concepts* supporting description of *concepts* are widely present.

3.3 Ecosystem Model

By adopting a reactive environment and the introduction of the design stage as another important activity besides just playing, some of the Gee's principles (Gee, 2005) are inherently present, especially those related to the categorized as "Empowered learners" and "Understanding"; however, the occurrence of other principles depends on how teachers design the activities, scenarios and tasks to be performed. The teacher, by assigning the responsibility of creating reactive entities to the students, defining their properties, actions, and rules, forces them to study the curriculum contents related to the proposed scenario in order to perform well. Moreover, in this way, the "Co-design" principle is addressed since students are also designing the interactivity and the reactions which will guide the evolution of the ecosystem. Similarly, the "Identity" principle is covered twice: students have to learn the role of the entity by designing it and by playing its role, too. Entities are seen as single components that can only be fully realized as a part of a more complex system where everything adopts a meaning, as the "System thinking" principle describes. In our case, the complex system is obviously the ecosystem, and this is, at the same time, correlated to some actual reality that the teacher has decided to involve in the scenario. In this sense, the prepared scenario is a simplified model of reality, and when playing, the consequences are simulated with low risk, supporting in this way Gee's "Fish tanks" and "Sandboxes" principles.

A reactive environment, so called ecosystem, is the place where several entities inhabit, interact with each other and perform their actions in order to achieve their objectives. These entities are first-order citizens. The primary concepts that an ecosystem defines are events, types of structures, entity types, entities, stages and rules. These are shown in the Figure 3, which depicts a UML class diagram representing this part of the model. Basically, an ecosystem is inhabited by entities that play in stages which they can be enacted according to physical attributes and event-driven rules.

An ecosystem is split up into stages (Zagal, 2008), and it has at least a stage definition. A stage is the basic piece of simulation specification that can be enacted in AGORAS and represents a scenario. At the moment, the space of stages is confined to the actual surface dimensions and has a background that decorates the stage's look conveniently. In addition, a stage specification indicates which entities participate in it from the beginning and a set of value properties for them. Notice that entities are not necessarily included in the specification of stages. The entities can exist in the ecosystem definition with-

out participating from the beginning in any stage. They can be included dynamically during the simulation of the current stage.

Rules cover the reactive behavior to be exhibited in each stage and will be presented in detail in Chapter 4.

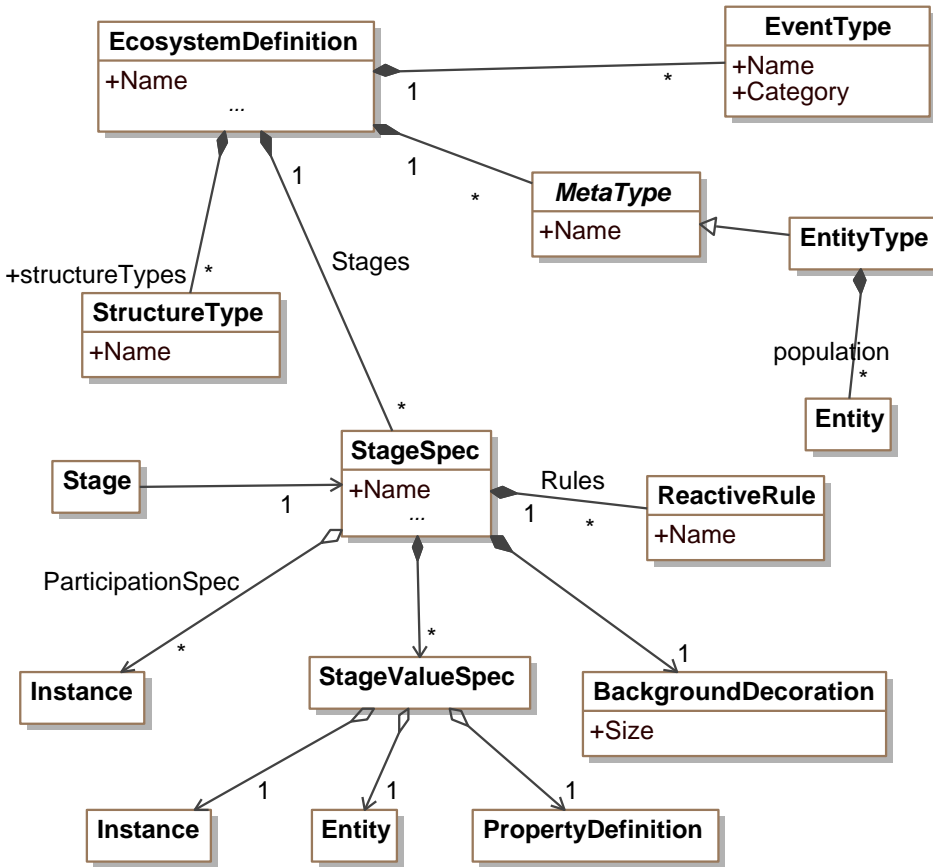


Figure 3. Primary concepts in an ecosystem.

3.4 Entity Model

To support entities, a strongly typed system has been devised to easily support specification re-use, similarly to the approach that introduces concepts such as *class* and *object* in Object-Oriented Programming and modeling languages (see Figure 4). The most important class is the *EntityType*, which supports the defi-

nition of entity types. Each entity type consists of a set of property definitions and a set of action specifications. An entity type can be specialized into an embodied entity type if it is to have a physical behavior and a visual representation during the stage simulation. *EmbodiedEntityType*s have a physical structure definition realized by a structure type in the model. This will be exhaustively described in the next section.

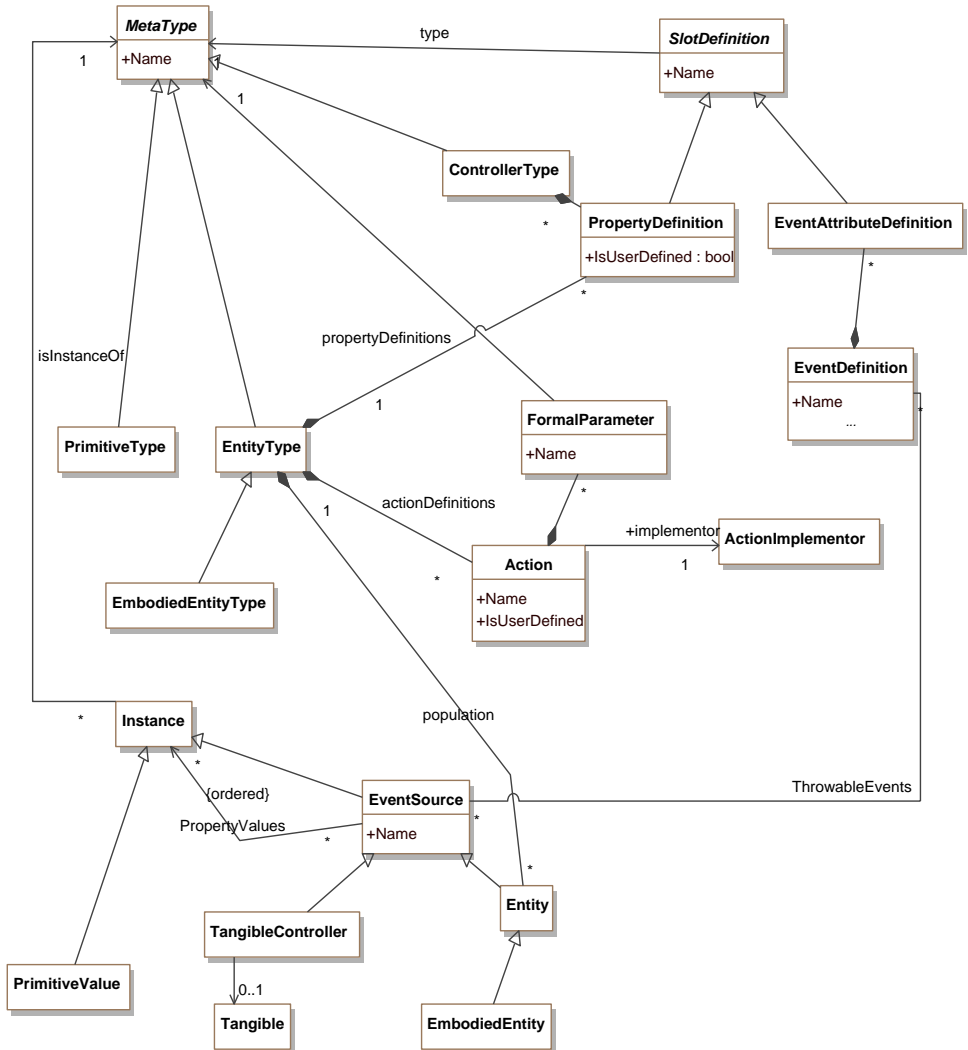


Figure 4. Type system diagram.

Properties are like qualities or attributes that entities have. They have a name and a type, which can either be an entity type or a primitive data type. For this reason, and because we are modeling types, we also provide the class *PrimitiveType* to support the non-entity basic predefined types, and the class *MetaType* that generalizes both sorts of types.

Actions can be understood as the active behavior that entities may exhibit. Therefore, an action is like an operation that the entity can perform. The class *Action* has a name and an ordered set of parameters. Moreover, a parameter is characterized by a type from the ones being defined. Beyond the action definition, we need to provide a mechanism that allows the specification of what the action actually does, and for this purpose a range of different specification languages such a graphical or scripting languages or just a custom third-party library that provides the implementation could be adopted. We abstract this requirement in our model with the class *ActionImplementor*. Actions can be invoked in three different ways: by another action, by a rule, or even by participants through their entities. In our prototype system, as an initial approach, actions have been implemented by means of a function library as explained in Chapter 6.

The part of the model presented above deals only with the meta-descriptions required to specify entities, which are the elements that will populate stages in the end. To represent the concrete instances conforming to the types defined in the ecosystem, a general term *instance* is needed. The corresponding class is specialized in entity, primitive value, etc. according to the specific type that an instance is conforming to in the end. On one hand, the class *PrimitiveValue* wraps actual values of primitive types and facilitates type conversion operations between compatible types (e.g. integer and float). On the other hand, the instances of entity have an ordered collection of instances for their property values, indexed by property definition. Indeed, the type for each property value must conform to the corresponding property definition which is specified in the type of the entity. Furthermore, the instances of embodied entities have also a concrete representation as they require attributes to extend entities with physical behavior.

3.5 Physically-based Entities

An important part regarding entities is the type support for properties and actions, since this computational approach to data and services will allow the construction of rule-based mechanisms on top as it will be discussed in Chap-

ter 4. However, a more important aspect that has not been described in detail yet is the physical behavior associated to embodied entities. Its relevance is twofold. Firstly, it provides a visualization facility. This is a key issue because an entity in an ecosystem is not only about computational concepts but also about how it is represented and can then be manipulated. Secondly, their physical representation, in terms of bodies and joints between them, determines how entities behave when they move around a stage being simulated.

In our model an entity-type just provides the definition support for an aggregation of computational concepts such as properties, actions, etc.; moreover, an entity is a specific instance conforming to an entity-type definition. When an entity requires of physical existence, the concept is extended to be an embodied entity. This makes an embodied entity to have an embodied entity-type as a type. It allows high re-use of definitions across multiple instances. The definition in which an embodied entity relies on is the embodied entity-type (see Figure 4 and Figure 5).

An embodied entity-type consists of a structure-type and a set of proto-costumes that will define the available visual representation. A structure-type effectively specifies the composition of the entity in terms of components and joints, conferring physical behavior. Figure 6 shows several structure-type samples, which could be used by embodied entity-type definitions. Basically, a structure-type consists of a set of structural-components, which represent rigid bodies, and a set of structural joints, which represent articulations that join them (see Figure 5).

The structural-components have physical rigid bodies, which means that they can collide with each other and be applied forces and impulses on them. The basic shapes for these blocks are rectangle, ellipse, and triangle, as well as any 2D polygon with more than three vertexes and whose sides do not cross with each other. These components have certain physical attributes. Most of them are typical and common such as the rotation, position, mass, size, etc. But some others are also relevant because they allow the physics engine to do the work. For instance, the physics engine combines the moment of inertia with the mass to compute inertial forces. Also, the friction coefficient determines how polished or rough the component is, and the physics engine takes this into account to simulate the friction forces. The attribute *IsStatic* is useful to indicate whether the component can be moved by applying forces or impulses. Actually, it is equivalent to establish either an infinite or ground value for the friction coefficient according to its Boolean value.

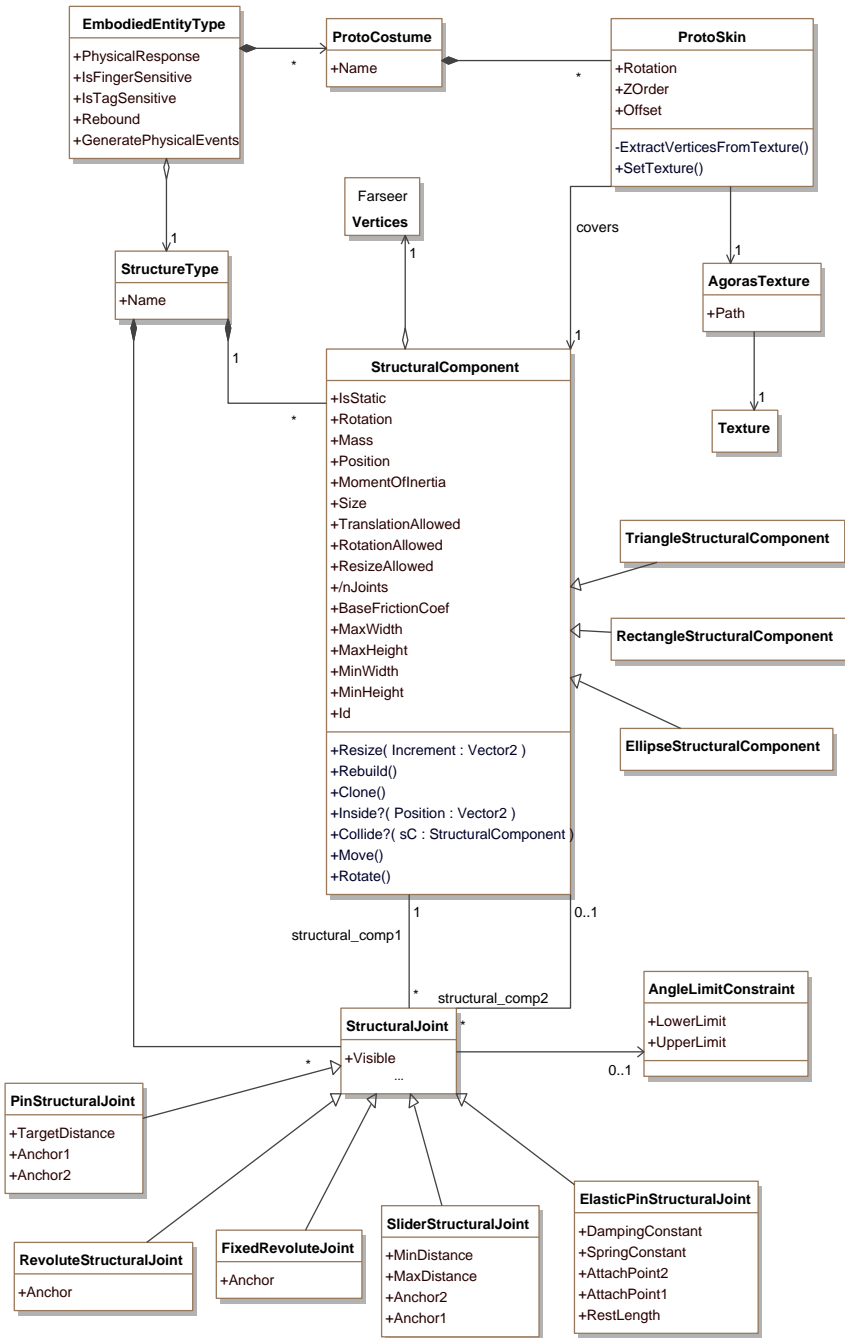


Figure 5. Physical description for the Embodied Entity-Type

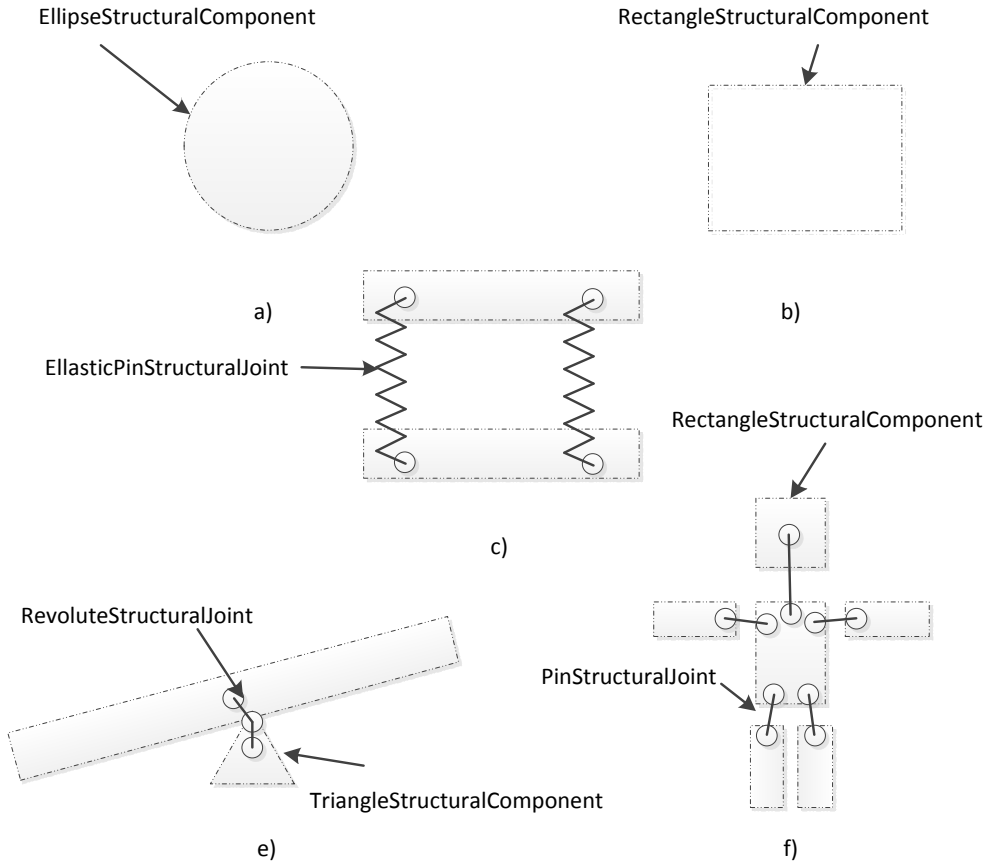


Figure 6. Several structure-type samples.

The other elements composing structures, the joints, are connectors either to join two components or anchor one component to a point in the surface background. Joints do not have associated physical bodies, such as structural components, and this means that they are more like forces or ropes keeping pieces joint rather than objects being able to collide with other elements. There are several structural joints, each one with different physical features and performance. The most basic joint is the *PinJoint*. The ends of this kind of joint are just fixed to a position in two different blocks, and is like a rigid rope or wire. A similar joint is the *FixedJoint*, whose aim is to anchor a component to the world, in such a way that the center of the component always remains to a prefixed distance with respect to the world anchoring point. The *RevoluteJoint* works by creating an intermediate point between two components, so that the distance from such a central point to each component's center remains con-

stant. The *SliderJoint* works similarly to a *PinJoint*, but with the difference that both maximum and minimum distances are specified to establish in which range of distances the connected components can eventually be. Another popular joint is the *ElasticJoint*, which behaves similarly to a spring, including both spring and damping constants. The last two joints are unique in the sense that they do not keep the distance between components constant.

To some extent the structure-type is like the skeleton or body for the entity type definitions. It specifies its physical behavior. Taking such a view, a proto-costume is like the skin or clothes that cover the structure-type. It therefore specifies its look. As a body should have several clothes in the wardrobe for each occasion, an embodied entity-type should have a set of proto-costumes that cover its structure-type. Figure 7 shows such an idea for three basic structure-types.

As a structure-type consisted of a set of structural components, a proto-costume consists of a set of proto-skins that visually cover these components. A proto-skin relies on a texture to do so, obtaining the geometry shape directly from it. The proto-skin is applied a rotation angle and positioned according to an offset from the center of the associated structural components.

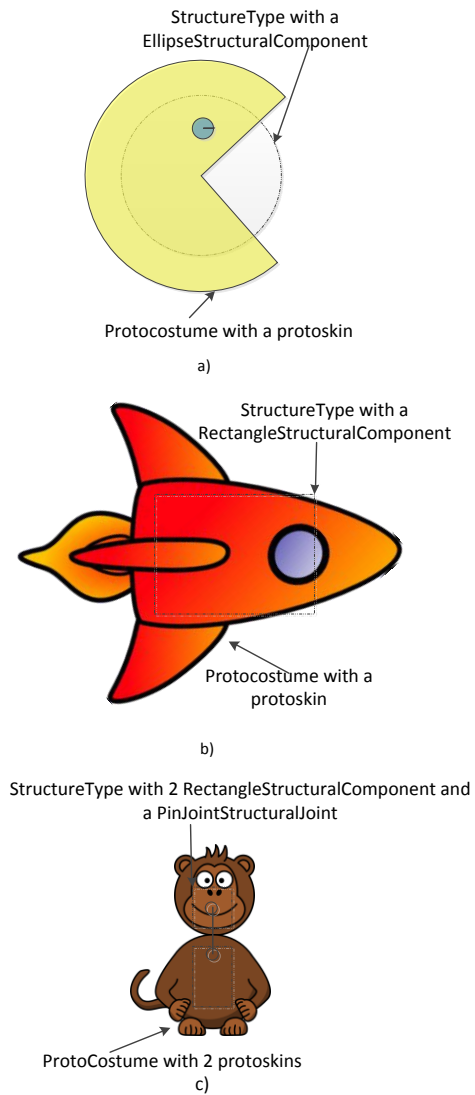


Figure 7. Several Proto-Costume samples.

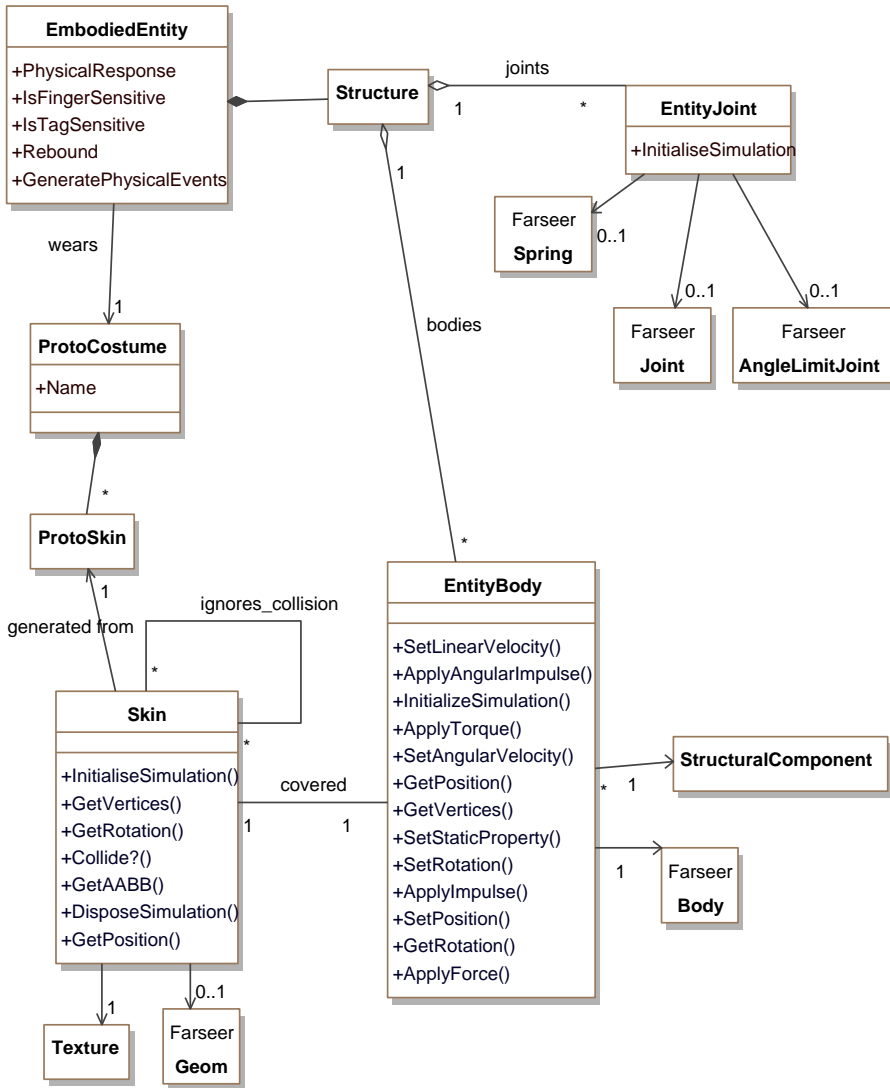


Figure 8. Structure of an entity.

With this basic conceptualization for the physical behavior, the mold to give shape to instances according to embodied entity-types is provided. But to definitely represent and give life to occurrences conforming to embodied entity-types, the concept “embodied entity” is still needed, along with other corre-

sponding concepts representing the specific counterparts for the structure-type and proto-costume used by embodied entity instances.

An embodied entity instance has several attributes aimed at specifying several visual, representation and interaction issues (see Figure 8). For instance, some of the most important ones are:

- *PhysicalResponse*, which indicates whether the embodied entity must participate in the physical simulation or not. This is useful to indicate whether an entity should collide with other entities or not. In this way, an embodied entity without physical response could be easily used to display information that cannot be affected by the physics engine simulation;
- *IsFingerSensitive*, which indicates if an embodied entity can be interacted by means of fingers on the surface;
- *IsTagSensitive*, which indicates if an embodied entity can be interacted by means of tagged tangibles on the surfaces;
- *Rebound*, specifying how the rebound must be simulated. This is important in some games that require some entities to rebound not completely following a real accurate physical simulation. For instance, in a Pong-like game, the energy's ball would get decreased at every bounce according to an accurate simulation model. Such an energy lost would lead the ball to stop in the end, and the game experience would be ruined. To avoid this, this attribute allows us to specify that the ball will not lose energy when it bounces;
- *GeneratePhysicalEvents*, which specifies whether events related to the physically-based simulation of the entity are notified or not. This is relevant for the simulator, to manage and evolve the stage and accordingly raise rule-based behavior depending on these events.

Notice that the previous attributes are also defined in the class *embodied entity-type*. They are in both places because when an instance is created, it takes the values from the type (coarse grain specification), but each instance can modify these values conveniently (fine grain specification).

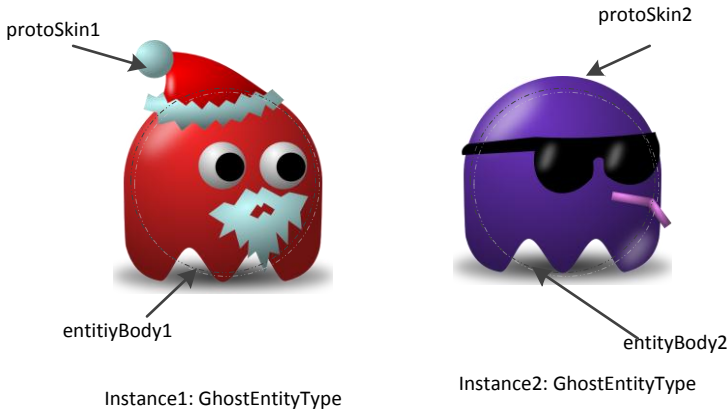


Figure 9. A visual example of objects instantiated to specify an embodied entity-type.

Embodied entities, which are the ones being actually simulated in the end, take physical and visual specification from the information described in the corresponding embodied entity-type. The structure-type of the entity-type along with a proto-costume that the entity is to be wearing, are taken to generate the structure. This structure is composed of a set of entity-bodies and a set of entity-joints. The elements of these two collections are generated from the specification in the structure-type. Each entity-body is covered by exactly a skin, which is generated from the corresponding proto-skin in the proto-costume being worn. These entity-bodies, entity-joints and skins are needed to allow multiple instances to be created from the same embodied entity-type, and are the building blocks of the simulation since they are the elements that can be physically simulated. Figure 9 depicts two instances of the same embodied entity-type, illustrating the instantiation needed of all these classes.

Once presented the model of AGORAS supporting physical 2D entities, and in order to fulfill one of the main objectives of this dissertation, now it would be interesting now to explore creativity in collaborative construction tasks as those which would be present in an AGORAS prototype. The following section introduces the foundations of such an exploration departing from the motivation about why creativity is important for human development, and presents the experimental evaluation conducted.

3.6 Exploring Creativity in Collaborative Construction Tasks

3.6.1 Creativity Assessment Model

People are continuously solving problems in their everyday activities. Some of these are "routine" problems, which are easy to solve and have obvious and well-known criteria for identifying the solution by applying knowledge directly. Conversely, other problems are more difficult to address and their solutions are not directly identifiable (*Cropley, 2001*), in which case other factors should be considered in order to come up with a solution to "complex" or "intractable" problems. The first relevant factor here is intelligence, which is commonly considered to be the ability of an individual to solve problems. In general, the more problems an individual can solve, the more intelligent he/she is considered to be. Intelligence has been widely studied and is traditionally measured by IQ tests, which are actually concerned with convergent aspects of thinking (*Guilford, 1970*). In many cases, people are unable to solve these "complex" problems even if they are considered to have an adequate level of intelligence. Such problems are complex precisely because they are difficult to define, which in turn makes the solution even more difficult to obtain due to the complex mental processes involved. Divergent thinking is a desirable characteristic and is often associated with highly creative abilities. In that sense, creativity can be considered to be directly linked to problem solving and is even a special form of problem solving (*Guilford, 1970*), (*Newell, 1972*), (*Mumford, 1996*) because of the mental processes involved in creating ideas, which include preparation, incubation, illumination and verification (*Rhodes, 1961*)(*Runco, 1995*).

Creativity is therefore important for learning and personal development. How it can be fostered as well as evaluated in Information and Communications Technologies (ICT) settings seems to be a key issue for research. *Farooq et al (Farooq, 2007)* performed a study aimed at detecting breakdowns in creativity by using the BRIDGE system, a desktop-based prototype of a collaborative infrastructure that provides integrated support for the process of creativity of graduate students in computer and information science. Four breakdowns were identified: under-consideration of minority ideas; loss of novel ideas; lack of critical evaluation of perspectives; and weak reflexivity during convergence.

Another study that includes creativity assessment in the context of Software Engineering is the one by *Wang et al (Wang, 2010)*. In this work, although participants were asked to collaborate and communicate, no particular medium was specified nor was specific software developed. The study explores the relationship between the design rationale in software design and creativity. The

task required participants to solve specific software design problems by capturing design rationale and implementing the design in Java. As they progressed, students produced design rationale documents following a uniform format. These documents described design issues found in the exercises, design alternatives considered, along with a tradeoff assessment for each alternative and, finally, a report on the selected alternative. An assessment of design rationale quality and design creativity based on three creativity traits (novelty of design alternatives, persuasiveness of tradeoffs, and insightfulness of tradeoffs) was performed on the documents handed in by students. The authors concluded that it is possible to foster design creativity by enhancing the quality of design rationale.

However, although some studies like the above ones have been published on the evaluation of creativity in ICT contexts, most have focused on systems development whose design is rooted in creativity theories, and which is thought to foster creativity because it is used to address creative tasks, and/or support typical creative processes. For example, *IncreTable* is a mixed reality tabletop game based on the idea of Rube-Goldberg machines (*Leitner, 2009*). Each level presents a puzzle requiring multi-modal interaction to encourage user creativity. The general objective of the platform is to arrange a given collection of items in a complex way in order to solve a puzzle. Subsequent evaluation of the platform explored the relationship of certain interaction aspects with flow (*Chen, 2009*) but creativity was not explicitly studied.

There are also other proposals in the context of tabletop systems but primarily focused on supporting brainstorming processes on collaborative conditions. Firstly, *Buisine et al (Buisine, 2007)* present a tabletop interface enabling groups to build mind-maps as a tool for associative thinking and group creativity. The study compared this interface to traditional paper-and-pencil mind-mapping sessions. Questionnaires were used to evaluate subjective perception on ease-of-use and usefulness of the system, and video analysis was included to evaluate participants' collaboration. The results showed no difference in the production of ideas, but the tabletop condition significantly improved gestural and verbal interactions, as well as the perceived efficiency and pleasure of working in groups.

Secondly, a system based on interactive tabletop and digital pen interaction is presented for browsing topics using a zoomable pin board metaphor in the work by *Geyer et al (Geyer, 2010)*. The digital pen is used to annotate idea scribbles that can be easily added to the system. An exploratory study was conducted from user feedback questionnaires involving professionals from crea-

tive industries participating in a workshop. The authors found that the design space for combining digital pen & paper with interactive tabletops is promising and therefore it is worth investigating these aspects for supporting creativity techniques and design methods in this professional context.

Thirdly, Friess et al (*Friess, 2010*) present a multi-touch based tabletop application including a study on the use of the interface by considering several creative techniques. The results showed that the subjects positively assessed the realistic behavior physically-based simulated objects as this provided a more intuitive interaction. They also felt they participated more actively as a consequence of using the tabletop application, which is important to foster collaboration. The application is designed to generically support a range of creativity techniques as established in a previous study (*Forster, 2009*).

Finally, a relevant contribution to the body of knowledge on creative design is the work by Vyas et. al. They widely explored several real design studios and settings such as educational design departments (*Vyas, 2009a*) and design companies (*Vyas, 2009b*) with the aim of identifying relevant practices that support the creativity of design professionals. The main broad themes identified by means of an ethnographic approach to such collaborative practices are reduced to externalization, use of physical space and use of bodies (*Vyas, 2009a*). Externalization refers to the representation of design knowledge in any form outside the thinker's mind (e.g. sketches, models, prototypes) typically used for establishing common-ground among co-designers. It plays an essential collaborative role in different activities and aspects of design such as exploration of ideas, thinking by doing, and coordination among teammates. The use of a physical space concerns to all the materials and artifacts that help co-designers organize, coordinate and manage their design work (e.g. to-do lists, project-related information, sketches, organizational details, etc.). All these artifacts accordingly arranged help in establishing creativity by elaborating the problem, supporting awareness and providing both personal and shared informational spaces. Finally, the use of body theme refers to design practices in which body expression plays an important role in creativity by enhancing processes as exploring and communicating design knowledge in group.

Departing from all these factors and considerations, the authors devised two conceptual systems that could potentially be used to support collaborative design activities in the context of the design studios studied. The first system is what they called *the resource sharing concept*, which basically consists of a tabletop system in which designers could use to discuss, share and see the design history of products in a co-located fashion. The second concept is called *live discussion*

concept. It is about a distributed system consisting of high resolution cameras, large displays and RFID technology that supports the discussion ideas and designs between two designer teams located in two different places. However, besides these conceptual proposals, the main technological contribution of the authors in this line is CAM (Cooperative Artifact Memory) (Vyas, 2010a)(Vyas, 2010b). It is a mobile-tagging application that supports designers to collaboratively store information such as messages, annotations and external web links related to design artifacts. CAM has shown to effectively support participants awareness and coordination as well as facilitating exploration extension of artifacts in the creative process of design work (Vyas, 2010c).

In view of how creativity is usually considered in the previous works, we may conclude that despite advanced technology is being used to provide systems according to creativity theories to support creativity they rarely assess creativity itself, what brings up doubts on whether technologies actually provide some benefit in the expected direction. Moreover, many of the studies focus quite often on usability or collaboration design issues that correspond to more general and broader studies in the field such as the ones by Hornecker et al (Hornecker, 2008) and Rogers et al (Rogers, 2009). The research presented in this section is motivated by the expectation that tabletop technology and the evaluation of creativity will lead us to a better understanding of the creative process itself and will allow us to generate better creativity support systems in computing in the future. Hence, with the aim of both exploring if interactive surface technology for creative tasks in the context of creative learning with teenagers is promising and validating the proposed model for physically-based entities, this chapter contributes by using a creativity assessment model and conducting an empirical study that measures creativity traits on two tabletop settings as an approach to evaluate how the environment can influence creativity.

In order to propose an effective creativity assessment model, a good underlying theoretical framework must be selected. However, as it was already discussed in the Chapter 1, the term creativity is very difficult to define as shown by the many different definitions offered in the literature (Treffinger, 1996b) by adults and even children (Aleinikov, 2000).

Given the difficulty of establishing a precise definition, creativity is normally considered as a construct composed of several traits, as described by psychologists (Guilford, 1970). In this respect, a range of different adjectives and traits have been jointly used as indications of creativity. Some typical examples are originality, trade-off assessment, independence, elaboration, curiosity, frustration tolerance, establishing remote relationships, being open to new experienc-

es, and many others. In addition, confluence theories nowadays consider more complex and multi-component approaches to creativity, since it is thought to emerge from several interrelated factors (*Sternberg, 1996*). To some extent they also include external variables from the environment since several factors in it may influence and be supportive for creativity (*Sanyer, 2006*).

Our creativity assessment model is based on this view on traits, considering a representative core set used in the psychology field, but with the idea that these can be to some extent impacted by the environment in which subjects interact. This model contains *novelty, fluency and flexibility of thinking, elaboration, and motivation*. The most important trait is undoubtedly novelty, which is defined as the characteristic conferring something unusual, unique or surprising. The fluency of thinking refers to the ability to generate new ideas and/or formulate significant problems and hypothesis (i.e. ability to provide a range of valid solutions); flexibility of thinking refers to a wide range of possible solutions and to the ability to change from one category or point of view to another. Elaboration is the ability to increase the complexity of ideas, including more details (although too much elaboration may have undesired effects by limiting the development of ideas). Finally, motivation was included as it is also important in human development and many other learning activities (*Csikszentmihalyi, 1988*) and it is also included in one of the seminal definitions given by Amabile. This creativity model will be used in the empirical study presented here with the aim of assessing the influence of using a platform in the environment.

Taking as an underlying creativity assessment model the one described above, our proposed study compares a tool based on our proposed conceptual model for entities implemented in interactive tabletop technology versus a completely physical and traditional tangible setting (i.e. with no computer mediation). Although a full prototype of AGORAS would be expected to be used in the evaluation covering all the concepts in the proposed entity model, there are some limiting issues to be considered. These mainly refer to the fact that we need a counterpart platform to be compared. However, there is no available alternative platform with so much functionality. Moreover, in the case of finding a suitable platform for comparison, in terms of functionality, the interaction methods are still an issue since the use of a platform with identical functions but with very different interaction profiles would not be a fair comparison at all. With all this in mind, the idea consisted of developing a limited tabletop version of the proposed model of entities but only focused on the construction of structures as entities that are physically based constrained, so that in such a prototype basic interactions and the creative process will still remain as required in the final and complete version of an AGORAS editor prototype. In

this way the digital system can be compared with a physical-only platform that we have built for conducting these experiments.

Two different creative collaborative tasks have been involved in the experiments on both platforms. The first task consisted of the creation of entity structures freely, which involved more artistic facets and simple use of physical building blocks. The second task was inspired by the idea of solving problems by means of Rube Goldberg Machines (RGMs) (*Rube-Goldberg, 2012*), which are mechanical systems mainly composed of building blocks connected to actionable devices, normally providing a complex solution to a simple problem.

3.6.2 Creativity Tangible Simulator Component

The creativity tangible simulator and editor component is an environment supporting the creation and simulation of physical structures that has been developed for this experimental evaluation based on the previously presented conceptual model. The user interface is based on an interactive surface enabled with multi-touch and tangible input. The software should basically support the construction of physical structures according to the model depicted in Figure 5. However, as the construction of the proto-costume is not going to be considered in the prototype (would provide the visual aspect of structure components) and there is no need to support multiple instances of structure-types at once, a simplified version of the model has been reworked. Similarly to the original definition, a structure is composed of blocks and joints, which are essentially the basic construction elements in the entity model of AGORAS.

The model in Figure 10 basically has collapsed the type-instance levels for building structures into a single level. In this way, the structural components are made directly visible according to their shapes and can be simulated by using the corresponding counterparts in the Farseer⁵ physics engine. So blocks are basic shapes that are able to collide and can be affected by forces by touching them.

User interaction relies on multi-touch input and tangible tools in the form of pucks, in such a way that users can interact at the same time and collaborate in the construction of structures. Touch input is used to positioning and rotating blocks and joints, while tools are used for a range of different operations. The tools are provided by tagged tangible pucks with a specific function (see Figure

⁵ Farseer Physics Engine in Codeplex: <http://farseerphysics.codeplex.com/>

eral menus for creating the parts of a structure that can be operated in parallel by the users. Figure 13 shows the representation of the different joints and the menu giving access to them in the system.

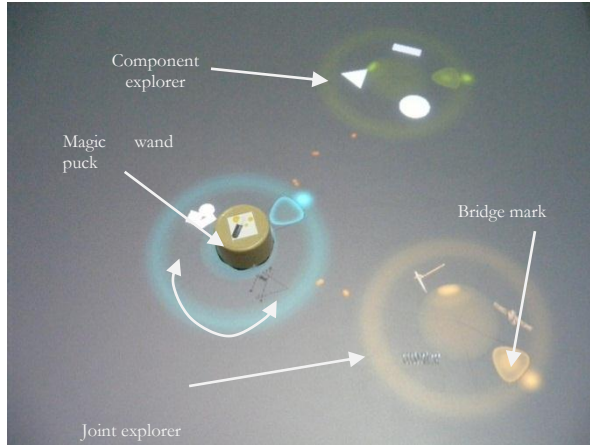


Figure 12. Parallel system menu.

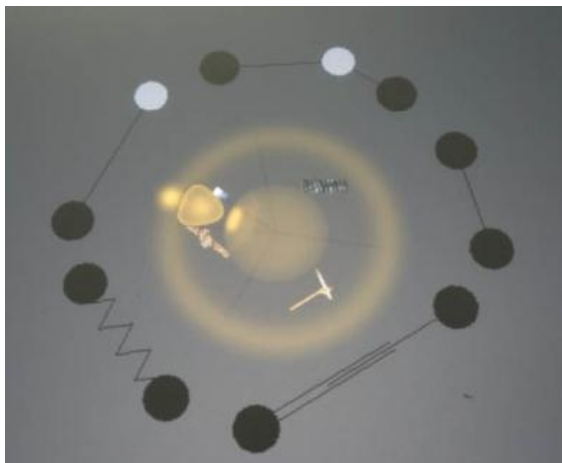


Figure 13. Joints and Joint Selector Menu.



Figure 14. Creating a sample structure.

The “clone” tool allows the copy-and-paste of blocks already existing in the workspace. To do that, the user only has to put it down on a block and then to be put back down in the place to create the copy. This tool furthermore allows the fine adjustment of blocks in terms of position and rotation. Another important tool is the “eraser”, which deletes any block or joint when the tool is applied on an element by means a zigzag gesture. The “friction modifier” tool allows the adjustment of the friction coefficient of the blocks to be used when block surfaces touch each other. Finally, the “simulation” tool alternates between the editor and the simulator. When this tool is present on the surface, the simulation is started and performed. This mechanism allows users to observe the structures evolving according to physics and also interact with them to introduce forces and impulses in the system getting blocks moving as desired. Figure 14 shows the creation of a structure by two people collaborating using touch input and tools.

3.6.3 Experimental Evaluation

The exploratory study was designed to get insight into whether interactive surfaces show promise as the base technology for collaborative creative tasks in terms of creativity traits. The study compares the performance of two different testing platforms used by teenagers in an experimental design that considers individual thinking-reflection, collective discussion, and action processes in an iterative task to foster creativity.

3.6.3.1 Participants

Twenty-two (14 male and 8 female) teenage students from several local secondary schools participated in the experiment. Two participants were left-handed and two were ambidextrous. Their ages ranged from 15 to 18 ($m=16.23$, $sd=1.6$). Almost all of them declared they used personal computers regularly. Regarding using touch-enabled devices, fourteen reported using them daily, four almost every day, three said seldom, and one never. None had any previous experience of surface computers.

Participants had previously taken part in a short course on new and emerging technologies designed to motivate teenagers to study core subjects such as physics and computing. The course was organized by a club dependent on the Education & Culture department of the local city council. Since the course was completely free and voluntary, there was no kind of participant pre-selection according to school performance profiles.

3.6.3.2 Equipment

Two tabletop platforms were developed for the experiments. One is the digitally-augmented platform presented in Section 3.6.2 based on an interactive surface that allows multi-touch and tangible input, whereas the other is completely physical and tangible without computer mediation as described next.

The alternative physical-only platform is made entirely from hardware with no software simulation. It consists of a conglomerate 590x700 mm. tabletop with a regular grid of 28x32 holes with a separation of 2 cm. (see Figure 15 and Figure 16). Several wooden blocks of similar size proportion to the ones in the digital platform are available. The tabletop has four legs to keep it horizontal and also a stand to configure it as a slanting plane to simulate similar conditions in the digital platform.



Figure 15. Wooden building blocks and joint elements for the physical-only setting.

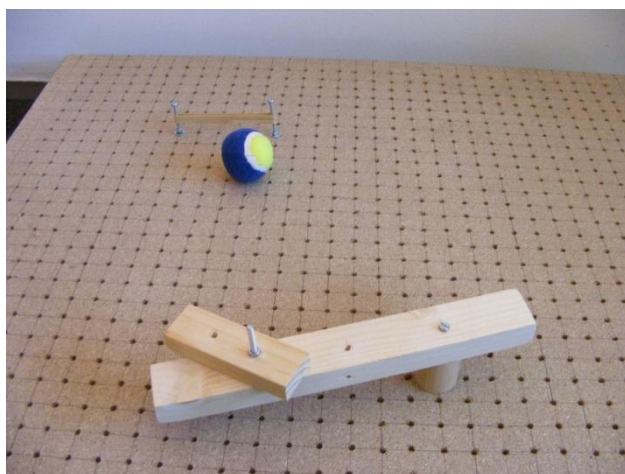


Figure 16. Example of joints and block assemblies.

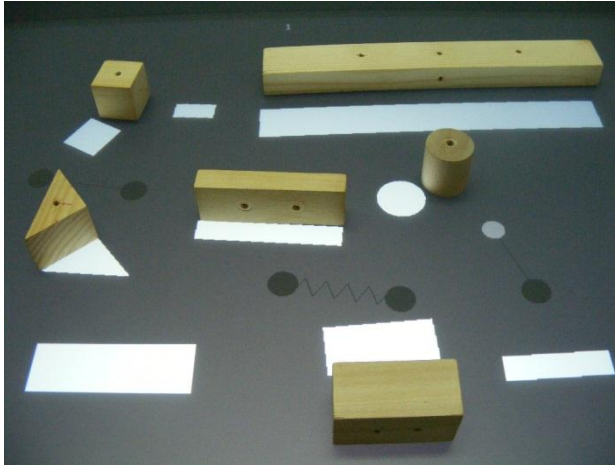


Figure 17. Building blocks and joint elements for the digital setting.

The blocks can be fixed and assembled as needed by using the holes drilled on them by means of screws, bolts and other joint elements such as elastic bands and pieces of string (see Figure 15). The blocks are basically wooden building blocks from construction play sets for children which have been drilled. They consist of cylinders, boxes, cubes and triangular prisms, which seen from a point of view orthogonal to their faces would describe circled, rectangular, squared, and triangular shapes (see Figure 17). The joint elements used to keep blocks fixed or to create movable constructions are basically based on short strings, elastic bands, screws, hooks, nuts and bolts. By combining several blocks and joint elements more complex joints and other functional components can be assembled. For instance an elbow joint, a revolute joint or even a catapult could be assembled as shown in Figure 16.

This tangible platform allows the construction of a variety set of fixed or articulated components based on basic rigid bodies and joints as described above. Users have a high number of pieces of each type at hand in a bucket and they only have to grasp them as needed.

The choice of a physical-only tangible platform instead of a platform based on a desktop application was made in order to have two similar platforms in terms of co-located user involvement and participation. A desktop-based application relying on WIMP (Windows-Icons-Menus-Pointers) interaction techniques would not have provided a fair comparison, since the degree-of-collaboration and participation would have been limited to non-parallel manipulation by single-user input interfaces such as keyboard and mouse.

3.6.3.3 Tools and Instrumentation

The experiment used the two platforms previously described. Participants were given answer forms to report proposed solutions by means of a sketch and notes before implementation on the platforms. Additionally, two video cameras were used to record the sessions to support video analysis. For this purpose, colored cards and pucks were used to identify participant groups and switches between workspaces, and colored strips were also tied around users' wrists for identifying participants' hands in the video.

3.6.3.4 Method and Procedure

The test sessions were carried out at the end of the technology course. Participants were assigned in sessions according to their availability and age, with a limit of 8 people per session and avoiding large differences although all were teenagers. They were grouped in pairs randomly, but always trying to balance the assignment to the experimentation platform (i.e. digitally-augmented or physical-only) following a between-subjects experimental design which would perform two experiments swapping the technological platform for testing (see Figure 18). After pairing, the average age difference between members was about 0.9 (group age: $m=16.27$, $sd=1.14$). This resulted in 6 mixed groups (i.e. composed of members from each gender), 4 formed by two male teenagers and 1 by two female. From these, only a group of friends were coincidentally paired, while other 6 groups declared to casually know the partner but only in the context of previous courses in the club, and 4 groups reported that they did not know the partner before the current course. Each group received an introductory talk on each experiment platform, followed by a live demo of how platforms could be used to solve a demo problem, and finally, they were required to give an alternative solution. Their proposals were implemented on each platform on their own under supervision. This introduction and training session took about 40 minutes.

Within every experiment, three distinguished places were considered in an iterative process (see Figure 19). In the individual thinking place, subjects had to generate solutions to the problem on paper. Once each member had produced various solutions, they discussed improvements and possible new solutions and decided what solutions to implement on the testing platform. As they had discussed the ideas on paper, they already knew what parts were needed to be constructed and could collaborate on implementing them. The first two stages are thus also important, as they promote divergent thinking, which is important for creativity, since the production of sketches supported by the traditional

paper and pencil may facilitate a greater generation of proposals, and also as they set the basis for collaboration on the experimentation platform.

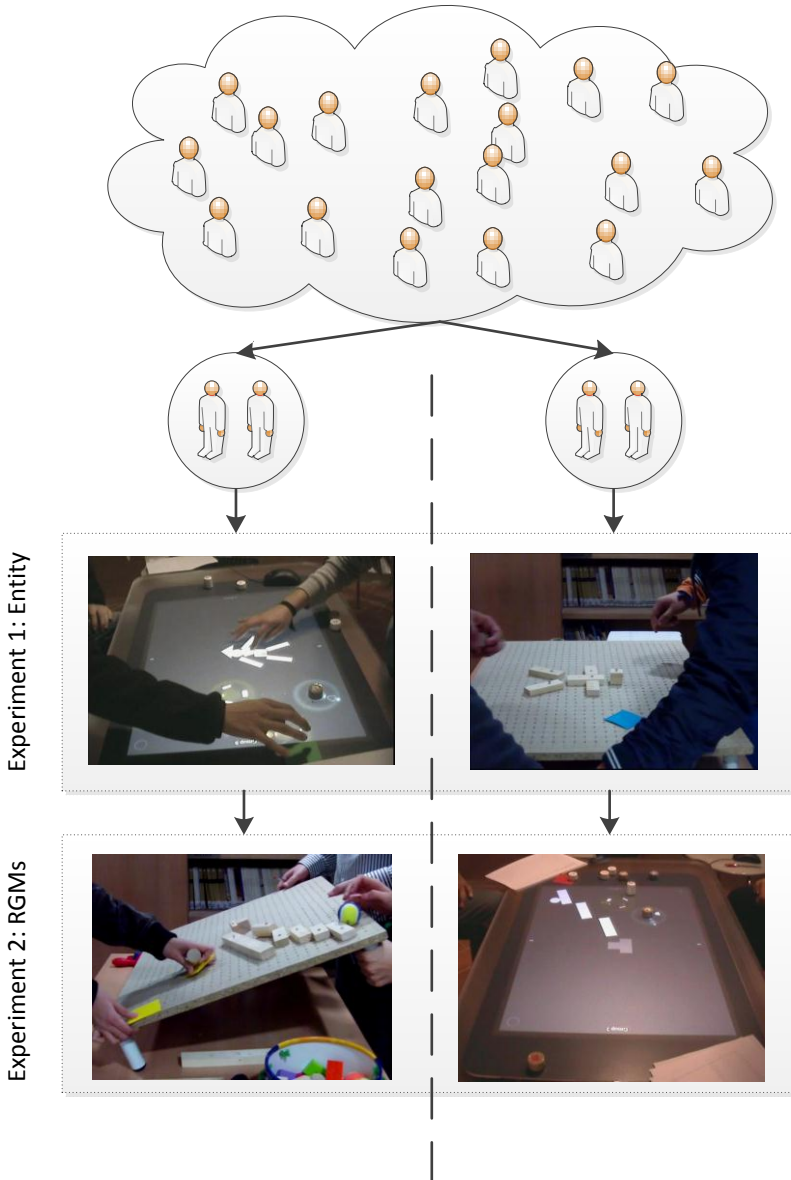


Figure 18. Experimental Method.

Participants were encouraged to perform as well as possible, and two prizes were awarded for the best two groups. They were told that performance would be judged based on the creativity and originality of the solutions. They were reminded that it was important to give expression to as many solution proposals as they could on paper, in order to promote divergent thinking and solution diversity. Although external rewards could be thought to go against intrinsic motivation and even have a negative effect on the creative performance according to some results by Amabile (*Amabile, 1978*). In this work, she concluded that “the imposition of an extrinsic constraint upon performance of an activity can lead to decrements in creativity” in adult women subjects. However, the study by John Baer replicated some experiments conducted previously by himself and included also some research questions as in the Amabile studies providing more knowledge and evidence on this issue (*Baer, 1998*). While Amabile mostly focused on girls, Baer includes in his studies balanced samples of girls-boys in the middle school. The findings are consistent with Amabile’s only when the sample is reduced to girls. The effect of evaluation awareness is low in the case of mixed samples and null if only boys are considered.

Moreover, some findings in this study suggest that if subjects are told that they will be evaluated and also how the evaluation process will be done, then such awareness can entail more creativity instead of less as suggested by Amabile. Thus, we think that allowing awareness about the evaluation and providing external rewards to increase external motivation are an additional way to get people motivated and do not hamper creativity.

Participants were obliged to go to the next place if the 10-minute time limit was reached. These three places were put in a loop until the 60 minutes experiment time was reached.

Following the experiment design considerations, the experiments were only conducted once, and balanced designation of groups for using the experiment platform was carried out because of time limitations. Thus according to the designation, each group only interacted with one platform in the task being reported. The recordings were analyzed off-line by the experiment designer extracting information about performance on implementing solutions, their complexity, behavior patterns and collaboration degree.

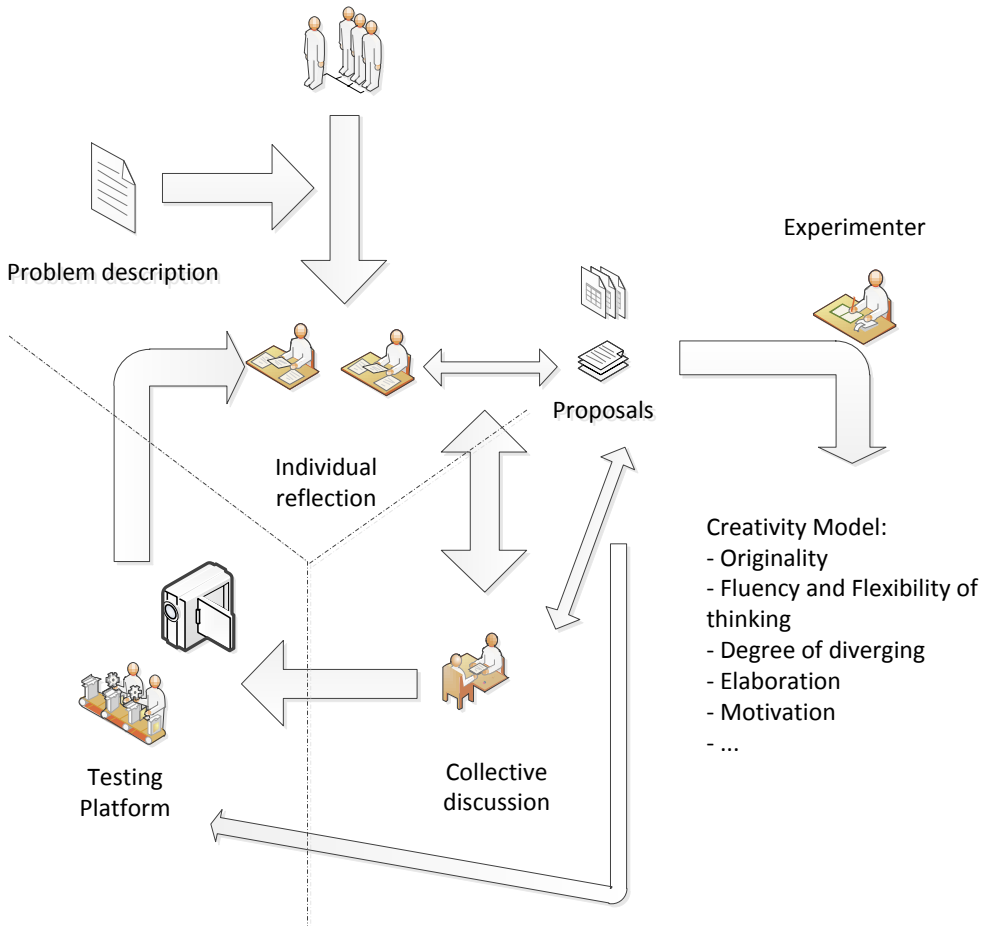


Figure 19. Procedure diagram in each experiment.

3.6.3.5 Experiment 1: Free Entity Creation

In this first experiment, the main interest was focused on exploring how participants perform in a general problem whose resolution is completely open. In this way, we need a task that is only constrained by the building blocks and the creativity of the participants.

3.6.3.5.1 Task

Participants were requested to produce as many solutions as possible to solve the creative problem stated like this: create entities with movable or articulated

components. By “entity” was meant anything, living entity or not, which could be represented with the material in the experimentation platforms. The creativity was not only expressed in the entities, but specially also in how they managed to make them movable by using joints.

3.6.3.5.2 Results

The participants formed eleven groups. Five were assigned to the digital platform and six to the physical-only. A total of 161 proposals were generated and 91 were tested in the end. In the digital platform, almost 5 proposals were tested per group on average, while groups using the physical-only platform tested on average 11 proposals.

The creativity model described previously determined the concrete variables to be measured to assess creativity. The traits in the creativity model have been measured as follows. The *fluency of thinking*, the ability to generate new ideas was considered to be related to number of proposals produced by the group in each cycle of the thinking-discussion-testing loop. This can give us an estimation of the capability of the platform to support the generation of new ideas although other uncontrolled factors can be present indeed. The groups working with the physical-only platform showed a significant higher fluency according to the comparison mean performed by a t-test ($t(27) = -2.689$, $p\text{-value} = 0.012$). On average, the tangible groups produced about 7 proposals per cycle ($m=7.1$, $sd=4.6$) and the digital ones 3 ($m=3.4$, $sd=2.4$).

Trait *elaboration* was measured as the complexity in terms of number of blocks and joints used to implement the solution (see Figure 20). The t-test also showed that differences were significant ($t(88)=4.106$, $p\text{-value}=0.000$) with subjects obtaining more elaborate solutions when using the digital platform (Digital: $m=11.96$, $sd=6.8$; Physical-only: $m=7.38$, $sd=3.6$).

Motivation was broadly considered by measuring the actual participation. Considering this objective approach may give us an estimation of how motivated the subjects were on using the platform, this measure was operationalized as the user manipulation time over implementation time. In both platforms this measure showed performances that were not significant. In the digital 65.94% while 58.93% in the physical-only platform.

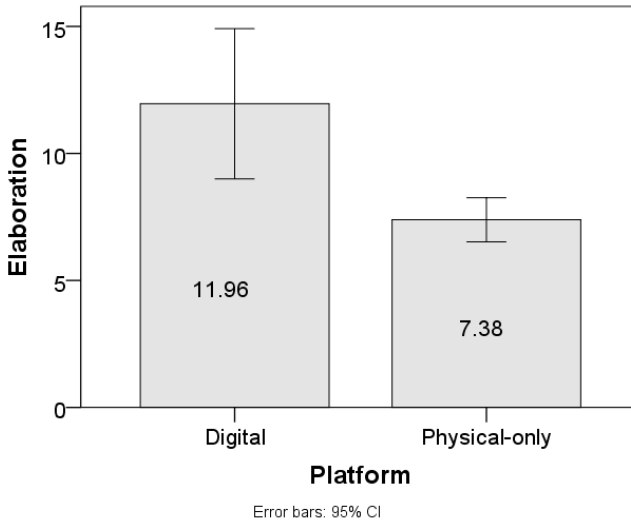


Figure 20. Elaboration by platform as entity complexity in terms of blocks and joints.

Finally, since *novelty* is difficult to assess and no clear objective measures can be found, we opted for ratings by experts. Two people with background in creativity studies were asked to rate each solution on a 5-point scale obtained as a cumulative assessment on several inner features. Each feature was described in a single scale of 3 levels (+0, +0.5, +1). These features concerned how unusual the creation was, whether the idea was useful or pointless, whether there was any surprising element or not, whether there were elements better suited to represent the idea or the mechanism or not, and whether the way of assembling pieces was commonplace or unexpected but advantageous. To check whether both judges agreed on the meaning of novelty and therefore on rating consistently the solutions, an inter-rater agreement test based on Kappa statistics was run.

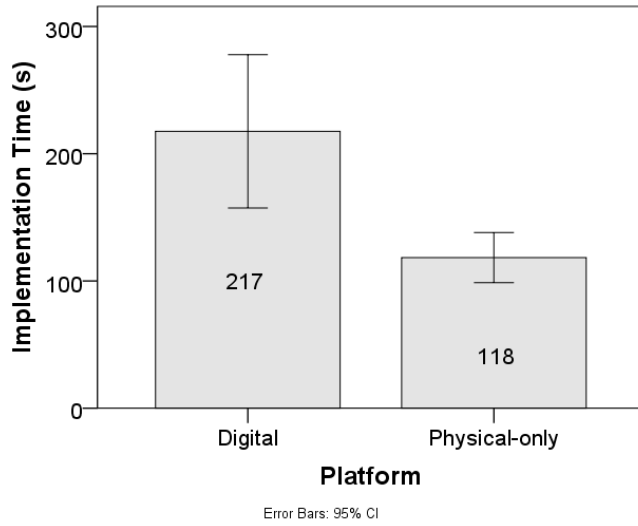


Figure 21. Implementation time by platform for the entity creation task.

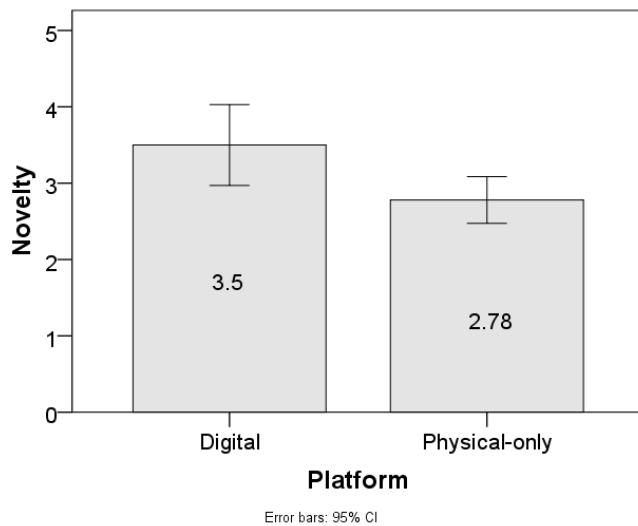


Figure 22. Novelty mean plot by platform in the entity creation task.

This test showed that the agreement was very good ($K=0.860$). Thus the rates were taken to perform a t-test to compare originality in both platforms ($t(79)=2.44$, $p=0.017$). The test showed significant differences in originality. On average, solutions in digital rated 3.5 ($m=3.5$, $sd=1.2$) and 2.78 in physical-only ($m=2.78$, $sd=1.2$).

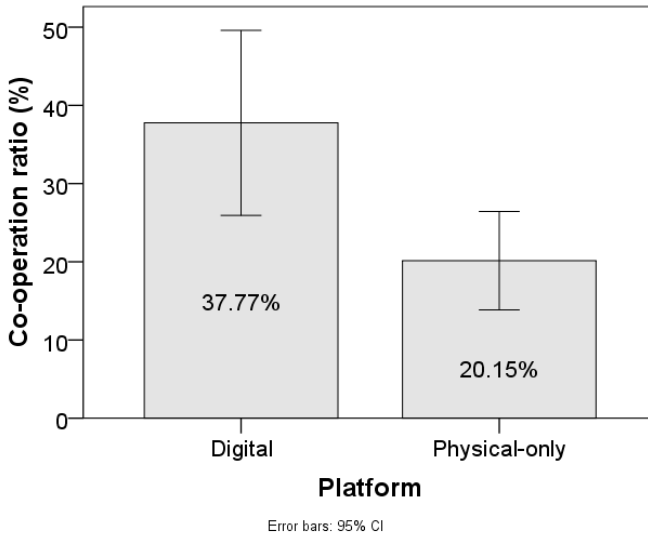


Figure 23. Mean plot for the co-operation ratio.

Besides creativity traits, the experiment is also useful to evaluate some interaction aspects primarily related to collaboration, and therefore to evaluate the suitability of technology to support collaborative tasks as the ones considered. On the one hand, the implementation time was measured (Digital: $m=217.56s$, $sd=139.35$; Physical-only: $m=118.36s$, $sd=80.74$). Since normality was not met in data, a Mann-Whitney test was run. It showed that the implementation time differences in both platforms were significant ($z=-3.19$, $p\text{-value}=0.01$). The time to implement solutions in the digital platform took longer (see Figure 21). Although some learning issues were observed in operating the digital platform, solutions were also more elaborated.

On the other hand, an interesting measure is co-operation time, which is the time that both participants in a group were effectively co-manipulating the platform, doing useful work, during the time needed to complete the solution implementation. It gives us an idea of how facilitating the platform is to support sharing and co-manipulation in the construction of structures. A priori, since both platforms are based on tabletops, an expected result would be obtaining similar cooperation profiles. However, co-operation was higher in the digital platform (about 37.7%) than in the physical-only (20.15%) as Figure 23 shows. Moreover this difference was showed highly significant according to a Mann-Whitney test comparison ($z=-4.1$, $p\text{-value}= 0.000$). This means that the digital platform is supporting better the co-operation of subjects and, therefore, it is advantageous in tasks requiring collaboration as the one performed in the experiment.

From the video recordings, more analysis can be performed. There is a lot of sequential work in the physical-only setting, typically with a member constructing the entity to some extent, then passing it to the other member to continue. Nevertheless an important amount of work is conducted in parallel in the digital platform. Figure 24 shows the creation of a human-like entity in parallel in the digital setting.



Figure 24. Co-creation of a proposal in the digital setting.



Figure 25. Example of non-cooperative work in the physical-only setting.

This is an important result that raises the need to have a look into more cooperative behavior patterns to understand what actually is happening in each platform.

In both platforms the pattern provider-constructor was widely observed. From time to time one member acted as a provider of blocks and joints while the other focused on assembling them. In general, one member played the role of director/leader, and the other assumed a more passive role of follower. Normally the first role corresponded to the person that had designed the proposal at the thinking place. This is explained by the fact that they perceived the proposal as a property despite belonging to the group. However, in the digital platform the follower has a tendency towards being alert to get involved as soon as the other participant needs anything. Although this behavior is also present in the physical-only approach, it is not as common, and participants just keep a block or joint at hand without any particular purpose. It rarely led to the participant testing blocks and joints to finally contribute to the creation.

In fact, it is remarkable that 39.06% of proposals tested in the physical-only setup did not have any concurrent manipulation in cooperation (as in the situation illustrated in Figure 25). This means that in 25 out of 64, a member created the entity without participation of his/her group mate, and normally without asking for help or even rejecting suggestions.

In the digital platform, this individualistic behavior only occurred once. This observation shows that humans have very present their notion of ownership when manipulating tangible elements. In this respect, in the physical-only setting participants very often wanted to help by manipulating what the others were creating and the corresponding reactions were grasping the entity to their territory avoiding interruptions and cooperation. Related to this behavior, the analysis of the conversations revealed that if participants had very clear in mind how to implement the whole proposal, they preferred to work alone.

All these observations can be partially explained from the basic difference between both platforms. Both are tables, supporting co-operation, sharing objects and enabling face-to-face communication, but certainly the digital platform is the one that enforces keeping objects on the sharing space and allowing a more transparent management of territoriality and ownership. Although in the digital setting participants usually considered the nearest area of influence as their territory, common areas still remained available. However the physical-only platform allowed the participants taking the entity under construction with them, to easily reflect human feelings related to ownership, individualism and non-cooperative work.

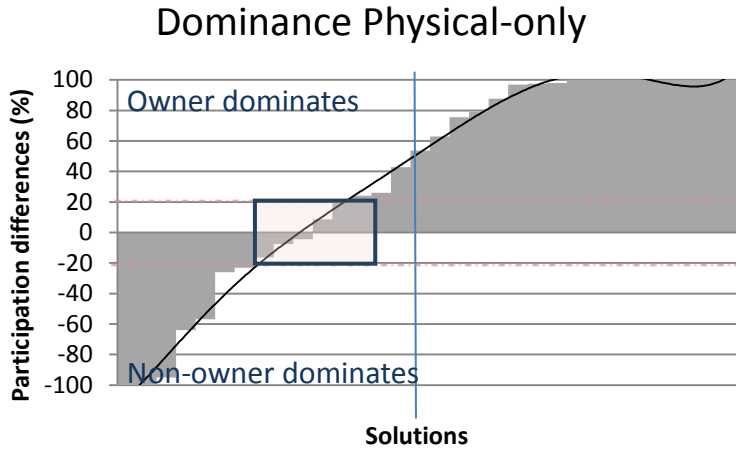


Figure 26. Dominance in Physical-only condition for entity creation task.

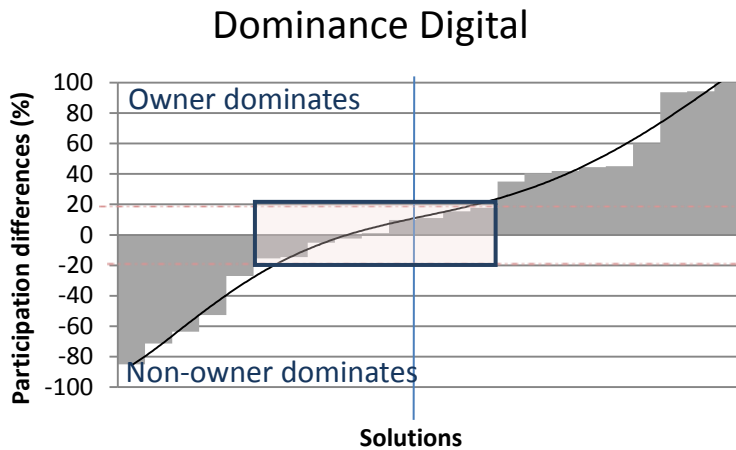


Figure 27. Dominance in Digital for entity creation task.

Another interesting result is the rate of successful testing of previously designed solutions. While all the tested designs in the digital platform were successfully completed, it is remarkable that four entities were not completed in the physical-only approach. From the recordings, the observation is that these unfinished creations were explained by either the complexity of the entity or the difficulty to join blocks as required. Participants tried to join pieces in many different ways, working sometimes sequentially but especially in parallel in two

of these cases. In fact, the work in cooperation reached 95.5% and 57.2% respectively in these cases. Therefore, this is an important evidence of the outstanding increase of the concurrent cooperation when facing the implementation of complex or elaborated ideas. It is observed that the digital platform facilitates concurrent cooperation and composition of the different subparts of the entities by means of the flat sharing space. However, in the physical-only approach cooperation patterns only arise when the complexity of the entity at hand is sufficiently high. Because participants usually take entities to their territory, the co-manipulation becomes difficult and the entity has to be complex and large enough in order to enable each participant to create different parts collaboratively. Nevertheless, this way of cooperation is commonly and easily supported by the digital platform as the recordings have revealed.

Finally since individual thinking-reflection, and collective discussion processes have been considered to foster creativity, and proposals were normally created by a single participant, another interesting point to analyze is how the original author of the proposal finally implemented influenced interaction on the testing platform. Dominance measures the relative differences in the participation between the members in a group creating a solution. A dominance value close to zero means that both members participated equally. However, if the magnitude dominance is about 50, it means that the difference in participation was 50% of the implementation time for that solution. The dominance value is used to indicate whether the most active member is the author of the solution (positive sign), i.e. the one that originally designed it on paper, or not (negative sign). Figure 26 and Figure 27 depict the actual dominance, allowing watching the profile dominance in both platforms and complementing the information about behavior patterns described before. In the physical-only platform, the owner dominated the construction of the solution and moreover the participation differed above +40% for the 50% of the solutions implemented. Clearly, a more balanced participation is being shown in the digital platform. There is an important number of solutions with balanced participation (central values close to zero), and in addition in both directions. This means that the non-owner was able to take the control in the process of implementing the solution.

3.6.3.6 Experiment 2: Rube-Goldberg Machines

In this experiment, we were more interested in exploring a creative task but presenting a clearer objective, as many real problems state. Therefore the creativity would be more in the problem-solving process; on how a solution is given to a very specific problem. The previous experiment resembled more to the idea of creating entities, in those similar conditions that would be expected in

the entity editor in AGORAS. However, the current experiment looks into the process of building a stage in the ecosystem to fulfill a specific goal as a consequence of the physical governance of basic components. With all this in mind, inspiring scenarios are those provided by Rube-Goldberg Machines, which could be good sandboxes for our purpose. In particular, the task for this experiment establishes a simple problem which requires the construction of a machine to solve it.

3.6.3.6.1 Task

Participants were requested to design as many creative RGMs as possible to solve a given problem, consisting of making a box fall from a shelf located in the center of the tabletop. See Figure 28 and Figure 29 for samples of actual machines constructed by participants.



Figure 28. Users creating an RGM in the digital platform.

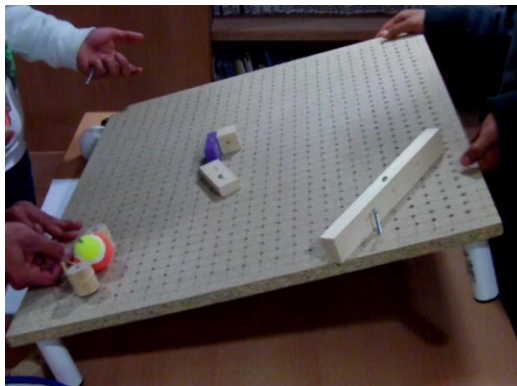


Figure 29. Users testing an RGM in the physical-only platform.

3.6.3.6.2 Results

This time the eleven groups worked with the other platform. It means that six worked with the digitally-augmented platform and five with the physical-only. A total of 122 solutions were proposed of which 38 were tested on the digital platform and 26 on the physical-only.

The creativity model was measured as in the previous experiment. Regarding the fluency of thinking, the groups in the digital platform produced on average almost twice as many solution proposals than in physical (Digital: $m=5.57$, $sd=2.59$; Physical: $m=3.14$, $sd=2.45$). A t-test comparing means showed these differences on fluency are significant ($t(26)=2.55$, $p\text{-value}=0.017$).

On the elaboration, taken again as the complexity in terms of numbers of blocks and joints used in the implementation, solutions were slightly more complex in the physical condition (Digital: $m=7.76$, $sd=3.6$; Physical: $m=8.38$, $sd=4.5$). However, the t-test comparison did not find significant differences.

Regarding motivation, participants using the digital platform actively participated about 63% of the time on average, while in the physical-only platform this ratio was about 52% (Digital: $m=0.6372$, $sd=0.239$; Physical: $m=0.5216$, $sd=0.260$). The corresponding t-test found that participants were significantly more participative when using the digital platform ($t(126)=2.587$, $p\text{-value}=0.011$). In addition, in a user questionnaire, participants rated on a 5-point scale their agreement with the statement “The platform keep me motivated to participate”. This subjective user perception consistently showed that self-rated motivation in the digital platform was on average higher ($m=3.83$, $sd=1.030$) than physical ($m=3.22$, $sd=0.833$).

Novelty was again measured departing from two judge assessments, using the same procedure explained in the other experiment. The corresponding Kappa statistic showed good agreement ($K=0.733$). The average rating for each solution was therefore taken from both judges. On average, the digital solutions were rated a little higher (Digital: $m=2.68$, $sd=1.07$; Physical: $m=2.16$, $sd=0.93$). However the t-test comparison of novelty showed that this difference cannot be considered significant, although it comes close ($t(61)=1.996$, $p\text{-value}=0.05$).

Concerning the collaboration aspect, Figure 30 shows that participants cooperated on average over 38.19% of the implementation time on the digital platform, while on the physical-only it was about 20.21%. The corresponding t-test found that differences are highly significant ($t(62)=3.770$, $p\text{-value}=0.000$).

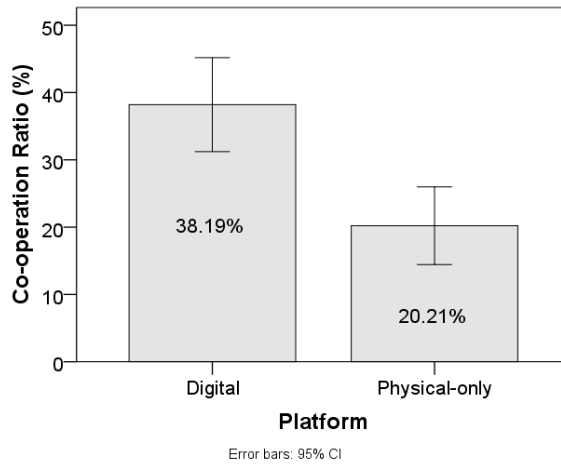


Figure 30. Co-operation ratio means.

Participants had to implement and then simulate to check whether their solutions were working. In this process, they were allowed to make small adjustments to tune up the implemented RGM. This is one of the specific inherent details related to the type of task. The number of trials per solutions was similar in both platforms (around 4) (Digital: $m=3.47$, $sd=2.7$; Physical: $m=4$, $sd=2.6$). However an interesting issue is what happened in those trials and how the platform was used. Having a look at the number of retrials that either involved an actual adjustment of RGM elements or just activated the machine again without any fine adjustment, the digital platform better supported the fine adjustment of the solutions before a retrial was run. In the physical-only, adjustments were hardly introduced ($m=0.46$, $sd=0.706$, $median=0.00$), and trials just involved testing the solution again with the hope of an eventual success. In contrast, the trials in the digital platform were mostly used to make adjustments rather than starting the simulation again ($m=1.95$, $sd=2.092$, $median=1.5$). As normality assumption was not met a Mann-Whitney test was run. This found significant differences ($z=-3.473$, $p\text{-value}=0.001$).

With respect to dominance, Figure 32 and Figure 31 illustrate the relationship between dominance and authorship of solutions in the digital and physical-only platforms respectively. In the physical platform, 18 out of 26 solutions have a positive value of dominance, which means that the author of the proposal also contributed more to implementation. However, in the digital platform, 15 out of 38 solutions have positive value, which show that dominance is more equally balanced between participants, and non-authors also have a fairer participation. Moreover the strength of author-dominance is higher in the physical-only,

since on average the dominance is about 49.9% while in the digital is about 38%.

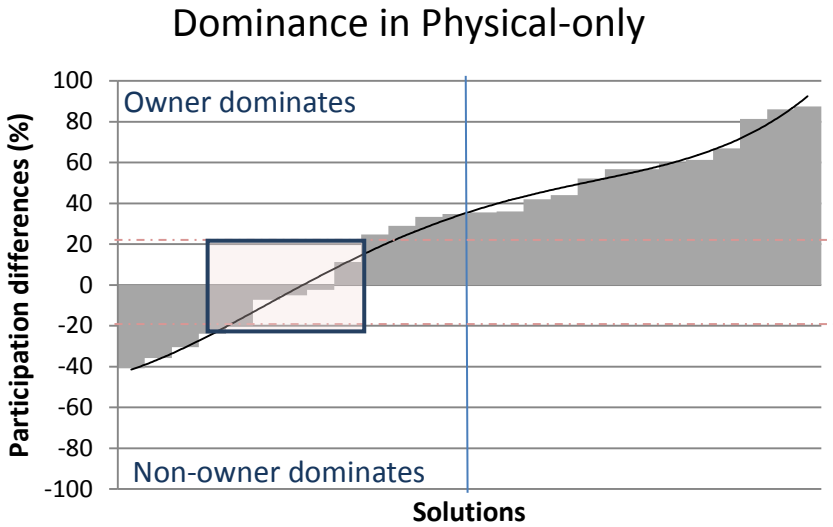


Figure 31. Dominance in the physical-only platform.

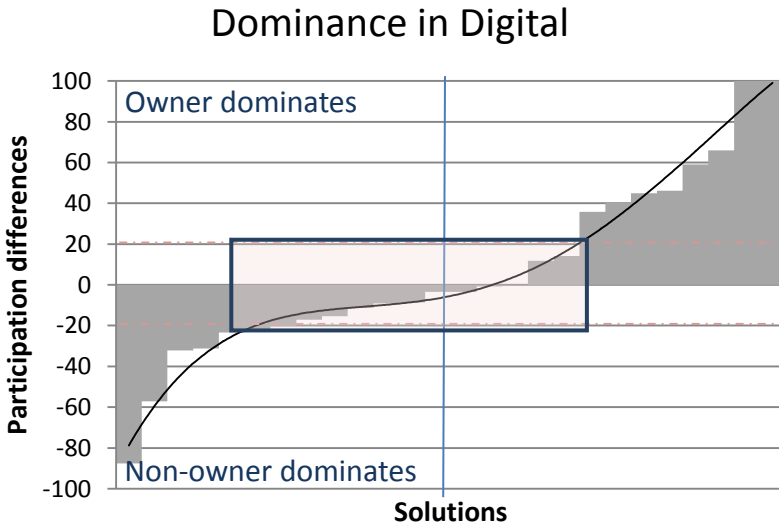


Figure 32. Dominance in the digital platform.

3.6.4 Overall Discussion

The experiments confronted two tabletop platforms in creative tasks. The experimental design and results obtained must be considered in the context of the limitations and constraints being faced. Apart from the number of subjects involved, the first thing to be aware of is that the technology course in which the experiment took place was organized by a club dependent on the Education & Culture section of a local city council and not by a school. This may have several implications. As the origin of participants was not homogeneous and school performance profiles (or similar assessments) were not available, there may still have been differences between groups despite being randomly paired taking ages into account. A way to face this issue would be to obtain this kind of profile information to try to control it by forming groups accordingly. On this assumption, it would be interesting in further work to correlate the information profile with creativity traits in order to explore their relationships and the validation of the operationalized creativity model.

Another issue is the time limit available to run each session, which made us opt for a between subjects design within each task. As both experiments involved different tasks, the data results from the experiments cannot be used as a whole, and the combination of both experiments cannot be taken as a within subject design. Thus, results have been reported separately, being aware that uncontrolled factors regarding subject-characteristics are not all controlled and the number of participants is not especially high. Despite all these issues, there is a chance to try extracting more results and conclusions emerging from the bunch of data available. In this respect, Table 5 summarizes the results from both experiments, indicating also the degree of significance in the comparison (ns=not statistical significant, *= $p < 0.05$, ** = $p < 0.01$).

With this summary at hand, the previous results on comparing the set of creativity traits in both platforms show that using interactive tabletops can have a positive effect on creativity, if more relevance is given to novelty and motivation. Firstly, according to the estimated motivation, we can establish that the digital platform motivated users more than using the non-technological approach. As this trait is directly based on user interaction times, it could also indicate a better facilitation of participation in the digital platform, despite both being tabletops. In the particular task of RGMs, motivation in the digital platform is significantly higher.

	Entity Creation			RGMs		
	Digital	Physical-only	Sig.	Digital	Physical-only	Sig.
Solutions per Group	5	11	-	6	5	-
Fluency	3.4	7.1	*	5.57	3.14	*
Elaboration	11.96	7.38	**	7.76	8.38	ns
Motivation	65.94%	58.93%	ns	63%	52%	*
Novelty	3.5	2.78	*	2.68	2.16	ns, p=0.05
Co-operation	37.7%	20.15%	**	38.19 %	20.21%	**
Dominance	Balanced participation promotion	Owner dominance	-	Balanced participation promotion	Owner dominance	-
Re-trials with modif.	n/a	n/a	-	mean=1.95 median=1.5	mean=0.46 median=0	**

Table 5. Creativity Experiments Summary

Secondly, as the thinking-reflection as well as the collective discussion places are not computer mediated, and the testing platforms are based on tabletops, it could be reasonably expected that measures for novelty, fluency of thinking and elaboration are similar in both platforms with corresponding parts. This has been the case for elaboration in the RGMs task, but not for fluency and nor even novelty. With respect to fluency in the RGMs task, the experimental results show that participants using the digital platform produced a higher number of proposals on paper than the ones using the physical-only. While in the Entity task the differences went in the opposite way. This suggests different explanations. The first one, taking an optimistic point of view, suggests that the suitability of a platform, or what the platform can supply, depends on the type of task. The second explanation is about the information we have and especially the one we do not. Probably, there are other uncontrolled factors that explain such dissimilarities related to the subjects' skills (e.g. intelligence, flexibility, etc.). With the experiment design, at least we were able to detect this. Thus, further research is needed to clarify this issue, by considering a within-subject design and even being focused on studying the possible factors that

could explain fluency. Related to the fluency, it would have been interesting to study the flexibility of thinking. It is more about quality rather than quantity. However, the information in the form of sketches was not enough to conduct such an analysis. Maybe the combination of fluency and flexibility, along with school performance profiles, would have provided more useful and valuable information to draw relevant conclusions. For the novelty and motivation, there was a tendency in favor of the digital platform, although significant differences depended on the tasks.

Thirdly, co-operation measurement also gives us an idea of how the platform facilitates cooperation. Although interaction on both platforms is natural and tabletop based, manipulation still has differences. On the physical-only platform the elements can be physically grasped and easily moved to the personal working space. In addition, interaction avoidance is possible by physically pushing away a partner's hand, as seen in the video recordings. In the digital, the phenomena of territoriality (*Scott, 2004*) and interaction avoidance or interference (*Hornecker, 2008*) are also observed but are not as stressed as in the physical platform. They seem to be better managed, as blocks remain on the surface level and are therefore reachable by two participants, also facilitating equitable object sharing. However, as co-operation time is partly based on participation time, and participants interacted more in the digital platform, it is more likely that co-operation happens then. It is therefore difficult to establish to what extent the effect on the co-operation ratio is due to the suitability of the digital tabletop and how much is due to the higher motivation since both are linked. In this respect, further experiments would be needed to precisely clarify this relationship.

Regarding dominance in the digital, the number of dominant non-author is more balanced, which suggests that the digital platform based on an interactive surface in combination with a discussion process facilitates the sharing and manipulation of objects by the non-author member. The existence of dominance in favor of non-authors in both platforms indicates that dedicating time to the discussion process, including explanation of solutions to the other partner, is useful and advantageous in terms of promoting collaborative interaction. Finally, the case of the RGMs task requires tuning different parts to achieve a goal by running the whole as a system. The simplifications to the real world that the digital platform offers as well as its reproducibility seem to benefit re-trials being done with a specific simulation goal in mind.

3.7 Conclusions

Departing from the basis that tabletop technology could foster creativity and interaction and the ideas of Abt about learning through gameplay creation, this chapter has provided a basic meta-model to support the processes of creation and simulation of virtual 2D ecosystems. With the aim of preparing an empirical study, an editor of structures has been implemented running in an interactive surface. The empirical study was conducted with the aim of exploring whether an interactive surface as base technology for collaborative creative tasks is promising in terms of both collaboration and creativity traits. In the experiment, the digital editor and a physical-only tabletop platform were involved in an experimental loop based on individual thinking-reflecting, collective discussion and testing processes to foster divergent thinking and idea generation to solve problems such as creating *entities* and *Rube-Goldberg machines*. The physical-only platform relied on a conglomerate tabletop and a toolbox with wooden blocks and connecting elements. The choice of using a pure tangible platform instead of one based on a desktop application was made on the assumption that, firstly, this non-technological platform is similar to some construction kits that are widely used during childhood, and secondly, it is better to have two similar platforms in terms of co-located user involvement and participation possibilities.

In terms of creativity traits, interactive surfaces seem promising, as groups working on the digital platform showed better performance in novelty and motivation. Other issues related to collaboration and interaction were also analyzed, including co-operation, retrieval fine adjustment and dominance, which showed that the properties of an interactive surface tabletop are better suited to facilitating the sharing of objects and participation in conditions of co-operation by co-located participants. With all these results, digital tabletop platforms can be considered suitable to build systems for collaborative construction tasks as AGORAS will require.

Currently, the stage size is limited to the dimensions of the tabletop. A future work would consist of allowing larger stages, so that more complex stages and games could be supported. Moreover, at the moment an ecosystem does not consider the definition of background decoration or backdrop. In the future, it would be highly interesting the expansion of the stage with this concept. In this way, the creation of the backdrop could be considered as a creative task as users would have to draw with their fingers and other tangible tools.

Chapter 4

A Model of Reactive Entities Based on DataFlow Enhanced Rules

In addition to the physical characteristics of entities, there is an important part concerning their behavior in terms of reactive choreographies that has been addressed here by means of rule specifications. This chapter describes the details of the rule model adopted. It relies on visual dataflows, intended as assignment expressions embedded in the rules as a formalism to facilitate the use of natural interaction techniques. Before the design and implementation of a user interface prototype, an empirical study has been conducted on the comprehension of dataflows versus textual expressions. Several requirements have been considered in the designed editor such as support for the construction of general expressions and possibility for co-operation and natural interaction.

4.1 Introduction

Those born in the digital age have grown up with digital information and are familiar with a variety of computer applications, many of which primarily consist of educational interactive games and videogames. This higher exposure of young to digital media and computer applications, especially those focused on the creation of some sort of digital expression, have facilitated the introduction of complex computational concepts and to some extent some sort of programmability as in Scratch (*Maloney, 2004*). This is important for the development of computational thinking, which is considered a fundamental skill, since it leverages powerful abstractions and tools to solve meaningful problems in many different disciplines (*Wing, 2006*). Despite the higher digital literacy and an early training, programming is still difficult for beginners and/or non-programmers, although possibly made easier by the environments and languages specifically designed for novice programmers.

According to Kelleher and Paush (*Kelleher, 2005*), not only textual but also visual languages have significantly contributed to lowering barriers to programming, although most of these are based on desktop settings, which determine user interaction, how users face these systems and how they learn and put these computational concepts into practice. As shown in Chapter 2, there is diversity in the way behavior specification is supported and the technological approach used. There have been a variety of game-based environments with some sort

of end-user programming language to express the behavior of their virtual entities/characters or the evolution of the whole virtual world, using different game technologies, goals, target users, language type, expressivity, etc. One of the most outstanding environments is Alice (*Kelleber, 2007b*), which has interfaced a highly expressive language by using a structured programming paradigm based on drag&drop and dropdown list controls. This has shown that non-programmers are able to program quite large chunks of code with this method. Scratch (*Maloney, 2004*) is focused on creative production of media and is inspired by the visual language based on virtual construction blocks by LogoBlocks (*Begel, 1996*). Although this makes it more visual, the expressiveness is a bit more limited. However, it still provides construction blocks for control flow structures besides data flow expressions. In the context of simulation, a relevant work is Agentsheets (*Repenning, 2000*), which uses a rule-based language to specify agent behaviors. All the above environments are desktop-based systems relying on well-known basic input methods such as mouse and keyboard. This has probably facilitated their development and given more suitable support for languages with high expressiveness targeted at non-programmers.

Nevertheless, advances in the HCI research field have produced new forms of interaction that have open new paradigms in interaction and learning computational concepts by incorporating natural interaction as well as embedded and tangible objects. In this respect, some works are of interest because they try to expand new horizons although they do not necessarily support rules. AlgoBlocks is a tangible programming language that pursues fostering and facilitating interactions among learners (*Suzuki, 1995*), which are difficult to achieve in desktop-based systems. Similarly, Tern is a tangible programming language that considers tangible interaction as a suitable alternative to textual and even visual languages for classroom use, where collaboration and cooperation are of interest (*Horn, 2007*). Turtan is a tangible programming language that has successfully combined the main Logo concepts with the interaction mechanisms offered by interactive tabletops (*Gallardo, 2008*).

Some environments have contextualized programming capability by expressing entity behavior, such as in (*Kelleber, 2007b*), (*Repenning, 2000b*). In the context of this dissertation, a generic platform for creating and playing 2D physically-based entity games fostering computational thinking in teenagers on an interactive surface is proposed. We are interested in enabling expressive but approachable rule based specification for virtual entities by means of a rule editor specifically designed for a tangible surface interface. This is promising for learning since it combines aspects such as experiential collaboration, co-located

```

IF S: _____ THROWS an EVENT E: _____
[AND _____]
THEN WITH T: _____
[SO THAT _____]
PERFORM O({DF}): _____

```

Figure 33. Rule structure.

cooperation, experiential learning and face-to-face communication (*Hornecker, 2006*), aspects which had not been considered in previous approaches except for some concrete cases such as in (*Gallardo, 2008*).

4.2 Rule-based Behavior

Many current applications and systems targeted at non-programmers are often enhanced by programming. As surveyed in (*Kelleber, 2005*), systems whose primary goal is not to teach programming but aim at another goal by means of programming are mostly event-based. In one way or another, events and rules are the basic components needed to express behavior in these systems. Just because they are targeted at non-programmers, designs have focused on trying to create languages, programming methods and interaction mechanisms that allow people to build those programs. An event-based approach has also been adopted in the present work, since rules have been traditionally used in reactive environments (*López de Ipiña, 2001*), and it seemed better suited to the expression of reactive behavior in virtual worlds, especially if rule structures are better understood and used by young people or non-programmers, as reported in (*Pane, 2001*)(*Good, 2010*). A rule in our approach is formally defined as an ordered pair $R = \langle P, Q \rangle$, where P is the antecedent and Q is the consequent. The antecedent is defined as $P = (E, S, C)$ where E is the event type or definition that must be thrown by a source S (e.g. an entity), and C is the condition that must hold involving data (i.e. properties and attributes) from E and S . The consequent is defined as $Q = (T, F, O, \{DP\})$ so that T is the target population, F is a condition that filters the entities within the target population to be affected by the outcome operation, O is the service operation on every target entity, and $\{DP\}$ is a set of data processes that specify how the operation parameters are established prior to execution.

Operation O is a service invocation consisting of either the execution of an action or the assignment of a value to a property on the target. Consequently, a

property assignment entails just a single data process in the set $\{DP\}$, which would specify how to compute the value to be established on the property. Similarly, if the operation is an invoked action then a variable set of data process structures would be needed, one for each action parameter.

The semantics is as follows: if an entity conforming to S throws an event occurrence of type E , and C holds, then the rule is triggered and instantiated. Operation O will be executed on those entities that belong to the target population T and meet the filter condition F . The operation parameters are established according to the assignment expression specifications $\{DP\}$. The structure of the formalized rule is as shown in Figure 33. For illustration purposes, we used a simple written English syntax at this point in the figures to clarify the conceptual structure of a rule. We will later elaborate on the visual and textual languages that support the proposed rule model.

In our proposal, data processes are basically assignment expressions. In this way, the C and F conditions are also considered assignments as data processes expected to produce a Boolean outcome that one of the rule conditions must take. The language for these expressions embedded in the rule must at least support assignment, arithmetic and logical operators with the semantics of general-purpose languages. A grammar describing a textual language with this expressiveness may be defined as follows:

```
Instr := Variable '<-' Expr
Operand := Variable | Constant | Expr | Func
Operand_par := '(' Operand ')' | Operand
Operator := '+' | '-' | 'x' | 'AND' | 'OR' | 'NOT' | '>' | '<' | '='
Expr := Operand_par [ Operator Operand_par ]
Func := Function_name '(' Params ')'
Params := NIL | '(' Expr [, Expr]* ')'
Function_name := 'MAX' | 'MIN' | 'ABS' | ...
```

This grammar only includes a typical core set of functions, but because the implementation of the rule proposal completely relies on a meta-model, many additional functions like `DISTANCE` and `NEAR_AT` could be supported, so that extensibility is easy in this respect. Figure 34 gives two examples of rules that can be defined in an AGORAS ecosystem. The first example refers to a rule which would increase the potion counter in the inventory of the entity called “ogre1” whenever it finds a potion item. The second example is slightly more complex as it involves source and target populations. First, the source in the precondition responds to any tumoral cell that changes its position. The

consequent specifies that all the cells in the neighborhood, indicated by means of a filter, will see its health decreased.

```
IF S: ogre1 THROWS an EVENT E: Item_Found
AND E.Item = POTION
THEN WITH T: ogre1
PERFORM O: potion_amount ← ogre1.potion_amount + 1
```

```
IF S: cell_class THROWS an EVENT E: Position_Changed
AND S.cell_type = Tumor
THEN WITH T: cell_class
SO THAT Distance(T.position, S.position) < 2
PERFORM O: T.health ← T.health - S.health
```

Figure 34. Example of rules.

Figure 35 shows the UML class diagram supporting the formalized rule conceptual model. Firstly, a reactive rule consists of an *EventType*, a *TargetPopulation* and a *SourcePopulation*. Additionally, a precondition expression (*PreconditionDataProcess*) and a filter condition (*FilterPreconditionDataProcess*) are respectively defined to establish the conditions that must hold to activate the rule and determine which target instances the consequent must be applied to. Finally, the operation-binding (*OperationBinding*) defines the operation that must be applied to the elements in the *TargetPopulation*, either a property assignment or an action invocation, and how the operation parameters must be computed by means of assignment expressions which are represented as data processes (*BindingDataProcess*).

Our main concern is considering the specification of the event source as well as the target to be either a specific entity in the ecosystem (*EntitySourcePopulation* and *EntityTargetPopulation*) or a class of entities. In other words, indicating a particular entity (e.g. thermostat in bedroom), or perhaps a type of entity that will represent all the entities instantiated from it that are present and operational in the environment (e.g. all thermostats). The significance of representing source and/or target populations by means of entity-types (*EntityTypeSourcePopulation* and *EntityTypeTargetPopulation*) must therefore be kept in mind. If the source is defined as an entity-type, the rule will be instantiated if any entity of the type causes the expected event. On the other hand, if the rule target population is specified as an entity-type, the rule will be instantiated for

every entity in the target population. This mechanism has advantages and allows the behavior specification to be simplified when the exact entity is not known in advance or when there are various entities of the same type that must behave in exactly the same way (e.g. if all lights must be turned on or all blinds lowered in a room).

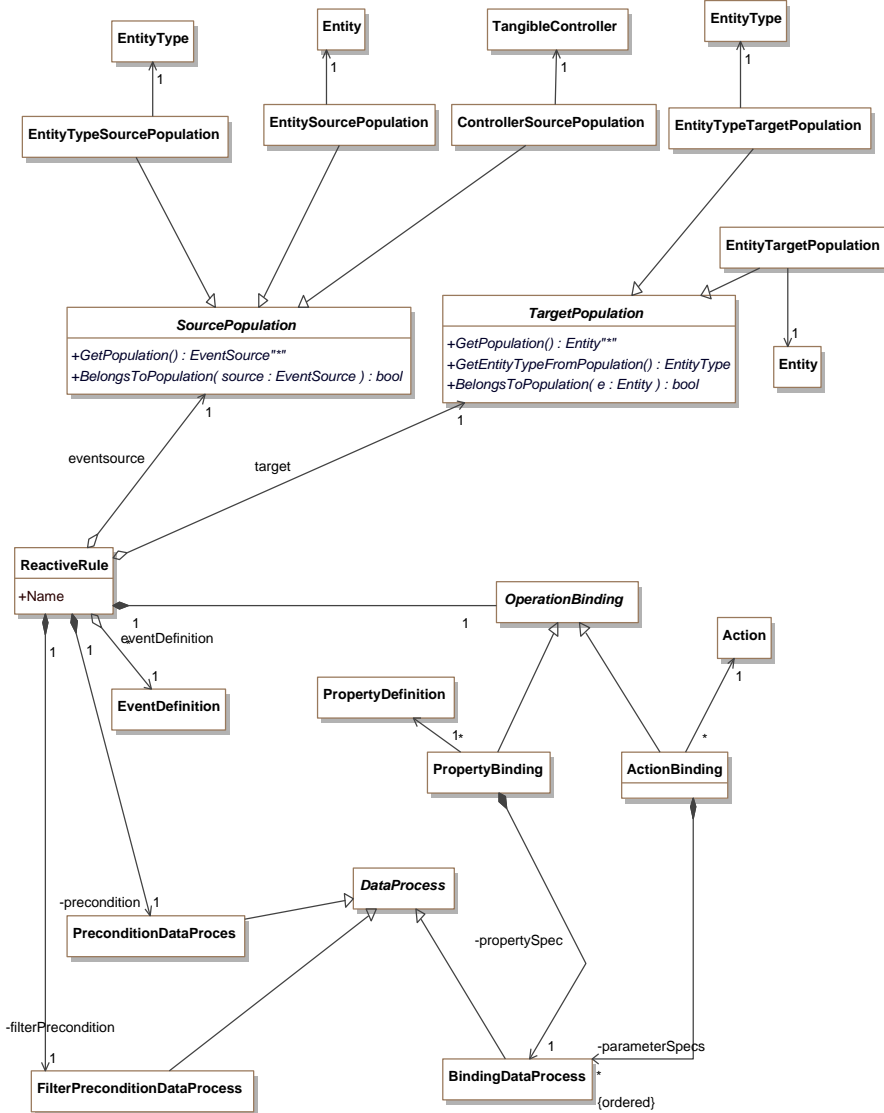


Figure 35. Model for the Reactive Rule.

In other systems discussed in Chapter 2, rules are specified in a simpler way, for example by preventing any data transformation to be applied before being established as operation parameters, thus limiting expressiveness. Instead, our model uses data processes to define transformation operations on the available data (event attributes and properties of source entities among others), thus obtaining much more elaborated constructions. A mechanism is also provided to facilitate the involvement of the properties of any entity present in the ecosystem, with the aim of supporting rules with unlimited expressiveness. However, despite this expressive power, it is still possible to define simpler rules, such as those described in related studies, having no data processes at all in which the consequent of the rule uses values from the antecedent without performing any data transformation. The expressive capacity of our rule model is quite different to those that contain a type of visual-based programming mechanism. However, as it will be discussed in the next sections, we put great emphasis on providing simple visual and intuitive mechanisms without limiting the expressiveness required for the proposed rule definition language. Our proposal is based on combining the simplicity of rule structures with the enormous expressiveness offered by data process structures in the form of dataflows, so that the resulting language is more graphically powerful and usable.

4.3 A Generic DataFlow Model

The rule model presented earlier enables a set of assignment expressions to be present in the rule specification. These expressions were generically represented in the meta-model by the *DataProcess* class (see Figure 35). However, nothing was discussed with regard to the actual syntactic representation of processes which could be represented in textual form, as in some previous examples. We are particularly interested in providing a rule editor that manages assignment expressions in a natural way and avoids writing code in the usual text-based form. This means devising an affordable mechanism to make programming easier, with a representation that facilitates the construction of a suitable editing tool. Although assignment expressions can be captured by different models (e.g. one based on operands and operators), the model considered is related to an explicit visual representation based on dataflows.

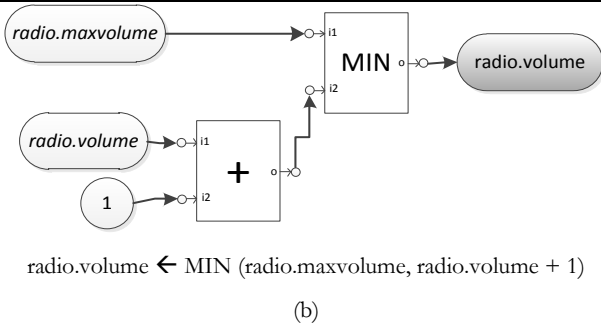
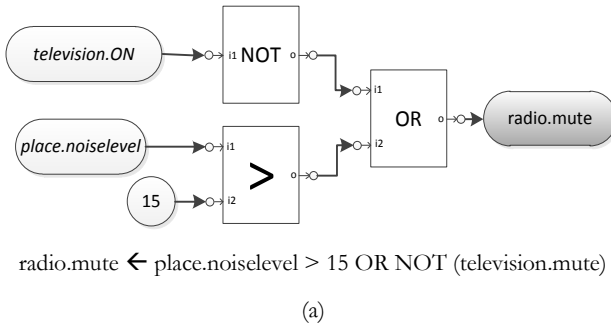


Figure 36. Sample expressions.

Figure 36 and Figure 37 show the type of expressions that can be supported. The expressions in the textual language look like many others used in computing and mathematics in general. The ones in the visual language have a tree-like organization, where the target variable to be assigned is on the right in grey, the operators are represented by boxes with inputs and outputs. The variables and constants involved in the process are conveniently connected by means of flows to the operators. The power of a visual paradigm relies on the fact that users can manipulate components as independent pieces, and simply join them without complicated interface controls.

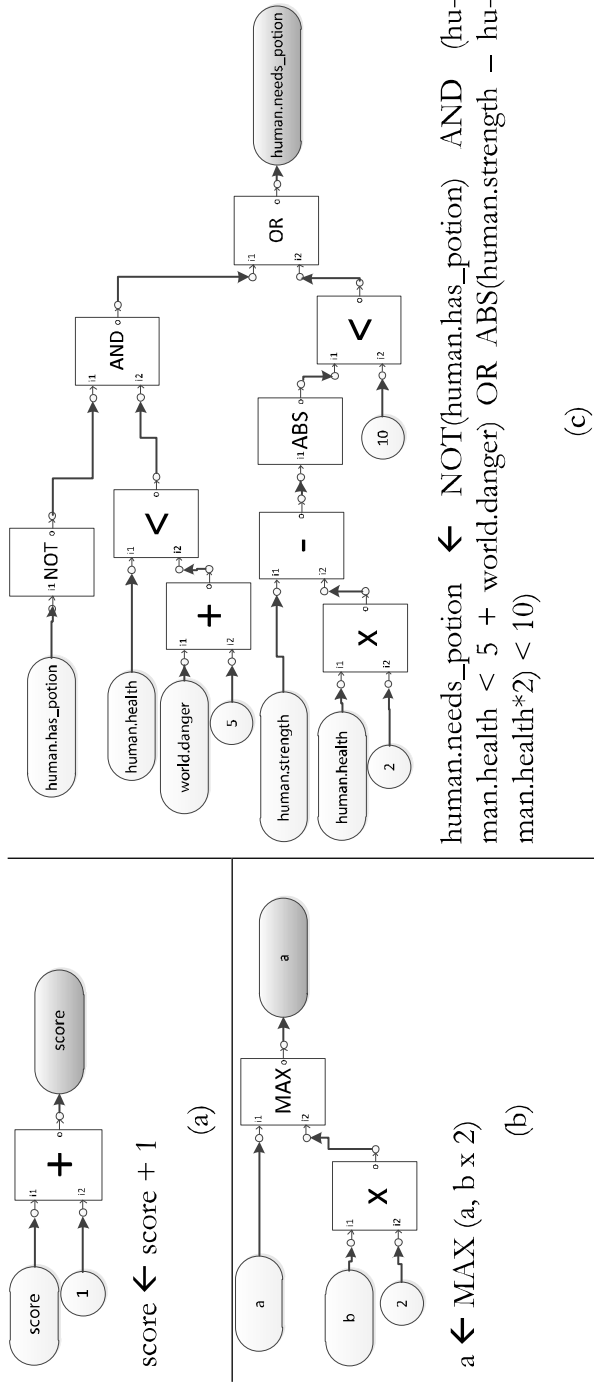


Figure 37. Expression samples. (a) and (b) are basic examples simply illustrating expressiveness. (c) illustrates an expression like the ones used in the worksheet of the empirical study.

The meta-model supporting the definition of data processes is illustrated in Figure 38, in terms of processors and flows. To represent a dataflow-based expression, the model follows a graph-like approach, so that a data process (*DataProcess*) consists of a set of nodes (*DataProcessNode*) and a set of dataflow connection node *Ports*. The nodes are specialized in a range of different supported node types, to essentially represent operators, value constants, or variables, and these nodes can be connected to others by means of dataflows (*DataFlow*) to form a complete expression. Our model contains an important type of node represented by the *DataProcessorInstance* class and allows the inclusion of operators in a flow, depicted as boxes in the figure samples. The definition of the specific operators available in the system is provided by the *DataProcessor* class, consisting of an operator implementation and a set of typed port definitions that will provide input parameters and an output result. As mentioned earlier, this is an important feature, since this is the way the meta-model supports the extension of the language with new and valuable operators as needed.

Besides the operators, there are other elements such as constants and variables which are also important in the definition of a data process. In this respect, the *ConstantProvider* node is an input node that provides the source of a pre-established value to be used in the data process (e.g. constant value “5”). This is to be directly connected to input ports. Variables can also be provided by three different types of nodes. If the value must be taken from a source’s property or an event type’s attribute involved in the rule, then the type of node is either a *PropertyDefinitionSource* or an *EventAttributeDefinitionSource*. Should it be required to take the value from a specific property of an entity other than the source or target, the *ClosedPropertyDefinitionSource* type node will facilitate such a specification. These different considerations with respect to the source element providing values to the variable are needed to support bound and free variables respectively. Essentially, the *PropertyDefinitionSource* nodes give access to properties as variables bound to the instantiation of S when triggering the rule. Alternatively, the *ClosedPropertyDefinitionSource* nodes allow the inclusion of property values from any entity as free variables.

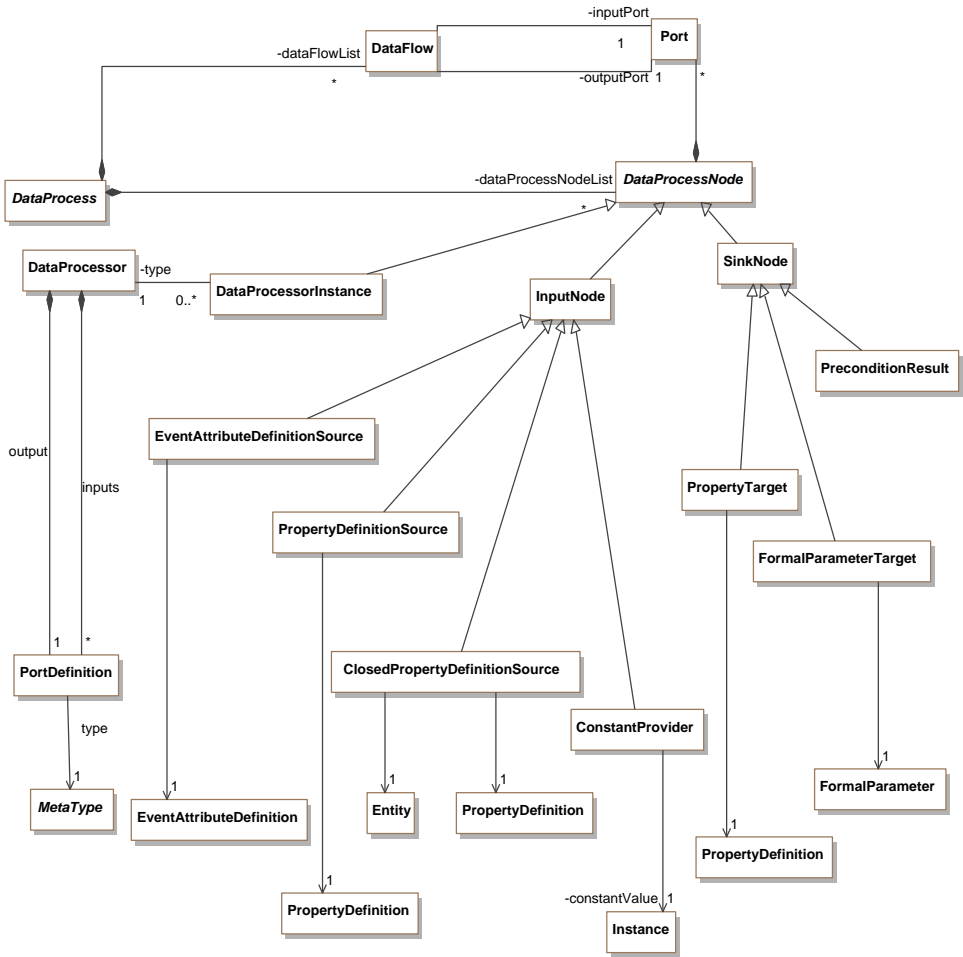


Figure 38. UML class diagram to support visual dataflows.

Finally, there is a set of possible sink nodes that can be used according to the type of the assignment expression that is being created. A sink node represents the final node in which all the dataprocess computation converges and each expression must have one and only one. This node implicitly represents the assignment operation in the dataprocess. For instance, when a precondition expression is being expressed, the sink node will be the *PreconditionResult* type. Alternatively, if the expression deals with an assignment to a target’s property or target’s action parameter, the sink node will be either the *PropertyTarget* or *FormalParameterTarget* type.

4.4 User based Study on DataFlow Comprehension

As mentioned earlier, one of the key elements for the specification of reactive behavior is the definition of data transformations by means of dataflow expressions. As end-users defining reactive behaviors in ecosystem environments with our rule model will have to effectively define these data transformation flows, it is important to evaluate the ease of understanding of the proposed visual dataflow language. However, unlike other studies that have done this on specific tools, using complex approximations including bifurcations and iterations, aimed at software engineering students with some experience in the subject (*Whitley, 2006*), our study focused on basic structures suited to young non-programmers. The objective of the study was therefore to evaluate the ease of understanding, creation and evolution of expressions and ease of use of the system by this user segment. The comparison between our proposed visual approach and a textual language with a similar level of expressiveness was deliberately chosen because subjects are used to perceiving expressions such as those introduced in the previous examples in textual form in scientific subjects like Mathematics, Physics and Computer Science. In theory, this makes the comparison less favorable for the visual approach, but, as the experimental results will show, the visual properties of the proposed language have advantages over the textual language under certain conditions.

4.4.1 Participants

The participants were 36 fifth-year secondary school students (25 boys, 11 girls) who were taking a course in Computer Science as part of the curriculum. Most of them were 16 years old ($M=16.39$, $SD=0.494$) and were going to face real computational concepts for the first time. None reported prior programming experience except for basic spreadsheet formula definitions. Figure 39 shows the frequencies of the agreement degree on the participants liking for a subject and their perceived performance in that subject. The students in general showed a real interest in Computer Science. A deeper analysis reveals that nineteen came from the Humanities stream whereas the rest came from the Sciences stream. The average agreement (1=strongly disagree, 5= strongly agree) by curriculum streams on those demographic questions is plotted in Figure 40. Those students coming from Humanities do like less mathematics and they consider themselves to perform worse than the students from the Sciences stream. Therefore, a reasonable expectation would be that students from Humanities could decrease the overall performance in exercises like those considered in the current study.

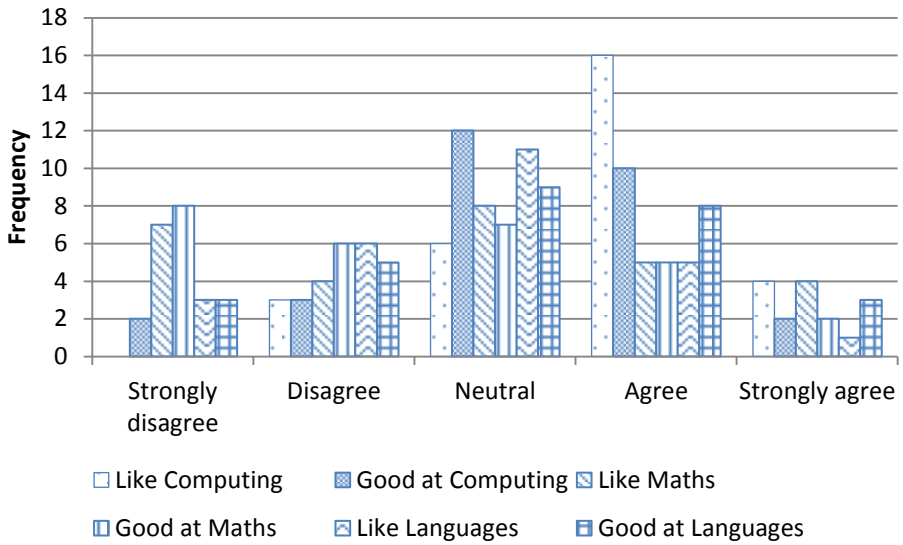


Figure 39. Demographic data on the preference and perceived performance in three main areas.

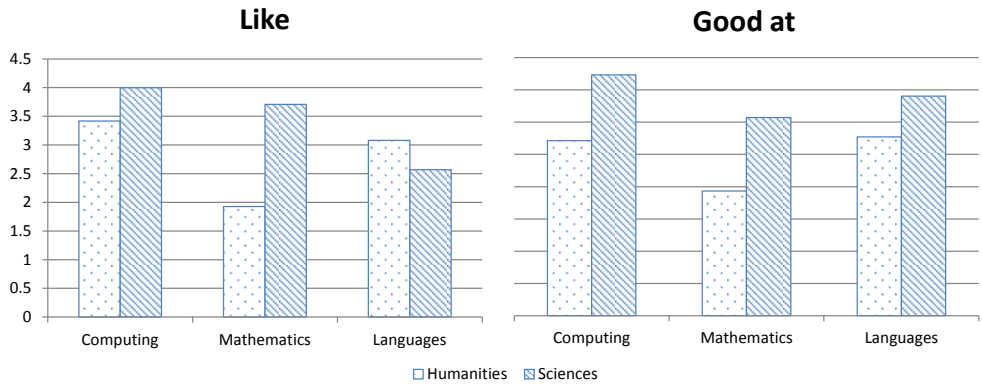


Figure 40. Average agreement for preference (like) and performance (good at) questions by curriculum.

4.4.2 Materials

Teaching materials were created and adapted to the objectives of the course (i.e. introduction to programming). The class teacher devoted four sessions to the necessary basic concepts (variables, operators, assignment instructions, etc.) using the textual language of the grammar described above and the equivalent visual dataflow language, as shown in Figure 36. Both approximations were given the same importance and were accompanied by examples. Some sample exercises were carried out as a team.

The worksheet for the test was designed to explore the different aspects involved and to take students to the maximum skill level. Four exercise categories were considered. The Compute (C) category, typically presented an expression written in either the textual or visual language taught in the classroom. The subject was required to compute the outcome of the expression, given the values taken by the variables present in the expression. This category was considered because it is an abstract mental process performed when working with expressions. When building and checking whether an expression is viable, people assign values to the variables and mentally test whether the computation matches the expected result. An example of a question in this category is shown in Figure 41-(a). The representation in our visual language is illustrated in Figure 41-(b).

Assuming the following attribute values:

human1.health equals 7,
world.dangerousness equals 3,
ogre1.has_potion equals No!,
ogre1.health equals 5,

Calculate the result of the following expression:

ogre1.needs_potion \leftarrow (*human1.health* > 6 + *world.dangerousness*) OR 18 > ABS (*ogre1.health* – *human1.health* x 3) AND INV (*ogre1.has_potion*)”

(a)

Assuming the following attribute values:

human1.health equals 8,
human1.has_potion equals Yes!,
world.dangerousness equals 4,
ogre1.health equals 5.

Calculate the result of the following expression:

[It is the one in Figure 37(c)]

(b)

Figure 41. Sample exercises (I).

Another category was Modify (M). This consisted of facing the subject with an expression in either textual or visual form. Given a statement written in natural language, the participant was then required to detect and make the necessary changes to the original expression so that the proposed statement would be correctly specified. Since natural language is prone to ambiguity, statements were carefully written with a single interpretation. This category was included in the study because rules are usually subject to modifications until they work as expected. We therefore consider that for this lifecycle, it is important to be able to understand and “repair” a previously given version of an expression.

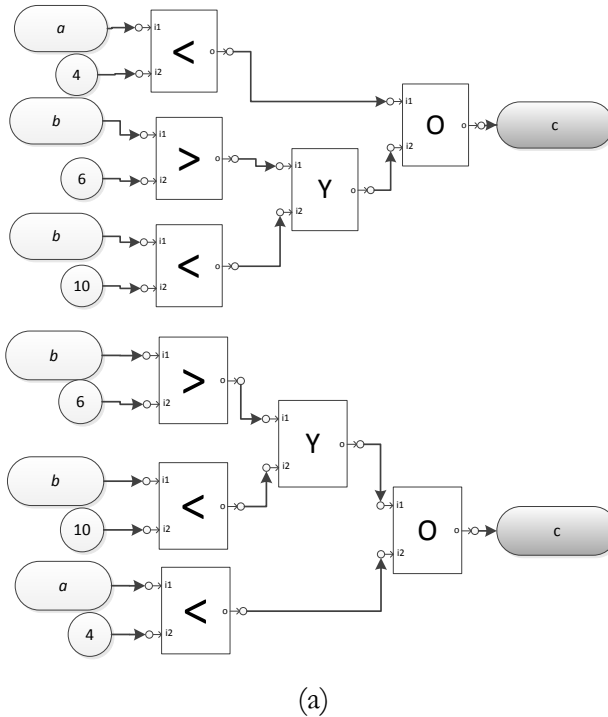
The Equivalence category (E), presented two expressions, both written in the same textual or visual language. The subject was required to indicate whether both were equivalent or not. This category is not only important during the lifecycle of expressions to detect mistakes and errors but also to evaluate the user’s understanding of each language. Comparing two expressions requires more abstract reasoning, since values for testing expressions are not provided. An example of this type of question in our experiment is shown in Figure 42 (a) and (b).

Finally, we included a Write (W) category in the study. The subject was required to write from scratch, in a specific language, an expression that matches a statement given in natural language. Again the statement was ensured to be non-ambiguous.

An example of a question in this category: “Write the following expression: z is the minimum between 10 and the maximum between a and the addition of b and c ”. The solutions to this question in the textual and visual languages are respectively given by the expressions in Figure 42(c).

The worksheet included sixteen exercises, four of each category considered above. Each exercise in one language had a corresponding exercise in the other, keeping the same expressional structure in order to ensure students were faced with expressions of similar difficulty in both languages. Four versions of the test were prepared. All of them had the same exercises but in a different order. Any two exercises from the same pair were not allowed to be contiguous and the categories were distributed thorough the test. Appendix B contains the model worksheet used in the study.

Given the following expressions, indicate whether they are equivalent or not:



Given the following expressions, indicate whether they are equivalent or not:

$c \leftarrow a < 3 \text{ OR } b > 5 \text{ AND } b < 12$

$c \leftarrow b < 12 \text{ AND } b > 5 \text{ OR } a < 3$

(b)

$Z \leftarrow \text{MIN}(10, \text{MAX}(a, b+c))$

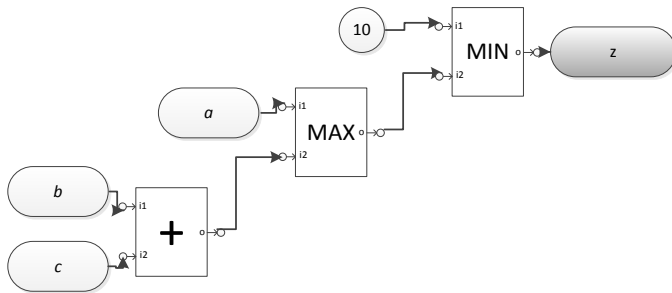


Figure 42. Samples exercises (II).

4.4.3 Method and Procedure

Both textual and visual languages were used to introduce primary concepts on programming in four one-hour introductory lectures. The test was administered to the students after these teaching sessions. Worksheet versions were randomly distributed in equal numbers.

4.4.4 Task

Participants were provided with a worksheet in the test session. They were asked to answer as many exercises as possible in the available time. After performing the exercises the subjects answered the several Likert-scaled questions on their subjective perception of each language regarding several factors such as understandability, comprehension and learning effectiveness and some choice questions concerning which language version, either textual or visual, the participant would prefer in each situation cited in the questions.

4.4.5 Results and Discussion

The students were given a test containing exercises on the textual and visual languages they had seen in the teaching lessons. Table 6 shows the cross table for answers by languages and category. Due to the high difficulty and the number of exercises to be addressed in such a limited time without any advice to study the contents before the test session, we expected some unsolved exercises. There were 73 exercises in the textual language while 85 in the visual, but a Binomial test showed that the proportion of unsolved exercises in each language does not significantly differ from the hypothesized value of 50% (p -value=0.382). Overall 158 exercises remained unsolved out of a total of 576, resulting in 4.3 unsolved exercises per participant on average. It is interesting to observe that the values of the mean and the mode for the number of unsolved exercises are higher in the students from the Humanities stream (mean=4.94, mode=5) than those from Sciences (mean=3.64, mode=3) as shown in Figure 43.

This is an expected result because computing is mathematics after all, and therefore the worksheet required mathematical skills, which in the students from Humanities seemed to be less developed according to their answers to the demographic questions. Moreover, a deeper analysis of the distribution of the unsolved exercises per task reveals that (see Table 6) most unsolved questions were present in the Compute (52%) and Write (34%) tasks and that, in these cases, there are no significant differences (Binomial tests for C and W resulted in 0.728 and 0.392 respectively) in the number of unsolved exercises between the visual and textual languages. This means that the reason for leav-

ing some exercises unsolved is not determined by the language but by the perceived difficulty of the tasks at hand given the limited available time to complete the experiment. In fact, this is confirmed by observing the subjective perceived difficulty of each task by the subjects (see column Overall for CQx questions in Table 7) who considered the Compute and Write tasks as the most difficult ones. All this suggests that the students tried to answer those exercises that considered more affordable given the limited time.

The answers to the exercises that were solved can be correct or incorrect. Unless the unsolved exercises, solved exercises did get an answer because students actually faced the given problem and made an effort trying to obtain the right solution, although they did not always succeed. Departing from these exercises we can assess the potential of the visual language in different tasks as the ones involved in the exercise categories.

Figure 44 gives the percentage of answer outcomes by category, arranged according to difficulty (Anderson, 2000). In the exercises with correct answers, the students performed better using the visual language (V) in Category C and M, while the textual language (T) gave better results in Category W. This tendency in favor of the textual language when writing expressions must be a consequence of the conditions of the experimental apparatus. We have to take into account that the visual language requires some sort of spatial arrangement of the elements and there was no tool support for this to help in the construction process, while in the textual condition requires writing expressions resembling the way the participants usually do (from left to right). Nevertheless, only the comparison in Category C was found to be highly significant, while the comparison in Category M was of low significance, as shown by the corresponding Chi-squared tests for homogeneity of proportions. This means that the use of a certain language has an effect on the proportion of correct answers in categories M and C (indicated in the plot by * and ** for significant and highly significant levels, respectively).

	Compute			Equivalence			Modify			Write		
	Inc.	Corr.	Uns.	Inc.	Corr.	Uns.	Inc.	Corr.	Uns.	Inc.	Corr.	Uns.
Visual	16	21	35	10	59	3	15	38	19	32	12	28
%Lang	22.2	29.2	48.6	13.9	81.9	4.2	20.8	52.8	26.4	44.4	16.7	38.9
Textual	28	4	40	7	59	6	31	35	6	32	19	21
% Lang	38.9	5.6	55.6	9.7	81.9	8.3	43.1	48.6	8.3	44.4	26.4	29.2
Total	44	25	75	17	118	9	46	73	25	64	31	49
% Lang	30.6	17.4	52.1	11.8	81.9	6.3	31.9	50.7	17.4	44.4	21.5	34

Table 6. Summary of exercises by Answer, Language and Category.

According to the overall results, the visual and textual languages performed similarly in tasks that were important for comparing and producing expressions. This is actually not a negative result, since the students had already seen textual expressions in mathematics and also in computer spreadsheets. The visual language performed better in exercises requiring computations and explicit evaluation of expressions. Although we expected tracing problems to be more difficult in the visual language because of having to follow data across parallel routes, the results indicate that in problems of the size dealt with in the present study this is not an issue and that textual representation is more prone to produce errors in calculations.

The worksheet also included a questionnaire to evaluate the user perception on several factors related to the understandability, comprehension and learning of the languages used on the worksheet. The first part of the questionnaire consisted of some general questions on learning the languages (GQx) and some basic questions to assess how difficult each category was (CQx). These questions used a 5-point Likert-scale in which “1” represented a strong disagreement with the given statement whereas “5” meant strongly agreement. Table 7 shows the descriptive statistics and the Mann-Whitney tests to check whether there were differences on the agreement degree by language. Although the assessment of general questions show that learning and understanding the visual language was better rated than the textual representation, the tests showed that these differences are not important. However, for the questions about categories, the students perceived more difficulty in checking the equivalence of expressions in visual as well as in calculating in the textual language. With respect to the tasks involving modification and writing of expressions, they considered them equally difficult in both languages.

The second part consisted of single-choice questions concerned with which language, textual or visual, the participant would prefer in the situations given in the questions (SQx). This part of the questionnaire was answered by 24 students. It seems that the missing answers for this survey occurred as a result of being the last page of the worksheet and having no way to check whether the students answered it.

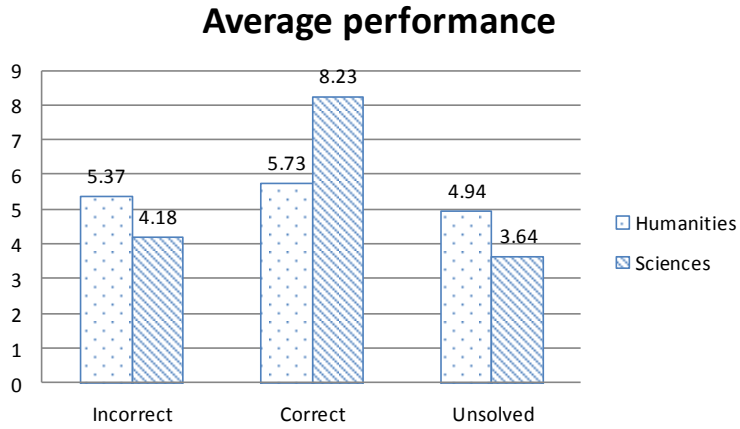


Figure 43. Average outcome per participant by curriculum stream.

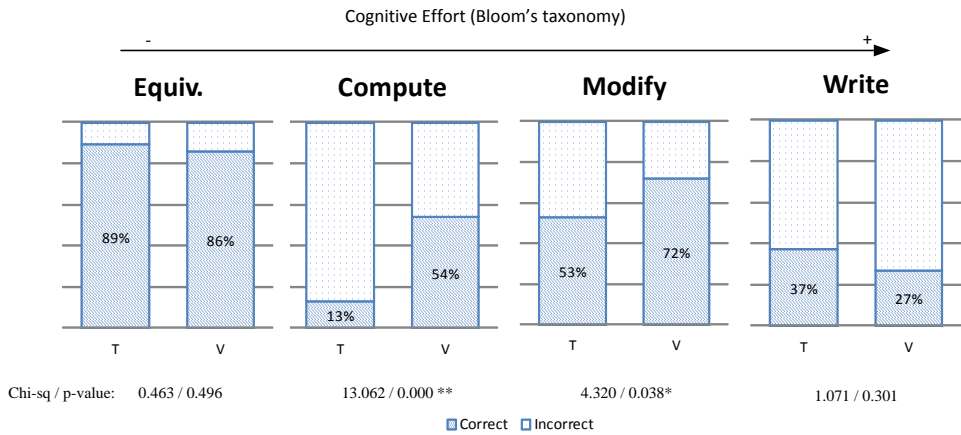


Figure 44. Answer performance per category.

Question	Overall			V			T			Mann-Whitney Test (Z / p)
	M	SD		M	SD		M	SD		
GQ1	I understand the meaning of the expressions	3.42	0.875	3.50	0.793	3.35	0.950	-0.629 / 0.530		
GQ2	It was hard to learn how to use the language.	2.86	0.854	2.77	0.863	2.94	0.854	-0.773 / 0.440		
GQ3	I consider easy to understand the different components of the language (operators, variables, etc.)	3.14	0.972	3.26	0.984	3.03	0.964	-0.979 / 0.327		
CQ1	I find it difficult to determine whether two expressions are equivalent or not	2.69	0.129	3.00	0.913	2.41	0.907	-2.420 / 0.016	*	
CQ2	I find it difficult to calculate expressions written in this language.	3.14	0.963	2.85	0.989	3.39	0.882	-2.202 / 0.028	*	
CQ3	It is hard to modify existing expressions	2.53	1.002	2.57	0.978	2.63	1.006	-0.125 / 0.900		
CQ4	I find it difficult to write expressions with the language	2.78	0.861	2.68	0.945	2.86	0.789	-1.071 / 0.284		

Table 7 User Assessment Summary.

Question	Textual (%)	Visual (%)	Test (Chi-sq / p-value)
SQ1 It is easier to learn the language...	30	70	3.522 / 0.061
SQ2 I find it easier to understand the expressions written using...	32	68	2.909 / 0.088
SQ3 I understand the different elements better (operators, variables, etc.) in...	22	78	7348 / 0.007
SQ4 I learned the language faster...	26	74	5.261 / 0.022
SQ5 It is easier to calculate outcome values with the language...	17	83	9.783 / 0.002
SQ6 I am able to write expressions using the language more easily in...	46	54	0.182 / 0.670
SQ7 It is easier for me to know whether two expressions are equivalent using the language...	57	43	0.391 / 0.532
SQ8 When modifying an expression so that it represents a given statement I find out faster what needs to be modified using the language...	39	61	1.87 / 0.297

Table 8. User preference summary.

Table 8 summarizes the results. There is a marked preference for the visual dataflow language with regard to ease-of-use, ease of understanding, learning and use in calculations. This indicates that using a dataflow model seems to be a promising conceptual tool and should be seriously considered for non-programmers. Besides this empirical evidence, dataflows are an alternative to writing code since natural interfaces based on touch input are required in intelligent environments.

The previous results only take into consideration the experimental conditions, using paper and pencil, but it is promising because the exercises dealt with abstract processes in conceptualization and understanding. In this respect, in the context of our line of research, in which collaboration and co-located operation around a tabletop are an important factor, visual languages have great advantages over textual alternatives. Visual dataflows such as those presented here can be created, discussed and manipulated in collaboration on an interactive table by using pucks that give access to entities and operations and flows may be manipulated by means of touch interactions. Since the results obtained demonstrate that the visual approach enhances the processes of calculation and modification of complex expressions, a tangible tool for defining such types of

expressions within reactive rules can be envisioned and implemented, as we will show in next section.

4.5 A Tabletop Based Rule Editor

In the design of a tool to support the proposed rule model, the main challenge is to supply such rule expressiveness while considering textual input techniques inconvenient for tangible surface interfaces. As already mentioned, rule models have been accepted and validated to be suitable to non-programmer by means of empirical studies and most rule editor proposals typically rely on drag&drop metaphors to establish the different parts of the rules. However, in our opinion these interaction techniques are not completely suitable and possibly not enough to deal with expressiveness complexity in a convenient way. From our previous study, some evidence has been provided in that visual dataflows can be promising to express short and medium assignment expressions. Beyond that, visual dataflows seem suitable to deal with such expression programmability in environments based on tangible surface interfaces. The main rationale is that a suitable editor in this context would require co-located, cooperative and collaborative performance by multiple users, flexibility in user interface layout and view-point independence, and a combination of touch and tangible techniques. All this can be achieved by a decomposable expression model such as the one based on visual dataflow expressions.

4.5.1 Tabletop Design

Developing a prototype to support rule editing with the expressive rule model being adopted is challenging. Thus, prior to the implementation we involved ourselves in an iterative design process based on a mock-up setting specifically aimed at exploring, discussing and reflecting on interaction issues that typically arise in multi-touch and tangible interaction on interactive horizontal tabletops, and how the information should be presented.

Some requirements to be considered are as follows. Firstly, concepts should be shown in such a way that users feel comfortable with new technology. Secondly, users should be able to interact as naturally as possible, exploiting the features that interactive tabletops offer, and benefiting from collaboration and cooperative interaction in co-located spaces.

Inside the rule model, two different types of artifacts can be distinguished. On one hand, those referring to elements in the ecosystem such as the entity popu-

lation source, the event and the population target. On the other hand, the expressions such as conditions and calculations for the operation parameters.

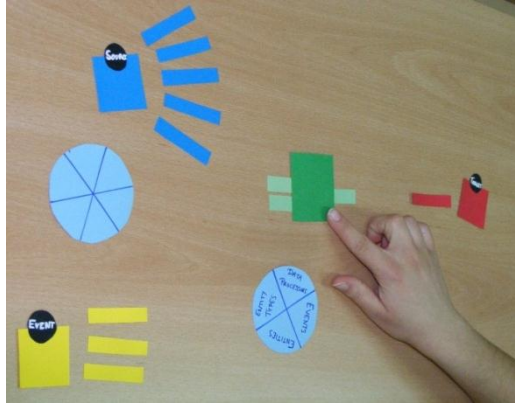


Figure 45. Mock-up setting that helped to test the design rationale behind the editor.

The elements in the first group are mandatory and give us access to the properties and attributes to be involved in the data transformation expressions. Containers for the source, target and event must be filled by means of collection explorers used to examine the entities and events present in the ecosystem. The containers are required to be flexible in terms of positioning and manipulation in order to facilitate parallel interaction by several users. Figure 45 shows a performance with the mock-up settings in which the containers have been supposedly established through the explorers and a dataflow expression is being manipulated.

Once these containers are correctly edited, the operation may be established and dataflow expressions edited. Since multiple dataflow visualization and edition would be difficult to be scalable and usable, only one dataflow can be edited at a time. Switching between data processes for conditions or operation parameters is performed by means of a specific selector. Thus, the whole rule is not shown but it is assembled as a set of views according to the needed data processes. According to the rule model, at least two views are needed, one for the condition and another for the filter. Additional views are required according to the operation being involved. In this way, only the source, the target, and the event are displayed and modifiable in all views, and therefore the user exclusively focuses on editing a single dataflow at time.

Dataflow editing requires operators being allowed into the workspace and positioning them. These are represented in the figure by green boxes with inputs and outputs shown as strips. Properties of the source, event and target, represented with strips, can be explored and made available for the dataflow being edited. Finally connections between properties and operators are established by means of finger dragging movements.

4.5.2 Implementation

The prototype was developed for a Microsoft Surface Unit and, as all the other software developed in this work, it was implemented using the Microsoft XNA Framework, the Microsoft Surface SDK v1.0, and the toolkit of controls for building tangible surface interfaces introduced in the previous chapters, which offer advantages on tabletop cooperative applications when selecting items from collections.

In order to meet the flexibility requirements as in the mock-up design, specific physical tagged objects (pucks) were used to provide access to the source, event and target panels respectively. By tapping on the panels we are able to explore collections of entities or events. Another specific puck provides access to operators, and connections are directly established by finger movements. All the components are freely movable and rotatable. The pucks associated to the panels can be lifted up so that panels become hidden, gaining more available space. In addition, a “remove” puck is available to facilitate the removal of any part of the dataflow expression. The change in view is performed by means of the “view” puck, which gives access to a selector.

When editing a data process, some online checking is performed so that visual feedback can be provided to facilitate the construction of syntactically correct rules. Operators and flows are colored in green, orange or red as elements are fully and correctly, partially or incorrectly configured respectively.

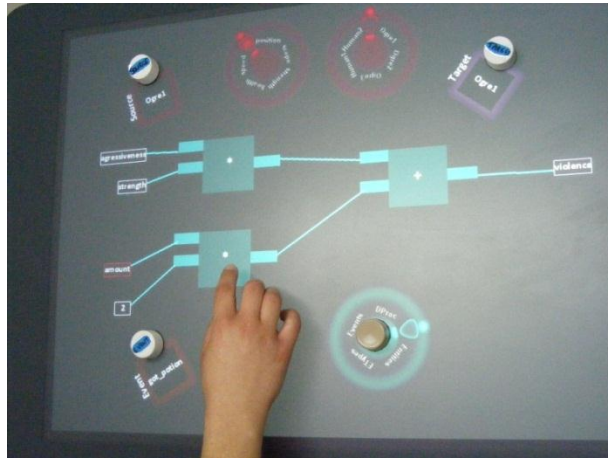


Figure 46. Rule editing prototype.

4.6 Conclusions

A behavioral rule model has been introduced following a visual structure for dataflow expressions as an expansion to the physical entities. Precisely, on the comprehension of dataflows, an empirical study has been conducted involving teenagers. The study revealed that small and medium dataflow expressions can be considered as a useful conceptual tool to express data transformation by non-programmers. Finally, the chapter presented the tabletop-based editor implemented to specify rule behavior. Chapter 6 will present the enactment process that supports the simulation of ecosystems and will introduce specific model instantiations to create some basic sample games.

The most immediate future work related to the tabletop rule editor is to conduct an empirical user study evaluating the use of the editor completely. It would consider how easy is to be used as well as how users collaborate on the tabletop to discuss and implement rules.

Chapter 5

Surface Interaction Toolkit Support

This chapter describes the interaction toolkit designed and implemented in the context of this thesis to support the construction of user interfaces for surfaces. This basic toolkit supports a range of widgets with multi-touch and tagged-tangible input. As seen in the previous chapters, exploring collections to build the ecosystem is one of the most extended interface requirements (e.g. blocks, operators, etc.). Thus, this chapter describes the evaluation of the design rationale of the control for collection provided by the toolkit. In particular, some experiments have been carried out, focused primarily on the TangiWheel control. The aim was to study the effect of input method, i.e. fingers or handlers (pucks), on performance in manipulating collections in terms of performance time and number of manipulations required. Experimental complexity was gradually increased to evaluate the use of the widget under different circumstances.

5.1 Tangible Surface Interfaces

In the field of Human-Computer Interaction (HCI) much research is being directed towards improving current methods of communication and interaction with computers (i.e. keyboards, mice, joysticks, etc.) so as to make these more natural and intuitive. Among the new techniques and devices, potential replacements for the mouse are interactive gloves, voice recognition, control by following the user's eyes, and the latest of all, tangible interfaces and interactive tables. In recent years, concepts such as multi-touch interaction or multi-user interfaces have come into fashion with the appearance on the market of multi-touch devices such as Apple's iPhone & iPad, Microsoft Surface, Google's Android phones and tablets, Lémur from Jazz Mutant, or the Reactable musical instrument. However, these technologies are not completely new, since their underlying concepts have been under development for more than 25 years (*Mehta, 1982*)(*Krieger, 1991*). What exactly do tangible interfaces and interactive tables consist of? And what can they contribute, apart from the already familiar multi-touch capacity?

Tangible User Interfaces (TUI) combine control and representation in a single physical device (*Ullmer, 2001*). This concept both amplifies and simplifies the direct manipulation paradigm (*Shneiderman, 1983*) on which conventional

graphic user interfaces (GUI) are based. In direct manipulation with GUI type graphic interfaces, users interact with digital information by selecting graphic representations (icons, windows, etc.) with pointing devices (e.g. a mouse). In the case of tangible interfaces such as interactive tables, users interact with the system with their hands and their fingers and also by manipulating specially configured physical objects. The concept goes thus far beyond the simple idea of multi-touch. These types of table are much more than flat screens that can detect many fingers simultaneously. Interacting with the fingers still belongs to the idea of pointing devices, while interacting with physical objects can take us much farther. Such objects may represent abstract concepts or real entities; they may relate to other objects on the surface; they may be moved and turned around on the table surface. All these spatial changes may affect their internal properties and relationships with objects in the neighborhood.

An early example of this type of interface, which may allow us to better understand its potential, is the URP, a system developed as a town planning aid in the late nineties at the MIT MediaLab (*Ben-Joseph, 2001*). URP is a simulator in which various users can analyze in real time the pros and cons of different urban layouts by arranging models of buildings on the surface of an interactive table, which represents the plan of a town or a district. The surface provides important information such as building shadows at different times of the day (see Figure 47). URP is executed on an augmented table with a projector and a camera pointed at the surface from above. This system permits the detection of changes in the position of the physical objects on the table and also projects visual information concerning the surface. Other elements can be included, such as a clock to control the time of day in the system. URP detects any changes made in real time and projects shadows according to the time of day. All of these properties could obviously be achieved with software, but the interest of this simple prototype lies not on its capacity to calculate and simulate shadows, but also on that information can be manipulated collectively, directly and intuitively.

URP was one of the first tangible interface prototypes. It is simple and demonstrates the benefits of this type of interface and its ability to integrate, manipulate and update both digital and physical information seamlessly and consistently. These tables strengthen concepts such as “social interaction” and “collaboration” (*Hornecker, 2006*)(*Marshall, 2007*) or “game interaction” (*Gaver, 2004*). If they are compared to conventional WIMP interaction (windows, icons, menus and pointers) they provide a form of interaction much closer to the real world.

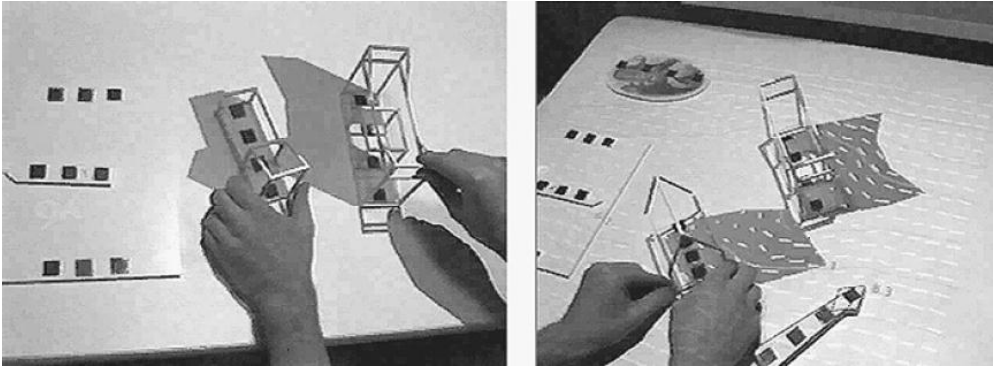


Figure 47. URP System (M.I.T MediaLab)

While in WIMP interaction the user input is restricted to an ordered sequence of events (click, double click, etc.), which permanently limits the workflow, with interactive tables any action is possible by any user, at any time and place, and in any order. Also, the seamless integration of physical control and visual output means that interacting with this type of systems takes place in the most natural and direct possible way. It is not necessary to know where to click to change a given parameter; whatever has to be changed can be touched and manipulated directly.

All the potential that surfaces can offer now, facilitating direct manipulation, multi-touch and multi-user experiences, is necessarily bound to the development and improvement of different touch technologies over decades that are broadly described next.



Figure 48. Example of Interactive Surface.

5.2 Touch Technologies

As reminded in (Schöning, 2008), Touch technologies are not new and root from early 70s (Buxton, 2012). The improvement and decrease in cost of electronics, along with the emergence of new materials, have allowed the exploitation of touch enabled displays primarily based on either resistance or capacitance based technology. These technologies require relatively complex electronics as the sensing mostly relies on hardware. They have facilitated the popularization of small-scale touch screen devices for the masses such as PDAs, mobile phones and tablets, and even provided some approach to large tabletop based displays, such as the Diamond Touch (Dietz, 2001). However what probably contributed more to intensive research on multi-touch applications have been the camera-based approaches to touch sensing, which allow building large-scale surfaces at a low-effective cost even in non-industrial settings.

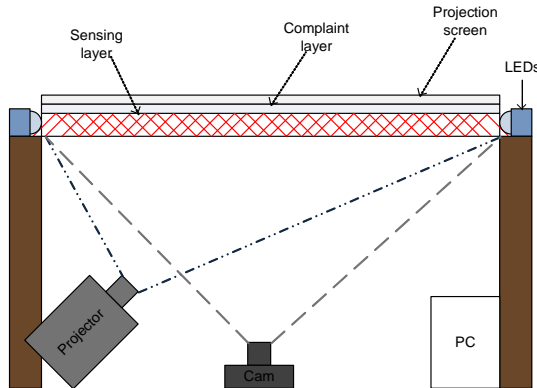


Figure 49. Sketch of an interactive surface.

Camera-based surface technologies require a camera to capture surface images along with some simple electronics to trigger the touch sensing effect on the surface. The main two streams for optical camera-based surfaces are based on the Frustrated Total Internal Reflection (FTIR) principle which popularized Jeff Han (Han, 2005), and the Diffuse Illumination (DI). Essentially, both approaches try first to produce a uniform pattern from some infrared light sources, and then, based on light refraction properties, capture disruptions in uniformity illumination when objects touch or are close to the surface.

The FTIR approach consists of a frame with mounted IR Light-Emitting Diodes (LEDs) which inject IR light within the sensing layer. This light is completely reflected internally within the sensing layer, if its material has a higher

refractive index than the outer material. When a user touches the sensing layer, typically made of acrylic, the light is reflected at the contact point due to its higher refractive index, and this disruption in the internally reflected light can be captured by an infrared sensitive camera. Figure 49 shows a schematic illustration for this setting.

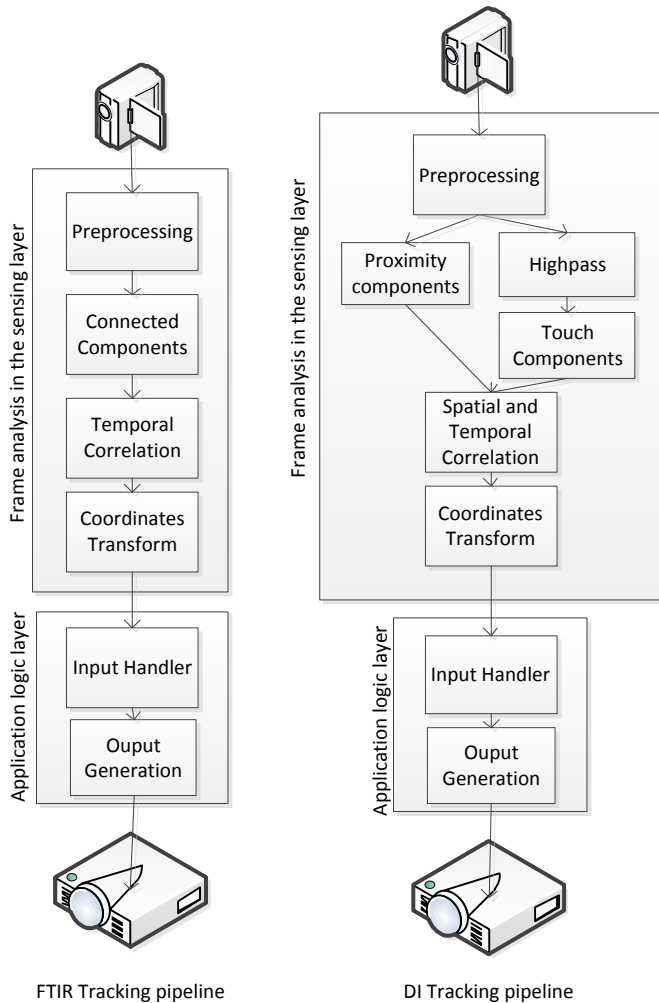


Figure 50. Tracking pipelines.

The sensing layer is responsible for enabling the physical principle required to be able to capture and detect contacts. Beside this, other layers may apply for an improved performance or in case a screen display is needed. For instance, the most normal configuration for tabletop systems using FTIR is one consist-

ing of two additional layers. The compliant layer, typically made of silicone or latex, facilitates and improves user performance, as it reduces the pressure that users have to apply to trigger the FTIR effect. On top, the projection or diffuse layer acts as a screen in which images are rear-projected. More details about all these issues as well as affordable materials can be found in (Schöning, 2008).

Similarly, the DI also uses infrared light sources, but this time they are placed behind the sensing layer. On the infrared illuminated surface, any object touching the surface, or in proximity, produces a disruption that the infrared sensitive camera can capture.

Both sensing approaches need of a software pipeline to process the images captured, and be able to detect and extract the contacts in the end. Figure 50 shows the typical steps in the software pipeline. The pipeline in both approaches is quite similar. Basically the sensing software layer, based on Computer Vision techniques, processes image frames captured by the camera in order to obtain the contact input information. First, the image is pre-processed so that unchanged regions with respect to previous frames are removed by means of thresholding and background subtraction techniques. Then image labeling along with connected components extraction techniques are applied to obtain the information of the areas in the image representing touch contacts or objects. The information of touches in several previous consecutive frames must be correlated, and then transformed to the screen coordinates. Finally, the application must receive all this touch information, execute code as a result, and produce the specific output of the system. Figure 51 shows the state of a sample frame capture in several stages.



Figure 51. Sample images in several specific stages in the tracking pipeline.

There are several toolkits and libraries available to facilitate the fulfillment with this pipeline at a high level. These try to cover the analysis process as a whole, with the aim of providing with information of touches that can be directly consumed by the application layer. Some examples are the *libang* (Zadon, 2012) and

Touchlib (Wallin, 2006) libraries. The most essential abstraction is broadly focused on contact information and a set of related basic events. For instance, Figure 52 shows the core data structure used in the TouchLib library and the basic interface offered by a touch listener. With each analyzed frame, each finger or object in touch with the surface produces a contact. Each contact is characterized by a position in the current frame, a physical description of the contact shape typically expressed in terms of an ellipse, and further information such as the variation of the position with respect to the corresponding contact in the previous frame. In order to keep the tracking information on contacts from subsequent frames, the temporal correlation process in the pipeline provides an identifier to each contact, so that two contacts in different frames will be related if the identifier is the same. For instance, when a finger is put down on the surface, a contact will be detected with a new identifier and delivered to the application layer by means of a finger down event. If the finger is not lifted up and then dragged across the surface, the subsequent frames detecting the finger will produce more contacts with updated values for the position and other properties but all these contacts will be related by means of a common unique identifier. These will be notified by means of the finger update event handler. Once the finger is lifted up, the analysis process will detect that a contact has been lost, and a finger up event will be notified with the information of the last contact for this finger. The identifier relating all these contacts will not be used any more in future contacts.

```

class ITouchListener
{
public:
    /// Notify that a finger has just been made active.
    virtual void fingerDown(TouchData data) = 0;
    /// Notify that a finger has just been made active.
    virtual void fingerUpdate(TouchData data) = 0;
    /// A finger is no longer active..
    virtual void fingerUp(TouchData data) = 0;
};
...

struct TouchData
{
    int ID;
    int tagID;

    float X;
    float Y;

    float height;
    float width;
    float angle;
    float area;

    float dX;
    float dY;
};

```

Figure 52. Core data structure in Touchlib.

From this basic data structure, more complex gestures can be detected and delivered to the application layer. For instance, higher level events could be provided to notify panning or stretching gestures that in the end consist of basic down, moving and up contacts.

In the present dissertation, to simplify the complexity of the hardware and software development process, the tabletop platform being used is the Microsoft Surface Unit, a state-of-art product for surface computing marketed by Microsoft (*Microsoft Surface, 2012*). Next subsection describes the basics of this platform and the associated Software Development Kit (SDK).

5.2.1 Microsoft Surface

The Microsoft Surface Unit⁶ is a computer with a table form factor enabled with a touch-screen display of 30 inches (*Microsoft Surface, 2012*). The unit basically relies on optical camera-based surface technology based on Diffuse Illu-

⁶ MS Surface and PixelSense: <http://www.microsoft.com/en-us/pixelsense/default.aspx>

At the moment of printing this work, Microsoft renamed MS Surface as PixelSense.

mination principles. It includes a 5-camera vision system able to sense objects, hands gestures and touch as seen before. It primarily provides two basic interaction mechanisms, namely *fingers* and *tags*, both subsumed by the general concept of *contact*. So a contact can actually be a finger or a tag. It supplies us with the information about location, rotation, etc., allowing us to track it over time, and capture gestures by means of some sort of manipulation processors. Hence contacts are considered as raw events running on the system as mouse and keyboard events are for desktop Windows applications. This is essentially the general idea behind the aforementioned libraries, but this time the Microsoft Surface SDK v1.0 delivers in a convenient way this and much more functionality to developers. The SDK provides two versions tailored to two different application programming interfaces (APIs) and programming models. This organization results in two different layers in the SDK: the Core layer and the Presentation layer. The Core layer is the basic part of the SDK supporting the development of surface computing applications in the context of the Microsoft XNA Framework (XNA), whereas the Presentation layer is oriented to the development of applications based on Windows Presentation Foundation (WPF) APIs. Although both parts of the SDK provide the same basic functionality, they differ in the provision of high-level abstractions which determines both the easiness and the effort to create new touch enabled controls.

The WPF Surface layer makes it easier the development of touch-enabled applications to programmers who are already developing Windows applications. Alike it uses a programming model conducted by events (*ContactUp* instead of *MouseUp*, *ContactDown* instead of *MouseDown*, etc.) where applications are built up of several basic GUI elements called Controls. These basic abstractions are visual containers of information and information themselves at the same time (i.e. they can contain other controls). Touch-enabled controls are not only about visualization (i.e. output) but also they are the elements which receive the inputs. This keeps the input management confined to the context of each control instance, so that programming becomes more affordable and scalable with this abstraction. The layer is responsible for routing the input contacts towards the corresponding controls and raising the events, releasing developers of these cumbersome tasks.

The XNA layer is more focused on the way videogames are developed, because it consists of a set of abstractions and APIs to develop computer games. Its programming model is slightly more complex, with fewer facilities but with more control on how the application elements and behavior are coded. The layer does not provide any control-based abstraction. Hence there is not an abstraction bringing together output and input management, and, consequently,

the application needs to implement a lower level and explicit mechanism in the message-loop to route contacts and events towards the visual elements. Fortunately, the SDK provides with some sample code, although incomplete and short in functionality, to facilitate the development of a control abstraction and an event manager to route events to the different UI elements. The implementation of our proposed interaction widgets explained in the following section relies on this approach. Table 9 contrasts the main differences between both layers available in the Surface SDK.

	WPF	XNA
Basic UI abstraction	UI element	any (from scratch established by the developer)
Layout	Automatic or managed by the UI element logic (containers and content)	any (from scratch established by the developer)
Input-Output management	UI Element / high level / event based	message loop / low level /message based
Captured frame available for further analysis	no	yes

Table 9. Capabilities facilitated by the WPF and XNA layers in Microsoft Surface.

5.3 User Interface Controls

To support the construction of user interfaces for surface applications, some basic middleware have been developed on top of the XNA core layer supplied by Microsoft Surface SDK. Basically, it has been necessary to develop a set of widgets (e.g. panels, buttons, menus, etc.) along with several interaction handlers that support a more transparent and convenient management of both touch and tag input. As an overview the Figure 53 illustrates the class diagram for the user interface controls that will be explained next.

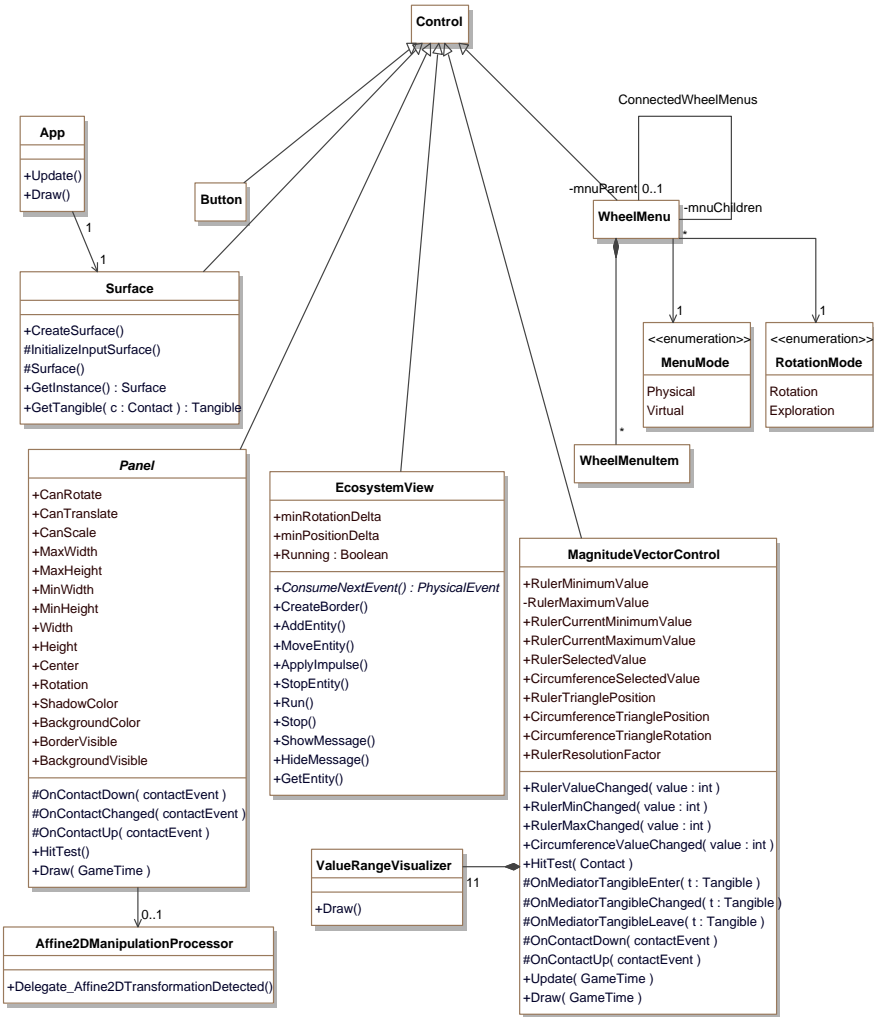


Figure 53. UML class diagram for control hierarchy.

5.3.1 Basic controls

On top of several programming interfaces provided by the SDK, we have created the class *Control* as the main abstraction to represent user interface elements being able to have a visual existence and receive input (see Figure 54 for base properties and services provided by the class *Control*). Every control has a visual representation that is rendered according to its internal state and properties (e.g. position, rotation, etc.) by invoking its *Draw* method. A control can be

positioned and rotated as desired across the surface with respect to a global screen coordinates frame. Controls can contain other control instances so that a hierarchy of controls can be built. The methods *AddChild* and *RemoveChild* help to keep this hierarchy at runtime. By default, this hierarchy of controls is drawn and behaves consistently when a top control is manipulated so that all the inner controls in the hierarchy are affected by the manipulation. For instance, if a control is scaled or rotated, the inner controls will be scaled and rotated accordingly, maintaining the logical relationship between controls. To facilitate this, the scale factors are interpreted as multiplicative factors with respect to the parent control, as well as the positions are specified in normalized coordinates between $[-1, 1]$ for similar purposes.

Controls can be easily shown, hidden, minimized and restored (methods *Show*, *Hide*, *Minimize* and *Restore*), and are notified of basic contact events by using the contact controller (*UIController*). The details about how the contacts are managed across all the controls and how they are routed to the specific control impacted by a contact event will be explained later, once the different controls and their relevant elements have been introduced.

Extending the abstract class *Control*, there are several base controls: *Panel*, *Button*, and *Surface*. Figure 55 shows a sample of *Panel* and a *Button*. The *Panel* is an extended control with a rectangular shape and ready to automatically handle complex built-in input manipulations such as rotation, translation, and resizing. This is implemented by the class *Affine2DManipulationProcessor* delivered with the Surface SDK.

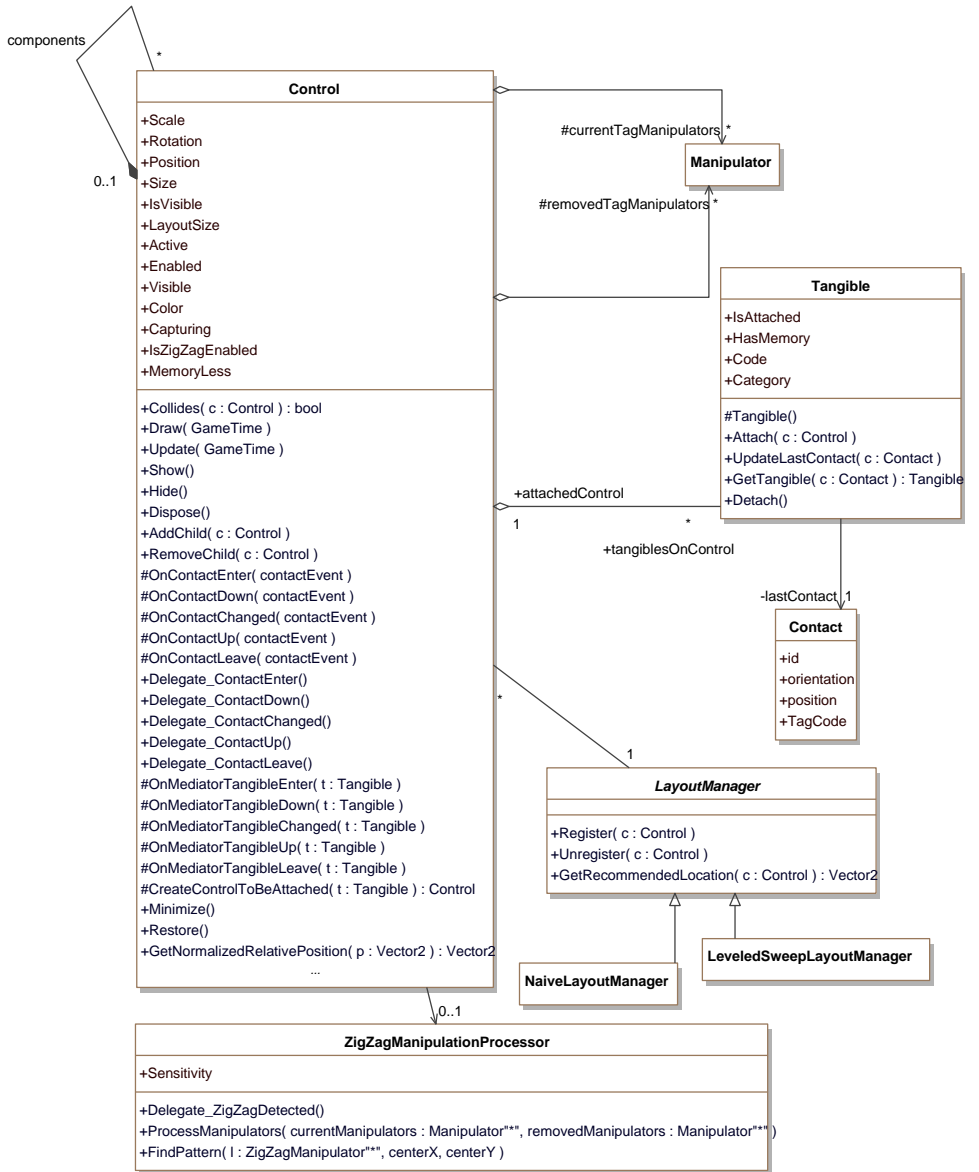


Figure 54. UML class diagram for the Control class.

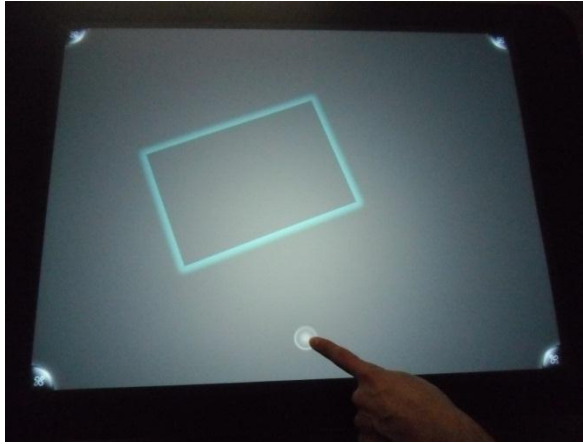


Figure 55. Samples of panel and button control instances.



Figure 56. Surface control, always present at the bottom of any other control.

The class *Surface* is one of the most important controls. Every application must have one and only one instance. Moreover, the *Surface* control will be the top-most control in the runtime hierarchy of nested controls (the parent control of the controls in Figure 56 is an instance of the *Surface* control). It means that any basic application will have a surface instance with the size of the tabletop, and every other control will have this instance as *Parent*. In this way all controls are related through parent/children links, except the surface control, which is the root element of the hierarchy. This facilitates traversing the control hierarchy to perform several tasks related to input control management.

5.3.2 Input Management

As it was mentioned before, the Microsoft Surface Unit provides with several types of interaction input embraced within the general term of contact. Each contact detected by the computer vision subsystem is correlated with others by means of a unique identifier and provides some basic information such as position, orientation, etc. If the contact is a tag, not a finger, then some extra information is available such as the tag data. Tags can be of two types, one providing an 8-bit code, and another providing a more complex coding scheme based on a pair series-value within 128-bit. Figure 57 shows a class diagram illustrating the types of contacts and their information as well as two samples for each type of visual tag.

Since the development of applications with the core layer of the Microsoft Surface SDK does not provide a transparent support for the development of controls and input management, the management of contacts must be automatically performed in order to be able to build functional controls.

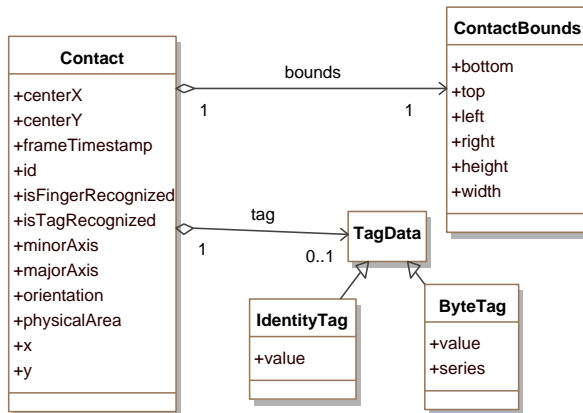


Figure 57. UML class diagram for contacts.

Contact processing can be carried out by an input controller or contact controller (*UIController* class) that is able to take directly the information of detected contacts from the computer vision middleware. This controller keeps several queues of contacts so that it is able to analyze the input interaction over the time departing from this raw contact information. The controller is able to raise higher level events expected to be consumed by the application layer, such as:

- *Enter*: it is raised when the contact has entered in a new user interface element.
- *Leave*: it is raised when a contact that had previously entered in, is leaving the current user interface element. For example, this happens either when a finger is dragged across a control beyond their boundaries, or when a finger on a user interface element is lifted up, not being in contact with the surface anymore.
- *Added/Down*: it is raised when a contact starts touching the surface.
- *Removed/Up*: it is raised when a contact leaves the surface.
- *Changed*: it is raised whenever a contact is changing any property value (position, orientation, etc.) as a consequence of moving the finger or the corresponding tag.

The typical life's cycle of a contact is as follows. When a finger/tag is put down on the surface, the vision subsystem creates a unique identifier for the contact and a *Down* event is raised. Subsequently, as the finger/tag is impacting on an interface element, an *Enter* event is also raised. Then if the finger/tag is moved (the position and orientation will change), events of type *Change* are raised in consequence. Finally, if the finger/tag is lifted up, *Leave* and *Up* events are raised.

These events, notified by the contact controller, are the basic events that the class *Control* provides. In addition, zigzag gestures are also notified, which are more complex events by far, but this additional function is provided by our implemented *ZigZagManipulationProcessor* class.

To enable the input controller to successfully carry out its work, it is necessary to define what a user interface element is and provide some abstract services to allow the controller the adequate routing of events. A user interface element is an abstraction that supports the reception of input and notification of events by the controller, which in our case is provided by the *Control* class. The abstract service that has to be implemented by controls is *HitTest*. This allows the controller to query whether a point of a contact is impacting on the control according to the runtime control hierarchy. In this way, the controller is able to determine which control the event should be routed to.

In our interaction toolkit, a tangible object is represented by the *Tangible* class. A tangible can contain an associated widget (methods *Attach/Detach*) so that the attached widget (attribute *AttachedControl*) is controlled by means of the tangible. This is useful to automatically manage the expected behavior of the

widget when the tangible is lifted up and put back again. By default, when the tangible has an associated control, the widget appears, disappears, moves, and rotates according to the manipulation of the tangible.

Because tangibles simply produce contacts with some additional specific information, they could be handled as regular contacts. This means that the contact controller (*UIController*) routes the corresponding tangible contacts to the control which is impacting on, and the specific application code could then indicate the response of the control. This approach requires developers to code the system response scattered in the code of each control which could potentially receive the input of a specific tangible. However, with the aim of mitigating this effect, and facilitating the construction of user interfaces, some separated event handlers have been considered in order to cope with this complexity. In particular, the logic of tangibles and controls is able to switch the routing contact events generated by a tangible towards its attached control. In this case, the tangible can be considered as a mediator of the interaction and the callback methods in the class *Control* that handle the events are the *OnMediatorTangible*{*Enter, Leave, Down, Up, Changed*} methods. In this way all the code can be gathered and centralized in the control to be associated to the tangible instead of being scattered across the application.

5.3.3 Complex controls

In addition to the previous typical controls, there are also some more elaborated controls that cover important functionality. They are the Linear List, Magnitude and TangiWheel controls. The first two are described in the following, while the TangiWheel control will be described separately and in more detail in Section 5.4 as it is an important piece in the AGORAS toolkit in terms of research effort and contribution to the field.

5.3.3.1 Linear List

The linear list is simply a rectangular container of items, similar to the ones in WIMP user interface toolkits. However it is resizable, rotatable and movable as a panel by means of fingers. Moreover, the visible elements can be conveniently changed by dragging the finger on the container in one or another direction. Figure 58 shows the implemented list. It has a border, which is large enough to allow manipulations (i.e. moving, resizing and rotating). There are two small arrow icons in the container area indicating whether there are hidden items in those directions.



Figure 58. Linear list control.

5.3.3.2 Magnitude Control

The magnitude control is a novel and generic control that fits to a wide variety of situations which require setting a value accurately. For example, quite often we have to resize elements with fingers, as required in our context when we are editing *entity* components on the surface. These changes are not always convenient due to the small original size element or the *exit error* observed in finger-based tabletop systems as reported by (Tuddenham, 2010). In the same context, it could be also interesting to establish the scale factor or the size of a figure while modifying also its orientation. This is typically performed by means of combined finger based gestures, but when high accuracy is needed, they could be inconvenient. In WIMP based interfaces the keyboard plays a key role when accuracy is required, since the values are directly introduced by users in the form of numbers. Our magnitude control will allow this without even using a surface based keyboard. Essentially, the base magnitude control is designed to set angles and/or numerical values within a range. There are three variants that are explained next.

The first variant basically consists of a rule (see Figure 59). It works on a numeric value. When the puck is put down, the rule and the current number value are visualized. The puck can be then rotated and moved for user convenience, producing the corresponding changes in visualization. The control supports ranged values, which go between some minimum and maximum limits. If users want to change the numeric value being visualized, they only have to put a finger at any point on the rule, and then to move the puck towards the desired value according to the rule scale. In this case, the finger simply enables that the changes in the position of the puck entail changes in the associated numeric value.

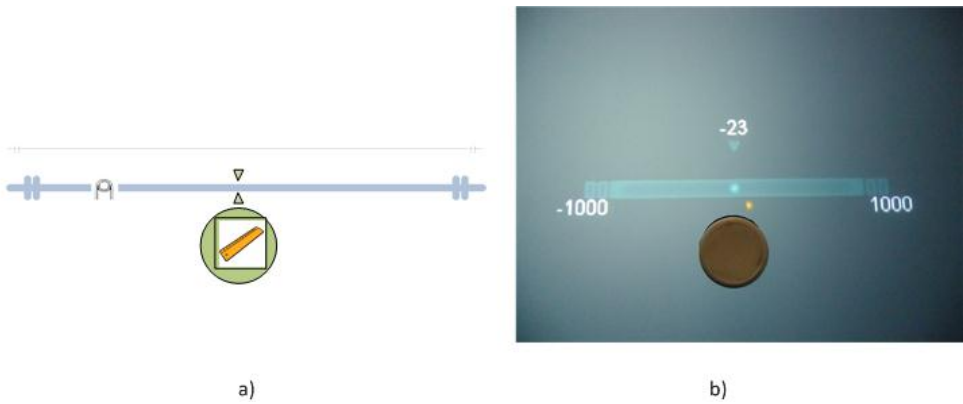


Figure 59. Rule variant of the magnitude control.

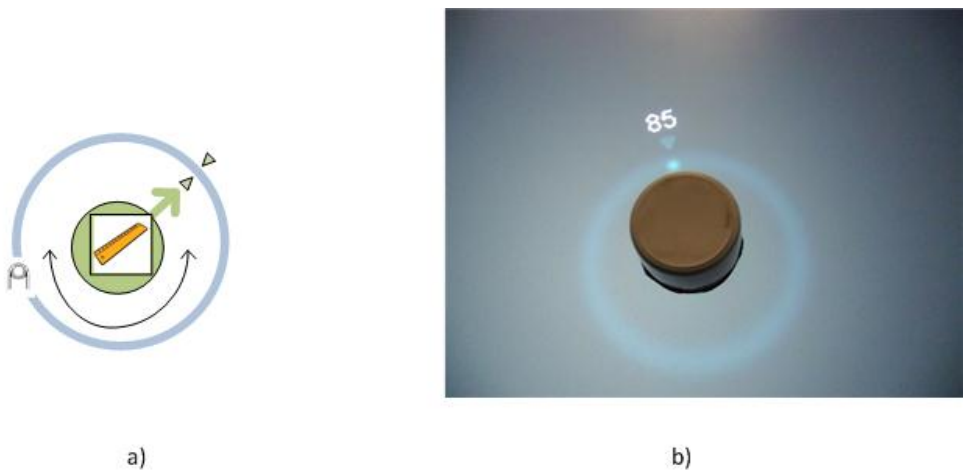


Figure 60. Circle variant of the magnitude control.

In addition, the control also supports zooming. It is useful in those situations in which the size of rule would not be large enough to cope with the desired accuracy. In this case, instead of providing a long rule, which could not always be feasible, the control may change its resolution, i.e. the visualization scale of the rule. If this feature is enabled, the resolution can be changed by means of puck rotations while the rule is hold with the fingers. Clockwise rotations cause *zoom in* effects in the resolution, whereas counterclockwise rotations produce *zoom out* ones. Indeed, the zoom rotations will always remain within the mini-

num and maximum limits, and will be governed by a set of attributes controlling the multiplicative resolution changes. As an additional feature, the rotation of the puck is always offered by the control. It is useful since it can be used in a given application for several purposes, providing an additional degree-of-freedom. For example, it could represent a vector that in combination with the numeric value provide the vector and its magnitude to be applied in a scaling operation.

The second variant consists of a circle (see Figure 60). The functionality is almost the same as the rule based control. However, it fits better to situations in which a circle or rotation metaphor is more suitable such as updating numeric values related to angles or orientations. As in the rule approach, the value will be changed if the finger is held on the round rule.

Finally, the third variant basically integrates both previous ones (see Figure 61). In this case, the control is able to provide readings for two numeric values that could be mapped to several properties of an object being edited in a given application (e.g. the orientation, scale factor, etc.).

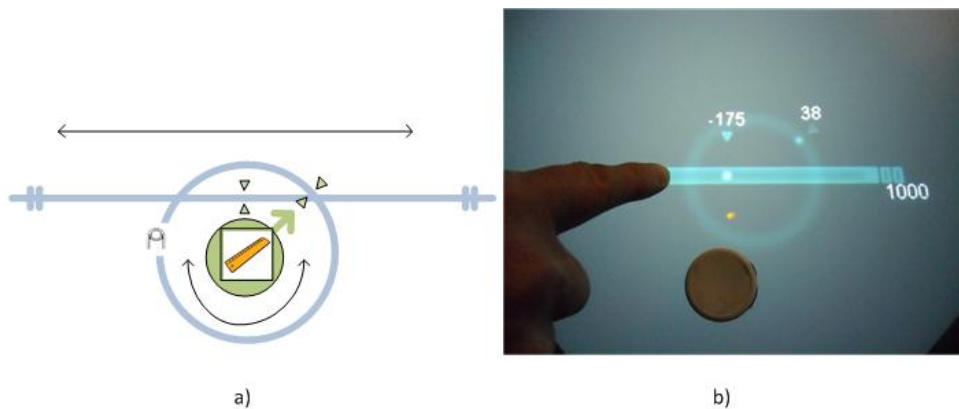


Figure 61. Combined variant of the magnitude control.

5.4 TangiWheel: A Hybrid Design for Collection Exploration

The previous section just presented the set of controls developed to build surface applications with potential different requirements. The creation of ecosystems requires exploring a variety of artifacts as well as parts and components to be selected in order to assemble entities or define different aspects of rules as shown in the previous chapters. Thus, exploring collections is one of the most often required interactions in the context of AGORAS. Consequently, the

TangiWheel widget has been considered as the cornerstone control within the toolkit and therefore it deserves a special attention.

Collections are typically supported on desktop applications by list or menu widgets. Lists generally contain a potentially unlimited collection of items, which are typically application-domain entities (i.e. data), whereas menus normally contain application commands or option selections. Menus are usually fixed, or have a limited collection, generally always-visible and linked to a point on the user interface. On tabletops, however, lists and menus do not have a predetermined form and quite often the difference between them is somewhat blurred. Firstly, neither of them is fixed to a point on the tabletop because menu options are usually attached to information elements scattered across the surface. Secondly, both must support a 360-degree interaction style, so that the interface is not oriented towards one user when several are collaborating on the same surface, or at least a way to manage this issue. Thirdly, both must provide effective mechanisms for the search and selection of elements so that the tabletop surface is not overloaded when multiple search and selection processes (on different lists/menus) are taking place simultaneously.

Before designing Tangiwheel, an extensive analysis of existing widgets for collection manipulation has been carried out across several aspects and features. Such comparative analysis has allowed us to determine interesting characteristics that Tangiwheel should support resulting in a distinguished design to be the first proposal supporting a hybrid input modality with high resemblance levels between touch and tangible interaction styles.

5.4.1 A Survey on Widgets for Collection Manipulation

The design of the collection explorer, Tangiwheel, was influenced by a range of different studies. Firstly, the seminal work of Fitzmaurice, Ishii and Buxton on graspables (tangibles) (*Fitzmaurice, 1995*), as well as Ishii & Ullmer's vision of "Tangible bits" (*Ishii, 1997*), have dramatically impacted the concept of the tangible surface interface, in which direct tangible manipulation is performed through *physical handlers* or *bricks*. Other studies have described the use of *phicons* for containing, transporting and manipulating digital objects (*Ullmer, 1998*).

Shen et. al presented CoR²D, a virtual control for interactive pop-ups, which allows the visualization and launch of commands (*Shen, 2005*). CoR²D menu items can be moved and rotated by means of virtual handlers strategically attached to the menu items. This flexibility greatly helps to achieve desirable features such as occlusion, reach, establishing context on cluttered display, readability, and concurrent, coordinated multiuser interaction. However the

unsystematic layout of menu items on the surface makes this approach unsuitable for the simultaneous manipulation of multiple collections by different users, since the visual feedback from the different items can cause confusion.

As pointed out in (*Lepinski, 2010*), even though a great deal of work has been carried out on menus in recent years, menu systems for multi-touch platforms have received little attention. Efforts have been focused on improving performance of large mouse-based menus (*Accot, 1997*), (*Song, 2010*) or pen-based interfaces, where marking menus have clearly taken the lead in their different approaches and variants (*Bailly, 2007*), (*Zhao, 2006*). These generally provide support for hierarchical collection exploration but not for virtually unlimited items, despite the effort to substantially extend the size of the collection.

Patten et. al describe a series of interaction techniques for the exploration and selection of items (*Patten, 2006*). These techniques primarily consist of an approach based on two-handed interaction via pucks, one of which represents the item to be modified and the other is the modifier. An important feature is the visualization of items as pie menus. Another technique is characterized by the use of non-circular “floating menus”. The activation of both menu techniques is bound to specific sensitive areas or hotspots limiting flexible access to menu instantiation, although tangibles are always required.

Weiss et. al in (*Weiss, 2009*) present a set of active silicone peripherals, such as knobs, keyboards and buttons, which can easily be stuck at any point on the surface. This is a more elaborate approach than the *phicons*, and also provides more physical feedback, which improves the performance of interaction tasks. This study explores the use of a knob to manipulate pie menus and concludes that for simple tasks this more physical approach provides significantly quicker task completion than a purely virtual approach. However, it neither considers nor explores the use of cascading menus, which would perhaps be useful to support more complex navigation and other tasks, as a consequence of relying solely on physical peripherals.

Lepinsky et. al evaluate a marking menu specifically designed for multi-touch platforms (*Lepinski, 2010*) based on directional chording gestures. His experiments showed that multi-touch marking menus perform faster than traditional hierarchical marking menus.

Hancock et. al present a radial control that supports 2D information exploration (*Hancock, 2009*). It is implemented in two versions, a direct-touch two-handed interaction implementation and a tangible knob with an embedded trackball. Both approaches require quite different manipulation techniques.

While the tangible device only requires one hand, the touch-based control needs two hands. This device is specifically designed to explore collections and no other function is considered, such as indistinctly containing collections or single items. The part of the study dealing with data exploration focuses on the use of both versions in terms of perception of ease-of-use. It concludes that the tangible approach is easier to use for navigation and sharing information.

Hilliges et. al present a tangible rotatory tabletop device for browsing digital photo collections (*Hilliges, 2007*). It uses a novel helix representation for the photos, which are anchored around the device itself. A pen is used to support application-specific manipulations, such as image subsets. Photohelix is oriented towards time-based visualization, which allows for a sequential spiral-based visualization of elements, but there is no support for nested sub-collections consistently.

Stacked half-pie menus are presented by Hesselmann et. al for navigation in nested hierarchic data structures optimized for touch-based interaction on interactive tabletops (*Hesselmann, 2009*). This design is especially interesting because it is conceptually unlimited in terms of menu depth and breadth while maintaining menu form. It is controlled entirely by means of touch and turn gestures, with no need to use a pen, gloves or any other tool. It also tackles the multi-touch occlusion problem by using only half-pies and sticking them to one side of the tabletop. This approach, however, only allows for parallel touches if several leaf items are at the same level, otherwise, interaction will be sequential. The control is conceived as a non-collaborative (single-user) menu and it is fixed to a specific location.

In the Tangible Jukebox (*Gallardo, 2010*) music navigation and management is carried out by means of a pie-based widget controlled by tangibles and fingers. It uses physical cards to virtually store a music collection. When a card is placed on the table, a set of items is deployed around it, composing a virtual wheel that the user can spin to visualize all the menu-level content. Several set operations can be performed by combining specific physical cards accordingly.

These related works can be described in terms of features for a more systematic comparison. These features are broadly related to main aspects concerning the generality of the approach, data organization features, input methods and multiuser support. Table 10 shows a description of related works.

The *generic-purpose* aspect indicates whether the proposal is to be used in either a generic or specific purpose domain. The *data organization* aspect is concerned with features on how data are supported and organized. *Length* and *nested hierar-*

chy support are related to the capability for breadth and depth exploration, respectively. The length supported is reported at three levels: short (S), when a short collection of elements are intended; large (L), when a larger, though still limited, set of elements can be contained; and finally virtually unlimited (U), when the length could be considered large or even infinite. The nested hierarchy support simply indicates whether or not the proposal supports hierarchical collections consistently across several depth levels. The feature *massive collection display* indicates whether several collections can be displayed and manipulated at the same time. *Dynamic hierarchies* refers to the ability to establish hierarchies of collections dynamically, in contrast to only statically predefined collection hierarchies. *Spatial compactness* indicates whether the proposal includes a characteristic to deal with spatial limitations in its design. Finally, *visual layout* reports on the visual arrangement of the items in the collection.

Regarding the relevant features describing the input methods, *modality* describes the input modality of the techniques. It refers to the supported input modalities: purely multi-touch, purely tangible, combined or flexible hybrid. The difference between the combined and the hybrid input modality is that the former requires an ad hoc combination of modalities (touch and tangible) from the user, according to the actions to be performed with the widget. The latter supports the primary functionality in both modalities and the user selects the most convenient modality for a given input. Some proposals provide a control using a single input modality or even a combination of tangible and touch inputs to support the primary function of the control. Another possibility is to provide dual techniques separately, offering a multi-touch finger-based input and another using tangible. It is then necessary to distinguish the resemblance between these dual input techniques in the homogeneity of the observable behavior of the collection management system with respect to the input modality. To avoid unnecessary cognitive efforts, a high level of resemblance is a desirable feature.

Another important feature for the design of tangible inputs is whether the tangible device used has been specifically designed or whether it is a widely used tangible model (e.g. pucks or cards). This implies easier acceptance and adoption in existing platforms (*Fishkin, 2004*).

Another aspect of interest is multi-user support. This set of features determines whether the proposal was designed for multi-user contexts or for single-user scenarios. The 360° control design reports on whether the proposal being properly seen from different points around the tabletop. In addition to other features already commented in the data organization that can contribute to

multi-user support, such as massive collection, parallel selections indicates whether the proposal allows the selection of several items in the same collections in parallel. *Collection replication* refers to the ability to clone collections to be explored separately by several users or simply create two different views. *Flexible instantiation* shows whether the proposal supports a flexible mechanism to facilitate accessibility and instantiation of collections.

The comparative analysis of the reported studies reveals that almost all the proposals are designed for generic purpose domains and use a pie-based or similar visual item arrangement system around a center. Nearly half of the proposals provide support for short collections only, and most do not fully consider multiuser contexts. Consequently, they do not facilitate massive comprehensive collection displays, collection replication, or parallel selections. For input modality, the proposals normally rely on a combination of touch and tangible techniques and only one offers two control versions supporting exploration in both input modalities.

Based on the previous analysis, TangiWheel is conceptually designed to overcome the weaknesses of previous systems and provide full support for new features, including collection breadth and depth, as it is desirable to have both virtually unlimited length and nested collections. Moreover, the proposal should support massive collection use by multiple users or single-user manipulation of several collections at once. Compactness should also be considered together with other additional features to provide flexibility and ease of use, such as collection replication and 360° control view. Dual input techniques should be provided, resembling each other as much as possible. It would facilitate the emergence of a hybrid input scheme, instead of a simple combination of input modalities, allowing users to select the best input at manipulation time without any additional cognitive effort. Finally, the tangible device should be standard and widely accepted, instead of a specifically designed system for the application in hand.

	Data Organization			Input Methods			Multuser Support						
Work	Generic-purpose	Length	Nested Hierarches	Dynamic Hierachies	Compactness	Massive Collection	Visual Layout	Modality	Resemblance	Specifically Designed Tangible	Parallel Selections	Collection Replication Flexible Instantiation 360° control	
COR ² Ds (Shen et al)	y	s	y	n	y	n	Unsystematic (around a center)	Touch	n/a	n/a	y	n	y
Audiopad tech. (Patten et al)	y	s	y	n	y	n	Pie/linear floating	Combined	n/a	n	n	n	y
SLAPS (Weiss et al)	y	s	n	n	n	n	pie	Tangible	n/a	y	n	n	y
MTMM (Lepinsky et al)	y	s	y	n	n	n	Chording menus (around a center)	Touch	n/a	n/a	n	n	n
Tableball (Hancock et al)	y	l	y	n	n	n	radial (pie based)	Touch / Tangible	n	y	n	n	y
photochelix (Hilliges et al)	n	u	n	n	y	n	spiral (pie)	Combined	n/a	y	n	n	y
Stacked (Hesselman et al)	y	u	y	n	n	n	Half-pie	Touch	n/a	n/a	n	n	n
Jukebox (Gallardo & Jorda)	n	u	y	y	y	y	Pie	Combined	n/a	n	y	y	y
Tangiwheel	y	u	y	y	y	y	Pie	Touch / Pucks / Hybrid	y	n	y	y	y

Table 10. Feature based comparison between related work.

5.4.2 Design Discussion

In this section we present the detailed design and implementation of TangiWheel in terms of the previously introduced aspects and corresponding features.

5.4.2.1 General-Purpose Use

Data intensive applications normally focus on exploring, retrieving information and selecting data of interest. This is the case of applications aiming at constructing artifacts, which require exploring and selecting their component parts. For instance, when creating a technical sketch from existing parts for concurrent exploration and selection on several collections by multiple users, while others assemble the parts using multi-touch input techniques to speed up collaboration. Another possibility is the specification of reactive behavioral rules in simulation environments, requiring the selection of the events, operations and entities involved, possibly ordered by event type, operation type or entity type in nested collections. In the end, we aim to support interaction scenarios on tabletops requiring massive collection exploration, facilitating concurrent collections as a consequence of either several collections that a single user needs or that must be shared, replicated and operated by multiple users. As these scenarios can be diverse, control design should be as flexible, generic and easy-to-use as possible to suit such a wide range of user scenarios and requirements. TangiWheel has been successfully deployed to support multi-user creation of articulated entities subject to the rules of physics and also to support the collective creation of Rube-Goldberg machines as described in Chapter 3. In addition, it has been used to support the editing of dataflow based rules as explained in Chapter 4.



Figure 62. TangiWheel controls in action.

5.4.2.2 Data Organization

The main design decision of TangiWheel with respect to the data organization aspect was influenced by the need to support conflicting features (see Table 10) such as handling heavily populated collections and sub-collections in a compact way, i.e. without compromising the available tabletop space and thus allowing many instances to be present on the surface.

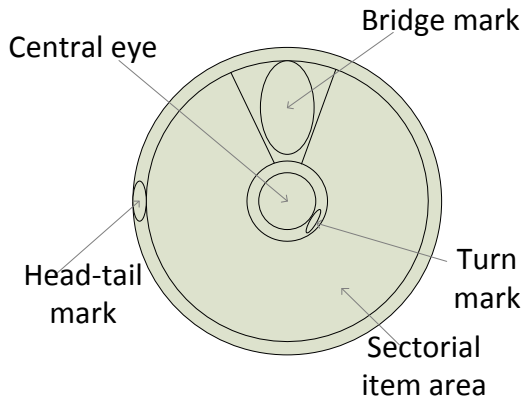


Figure 63. Regions in a TangiWheel instance.

In this respect, TangiWheel displays the visible items arranged in a circle following the pie menu model. We consider this appropriate since it is not only compact but also reduces seeking-time and selection errors on visible items by fixing the distance factor (Hopkins, 1987) and not pre-establishing orientation in a specific direction. An alternative design based on a linear arrangement, which is widely used in other technological platforms (i.e. mostly in desktop based metaphors for single-users) was not considered appropriate, even though it can provide an arrangement that people are more used to, and probably more in line with the way they usually read and consume written information. The fact that TangiWheel is also designed to support (see below) the concurrent manipulation of collections containing both textual and graphical elements in multi-user environments to motivate collective creativity contributed to this decision. Orienting the collection towards a specific user using a linear list would interfere with equal concurrent access to items in collections by multiple users in a tabletop setting. As illustrated in Figure 62, the control consists of items represented by text or icons, and marked areas for interaction or visualization.

TangiWheel displays only a subset of the elements so that the collection is compact no matter how many elements it contains. This also means that it has the same appearance whatever its length. To explore the invisible elements, TangiWheel has a special region, the bridge mark (see Figure 63), located in a sector of the widget in the form of a button which allows for virtually unlimited collections. Simply tapping the button switches between the exploration and rotation modes. In exploration mode, rotating the widget entails sliding the visible items in the pie, which means that some items are hidden below the bridge mark while others are shown in their place. To obtain maximum compactness when several collections are on the surface, a minimization state is supported in which all the contained elements are hidden. This state is reached/left by means of prolonged tapping on the bridge mark.

As we have a pie arrangement, we need to keep buttons or any other actionable elements near the center point without breaking the visual circular shape of the widget. Besides, placing the bridge mark in a sector makes the presence of a border region naturally represent the place where items are hidden. This design is also flexible, because it offers the possibility of including additional virtual buttons in this special region if it should become necessary to support further functions in the future.

Another visual feedback included in TangiWheel is a brighter point in the perimeter of the pie called head-tail mark (see Figure 63). Since the widget supports collections of unlimited length, the head-tail mark gives users a clue on what is the relative position of the displayed items within the overall collection, similar to the box provided in traditional scrollbars. Although not currently implemented, an additional improvement to this would be to show an expanded preview of several adjacent items near the bridge mark, to further help users in which direction they should explore. Another feature that could improve data visualization when handling very large collections would be the inclusion of a collection of recently-used elements around the bridge mark for direct access. These add-ons will be included in future versions of the widget. Finally, the turn mark provides information on how far the orientation of the control is from the rotation needed to show another invisible element.

Another important feature with respect to data organization is the hierarchical arrangement of collections so that unlimited numbers of nested sub-collections may be defined. Sub-collections may either be statically (at instantiation time) or dynamically created to support the dynamic coupling and decoupling of collections with an arbitrary number of nested sub-collections. Hierarchical collections can be created dynamically by simply dragging an existing collection

and releasing it into another one. The dragged sub-collection is inserted as an item in the section in which it is released. Once two collections are coupled, their relationship is shown graphically by means of a link (see Figure 62) when both are displayed, so that there is a visual feedback even when the collections are at a distance from each other on the surface. To decouple two collections the user may break the link by performing a zigzag movement over the link with a puck or a finger.

An expanded sub-collection is represented as a separate TangiWheel instance (see Figure 62) for several reasons. Firstly, it is an easy method of supporting collection replication and flexible instantiation by means of pucks (i.e., the ability of direct association), which allows any element (sub-collection or single terminal element) to be associated with pucks for direct-access. In TangiWheel a non-associated or empty puck may be placed over an item in a collection at any time. The puck then becomes associated with it and will contain either a single digital object or a TangiWheel sub-collection, depending on whether the item is or is not a leaf node in the collection. To dissociate a container puck, the user has to perform a simple zigzag movement. This mechanism gives direct-access to sub-collections in deeply organized nested structures and reduces seeking time when items have to be re-visited. This association mechanism may also be activated by placing an empty puck at the center of a virtual TangiWheel (displayed as a result of finger selections). Representing collections and sub-collections in the same way makes the previous association mechanism homogeneous from the point of view of the actions required from the user (simply placing an empty puck at the center of the ring or central eye).

Secondly, opening nested collections in separate TangiWheel instances keeps the same exploration process through hierarchical collections, and therefore the input methods designed for a root-level collection are the same as for sub-collections. Additionally, in situations with a large number of collections displayed on the surface, users may easily control orientation, location and visibility of the displayed sub-collections.

This gives users flexibility in deciding how the existing sub-collections are displayed on the available space. An alternative design to the one proposed in this work for handling nested hierarchies would be to expand the sub-collection around the selected item to obtain a more compact layout. However this option would compromise the achievement of other features. For instance, input techniques would have to change to support manipulations in each case and would affect the resemblance and consistency between techniques for root-level and nested collections. If compactness was the primary feature to be sup-

ported, then a better alternative would be considering in-place versions of the control. In this case the child collection would simply replace the parent, keeping most of the properties of regular collections while being operated in the same way. A future version of TangiWheel will be context-aware and will decide whether to display sub-collections by replacing in-place parent collections (on heavily populated surfaces) or to display them as separate TangiWheel instances, as in the current version of the widget.

5.4.2.3 Input Methods

In this aspect we consider some of the features related to input management when handling collections and the associated design solutions implemented in TangiWheel to cope with them. Two main features were considered here: modality and resemblance.

TangiWheel supports pure tangible, pure multi-touch or hybrid interaction modes. In tangible mode, a puck is tracked across the surface keeping control rotation and position consistently tethered in rotation mode. In exploration mode, rotating the puck changes the items currently visible. A pie arrangement integrates this interaction more intuitively, since exploring by rotating seems more natural in a circular layout rather than a linear one, in which displacements rather than rotations could be expected. If the puck leaves the surface, the widget disappears but reappears if the puck is repositioned. Items can also be selected by means of tangibles. The use of tangible pucks for selection is based on the idea of tangibles as hypercards, phicons and phandlers (*Ullmer, 1998*) for containment and transportation of digital elements.

In the case of completely virtual TangiWheels (i.e. those created by finger selections on non-leaf items or from specific hotspots), the preceding function is also supported in a multi-touch input modality. Rotation of the widget or exploration of invisible items is performed by a circular movement of the finger following the sectorial item area. An alternative to finger-controlled rotation would be a bi-manual technique using two fingers to describe opposite trajectories, as in (*Tuddenham, 2010*). Nevertheless, this technique would affect our resemblance criterion, since the manipulation of a single tangible only requires a single-handed interaction. However, in the adopted input action, in which rotating means exploring, it is very likely when exploring large collections that the rotary action will be continuous and steady to achieve several full 360° rotations of the collection. Therefore, while describing circles can be performed continuously and without lifting the finger from the surface, the bi-manual technique would require repeatedly re-starting the interaction as soon as both

fingers meet. It would make this alternative less effective for data exploration. Finally, in multi-touch modality, the closing action is performed by tapping on the central eye and dragging the widget to a new position on the surface.

5.4.2.4 Multi-user support

TangiWheel is designed to be used in scenarios in which multiple users may collaborate in the creative process by working together on a common creation space (the surface) and sharing multiple collections of elementary constituent elements used as building blocks to create complex composite entities. All the team members must therefore have equal access to the collections and a similar perception of them, no matter where they are located around the tabletop. Thus, a 360-degree control and reach mechanism (*MSUxGuidelines, 2011*) is a key feature in TangiWheel to make the widget re-orientable and usable by multiple users in a shared space. In addition, to facilitate user access from different points, techniques are provided to move and re-position widgets on the tabletop, unlike other approaches described above, in which either collections or sub-collections are attached to predefined fixed points.

The pie-based arrangement in TangiWheel avoids facing information towards a single fixed point. Nevertheless, if a user prefers a different orientation of the widget with respect to his/her location, he/she may change the widget from exploration to rotation mode, so that rotating entails the rotation of the graphic control: visible items will still be visible but the whole control will face in a new direction. Visible items simply adopt a new position, keeping their location with respect to each other while facing in a new direction. Moreover, some of the mechanisms that were previously discussed are powerful characteristics that facilitate access in multi-user scenarios. These mechanisms are: relocating the widget at any point on the surface; the association of pucks to items and collections that can be removed and replaced on the surface wherever needed; the support of nested hierarchies and the parallel manipulation of replicated collections.

5.4.2.5 An example of interaction

As examples of the proposed techniques in action, Figure 64, Figure 65 and Figure 66 illustrate the interactions required to select an item from a collection and take it to a specific sensitive area on the surface. The performance is explained in terms of user actions (Ux) and system responses (Sx) as follows:

S0: The application shows a target item to be selected representing a Soccer ball.

U1: User places an empty tangible puck on the surface.

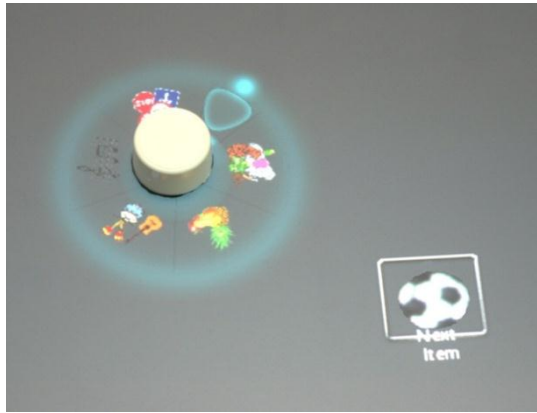


Figure 64. Interactions 0 and 1. To select the *Soccer ball* item (S0), a tangible is first put on the surface (U1) after which a category menu is shown (S1).

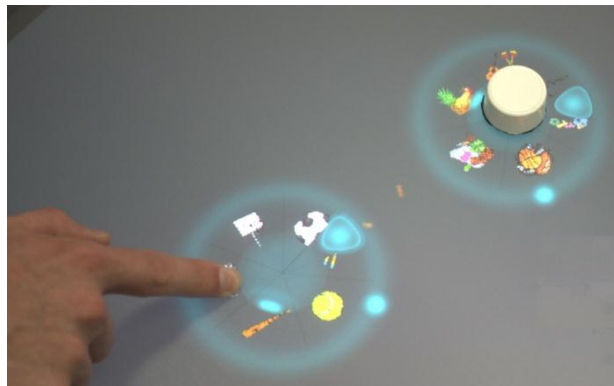


Figure 65. Interactions 2, 3, and 4. The user has enabled the exploration mode (U2) and has rotated the puck until the *Sports* category has been displayed and tapped on the item (U3). The user explores the *Sport* submenu by dragging his/her finger over the items (U4).

S1: The system deploys a collection of categories implicitly associated with the puck. The control will therefore respond to puck rotations and movements as described above.

U2: As the Sports category is not visible the user needs to explore the collection. To do so he/she taps on the bridge mark.

S2: The menu toggles to the exploration mode, since it was previously in rotation mode.

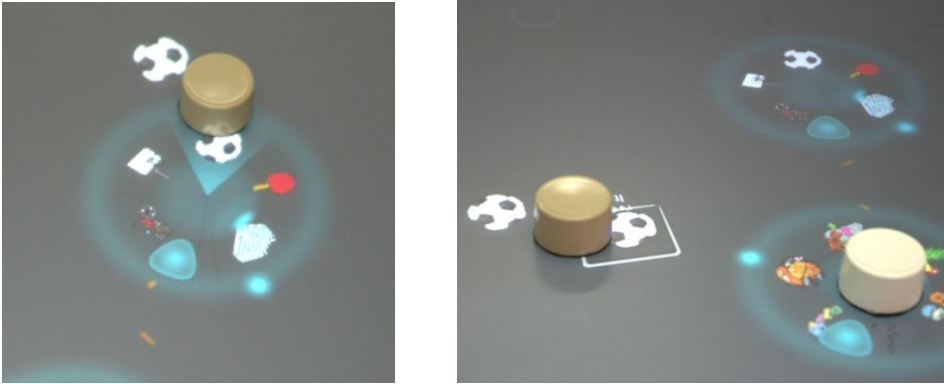


Figure 66. Interactions 5 (*left*) and 6 (*right*). The *Soccer ball* is selected by using a tangible (U5) and it is then brought to the target area (U6).

U3: The user rotates the puck, changing the visible items on display and taps on *Sports* when it appears.

S3: In response to this selection, since the item is non-leaf, a completely virtual TangiWheel is displayed to explore the hierarchical sub-collection. In this example the sports collection is displayed, containing items such as *tennis ball*, *bicycle*, and so on.

U4: Since this sub-collection has been selected by a finger, it will have to be manipulated by a finger unless a puck is explicitly associated to the widget to turn to tangible interaction. The user can use either method to find the Soccer ball.

S4: The items being displayed change as the user explores the collection.

U5: To perform the selection the user decides to use tangibles. He/She puts a puck on the item.

S5: The item is then virtually linked to this tangible.

U6: The puck containing the Soccer ball item can be placed over the target area to accomplish the goal.

5.5 Experimental Evaluation of TangiWheel

Multi-touch and tangible surface interfaces have become popular in recent years. While multi-touch tabletops allow collective and multiple interaction with virtual objects through finger contact on the surface, tangible tabletops expand multi-touch input capabilities by using specially designed tangible objects. These interaction mechanisms provide more intuitive performance and interaction that would become especially relevant in collaborative human oriented tasks such as in learning (*Rick, 2011*) or in those collaborative tasks that are highly cognitive demanding and stressful such as in emergency management (*Engelbrecht, 2011*). Several innovative applications such as Reactable (*Jordà, 2008b*), IntuPaint (*Vandoren, 2008*) or Multi-Touch VirtualGlobe (*Schöning, 2008b*) have been developed in the research community to demonstrate the new capabilities of tabletop interfaces.

The growing success of these new interaction mechanisms has also fostered a number of comparative studies. In one of the seminal studies, Fitzmaurice & Buxton conducted an empirical comparison of spatial-multiplexed versus time-multiplexed input schemes (*Fitzmaurice, 1997*). The spatial-multiplexed conditions used specialized input devices, such as a rotor, a brick, a stretchable square and ruler, as well as several generic puck and brick sets, whereas the time-multiplexed condition only used one puck and brick. The study concludes that spatial-multiplexed conditions are superior to time-multiplexed input schemes in terms of tracking error.

In the same line of research, another experimental comparison was carried out by Tuddenham, who explored the benefits of Tangible User Interfaces (TUIs) versus multi-touch input, measuring the performance in terms of time and tracking error (*Tuddenham, 2010*). Since the explored TUI elements can be used in tangible tabletops, many of their benefits could be applied to tangible surfaces, given that these are a specific TUI type. The comparison explores simple but common basic interface actions found in many multi-touch and tangible surface interfaces focused on acquisition and manipulation issues. This is an important contribution because it provides designers with the information needed to decide between TUIs and multi-touch input on interactive surfaces. It also assesses the value of the tangible element to the user in this type of system, especially in the case of the TUI elements explored in the study, whose physical shape and size perfectly match the characteristics of the virtual counterparts.

Another empirical study of interest is reported in (*Lucchi, 2010*), which also compares touch-versus-tangible input in a tabletop, but focuses on general manipulations on box-like elements, such as walls and shelves. The study reports on an experiment which asked subjects to reproduce models of walls and shelves as fast and accurately as possible. In this task, requiring spatial layout on a tabletop, tangible input took much less time but it was slightly less accurate than touch.

However, despite these research and development efforts, further studies are needed to obtain a better understanding of interactions with more complex types of controls to enable the development of applications that address the more challenging user tabletop requirements. Among the many interactions that will have to be addressed on tabletops, there is a subset of special interest to us: the manipulation of collections. Thus, a study of the more complex interactions related to collection manipulation for both existing interaction styles, multi-touch and tangible, is a necessary step forward with respect to previous comparative studies that only take into consideration simple widgets and actions.

In this study, we present an empirical comparison of pure multi-touch input versus tangible-input along with a hybrid input condition using our design of TangiWheel. The final goal is thus to study the effectiveness of the supported interaction styles incorporated in our TangiWheel proposal as a validation of the design decisions made.

Of the different aspects considered in the design of TangiWheel - data organization, input methods and multi-user support - we focus on the experimental evaluation of the techniques used in each input modality. The following aspects are considered in the present study:

- Acquisition and basic manipulations typically performed to establish the location and orientation of collections.
- Selection of a sequence of items over a range of category collections.
- Series edition (composed of items) that requires not only exploring collections but also insertion or deletion of items.

The overall goal of our experimental design was to study the effect of input method, i.e. fingers or handlers (pucks), on performance in manipulating collections in terms of performance time and number of manipulations required. Experimental complexity was gradually increased to evaluate the use of the widget under different circumstances. TangiWheel is the first tabletop-oriented widget for manipulating collections that supports a hybrid input method with

high levels of resemblance. It is therefore important to obtain experimental evidence on what types of situations would benefit from this new type of hybrid input method with respect to pure multi-touch or pure tangible in the context of our proposal.

5.5.1 Participants

Twenty-three volunteers participated in our study, 16 male and 7 female, mostly undergraduates or Ph.D. students from our university. One participant was left-handed and two were ambidextrous. Ages ranged from 21 to 40 ($M=27.5$, $SD=5.23$). Eighteen participants reported using a personal computer every day, three almost every day, and two reported using one once or twice per week. Thirteen participants stated they were regular users of touch-enabled devices, whereas ten had seldom or never used one. Eight participants had not had any previous experience of surface computers and the remainder had had limited experience, mostly at shows and exhibitions.

5.5.2 Equipment

Several testing applications to cover the experimental tasks were developed using the controls presented in the previous chapter. Two Microsoft Surface units were used to conduct the experiments.

5.5.3 Method

Test sessions were arranged according to participants' availability. In order to avoid participants learning by observing peer-actions, only two people were in the laboratory at the same time but at two different surface units separated by a folding screen. The participants received an introductory talk, a live demonstration of typical interactions, followed by free, although supervised, interaction training. This procedure took about 15~20 minutes until the participants felt comfortable enough with the interaction mechanisms. The study itself began with each participant on his/her own performing the experiments on acquisition and basic manipulations, followed by the selection experiments and the series-edition experiments. In order to avoid all participants interacting with the different alternative designs in the same order, the interaction input modality was established according to a Latin Square design. Participants were encouraged to perform as fast as possible with a reward for the fastest participant.



Figure 67. Interaction in the acquisition and basic manipulation task: tangible condition.

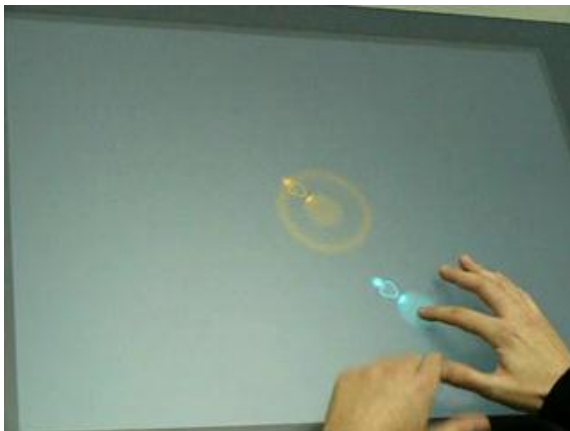
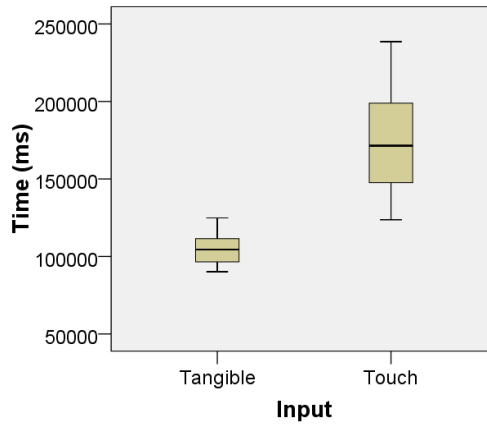
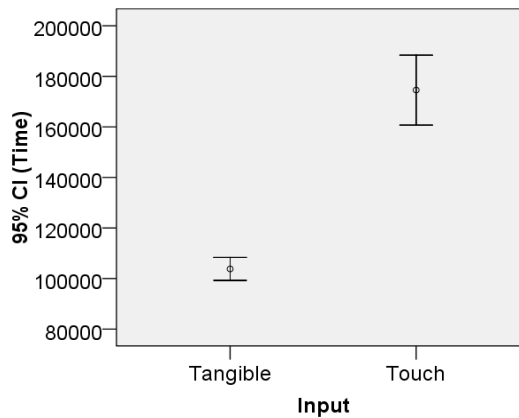


Figure 68. Interaction in the Acquisition and basic manipulation task: Touch condition.

Figure 69. Box-plot for CT_{BAS} Figure 70. Confidence Interval plot for CT_{BAS}

The task execution-support software automatically recorded all the input interactions in a log file. The set of logs were post-processed to obtain information for the quantitative statistical analysis. The main information extracted was related to task completion times and the corresponding number of actions required. Different parts of each session were video recorded for further analysis, basically to study participants' behavior patterns. Several questionnaires were also filled in by the volunteers to assess ease of use and effectiveness.

5.5.4 Experiment 1: Acquisition and Basic Manipulation

At least two interaction phases are involved in the use of an interface element (*Fitzmaurice, 1997*), typically, acquisition and manipulation. In the TangiWheel widget, acquisition and basic manipulation tasks are necessary to establish the position and orientation of the control before exploring and selecting an element. The goal of this experiment was therefore to explore the effectiveness of each input method in performing the typical preliminary interactions.

5.5.4.1.1 Task

Participants are requested to establish the position and orientation of a series of 25 TangiWheel widgets. An orange target with the desired orientation is displayed in the center of the tabletop. A single green TangiWheel widget appears in a predefined pseudo-random position (see Figure 68 and Figure 67). The user has to acquire and perform basic manipulations to match the position and orientation of the widget with the target. When they match, the widget disappears, the target takes on another orientation and a new widget appears in another location. The series of positions and orientations are predefined but the order of appearance is shuffled for each run.

5.5.4.1.2 Procedure

This experiment was performed by each participant twice, once using the tangible input method and once using fingers only. The input method was assigned to each participant according to a Latin Square design in order to avoid order effects. In the Tangible method, the participant had to take the tangible in one hand, acquire the widget by putting the tangible in its inner region in order to make the binding, and then take it to the target position, performing all the required rotation adjustments. Matching with the target was evaluated when the tangible remained still for about 250ms. In the Touch method, the participant had to accomplish the positioning by dragging the widget from its inner region and adjust orientation by describing circles on the widget with a finger. The evaluation was made when the finger released the widget. The participant had to establish the position of each widget as quickly as possible according to the instructions.

5.5.4.1.3 Results

Three variables were measured for each input method: time to complete the experiment CT_{BAS} , time to accomplish a single matching T_{BAS} and number of actions needed to complete each matching A_{BAS} . The superscripts *TOUCH* and *TANGIBLE* were used with each variable for notation purposes.

As can be seen in Figure 69 and Figure 70, on average, the participants took longer to complete the task in the Touch condition CT_{BAS}^{Touch} ($M=174.56s$, $SD=31.210$) than the Tangible $CT_{BAS}^{Tangible}$ ($M=103.799s$, $SD=9.714$). The analysis of variance (ANOVA) conducted on the time needed to complete the experiment demonstrated that the input method has an effect on the preliminary acquisition and basic manipulations ($F= 94.327$, $p=.000$) and that the tangible input outperforms touch. This result agrees with other empirical studies involving repositioning of different interface elements on both multi-touch and tangible interfaces (Tuddenham, 2010)(Lucchi, 2010), which showed that tangible interfaces are effective in spatial layout tasks. Our experiments therefore provide additional empirical evidence on this issue with respect to our techniques for repositioning and reorienting the widget.

In this respect, the result can be explained by the fact that hands are used to rotate and position objects in many everyday actions (grasping) rather than single fingers, whose primary function is to touch (tapping). Grasping objects is therefore a more natural interaction for moving objects than dragging them across a surface with the fingers, considering the techniques involved. Moreover, the Touch condition suffered from an interaction issue already observed and reported in (Tuddenham, 2010) as *exit error*.

This refers to the difficulty of disengaging from the virtual object without causing some form of unintended extra movement. This is an inherent problem in touch input on tabletops that may affect users to different degrees. It was seen to affect the correct positioning of widgets and in the end contributed to considerably higher times in the Touch condition.

A study was made of the time needed to take the widget to each target and the number of actions required. Matching single targets took more time with Touch T_{BAS}^{Touch} ($M=5.861s$, $SD=2.024$) than Tangible $T_{BAS}^{Tangible}$ ($M=3.772s$, $SD=0.930$). However, fewer actions were required using Touch A_{BAS}^{Touch} ($M=3.45$, $SD=2.53$) than Tangible $A_{BAS}^{Tangible}$ ($M=7.17$, $SD=2.43$). This can be explained by the fact that a complete rotation of the widget can be performed continuously with a finger describing circles on the surface, which counts as a single action. However, when the same action is performed by a puck, the continuous rotation cannot be reproduced, so that every movement of the hand counts as an additional action.

Since the assumption of normality was not met, a comparison was made using Mann-Whitney tests. Significant differences were found in the time taken to accomplish a single target ($H_0: T_{BAS}^{Tangible} = T_{BAS}^{Touch}$; $z=-19.06$, $p\text{-value}=.000$)

and also in the number of actions required ($H_0: A_{BAS}^{Tangible} = A_{BAS}^{Touch}$; $z=-22.46$, $p\text{-value}=.000$).

In view of these results, although it could be thought that our finger-touch mechanism is not a suitable interaction technique for TangiWheel, this premature conclusion will be questioned in Experiment 2.

5.5.5 Experiment 2: Match to Sample (MTS)

Visual matching to sample is a classical technique in cognitive studies (*Maydak, 1995*) in which a visual stimuli is produced and remains on display until the subject makes a selection. The aim of this experiment was to compare the effectiveness of different TangiWheel input methods when searching for and selecting elements in different collections in response to a visual stimulus.



Figure 71. Interaction in the MTS task: Tangible condition.

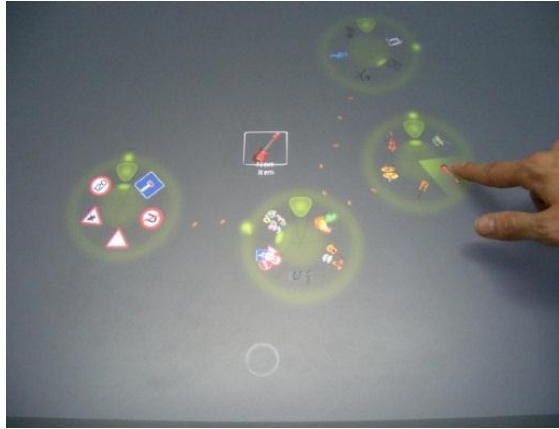


Figure 72. Interaction in the MTS task: Touch condition.

5.5.5.1.1 Task

The visual stimuli were pictures belonging to the following categories: numbers, instruments, animals, fruits, musical notes and road signs. Participants had to sequentially match a total of 20 elements belonging to different collections. On average each collection contained eleven items. They were encouraged to react as quickly as possible by searching for the target element in the displayed categories. Once a selection was made correctly, the next element was displayed.

5.5.5.1.2 Procedure

Each participant performed three runs to vary the input method: touch, tangible and hybrid. The order of the runs was arranged according to a Latin Square design to avoid order effects. While the hybrid input method allows the use of pucks and fingers in any combination, following the full functionality of the TangiWheel widget, the tangible and touch conditions only responded to pucks or fingers, respectively. This means that in the tangible condition the pucks are used both to open collections and to select items whereas in the touch condition explorations and selections are made with fingers only (see Figure 75 and Figure 76).

5.5.5.1.3 Results

The application of ANOVA to the experimental times demonstrated that they are significantly influenced by input method ($F= 15.512$, $p=.000$). Post-hoc

pairwise comparisons using t-tests (Bonferroni corrected) showed that all three conditions were significantly different (see Table 11). The results show that the Touch input condition performed best, followed by the Hybrid and Tangible.

We also analyzed the time needed to complete a single target item selection and the number of actions required. Figure 73 shows the mean times for each condition. Using fingers only gave better results than the other two input conditions, which are enabled to use pucks. Figure 74 shows that the Touch condition also required the lowest number of actions, followed by Hybrid and Tangible.

Comparison	Mean differences	p-value
Hybrid - Tangible	-19662.57	.013
Hybrid - Touch	16804.16	.041
Tangible - Touch	36466.73	.000

Table 11. Post-hoc pairwise comparisons using t-tests (Bonferroni corrected) for time required to complete the MTS task.

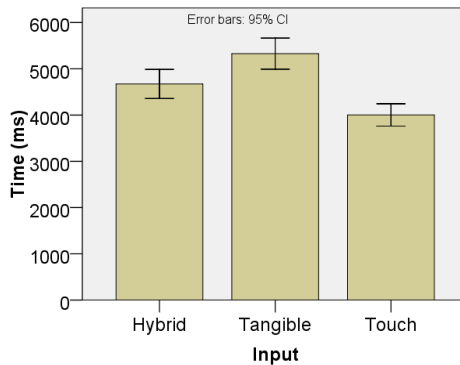


Figure 73. Mean plot for times T_{MTS}

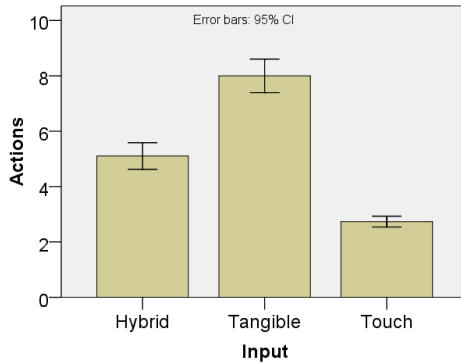


Figure 74. Mean plot for the number of actions A_{MTS} .

As the normality assumption was not met, a pairwise comparison was conducted on the time to complete a single target selection and the number of actions needed using Mann-Whitney tests to determine whether the differences were significant. As can be seen in Table 12, the tests showed there were significant differences between all three input conditions for the two measured variables.

The results indicate that the number of actions and the average time needed to search and select an individual element are also significantly lower for the Touch condition and therefore suggest that fingers are more effective in scenarios requiring focused selections from a collection at a fixed position with no basic manipulations (rotations and translations) on the surface. This technique would therefore be suitable for single users working on interfaces with collections displayed at predefined fixed locations in the work space and a predefined 2D orientation of the interface.

H₀	Z	P
$T_{MTS}^{Hybrid} = T_{MTS}^{Tangible}$	-3.089	.002
$A_{MTS}^{Hybrid} = A_{MTS}^{Tangible}$	-9.488	.000
$T_{MTS}^{Hybrid} = T_{MTS}^{Touch}$	-2.388	.017
$A_{MTS}^{Hybrid} = A_{MTS}^{Touch}$	-5.932	.000
$T_{MTS}^{Tangible} = T_{MTS}^{Touch}$	-5.588	.000
$A_{MTS}^{Tangible} = A_{MTS}^{Touch}$	-16.882	.000

Table 12. Mann-Whitney tests to compare T_{MTS} and A_{MTS} .

However, in applications in which different subjects have to acquire and bring low-item collections to their personal space, the tangible input method would outperform touch because the number of acquisition actions clearly exceeds

the number of explorations in this case. Finally, in scenarios with a relatively high number of acquisition actions and large collections, a hybrid approach would integrate the best features of both interaction modalities. Tangiwheel supports all three modes and, unlike those approaches described above, can therefore be considered a flexible widget that can be effectively used in a wide range of scenarios.

5.5.5.1.4 Hybrid Performance Results

Regarding the performance profile on the Hybrid input, Table 13 summarizes the use of fingers and pucks for several important actions in terms of average number of actions per user and overall percentage. The results show that the subjects generally preferred fingers (72.5%) to pucks (27.5%), when given the choice. Fingers were mostly used to select and establish the target (82.75% and 97.23% respectively). Evidence was obtained from the video recording that subjects had difficulties when using a puck to select an item on a TangiWheel widget.

Action	Finger		Puck	
	Avg.	%	Avg.	%
Manipulation	41.60	72.5	15.78	27.5
Menu instantiation	4.04	38.75	6.39	61.25
Selection	25.86	82.75	5.39	17.24
Target establish.	21.30	97.23	0.61	2.77

Table 13. Use of fingers/pucks in the hybrid input condition on MTS task.

However, the results also show that the subjects made greater use of pucks to create menus (61.25%) that were then explored tangibly. This suggests that they found it easier to explore. On average, each participant used almost 6 different pucks ($M=5.7$) to complete the task, suggesting that they made extensive use of them and did not limit the number of pucks used simultaneously.

5.5.6 Experiment 3: Series Edition

Experiments 1 and 2 evaluated performance in basic manipulation activities under different interaction conditions and the location and selection of elements in structured collections to match individual samples drawn from a series. In other words, there was a one-to-one relationship between the visual stimulus and the target element being manipulated when performing a simple action (rotation, translation and selection). However, to fully explore the possible influence on performance of the different TangiWheel input conditions, an experiment was designed with a higher number of samples and possible

actions. Our main goal was to increase the cognitive overload when manipulating a collection under different conditions and to involve the user in a complex task that requires the combination of rotation, translation, search, selection, insertion and deletion of elements in collections. This allows us to simulate an authentic user scenario by increasing cognitive complexity while maintaining a controlled experimental environment in which a comparative study can be made of the different input modalities.

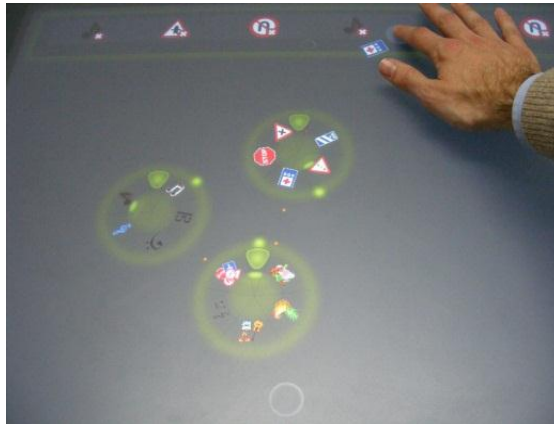


Figure 75. Edition of series in Touch condition.



Figure 76. Edition of series in Tangible condition.

5.5.6.1.1 Task

Participants were requested to either create a series defined as a sequence of elements belonging to different collections, or to modify an existing one. For this purpose, they had to explore existing collections and add or remove elements to/from the series.

5.5.6.1.2 Procedure

Three runs of this experiment (Tangible, Touch and Hybrid) were conducted (see Figure 75 and Figure 76). A Latin Square design was used to avoid order effects. Participants were individually given an instruction sheet and were told to create two new series and modify an existing one. The proposed series for each interaction style were of similar difficulty in terms of length and variability of the items included. In the Touch condition the subjects had to search for and select items in TangiWheel collections using their fingers on virtual TangiWheel widgets, which may be rotated or dragged across the surface by using touch, as in the previous experiments. To insert an element in a specific position of the reference series they had to drag the element from its collection and drop it in the appropriate position in the sequence, without raising their finger from the surface. To remove an element from a series a deletion button is displayed next to each item. On the other hand, to add an element to the series in the Tangible condition the subjects had to use pucks to search and select items. Since the elements were arranged in several nested categories, they could at any time associate a puck with a category for direct access and use the puck to rotate or move the collection to any area on the surface. Elements were selected by placing a puck over them and were inserted in the proper position in the series by releasing the puck in the desired area. To remove an element from a series, a deletion puck was placed on top of the element.

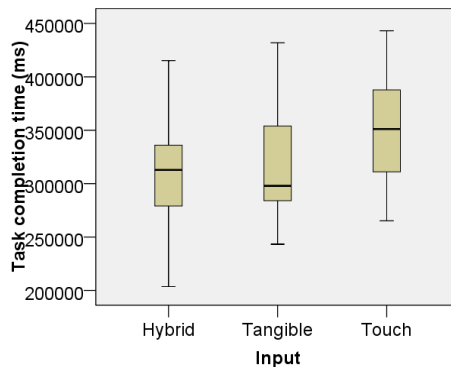


Figure 77. Box-plots for CT_{SER}

Comparison	Mean differences	p-value
Hybrid - Tangible	-11892.487	1.000
Hybrid - Touch	-44728.876	.018
Tangible - Touch	-32836.389	.103

Table 14. Comparisons for the time required to complete the series task.

5.5.6.1.3 Results

Figure 77 and Figure 78 show the distribution of the time required to complete the experiment in each condition and the times needed to complete the edition of a single series. The application of ANOVA to the times needed to complete the experiment demonstrated that this is affected by the input method ($F=4.413$, $p=0.016$).

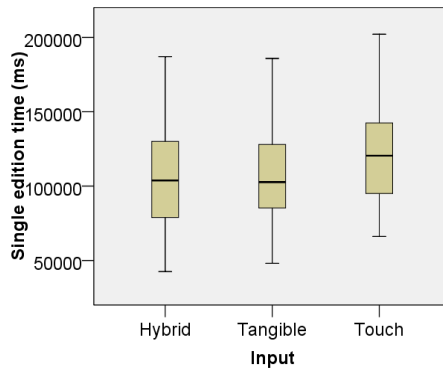


Figure 78. Box-plots for T_{SER}

Post-hoc pairwise comparisons using t-tests (Bonferroni-corrected) showed that Touch is significantly slower than Hybrid, but found no significant differences between Touch and Tangible or between Tangible and Hybrid (see Table 14). Nevertheless, the mean time needed to complete the task was higher for Touch and Tangible than Hybrid, and Touch higher than Tangible.

We also analyzed the time needed to complete the edition of a single series and the number of actions it required. Figure 79 and Figure 80 show the corresponding mean plots. The application of ANOVA to the time needed to complete a single edition demonstrated that this is affected by the input method ($F= 4.503$, $p=.012$).

Post-hoc pairwise comparisons using t-tests (Bonferroni-corrected) showed that Touch is significantly slower than Hybrid. Touch also fared worse in the

comparison with Tangible and is close to significance ($p=0.053$ at 5% CL). The tests showed no significant difference between Hybrid and Tangible (see Table 15).

The application of ANOVA to the number of actions required showed that this parameter is affected by the input method ($F=60.269$, $p=.000$). Post-hoc pairwise comparisons using t-tests (Bonferroni-corrected) showed that all three conditions were significantly different (see Table 16). Touch required a significantly lower number of actions than Tangible and Hybrid conditions.

Comparison	Mean differences	p-value
Hybrid - Tangible	-2153.072	1.000
Hybrid - Touch	-15611.345	.018
Tangible - Touch	-13458.273	.053

Table 15. Comparisons for T_{SER} .

Comparison	Mean differences	p-value
Hybrid - Tangible	-22.253	.002
Hybrid - Touch	46.350	.000
Tangible - Touch	68.603	.000

Table 16. Mean plots for A_{SER} .

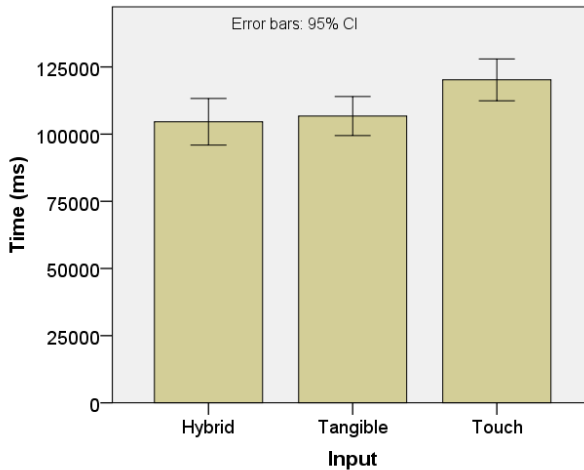


Figure 79. Mean plots for T_{SER}

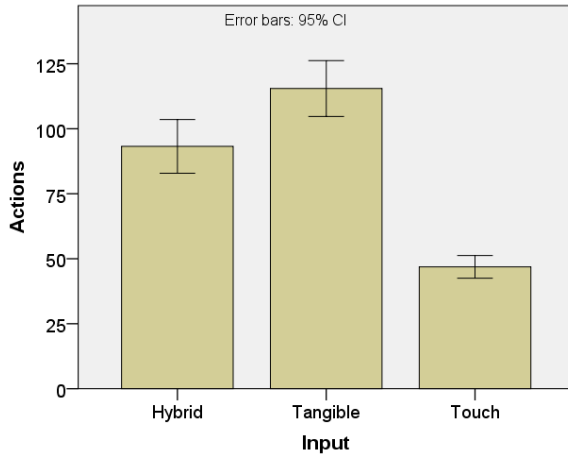


Figure 80. Mean plots for A_{SER} .

The results confirm that the Hybrid approach provides more effective task completion in terms of the time required than the other two approaches, especially when comparing Hybrid with Touch. This means that the Hybrid interaction is the most effective for more complex tasks than the one described in the MTS Experiment. This is due to the fact that editing a series requires a combination of the actions studied in the two previous experiments. When this situation arises, the poor performance of the Touch interaction style for basic manipulation tasks (dragging and rotating) makes this approach less effective, even though it shows good behavior for searching and selecting items in collections.

In fact, the analysis of the recorded videos reveals two important situations in this respect. Firstly, in both the Hybrid and Tangible conditions the subjects benefited from using pucks as element containers and consistently used a single container puck for multiple insertions of elements that appeared repeatedly in a series. On the other hand, those using the Touch condition had to drag the element across the surface several times to insert it in different parts of the list. This penalizes the task completion time because, as shown in the first experiment, the Touch condition is less effective for basic manipulation activities. Secondly, we also observed that, because the number of elements in a series was high, the number of TangiWheel instances displayed on the surface increased over time, forcing those using Touch to relocate them so that they would not interfere with the task. Again, the need to relocate the TangiWheel instances by touch makes the overall task less effective.

We may conclude from the results obtained that hybrid designs allowing both Tangible and Touch interaction styles outperform those based on touch or tangible-only styles in situations requiring: movement of collections on surfaces that have a great number of elements displayed simultaneously; the re-use of elements in collections in different areas on the surface; and re-orientation of the containers that hold the collections. These requirements are certainly present when several participants share the working space or in situations that require flexible collection layouts. The better performance of the Hybrid interaction style in these situations is important, given that surfaces are mainly designed to support touch-based interactions.

5.5.6.1.4 Hybrid Performance Results

As shown in Table 17, the results of the Hybrid input condition showed that pucks were used almost to the same extent as fingers. In contrast to the MTS task results, selection operations with pucks (69.2%) and menu instantiation using fingers (61.44%) were preferred. In addition, items were inserted in the series mostly by tangibles. On average, users used 6-7 different pucks to complete the task ($M=6.48$).

All these results confirm the evidence obtained from the videos. Tangible techniques perform better with increased task difficulty and overload. As the surface easily becomes cluttered and previously selected and inserted items are likely to be reused several times, pucks provide advantages as they are easier to handle in these situations. The subjects appeared to get overworked as the task progressed when dragging items repeatedly with the fingers.

Action	Finger		Puck	
	Avg.	%	Avg.	%
Manipulation	78.26	51.18	74.65	48.82
Menu instantiation	8.52	61.44	5.35	38.56
Selection	9.87	30.80	22.17	69.20
Item insertion	5.22	13.82	32.52	86.18

Table 17. Use of fingers/pucks in the hybrid input condition on series task.

5.5.7 Questionnaire Results

After the participants had finished each task, they were asked to complete questionnaires on ease of use and usefulness. The 5-point Likert scale questions regarding selection and series tasks were designed to compare the three input methods evaluated by pairwise Mann-Whitney tests.

Table 18 contains the statements that were scored by those who participated in the tests to questions in the basic questionnaire. Figure 81 shows the mean plots of the average score given.

In general, the three methods were assessed positively and no design issues were raised from the answers. For instance, the subjects found the collection exploration techniques easy to remember in all three methods (Q1), with no significant differences between methods. They also considered that the exploration interaction techniques were clear and intelligible (Q2). They found hybrid control techniques to be the most intuitive (Q3). These answers showed that the subjects anticipated the behavior of tangibles when performing natural actions, such as acquisition, rotation and translation, whereas in Touch more complex actions were necessary and therefore required a higher cognitive effort. They considered the hybrid design enabled them to select the most intuitive interactions from the tangible and multi-touch approaches.

Q	Questions: I consider that...
Q1	the mechanisms to explore collections are easy to remember.
Q2	the interaction is clear and intelligible.
Q3	the control for collections used is intuitive when handling.
Q4	the mechanism to interact with collections is useful.
Q5	the mechanism to interact with collections is flexible.
Q6	the way of interacting with collections is novel.
Q7	the metaphor for deletion of items is easy to understand.
Q8	the metaphor for deletion of items is easy to use.
Q9	the metaphor for insertion of items is easy to understand.
Q10	the metaphor for insertion of items is easy to use.

Table 18. Statements scored by participants in the basic questionnaire.

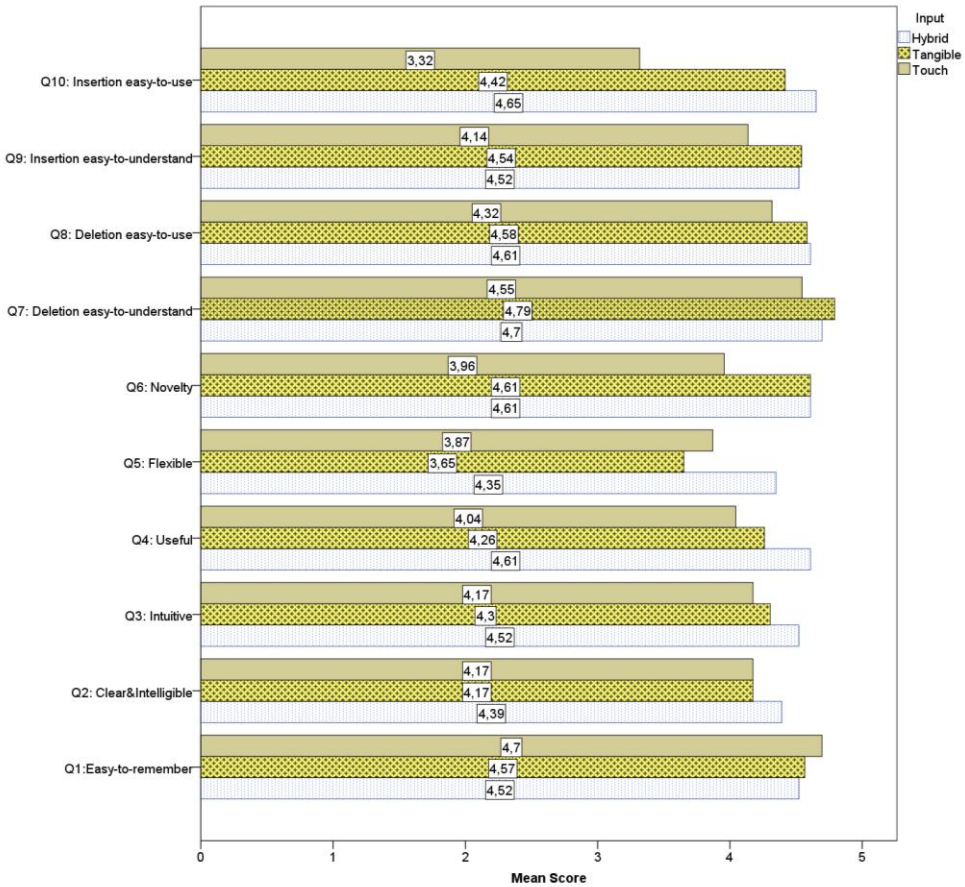


Figure 81. Mean plots for the basic questionnaire.

Interacting with collections was found to be mostly useful in the three conditions (Q4). However, Hybrid was given a significantly higher score ($z=-2.896$, $p= 0.004$). This result suggests that users perceive as most useful the possibility of selecting the most appropriate input condition according to the task at hand.

There was some discrepancy on method flexibility and novelty. Hybrid was considered more flexible and Tangible more cumbersome (Q5). The differences between Hybrid and Tangible were highly significant ($z=-2.625$, $p=0.009$) while between Hybrid and Touch they were close to significance ($z=-1.839$, $p=0.066$). These results suggest that the proposed combination of touch and tangible styles is an advance in flexibility.

Q	Questions	Strongly disagree	Disagree	Neutral	Agree	Strongly Agree
HS1	Tangibles being able to contain items is advantageous	0	2/8.7%	3/13%	8/34.8%	10/43.5%
HS2	I prefer using virtual components when possible	0	5/21.7%	13/56.5%	5/21.7%	0
HS3	I prefer using tangibles when possible	0	6/26.1%	8/34.8%	8/34.8%	1/4.3%
HS4	Using TangiWheel in Hybrid modality improves my effectiveness in selecting items	0	1/4.3%	2/8.7%	6/26.1%	14/60.9%
HS5	Using TangiWheel in Hybrid modality improves allows me selecting items quicker	0	1/4.3%	0	8/34.8%	14/60.9%

Table 19. Summary on agreement with specific statements in the Hybrid condition. Participant counts and percentages for each agreement level and question are summarized.

They also considered both Hybrid and Tangible to be equally novel (Q6). Touch was thought to be significantly less novel ($z=-2.410$, $p=0.016$), which means tangible components increase the perception of novelty.

Questions Q7-Q10 in Figure 81 depict the mean scores for questions related to ease of understand and use of the metaphors for inserting and deleting elements. While the metaphors and techniques involved in deletions were considered easy to understand and use in all the input conditions (Q7 and Q8), significant differences were found to exist in insertions. The insertion metaphor in Touch is significantly harder to understand than Tangible according to the test on Q9 scores ($z=-2.034$, $p=0.042$). Also, the insertion operation in Touch mode is significantly more difficult to handle (Q10) than that in Hybrid ($z=-4.333$, $p=0.000$) and Tangible ($z=-3.675$, $p=0.000$) modes. The metaphor associated with obtaining, lifting and releasing a tangible container to insert an item at a specific section within a collection was better understood than having to drag items across the surface with a finger.

Some additional questions were specific to Hybrid selection (HS) in order to assess preferences and perceptions.

Table 19 shows the statements assessed and summarizes to what extent subjects agreed with the statements. The results in general are in favor of the hybrid method. Firstly, the capability of tangibles to contain items was rated positively. Secondly, they were slightly in favor of using tangibles when asked about

personal preferences. Thirdly, they felt that using TangiWheel in hybrid mode clearly improves both effectiveness and productivity when selecting items.

At the end of the session, a final questionnaire (FQ) was administered consisting of single-choice questions which explicitly asked which input condition, if any, was perceived as most appropriate in performing specific actions.

As results show (see FQ1, FQ2, and FQ3 in Table 20), basic manipulations were clearly considered easier with Tangible. They found it easier to perform explorations using tangibles (FQ4). This also includes Hybrid, since many collection instantiation operations in the MTS task involved the use of pucks.

Tangible interaction was seen to be easier than Touch interaction for moving an item from a collection to a target (FQ6), as required in insertion operations. For deletions (FQ7), Touch was given higher preference. They considered that grasping a puck just to perform a deletion interrupted the sequence of operations. However, the overall editing process was considered to be easier in Hybrid, closely followed by Tangible (FQ8). None of the subjects chose Touch, which suggests that dragging items through cluttered conditions is to be avoided when designing a multi-touch tangible user interface. This finding also supports the idea that more complex and general scenarios, with a range of different actions such as exploring, selecting, dragging, etc., are better suited for hybrid interaction styles.

The participants felt that Hybrid allowed them to perform selections more quickly (FQ10), probably as a consequence of the combination of exploring with pucks and selecting by fingers, as had been shown by the detailed analysis. Hybrid was perceived as effective for editing series (FQ11), although not as fast as Tangible, as it could be expected. Finally, the Tangible memory capacity was a positive feature, as repetitive manipulations were clearly considered more effective in the Tangible approach (FQ12).

Q	Questions: In general I consider that...	Hybrid	Tangible	Touch	No matter
FQ1	Acquiring a TangiWheel control is easier in...	8/34.8%	14/60.9%	1/4.3%	0
FQ2	Moving a TangiWheel control to a target location is easier in...	2/8.7%	16/69.6%	2/8.7%	3/13%
FQ3	Rotating a TangiWheel control to a target angle is easier in...	1/4.3%	18/78.3%	4/17.4%	0
FQ4	Exploring items in collections is easier in...	8/34.8%	8/34.8%	5/21.7%	2/8.7%
FQ5	Selecting items in collections is easier in...	7/30.4%	5/21.7%	11/47.8%	0
FQ6	Moving an item from a collection to a target is easier in...	2/8.7%	19/82.6%	2/8.7%	0
FQ7	Deletions are easier in...	1/4.3%	6/26.1%	11/47.8%	5/21.7%
FQ8	Editing a series is easier in...	11/47.8%	10/43.5%	0	2/8.7%
FQ9	I explore items more accurately in...	7/30.4%	10/43.5%	5/21.7%	1/4.3%
FQ10	I carry out the overall selection process quicker ...	16/69.6%	4/17.4%	3/13%	0
FQ11	Editing a series is quicker in...	10/43.5%	12/52.2%	0	1/4.3%
FQ12	Repetitive manipulations on an item are more effective in...	1/4.3%	22/95.7%	0	0

Table 20. Final questionnaire summary. Participant counts and the percentage on each question are summarized.

5.6 Conclusions

In this chapter, we have reviewed the technology being used to build tabletop surface interfaces. On the Microsoft Surface Unit and SDK we have designed and implement a set of basic widgets to build surface applications. They have

been extensively used in the software developed in the experimental evaluations presented in the previous chapters. One of the most important contributions is the design of a widget for collections management called TangiWheel. Several proposals for handling collections on tabletop interfaces were reported and analyzed. In the course of this analysis, several features were extracted to allow us to describe all the proposals in a similar fashion to facilitate a systematic comparison. Tangiwheel supports not only direct finger and tangible interaction by pucks or handlers, but also a hybrid interaction scheme that allows users to combine both input methods as required. An empirical evaluation has been conducted on how the input method (Touch, Tangible or Hybrid) affects performance in the typical interactions involved in manipulating collections using TangiWheel on tabletop displays under different circumstances.

As suggested by Experiment 1, pucks are more effective than dragging and rotational finger-based input when basic acquisition and manipulation actions (such as fine-grained rotations) are performed. However, for exploration actions on collections, gestures based on one-finger rotations are more effective than rotations with pucks. This is mainly due to the lack of interruptions, allowing for a continuous interaction no matter how large the collection is. For the same reason it could be expected that a linear touch gesture for exploring large collections would be less effective than our rotational one, but additional experiments would be needed to confirm this hypothesis. However, if explorations are accompanied by highly recurrent selections and frequent movements of elements in the collections to other areas on the surface, and if the application needs an increasing number of collections to be available on the surface with a non-negligible number of actions for the selection and movement of collections, similar to the conditions in Experiment 3, then the use of pucks for selecting and moving collections and items has advantages over touch-based gestures. No single modality is therefore preferable in all cases and effectiveness varies according to the nature of the application and the actions to be carried out. A mixed input modality, such as that included in TangiWheel, offers advantages over existing tabletop-oriented widgets for collection management in situations in which the best of both methods is required. This is especially the case in applications like the required to develop this thesis, in which multiple users have to share many large collections and work simultaneously on a shared space.

At the moment, the layout manager that automatically handles and suggests the position for new widget instances on the surface is limited and naïve. In this respect, interesting further work would be to develop an improved layout manager. It could learn as the user interacts to exhibit some kind of intelligence

mainly used to adapt the positioning of the widgets according to the user task and performance. Another future work would be to explore the use of the magnitude control, contrasting it to finger based gestures when carrying out some typical tasks such as positioning, re-orienting and scaling accurately.

Chapter 6

Simulation Middleware

This chapter provides the description of the simulator and the middleware details to put the AGORAS model to work. This basically requires bridging the gap between the physics and the rule sides by means of event production and adaptation. Several parts of the simulator are tested. Firstly, the rule matching processor is verified by means of some basic synthetic performance tests to demonstrate that the matching process scales to a large number of entities and entity-types by using the relationships available in the meta-model. Secondly, a stress test is designed to evaluate that the middleware supports a reasonable high workload. Finally, the simulator specification functionality is tested by designing two sample ecosystems inspired in classic vintage games.

6.1 Introduction

In previous chapters, a model to support the creation and instantiation of ecosystems has been presented as well as its two behavioral dimensions: rules and physics. With all these middleware elements implemented, we are enabled to build the editors and basic simulators that were already discussed in the previous chapters to build specific software to conduct concrete experiments. However, some final steps are required to put a bridge between both behavioral aspects, and to be able in the end to load, simulate and interact with a stage ecosystem on the surface. Basically, a simulator must orchestrate the physics simulation, the rule triggering and execution and the user input. The key element that allows this is the event catching process from the physics simulator, so that both simulation processes, physics and rules, can be combined in a single but more complex simulation process.

This chapter presents the simulation middleware which actually orchestrates the whole process, and how the event processing pipeline works to connect the physics/user input and the rule enactment worlds. Two basic case studies regarding vintage popular games are presented to show that the simulator can effectively load some stages conforming to the AGORAS meta-model and enact them. The first case study deals with the definition of a pong-like game, and the second one is about an asteroids-like game.

6.2 Meta-Object Instantiation

Before the simulator can carry out the enactment of an ecosystem's stage, several particular concepts (meta-classes) from the model must be instantiated to produce specific elements (meta-objects). These meta-objects are assumed to exist at runtime, and the appropriate actions are indeed taken by the middleware to ensure this. They are relevant since they hold the basic bindings enabling the simulator to work properly, as they provide the core support to the know-how process. Basically, as the AGORAS middleware is about 2D entities being simulated in a stage, reacting to events and executing actions from the triggered rules, there must be some elements populating the model and having special semantics that the simulator interprets accordingly. For example, the event types to be available must be determined and instantiated by the runtime middleware, and known by users. They will be related to changes in the entity features and the production of physical events (e.g. *Collision*, *PositionChanged*, etc.). Similarly, actions to be included in rules must exist and will provide also services related to changing entity features and applying physics operations. Next, the necessary meta-object instances to put the AGORAS model to work are introduced, grouped by subsystem or category, and their meaning is outlined.

6.2.1 Type System

The most straightforward meta-model bindings are those related to the type system. The *MetaType* class is specialized into *PrimitiveType* and *EntityType* (see Figure 4 in Chapter 3). The primitive types must be mapped to the actual data-type system provided by the implementation programming language in which the AGORAS middleware has been implemented. In particular, the single primitive types supported are *Integer*, *Float*, *Bool* and *String*. Another complex primitive type is the *Vector2*, which represents a type definition with two float components. Figure 82 shows the generated unique meta-classes instances or meta-objects injected in the run-time middleware to support them. The AGORAS middleware interprets and maps them to the data types in the underlying programming language. Moreover, there are two special type meta-objects. They are the *GeneralMetaType* and the *GeneralEntityType*. The first one represents any type, either primitive or entity. It is the only existing direct instance of the class *MetaType*, and it acts as a wildcard to be used wherever a value can be required regardless of the actual type. An example of this is the special *PropertyDefinition* named *Value* that will be presented later. Similarly, the *GeneralEntityType* represents generality but this time only limited to entity-types. For in-

stance, the event-type named *Collision* needs to indicate the entities that collide. To do so, the event-type definition needs to specify slots to hold such information at runtime, but the definition of these event-attributes cannot be bound to a specific entity-type because the definition is created at design-time. Another point is the need to provide a special element to be used as a wildcard entity type in the case that rules that require specifying no particular entity as an event source. Those cases are covered by the *GeneralEntityType*.

Regarding the *EntityType* class, there is also a need to include predefined meta-objects to easily cover some cases and functionality. The first point is that stages should be also accessible as entities because, conceptually, some events are more naturally intended to be launched by them and because some rule specifications may be defined in terms of the internal state of the current stage. To cope with these situations, the run-time system automatically injects and interprets one instance of *EntityType* called “StageType” representing the stage type and an instance of *Entity* representing the stage to be executed conforming to the injected “StageType”.

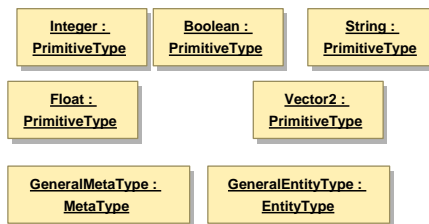


Figure 82. Type system instantiation.

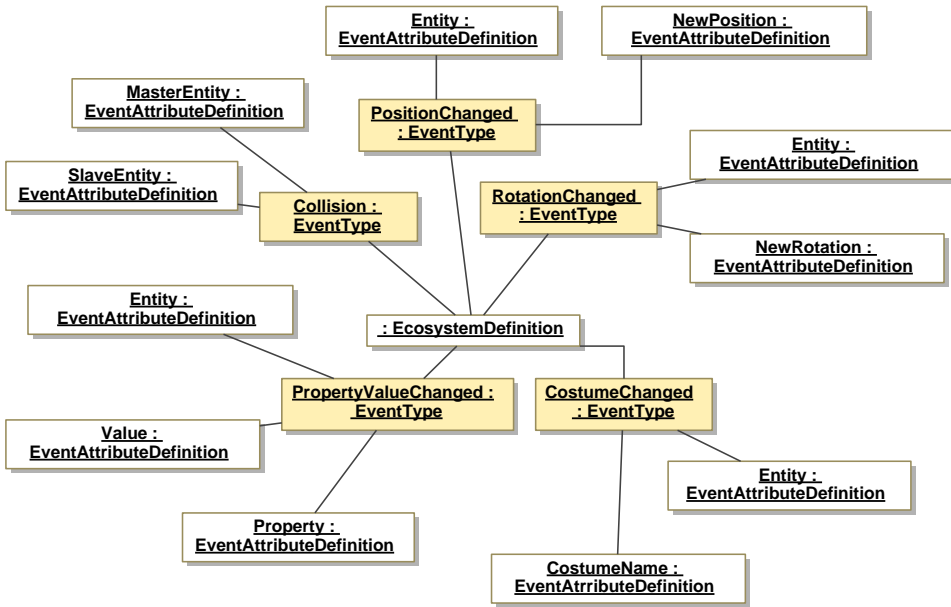


Figure 83. Predefined event types: entity changes.

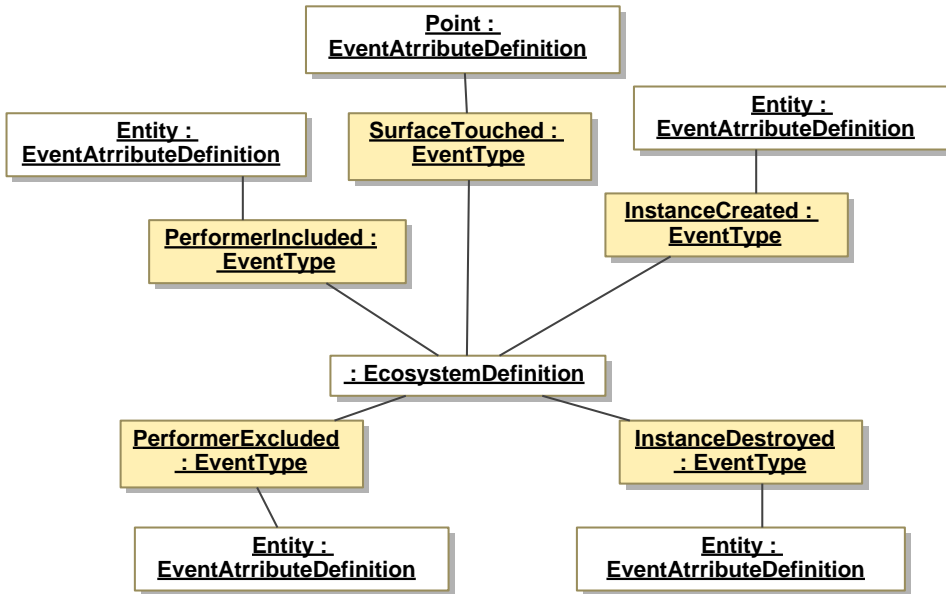


Figure 84. Predefined event types: simulation management.

6.2.2 Event System

The event occurrences that event sources throw must conform to concrete event types. As a prototype set of event types, we have considered a collection of events related to physics and operations on entities. In particular, we have considered some event types related to changes in the physical properties of the entities such as rotation, position and collisions (respectively *RotationChanged*, *PositionChanged*, and *Collision*); and also those changes in non-physical properties such as the values of the properties (*PropertyValueChanged*) and costumes being worn by entities (*CostumeChanged*). Regarding the management of the simulation, some useful event types are those notifying on the creation and destruction of entities (*InstanceCreated/Destroyed*) and the inclusion or exclusion of existing entities in the current simulation (*PerformerIncluded/Excluded*). Finally, a basic event type has been considered to notify when the user touches the surface (*SurfaceTouched*). Figure 83 and Figure 84 show the object diagrams for these meta-objects of *Event-Type* injected in the middleware.

6.2.3 Property Definitions

Although entity types are allowed and intended to have custom property definitions, there are some basic or essential property definitions that every embodied entity type must have. Figure 85 depicts the object diagram with these essential property definitions. Entities are supposed to be simulated in a 2D world. So the entity types have property definitions to hold the position, rotation, and the speed. Other property definitions could be considered in the future, linking to more properties existing in the physics engine. The property definition *Value* is simply a slot artificially providing a reference to the entity instance itself in run-time. This is useful to deal with entities as values in the rule specification since this property allows tackling entity instances as property values.

Finally, the *StageType* meta-object has the special *Gravity* property definition that the middleware will use to establish the gravity direction in the stage when simulated.

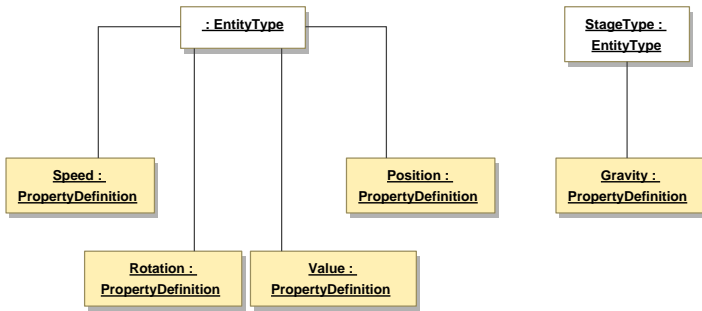


Figure 85. Essential Property Definitions.

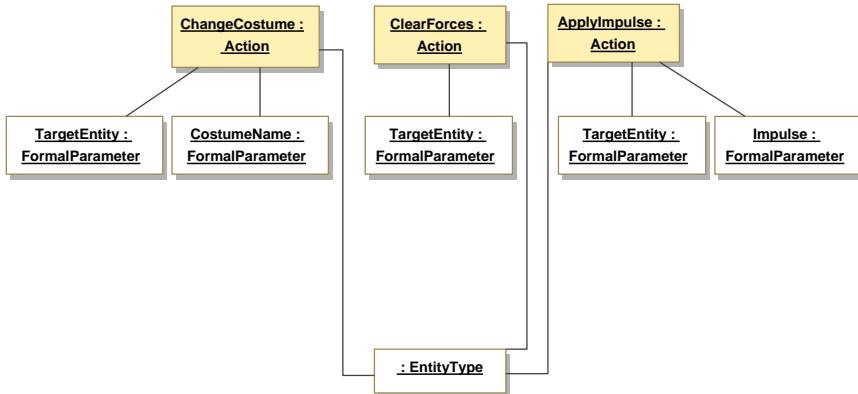


Figure 86. Action meta-objects related to regular entity-types.

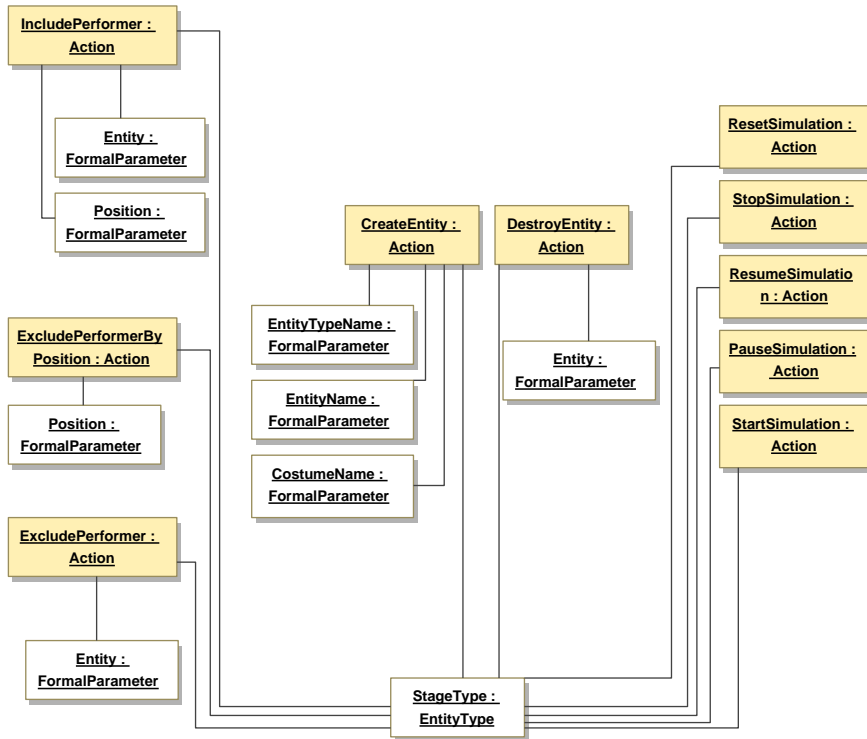


Figure 87. Action meta-objects related to the entity-types representing stages.

6.2.4 Actions Library

The model has two key concepts in the definition of actions (see Figure 4 in Chapter 3). One is the Action class that trivially represents an action to be executed by an entity. This is an important class for two reasons. Firstly, it supports the definition of different actions across different entity types. Secondly, these action definitions will be used in the rule specifications. However, there is possibly another more important concept related to actions. It is the *Action-Implementor* class and it represents the actual implementation of an action. Although the existing actions in this prototype are pre-established, this separation between specification and implementation by means of the *Action* and *Action-Implementor* classes facilitates reuse and provides a useful abstraction layer that would allow building custom editors for actions on top. These editors could be graphical or scripting languages or just a custom third-party library providing the implementation, indeed using the services supplied by the middleware implementation of the AGORAS model.

```

ExcludePerformer (Entity excludedEntity)
BEGIN
ecoInstance.performers.Remove (excludedEntity);
THROW EVENTOCCURRENCE (PerformedExcluded, excludedEntity));
END;

ChangeCostume ([Target]EmbodiedEntity eInstance, ProtoCostume costume)
PRE: ((Cast (eInstance.MetaType, EmbodiedEntityType)).ProtoCostumes[costume] is not
null)
BEGIN
eInstance.wears = costume;
THROW EVENTOCCURRENCE (CostumeChanged, eInstance, costume));
END;

ApplyImpulse ([Target]EmbodiedEntity eInstance, Vector2 direction)
BEGIN
FOREACH EntityBody b in Body eInstance.Structure.bodies
DO
b.ApplyImpulse (direction);
DONE;
END;

```

Listing 1. Sample actions in pseudo-code.

Figure 86 and Figure 87 show the object instantiation for the predefined actions currently available. This is a basic prototype set of actions that could be eventually expanded in the future. As a brief sample, some implemented actions are described in pseudo-code in Listing 1. The first one is the *ExcludePerformer* action, which removes the specified entity from the list that the ecosystem simulator keeps with the entities participating in the current simulation. The second action (*ChangeCostume*) changes the costume to be worn by the specified embodied entity. To do so, the action modifies the associated costume of the embodied entity described in the meta-model. The last action applies an impulse in some particular direction to the specified entity by applying the impulse to all the physical bodies that according to the model compose this entity. As shown in these examples, the actions are basically implemented in terms of services provided by the middleware and operations traversing the meta-model.

6.2.5 Data Processors

Finally, other meta-objects instantiated by the middleware are those for the *DataProcessors* to be used as operators in the rule's data processes. Table 21 shows the data processors available and their semantics.

Operation	Semantics
Integer / Float	
ADD(in p1: int, in p2: int, out p3:int)	$p3 \leftarrow p1 + p2$
MULT(in p1: int, in p2: int, out p3:int)	$p3 \leftarrow p1 * p2$
DIV(in p1: int, in p2: int, out p3: int)	$p3 \leftarrow p1 / p2$
CLAMP(in p1:int, in p2:int, in p3:int, out p4:int)	$p4 \leftarrow ((p1 < p2)? p2: (p1 > p3)? p3: p1)$
UTRIM (in p1:int, in p2:int, out p4:int)	$p4 \leftarrow ((p1 > p2)? p2: p1)$
LTRIM(in p1:int, in p2:int, out p4:int)	$p4 \leftarrow ((p1 < p2)? p2: p1)$
Vector2	
DemuxCompX (in p1: Vector2, out p2: float)	$p2 \leftarrow p1.X$
DemuxCompY (in p1: Vector2, out p2: float)	$p2 \leftarrow p1.Y$
ToVector(in p1: float, in p2: float, out p3 : Vector2)	$p3 \leftarrow [p1, p2]$
Boolean	
EQ (in p1: GMT , in p2: GMT, out p3: bool)	$p3 \leftarrow (p1 == p2)$
NOT (in p1: bool, out p2: bool)	$p2 \leftarrow \text{not } (p1)$
AND(in p1: bool, in p2: bool, out p3: bool)	$p3 \leftarrow (p1 \text{ AND } p2)$
OR(in p1: bool, in p2: bool, out p3: bool)	$p3 \leftarrow (p1 \text{ OR } p2)$
TRUE(out p1: bool)	$p1 \leftarrow \text{true}$
False(out p1: bool)	$p1 \leftarrow \text{false}$
LessThan (in p1: int, in p2: int, out p3: bool)	$p3 \leftarrow (p1 < p2)$
GreaterThan (in p1: int, in p2: int, out p3: bool)	$p3 \leftarrow (p1 > p2)$
LessOrEqualThan (in p1: int, in p2: int, out p3: bool)	$p3 \leftarrow (p1 \leq p2)$
GreaterOrEqualThan (in p1: int, in p2: int, out p3: bool)	$p3 \leftarrow (p1 \geq p2)$
InRange (in p1: int, in p2: int, in p3: int, out p4: bool)	$p4 \leftarrow (p1 \geq p2 \ \&\& \ p1 \leq p3)$
OutOfRange (in p1: int, in p2: int, in p3: int, out p4: bool)	$p4 \leftarrow (p1 < p2 \ \ p1 > p3)$
Misc	
Constant (param p1: GMT , out p2: GMT)	$p2 \leftarrow p1$
Biplexor (in p1: bool, in p2: GMT, in p3: GMT, out p4: GMT)	$p4 \leftarrow (p1? p2 : p3)$
RotationToValueInRange (in angle: float, in min: int, in max: int, out res: float)	$\text{res} \leftarrow \text{value in } [min, max] \text{ so that angle in the range } -2PI \ y \ 2PI$

Table 21. DataProcessors in the middleware.

6.3 Ecosystem Enactment

After presenting how the AGORAS model supports the description of ecosystems, and how several special elements (event types, primitive types, actions, etc.) are instantiated beyond the particular elements specified by the user, the simulation process can then be described.

Figure 88 shows the logical view of the AGORAS middleware. Basically, it can be divided into three separated layers that somehow resemble to the Model-View-Controller (MVC) approach. Firstly, the Ecosystem Definition layer contains all the definitions and data describing the ecosystem in full, in terms of the meta-model. This Model layer offers services to retrieve all the relevant elements and query the model as needed by the Controller layer.

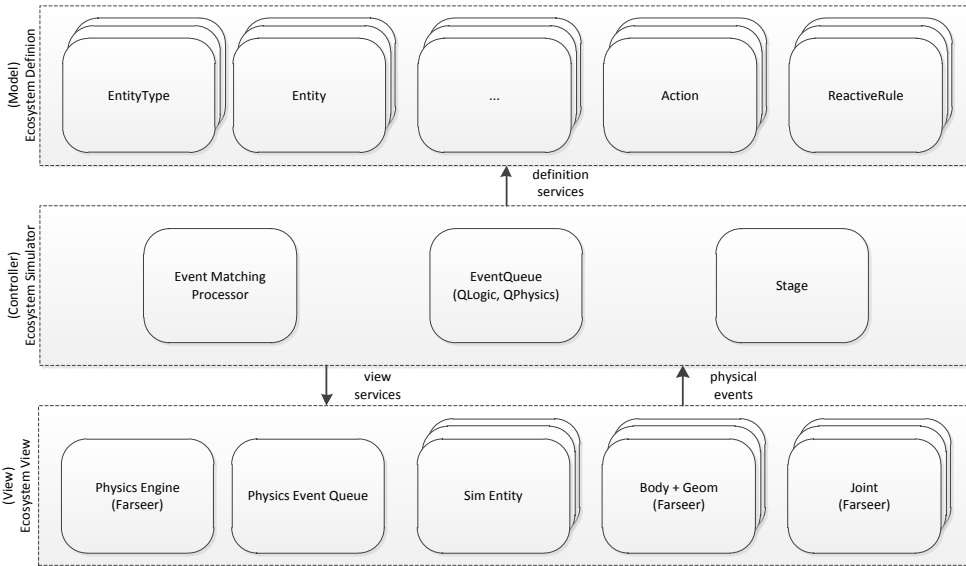


Figure 88. Architectural logical view of the AGORAS middleware.

Secondly, the Ecosystem Simulator layer is responsible for the simulation process. It holds the core functionality to orchestrate the simulation of a stage. Basically the simulator has to take a stage to be simulated from the ecosystem, and all the data from the Model layer. The simulator has an event queue for the *event occurrences* produced during the simulation. Three types of events are queued: those thrown by the actions when executed in this layer, those being consequence of the physical simulation carried out by the underlying physics engine; and those related to the gestures or interactions of the user on the surface. This queue is regularly consumed by the rule processor, which determines which rule must be triggered, and eventually performs the execution of the action of the matched rules. In this way, the simulator controls the evolution of the stage simulation. Thirdly, the Ecosystem View layer is responsible for visualizing the representation of entities under simulation. It offers a core set of view services that allow us to include entities and change their visual properties.

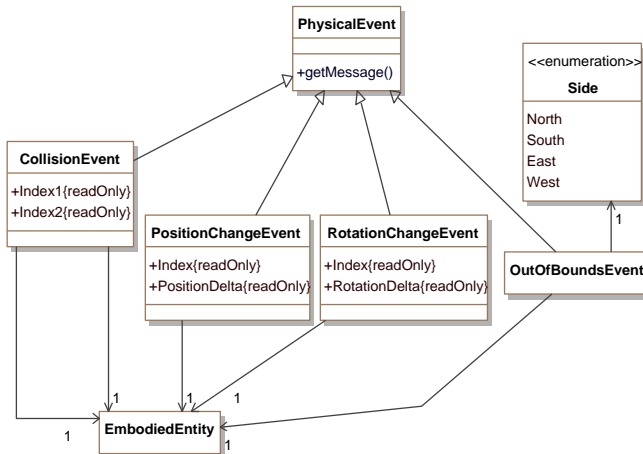
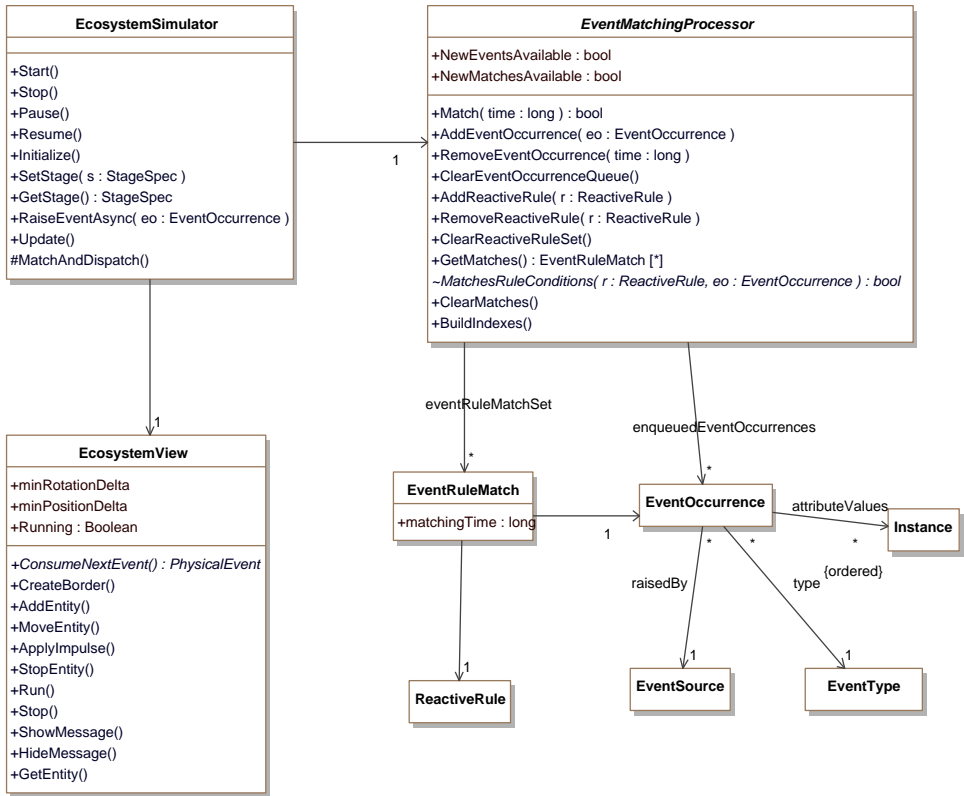


Figure 89. Class diagram for the simulator.

In Chapter 4, the simulator was simply mentioned but not detailed. Figure 89 now completes the model, by showing the relationship between the ecosystem simulator and their satellite classes that implement the processing of rules to orchestrate the stage simulation. The ecosystem simulator relies on an event matching process on top of the meta-model described (see the class *EventMatchingProcessor*) and the *EcosystemView* class. The simulator first has to deliver the information of the entities to be physically simulated to the view and starts the simulation. Then the simulator consumes the physical events from the view, and adapts them to be handled by the event matching processor. This process is responsible for determining which rules must be instantiated and triggered according to the events occurring in the environment. Given an event occurrence *ev*, the processor obtains the collection of rules whose event type matches the one in *ev*. The *EventRuleMatch* class represents such a matching. Listings 2 and 3 show the specification of the matching process in pseudo-code, assuming that the rules are properly included during the initialization and the event occurrences are injected by the middleware as they occur.

```

01: EcosystemSimulator::MatchAndDispatch ()
02: BEGIN
03:   time <- System:Time();
           // emp is the reference to the EventMatchingProcessor instance
04:   IF (emp.NewEventsAvailable) THEN
05:     BEGIN
06:       // performs matching (EventOccurrences against ReactiveRules; establishes the
           flags NewMatchesAvailable accordingly
07:       emp.Match ( time);
08:       IF (emp.NewMatchesAvailable) THEN
09:         FORAll erm: EventRuleMatch in emp.GetMatches()
10:           BEGIN
11:             ExecuteRule (erm);
12:           ENDFOR;
13:       emp.ClearEventOccurrencesQueue ();
14:
15:       // it empties the set and establishes the flag NewMatchesAvailable to FALSE
16:       emp.ClearMatches ();
17:     ENDIF;
18:   ENDIF;
19: END

```

Listing 2. Rule Execution process.

Once the *EventRuleMatch* collection is constructed each rule in the collection is executed by evaluating the precondition data process and, if this is successful, by evaluating the existing transformation data processes to execute the action defined in the consequent. The build process of the *EventRuleMatch* collection

is a key step that must be carefully designed if the performance of the overall rule execution process is not to be compromised. We will demonstrate in the next section that our engine is able to support an effective matching process under scalability conditions consisting of millions of entities in the ecosystem and thousands of entity types. To scale up to this problem size our matching process makes effective use of the meta-model relationships in which objects representing event-definitions, entity-types and entities that are part of a rule definition are conveniently linked with each other so that the computational complexity of the matching process does not increase linearly with the ecosystem size (i.e. average number of entity-types, average number of entities per entity type and number of event-types).

The algorithm illustrated in Listing 2 is invoked by the middleware to perform the execution of rules in two steps. Firstly, the matching process obtains the *EventRuleMatch* collection (line 7) and, secondly, the effective execution of the related operations is finally performed (line 11). Listing 3 details how the collection of correspondences between events and rules is calculated. The algorithm inspects the pending event occurrences (line 5) and determines which rules match for each one (lines 6-8).

```

01: EventMatchingProcessor::Match (gameTime:long)
02: BEGIN
03: NewMatchesAvailable <- FALSE;

05: FORALL eo: EventOccurrence in pendingEventOccurrences
06:  FORALL r:ReactiveRule in currentRuleSet

        // the private method MatchesRuleConditions checks for the agreement of rule condi-
        // tions in both source and event against the event occurrence
08:  IF (r.EventType = eo.EventType AND eo.RaisedBy BELONGS r.SOURCE.Population
        AND self.MatchesRuleConditions(r, eo))
09:  THEN
10:    currentMatches.Add (new EventRuleMatch (eo, r));
11:    NewMatchesAvailable <- TRUE;
12:  ENDIF
13: ENDFOR
14: ENDFOR
// clears the list of event occurrences and establishes the flag NewEventsAvailable to FALSE
16: pendingEventOccurrences.Clear ();
17: END

```

Listing 3. Rule matching process.

6.4 Meta-Model Performance Verification

The use of a meta-model, providing typed entities and reflection services, facilitates the efficient matching of events and rules, i.e., a series of relationships have been defined to effectively extract only those rules of interest for a given event occurrence. The end goal of this experimental verification was to demonstrate the advantages of using a meta-model and its associated engine based on reflection properties to allow our proposed system to scale up to highly demanding ecosystem scenarios. Two synthetic tests have been specifically designed to show that the current implementation is scalable with respect to the time required to find the set of potential rules to be enacted (i.e. matching time). Essentially, the tests have only focused on the matching time and the study on how the typology of the source population (i.e. a specific entity vs. a population given as a type) can affect the matching time. Therefore, it must be understood as a preliminary verification before further and more extended evaluation in more realistic conditions.

The first test focused on rules whose source population is specified as an entity-type (e.g. when any *ball collides any other entity*). Since rules expressed in this way refer to all the entities that belong to the population indicated by means of the type, the point to verify is that the matching time does not dramatically increase as either the number of entity-types or the number of entities within each entity-type grows. With this focus, the stress test considered a synthetic scenario in which entities injects events over the time to activate and trigger the rules, and the number of entities and the number of entity-type present in the system varied to see the evolution of the average matching time. The number of entity-types ranges between 1 and 40,000 whereas the number of entities within each entity-type is between 1 and 100. The number of rules available in the system is set to the number of entity-types present in the system at a given time. The aim was to demonstrate that the rule engine would be able to cope with a large number of entity instances in the environment while keeping the matching time constant for an event-rule pair when using rules related to populations. Figure 90 shows that the matching time remains nearly constant when the number of instances is increased to 4×10^6 .

Moreover, as can be seen in Figure 91, the matching time is not affected by the number of entity-types in the ecosystem. The meta-model reflection feature associating rules with event-types provides a retrieval process for the set of rules associated to a given event-definition at a constant time, as confirmed by the experimental results.

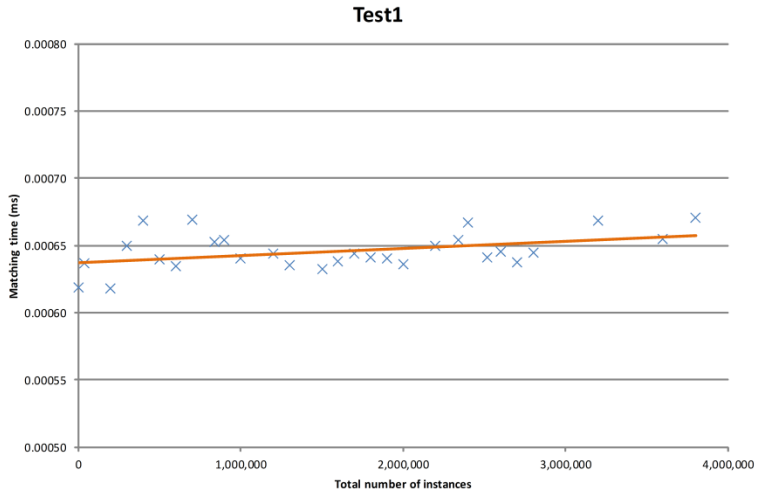


Figure 90. Matching time vs. total number of entity instances under the conditions of Test 1.

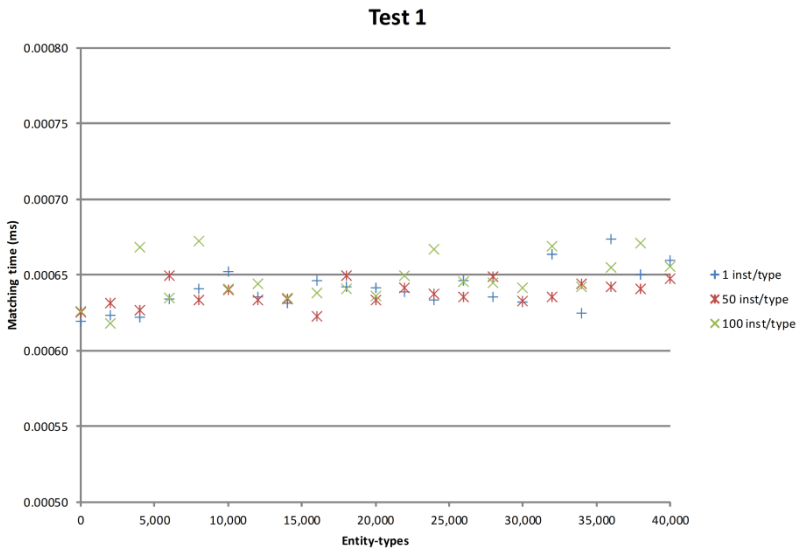


Figure 91. Matching time vs. the number of entity-types in the conditions of Test 1.

Given that the current implementation is suitable to cope with event sources expressed as population by means of types no matter the size of the entities present, a second essential test would consist of exploring the matching time when using rules whose event source is defined over specific entities. In this particular case, every rule only refers to an event thrown by a concrete entity. Thus, in this second stress test the matching time has to be analyzed with respect to the variation of the number of rules defined for each pair consisting of an event-type and entity.

For each pair $\langle E, S \rangle$, i.e. each pair event-type and source entity, a number of rules R were created ($R=1, 5, 10$). In this way, when an entity throws a given event occurrence, a set of rules will be triggered, since there are several rules that match the same pair E and S . As the total amount of rules depends on the number of event-types present in the system, it was varied between 1 and 2000, in order to explore how the matching time evolves as the set of matching rules increases. In addition, to show that if the relationships present in the meta-model are exploited then a set of efficient indexes can be built so that the matching time remains constant no matter the number of event types considered, two versions of the event-matching processor have been compared (optimized and non-optimized).

The first version was implemented without taking into account the valuable information contained in the meta-model of the rule specification, which needs to examine the whole rule set regardless of the current event occurrence or the entity that raised it. The second version, however, considered the construction of indexes on the rule set based on the relationships available at the meta-model level, so that the processor was able to effectively determine the rule set of interest for a given event occurrence.

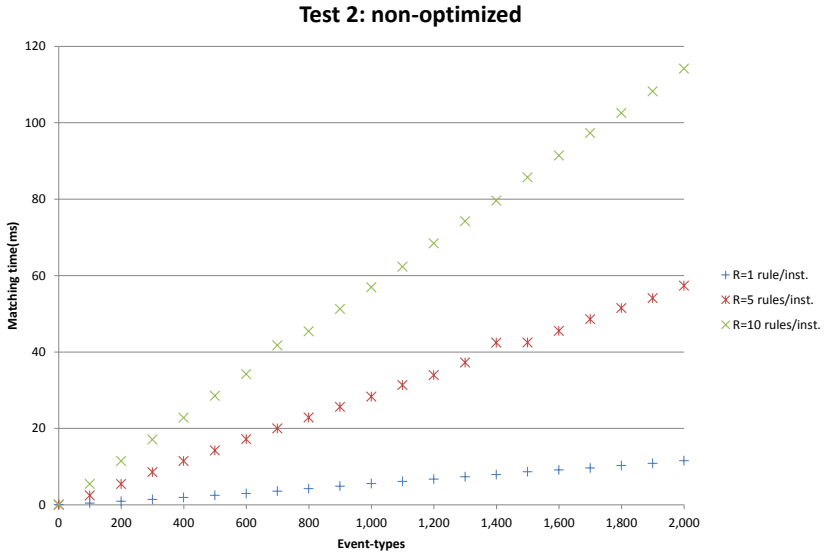


Figure 92. Matching time vs. event-types with repetitions under non-optimized conditions in Test 2.

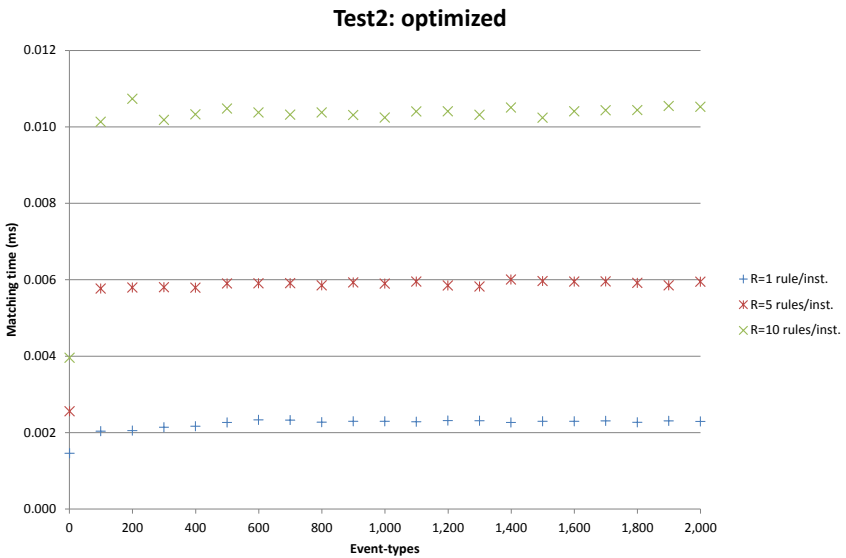


Figure 93. Matching time vs. event-types and rule repetitions under optimized conditions in Test 2.

Figure 92 and Figure 93 plot the matching time in the above-described conditions for the non-optimized and optimized versions, respectively. As Figure 92 shows, the matching time is linear up to the number of event-types ($|\text{rules}| = R * |E| * 10$), as the set of rules must be inspected in the non-optimized version. The optimized version obtained nearly constant matching times, as the optimizations are able to provide the appropriate rule set without any extra cost (see Figure 93).

The previous experiments show that the matching process scales with the number of entities, entity-types and rules per event-type. This makes the scalability of the system in terms of the obtained throughput (i.e. number of events processed per unit of time) a matter of the number of available parallel processing nodes (threads, cores or devices) that are used so that the matched rules may be executed in parallel.

6.5 Physics Simulation Stress Test

Chapter 3 presented an implementation of a simplified structure editor supporting a simulation mode. It worked smoothly in the user studies conducted. However, it would be interesting to check that the ecosystem view performs well enough for highly demanding stages. To verify this performance requirement we designed a stress test consisting of a stage throwing entity balls at every 350 milliseconds in several pre-established directions while notifying the physical events occurring in the simulation. The idea is that the simulation is populated with a large number balls moving in a few seconds, and then verify that the performance does not drop dramatically. On average, the test required to process a total of 724 event occurrences per second, which is large enough to support highly dynamic ecosystems. Figure 94 shows the visualization of the test running on the surface.

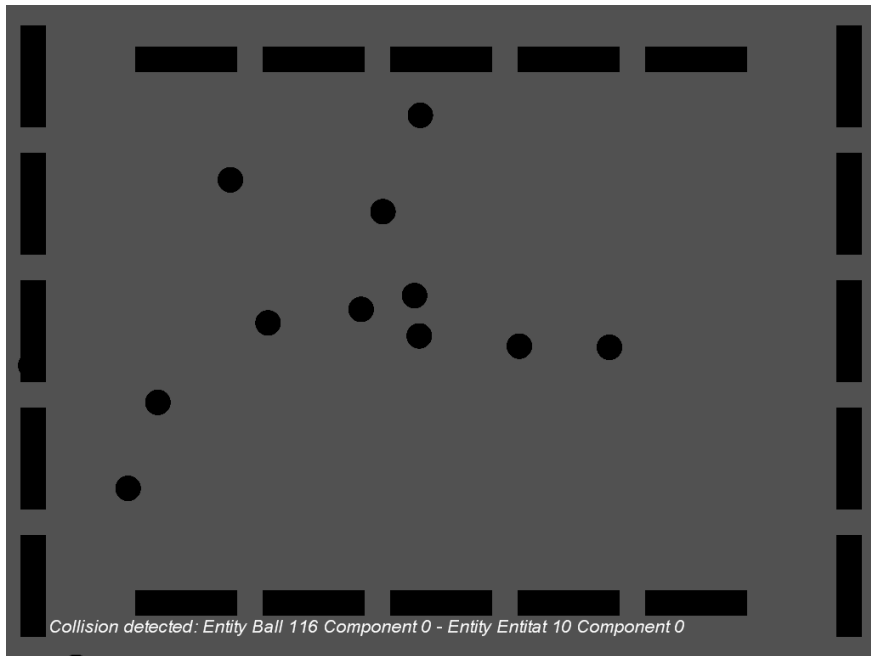


Figure 94. Screenshot of the simulation stress test.

6.6 Case Studies

In order to validate that the whole meta-model is properly supporting the creation and the simulation of ecosystems, in this section two different ecosystems are presented inspired by classic vintage games.

6.6.1 A Pong-like Ecosystem

The first basic ecosystem is a game inspired in Pong⁷. It was a video game resembling a tennis match in the computers and the first generation of home video-consoles in the 1960s. As Figure 95 shows, the most essential version of this game consists of two paddles located in two opposite sides controlled by the users, and a ball that bounces from one side to another. Appendix C contains the file specification for the description of the AGORAS ecosystem for this game. The file format is mostly a human-readable structured textual language, being simply a serialization of the instances of the AGORAS meta-

⁷ <http://en.wikipedia.org/wiki/Pong>

model. In this section, the specification is going to be dissected, emphasizing on how the most important parts of the stage have been modeled.

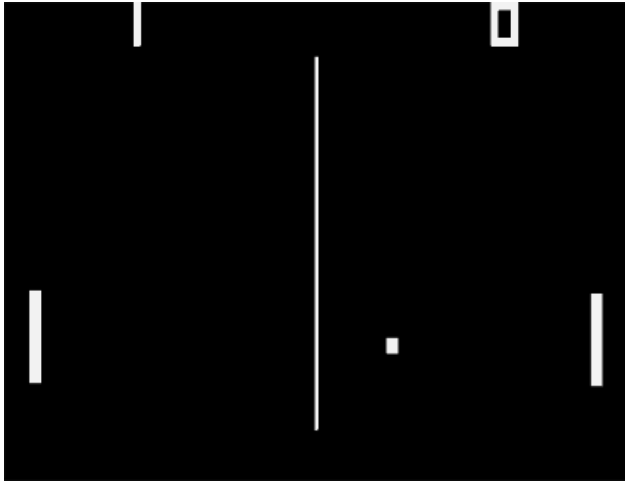


Figure 95. Pong in action.

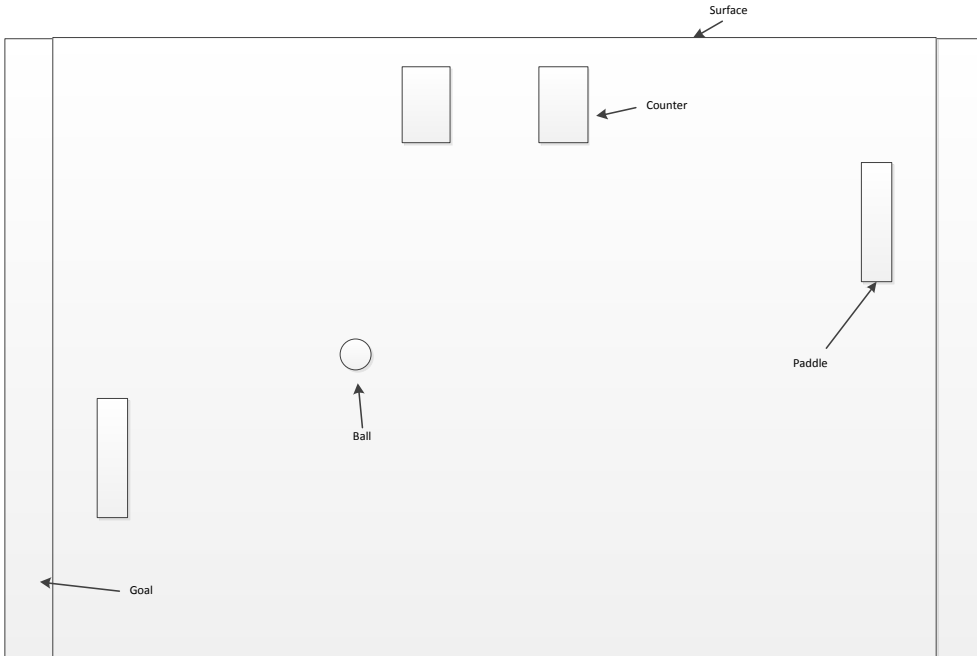


Figure 96. Sketch of Pong in terms of AGORAS concepts.

Figure 96 depicts a sketch of the stage for the pong game. The most evident entities are the paddles and the ball. Other elements that have been modeled by means of entities are the displays, which will visualize the score counts of the player. They all are embodied entity instances, which require the creation of entity-types as well as their structure-types with their corresponding proto-costumes and proto-skins. Finally, additional goal entities have been considered to facilitate the detection of scoring a goal, located in the boundary and with an invisible look. The barriers, on the top and bottom of the stage have not been modeled as entities since the stages have already a meta-attribute that indicates whether the borders behave as physical barriers, so that any entity reaching them would be bounced back to the stage.

First of all, the specification contains a delimited section for the definition of structure-types. Listing 4 shows the lines defining structure-types. Notice that these definitions refer to external specification files (.st) expressed by means of relative paths. These separate files contain the specification of structure-types in terms of structural-components and joints. The details are not discussed here, although they can be found in the Appendix C for reference. However, in this sample case study, its physical structures use basic shapes, i.e., single boxes for the paddles and goals, a circle for the ball, etc.

```

STRUCTURE_TYPES
STRUCTURE_TYPE NAME BallStructure
    PATH "Content/Simulator/BallStructure.st"
END_STRUCTURE_TYPE
STRUCTURE_TYPE NAME PaddleStructure
    PATH "Content/Simulator/PaddleStructure.st"
END_STRUCTURE_TYPE
STRUCTURE_TYPE NAME DisplayStructure
    PATH "Content/Simulator/DisplayStructure.st"
END_STRUCTURE_TYPE
STRUCTURE_TYPE NAME GoalStructure
    PATH "Content/Simulator/GoalStructure.st"
END_STRUCTURE_TYPE
END_STRUCTURE_TYPES

```

Listing 4. Structure-type definition for Pong

Similarly, there is a delimited section for the specification of entity-types (see Listing 5). To reduce the complexity of the specification file, the proto-costume definitions are also indicated as external referenced files. They will contain the specification of proto-costumes in terms of proto-skins. The ball and paddles are expected to exhibit physical behavior, besides having a visual look. To this purpose the attribute *PHYSICAL_RESPONSE* of the corre-

sponding entity-types is set to *true*. Moreover, as the paddle can be interacted by the user, the attributes *IS_FINGER_SENSITIVE* and *IS_TAG_SENSITIVE* are set to true, only for this concrete entity-type. However, there are some special considerations to be aware of in the specification of the entity-type for displays. Firstly, its physical response attribute is set to false, since the displays are only intended for visualization purposes. Secondly, this entity-type specifies a user-defined integer property, which will be used to count the score of the player. Thirdly, at the present moment, there is no mechanism to specify or generate custom proto-costumes from numbers or strings. Thus, it is necessary to provide a quite enough large amount of proto-costumes to represent the range of scores. In this way, this entity-type has a list of proto-costumes so that each one contains an image for a number representation.

ENTITY_TYPES

```

ENTITY_TYPE BallType
  PHYSICAL_RESPONSE true
  REBOUND true
  STRUCTURE_TYPE BallStructure
  PROTO_COSTUMES
    PROTO_COSTUME BallCostume
      PATH "Content/Simulator/BallCostume.pc"
    END_PROTO_COSTUME
  END_PROTO_COSTUMES
END_ENTITY_TYPE
ENTITY_TYPE PaddleType
  PHYSICAL_RESPONSE true
  IS_FINGER_SENSITIVE true
  IS_TAG_SENSITIVE true
  STRUCTURE_TYPE PaddleStructure
  PROTO_COSTUMES
    PROTO_COSTUME PaddleCostume
      PATH "Content/Simulator/PaddleCostume.pc"
    END_PROTO_COSTUME
  END_PROTO_COSTUMES
END_ENTITY_TYPE
...
ENTITY_TYPE DisplayType
  PHYSICAL_RESPONSE false
  STRUCTURE_TYPE DisplayStructure
  PROTO_COSTUMES
    PROTO_COSTUME DisplayCostume0
      PATH "Content/Simulator/DisplayCostume0.pc"
    END_PROTO_COSTUME
    PROTO_COSTUME DisplayCostume1

```

```

    PATH "Content/Simulator/DisplayCostume1.pc"
  END_PROTO_COSTUME
  PROTO_COSTUME DisplayCostume2
    PATH "Content/Simulator/DisplayCostume2.pc"
  END_PROTO_COSTUME
  PROTO_COSTUME DisplayCostume3
    PATH "Content/Simulator/DisplayCostume3.pc"
  END_PROTO_COSTUME
  PROTO_COSTUME DisplayCostume4
    PATH "Content/Simulator/DisplayCostume4.pc"
  END_PROTO_COSTUME
...
END_PROTO_COSTUMES
PROPERTIES
  PROPERTY NAME Count TYPE Integer END_PROPERTY
END_PROPERTIES
END_ENTITY_TYPE
...
END_ENTITY_TYPES

```

Listing 5. Entity-types for Pong

The delimited section for the definition of entities simply contains the relation of entity instances put in terms of the entity-types specified before (see Listing 5). They will be available to be used in the stages of the ecosystem.

The remaining part of the specification is related to the definition of stages, which are the pieces that will eventually be simulated. The definition of our stage TablePong (see Listing 6) requires enabling the top and bottom borders, to make the ball bounce back when they are hit.

```

ENTITIES
  ENTITY
    NAME Ball TYPE BallType COSTUME BallCostume
  END_ENTITY
  ENTITY
    NAME Paddle1 TYPE PaddleType COSTUME PaddleCostume
  END_ENTITY
  ENTITY
    NAME Paddle2 TYPE PaddleType COSTUME PaddleCostume
  END_ENTITY
  ENTITY
    NAME Display1 TYPE DisplayType COSTUME DisplayCostume0
  END_ENTITY
  ENTITY
    NAME Display2 TYPE DisplayType COSTUME DisplayCostume0

```

```

END_ENTITY
ENTITY
  NAME Goal1 TYPE GoalType COSTUME Faded
END_ENTITY
ENTITY
  NAME Goal2 TYPE GoalType COSTUME Faded
END_ENTITY
END_ENTITIES

```

Listing 6. Entities for Pong

The stage definition also includes the effective participation of entities on it, indicating the initialization values for their properties. Finally, the pong game is much more than a ball bouncing against the borders and the paddles according to the physical simulation. It requires counting the goal scores. To this purpose, the stage specification includes the definition of the reactive rules governing the logic simulation. In the specification file in the Appendix C the reactive rules are expressed in terms of dataflow expressions, as they are intended to be edited with the rule editing tool presented in the Chapter 4, and therefore it includes some specific layout meta-data. Nevertheless, the rules are introduced and explained here using a high-level textual rule format to avoid unnecessary complexity in the discourse.

```

STAGES
STAGE
  NAME TablePong
  TOP_BORDER true
  BOTTOM_BORDER true
  LEFT_BORDER false
  RIGHT_BORDER false

  PARTICIPATIONS
  ENTITY BallType::Ball
    PROPERTY NAME Position TYPE Vector2 VALUE 0 0 END_PROPERTY
    PROPERTY NAME Speed TYPE Vector2 VALUE -500 500 END_PROPERTY
  END_ENTITY
  ENTITY PaddleType::Paddle1
    PROPERTY NAME Position TYPE Vector2 VALUE -480 0 END_PROPERTY
  END_ENTITY
  ENTITY PaddleType::Paddle2
    PROPERTY NAME Position TYPE Vector2 VALUE 480 0 END_PROPERTY
  END_ENTITY
  ENTITY DisplayType::Display1
    PROPERTY NAME Position TYPE Vector2 VALUE -50 350 END_PROPERTY
    PROPERTY NAME Count TYPE Integer VALUE 0 END_PROPERTY
  END_ENTITY

```

```

ENTITY DisplayType::Display2
  PROPERTY NAME Position TYPE Vector2 VALUE 50 350 END_PROPERTY
  PROPERTY NAME Count TYPE Integer VALUE 0 END_PROPERTY
END_ENTITY
ENTITY GoalType::Goal1
  PROPERTY NAME Position TYPE Vector2 VALUE -540 0 END_PROPERTY
END_ENTITY
ENTITY GoalType::Goal2
  PROPERTY NAME Position TYPE Vector2 VALUE 540 0 END_PROPERTY
END_ENTITY
END_PARTICIPATIONS

REACTIVE_RULES
...
END_REACTIVE_RULES
END_STAGE
END_STAGES

CURRENT_STAGE TablePong
END_ECOSYSTEM

```

Listing 7. TablePong Stage specification

For this essential stage, a basic rule set has been considered. The first rule, `GameOver`, defines when the game should stop (see Listing 8). The rule will stop the simulation whenever any player scores ten times.

To control the increment of the score counters, two rules have been defined (see Listing 7). These rules are really similar to each other, as each one is responsible of increasing the score by one unit of a concrete display in the stage. Basically, these rules detect when the ball hits a goal and, consequently, the count property of the corresponding display is incremented by 1.

```

RULE GameOver
PRIORITY: 10
IF S: ANY DisplayType THROWS an EVENT E: PropertyValueChanged
AND S.Count = 10
THEN WITH T: TablePong
PERFORM O: T.PauseSimulation

```

Listing 8. GameOver rule specification

```

RULE Score1Increment
PRIORITY: 1
IF S: ANY BallType THROWS an EVENT E: Collision
AND E.SlaveEntity = Goal2
THEN WITH T: Display1

```

```
PERFORM O: T.Count ← T.Count + 1
```

```
RULE Score2Increment
```

```
PRIORITY: 1
```

```
IF S: ANY BallType THROWS an EVENT E: Collision
```

```
AND E.SlaveEntity = Goal1
```

```
THEN WITH T: Display2
```

```
PERFORM O: T.Count ← T.Count + 1
```

Listing 9. Rules to increment the score

The rules listed above are able to change the counter when a goal is scored. However, some additional actions would have to be carried out when this happens. For example, the costume of the display should change to visualize the corresponding number, and the ball should be repositioned at the center to start a new point. The current rule model of AGORAS does not support composed actions yet, that is, several actions cannot be specified in the consequent of a single rule. Thus, when required, either several rules are defined with exactly the same precondition, or a set of rules must be created composing a chain of rules, which will be triggered in sequence. The next two rules follow the pattern of chaining rules. In this way, every time a display's score is increased, the associated costume being "worn" by this display entity will change accordingly (see Listing 9). Finally, when any property of a display is changed, the ball is repositioned at the center (see Listing 11). Indeed, this rule is assuming that there is no change in a display's property that does not entail that a goal is scored. Figure 97 shows the pong being simulated in AGORAS.

```
RULE ChangeScoreCostume
```

```
PRIORITY: 0
```

```
IF S: ANY DisplayType THROWS an EVENT E: PropertyValueChanged
```

```
AND E.Property = Count
```

```
THEN WITH T: ANY DisplayType
```

```
SO THAT F: E = T
```

```
PERFORM O: T.ChangeCostume("DisplayCostume" CONCATENATED T.Count)
```

Listing 10. Rule to change the score counters' costumes

```
RULE ResetBallPosition
```

```
PRIORITY: 15
```

```
IF S: ANY DisplayType THROWS an EVENT E: PropertyValueChanged
```

```
THEN WITH T: Ball
```

```
PERFORM O: T.Position = (0, 0)
```

Listing 11. Rule to start a new point

Finally the movement of the paddle in terms of the input provided by the user is controlled by means of the MovePaddle rule. It detects when a finger is moving on a paddle entity. When this happens the paddle impacted by the finger is moved in the Y axis accordingly (see Listing 12).

```

RULE MovePaddle
PRIORITY: 1
IF S: ANY PaddleType THROWS an EVENT E: FingerContactChanged
THEN WITH T: ANY PaddleType
SO THAT F: S = T
PERFORM O: T.Move(0, E.ContactPosition.Y - E.PreviousContactPosition.Y)

```

Listing 12. Rule to control the paddle movement.

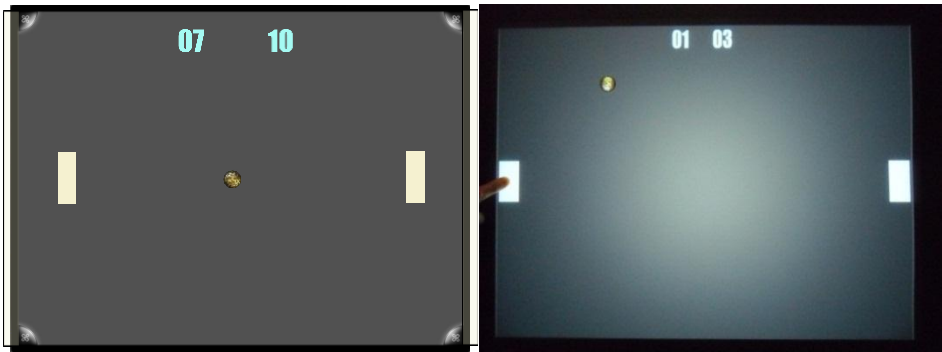


Figure 97. Screenshot and picture of the pong game simulated in AGORAS.

6.6.2 An Asteroids Ecosystem

The second game implemented with AGORAS is a simplified version of the Asteroids videogame⁸. Basically, the game consists of a spaceship and asteroids of different sizes that can be fragmented until destruction into smaller pieces (see Figure 98).

⁸ http://en.wikipedia.org/wiki/Asteroids_%28video_game%29

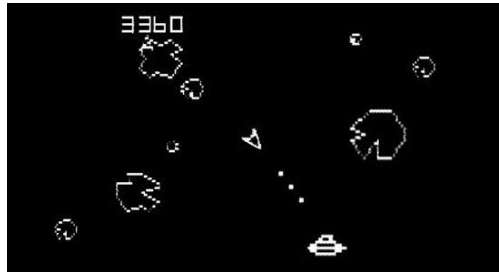


Figure 98. Asteroids game screenshot.

In our simplified version we have modeled the components in terms of structures, entity-types and entities as Figure 99 shows. The basic stage considered in the ecosystem sets the spaceship at the center of the screen, which may be rotated by means of buttons, and includes a collection of asteroids moving towards the spaceship. Figure 100 shows a sketch of the modeled AGORAS stage.

There is a single instance for the `SpaceshipType` which is named `PlayerSpaceship` and it is fixed (i.e. it is static) to the center of the screen and may only be moved by means of specific controls like buttons.

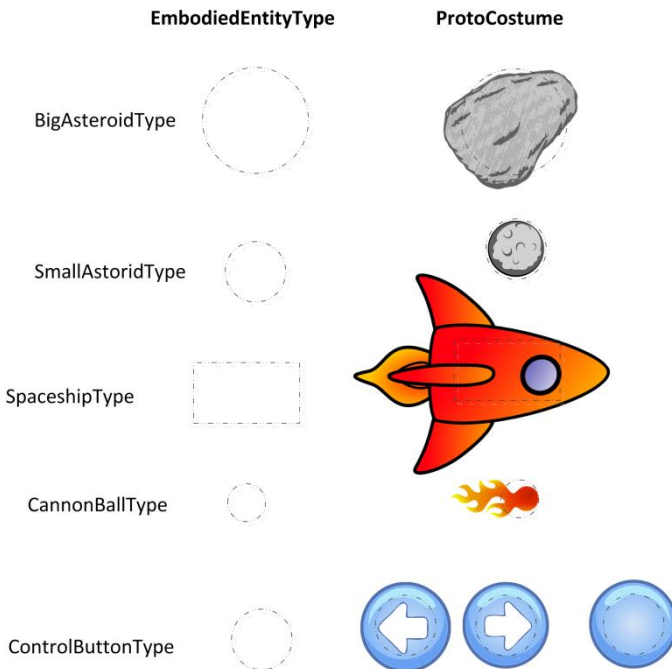


Figure 99. Modeling Asteroids in AGORAS.

In particular, there are three blue buttons that are instances of the `ControlButton`, namely `RightControl`, `LeftControl` and `ShootControl`. They do not interact with the physical instances (i.e. *PhysicalResponse* to *false*) but they must accept finger interaction (i.e. *IsfingerSensitive* to *true*). The Left and Right controls allow the rotation of the spaceship some degrees in the corresponding direction when they are touched. The rules controlling the rotation of the spaceship are as specified in Listing 13. They encode the variation of the rotation angle of the spaceship whenever the corresponding control is touched.

```

RULE RotateLeft
PRIORITY: 2
IF S: LeftControl THROWS an EVENT E: SurfaceTouched
THEN WITH T: PlayerSpaceship
PERFORM O: T.Rotation = T.Rotation + 30°

```

```

RULE RotateRight
PRIORITY: 2
IF S: RightControl THROWS an EVENT E: SurfaceTouched
THEN WITH T: PlayerSpaceship
PERFORM O: T.Rotation = T.Rotation - 30°

```

Listing 13. Rules to rotate the spaceship

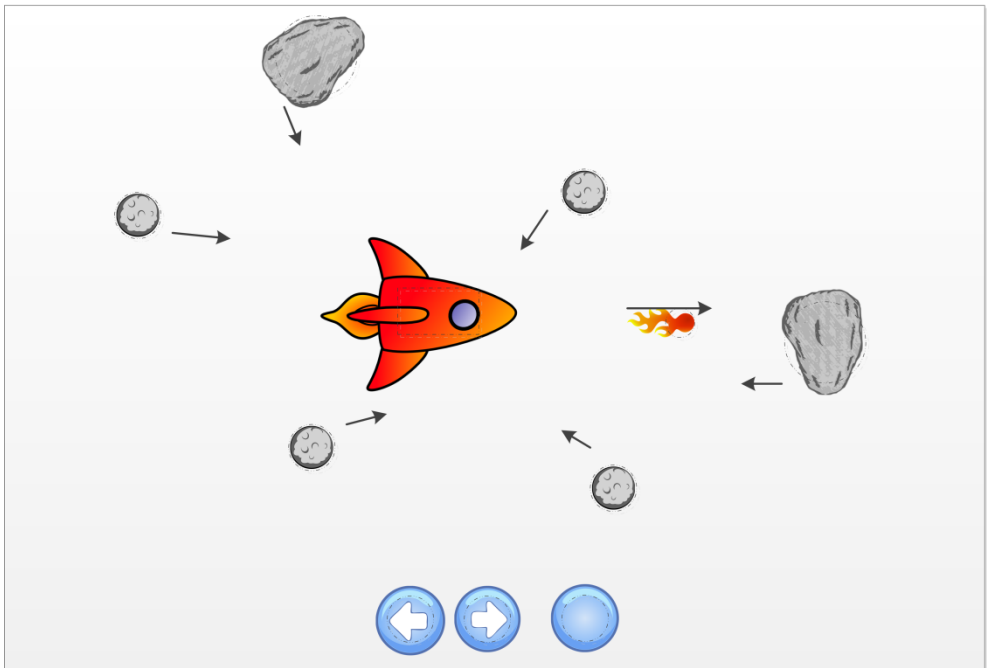


Figure 100. Sketch of the modeled Asteroids.

The Shoot control creates and launches a CannonBall instance from the spaceship in the corresponding direction (see Listing 14). However there is an important and non-trivial calculation to set both the position and speed of the cannon ball so that it must be consistently in alignment with the rotation and position of the spaceship's nose. These are the more complex calculations in the parameters of the operation.

```

RULE Shoot
PRIORITY: 2
IF S: ShootControl THROWS an EVENT E: SurfaceTouched
THEN WITH T: AsteroidGameStage
PERFORM O: T.CreateAndIncludeEntity(
    EntityType= CannonBallType,
    EntityBaseName = "CannonBallNew",
    CostumeName = "CannonBallCostume",
    Position = RotateAboutOrigin (Vector2 (0, 60), Vector2 (0, 0),
    PlayerSpaceship.Rotation)
    Speed = RotateAboutOrigin (Vector2 (0, 30), Vector2 (0, 0), PlayerSpaceship.Rotation)
)

```

Listing 14. Shooting rule.

There are two types of asteroids with different size: the big and the small asteroids. The “big” ones are expected to be fragmented into two “small” ones when hit by a cannonball. The “small” asteroids simply disappear in such a case. Therefore, when a cannonball impacts on an asteroid either small or big, the operation in the consequent must remove from the simulation the entities involved in the impact. Listing 15 shows the rules that specify the exclusion of entities from the current simulation. As there are two types of asteroids, there are two sets of rules contextualized to each asteroid type. As the rule language does not allow several operations in the consequent by design, it is necessary to write several rules with exactly the same condition.

```

RULE ExcludeSmallAsteroid
PRIORITY: 5
IF S: ANY SmallAsteroidType THROWS an EVENT E: Collision
AND IntanceOf( E.SlaveEntity, CannonBallType)
THEN WITH T: AsteroidGameStage
PERFORM O: T.ExcludePerformer(Entity = S)

```

```

RULE ExcludeCannonBallForSmallAsteroidCollision
PRIORITY: 5
IF S: ANY SmallAsteroidType THROWS an EVENT E: Collision
AND IntanceOf( E.SlaveEntity, CannonBallType)
THEN WITH T: AsteroidGameStage
PERFORM O: T.ExcludePerformer(Entity = E.SlaveEntity)

```

```

RULE ExcludeBigAsteroid
PRIORITY: 5
IF S: ANY BigAsteroidType THROWS an EVENT E: Collision
AND InstanceOf( E.SlaveEntity, CannonBallType)
THEN WITH T: AsteroidGameStage
PERFORM O: T.ExcludePerformer(Entity = S)

```

```

RULE ExcludeCannonBallForBigAsteroidCollision
PRIORITY: 5
IF S: ANY BigAsteroidType THROWS an EVENT E: Collision
AND InstanceOf( E.SlaveEntity, CannonBallType)
THEN WITH T: AsteroidGameStage
PERFORM O: T.ExcludePerformer(Entity = E.SlaveEntity)

```

Listing 15. Rules excluding the entities involved in collisions with cannon ball.

One challenging behavior is the fragmentation of a big asteroid when impacted by a cannon ball. Two rules allow the virtual fragmentation of a big asteroid into two smaller ones (see Listing 16). Both rules have again the same antecedent but the consequent specifications are slightly different to produce each a small asteroid with different position and velocity vectors.

```

RULE FragmentBigAsteroid_Part1
PRIORITY: 4
IF S: ANY BigAsteroidType THROWS an EVENT E: Collision
AND InstanceOf( E.SlaveEntity, CannonBallType)
THEN WITH T: AsteroidGameStage
PERFORM O: T.CreateAndIncludeEntity(
    EntityType=SmallAsteroidType,
    EntityBaseName = "LeftSmallAsteroid",
    CostumeName = "SmallAsteroidCostume",
    Position =S.Position + Vector2 (-35, 0),
    Speed = RotateVectorAboutOrigin (S.Speed, S.Position, -45°)
)

```

```

RULE FragmentBigAsteroid_Part2
PRIORITY: 4
IF S: ANY BigAsteroidType THROWS an EVENT E: Collision
AND InstanceOf( E.SlaveEntity, CannonBallType)
THEN WITH T: AsteroidGameStage
PERFORM O: T.CreateAndIncludeEntity(
    EntityType=SmallAsteroidType,
    EntityBaseName = "LeftSmallAsteroid",
    CostumeName = "SmallAsteroidCostume",
    Position =S.Position + Vector2 (35, 0),
    Speed = RotateVectorAboutOrigin (S.Speed, S.Position, 45°)
)

```

)

Listing 16. Rules for the fragmentation of a big asteroid into two smaller ones.

Finally some basic rules to finish the game and destroy the entity instances that are not available any more are required (see Listing 17). The game over condition consists simply of either a big or small asteroid impacting on the spaceship.

RULE GameOverBigAsteroid

PRIORITY: 10

IF S: ANY BigAsteroidType THROWS an EVENT E: Collision

AND E.SlaveEntity = PlayerSpaceship

THEN WITH T: AsteroidGameStage

PERFORM O: T.PauseSimulation

RULE GameOverSmallAsteroid

PRIORITY: 10

IF S: ANY SmallAsteroidType THROWS an EVENT E: Collision

AND E.SlaveEntity = PlayerSpaceship

THEN WITH T: AsteroidGameStage

PERFORM O: T.PauseSimulation

RULE RemoveExcludedEntity

PRIORITY: 2

IF S: AsteroidGameStage THROWS an EVENT E: PerformerExcluded

THEN WITH T: AsteroidGameStage

PERFORM O: T.DestroyEntity (Entity = E.Entity)

Listing 17. Game over and entity removal rules.

Figure 101 shows a screenshot of the Asteroids game modeled in AGORAS under simulation. The full specification can be found in the Appendix D. Many other rules could be added to improve the game (e.g. additional asteroids are included in the game when some others go beyond the visible area of the surface, etc.).



Figure 101. Screenshot of the Asteroids running in AGORAS.

6.7 Conclusion

This chapter introduced the details of the simulation middleware to put the meta-model to work. Along with the description of the three layers in which the middleware has been organized, several tests have been applied to check that it is functional and it suits to highly demanding ecosystem's stages in terms of performance. The tests under consideration were extreme in the sense that they tested much larger problem sizes than the types of ecosystems that will be eventually created by non-expert young programmers. Nevertheless, the matching process tests showed that the implemented middleware scales with the number of entities, entity-types and rules per event-type, and the synthetic test of visualization and event processing showed that a large enough number of events can be successfully processed without performance degradation. Finally, the case studies validated that the middleware works and that meaningful rules can be defined to simulate classic games with the meta-model of AGORAS.

This chapter summarizes and analyzes the contributions of this dissertation, and reports the results derived from it. Future work is also described and discussed as the continuous improvement is the basis of any research activity.

7.1 Summary

The thesis departed from the idea that *creativity* is relevant in the future development of individuals, especially in the case of children and teenagers, and as a consequence it is also relevant for the society. It is therefore worthy to do research investigating the phenomenon with relation to information system technology. The review on how creativity and other related learning aspects are approached by ICT-based proposals has revealed that two main categories of works for children and teenagers are available. The first category concerns about approaches in which users cannot create the characters or entities but these already exist and users are allowed to perform with them or specify their behavior by encoding a program to tell a story. The second comprises those proposals that have considered the creation of characters/entities as central tasks, what makes this group especially interesting. The proposals of this group use some sort of computational model to allow users to build the simulation or the story once characters are created. Although some of the related work has already considered some kind of natural interface rather than drag&drop techniques in a WIMP approach, an important amount of work still relies on them, especially in this second group. In addition, approaches are often said to be designed according to learning theories. Indeed, this is a good way to build new artifacts or environments as design principles can be extracted from them. However, there is a lack in studies reporting on the aspects related to the theory in order to validate that those principles are producing the expected effect.

In this thesis, two main aspects have led us to consider a tabletop game-based approach. Firstly, the vision of taking into consideration the creation of games as a more rewarding and engaging learning process as proposed by Abt in the pre-digital age. Secondly, the idea of following more natural interaction platforms, facilitating the discussion and collaboration among participants, and therefore not being the technology a significant barrier but a medium to enhance the overall experience.

A necessary step for the construction of the required software is either taking or developing a user interface toolkit for tabletops. As the construction of digital artifacts requires the use of collections as key *building blocks*, a strong interface requirement is to support massively collection exploring. A review of widgets and techniques to explore collections on surfaces has been conducted, describing and comparing them along several aspects and features. These features are broadly related to main aspects concerning the generality of the approach, data organization features, input methods and multiuser support. From this comparison we have extracted valuable design information, and decided to expand and develop the basic abstractions provided by the Microsoft Surface SDK for the construction of a UI toolkit. In this line, one of the contributions of this thesis is therefore the design, development and evaluation of a widget for handling collections called TangiWheel. An important feature of the proposed widget is the support of Touch and Tangible input schemes separately with high levels of resemblance. This has made it possible to consider a Hybrid design in which users are able to decide the input that best suits to their preferences and tasks' condition demands. The empirical evaluation has focused on the analysis of the effect of input method, i.e. fingers or handlers (pucks), on performance in manipulating collections in terms of performance time and number of manipulations required in a set of tasks involving basic manipulations and explorations under several circumstances. The first experiment has shown that the pucks are more effective in performance than the rotational finger-based technique in basic acquisition and manipulation of the widget, especially when fine-grained accuracy is required. However, the second experiment has shown that the Touch techniques as more effective in the task of selecting target items since the pattern "rotate-to-explore" and "select" seems better suited on non-cluttered surfaces. This is mainly explained by the fact that users can perform interaction continuously, without interruptions that require dragging or moving interface elements, and eventually be focused on a set of TangiWheel instances already open. The third experiment, which included not only exploration and selection actions but also the movement of elements to other areas on the surface and requiring a higher number of collections, has shown that Touch techniques drop in performance instead. In this case, the use of pucks can be advantageous under such high demanding conditions, and the combination of techniques offered in the Hybrid techniques are especially relevant. As a conclusion, no single modality is therefore preferable in all cases and effectiveness varies according to the nature of the application and the actions to be carried out. A mixed input modality, such as that included in TangiWheel, offers advantages over existing tabletop-oriented widgets

for collection management in situations in which the best of both methods is required.

Once this base layer of the interaction toolkit was ready to support the construction of any other interface element, the technological development of the thesis has resulted in software that is able to load 2D entity-based ecosystem to be enacted and interacted on a digital tabletop by means of physics and rules. This is actually the first step forward towards the environment embraced by the proposal presented at the beginning of this dissertation. The remaining would consist of providing the corresponding integrated editing tools for the several aspects to deal with such as rule-based and physics behavior.

With the aim of covering a core set of these editing tools as well as their basis supporting them, several research actions have been conducted to contribute to the field. As many other proposals analyzed in the state of the art, we are interested in supporting some sort of programming to control the behavior of entities in the ecosystems and trigger to some extent the computational thinking in teenagers. A rule model has been adopted but tailored to support enhanced expressiveness in the specification of consequent action parameters. The challenge here consists of designing a rule model whose expressiveness for assignments is comparable to the one found in textual languages while maintaining a visual programming technique suitable to manage such expressiveness for non-programmers. The formalism to express these data transformations with assignment semantics followed a visual structure based on dataflow expressions. The power of this visual paradigm relies on the fact that users can manipulate components as independent pieces, and simply join them without complicated interface controls.

As end-users defining reactive behaviors with our rule model will have to effectively define these dataflows expressions, we have considered evaluating the ease of understanding by means of a user study facing the visual dataflow language against an equivalent textual syntax. In the study, the teenager students had to face four types of exercises in both languages, namely *Compare*, *Modify*, *Compute*, and *Write*. Significant differences are found in favor of the visual language for the *Modify* and *Compute* exercises. The high rate of missed exercises in the *Write* category suggests that writing expressions from scratch is really challenging in any language for non-programmers, and therefore the tools should carefully assist users by giving redundant information and visual feedback timely. Thus, for small and medium expressions as the ones dealt in the study, the dataflows can be considered as a suitable formalism to specify assignment expressions by non-programmers. Departing from these observa-

tions, we proceeded to design the rule editing tool. A mock-up setting has been used to come up with a set of interactions that would allow the edition of rules. Instead of visualizing the rule as a whole entity, the proposed mechanism offers partial views of the rules in terms of dataflows expressions, so that users only have to focus on one conceptual dimension of the rule at a given time. The editor has been implemented using the TangiWheel control to give access to the operators and other required components, and finger-based gestures to edit the data flows between operators.

Another relevant step of the creation process is the edition of entities that exhibit physically-based behavior. As our main aim in this research is to explore how the processes involved in our proposal can contribute to creativity, some simplifications have been taken. Instead of developing a vertical prototype of the entity editing tool, a reduced version only focused on the basic building blocks has been developed, leaving out superfluous elements such as costumes, etc. This simplification is essential to our purposes of conducting experiments to test the thinking-discussion-reflection loop in creative tasks. In the empirical study, teenagers were proposed two different tasks, namely freely creation of entity-like figures, and object-oriented creation of Rube-Goldberg machines, using the digital tabletop and a physical-only tabletop specifically designed for the experiments. This physical-only platform is made of a conglomerate tabletop and a toolbox with wooden blocks and connecting elements. The choice of using a tangible physical-only platform instead of one based on a desktop application was made on the assumption that, firstly, this non-technological platform is similar to some construction kits that are widely used during childhood, and secondly, it is better to have two similar platforms in terms of co-located user involvement and participation possibilities. Moreover, the tangible platform resembles the way people learnt by means of physical play sets in their childhood.

For the study, we operationalized some creativity factors, such as fluency of thinking, elaboration, novelty and motivation. The performed experiments have shown that the digital platform is an adequate mechanism to support creativity, as the groups working on it showed better performance in novelty and motivation. Other issues related to collaboration and interaction were also analyzed, including co-operation, retrieval fine adjustment and dominance, which has shown that the properties of an interactive surface tabletop are better suited to facilitating the sharing of objects and fair participation in conditions of co-operation by co-located participants.

7.2 Results of the Thesis

Here is the list of scientific publications derived from this dissertation:

- Alejandro Catalá, Patricia Pons, Javier Jaén, Jose A. Mocholí, Elena Navarro. A Meta-Model for DataFlow-Based Rules in Smart Environments: Evaluating User Comprehension and Performance. Invited submission to special issue “Software Engineering for Ambient Intelligence” in *Science of Computer Programming Journal* (minor revision submitted).
- Alejandro Catalá, Javier Jaén, Betsy van Dijk, and Sergi Jordà. 2012. Exploring tabletops as an effective tool to foster creativity traits. In *Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction (TEI '12)*, Stephen N. Spencer (Ed.). ACM, New York, NY, USA, 143-150. (Acceptance rate: 17%)
- Alejandro Catalá, Fernando García-Sanjuan, Javier Jaén, Jose A. Mocholí. TangiWheel: A Widget for Manipulating Collections on Tabletop Displays Supporting Hybrid Input Modality. *Journal Of Computer Science And Technology*. 27(4): 1 (JCR, IF: 0.656).
- Alejandro Catalá, Fernando García-Sanjuan, Jose Azorín, Javier Jaén, Jose A. Mocholí. Exploring Direct Communication and Manipulation on Interactive Surfaces to Foster Novelty in a Creative Learning Environment. *International Journal of Computer Science Research and Application 2012*, Vol. 02, Issue. 01 (Special Issue. Extended Selected Papers ICVL2011), pp. 15-24. ISSN 2012-9572. Acceptance rate: 20%.
- Alejandro Catalá, Javier Jaén, Adrià A. Martínez, Jose A. Mocholí. AGORAS: Exploring Creative Learning on Tangible User Interfaces. In *Proc. Compsac, IEEE 35th Annual Computer Software and Applications Conference*, pp.326-335, 2011. (Acceptance rate: 20%) (ISBN 978-0-7695-4439-7)
- Adrià A. Martínez, Alejandro Catalá, Javier Jaén, Jose A. Mocholí. CRETASICO: Un entorno de simulación de física para superficies interactivas. *Actas de las Jornadas de Ingeniería del Software y Bases de Datos, “JISBD 2011”*, pp. 975-988, 2011 (ISBN: 978-84-9749-486-1)
- Patricia Pons, Alejandro Catalá, Javier Jaén, Jose A. Mocholí. DaFRule: Un Modelo de Reglas Enriquecido mediante Flujos de Datos para la Definición Visual del Comportamiento Reactivo de Entidades Virtuales. *Actas de las Jornadas de Ingeniería del Software y Bases de Datos, “JISBD 2011”*, pp. 989-1002, 2011 (ISBN: 978-84-9749-486-1)

- Alejandro Catalá, Patricia Pons, Javier Jaén, Jose A. Mocholí. Evaluating User Comprehension of DataFlows in Reactive Rules for Event-Driven AMI Environments. *In Proc. V International Symposium on Ubiquitous Computing and Ambient Intelligence (UCAmI'11)*, pp. 337-344, 2011
- Fernando García-Sanjuan, Alejandro Catalá, Javier Jaén, Jose A. Mocholí. Una Interfaz Tangible para la Navegación Genérica de Estructuras de Colección. *Actas de las Jornadas de Ingeniería del Software y Bases de Datos, "JISBD 2011"*, pp. 1031-1044, 2011 (ISBN: 978-84-9749-486-1)
- Alejandro Catalá, Fernando García-Sanjuan Jose Azorín, Javier Jaén, Jose A. Mocholí. Exploring. Direct Communication and Manipulation on Interactive Surfaces to Foster Novelty in a Creative Learning Environment. *Proceedings of the International Conference on Virtual Learning (ICVL'11)*, pp. 373-379, - Distinguished with the *Excellence Award* 2011
- Alejandro Catalá, Javier Jaén. "A semantic model for reactive entities to support collaborative game design". *In Proceedings of the 2008 Conference on Future Play: Research, Play, Share. Future Play '08*. ACM, New York, NY, 216-219, 2008.
- Alejandro Catalá, Javier Jaén, Jose A. Mocholí. "Juegos ubicuos: experiencias de aprendizaje óptimas". En B. Gros eds. "Videojuegos y aprendizaje", pp 133-149, Graó, 2008. ISBN: 978-84-7827-539-7.
- Alejandro Catalá, Javier Jaén, and Jose A. Mocholí. "A Semantic Publish/Subscribe Approach for U-VR Systems Interoperation". In *Proceedings of the 2008 international Symposium on Ubiquitous Virtual Reality - ISUVR*. IEEE Computer Society, Washington, DC, 29-32, 2008.

7.3 Acknowledgements

Our thanks to the Alaquas city council and the clubhouse's managers who made it possible to conduct the experiment on tabletops for creativity described in Chapter 3. Our thanks also to the high school "Col•legi Parroquial D. José Lluch – Alboraya", especially the teachers and students who participated in the empirical study reported in Chapter 4. Finally, thanks to Polimedia/ASIC for providing the needed hardware infrastructure.

7.4 Further Research

Improvement is the basis of any research and development work. As any other thesis, it can be continuously improved in several ways. In the following, the most immediate future work related to the different parts of this thesis is introduced.

The user interface toolkit has been used in all the software that has been developed. It therefore establishes that the toolkit can successfully be used to build surface applications. However, a widget to be studied is the Magnitude control, which would be of interest for the community given its generic and multi-purpose nature. In this sense, a potential future work would consist of comparing the magnitude control with finger-based techniques to perform several tasks related to moving, resizing, scaling and rotating in different degrees of accuracy. The interest lies in the fact that while finger-based techniques are widely extended and accepted, they present some issues when they are supposed to be performed accurately. Thus, the exploration of cheap alternatives like ours would deserve special attention.

Regarding the ecosystem definition, it would be interesting to expand the concept of *stage* in order to support the explicit creation of background or backdrops. This creative task could be included, considering the editing by means of fingers and tangible tools like a painting tool. Also the support of larger stage sizes could be an interesting expansion as well as challenging since a transparent navigation technique should be considered.

Rules are a conceptual formalism understood by non-programmers according to several empirical studies, and we have obtained evidence on the comprehension of data flow expressions. However, the tangible edition tool that has been implemented has not been evaluated in experiments with teenagers. It is therefore a very important future work to be considered since it could really contribute to the field, basically with respect to the way that traditionally rule editors have considered rule editing.

Another immediate future work consists of carrying out more performance verification and optimization tasks of the rule processor. The aim would be to exploit cores and multi-threading possibilities as well as considering distributed processing that would allow using the rule model in other interesting domains such as Ambient Intelligence.

Finally, the experience in creating some game ecosystems in AGORAS raises the need for several improvements in the middleware. For example, rules could

be improved by supporting multiple operations in a single rule, or by facilitating aggregated operations (e.g. additions or any other calculation across a population). In addition, currently there is only response to the touches by the user, what means that ecosystems can only react to single touch events in terms of user interaction. Therefore, in order to expand the possibilities of providing more engaging interaction and support the construction of more advanced playful games, a core set of user interface events should be included based on meaningful specific touch gestures or operations with tangibles.

Appendix A. Experiments on Creativity

This appendix contains the forms used in the experiments described in Chapter 3 about creativity on tabletops. The first two forms are demographic surveys and record the relationship of participants in teams (#Q1 and #Q2). The form of type #Q3 was used to record the proposals in paper and pencil. Each participant used as many forms like this as proposals produced to the problem.

The forms #Q4 and #Q5 were used in the individual reflection and collective discussion places respectively. Participants had to register in these forms the identifiers of proposals produced at every reflection-discussion loop as they create the proposals in the #Q3 form. The #Q6 form was used by the experimenter in a testing platform to annotate which proposals were implemented by each team and facilitate the tracking of video-recordings.

Finally, the #Q7 survey was provided to each participant after each experiment to obtain the subjective perceptions on several aspects using the platforms.

#Q1: Participant form					ID	
Alias		Age		<input type="checkbox"/> Girl <input type="checkbox"/> Boy	<input type="checkbox"/> Left-handed <input type="checkbox"/> Right-handed <input type="checkbox"/> Ambidextrous	
Course/Stream		School				
Subject					Mark	
Previous experience in...	Never	Hardly ever	Once or twice per week	Almost every day	Every day	
... computers						
... touch-enabled devices						
... interactive surfaces						

Please indicate which of the following adjectives best describe yourself. Check all that apply.			
<input type="checkbox"/> Capable	<input type="checkbox"/> Artificial	<input type="checkbox"/> Clever	<input type="checkbox"/> Cautious
<input type="checkbox"/> Confident	<input type="checkbox"/> Egotistical	<input type="checkbox"/> Commonplace	<input type="checkbox"/> Humorous
<input type="checkbox"/> Conservative	<input type="checkbox"/> Individualistic	<input type="checkbox"/> Conventional	<input type="checkbox"/> Informal
<input type="checkbox"/> Dissatisfied	<input type="checkbox"/> Insightful	<input type="checkbox"/> Suspicious	<input type="checkbox"/> Honest
<input type="checkbox"/> Intelligent	<input type="checkbox"/> Well-mannered	<input type="checkbox"/> Wide interests	<input type="checkbox"/> Inventive
<input type="checkbox"/> Original	<input type="checkbox"/> Narrow interests	<input type="checkbox"/> Reflective	<input type="checkbox"/> Sincere
<input type="checkbox"/> Resourceful	<input type="checkbox"/> Self-confident	<input type="checkbox"/> Sexy	<input type="checkbox"/> Submissive
<input type="checkbox"/> Snobbish	<input type="checkbox"/> Unconventional		

⁹ Gough, H. G. (1979). A creative personality scale for the Adjective Check List. *Journal of Personality and Social Psychology*, 37, 1398-1405.

#Q2: Team form			GID	
Member Alias (blue)		Member Alias (black)		
Member Alias (red)		Member Alias (green)		
Group Alias			Color	
We know each other because... (Check all that apply)				
<input type="checkbox"/> ... we are friends				
<input type="checkbox"/> ... we attend the same school				
<input type="checkbox"/> ... we are classmates				
<input type="checkbox"/> ... we are partners at a sport team				
<input type="checkbox"/> ... we are partners in activities usually carried out in this club				
<input type="checkbox"/> ... (other case to specify):				

#Q3 Propo- sal Form	Task	Alias	Number	#Index
Description and rationale				
Sketch				

#Q5 Use of the Collective discussion Place		Task	Team Alias	
Start	End	Identifiers of the produced proposals		

#Q7	<input type="checkbox"/> Physical-only <input type="checkbox"/> Digital	Alias				Task	
#Q	Cuestión	Strongly disagree	Disagree	Neither disagree or agree	Agree	Atringly Agree	
1	I find this platform easy to use.						
2	Learning to handle this platform is easy.						
3	Interacting with this platform is clear and intelisible.						
4	I find this platform flexible to use.						
5	I find It easy to get the platform to do what I mean.						
6	Use this platform requires significant mental effort.						
7	I participated more than my partner in designing proposals.						
8	I participated more than my partner in testing solutions in the platform.						
9	The platform keep me more motivated to participate.						
10	My proposals are novel.						
11	The platform allowed me to créate porposals in paper that otherwise they would have been devised.						
12	The platform encourage me to explore the ideas porposed in paper.						
13	The platform allowed me to						

	create complex solutions.					
14	My proposals are varied among them.					
15	The platform allowed me to express myself as more creative.					
16	The use of this platform allowed me complete the creation tasks quickly.					
17	In general, the proposals of my partner were very different from mine.					

Appendix B. DataFlow Comprehension

The current appendix contains the basic worksheet used in the user-based study on dataflow comprehension. This or a variant shuffling the exercises was handed out to students in the evaluation session.

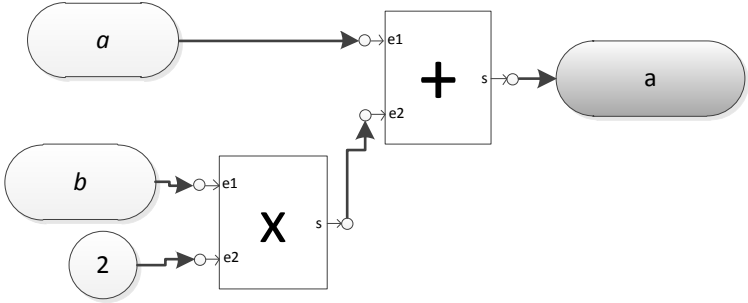
Test Languages

(Model 10)

Notes and instructions:

- The test is to be carried out with the material included here only.
- Only black/blue pens or pencils can be used.
- The current worksheet consists of three parts: Demographic survey (page 5), Exercises (page 6 through 21), y final surveys (pages 22, 23 y 24).
- All questions in the demographic and final surveys must be answered.
- When administering this test, about 10 minutes must be reserved, conveniently distributed at the beginning and at the end. This time are intended to fill in the demographic survey and answer the final questionnaires.
- Regarding the exercises, they must be completed in order as many as possible within the available time.
- If you completely finish this worksheet, you will be provided with an extra worksheet with further exercises.
- Each exercise has two especial fields in its header, namely Start and End. In these, you have to indicate the time in which you start and end the exercise respectively, following the format minutes:seconds (mm:ss). To this purpose, a monitor in the classroom will display the time. You will have to have a look at the monitor, and immediately to write down the time being displayed, without making any other action or calculating or estimating times. It means that you simply have to note down the instant you see in the monitor.
- Even if you are not able to get an answer to an exercise, you will have to indicate the Start and End times.
- You are allowed to write on the exercise pages (e.g. supporting calculations or expressions).
- There are four exercise categories clearly differentiated: COMPUTE, COMPARE, MODIFY, and WRITE. The COMPUTE exercises require calculating the outcome value of the expression given a set of parameter values. The COMPARE exercises require determining whether two given expressions are equivalent or not according to logics. The MODIFY category asks for making all the necessary changes on the expression to mean as a statement provided. Finally, the WRITE category requires writing from the scratch an expression to mean as the provided statement.

Examples (The example outcome is highlighted in dark grey):

Example 1	COMPUTE	Start:	End:
<p>Assuming the following attribute values: <i>a</i> equals 6, <i>b</i> equals 3.</p> <p>Calculate the result of the following expression:</p>			
			
<p>Outcome: 12</p>		<input type="checkbox"/> I tried but I couldn't solve it.	

Example 2	COMPARE	Start:	End:
<p>Given the following expressions, indicate whether they are equivalent:</p> $z \leftarrow -b + c + a$ $z \leftarrow -(c + a) + b$			
<p>Outcome: YES</p>		<input type="checkbox"/> I tried but I couldn't solve it	

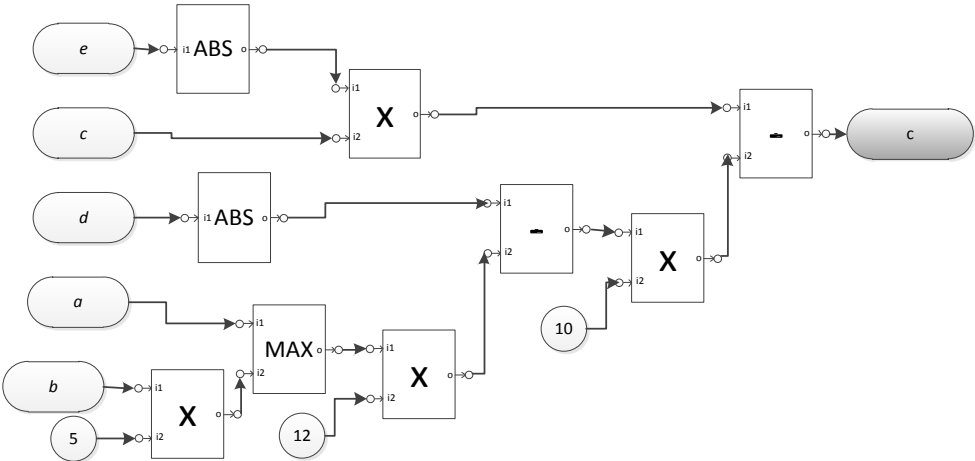
Example 3	MODIFY	Start:	End:
<p>Make the necessary changes to the original expression so that the proposed statement would be correctly specified:</p> <p>“a is calculated according to the following condition: b is less than 15 and at the same time either c is greater than 19 or d is greater than the maximum between 10 and b”.</p>			
$a \leftarrow (b < 15 \vee c > 19) \circ d > \text{MAX}(10, b)$ <p>It would be rewritten as:</p> $a \leftarrow b < 15 \vee (c > 19 \circ d > \text{MAX}(10, b))$			
			<input type="checkbox"/> I tried but I couldn't solve it

Example 4	WRITE	Start:	End:
<p>Write the following expression in the <u>visual</u> language: “a is calculated as the value that had a added to the product of b and 2”.</p>			
			<input type="checkbox"/> I tried but I couldn't solve it

Demographic survey	
Teacher	
Course	
Group	
Number	
Stream	
Age	
Gender	<input type="checkbox"/> Girl <input type="checkbox"/> Boy
Tell us if you had any previous experience on programming:	

Please, indicate your degree of agreement or disagreement with the following statements:

	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
I like Computing					
I'm good at Computing					
I like Mathematics					
I'm good at Mathematics					
I like Languages					
I'm good at Languages					

Exercise 11003	COMPUTE	Start:	End:
<p>Assuming the following attribute values:</p> <p>a equals 9, b equals 2, c equals 8, d equals -130, e equals 7.</p> <p>Calculate the result of the following expression:</p> 			
Outcome:		<input type="checkbox"/> I tried but I couldn't solve it	

Exercise 21024	MODIFY	Start:	End:
<p>Make the necessary changes to the original expression so that the proposed statement would be correctly specified:</p> <p>“c is calculated according to the following condition: a is less than 10 or greater than 12, and b is greater than 7 or less than 11.</p>			
$c \leftarrow a < 10 \text{ OR } (a > 12 \text{ AND } b > 7) \text{ AND } b < 11$			
		<input type="checkbox"/> I tried but I couldn't solve it	

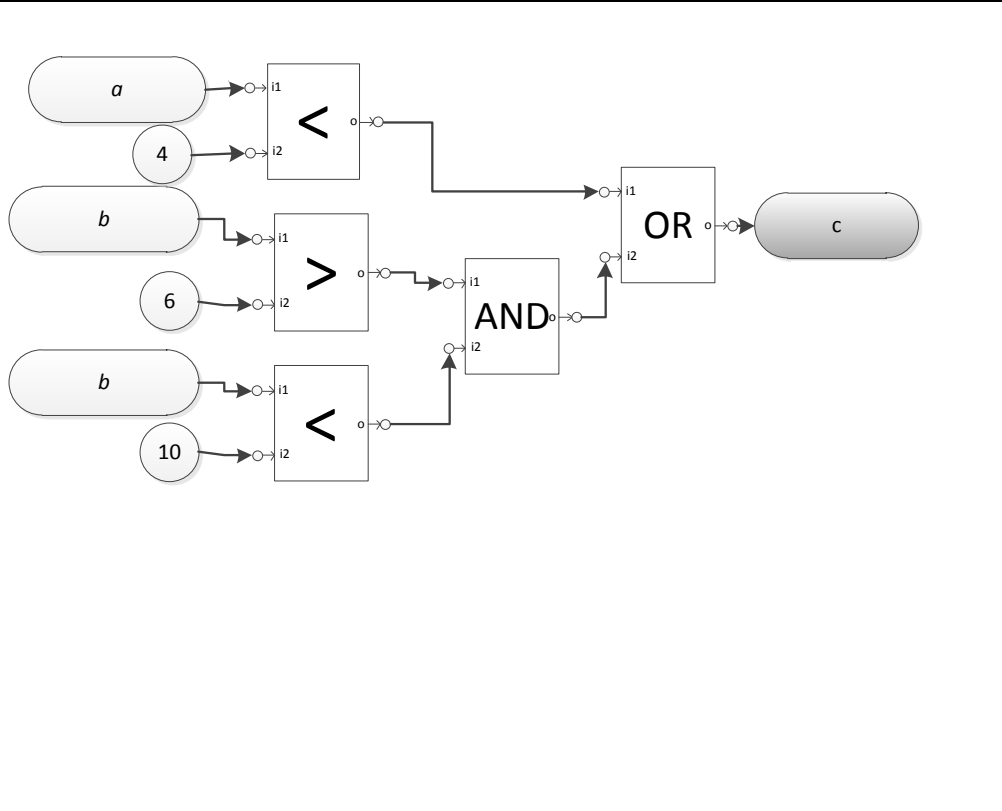
Exercise 11029	COMPARE	Start:	End:
<p>Given the following expressions, indicate whether they are equivalent:</p>			
<p>Outcome:</p>		<input type="checkbox"/> I tried but I couldn't solve it	

Exercise 21018	COMPUTE	Start:	End:
<p>Assuming the following attribute values:</p> <p><i>human1.health</i> equals 7,</p> <p><i>world.danger</i> equals 3,</p> <p><i>ogre1.has_potion</i> equals No!,</p> <p><i>ogre1.health</i> equals 5,</p> <p>Calculate the result of the following expression:</p> <p>$ogre1.needs_potion \leftarrow (human1.health > 6 + world.danger) \text{ OR } 18 > ABS(ogre1.health - human1.health \times 3) \text{ AND NOT}(ogre1.has_potion)$</p>			
Outcome:		<input type="checkbox"/> I tried but I couldn't solve it	

Exercise 11023	MODIFY	Start:	End:
----------------	--------	--------	------

Make the necessary changes to the original expression so that the proposed statement would be correctly specified:

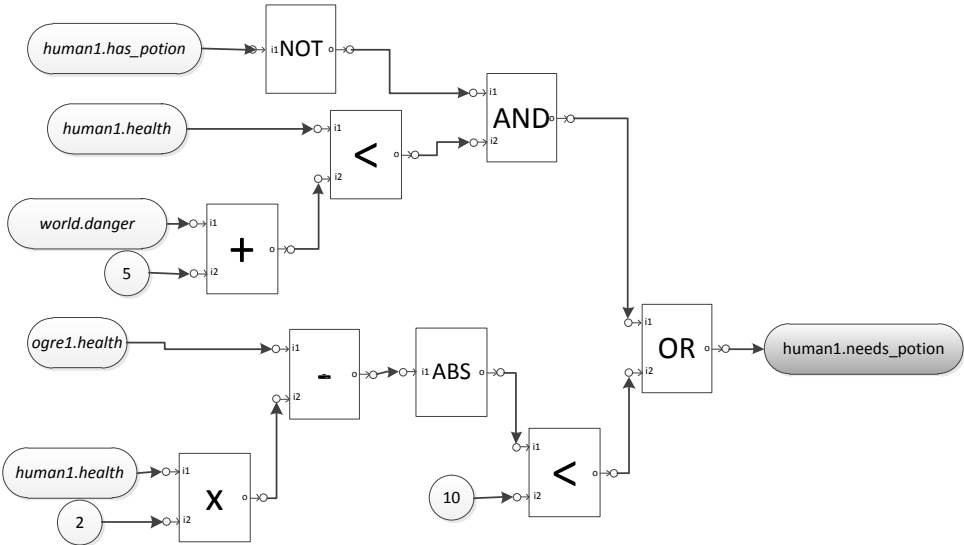
“c is calculated according to the following condition: a is less than 4 and b is greater than 6 and less than 10”.



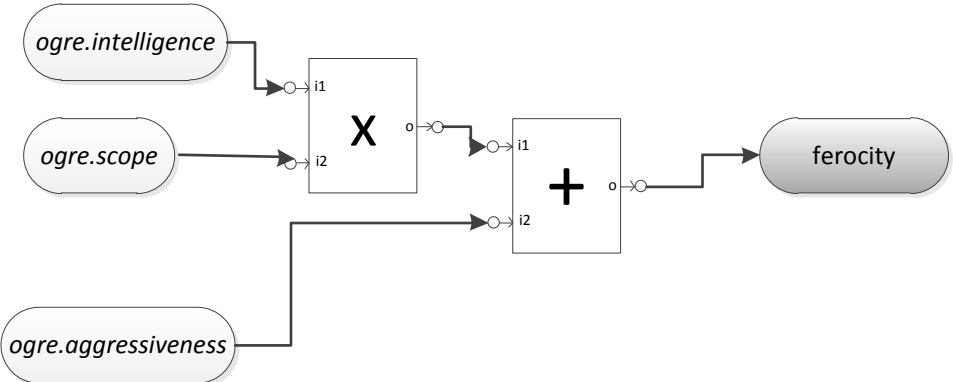
I tried but I couldn't solve it

Exercise 11034	WRITE	Start:	End:
Write the following expression in the visual language: z is the minimum between 10 and the maximum of a and the sum of b and c .			
			<input type="checkbox"/> I tried but I couldn't solve it

Exercise 21030	COMPARE	Start:	End:
Given the following expressions, indicate whether they are equivalent:			
$z \leftarrow b + a + d \times c \times z + \text{MAX}(12, a + b \times c) \times c$			
<hr/>			
$z \leftarrow (b + a) + d \times (z + \text{MAX}(12, a + b \times c)) \times c$			
Outcome:	<input type="checkbox"/> I tried but I couldn't solve it		

Exercise 11017	COMPUTE	Start:	End:
<p>Assuming the following attribute values:</p> <p><i>human1.health</i> equals 8,</p> <p><i>human1.has_potion</i> equals <i>Yes!</i>,</p> <p><i>world.danger</i> equals 4,</p> <p><i>ogre1.health</i> equals 5.</p> <p>Calculate the result of the following expression:</p> 			
Outcome:	<input type="checkbox"/> I tried but I couldn't solve it		

Exercise 21026	COMPARE	Start:	End:
Given the following expressions, indicate whether they are equivalent:			
$c \leftarrow a < 3 \text{ OR } b > 5 \text{ AND } b < 12$			
<hr/>			
$c \leftarrow b < 12 \text{ AND } b > 5 \text{ OR } a < 3$			
Outcome:		<input type="checkbox"/> I tried but I couldn't solve it	

Exercise 11020	MODIFY	Start:	End:
<p>Make the necessary changes to the original expression so that the proposed statement would be correctly specified:</p> <p>“The <i>ferocity</i> is calculated as the product of the ogre’s <i>intelligence</i> and the sum of its <i>scope</i> and <i>aggressiveness</i>”.</p>			
 <pre>graph LR; I(ogre.intelligence) --> X[X]; S(ogre.scope) --> X; X --> P1(()); A(ogre.aggressiveness) --> P2(()); S --> P2; P1 --> P3(()); P2 --> P3; P3 --> F(ferocity);</pre>			
			<input type="checkbox"/> I tried but I couldn't solve it

Exercise 11036	WRITE	Start:	End:
<p>Write the following expression in the visual language:</p> <p>“The condition c is calculated as follows: either the human intelligence is greater than 5 and the ogre strength less than 7, or the ogre strength at least doubles the human strength while the human does not have the potion”</p>			
<div style="border: 1px solid black; height: 350px; width: 100%;"></div>			
			<input type="checkbox"/> I tried but I couldn't solve it

Exercise 11025	COMPARE	Start:	End:
<p>Given the following expressions, indicate whether they are equivalent:</p>			
<p>Outcome:</p>	<input type="checkbox"/> I tried but I couldn't solve it		

Exercise 21004	COMPUTE	Start:	End:
<p>Assuming the following attribute values:</p> <p>a equals 155, b equals 3, c equals 7, d equals -8, e equals 10.</p> <p>Calculate the result of the following expression:</p>			
$c \leftarrow \text{ABS}(d) \times c - 10 \times (\text{ABS}(a) - 14 \times \text{MAX}(e, b \times (8 - 5)))$			
Outcome:		<input type="checkbox"/> I tried but I couldn't solve it	

Exercise 21019	MODIFY	Start:	End:
Make the necessary changes to the original expression so that the proposed statement would be correctly specified: “The violence is calculated as the product of the human irritation with the sum of its strength and aggressiveness”.			
$violence \leftarrow human.irritation \times human.strength + human.aggressiveness$			
		<input type="checkbox"/> I tried but I couldn't solve it	

Exercise 21033	WRITE	Start:	End:
Write the following expression in the textual language: z is the minimum between 11 and the maximum of b and the sum of a with c .			
		<input type="checkbox"/> I tried but I couldn't solve it	

Focusing only on what you have learnt about the textual language, indicate your agreement degree with the following sentences:

		Strongly disagree	Disagree	Neutral	Agree	Strongly agree
1	I understand the meaning of the expressions					
2	I consider easy to understand the different components of the language (operators, variables, etc.)					
3	I find it difficult to calculate expressions written in this language.					
4	I consider easy to calculate the outcome of expressions written in this language.					
5	I consider difficult to modify already existing expressions.					
6	I find it difficult to follow and obtaining outcomes from the variables values.					
7	The language was easy for me to learn.					
8	If find it easy to modify already existing expressions.					
9	It was hard to learn how to use the language.					
10	I find it easy to write expressions with the language					
11	It is hard to modify existing expressions					
12	I consider that following and obtaining values from variables to the outcome is easy.					
13	I find it easy to determine whether two expressions are equivalent or not.					
14	It was hard to find out if two expressions were equivalent or not.					

Focusing only on what you have learnt about the visual language, indicate your agreement degree with the following sentences:

		Strongly disagree	Disagree	Neutral	Agree	Strongly agree
1	I understand the meaning of the expressions					
2	I consider easy to understand the different components of the language (operators, variables, etc.)					
3	I find it difficult to calculate expressions written in this language.					
4	I consider easy to calculate the outcome of expressions written in this language.					
5	I consider difficult to modify already existing expressions.					
6	I find it difficult to follow and obtaining outcomes from the variables values.					
7	The language was easy for me to learn.					
8	If find it easy to modify already existing expressions.					
9	It was hard to learn how to use the language.					
10	I find it easy to write expressions with the language					
11	It is hard to modify existing expressions					
12	I consider that following and obtaining values from variables to the outcome is easy.					
13	I find it easy to determine whether two expressions are equivalent or not.					
14	It was hard to find out if two expressions were equivalent or not.					

Indicate with an X, which language fits better to each one of the following situations:

		Textual	Visual
1	It is easier to learn the language...		
2	I find it easier to understand the expressions written using...		
3	I understand the different elements better (operators, variables, etc.) in...		
4	I learned the language faster...		
5	It is easier to calculate outcome values with the language...		
6	I am able to write expressions using the language more easily in...		
7	It is easier for me to know whether two expressions are equivalent using the language...		
8	When modifying an expression so that it represents a given statement I find out faster what needs to be modified using the language...		

Appendix C. Pong AGORAS Specification

This appendix contains the specification in full for the Pong AGORAS game.

Pong.eco file (primary specification file):

```
ECOSYSTEM NAME Pong
STRUCTURE_TYPES
  STRUCTURE_TYPE NAME BallStructure PATH "Content/Simulator/BallStructure.st" END_STRUCTURE_TYPE
  STRUCTURE_TYPE NAME PaddleStructure PATH "Content/Simulator/PaddleStructure.st" END_STRUCTURE_TYPE
  STRUCTURE_TYPE NAME DisplayStructure PATH "Content/Simulator/DisplayStructure.st" END_STRUCTURE_TYPE
  STRUCTURE_TYPE NAME GoalStructure PATH "Content/Simulator/GoalStructure.st" END_STRUCTURE_TYPE
END_STRUCTURE_TYPES

ENTITY_TYPES
ENTITY_TYPE BallType
  PHYSICAL_RESPONSE true
  REBOUND true
  STRUCTURE_TYPE BallStructure
  PROTO_COSTUMES
  PROTO_COSTUME BallCostume PATH "Content/Simulator/BallCostume.pc" END_PROTO_COSTUME
  END_PROTO_COSTUMES
END_ENTITY_TYPE
ENTITY_TYPE PaddleType
  PHYSICAL_RESPONSE true
  IS_FINGER_SENSITIVE true
  IS_TAG_SENSITIVE true
  STRUCTURE_TYPE PaddleStructure
  PROTO_COSTUMES
  PROTO_COSTUME PaddleCostume PATH "Content/Simulator/PaddleCostume.pc" END_PROTO_COSTUME
  END_PROTO_COSTUMES
END_ENTITY_TYPE
ENTITY_TYPE GoalType
  PHYSICAL_RESPONSE true
  STRUCTURE_TYPE GoalStructure
  PROTO_COSTUMES
  PROTO_COSTUME Faded PATH "Content/Simulator/Faded.pc" END_PROTO_COSTUME
  END_PROTO_COSTUMES
END_ENTITY_TYPE
ENTITY_TYPE DisplayType
  PHYSICAL_RESPONSE false
  STRUCTURE_TYPE DisplayStructure
  PROTO_COSTUMES
  PROTO_COSTUME DisplayCostume0 PATH "Content/Simulator/DisplayCostume0.pc" END_PROTO_COSTUME
  PROTO_COSTUME DisplayCostume1 PATH "Content/Simulator/DisplayCostume1.pc" END_PROTO_COSTUME
  PROTO_COSTUME DisplayCostume2 PATH "Content/Simulator/DisplayCostume2.pc" END_PROTO_COSTUME
  PROTO_COSTUME DisplayCostume3 PATH "Content/Simulator/DisplayCostume3.pc" END_PROTO_COSTUME
  PROTO_COSTUME DisplayCostume4 PATH "Content/Simulator/DisplayCostume4.pc" END_PROTO_COSTUME
  PROTO_COSTUME DisplayCostume5 PATH "Content/Simulator/DisplayCostume5.pc" END_PROTO_COSTUME
  PROTO_COSTUME DisplayCostume6 PATH "Content/Simulator/DisplayCostume6.pc" END_PROTO_COSTUME
  PROTO_COSTUME DisplayCostume7 PATH "Content/Simulator/DisplayCostume7.pc" END_PROTO_COSTUME
  PROTO_COSTUME DisplayCostume8 PATH "Content/Simulator/DisplayCostume8.pc" END_PROTO_COSTUME
  PROTO_COSTUME DisplayCostume9 PATH "Content/Simulator/DisplayCostume9.pc" END_PROTO_COSTUME
  PROTO_COSTUME DisplayCostume10 PATH "Content/Simulator/DisplayCostume10.pc" END_PROTO_COSTUME
  END_PROTO_COSTUMES
```

```

PROPERTIES
  PROPERTY NAME Count TYPE Integer END_PROPERTY
END_PROPERTIES
END_ENTITY_TYPE
END_ENTITY_TYPES

ENTITIES
ENTITY
  NAME Ball TYPE BallType COSTUME BallCostume
END_ENTITY
ENTITY
  NAME Paddle1 TYPE PaddleType COSTUME PaddleCostume
END_ENTITY
ENTITY
  NAME Paddle2 TYPE PaddleType COSTUME PaddleCostume
END_ENTITY
ENTITY
  NAME Display1 TYPE DisplayType COSTUME DisplayCostume0
END_ENTITY
ENTITY
  NAME Display2 TYPE DisplayType COSTUME DisplayCostume0
END_ENTITY
ENTITY
  NAME Goal1 TYPE GoalType COSTUME Faded
END_ENTITY
ENTITY
  NAME Goal2 TYPE GoalType COSTUME Faded
END_ENTITY
END_ENTITIES

STAGES
STAGE
  NAME Table
  TOP_BORDER true
  BOTTOM_BORDER true
  LEFT_BORDER false
  RIGHT_BORDER false
PARTICIPATIONS
  ENTITY BallType::Ball
    PROPERTY NAME Position TYPE Vector2 VALUE 0,0 0,0 END_PROPERTY
    PROPERTY NAME InitialSpeed TYPE Vector2 VALUE -500,0 500,0 END_PROPERTY
  END_ENTITY
  ENTITY PaddleType::Paddle1
    PROPERTY NAME Position TYPE Vector2 VALUE -480,0 0,0 END_PROPERTY
  END_ENTITY
  ENTITY PaddleType::Paddle2
    PROPERTY NAME Position TYPE Vector2 VALUE 480,0 0,0 END_PROPERTY
  END_ENTITY
  ENTITY DisplayType::Display1
    PROPERTY NAME Position TYPE Vector2 VALUE -50,0 350,0 END_PROPERTY
    PROPERTY NAME Count TYPE Integer VALUE 0 END_PROPERTY
  END_ENTITY
  ENTITY DisplayType::Display2
    PROPERTY NAME Position TYPE Vector2 VALUE 50,0 350,0 END_PROPERTY
    PROPERTY NAME Count TYPE Integer VALUE 0 END_PROPERTY
  END_ENTITY
  ENTITY GoalType::Goal1
    PROPERTY NAME Position TYPE Vector2 VALUE -540,0 0,0 END_PROPERTY

```

```

END_ENTITY
ENTITY GoalType::Goal2
  PROPERTY NAME Position TYPE Vector2 VALUE 540,0 0,0 END_PROPERTY
END_ENTITY
END_PARTICIPATIONS

REACTIVE_RULE NAME GameOver
PRIORITY 10
EVENT PropertyValueChanged
SOURCE ENTITY_TYPE DisplayType
TARGET ENTITY TableStageType::Table
OPERATION ACTION PauseSimulation
PRECONDITION_DATA_PROCESS
  DATA_PROCESS
  NODES
    CONSTANT_PROVIDER NAME finalScore TYPE Integer VALUE 10
    POSITION 0,0 0,0
  END_CONSTANT_PROVIDER
  DATA_PROCESSOR_INSTANCE NAME ==1 TYPE EQ
  POSITION 0,0 0,0
  END_DATA_PROCESSOR_INSTANCE
  END_NODES
  DATA_FLOWS
    DATA_FLOW FROM SOURCE.Count::out TO ==1::in1 END_DATA_FLOW
    DATA_FLOW FROM finalScore::out TO ==1::in2 END_DATA_FLOW
    DATA_FLOW FROM ==1::out TO condition_result::in1 END_DATA_FLOW
  END_DATA_FLOWS
  END_DATA_PROCESS
  END_PRECONDITION_DATA_PROCESS
END_REACTIVE_RULE

REACTIVE_RULE NAME Score1Increment
EVENT Collision
SOURCE ENTITY_TYPE BallType
TARGET ENTITY DisplayType::Display1
OPERATION PROPERTY Count
PRECONDITION_DATA_PROCESS
  DATA_PROCESS
  NODES
    CONSTANT_PROVIDER NAME ConstantEntityGoal2 TYPE GoalType VALUE GoalType::Goal2
    POSITION 0,0 0,0
  END_CONSTANT_PROVIDER
  DATA_PROCESSOR_INSTANCE NAME ==1 TYPE EQ
  POSITION 0,0 0,0
  END_DATA_PROCESSOR_INSTANCE
  END_NODES
  DATA_FLOWS
    DATA_FLOW FROM ConstantEntityGoal2::out TO ==1::in1 END_DATA_FLOW
    DATA_FLOW FROM EVENT.SlaveEntity::out TO ==1::in2 END_DATA_FLOW
    DATA_FLOW FROM ==1::out TO condition_result::in1 END_DATA_FLOW
  END_DATA_FLOWS
  END_DATA_PROCESS
  END_PRECONDITION_DATA_PROCESS
  OPERATION_DATA_PROCESS
  DATA_PROCESS
  NODES
    CONSTANT_PROVIDER NAME constant1 TYPE Integer VALUE 1
    POSITION 0,0 0,0

```

```

END_CONSTANT_PROVIDER
DATA_PROCESSOR_INSTANCE NAME +i1 TYPE ADD
    POSITION 0,0 0,0
END_DATA_PROCESSOR_INSTANCE
END_NODES
DATA_FLOWS
    DATA_FLOW FROM constant1::out TO +i1::in1 END_DATA_FLOW
    DATA_FLOW FROM TARGET.Count::out TO +i1::in2 END_DATA_FLOW
    DATA_FLOW FROM +i1::out TO TARGET.Count::in1 END_DATA_FLOW
END_DATA_FLOWS
END_DATA_PROCESS
END_OPERATION_DATA_PROCESS
END_REACTIVE_RULE

REACTIVE_RULE NAME Score2Increment
EVENT Collision
SOURCE ENTITY_TYPE BallType
TARGET ENTITY DisplayType::Display2
OPERATION PROPERTY Count
PRECONDITION_DATA_PROCESS
DATA_PROCESS
    NODES
        CONSTANT_PROVIDER NAME ConstantEntityGoal1 TYPE GoalType VALUE GoalType::Goal1
            POSITION 0,0 0,0
        END_CONSTANT_PROVIDER
        DATA_PROCESSOR_INSTANCE NAME ==2 TYPE EQ
            POSITION 0,0 0,0
        END_DATA_PROCESSOR_INSTANCE
    END_NODES
    DATA_FLOWS
        DATA_FLOW FROM ConstantEntityGoal1::out TO ==2::in1 END_DATA_FLOW
        DATA_FLOW FROM EVENT.SlaveEntity::out TO ==2::in2 END_DATA_FLOW
        DATA_FLOW FROM ==2::out TO condition_result::in1 END_DATA_FLOW
    END_DATA_FLOWS
    END_DATA_PROCESS
END_PRECONDITION_DATA_PROCESS
OPERATION_DATA_PROCESS
DATA_PROCESS
    NODES
        CONSTANT_PROVIDER NAME constant1_2 TYPE Integer VALUE 1
            POSITION 0,0 0,0
        END_CONSTANT_PROVIDER
        DATA_PROCESSOR_INSTANCE NAME +i2 TYPE ADD
            POSITION 0,0 0,0
        END_DATA_PROCESSOR_INSTANCE
    END_NODES
    DATA_FLOWS
        DATA_FLOW FROM constant1_2::out TO +i2::in1 END_DATA_FLOW
        DATA_FLOW FROM TARGET.Count::out TO +i2::in2 END_DATA_FLOW
        DATA_FLOW FROM +i2::out TO TARGET.Count::in1 END_DATA_FLOW
    END_DATA_FLOWS
    END_DATA_PROCESS
END_OPERATION_DATA_PROCESS
END_REACTIVE_RULE

REACTIVE_RULE NAME ChangeCostume
PRIORITY 0
EVENT PropertyValueChanged

```

```

SOURCE ENTITY_TYPE DisplayType
TARGET ENTITY_TYPE DisplayType
OPERATION ACTION ChangeCostume
PRECONDITION_DATA_PROCESS
  DATA_PROCESS
    NODES
      CONSTANT_PROVIDER NAME propName TYPE String VALUE "Count"
        POSITION 0,0 0,0
      END_CONSTANT_PROVIDER
      DATA_PROCESSOR_INSTANCE NAME ==2 TYPE EQ
        POSITION 0,0 0,0
      END_DATA_PROCESSOR_INSTANCE
    END_NODES
  DATA_FLOWS
      DATA_FLOW FROM propName::out TO ==2::in1 END_DATA_FLOW
      DATA_FLOW FROM EVENT.Property::out TO ==2::in2 END_DATA_FLOW
      DATA_FLOW FROM ==2::out TO condition_result::in1 END_DATA_FLOW
    END_DATA_FLOWS
  END_DATA_PROCESS
END_PRECONDITION_DATA_PROCESS
FILTER_PRECONDITION_DATA_PROCESS
  DATA_PROCESS
    NODES
      DATA_PROCESSOR_INSTANCE NAME ==3 TYPE EQ
        POSITION 0,0 0,0
      END_DATA_PROCESSOR_INSTANCE
    END_NODES
  DATA_FLOWS
      DATA_FLOW FROM EVENT.Entity::out TO ==3::in1 END_DATA_FLOW
      DATA_FLOW FROM TARGET.Value::out TO ==3::in2 END_DATA_FLOW
      DATA_FLOW FROM ==3::out TO condition_result::in1 END_DATA_FLOW
    END_DATA_FLOWS
  END_DATA_PROCESS
END_FILTER_PRECONDITION_DATA_PROCESS
OPERATION_DATA_PROCESS
  DATA_PROCESS PARAM TargetEntity
    NODES
      END_NODES
    DATA_FLOWS
      DATA_FLOW FROM TARGET.Value::out TO ChangeCostume.TargetEntity::in1 END_DATA_FLOW
    END_DATA_FLOWS
  END_DATA_PROCESS
  DATA_PROCESS PARAM CostumeName
    NODES
      DATA_PROCESSOR_INSTANCE NAME concatenator TYPE CONCAT
        POSITION 0,0 0,0
      END_DATA_PROCESSOR_INSTANCE
      CONSTANT_PROVIDER NAME partialName TYPE String VALUE "DisplayCostume"
        POSITION 0,0 0,0
      END_CONSTANT_PROVIDER
      /*DATA_PROCESSOR_INSTANCE NAME rand TYPE RAND
        POSITION 0,0 0,0
      END_DATA_PROCESSOR_INSTANCE
      CONSTANT_PROVIDER NAME min TYPE int VALUE 1
        POSITION 0,0 0,0
      END_CONSTANT_PROVIDER
      CONSTANT_PROVIDER NAME max TYPE int VALUE 10
        POSITION 0,0 0,0

```

```

    END_CONSTANT_PROVIDER*/
  END_NODES
  DATA_FLOWS
    DATA_FLOW FROM partialName::out TO concatenator::in1 END_DATA_FLOW
    DATA_FLOW FROM SOURCE.Count::out TO concatenator::in2 END_DATA_FLOW
    //DATA_FLOW FROM min::out TO rand::in1 END_DATA_FLOW
    //DATA_FLOW FROM max::out TO rand::in2 END_DATA_FLOW
    //DATA_FLOW FROM rand::out TO concatenator::in2 END_DATA_FLOW
    DATA_FLOW FROM concatenator::out TO ChangeCostume.CostumeName::in1 END_DATA_FLOW
  END_DATA_FLOWS
  END_DATA_PROCESS
  END_OPERATION_DATA_PROCESS
  END_REACTIVE_RULE

  REACTIVE_RULE NAME ResetBallPosition
  PRIORITY 15
  EVENT PropertyValueChanged
  SOURCE ENTITY_TYPE DisplayType
  TARGET ENTITY BallType::Ball
  OPERATION PROPERTY Position
  OPERATION_DATA_PROCESS
  DATA_PROCESS
  NODES
    CONSTANT_PROVIDER NAME initialPos TYPE Vector2 VALUE 0,0,0
    POSITION 0,0,0
    CONSTANT_PROVIDER
  END_NODES
  DATA_FLOWS
    DATA_FLOW FROM initialPos::out TO TARGET.Position::in1 END_DATA_FLOW
  END_DATA_FLOWS
  END_DATA_PROCESS
  END_OPERATION_DATA_PROCESS
  END_REACTIVE_RULE
  END_STAGE
  END_STAGES

  CURRENT_STAGE Table

  END_ECOSYSTEM

```

Structure-Type specification files

BallStructure.st file:

```

STRUCTURE TYPE
BallStructure

COMPONENTS 1
ELLIPSE
# Id
0
# layer
0
# static {-1 = static; 0..n= fricCoef }
0

```

```

# position
0 0
# radius
20 20
# rotation
0
JOINTS 0

```

PaddleStructure.st

```

STRUCTURE TYPE
PaddleStructure

COMPONENTS 1
RECTANGLE
# Id
0
# layer
0
# static {-1 = static; 0..n= fricCoef }
-1
# position
0 0
# dimensions x and y
50 100
# rotation
0
JOINTS 0

```

GoalStructure.st

```

STRUCTURE TYPE
GoalStructure

COMPONENTS 1
RECTANGLE
# Id
0
# layer
0
# static {-1 = static; 0..n= fricCoef }
-1
# position
0 0
# dimensions x and y
60 768
# rotation
0
JOINTS 0

```


DisplayStructure.st

```

STRUCTURE TYPE
DisplayStructure

COMPONENTS 1
RECTANGLE
# id
0
# layer
0
# static {-1 = static; 0..n= fricCoef }
-1
# position
0 0
# dimensions x and y
50 50
# rotation
0

JOINTS 0

```

ProtoCostume files

PaddleCostume.pc

```

PROTOCOLCOSTUME
PaddleCostume

STRUCTURE PaddleStructure

SKINS 1
Content\Simulator\RectangleSkin.png

# id
0
# offset
0 0
# scale
1 1
# rotation
0

```

Faded.pc

```

PROTOCOLCOSTUME
Faded

STRUCTURE GoalStructure

SKINS 1
Content\Simulator\thinRectSkin.png

# id
0

```

```
# offset
0 0
# scale
1 1
# rotation
0
```

DisplayCostume0.pc – DisplayCostume10.pc

```
PROTOCOLCOSTUME
DisplayCostume0

STRUCTURE DisplayStructure

SKINS 1
Content\Simulator\DisplaySkin00.png

# id
0
# offset
0 0
# scale
1 1
# rotation
0
```

```
PROTOCOLCOSTUME
DisplayCostume1

STRUCTURE DisplayStructure

SKINS 1
Content\Simulator\DisplaySkin01.png

# id
0
# offset
0 0
# scale
1 1
# rotation
0
```

BallCostume.pc

```
PROTOCOLCOSTUME
BallCostume




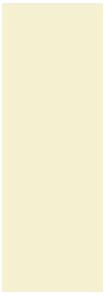

STRUCTURE BallStructure

SKINS 1
Content\Simulator\BallSkin.png

# id
0
# offset
```

```
0 0
# scale
1 1
# rotation
0
```

Image files for Skins

<p>BallSkin.png</p> 	<p>DisplaySkin00.png</p> 	<p>DisplaySkin01.png</p> 	<p>DisplaySkin02.png</p> 
<p>...</p>	<p>DisplaySkin10.png</p> 	<p>RectangleSkin.png</p> 	<p>thinRectangleSkin.png</p> 

Appendix D. Asteroids AGORAS Specification

This appendix contains the specification in full for the Asteroids AGORAS game.

Asteroids.eco file (primary specification file):

```

ECOSYSTEM NAME AsteroidsGame
STRUCTURE_TYPES
  STRUCTURE_TYPE NAME SpaceshipStructure PATH "Content/Simulator/asteroids/SpaceshipStructure.st"
END_STRUCTURE_TYPE
  STRUCTURE_TYPE NAME CannonBallStructure PATH "Content/Simulator/asteroids/CannonBallStructure.st"
END_STRUCTURE_TYPE
  STRUCTURE_TYPE NAME BigAsteroidStructure PATH "Content/Simulator/asteroids/BigAsteroidStructure.st"
END_STRUCTURE_TYPE
  STRUCTURE_TYPE NAME SmallAsteroidStructure PATH "Content/Simulator/asteroids/SmallAsteroidStructure.st"
END_STRUCTURE_TYPE
  STRUCTURE_TYPE NAME ButtonControlStructure PATH "Content/Simulator/asteroids/ButtonControlStructure.st"
END_STRUCTURE_TYPE
END_STRUCTURE_TYPES

ENTITY_TYPES
  ENTITY_TYPE SpaceshipType
    PHYSICAL_RESPONSE true
    STRUCTURE_TYPE SpaceshipStructure
    PROTO_COSTUMES
      PROTO_COSTUME SpaceshipCostume PATH "Content/Simulator/asteroids/SpaceshipCostume.pc"
    END_PROTO_COSTUME
  END_ENTITY_TYPE
  ENTITY_TYPE CannonBallType
    PHYSICAL_RESPONSE true
    STRUCTURE_TYPE CannonBallStructure
    PROTO_COSTUMES
      PROTO_COSTUME CannonBallCostume PATH "Content/Simulator/asteroids/CannonBallCostume.pc"
    END_PROTO_COSTUME
  END_ENTITY_TYPE
  ENTITY_TYPE BigAsteroidType
    PHYSICAL_RESPONSE true
    REBOUND true
    STRUCTURE_TYPE BigAsteroidStructure
    PROTO_COSTUMES
      PROTO_COSTUME BigAsteroidCostume PATH "Content/Simulator/asteroids/BigAsteroidCostume.pc"
    END_PROTO_COSTUME
  END_ENTITY_TYPE
  ENTITY_TYPE SmallAsteroidType
    PHYSICAL_RESPONSE true
    REBOUND true
    STRUCTURE_TYPE SmallAsteroidStructure
    PROTO_COSTUMES
      PROTO_COSTUME SmallAsteroidCostume PATH "Content/Simulator/asteroids/SmallAsteroidCostume.pc"
    END_PROTO_COSTUME
  END_ENTITY_TYPE

```



```

PROPERTY NAME Position TYPE Vector2 VALUE -70,0 -350,0 END_PROPERTY
END_ENTITY
ENTITY ButtonControlType::RightControl
PROPERTY NAME Position TYPE Vector2 VALUE -20,0 -350,0 END_PROPERTY
END_ENTITY
ENTITY ButtonControlType::ShootControl
PROPERTY NAME Position TYPE Vector2 VALUE 50,0 -350,0 END_PROPERTY
END_ENTITY
ENTITY BigAsteroidType::bast1
PROPERTY NAME Position TYPE Vector2 VALUE -400,0 0,0 END_PROPERTY
PROPERTY NAME InitialSpeed TYPE Vector2 VALUE 4,0 0,0 END_PROPERTY
END_ENTITY
ENTITY BigAsteroidType::bast2
PROPERTY NAME Position TYPE Vector2 VALUE 400,0 00,0 END_PROPERTY
PROPERTY NAME InitialSpeed TYPE Vector2 VALUE -4,0 0,0 END_PROPERTY
END_ENTITY
ENTITY SmallAsteroidType::sast1
PROPERTY NAME Position TYPE Vector2 VALUE -20,0 250,0 END_PROPERTY
PROPERTY NAME InitialSpeed TYPE Vector2 VALUE 0,2 -2,50 END_PROPERTY
END_ENTITY
ENTITY SmallAsteroidType::sast2
PROPERTY NAME Position TYPE Vector2 VALUE 0,0 -200,0 END_PROPERTY
PROPERTY NAME InitialSpeed TYPE Vector2 VALUE 0,0 2,0 END_PROPERTY
END_ENTITY
END_PARTICIPATIONS

```

```

////////////////////////////////////
/// RULES:
////////////////////////////////////

```

```

REACTIVE_RULE NAME GameOverBigAsteroid
PRIORITY 10
EVENT Collision
SOURCE ENTITY_TYPE BigAsteroidType
TARGET ENTITY AsteroidGameStageStageType::AsteroidGameStage
OPERATION ACTION PauseSimulation
PRECONDITION_DATA_PROCESS
DATA_PROCESS
NODES
CONSTANT_PROVIDER NAME ConstantSpaceship TYPE SpaceshipType VALUE
SpaceshipType::PlayerSpaceship
POSITION 0,0 0,0
END_CONSTANT_PROVIDER
DATA_PROCESSOR_INSTANCE NAME ==1 TYPE EQ
POSITION 0,0 0,0
END_DATA_PROCESSOR_INSTANCE
END_NODES
DATA_FLOWS
DATA_FLOW FROM ConstantSpaceship::out TO ==1::in1 END_DATA_FLOW
DATA_FLOW FROM EVENT.SlaveEntity::out TO ==1::in2 END_DATA_FLOW
DATA_FLOW FROM ==1::out TO condition_result::in1 END_DATA_FLOW
END_DATA_FLOWS
END_DATA_PROCESS
END_PRECONDITION_DATA_PROCESS
END_REACTIVE_RULE

```

```

REACTIVE_RULE NAME GameOverSmallAsteroid
  PRIORITY 10
  EVENT Collision
  SOURCE ENTITY_TYPE SmallAsteroidType
  TARGET ENTITY AsteroidGameStageStageType::AsteroidGameStage
  OPERATION ACTION PauseSimulation
  PRECONDITION_DATA_PROCESS
    DATA_PROCESS
      NODES
        CONSTANT_PROVIDER NAME ConstantSpaceship TYPE SpaceshipType VALUE
SpaceshipType::PlayerSpaceship
        POSITION 0,0 0,0
      END_CONSTANT_PROVIDER
      DATA_PROCESSOR_INSTANCE NAME ==1 TYPE EQ
        POSITION 0,0 0,0
      END_DATA_PROCESSOR_INSTANCE
    END_NODES
    DATA_FLOWS
      DATA_FLOW FROM ConstantSpaceship::out TO ==1::in1 END_DATA_FLOW
      DATA_FLOW FROM EVENT.SlaveEntity::out TO ==1::in2 END_DATA_FLOW
      DATA_FLOW FROM ==1::out TO condition_result::in1 END_DATA_FLOW
    END_DATA_FLOWS
  END_DATA_PROCESS
  END_PRECONDITION_DATA_PROCESS
END_REACTIVE_RULE

REACTIVE_RULE NAME RotateLeft
  PRIORITY 2
  EVENT SurfaceTouched
  SOURCE ENTITY ButtonControlType::LeftControl
  TARGET ENTITY SpaceshipType::PlayerSpaceship
  OPERATION PROPERTY Rotation
  OPERATION_DATA_PROCESS
    DATA_PROCESS
      NODES
        CONSTANT_PROVIDER NAME rotIncLeft TYPE Float VALUE 0,52 // Ship rotation increment:
30° = 0,52 rad
        POSITION 0,00 0,00
      END_CONSTANT_PROVIDER
      DATA_PROCESSOR_INSTANCE NAME ADDF1 TYPE ADDF
        POSITION 0,00 0,00
      END_DATA_PROCESSOR_INSTANCE
    END_NODES
    DATA_FLOWS
      DATA_FLOW FROM TARGET.Rotation::out TO ADDF1::in1 END_DATA_FLOW
      DATA_FLOW FROM rotIncLeft::out TO ADDF1::in2 END_DATA_FLOW
      DATA_FLOW FROM ADDF1::out TO TARGET.Rotation::in1 END_DATA_FLOW
    END_DATA_FLOWS
  END_DATA_PROCESS
  END_OPERATION_DATA_PROCESS
END_REACTIVE_RULE

REACTIVE_RULE NAME RotateRight
  PRIORITY 2
  EVENT SurfaceTouched
  SOURCE ENTITY ButtonControlType::RightControl
  TARGET ENTITY SpaceshipType::PlayerSpaceship

```

```

OPERATION PROPERTY Rotation
OPERATION_DATA_PROCESS
  DATA_PROCESS
    NODES
      CONSTANT_PROVIDER NAME rotInclLeft TYPE Float VALUE -0,52 // Ship rotation increment:
30° = 0,52 rad
      POSITION 0,00 0,00
      END_CONSTANT_PROVIDER
      DATA_PROCESSOR_INSTANCE NAME ADDF1 TYPE ADDF
      POSITION 0,00 0,00
      END_DATA_PROCESSOR_INSTANCE
    END_NODES
  DATA_FLOWS
    DATA_FLOW FROM TARGET.Rotation::out TO ADDF1::in1 END_DATA_FLOW
    DATA_FLOW FROM rotInclLeft::out TO ADDF1::in2 END_DATA_FLOW
    DATA_FLOW FROM ADDF1::out TO TARGET.Rotation::in1 END_DATA_FLOW
  END_DATA_FLOWS
END_DATA_PROCESS
END_OPERATION_DATA_PROCESS
END_REACTIVE_RULE

REACTIVE_RULE NAME Shoot
EVENT SurfaceTouched
SOURCE ENTITY ButtonControlType::ShootControl
TARGET ENTITY AsteroidGameStageStageType::AsteroidGameStage
OPERATION ACTION CreateAndIncludeEntity
OPERATION_DATA_PROCESS
  DATA_PROCESS PARAM EntityTypeName
  NODES
    CONSTANT_PROVIDER NAME constant1 TYPE String VALUE "CannonBallType"
    POSITION 922,00 150,00
    END_CONSTANT_PROVIDER
  END_NODES
  DATA_FLOWS
    DATA_FLOW FROM constant1::out TO CreateAndIncludeEntity.EntityTypeName::in1 END_DATA_FLOW
  END_DATA_FLOWS
END_DATA_PROCESS
  DATA_PROCESS PARAM EntityName
  NODES
    CONSTANT_PROVIDER NAME constant1 TYPE String VALUE "CannonBall"
    POSITION 1036,00 182,00
    END_CONSTANT_PROVIDER
  END_NODES
  DATA_FLOWS
    DATA_FLOW FROM constant1::out TO CreateAndIncludeEntity.EntityName::in1 END_DATA_FLOW
  END_DATA_FLOWS
END_DATA_PROCESS
  DATA_PROCESS PARAM CostumeName
  NODES
    CONSTANT_PROVIDER NAME constant1 TYPE String VALUE "CannonBallCostume"
    POSITION 1012,00 113,00
    END_CONSTANT_PROVIDER
  END_NODES
  DATA_FLOWS
    DATA_FLOW FROM constant1::out TO CreateAndIncludeEntity.CostumeName::in1 END_DATA_FLOW
  END_DATA_FLOWS
END_DATA_PROCESS
  DATA_PROCESS PARAM Position

```



```

    NODES
    BOUNDED_PROPERTY ENTITY PlayerSpaceship TYPE SpaceshipType PROPERTY Rotation
        POSITION 0,0 0,0
    END_BOUNDED_PROPERTY
    END_BOUNDED_PROPERTY
    CONSTANT_PROVIDER NAME constant2 TYPE Vector2 VALUE 0,00 0,00
        POSITION 650,00 277,00
    END_CONSTANT_PROVIDER
    CONSTANT_PROVIDER NAME constant1 TYPE Vector2 VALUE 0,00 60,00 // Bullet
distance from ship
        POSITION 440,00 121,00
    END_CONSTANT_PROVIDER
    DATA_PROCESSOR_INSTANCE NAME ROTATEABOUTORIGIN1 TYPE ROTATEABOUTORIGIN
        POSITION 0,00 0,00
    END_DATA_PROCESSOR_INSTANCE
    END_NODES
    DATA_FLOWS
    DATA_FLOW FROM constant1::out TO ROTATEABOUTORIGIN1::in1 END_DATA_FLOW
    DATA_FLOW FROM constant2::out TO ROTATEABOUTORIGIN1::in2 END_DATA_FLOW
    DATA_FLOW FROM PlayerSpaceship.Rotation::out TO ROTATEABOUTORIGIN1::in3 END_DATA_FLOW
    DATA_FLOW FROM ROTATEABOUTORIGIN1::out TO CreateAndIncludeEntity.Position::in1
END_DATA_FLOW
    END_DATA_FLOWS
    END_DATA_PROCESS
    DATA_PROCESS PARAM InitialSpeed
    NODES
    BOUNDED_PROPERTY ENTITY PlayerSpaceship TYPE SpaceshipType PROPERTY Rotation
        POSITION 0,0 0,0
    END_BOUNDED_PROPERTY
    END_BOUNDED_PROPERTY
    CONSTANT_PROVIDER NAME constant2 TYPE Vector2 VALUE 0,00 0,00
        POSITION 650,00 277,00
    END_CONSTANT_PROVIDER
    CONSTANT_PROVIDER NAME constant1 TYPE Vector2 VALUE 0,00 30,00 // Bullet
speed
        POSITION 440,00 121,00
    END_CONSTANT_PROVIDER
    DATA_PROCESSOR_INSTANCE NAME ROTATEABOUTORIGIN1 TYPE ROTATEABOUTORIGIN
        POSITION 0,00 0,00
    END_DATA_PROCESSOR_INSTANCE
    END_NODES
    DATA_FLOWS
    DATA_FLOW FROM constant1::out TO ROTATEABOUTORIGIN1::in1 END_DATA_FLOW
    DATA_FLOW FROM constant2::out TO ROTATEABOUTORIGIN1::in2 END_DATA_FLOW
    DATA_FLOW FROM PlayerSpaceship.Rotation::out TO ROTATEABOUTORIGIN1::in3 END_DATA_FLOW
    DATA_FLOW FROM ROTATEABOUTORIGIN1::out TO CreateAndIncludeEntity.InitialSpeed::in1
END_DATA_FLOW
    END_DATA_FLOWS
    END_DATA_PROCESS
    END_OPERATION_DATA_PROCESS
    END_REACTIVE_RULE

    REACTIVE_RULE NAME ExcludeSmallAsteroid
    PRIORITY 5
    EVENT Collision
    SOURCE ENTITY_TYPE SmallAsteroidType
    TARGET ENTITY AsteroidGameStageStageType::AsteroidGameStage
    OPERATION ACTION ExcludePerformer
    PRECONDITION_DATA_PROCESS
    NODE_CONDITIONS

```

```

END_NODE_CONDITIONS
DATA_PROCESS
  NODES
    CONSTANT_PROVIDER NAME constant1 TYPE String VALUE "CannonBallType"
    POSITION 415,00 506,00
  END_CONSTANT_PROVIDER
  DATA_PROCESSOR_INSTANCE NAME INSTANCEOF1 TYPE INSTANCEOF
    POSITION 0,00 0,00
  END_DATA_PROCESSOR_INSTANCE
END_NODES
DATA_FLOWS
  DATA_FLOW FROM EVENT.SlaveEntity::out TO INSTANCEOF1::in1 END_DATA_FLOW
  DATA_FLOW FROM constant1::out TO INSTANCEOF1::in2 END_DATA_FLOW
  DATA_FLOW FROM INSTANCEOF1::out TO condition_result::in1 END_DATA_FLOW
END_DATA_FLOWS
END_DATA_PROCESS
END_PRECONDITION_DATA_PROCESS
OPERATION_DATA_PROCESS
  DATA_PROCESS PARAM Entity
  NODES
  END_NODES
  DATA_FLOWS
    DATA_FLOW FROM SOURCE.Value::out TO ExcludePerformer.Entity::in1 END_DATA_FLOW
  END_DATA_FLOWS
END_OPERATION_DATA_PROCESS
END_REACTIVE_RULE

REACTIVE_RULE NAME ExcludeBigAsteroid
  PRIORITY 5
  EVENT Collision
  SOURCE ENTITY_TYPE BigAsteroidType
  TARGET ENTITY AsteroidGameStageType::AsteroidGameStage
  OPERATION ACTION ExcludePerformer
  PRECONDITION_DATA_PROCESS
    NODE_CONDITIONS
    END_NODE_CONDITIONS
    DATA_PROCESS
      NODES
        CONSTANT_PROVIDER NAME constant1 TYPE String VALUE "CannonBallType"
        POSITION 415,00 506,00
      END_CONSTANT_PROVIDER
      DATA_PROCESSOR_INSTANCE NAME INSTANCEOF1 TYPE INSTANCEOF
        POSITION 0,00 0,00
      END_DATA_PROCESSOR_INSTANCE
    END_NODES
    DATA_FLOWS
      DATA_FLOW FROM EVENT.SlaveEntity::out TO INSTANCEOF1::in1 END_DATA_FLOW
      DATA_FLOW FROM constant1::out TO INSTANCEOF1::in2 END_DATA_FLOW
      DATA_FLOW FROM INSTANCEOF1::out TO condition_result::in1 END_DATA_FLOW
    END_DATA_FLOWS
  END_DATA_PROCESS
END_PRECONDITION_DATA_PROCESS
OPERATION_DATA_PROCESS
  DATA_PROCESS PARAM Entity
  NODES
  END_NODES
  DATA_FLOWS

```

```

        DATA_FLOW FROM SOURCE.Value::out TO ExcludePerformer.Entity::in1 END_DATA_FLOW
    END_DATA_FLOWS
END_DATA_PROCESS
END_OPERATION_DATA_PROCESS
END_REACTIVE_RULE

REACTIVE_RULE NAME ExcludeCannonBallForSmallAsteroidCollision
PRIORITY 5
EVENT Collision
SOURCE ENTITY_TYPE SmallAsteroidType
TARGET ENTITY AsteroidGameStageStageType::AsteroidGameStage
OPERATION ACTION ExcludePerformer
PRECONDITION_DATA_PROCESS
    NODE_CONDITIONS
    END_NODE_CONDITIONS
    DATA_PROCESS
    NODES
        CONSTANT_PROVIDER NAME constant1 TYPE String VALUE "CannonBallType"
        POSITION 415,00 506,00
    END_CONSTANT_PROVIDER
    DATA_PROCESSOR_INSTANCE NAME INSTANCEOF1 TYPE INSTANCEOF
    POSITION 0,00 0,00
    END_DATA_PROCESSOR_INSTANCE
    END_NODES
    DATA_FLOWS
        DATA_FLOW FROM EVENT.SlaveEntity::out TO INSTANCEOF1::in1 END_DATA_FLOW
        DATA_FLOW FROM constant1::out TO INSTANCEOF1::in2 END_DATA_FLOW
        DATA_FLOW FROM INSTANCEOF1::out TO condition_result::in1 END_DATA_FLOW
    END_DATA_FLOWS
    END_DATA_PROCESS
END_PRECONDITION_DATA_PROCESS
OPERATION_DATA_PROCESS
    DATA_PROCESS PARAM Entity
    NODES
    END_NODES
    DATA_FLOWS
        DATA_FLOW FROM EVENT.SlaveEntity::out TO ExcludePerformer.Entity::in1 END_DATA_FLOW
    END_DATA_FLOWS
    END_DATA_PROCESS
END_OPERATION_DATA_PROCESS
END_REACTIVE_RULE

REACTIVE_RULE NAME ExcludeCannonBallForBigAsteroidCollision
PRIORITY 5
EVENT Collision
SOURCE ENTITY_TYPE BigAsteroidType
TARGET ENTITY AsteroidGameStageStageType::AsteroidGameStage
OPERATION ACTION ExcludePerformer
PRECONDITION_DATA_PROCESS
    NODE_CONDITIONS
    END_NODE_CONDITIONS
    DATA_PROCESS
    NODES
        CONSTANT_PROVIDER NAME constant1 TYPE String VALUE "CannonBallType"
        POSITION 415,00 506,00
    END_CONSTANT_PROVIDER
    DATA_PROCESSOR_INSTANCE NAME INSTANCEOF1 TYPE INSTANCEOF
    POSITION 0,00 0,00

```

```

    END_DATA_PROCESSOR_INSTANCE
  END_NODES
  DATA_FLOWS
    DATA_FLOW FROM EVENT.SlaveEntity::out TO INSTANCEOF1::in1 END_DATA_FLOW
    DATA_FLOW FROM constant1::out TO INSTANCEOF1::in2 END_DATA_FLOW
    DATA_FLOW FROM INSTANCEOF1::out TO condition_result::in1 END_DATA_FLOW
  END_DATA_FLOWS
  END_DATA_PROCESS
  END_PRECONDITION_DATA_PROCESS
  OPERATION_DATA_PROCESS
  DATA_PROCESS PARAM Entity
  NODES
  END_NODES
  DATA_FLOWS
    DATA_FLOW FROM EVENT.SlaveEntity::out TO ExcludePerformer.Entity::in1 END_DATA_FLOW
  END_DATA_FLOWS
  END_DATA_PROCESS
  END_OPERATION_DATA_PROCESS
  END_REACTIVE_RULE

  REACTIVE_RULE NAME DivideBigAsteroid_Left
  PRIORITY 4
  EVENT Collision
  SOURCE ENTITY_TYPE BigAsteroidType
  TARGET ENTITY AsteroidGameStageStageType::AsteroidGameStage
  OPERATION ACTION CreateAndIncludeEntity
  PRECONDITION_DATA_PROCESS
  NODE_CONDITIONS
  END_NODE_CONDITIONS
  DATA_PROCESS
  NODES
    CONSTANT_PROVIDER NAME constant1 TYPE String VALUE "CannonBallType"
    POSITION 415,00 506,00
  END_CONSTANT_PROVIDER
  DATA_PROCESSOR_INSTANCE NAME INSTANCEOF1 TYPE INSTANCEOF
  POSITION 0,00 0,00
  END_DATA_PROCESSOR_INSTANCE
  END_NODES
  DATA_FLOWS
    DATA_FLOW FROM EVENT.SlaveEntity::out TO INSTANCEOF1::in1 END_DATA_FLOW
    DATA_FLOW FROM constant1::out TO INSTANCEOF1::in2 END_DATA_FLOW
    DATA_FLOW FROM INSTANCEOF1::out TO condition_result::in1 END_DATA_FLOW
  END_DATA_FLOWS
  END_DATA_PROCESS
  END_PRECONDITION_DATA_PROCESS
  OPERATION_DATA_PROCESS
  DATA_PROCESS PARAM EntityTypeName
  NODES
    CONSTANT_PROVIDER NAME constant1 TYPE String VALUE "SmallAsteroidType"
    POSITION 1166,00 108,00
  END_CONSTANT_PROVIDER
  END_NODES
  DATA_FLOWS
    DATA_FLOW FROM constant1::out TO CreateAndIncludeEntity.EntityTypeName::in1 END_DATA_FLOW
  END_DATA_FLOWS
  END_DATA_PROCESS
  DATA_PROCESS PARAM EntityName
  NODES

```

```

CONSTANT_PROVIDER NAME constant1 TYPE String VALUE "LeftSmallAsteroid"
  POSITION 1211,00 103,00
END_CONSTANT_PROVIDER
END_NODES
DATA_FLOWS
  DATA_FLOW FROM constant1::out TO CreateAndIncludeEntity.EntityName::in1 END_DATA_FLOW
END_DATA_FLOWS
END_DATA_PROCESS
DATA_PROCESS PARAM CostumeName
  NODES
    CONSTANT_PROVIDER NAME constant1 TYPE String VALUE "SmallAsteroidCostume"
      POSITION 1239,00 119,00
    END_CONSTANT_PROVIDER
  END_NODES
  DATA_FLOWS
    DATA_FLOW FROM constant1::out TO CreateAndIncludeEntity.CostumeName::in1 END_DATA_FLOW
  END_DATA_FLOWS
END_DATA_PROCESS
DATA_PROCESS PARAM Position
  NODES
    CONSTANT_PROVIDER NAME constant1 TYPE Vector2 VALUE -35,00 0,00 // Distance from the
big asteroid
    POSITION 474,00 215,00
    END_CONSTANT_PROVIDER
    DATA_PROCESSOR_INSTANCE NAME ADDV1 TYPE ADDV
      POSITION 0,00 0,00
    END_DATA_PROCESSOR_INSTANCE
  END_NODES
  DATA_FLOWS
    DATA_FLOW FROM SOURCE.Position::out TO ADDV1::in1 END_DATA_FLOW
    DATA_FLOW FROM constant1::out TO ADDV1::in2 END_DATA_FLOW
    DATA_FLOW FROM ADDV1::out TO CreateAndIncludeEntity.Position::in1 END_DATA_FLOW
  END_DATA_FLOWS
END_DATA_PROCESS
DATA_PROCESS PARAM InitialSpeed
  NODES
    CONSTANT_PROVIDER NAME constant1 TYPE Float VALUE -0,78 // 45° deviation
      POSITION 357,00 222,00
    END_CONSTANT_PROVIDER
    DATA_PROCESSOR_INSTANCE NAME ROTATEABOUTORIGIN1 TYPE ROTATEABOUTORIGIN
      POSITION 0,00 0,00
    END_DATA_PROCESSOR_INSTANCE
  END_NODES
  DATA_FLOWS
    DATA_FLOW FROM SOURCE.InitialSpeed::out TO ROTATEABOUTORIGIN1::in1 END_DATA_FLOW
    DATA_FLOW FROM SOURCE.Position::out TO ROTATEABOUTORIGIN1::in2 END_DATA_FLOW
    DATA_FLOW FROM constant1::out TO ROTATEABOUTORIGIN1::in3 END_DATA_FLOW
    DATA_FLOW FROM ROTATEABOUTORIGIN1::out TO CreateAndIncludeEntity.InitialSpeed::in1
END_DATA_FLOW
  END_DATA_FLOWS
END_DATA_PROCESS
END_OPERATION_DATA_PROCESS
END_REACTIVE_RULE

REACTIVE_RULE NAME DivideBigAsteroid_Right
  PRIORITY 4
  EVENT Collision
  SOURCE ENTITY_TYPE BigAsteroidType

```

```

TARGET ENTITY AsteroidGameStageStageType::AsteroidGameStage
OPERATION ACTION CreateAndIncludeEntity
PRECONDITION_DATA_PROCESS
  NODE_CONDITIONS
  END_NODE_CONDITIONS
  DATA_PROCESS
  NODES
    CONSTANT_PROVIDER NAME constant1 TYPE String VALUE "CannonBallType"
    POSITION 415,00 506,00
  END_CONSTANT_PROVIDER
  DATA_PROCESSOR_INSTANCE NAME INSTANCEOF1 TYPE INSTANCEOF
    POSITION 0,00 0,00
  END_DATA_PROCESSOR_INSTANCE
  END_NODES
  DATA_FLOWS
    DATA_FLOW FROM EVENT.SlaveEntity::out TO INSTANCEOF1::in1 END_DATA_FLOW
    DATA_FLOW FROM constant1::out TO INSTANCEOF1::in2 END_DATA_FLOW
    DATA_FLOW FROM INSTANCEOF1::out TO condition_result::in1 END_DATA_FLOW
  END_DATA_FLOWS
  END_DATA_PROCESS
END_PRECONDITION_DATA_PROCESS
OPERATION_DATA_PROCESS
  DATA_PROCESS PARAM EntityTypeName
  NODES
    CONSTANT_PROVIDER NAME constant1 TYPE String VALUE "SmallAsteroidType"
    POSITION 1166,00 108,00
  END_CONSTANT_PROVIDER
  END_NODES
  DATA_FLOWS
    DATA_FLOW FROM constant1::out TO CreateAndIncludeEntity.EntityTypeName::in1 END_DATA_FLOW
  END_DATA_FLOWS
  END_DATA_PROCESS
  DATA_PROCESS PARAM EntityName
  NODES
    CONSTANT_PROVIDER NAME constant1 TYPE String VALUE "LeftSmallAsteroid"
    POSITION 1211,00 103,00
  END_CONSTANT_PROVIDER
  END_NODES
  DATA_FLOWS
    DATA_FLOW FROM constant1::out TO CreateAndIncludeEntity.EntityName::in1 END_DATA_FLOW
  END_DATA_FLOWS
  END_DATA_PROCESS
  DATA_PROCESS PARAM CostumeName
  NODES
    CONSTANT_PROVIDER NAME constant1 TYPE String VALUE "SmallAsteroidCostume"
    POSITION 1239,00 119,00
  END_CONSTANT_PROVIDER
  END_NODES
  DATA_FLOWS
    DATA_FLOW FROM constant1::out TO CreateAndIncludeEntity.CostumeName::in1 END_DATA_FLOW
  END_DATA_FLOWS
  END_DATA_PROCESS
  DATA_PROCESS PARAM Position
  NODES
    CONSTANT_PROVIDER NAME constant1 TYPE Vector2 VALUE 35,00 0,00 // Distance from the
big asteroid
    POSITION 474,00 215,00
  END_CONSTANT_PROVIDER

```

```

    DATA_PROCESSOR_INSTANCE NAME ADDV1 TYPE ADDV
      POSITION 0,00 0,00
    END_DATA_PROCESSOR_INSTANCE
  END_NODES
  DATA_FLOWS
    DATA_FLOW FROM SOURCE.Position::out TO ADDV1::in1 END_DATA_FLOW
    DATA_FLOW FROM constant1::out TO ADDV1::in2 END_DATA_FLOW
    DATA_FLOW FROM ADDV1::out TO CreateAndIncludeEntity.Position::in1 END_DATA_FLOW
  END_DATA_FLOWS
  END_DATA_PROCESS
  DATA_PROCESS PARAM InitialSpeed
  NODES
    CONSTANT_PROVIDER NAME constant1 TYPE Float VALUE 0,78 // 45° deviation
      POSITION 357,00 222,00
    END_CONSTANT_PROVIDER
    DATA_PROCESSOR_INSTANCE NAME ROTATEABOUTORIGIN1 TYPE ROTATEABOUTORIGIN
      POSITION 0,00 0,00
    END_DATA_PROCESSOR_INSTANCE
  END_NODES
  DATA_FLOWS
    DATA_FLOW FROM SOURCE.InitialSpeed::out TO ROTATEABOUTORIGIN1::in1 END_DATA_FLOW
    DATA_FLOW FROM SOURCE.Position::out TO ROTATEABOUTORIGIN1::in2 END_DATA_FLOW
    DATA_FLOW FROM constant1::out TO ROTATEABOUTORIGIN1::in3 END_DATA_FLOW
    DATA_FLOW FROM ROTATEABOUTORIGIN1::out TO CreateAndIncludeEntity.InitialSpeed::in1
  END_DATA_FLOW
  END_DATA_FLOWS
  END_DATA_PROCESS
  END_OPERATION_DATA_PROCESS
  END_REACTIVE_RULE

  REACTIVE_RULE NAME DeleteExcludedEntity
  PRIORITY 2
  EVENT PerformerExcluded
  SOURCE ENTITY AsteroidGameStageStageType::AsteroidGameStage
  TARGET ENTITY AsteroidGameStageStageType::AsteroidGameStage
  OPERATION ACTION DestroyEntity
  OPERATION_DATA_PROCESS
    DATA_PROCESS PARAM Entity
    NODES
      END_NODES
    DATA_FLOWS
      DATA_FLOW FROM EVENT.Entity::out TO DestroyEntity.Entity::in1 END_DATA_FLOW
    END_DATA_FLOWS
  END_DATA_PROCESS
  END_OPERATION_DATA_PROCESS
  END_REACTIVE_RULE
  END_STAGE
  END_STAGES

  CURRENT_STAGE AsteroidGameStage

  END_ECOSYSTEM

```

Structure-Type specification files

ButtonControlStructure.st file:

```

STRUCTURE TYPE
ButtonControlStructure

COMPONENTS 1
ELLIPSE
# Id
0
# layer
0
# static {-1 = static; 0..n= fricCoef }
-1
# position
0 0
# radius
20 20
# rotation
0

JOINTS 0
    
```

SpaceshipStructure.st file:

```

STRUCTURE TYPE
SpaceshipStructure

COMPONENTS 1
RECTANGLE
# Id
0
# layer
0
# static {-1 = static; 0..n= fricCoef }
-1
# position
0 0
# dimensions x and y
50 100
# rotation
0

JOINTS 0
    
```

CannonballStructure.st

```

STRUCTURE TYPE
CannonBallStructure

COMPONENTS 1
ELLIPSE
# Id
0
# layer
0
# static {-1 = static; 0..n= fricCoef }
0
    
```



```
# position
0 0
# radius
10 10
# rotation
0
JOINTS 0
```

SmallAsteroidStructure.st

```
STRUCTURE TYPE
SmallAsteroidStructure

COMPONENTS 1
ELLIPSE
# ld
0
# layer
0
# static {-1 = static; 0..n= fricCoef }
0
# position
0 0
# radius
30 30
# rotation
0
JOINTS 0
```

BigAsteroidStructure.st

```
STRUCTURE TYPE
BigAsteroidStructure

COMPONENTS 1
ELLIPSE
# ld
0
# layer
0
# static {-1 = static; 0..n= fricCoef }
0
# position
0 0
# radius
60 60
# rotation
0
JOINTS 0
```

ProtoCostume files

SpaceshipCostume.pc

```

PROTOCOLCOSTUME
SpaceshipCostume

STRUCTURE SpaceshipStructure

SKINS 1
Content\Simulator\asteroids\SpaceshipSkin.png

# id
0
# offset
0 0
# scale
1 1
# rotation
0
    
```

LeftControlCostume.pc

```

PROTOCOLCOSTUME
LeftControlCostume

STRUCTURE ButtonControlStructure

SKINS 1
Content\Simulator\asteroids\LeftControlSkin.png

# id
0
# offset
0 0
# scale
1 1
# rotation
0
    
```

CannonballCostume.pc

```

PROTOCOLCOSTUME
CannonBallCostume

STRUCTURE CannonBallStructure

SKINS 1
Content\Simulator\asteroids\CannonBallSkin.png

# id
0
# offset
0 0
    
```

```
# scale
1 1
# rotation
0
```

SmallAsteroidCostume.pc

```
PROTOCOLCOSTUME
SmallAsteroidCostume

STRUCTURE SmallAsteroidStructure

SKINS 1
Content\Simulator\asteroids\SmallAsteroidSkin.png

# id
0
# offset
0 0
# scale
1 1
# rotation
0
```

BigAsteroidCostume.pc

```
PROTOCOLCOSTUME
BigAsteroidCostume





STRUCTURE BigAsteroidStructure

SKINS 1
Content\Simulator\asteroids\BigAsteroidSkin.png

# id
0
# offset
0 0
# scale
1 1
# rotation
0
```

Image files for Skins

SpaceshipS-kin.png	LeftControlS-kin.png	RightControlS-kin.png	ShootControlS-kin.png
			

			
<p>Cannon-BallSkin.png</p> 	<p>SmallAsteroidSkin.png</p> 	<p>BigAsteroidSkin.png</p> 	

References

- (*Abt, 1970*) Abt, C. Serious Games. New York: Viking Press, 1970.
- (*Accot, 1997*) Accot, J. and Zhai, S. Beyond Fitts' law: models for trajectory-based HCI tasks. In Proc. CHI97, ACM, pp. 295-302, 1997
- (*Aleinikov, 2000*) Aleinikov A., Kackmeister S. and Koenig R. Creating creativity: 101 definitions. Midland, MI: Alden B. Dow Creativity Center, Northwoods University, 2000.
- (*Amabile, 1978*) Amabile, T. M. Effects of Evaluation Expectation on Artistic Creativity. Paper presented at the Annual Meeting of the Eastern Psychological Association, Washington, D.C., March 29-April 1, 20 pages, 1978.
- (*Amabile, 1983*) Amabile, T. M. The social psychology of creativity. New York: Springer-Verlag, 1983.
- (*Amabile, 1999*) Amabile, T. M. Consensual assessment. In M. A. Runco & S. Pritzker (Eds.), Encyclopedia of creativity. San Diego, CA: Academic Press, pp. 346-349, 1999.
- (*Anderson, 2000*) Anderson, L. W., Krathwohl, D. R., Airasian, P. W., Cruikshank, K. A., Mayer, R. E., Pintrich, P. R., Raths J. and Wittrock, M. C. A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives, Allyn & Bacon, 2000.
- (*Aragon, 2009*) Aragon, C. R., Poon, S. S., Monroy-Hernández, A., and Aragon, D. A tale of two online communities: fostering collaboration and creativity in scientists and children. In Proc. C&C 2009. ACM Press, pp. 9-18, 2009.
- (*Atkinson, 2003*) Atkinson, C. and Kühne, T. Model-Driven Development: A Metamodeling Foundation. IEEE Softw.

20, 5 (September 2003), 36-41, 2003

- (Baer, 1998) Baer, J. Gender differences in the effects of extrinsic motivation on creativity. *The Journal of Creative Behavior*, 32(1), 18-37, 1998.
- (Bailly, 2007) Bailly, G., Lecolinet, E., and Nigay, L. Wave menus: improving the novice mode of hierarchical marking menus. In *Proc. INTERACT 2007. Lecture Notes In Computer Science*. Springer-Verlag, pp. 475-488, 2007.
- (Baskerville, 1999) Baskerville, R., Pries-Heje, J. Grounded action research: a method for understanding IT in practice. *Accounting Management and Information Technologies*, 9(1), 1-23, 1999.
- (Begel, 1996) Begel, A. LogoBlocks: A Graphical Programming Language for Interacting with the World. MIT, 1996.
- (Ben-Joseph, 2001) Ben-Joseph, E., Ishii, H., Underkoffler, J., Piper, B., and Yeung, L. Urban Simulation and the Luminous Planning Table: Bridging the Gap between the Digital and the Tangible, *Journal of Planning Education and Research*, 21(2), 196-203, 2001.
- (Brown, 1989) Brown, J. S., Collins, A., and Duguid, P. Situated cognition and the culture of learning. *Educational Researcher* 18(1), 32-42, 1989.
- (Brown, 2002) Brown, B., Larson, R.W., and Saraswathi, T.S. *The worlds' youth: adolescence in eight regions of the world*. Cambridge University Press, 2002.
- (Bruner, 1960) Bruner, J. *The Process of Education*. Cambridge, MA: Harvard University Press, 1960.
- (Buisine, 2007) Buisine, S., Besacier, G., Najm, M., Aoussat, A., and Vernier, F. Computer-supported creativity: Evaluation of a tabletop mind-map application. In *Proc. EPCE'07*. Springer-Verlag, pp. 22-31, 2007.
- (Buxton, 2012) Buxton, B. Multi-Touch Systems that I Have Known and Loved.

<http://www.billbuxton.com/multitouchOverview.html>. Last access: 02/02/2012.

- (Chen, 2009) Chen, V.H.H., Lin., W., Haller, M., Leitner, J., and Duh, H.B.L. Communicative behaviors and flow experience in tabletop gaming. In Proc. ACE'09, ACM Press, pp. 281-286, 2009.
- (Cockburn, 1998) Cockburn, A. and Bryant, A. Cleogo: Collaborative and Multi-Metaphor Programming for Kids. APOCHI '98 Proceedings of the Third Asian Pacific Computer and Human Interaction, pp.189-194, 1998.
- (Cropley, 2001) Cropley A.J. Creativity in Education and Learning: A Guide for Teachers and Educators. Kogan Page, London, 2001.
- (Csikszentmihalyi, 1988) Csikszentmihalyi M. and Csikszentmihalyi I., Optimal Experience. Psychological Studies of Flow in Consciousness. Cambridge University Press, 1988.
- (Deci, 1985) Deci, E.L. and Ryan R. M. Intrinsic Motivation and Self-Determination in Human Behaviour. Plenum Press, 1985.
- (De Freitas, 2007) De Freitas, S. Learning in Immersive Worlds: a review of game based learning. Joint Information Systems Committee (JISC), 2007.
- (Delle Fave, 2002) Delle Fave, A., Bassi, M., and Massimini, F. Quality of experience and daily social context of Italian adolescents. In A.L. Comunian, U.P. Gielen eds. It's all about relationships, pp. 159-172, 2002.
- (Dewey, 1963) Dewey J. Experience and Education. New York: Collier, 1963.
- (Dietz, 2001) Dietz, P. and Leigh, D. DiamondTouch: a multi-user touch technology. In Proc. of UIST'01, ACM, pp. 219-226, 2001.
- (Ellis, 2006) Ellis, H., Heppell, S., Kirriemuir, J., Krotoski, A., and McFarlane, A. Unlimited Learning: The role of computer and video games in the learning landscape. ELSPA: Entertainment and Leisure Software Pub-

- lishers Association, 2006.
- (*Engelbrecht, 2011*) Engelbrecht, A., Borges, M. and Vivacqua, A.S. Digital tabletops for situational awareness in emergency situations. In Proc. International Conference on Computer Supported Cooperative Work in Design, 2011, IEEE Computer Press, pp. 669-676, 2011.
- (*Facer, 2004*) Facer, K., Joiner, R., Stanton, D., Reid, J., Hull, R. and Kirk, D. Savannah: mobile gaming and learning, Journal of Computer Assisted Learning vol. 20, pp. 399-409, 2004.
- (*Farooq, 2007*) Farooq, U., Carroll, J.M., and Ganoë, C.H. Supporting creativity with awareness in distributed collaboration. In Proc. GROUP '07. ACM Press, pp. 31-40, 2007.
- (*Fernaëus, 2006*) Fernæus, Y. and Tholander, J. Finding design qualities in a tangible programming space. In Proc. CHI '06 SIGCHI conference on Human Factors in computing systems, ACM, pp. 447-456, 2006.
- (*Fernaëus, 2008*) Fernæus, Y., Tholander, J. and Jonsson, M. Beyond representations: towards an action-centric perspective on tangible interaction. International Journal of Arts and Technology, 1 (3-4), 249-267, 2008.
- (*Fishkin, 2004*) Fishkin, K.. A Taxonomy for Analysis of Tangible Interfaces. In Proc. Personal and Ubiquitous Computing, pp. 347-358, 2004
- (*Fitzmaurice, 1995*) Fitzmaurice, G.W., Ishii, H. and Buxton, W. Bricks: Laying the Foundations for Graspable User Interfaces. In Proc. CHI'95, ACM Press, pp. 442-449, 1995.
- (*Fitzmaurice, 1997*) Fitzmaurice, G.W. and Buxton, W. An Empirical Evaluation of Graspable User Interfaces: towards specialized, space-multiplexed input. In Proc. CHI '97, ACM, pp. 43-50, 1997.
- (*Forster, 2009*) Forster, F. Improving creative thinking abilities using a generic collaborative creativity support system.

- In Proc. M-ICTE'09, pp. 539-543, 2009
- (*Friess, 2010*) Friess, M.R, Kleinhans, M., Forster, F., Echtler, F., and Groh, G. A Tabletop Interface for Generic Creativity Techniques. In Proc. MCCSIS '10. IADIS, pp. 203-210, 2010.
- (*Gallardo, 2008*) Gallardo, D., Julia, C.F., and Jordà, S. TurTan: A tangible programming language for creative exploration. In Proc. Tabletops'08. IEEE CS, pp. 89-92, 2008.
- (*Gallardo, 2010*) Gallardo, D. and Jordà, S. Tangible jukebox: back to palpable music. In Proc. TEI '10. ACM, pp. 199-202, 2010
- (*Gardner, 1993*) Gardner, H. Creating minds. New York: Basic Books, 1993.
- (*Gaver, 2004*) Gaver, W.W., Bowers, J., Boucher, A., Gellerson, H., Pennington, S., Schmidt, A., Steed, A., Villars, N. and Walker, B. The drift table: designing for ludic engagement. In Proc. CHI '04 extended abstracts on Human factors in computing systems, ACM Press, pp.885-900, 2004.
- (*Gee, 2005*) Gee, J.P. Learning by Design: good video games as learning machines. E-Learning, 2(1), 2005.
- (*Geyer, 2010*) Geyer, F., Klinkhammer, D., Reiterer, H. Supporting creativity workshops with interactive tabletops and digital pen and paper. In Proc. ITS '10. ACM Press, pp. 261-262, 2010
- (*Gómez-Martín, 2007*) Gómez-Martín, P. P, Gómez-Martín, M. A, Palmier-Campos, P. and González-Calero, P. A. Using Metaphors in Game-Based Education. Technologies for E-Learning and Digital Entertainment. Second International Conference of E-Learning and Games (Edutainment'07), pp. 477-488, LNCS, Springer, 2007.
- (*Good, 2010*) Good, J., Howland, K. and Nicholson, K. Young People's Descriptions of Computational Rules in

- Role-Playing Games: An Empirical Study, In Proc. of VL/HCC 2010, pp. 67-74, 2010.
- (Gros, 2004) Gros, B. Pantallas, juegos y educación: la alfabetización digital en la escuela.(España): Desclée de Brouwer, 2004.ISBN 84-330-1923-6
- (Gros, 2007) Gros, B. The Design of Learning Environments Using Videogames in Formal Education. In Proc. of IEEE International Workshop on Digital Game and Intelligent Toy Enhanced Learning (DIGITEL '07). IEEE Computer Society, Washington, DC, USA, pp. 19-24, 2007.
- (Gros, 2008) Gros B. (Coord). Videojuegos y aprendizaje. Editorial Graó, 2008.
- (Guilford, 1970) Guilford, J. P. Traits of creativity. In P. E. Vernon (Ed.), Creativity. Baltimore: Penguin, pp. 167-188, 1970.
- (Han, 2005) Han, J. Y. Low-cost multi-touch sensing through frustrated total internal reflection. In UIST '05: 18th annual ACM symposium on User interface software and technology, pp 115-118, New York, NY, USA. ACM Press, 2005
- (Hancock, 2009) Hancock, M., Hilliges, O., Collins, C., Baur, D. and Carpendale, S. Exploring tangible and direct touch interfaces for manipulating 2D and 3D information on a digital table. In Proc. ITS '09. ACM, pp. 77-84, 2009.
- (Hesselmann, 2009) Hesselmann, T., Flöring, S. and Schmidt, M. Stacked Half-Pie menus: navigating nested menus on interactive tabletops. In Proc. ITS'09. ACM Press, pp. 173-180, 2009.
- (Hevner, 2004) Hevner, A.R., March, S.T., Park, J, and Ram, S. Design science in information system research. MIS Quarterly, 28(1):75-105, March 2004.
- (Hilgard, 1973) Hilgard, E.R. and Bower, G.H. Teorías del aprendizaje. México, Trillas, 1973.

- (*Hilliges, 2007*) Hilliges, O., Baur, D. and Butz, A. Photohelix: Browsing, Sorting and Sharing Digital Photo Collections. In Proc. Horizontal Interactive Human-Computer Systems (TABLETOP'07), pp. 87-94, 2007.
- (*Hopkins, 1987*) Hopkins, D. Directional Selection is Easy as Pie Menus!, In: The Usenix Association Newsletter, 12(5), 1987.
- (*Horn, 2007*) Horn, M.S. and Jacob, R.J.K. Designing Tangible Programming Languages for Classroom Use. In Proc. TEI '07, pp. 159-162, 2007.
- (*Hornecker, 2006*) Hornecker, E. and Buur, J. Getting a Grip on Tangible Interaction: A Framework on Physical Space and Social Interaction'. In Proc. CHI'06, pp.437-446, 2006.
- (*Hornecker, 2008*) Hornecker, E., Marshall, P., Sheep Dalton, N. and Rogers, Y. Collaboration and Interference: Awareness with Mice or Touch Input. In Proc. CSCW08, ACM, pp. 167-176, 2008.
- (*Ishii, 1997*) Ishii, H. and Ullmer, B. Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms. In Proc. CHI'97, ACM Press, pp. 234-241, 1997.
- (*Johansson, 2004*) Johansson, F. The Medici Effect: Breakthrough Insights at the Intersection of Ideas, Concepts, and Cultures. Harvard Business Press, 2004.
- (*Jordà, 2003*) Jordà, S. Interactive Music Systems for Everyone: Exploring Visual Feedback as a Way for Creating More Intuitive, Efficient and Learnable Instruments. Proceedings of the Stockholm Music Acoustics Conference (SMAC 03), Stockholm (Sweden), 2003.
- (*Jordà, 2005*) Jordà, S., Kaltenbrunner, M., Geiger, G. and Bencina, R. The reacTable, In Proc. of the International Computer Music Conference (ICM05), 2005.
- (*Jordà, 2008*) Jordà, S. On stage: the reactable and other musical

- tangibles go real. *Int. J. Arts and Technology*, 1 (3-4): 268–287, 2008.
- (Jordà, 2008b)* Jordà S., Geiger, G., Alonso, M., and Kaltenbrunner, M. The reacTable: exploring the synergy between live music performance and tabletop tangible interfaces. In *Proc. TEF'07*, ACM Press, pp. 139-146, 2007.
- (Joyce, 1992)* Joyce B., Wil, M. and Showers, B. *Models of Teaching*, Boston: Allyn and Bacon, 1992.
- (Khandelwal, 2007)* Khandelwal, M. and Mazalek, A. Teaching table: a tangible mentor for pre-k math education. In *Proc. TEI '07*, pp. 191-194, 2007.
- (Kebritchi, 2008)* Kebritchi, M. and Hirumi A. Examining the pedagogical foundations of modern educational computer games. *Comput. Educ.* 51(4), 1729-1743, 2008.
- (Kelleher, 2005)* Kelleher, C. and Pausch, R. Lowering the Barriers to Programming: A Taxonomy of Programming Environments and Languages for Novice Programmers. *ACM Computing Surveys (CSUR)* 37(2), 83-137, June 2005.
- (Kelleher, 2007)* Kelleher, C. and Pausch, R. Using storytelling to motivate programming. *Magazine Communications of the ACM - Creating a science of games CACM* 50(7), 58-64, July 2007.
- (Kelleher, 2007b)* Kelleher, C. and Pausch, R. and Kiesler, S. Storytelling Alice Motivates Middle School Girls to Learn Computer Programming. In *Proc. CHI 2007*, pp. 1455-1464, 2007.
- (Kindley, 2002)* Kindley, R. The power of simulation-based e-learning. *The E-learning Developers' Journal*. pp. 1-8, September 17th, 2002.
- (Kolb, 1984)* Kolb, D. A. *Experiential Learning: Experience as the source of learning and development*. Englewood Cliffs, NJ: Prentice-Hall, 1984.
- (Krüger, 1991)* Krueger, M. W. *Artificial Reality II*. Addison-Wesley

Reading, Mass., 1991.

- (*Lampe, 2007*) Lampe, M. and Hinske, S. The Augmented Knight's Castle – Integrating Mobile and Pervasive Computing Technologies into Traditional Toy Environments”. In: Carsten Magerkurth, Carsten Roecker (Eds.): Concepts and technologies for Pervasive Games - A Reader for Pervasive Gaming Research. Vol. 1, Shaker Verlag, Aachen, Germany, 2007.
- (*Leitner, 2008*) Leitner, J., Haller, M., Yun, K., Woo, W., Sugimoto, M. and Inami, M. IncreTable, a mixed reality tabletop game experience. In Proc. ACE'08, pp. 9-16, 2008.
- (*Leitner, 2009*) Leitner, J., Haller, M., Yun, K., Woo, W., Sugimoto, M., Inami, M., Cheok, A. D. and Been-Lirn, H. D. Physical interfaces for tabletop games. ACM Comput. Entertain. 7, 4, Article 61, December 2009.
- (*Lepinski, 2010*) Lepinski, G. J., Grossman, T. and Fitzmaurice, G. The design and evaluation of multitouch marking menus. In Proc. CHI '10. ACM Press, pp. 2233-2242, 2010.
- (*López de Ipiña, 2001*) López-de-Ipiña, D. An ECA Rule-Matching Service for Simpler Development of Reactive Applications. Middleware 2001, IEEE Distributed Systems Online, 2(7), November 2001.
- (*Lu, 2011*) Lu, F., Tian, F., Jiang, Y., Cao, X., Luo, W, Li, G., Zhang, X., Dai, G. and Wang, H. ShadowStory: Creative and Collaborative Digital Storytelling Inspired by Cultural Heritage. In Proc. CHI '11, pp. 1919-1928, 2011.
- (*Lucchi, 2010*) Lucchi, A., Jermann, P., Zufferey G., Dillenbourg, P. An Empirical Evaluation of Touch and Tangible Interfaces for Tabletop Displays. In Proc. TEI 2010. ACM, pp. 177-184, 2010.
- (*Mindstorms, 2012*) Lego MINDSTORMS NXT website:
<http://mindstorms.lego.com/> Last access:

02/02/2012.

- (Magerkurth, 2003)* Magerkurth, C., Stenzel, R., Streitz, N. and Neuhold, E. A Multimodal Interaction Framework for Pervasive Game Applications. Workshop at Artificial Intelligence in Mobile System 2003(AIMS 2003), Seattle, USA, Oct. 12, 2003.
- (Maloney, 2004)* Maloney, J., Resnick, M. and Rusk, N. Scratch: A Sneak Preview. In Proc. C5'04, IEEE CS, pp. 104-109, 2004.
- (Marshall, 2007)* Marshall, P., Rogers, Y. and Hornecker, E. Are Tangible Interfaces Really Any Better Than Other Kinds of Interfaces? Workshop on Tangible User Interfaces in Context and Theory, ACM CHI 2007.
- (Maydak, 1995)* Maydak, M., Stromer, R., Mackay, H.A. and Stoddard, L.T. Stimulus classes in matching to sample and sequence production: The emergence of numeric relations. *Research in Developmental Disabilities*, 16-3, 179-204, 1995.
- (Mazalek, 2008)* Mazalek, A., Mironer, B. and Van Devender, D. The TVViews Table Role-Playing Game. *Journal of Virtual Reality and Broadcasting*, 5 (8), 2008.
- (McFarlane, 2002)* McFarlane, A., Sparrowhawk, A. and Heald, Y. Report on the educational use of games. *Teem: Teachers Evaluating Educational Multimedia*, 2002.
- (Mehta, 1982)* Mehta, N. A Flexible Machine Interface, M.A.Sc. Thesis, Department of Electrical Engineering, University of Toronto supervised by Professor K.C. Smith, 1982.
- (Michael, 2005)* Michael, D. and Chen, S. *Serious Games: Games that Educate, Train, and Inform*. Course Technology PTR, 2005.
- (Microsoft Surface, 2012)* Microsoft Corporation. Microsoft Surface. <http://www.microsoft.com/surface>. Last access: 02/02/2012.
- (MSUxGuidelines,* Microsoft Surface User Experience Guidelines,

- 2011) 26/05/2011, MSDN Library:
<http://msdn.microsoft.com/en-us/library/ff318692.aspx>
- (*Ministerio, 2007*) “Las cifras de la Educación en España. Estadísticas e indicadores. Edición 2008”, Estadísticas de la Educación, Ministerio de Educación, Política Social y Deporte, 11 Octubre, 2007. Disponible en línea en:
<http://www.mepsyd.es/mecd/jsp/plantilla.jsp?id=3131&area=estadisticas&contenido=/estadisticas/educativas/cee/2007A/cee-2007A.html>
- (*Mocholí, 2006*) Mocholí, J.A., Esteve, J.M., Jaén, J., Acosta, R. and Xech P.L. An Emotional Path Finding Mechanism for Augmented Reality Applications. In Proc. of the 5th IFIP International Conference on Entertainment Computing, pp 13-24, 2006.
- (*Mocholí, 2007a*) Mocholí, J.A. and Jaén, J. An Application of Ant Colony Optimization to Decision Making on Affective Virtual Entities. In Proc. of the 9th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, pp 419-426, 2007.
- (*Mocholí, 2007b*) Mocholí, J.A., Jaén, J. and Catalá, A. A Model of Affective Entities for Effective Ubiquitous Learning Environments. In Proc. of the 2nd International Workshop on Hybrid Artificial Intelligence Systems, pp 337-344, 2007.
- (*Mumford, 1996*) Mumford, M. D., Baughman, W. A., Threlfall, K. V., Supinski, E. P. and Constanza, D. P. Process-based measures of creative problem-solving skills: I. Problem construction. *Creativity Research Journal*, 9 (1), 63-76, 1996.
- (*Newell, 1972*) Newell, A. and Simon, H.A. *Human Problem Solving*. Englewood Cliffs: Prentice-Hall, 1972
- (*OpenLifeGrid*) Open Life Grid website, <http://openlifegrid.com/>
- (*OpenSim, 2012*) The Open Simulator (OpenSim) website, <http://opensimulator.org> Last access: 02/02/2012

- (*OSgrid, 2012*) OSGrid: the Open Source Metaverse website <http://osgrid.org/> Last access: 02/02/2012
- (*Pane, 2001*) Pane, J.F., Ratanamahatana, C.A. and Myers, B.A. Studying the Language and Structure in Non-Programmers Solutions to Programming Problems, *International Journal of Human-Computer Studies*, 54 (2), 237-264, February 2001.
- (*Papert, 1985*) Papert, S. Different Visions of Logo, *Computers in the Schools*, 2(2-3), 1985.
- (*Parkes, 2008*) Parkes, A.J., Raffle, H.S. and Ishii, H. Topobo in the wild: longitudinal evaluations of educators appropriating a tangible interface. In *Proc. CHI '08*, pp, 1129-1138, 2008.
- (*Patten, 2006*) Patten, J., Recht, B., Ishii, H. Interaction techniques for musical performance with tabletop tangible interfaces. In *Proc. ACE'06*, article 27, 2006
- (*Pérez, 2008*) Pérez De Pablos, S. "Es preocupante que España tenga un 30% de fracaso escolar", *Entrevista a Fernando Reimers, Catedrático de Educación en la Universidad de Harvard. El País (Edición Impresa), Sección Educación, Madrid, 17 Noviembre 2008.* También disponible en línea:
- (*Piaget, 1967*) Piaget, J. *Six psychological studies*. New York: Vintage books, 1967.
- (*Picocricket, 2012*) PicoCricket website: <http://www.picocricket.com/> Last access: 02/02/2012
- (*Pisa, 2007*) "PISA 2006: Science Competencies for Tomorrow's World", Programme for International Student Assesment (PISA), Organization for Economic Cooperation and development (OECD), OECD Publishing, 14 Dec. 2007. Disponible en línea en: <http://www.pisa.oecd.org/>
- (*Raffle, 2004*) Raffle, H.S., Parkes, A.J. and Ishii, H. Topobo: a constructive assembly system with kinetic memory. In *Proc. CHI '04, ACM*, pp. 647-654, 2004

- (*Repenning, 2000*) Repenning, A. AgentSheets®: an Interactive Simulation Environment with End-User Programmable Agents. *Interaction*, 2000.
- (*Repenning, 2000b*) Repenning A., Ioannidou A., and Zola J. AgentSheets: End-User Programmable Simulations. *Journal of Artificial Societies and Social Simulation*, 3(3). 2000.
- (*Resnick, 1988*) Resnick, M., Ocko S. and Papert S. LEGO, Logo, and Design, *Children's Environments Quarterly* 5(4), 14-18, 1988.
- (*Resnick, 1996*) Resnick, M., Martin, F., Sargent, R. and Silverman B. Programmable Bricks: Toys to Think With. *IBM Systems Journal* 35(3-4), 443-452, 1996.
- (*Resnick, 1998*) Resnick, M. Technologies for Lifelong Kindergarten. *Educational Technology Research and Development*, 46(4), 43-55, 1998.
- (*Resnick, 2002*) Resnick, M. Rethinking Learning in the Digital Age. In *The Global Information Technology Report: Readiness for the Networked World*, edited by G. Kirkman. Oxford University Press, 2002.
- (*Resnick, 2007*) Resnick, M. All I Really Need to Know (About Creative Thinking) I Learned (By Studying How Children Learn) in Kindergarten. In *Proc. C&C'07, ACM*, 1-6, 2007
- (*Robocup, 2012*) RoboCup website, <http://www.robocup.org/> Last access: 02/02/2012
- (*Rogers, 2009*) Rogers, Y., Lim, Y-K., Hazlewood, W.R. and Marshall, P. Equal opportunities: Do shareable interfaces promote more group participation than single users displays? *Human-Computer Interaction*, 24(1-2), 79-116, 2009
- (*Rhodes, 1961*) Rhodes, M. An analysis of creativity. *Phi Delta Kappan*, 42(7), 305-310, April 1961.
- (*Rick, 2011*) Rick, J., Marshall, P. and Yuill, N. Beyond one-size-fits-all: how interactive tabletops support collabora-

- tive learning. In Proceedings of the 10th International Conference on Interaction Design and Children (IDC '11). ACM, pp. 109-117, 2011.
- (Rube-Goldberg, 2012)* The Official Rube-Goldberg Machine website: <http://www.rubegoldberg.com/?page=bio>
- (Runco, 1995)* Runco, M. A. and Chand, I. Cognition and Creativity. *Educational Psychology Review*, 7, 243-267, 1995.
- (Sawyer, 2006)* Sawyer, R. K. Explaining creativity: The science of human innovation. New York: Oxford University Press, 2006.
- (Schöning, 2008b)* Schöning, J., Hecht, B., Raubal, M., Krüger, A., Marsh, M. and Rohs, M. Improving interaction with virtual globes through spatial thinking: helping users ask "why?". In Proc. IUI '08. ACM, pp. 129-138, 2008.
- (Schöning, 2008)* Schöning, J., Brandl, P., Daiber, F., Echtler, F., Hilliges, O., Hook, J., Löchtefeld, M., Motamedi, N., Muller, L., Olivier, P., Roth, T. and von Zadow, U: Multi-Touch Surfaces: A Technical Guide. Technical Report TUM-I0833: Technical Reports of the Technical University of Munich, (2008)
- (Scott, 2004)* Scott, G., Leritz, L. and Mumford, M. The Effectiveness of Creativity Training: A Quantitative Review. *Creativity Research Journal*, 16(4), 361-388, 2004.
- (SecondLife, 2012)* Second Life Web site: <http://secondlife.com/> Last access: 02/02/2012
- (Shen, 2005)* Shen, C., Hancock, M.S., Forlines, C., Vernier, F.D., CoR2Ds: Context-Rooted Rotatable Draggables for Tabletop Interaction, In Proc. CHI'05. ACM Press, 2005
- (Shneiderman, 1983)* Shneiderman, B., Direct Manipulation: A Step Beyond Programming Languages, *Computer*, 16(8), 57-69, 1983.

- (*Skinner, 1953*) Skinner, B. F. Science and Human Behavior. Macmillan Free Press, 1953.
- (*Smith, 2010*) Smith J.D. and Graham, T.C.N. Raptor: Sketching Games with a Tabletop Computer. In Proc. Future-play'10, pp. 191-198, 2010.
- (*Smith, 2009*) Smith, J.D. Raptor: Sketching Video Games With a Tabletop Computer. Thesis. 2009.
- (*Song, 2010*) Song, H., Kim, B., Lee, B. and Seo, J. A comparative evaluation on tree visualization methods for hierarchical structures with large fan-outs. In Proc. CHI '10. ACM, pp. 223-232, 2010.
- (*Sternberg, 1996*) Sternberg, R.J. and Lubart, T.I. Investing in creativity. American Psychologist 51(7), 677–688, 1996.
- (*Sternberg, 1999*) Sternberg, J.S. et al. Handbook of Creativity, Robert. J. Stenberg Ed., Cambridge University Press, 1999.
- (*Streitz, 2001*) Streitz, N.A., Tandler, P., Müller-Tomfelde, C., Konomi and S. Roomware: Towards the Next Generation of Human-Computer Interaction based on an Integrated Design of Real and Virtual Worlds. In: J. A. Carroll (Ed.): Human-Computer Interaction in the New Millennium, Addison Wesley, pp. 553-578, 2001.
- (*Susman, 1978*) Susman, G. and Evered, R. D. An Assessment of the Scientific Merits of Action Research. Administrative Science Quarterly 23, 582-603, December 1978.
- (*Suzuki, 1995*) Suzuki, H. and Kato, H. Interaction-level support for collaborative learning: AlgoBlock—an open programming language. In Proc. CSCIL'95, John L. Schnase and Edward L. Cunnius (Eds.). L. Erlbaum Associates Inc., Hillsdale, NJ, USA, pp. 349-355, 1995.
- (*Treffinger, 1996*) Treffinger, D. J., Isaksen, S. G. and Dorval, K. B. Climate for creativity and innovation: Educational implications. Sarasota, FL: Center for Creative

- Learning, 1996.
- (*Treffinger, 1996b*) Treffinger, D. J. Creativity, creative thinking, and critical thinking: In search of definitions. Sarasota, FL: Center for Creative Learning, 1996
- (*Treffinger, 2002*) Treffinger, D. J., Young, G. C., Selby, E. C. and Shepardson, C. Assessing creativity: A guide for educators (RM02170). Storrs, CT: The National Research Center on the Gifted and Talented, University of Connecticut, 2002.
- (*Tuddenham, 2010*) Tuddenham, P., Kirk, D. and Izadi, S. Graspables revisited: multi-touch vs. tangible input for tabletop displays in acquisition and manipulation tasks. In Proc. CHI '10. ACM Press, pp. 2223-2232, 2010.
- (*Ullmer, 1998*) Ullmer, B., Ishii, H., Glas D. mediaBlocks: physical containers, transports, and controls for online media. In Proc. SIGGRAPH '98, pp. 379-386, 1998
- (*Ullmer, 2001*) Ullmer, B. and Ishii, H. Emerging Frameworks for Tangible User Interfaces, In Human-Computer Interaction in the New Millenium, Ed. John M. Carroll, Addison-Wesley, pp.579-601, 2001.
- (*Vandoren, 2008*) Vandoren, P., Van Laerhoven, T., Claesen, L., Taelman, J., Raymaekers, C. and Van Reeth, F. IntuPaint: Bridging the gap between physical and digital painting, In Proc. ITS'08, ACM Press, pp. 65-72, 2008.
- (*van Strien, 1997*) van Strien, P.J. Towards a methodology of psychological practice: The regulative cycle. Theory & Psychology, 7(5), 683-700, 1997.
- (*Verma, 2003*) Verma, S. and Larsson, R.W. Examining Adolescent Leisure Time Across Cultures: Developmental Opportunities and Risks. New Directions in Child and Adolescent Development Series, 2003.
- (*Vyas, 2009a*) Vyas, D.M., Heylen, D.K.J., Nijholt, A. and van der Veer, G.C. Collaborative Practices that Support Creativity in Design,. In Proc. of the 11th European

- Conference on Computer-Supported Cooperative Work, I. Wagner (eds), Springer Verlag, Berlin, pp. 151-170, 2009.
- (*Vyas, 2009b*) Vyas, D.M., van der Veer, G.C., Heylen, D.K.J. and Nijholt, A. Space as a Resource in Creative Design Practices,. In Proc. INTERACT 2009, Springer Verlag, pp. 169-172, 2009.
- (*Vyas, 2010a*) Vyas, D., Nijholt, A., Heylen, D., Kröner, A. and van der Veer, G. Remarkable Objects: Supporting Collaboration in a Creative Environment. In Proc. UBICOMP 2010, pp. 37-40, 2010.
- (*Vyas, 2010b*) Vyas, D.M., Nijholt, A. and Kroener, A. CAM: A Collaborative Object Memory System. In Proc. of the 12th International Conference on Human-Computer Interaction with Mobile Devices and Services (Mobile HCI), ACM, New York, pp. 415-416, 2010.
- (*Vyas, 2010c*) Vyas, D.M. and Nijholt, A. Building boundaries on Boundary Objects: A Field study of a UbiComp tool in a Design Studio, International reports on socio-informatics, 7(1), 282-299, 2010.
- (*Vygotsky, 1978*) Vygotsky, L.S. Mind in Society: The development of Higher Psychological Processes. Harvard University Press, 1978.
- (*Wallin, 2006*) Wallin, D. Touchlib: an opensource multi-touch framework. 2006.
- (*Wang, 2010*) Wang, J., Farroq, U. and Carroll J. M. Does Design Rationale Enhance Creativity. Human technology 6(1), 129-149, May 2010.
- (*Weiss, 2009*) Weiss, M., Jennings, R., Khoshabeh, R., Borchers, J., Wagner, J., Jansen, Y. and Hollan, J. D. SLAP widgets: bridging the gap between virtual and physical controls on tabletops. In Proc. CHI '09. ACM, pp. 3229-3234, 2009.
- (*Whitley, 2006*) Whitley, K. N., Novick, L. R. and Fisher, D. Evidence

in favor of visual representation for the dataflow paradigm: An experiment testing LabVIEW's comprehensibility. *Int. J. Hum.-Comput. Stud.* 64(4), 281-303, 2006.

(Wieringa, 2009)

Wieringa, R. 2009. Design science as nested problem solving. In Proc. of the 4th International Conference on Design Science Research in Information Systems and Technology (DESRIST '09). ACM, New York, NY, USA, Article 8, 12 pages, 2009.

(Wing, 2006)

Wing, J. M. Computational Thinking. *Commun. ACM* 49(3), 33-35, March 2006.

(Zadow, 2012)

Zadow, U. libavg: A high-level Multimedia Platform. <http://www.libavg.de>, Last access: 02/02/2012

(Zagal, 2008)

Zagal, J.P., Fernandez-Vara, C. and Mateas, M. Rounds, Levels, and Waves: The Early Evolution of Gameplay Segmentation", *Games & Culture* 3(1), 175-198, 2008.

(Zhao, 2006)

Zhao, S., Agrawala, M., Hinckley, K. Zone and polygon menus: using relative position to increase the breadth of multi-stroke marking menus. In Proc. CHI'06, ACM Press, pp. 1077-1086, 2006.

Creativity is a skill of special interest for human development since it is a dimension that allows individuals and eventually society to face new problems and challenges successfully. The environment, as well as other social factors, can have an effect on the development of such a relevant skill. It is therefore interesting to explore this skill in the context of using new information technology.

Tabletop technology is to a great extent facilitator of characteristics behind creative processes such as communicative processes, ideas exchange and collaborative interaction between individuals. This thesis explores the suitability of interactive surfaces in collaborative creative tasks with teenager students by means of software for the construction of 2D game ecosystems for tabletops. It aims to consider more rewarding and engaging learning activities such as those focused on playing to create games and their subsequent play.

