UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Departamento de Sistemas Informáticos y Computación
Máster Universitario en Ingeniería y Tecnología de Sistemas Software

DISEÑO, IMPLEMENTACIÓN Y EVALUACIÓN DE UN SISTEMA DE JUEGO COLABORATIVO BASADO EN REALIDAD AUMENTADA SIN MARCADORES

Trabajo Fin de Máster

Autor: Lissette Geoconda Lopez Faican

Tutor: Francisco Javier Jaén Martínez

Julio 2018

Resumen

El presente trabajo describe el diseño e implementación de un juego móvil de realidad aumentada que integra un escenario multijugador con dos modalidades de juego, competitivo vs colaborativo. El juego tiene como objetivo principal fomentar la comprensión de los estados emocionales, estimular en los niños su imaginación e incentivar la comunicación y colaboración entre ellos, agregando a esto, una visión de realidad aumentada sin marcadores para combinar objetos virtuales y físicos tangibles del mundo real. Finalmente se aborda una evaluación experimental para un ambiente de juego de emociones a fin de analizar las aproximaciones existentes entre las dos modalidades de juego.

Palabras clave: juego móvil, multijugador, realidad aumentada, colaborativo, competitivo.

Summary

This work describes the design and implementation of a mobile game of augmented reality that integrates a multiplayer scenario with two game modes, competitive vs collaborative. The main objective of the game is to foster the understanding of emotional states, stimulate children's imagination and encourage communication and collaboration between them, adding to this, a vision of augmented reality without markers to combine virtual objects and tangible physical from the real world. Finally, an experimental evaluation for an environment of emotions game is approached in order to analyze the existing approximations between the two game modalities.

Keywords: mobile game, multiplayer, augmented reality, collaborative, competitive.

Agradecimientos

Mi más profundo y sincero agradecimiento a Dios por guiar y bendecir todo esfuerzo y dedicación en mi vida, y a mis padres y hermanas, por su apoyo incondicional.

A Javier Salavert, compañero de trabajo del DSIC por facilitarme el tiempo necesario para poder culminar con éxito el presente trabajo.

A los distinguidos catedráticos de la Universidad Politécnica de Valencia, por guiarme y compartir sus sabias enseñanzas en el transcurso de mi formación académica.

Al colegio Vicente Gaos por las facilidades y apoyo brindado durante el desarrollo de los experimentos presentados.

A todos los miembros del grupo de investigación ISSI por sus contribuciones brindadas y en particular a Jorge Montaner Marco por su colaboración en el proceso de observación para la evaluación experimental.

De manera especial agradezco a mi director Francisco Javier Jaén Martínez, quien con sus valiosos conocimientos y dedicación supo guiarme en el desarrollo del presente trabajo.

Tabla de contenidos

1. Introducción 9
   1.1. Contexto 9
   1.2. Objetivos del proyecto 10
   1.3. Estructura del proyecto 11
2. Estado del arte 12
   2.1. Gamificación para el aprendizaje colaborativo 12
   2.2. Realidad aumentada en dispositivos móviles 13
      2.2.1. Entorno educativo 13
      2.2.2. Entorno de inclusión 14
      2.2.3. Entorno médico 14
      2.2.4. Juegos móviles para la motivación 15
3. Análisis 17
   3.1. Requerimientos del juego multijugador 17
      3.1.1. Generales 17
      3.1.2. Competitivo 18
      3.1.3. Colaborativo 19
   3.2. Historias de usuario 20
   3.3. Diagrama de clases 22
4. Diseño e Implementación 26
   4.1. Tecnología utilizada 26
      4.1.1. Motor de videojuegos 26
      4.1.2. Infraestructura de Red y Comunicación 26
      4.1.3. Plataforma de realidad aumentada 27
      4.1.4. Plataforma hardware de Destino 28
   4.2. Diseño de la Interfaz de usuario 28
   4.3. Diseño e Implementación de la lógica del juego multijugador en Unity 35
      4.3.1. Configuración inicial del juego 37
      4.3.2. GameObjects implementados 40
      4.3.3. Prefabs implementados 45
5. Evaluación experimental 55

5.1. Descripción de las tareas 55
5.2. Participantes 57
5.3. Resultados 58
5.3.1. Resultados de los cuestionarios 58
5.3.2. Resultados observacionales 66
5.4. Discusión 76
5.4.1. Resultados de los cuestionarios 76
5.4.2. Resultados observacionales 77
6. Conclusiones y trabajos futuros 80
7. Bibliografía 82

Lista de tablas

Tabla 1: Realidad aumentada en la educación 14
Tabla 2: Realidad aumentada en la inclusión 14
Tabla 3: Realidad aumentada en el entorno medico 15
Tabla 4: Juegos móviles de realidad aumentada para la motivación 16
Tabla 5: Cuestionario para evaluar cada modalidad de juego 56
Tabla 6: Cuestionario para comparar las modalidades del juego 56
Tabla 7: Plantilla Observacional para cada modalidad del juego 57
Tabla 8: Plantilla para la observación de la comunicación en el juego colaborativo 57
Tabla 9: Resultados pregunta Nro. 1, modalidad del juego 59
Tabla 10: Resultados pregunta Nro. 2, modalidad de juego 59
Tabla 11: Resultados pregunta Nro. 3, modalidad de juego 60
Tabla 12: Resultados pregunta Nro. 4, modalidad de juego 60
Tabla 13: Resultados pregunta Nro. 5, modalidad de juego 61
Tabla 14: Resultados pregunta Nro. 6, modalidad de juego 62
Tabla 15: Resultados pregunta Nro. 7, modalidad de juego 63
Tabla 16: Resultados del test Wilcoxon entre las dos modalidades de juego (*P < 0.05) 63
Tabla 17: Resultados pregunta Nro. 1, modalidad competitivo vs colaborativo 64
Tabla 18: Resultados pregunta Nro. 2, modalidad competitivo vs colaborativo 64
Tabla 19: Resultados pregunta Nro. 3, modalidad competitivo vs colaborativo 65
Tabla 20: Resultados pregunta Nro. 4 y 5, modalidad competitivo vs colaborativo 65
Tabla 21: Resultados observacionales, afecto 66
Tabla 22: Resultados observacionales, actividad física 67
Tabla 23: Resultados observacionales, interacción social 68
Tabla 24: Resultados observacionales, uso de la herramienta 68
Tabla 25: Resultados observacionales, satisfacción 69
Tabla 26: Resultados índice Kappa, observación factores competitivo y colaborativo 70
Tabla 27: Resultados del test Wilcoxon de la plantilla observacional (*P < 0.05) 70
Tabla 28: Resultados observacionales, Mantener la compresión mutua 71
Tabla 29: Resultados observacionales, Gestión de diálogo 71
Tabla 30: Resultados observacionales, Conjunto de Información 72
Tabla 31: Resultados observacionales, Alcanzando el consenso 72
Tabla 32: Resultados observacionales, División de tareas 73
Tabla 33: Resultados observacionales, Gestión del tiempo 74
Tabla 34: Resultados observacionales, Coordinación técnica 74
Tabla 35: Resultados observacionales, Interacción recíproca 75
Tabla 36: Resultados observacionales, Orientación de tareas individuales 75
Tabla 37: Resultados índice Kappa, observación de la comunicación 76

Lista de figuras

Ilustración 1: Diagrama de clases 23
Ilustración 2: Photon Unity Networking (PUN) 27
Ilustración 3: Placenote 28
Ilustración 4: Interfaz gráfica, pantalla inicial 29
Ilustración 5: Interfaz gráfica, app de Placenote 29
Ilustración 6: Interfaz gráfica, escena del juego 30
Ilustración 7: Interfaz gráfica, nivel de vida del personaje 31
Ilustración 8: Interfaz gráfica, selección y lanzamiento del proyectil 32
Ilustración 9: Interfaz gráfica, proyectiles de tipo recuperación 33
Ilustración 10: Interfaz gráfica, tiempo colaborativo 34
Ilustración 11: Interfaz gráfica, información del juego 35
Ilustración 12: Escena Conexión 35
Ilustración 13: Diagrama de clases y componentes del Sistema de Juego 36
Ilustración 14: Configuración Placenote 37
Ilustración 15: Configuración PUN 37
Ilustración 16: Configuración inicial del juego 38
Ilustración 17: Configuración del juego, clases 39
Ilustración 18: GameObjectGame 40
Ilustración 19: GameObjectRoom 41
Ilustración 20: PhotonNetwork, créate Room 41
Ilustración 21: GameObjectGame 42
Ilustración 22: función startGame 42
Ilustración 23: Lista personajes y proyectiles 43
Ilustración 24: función createConfigurationInstantiate 43
Ilustración 25: función showProjectilePlayer 44
Ilustración 26: fución createProjectileTypeRecover 44
Ilustración 27: GameObjectAudioGame 45
Ilustración 28: Emociones_actor_enojado_01 y 02 46
Ilustración 29: Emociones_actor_triste_01 y 02 46
Ilustración 30: Emociones_actor_alegre_01 y 02 46
Ilustración 31: Prefab Actor 47
Ilustración 32: Prefab Scritp Actor 47
Ilustración 33: Prefab Scritp Actor, porcentaje de daño 49
Ilustración 34: Prefab Actor, componentes Photon Pun 50
Ilustración 35: Proyectiles 51
Ilustración 36: Prefab Projectile 52
Ilustración 37: Prefab Script Projectile 52
Ilustración 38: Prefab Script Projectile, método Update 53
Ilustración 39: Prefab Projectile, componentes Photon Pun 54
Ilustración 40: Evaluación experimental, pareja de niños 58
Ilustración 41: Resultados pregunta Nro. 1, modalidad de juego 59
Ilustración 42: Resultados pregunta Nro. 2, modalidad de juego 59

Ilustración 43: Resultados pregunta Nro. 3, modalidad de juego..... 60
Ilustración 44: Resultados pregunta Nro. 4, modalidad de juego..... 61
Ilustración 45: Resultados pregunta Nro. 5, modalidad de juego..... 61
Ilustración 46: Resultados pregunta Nro. 7, modalidad de juego..... 62
Ilustración 47: Resultados pregunta Nro. 1, modalidad competitivo vs colaborativo..... 64
Ilustración 48: Resultados pregunta Nro. 2, modalidad competitivo vs colaborativo..... 64
Ilustración 49: Resultados pregunta Nro. 3, modalidad competitivo vs colaborativo..... 65
Ilustración 50: Resultados pregunta Nro. 4 y 5, modalidad competitivo vs colaborativo..... 65
Ilustración 51: Resultados observacionales, afecto..... 66
Ilustración 52: Resultados observacionales, actividad física..... 67
Ilustración 53: Resultados observacionales, interacción social..... 68
Ilustración 54: Resultados observacionales, uso de la herramienta..... 69
Ilustración 55: Resultados observacionales, satisfacción..... 69
Ilustración 56: Resultados observacionales, Mantener la compresión mutua..... 71
Ilustración 57: Resultados observacionales, Gestión de diálogo..... 71
Ilustración 58: Resultados observacionales, Conjunto de Información..... 72
Ilustración 59: Resultados observacionales, Alcanzando el consenso..... 73
Ilustración 60: Resultados observacionales, División de tareas..... 73
Ilustración 61: Resultados observacionales, Gestión del tiempo..... 74
Ilustración 62: Resultados observacionales, Coordinación técnica..... 74
Ilustración 63: Resultados observacionales, Interacción recíproca..... 75
Ilustración 64: Resultados observacionales, Orientación de tareas individuales..... 76

1. Introducción

Actualmente, las tendencias pedagógicas han puesto su atención en la llamada educación afectiva, una estrategia que permite mejorar el clima de aprendizaje mediante el uso correcto de emociones y sentimiento[1].

Existen diversas técnicas que son poderosos generadores de emociones para fomentar un aprendizaje activo, una de ellas, la gamificación, particularmente útil en entornos de aprendizaje para promover la colaboración[1][2].

La colaboración actualmente se ha convertido en un tema de investigación muy popular ya que presenta una gran cantidad de beneficios para el aprendizaje, como, propiciar un ambiente de cooperación e intercambio de ideas para activar procesos mentales de razonamiento, comprensión y pensamiento crítico. De hecho, la colaboración ha sido identificada como una habilidad clave del siglo 21 que se incluye en la mayoría de los modelos educativos actuales[1].

Bajo este panorama, el presente trabajo titulado "Diseño, implementación y evaluación de un sistema de juego colaborativo basado en realidad aumentada sin marcadores" pretende evaluar el uso de la realidad aumentada sin marcadores como tecnología para implementar escenarios de juego colaborativo que puedan ser usados para la mejora de la socialización y la adquisición de habilidades de comunicación y de inteligencia emocional. Para ello se basa en el desarrollo de una aplicación para dispositivos móviles que implementa un escenario multijugador configurable y disponible en dos modalidades de juego competitivo vs colaborativo. El juego integra tecnología de realidad aumentada sin marcadores para crear un escenario de geolocalización que hará que las posiciones de los objetos virtuales relativas al mundo real sean siempre persistentes en todos los jugadores.

1.1. Contexto

La Realidad Aumentada (RA) es una tecnología que permite superponer objetos virtuales sobre nuestra visión de la realidad, lo que da como resultado un escenario que permite al usuario no perder el contacto con el mundo real e interactuar al mismo tiempo con la información virtual superpuesta[3].

La Realidad Aumentada para la superposición de información sobre la realidad funciona a partir de tres enfoques: los patrones de disparo del software o marcadores, la geo-localización y la interacción con Internet[4].

- **Los patrones de disparo del software o marcadores**: es una marca, un dibujo o una imagen que una aplicación de realidad aumentada puede reconocer fácilmente, para a través de ella y de una cámara incorporar al entorno real una serie de objetos.
- **La geo-localización**: consiste en detectar la posición geográfica del usuario con Global Posición Sistema (GPS). Al identificar la ubicación, podemos identificar qué es lo que se encuentra dentro del rango de visión del usuario y superponer objetos virtuales referentes a su visión espacial. Sin embargo la realidad aumentada con geolocalización GPS tiene algunas limitaciones, entre ellas: solo es preciso a 9 metros de distancia relacionados a la posición del usuario y no funciona correctamente en interiores.
- **Interacción con Internet**: existen aplicaciones que relacionan las imágenes captadas con otras similares existentes en las bases de datos de Internet. El equipo busca los elementos esenciales de la imagen real y superpone los hipervínculos con información complementaria de Internet.

La RA está siendo desarrollada por diferentes grupos de investigación en diferentes contextos, como, el seguimiento de la posición del usuario, procesado de la señal, visualización de la información, visión por ordenador, generación de imágenes virtuales, renderizado de gráficos, estructuración de la información, y computación distribuida [3]. Un ejemplo de ello es la comunidad de Vertical que está desarrollando el SDK Placenote[1] de realidad aumentada sin marcadores para enlazar de forma persistente el contenido virtual a lugares y objetos del mundo real.

Actualmente, la RA es una tecnología emergente e innovadora que se está incorporando en diferentes disciplinas, entre algunas citadas, educación, inclusión, aulas hospitalarias, turismo, simulación de entrenamiento, etc. Sin embargo, es necesario también crear aplicaciones en la industria de videojuegos que estén principalmente enfocados a ser un complemento motivacional en los niños y que permitan favorecer en ellos acciones socializantes como la comunicación y colaboración.

1.2. Objetivos del proyecto

Para culminar con éxito el presente proyecto se planteó los siguientes objetivos de trabajo:

[1] https://placenote.com/

- Definir un prototipo de juego multijugador configurable disponible en dos modalidades de juego competitivo y colaborativo.
- Implementar el prototipo en una plataforma de desarrollo de juegos para dispositivos móviles e integrar tecnología de realidad aumentada en la escena del juego.
- Evaluar el juego móvil realizando una comparativa entre la experiencia de usuario para el entorno competitivo y colaborativo.

1.3. Estructura del proyecto

La estructura del proyecto está compuesta por siguientes capítulos:

- *Capítulo 2,* Estado de arte: detalla las temáticas que son útiles para la comprensión del presente trabajo.
- *Capítulo 3,* Análisis: lista los requerimientos, las historias de usuario y el diagrama de clases que son la base para la implementación del juego multijugador.
- *Capítulo 4*, Diseño e Implementación: detalla la tecnología utilizada, la interfaz de usuario así como los componentes implementados.
- *Capítulo 5,* Evaluación experimental*:* describe los resultados y discusión de la evaluación del juego realizado a un agrupo de niños de quinto curso de primaria del colegio Vicente Gaos.
- *Capítulo 6*, Conclusiones y trabajos futuros: establece las ideas generadas tras la culminación del trabajo y propuestas de mejora.
- *Capítulo 7,* Bibliografía*:* lista las fuentes bibliográficas utilizadas en el desarrollo del proyecto.

Finalmente el trabajo culmina con los anexos que contiene los scripts implementados en la plataforma de desarrollo de juegos Unity.

2. Estado del arte

2.1. Gamificación para el aprendizaje colaborativo

La colaboración en el aula se ha convertido en un tema de investigación muy popular ya que permite a los estudiantes involucrarse en actividades de grupo que no sólo hacen incrementar el aprendizaje, sino que también producen otros beneficios, como la mejora del desarrollo de relaciones y habilidades sociales [5]. De hecho, existen diversos trabajos enfocados al aprendizaje colorativo. Uno de ellos, el estudio de Tsay y Brandy [6] que explora la relación entre el aprendizaje cooperativo y el rendimiento académico, específicamente en el campo de la comunicación, donde indican que el aprendizaje cooperativo es un fuerte predictor del rendimiento académico de un estudiante.

Hay muchas técnicas para promover la colaboración entre los estudiantes, una de ellas es a través de gamificación [1]. La gamificación sugiere el poder utilizar elementos del juego, y el diseño de juegos, para mejorar el compromiso y la motivación de los participantes [5]. Tal y como define Deterding et al. [7], se refiere al "uso de elementos de diseño característicos de juegos en contextos que no son de juego".

A partir del análisis de la literatura, Kocadere and Çağlar [8], identifican algunos beneficios de gamificación en contextos de aprendizaje, tales como: ser capaz de crear un ambiente de aprendizaje agradable, asegurar la participación activa, y aumentar el rendimiento [1].

Las tecnologías más populares de juegos para el aprendizaje siguen siendo las consolas de video tradicionales y ordenadores de sobremesa o portátiles. Estas plataformas, sin embargo, presentan varios inconvenientes para los niños, por ejemplo, requieren que los usuarios permanezcan en un solo lugar lo que les impide moverse y hacer ejercicio; y la mayoría de estas son mono-usuario, lo que complica el diseño de actividades para fomentar las habilidades sociales y de colaboración [1].

La asequibilidad y el uso común de los dispositivos tales como teléfonos inteligentes o tabletas, son alternativas de tecnología que mediante juegos pueden apoyar la construcción de un espacio social positivo para el aprendizaje colaborativo. Además, si los dispositivos se encuentran dispersos en un área grande, la actividad física se puede fomentar, lo cual es un factor clave en el desarrollo de los niños. De ahí el interés de combinar la gamificación y el aprendizaje colaborativo [1].

2.2. Realidad aumentada en dispositivos móviles

La Realidad Aumentada Móvil (RAM) se define como la realidad aumentada generada y recuperada con dispositivos móviles en entornos móviles. Varias tecnologías como el procesamiento móvil, el reconocimiento de imágenes, el seguimiento de objetos, la pantalla y los sensores para medir la ubicación y la orientación se utilizan en teléfonos inteligentes, lo que permite que la RA se vuelva RAM [9].

Las tecnologías de RA permiten desarrollar aplicaciones de forma más innovadora abordando una gama de áreas de aplicación como, la educación, publicidad, turismo, juegos, servicios médicos, inclusión, otros. A continuación se da a conocer algunas investigaciones de la realidad aumentada en el contexto educativo, inclusión, entorno medico e industria de videojuegos.

2.2.1. Entorno educativo

La realidad aumentada es una prometedora tecnología, ya presente en muchas aulas como ayuda al proceso de enseñanza-aprendizaje. La tabla 1 detalla algunas investigaciones de la RA en el campo educativo:

Investigación	**Descripción**
Realidad aumentada y educación: análisis de experiencias prácticas [10].	Describe proyectos RA realizados en centros educativos, entre algunos citados: • Libros didácticos con RA • Software realitat3 • Proyectos que integran juegos y educación: Environmental detectives del Instituto Tecnológico de Massachusetts (MIT), Proyecto enredaMadrid (Telefónica) y Libregeosocial (Universidad Rey Juan Carlos)
La realidad aumentada y su aplicación en la educación superior [11].	Describen estudios realizados sobre la explotación didáctica de la Realidad Aumentada, así como las potencialidades que ofrece.
La Realidad Aumentada como complemento motivacional. Libros Aumentados y Reconstrucción 3D [12].	Proponen una metodología para incentivar el aprendizaje en tareas de campo mediante el uso de la reconstrucción 3D y en el ámbito escolar mediante libros aumentados.
Juegos perceptivos con realidad aumentada para trabajar contenido científico [13].	Detalla aplicaciones RA que se están implementando para abordar contenidos científicos en la educación infantil, primaria y secundaria, dada su capacidad para despertar el interés y la motivación entre los estudiantes.

Estudio de apps de realidad aumentada para su uso en campos de aprendizaje en un entorno natural [14].	Presenta un análisis de tres aplicaciones de RA gratuitas para la educación primaria, Aurasma, Augment y Aumentaty.

Tabla 1: Realidad aumentada en la educación

2.2.2. Entorno de inclusión

Los sistemas basados en RA pueden ser de gran utilidad para el proceso de inclusión, un ejemplo de ello, el ser usados como recursos didácticos en el proceso comunicacional, social y cognitivo. La tabla 2, describe algunas investigaciones en este contexto.

Investigaciones	**Descripción**
Aplicación móvil para ayudar al aprendizaje de niños autistas [15].	Presentan una aplicación Android que genera un ambiente visual y aural en las actividades diarias de las personas con TEA. En el entorno visual se implementa la realidad aumentada para describir visualmente el uso de artículos comunes y el entorno auditivo se utiliza sonidos para cada elemento común.
Development of a mobile application in augmented reality to improve the communication field of autistic children at a Neurorehabilitar Clinic [16].	Presenta una aplicación móvil de realidad aumentada que tiene como objetivo mejorar la comunicación en niños con autismo.
Uso de nuevas tecnologías TICS - realidad aumentada para tratamiento de niños TEA un diagnóstico inicial [17].	Realizan un diagnóstico de la realidad aumentada como recurso didáctico en el proceso comunicacional, social y cognitivo en niños diagnosticados con Trastorno del Espectro Autista (TEA).
La realidad aumentada aplicada al aprendizaje en personas con Síndrome de Down: un estudio exploratorio [18].	Exploran la utilidad de la Realidad Aumentada en la comunicación y el aprendizaje de personas con Síndrome de Down.

Tabla 2: Realidad aumentada en la inclusión

2.2.3. Entorno médico

El uso de las aplicaciones de realidad aumentada es de verdadero interés para el entorno médico con diferentes propósitos, educación, entrenamiento, proceso de rehabilitación, entre otros. La tabla 3 da a conocer algunas investigaciones.

Investigaciones	**Descripción**
BandApp: Monitorización del ritmo respiratorio con realidad aumentada [19].	Detallan una aplicación que permite visualizar en tiempo real los datos de un sensor para la detección y prevención del "Síndrome de muerte

	súbita del lactante". La aplicación emplea técnicas de realidad aumentada (RA) para enriquecer el entorno.
Systematic review on the effectiveness of augmented reality applications in medical training [20].	Presentan una revisión sistemática para saber en qué medida la realidad aumentada se utiliza para el apoyo, entrenamiento o educación a los médicos profesionales.
Realidad aumentada y virtual en entornos hospitalarios [21].	Describe una aplicación que simula un quirófano con el equipamiento médico básico y el mobiliario habitual.
Aplicación de la Realidad Aumentada para la asistencia en enseñanza médica [22].	Dan a conocer una aplicación de realidad aumentada para la unidad de telemedicina, que tiene como funcionalidad en tiempo real enviar al hospital o a un aula lo mismo que ve el cirujano en el quirófano.
Integración de una Interfaz de Realidad Aumentada para Estimulación Visual de los Usuarios en Sistemas Robóticos para Terapia Ocupacional [23].	Describe una interfaz de realidad aumentada para la rehabilitación de hombro y codo. La interfaz es presentada como una herramienta complementaria de estimulación visual para la recuperación cognitiva del paciente en tareas de terapia ocupacional.

Tabla 3: Realidad aumentada en el entorno medico

2.2.4. Juegos móviles para la motivación

La RA puede revolucionar juegos pensados para una nueva era, como Pokémon Go, que incluso provocó que la gente saliera de su casa para lograr capturar todas las criaturas de ese universo [24]. Estos juegos pueden ser creados con diferentes propósitos, un ejemplo, el favorecer acciones socializantes y fomentar vínculos emocionales y motivacionales. La tabla 4 presenta algunas aplicaciones que estas orientadas para este propósito.

App	**Descripción**
TGo Games, una app con Realidad Aumentada para fomentar el compromiso de trabajadores y clientes [24].	Aplicación móvil que se presenta en forma de juego, para proponer iniciativas y fomentar la participación, compromiso y motivación. La aplicación TGo Games desarrollada por Indra [2] es adaptable para configurar misiones o retos, tanto individuales como colectivos, en los que los usuarios tienen que conseguir diferentes objetivos de manera lúdica y creativa. Su objetivo es crear mayores vínculos emocionales y

[2] https://www.indracompany.com/

	motivacionales con los participantes.
Juglar, realidad aumentada educativa [25].	El juego educativo es una aplicación para dispositivo Tablet que dispone de varias actividades con niveles de dificultad, permitiendo al niño interactuar con el sistema mediante una serie de fichas de diferentes temáticas para desarrollar la atención, memoria, percepción y razonamiento lógico.

Tabla 4: Juegos móviles de realidad aumentada para la motivación

La RA está siendo utilizada en múltiples escenarios, pero un reto que actualmente se está abordando es la utilización de esta tecnología de forma cooperativa. La realidad aumentada "multipersona" que consiste en la participación de varios usuarios interactuando con los mismos objetos virtuales y al mismo tiempo. Enfoque que puede resultar muy útil e interesante en aquellos juegos multijugador que buscan fomentar actividades socializantes e incentivar la comunicación y colaboración [26]. El presente trabajo es un paso más en esta dirección en el que se plantea la definición de un espacio de juego multijugador mediante RAM sin necesidad de utilizar marcadores fiduciales que limitan su implantación. El dominio tratado en el juego propuesto es el de la identificación y manipulación de estados emocionales básicos.

3. Análisis

La fase de análisis es una de las etapas iniciales del ciclo de vida del software la cual permite definir requisitos, restricciones y diversos modelos que caracterizan la solución a implementar. Por lo mencionado anteriormente se detalla a continuación los requerimientos, las historias de usuario y el diagrama de clases que son la base para la implementación del juego de realidad aumentada enfocado a un escenario multijugador:

3.1. Requerimientos del juego multijugador

La aplicación móvil deberá generar un entorno de juego que permita en los niños identificar y actuar con estados emocionales básicos, estimular su imaginación e incentivar la comunicación y colaboración entre ellos, agregando a esto, una visión de realidad aumentada para añadir información virtual a la información física ya existente. Con el fin de evaluar el impacto de la modalidad de juego colaborativo en la experiencia de usuario el prototipo se desarrollará en dos modalidades: competitivo vs colaborativo.

3.1.1. Generales

El juego admitirá la participación de varios jugadores (mínimo 2) para los dos tipos de entornos, colaborativo y competitivo. En el juego, los niños deberán descubrir personajes que muestren emociones ya sea de enfado, tristeza, alegría, u otros. El objetivo del juego será ser capaces de identificar las emociones mostradas por los personajes y poder actuar sobre las mismas mediante el lanzamiento de objetos que representan acciones que puedan conducir a los personajes a cambiar su estado emocional. Como consecuencia de dichos lanzamientos el personaje en cuestión puede llegar a ser capturado por los participantes en el juego. Esta estrategia de juego ha imitado modelos de juego como el conocido PokemonGo [27] pero forzando a los participantes a identificar y actuar sobre los estados emocionales de los personajes.

En base a las dinámicas de gamificación definidas por Werbach and Hunter [28] se han introducido restricciones en el juego para incrementar su nivel de dificultad. En particular, en el prototipo diseñado los personajes o actores del juego se desplazan por el espacio aumentado 3D al transcurrir un periodo de tiempo de forma que no permanezcan estáticos y sea más difícil su localización y el lanzamiento de objetos sobre los mismos. Adicionalmente los objetos que serán lanzados hacia estos personajes, denominados proyectiles, dispondrán de un solo lanzamiento, por lo cual,

el jugador deberá buscar dichos proyectiles lanzados en el entorno y recuperarlos para disponer nuevamente de ellos. De igual forma los proyectiles cada cierto tiempo serán reubicados en nuevas posiciones, logrando con ello que la lógica del juego se aún más dinámica.

También siguiendo la dinámica de progresión definida por Werbach and Hunter [28] se ha definido un sistema de puntaje en el juego, de forma que los actores y proyectiles dispondrán de un número de puntos. Los puntos de los actores definirán su nivel de vida (100%) y la suma de todos estos darán como resultado los puntos totales que tiene el juego, adicionalmente cada personaje estará definido por los proyectiles que le afectan de forma positiva o negativa en su nivel de vida. Por otra parte, cada proyectil podrá afectar de forma diferente a cada actor ya sea disminuyendo o incremento su nivel de vida con igual o diferentes puntos.

Cuando un personaje llega a su nivel de vida mínimo (0%), este es capturado por el jugador que tiene la mayor cantidad de puntos de dicho personaje comparado a la de sus contrincantes (puntos de daño).

El juego al ser multijugador, se requiere que la posición y el nivel de vida de los personajes este siempre sincronizada con todos los jugadores, con ello se podrá garantizar una escena de juego única. Adicionalmente cada jugador podrá visualizar en el espacio aumentado común el proyectil de su compañero y la ejecución del disparo.

Finalmente, el juego será culminado cuando se hayan capturado todos los personajes, lo que equivale que cada personaje debe llegar a su nivel mínimo de vida.

3.1.2. Competitivo

El juego competitivo estará enfocado a que los jugadores puedan alcanzar una meta, la cual será obtener la mayor cantidad de puntos en función de los disparos que han ejecutado hacia los personajes para su captura. La captura de cada personaje estará definida por aquel jugador que obtiene la mayor cantidad de puntos respecto a los disponibles.

Al finalizar el juego, existirá un ganador que será aquel que obtiene la mayor cantidad de puntos referente a los disponibles en todo el juego o empate en el caso que los puntos sean iguales.

3.1.3. Colaborativo

El juego colaborativo está basado en las seis condiciones definidas por Szewkis et al. [29] para el aprendizaje colaborativo [1]:

- *La existencia de un objetivo común*. Al proporcionar un objetivo común para todos los participantes, es decir, hacer que actúen sobre los mismos elementos del espacio aumentado, el sistema permite las interacciones sociales necesarias para que los alumnos aprendan a través de la colaboración.
- *Coordinación y comunicación entre pares*. Los estudiantes necesitan comunicarse entre sí e interactuar en el orden correcto y en el momento adecuado para resolver el objetivo común.
- *Interdependencia positiva entre pares*. Los estudiantes se sienten conectados entre sí para que sientan que solo pueden tener éxito mientras el resto lo haga. Puede afectar positivamente el aprendizaje colaborativo animando procesos de negociación y fomentando la resolución conjunta de conflictos. La colaboración puede apoyar la interdependencia positiva, mientras que las actividades que son solo competitivas promueven la interdependencia negativa, aquellas que incluyen la cooperación fomenta la interdependencia positiva.
- *Conciencia del trabajo de los compañeros*. Para que la colaboración funcione, todos los participantes deben ser capaces de ver lo que los demás están haciendo en cualquier momento, para que puedan recibir comentarios comunes que respaldarán su proceso de toma de decisiones. Para respaldar esto, el sistema proporciona un espacio aumentado compartido común.
- *La responsabilidad individual*. Para asegurar su contribución, cada estudiante debe ser responsable de sus acciones antes que los demás. Para esto, cada participante debe ser capaz de ver las consecuencias de cualquiera de las acciones de sus pares. Esto también se ve respaldado por la presencia de un espacio aumentado compartido en el que las acciones individuales de los participantes son visibles para los demás.
- *Recompensas conjuntas*. Como todos los miembros del equipo son recompensados o castigados por igual, lo que los hace constantemente tratar de mejorar su colaboración.

En este sentido el juego colaborativo tendrá una regla principal para la captura de los personajes u actores definida como: Un actor puede recibir puntos positivos o negativos solo si recibe el disparo de dos proyectiles en un periodo de tiempo. Si ocurren estos dos disparos y si estos perjudican a este actor, entonces los puntos acumulados de estos dos proyectiles afectarán la vida de este personaje. Mencionando además que los dos disparos deben ser ejecutados de forma colaborativa, lo que significa que deben proceder de diferentes jugadores.

Al finalizar el juego, todos los jugadores tendrán igual número de puntos así como igual número de objetos capturados.

3.2. Historias de usuario

Las historias de usuario son descripciones simples y concretas que detallan las funcionalidades que debe implementar un sistema, es por ello que nos basamos en esta descripción para definir de manera formal los requerimientos del juego, el mismo que integrará dos modalidades, competitivo vs colaborativo. El formato a seguir para la redacción de las historias de usuario consta de las siguiente estructura: “como <rol del usuario>, quiero <característica o función> [de manera que/con el fin de <razón u objetivo>]” [30]:

- *Historia de usuario 1*: Como jugador, quiero poder ingresar ni nombre en el juego de manera que mis compañeros me puedan identificar.
- *Historia de usuario 2*: Como jugador, quiero poder seleccionar el tipo de juego colaborativo o competitivo.
- *Historia de usuario 3*: Como jugador, quiero visualizar los personajes del juego con el fin de saber su posición para capturarlos.
- *Historia de usuario 4*: Como jugador, quiero poder movilizarme por el entorno con el fin de visualizar desde diferentes puntos los personajes del juego.
- *Historia de usuario 5*: Como jugador, quiero visualizar el nivel de vida de los personajes con el fin de saber cuándo incrementa y decremento.
- *Historia de usuario 6:* Como jugador, quiero que los personajes estén sincronizados en posición y nivel de vida con los otros jugadores de manera que estamos observando el mismo escenario.
- *Historia de usuario 7*: Como jugador, quiero que los personajes cada cierto tiempo cambien de posición con el fin de incrementar la dificultad de captura.

- *Historia de usuario* 8: Como jugador, quiero disponer de un panel que muestre los personajes que he capturado.
- *Historia de usuario 9*: Como jugador, quiero disponer de una la lista de proyectiles para poder seleccionar uno de ellos.
- *Historia de usuario 10*: Como jugador, quiero poder seleccionar un proyectil para poder lanzarlo hacia un personaje del juego.
- *Historia de usuario 11*: Como jugador, quiero que el lanzamiento del proyectil sea ejecutado cuando lo colisiono, de manera que dispongo de un acceso rápido a la acción.
- *Historia de usuario 12*: Como jugador, quiero que el lanzamiento del proyectil sea en dirección frontal a la cámara con el fin de poder elegir la trayectoria que deseo ejecutar.
- *Historia de usuario 13*: Como jugador, quiero que los proyectiles contengan puntos positivos o negativos de manera que permitan aumentar o disminuir el nivel de vida de los personajes.
- *Historia de usuario 14*: Como jugador, quiero que cuando un proyectil es lanzado se elimine de la lista de proyectiles disponibles de manera que debo buscarlo en la escena para recuperarlo.
- *Historia de usuario 15*: Como jugador, quiero que los proyectiles de recuperación se ubiquen de forma aleatoria en la escena y se muevan cada cierto tiempo de manera que sus posiciones no son estáticas.
- *Historia de usuario 16*: Como jugador, quiero que los proyectiles de recuperación sean visibles y accesibles solo por el jugador que los disparo con el fin de no generar confusión para los otros jugadores.
- *Historia de usuario 17*: Como jugador, quiero que al topar un proyectil de recuperación se agregue a la lista de proyectiles disponibles de manera que pueda seleccionarlo nuevamente para lanzarlo.
- *Historia de usuario 18*: Como jugador, quiero visualizar en la escena el proyectil de los otros jugadores y su disparo.
- *Historia de usuario 19*: Como jugador quiero visualizar los puntos obtenidos y los puntos totales del juego.
- *Historia de usuario 20*: Como jugador, quiero saber cuándo el juego termina.

- *Historia de usuario 21*: Como jugador (competitivo) quiero saber si él proyectil le afecta de forma positivo, negativo o ninguna acción cuando se impacta con un personaje del juego.
- *Historia de usuario 22*: Como jugador (competitivo) quiero visualizar los puntos obtenidos por cada jugador con el fin de conocer el puesto en el que estoy.
- *Historia de usuario 23*: Como jugador (competitivo) quiero saber si he ganado cuándo termina el juego.
- *Historia de usuario 24*: Como jugador (colaborativo) quiero los personajes incrementen o disminuyan su vida solo si recibe impacto de dos proyectiles instanciados de jugadores diferentes en un límite de tiempo.
- *Historia de usuario 25*: Como jugador (colaborativo) quiero que cuando el primer proyectil se impacta con un personaje que es considerado como objetivo para el proyectil, se inicie un contador de tiempo con el fin de definir el tiempo máximo de colaboración.
- *Historia de usuario 26*: Como jugador (colaborativo) quiero que cuando se inicie el tiempo de colaboración, este se sincronice con los demás jugadores de manera que sepan que la obtención de un personaje de forma colaborativa inicio.
- *Historia de usuario 27*: Como jugador (colaborativo) quiero visualizar la lista de jugadores y los puntos obtenidos de forma colaborativa.

3.3. Diagrama de clases

Basado en los requerimientos e historias de usuario especificadas en el apartado anterior se procedió a diseñar el diagrama de clases que está orientado para que la lógica del juego sea configurable, logrando con ello un juego dinámico en cuanto a los objetos que intervienen en el mismo, como son los actores o personajes y proyectiles. El modelo especificado en la ilustración 1 puede ser implementado en cualquier escenario real, siendo para nuestro estudio la plataforma de desarrollo de juegos Unity.

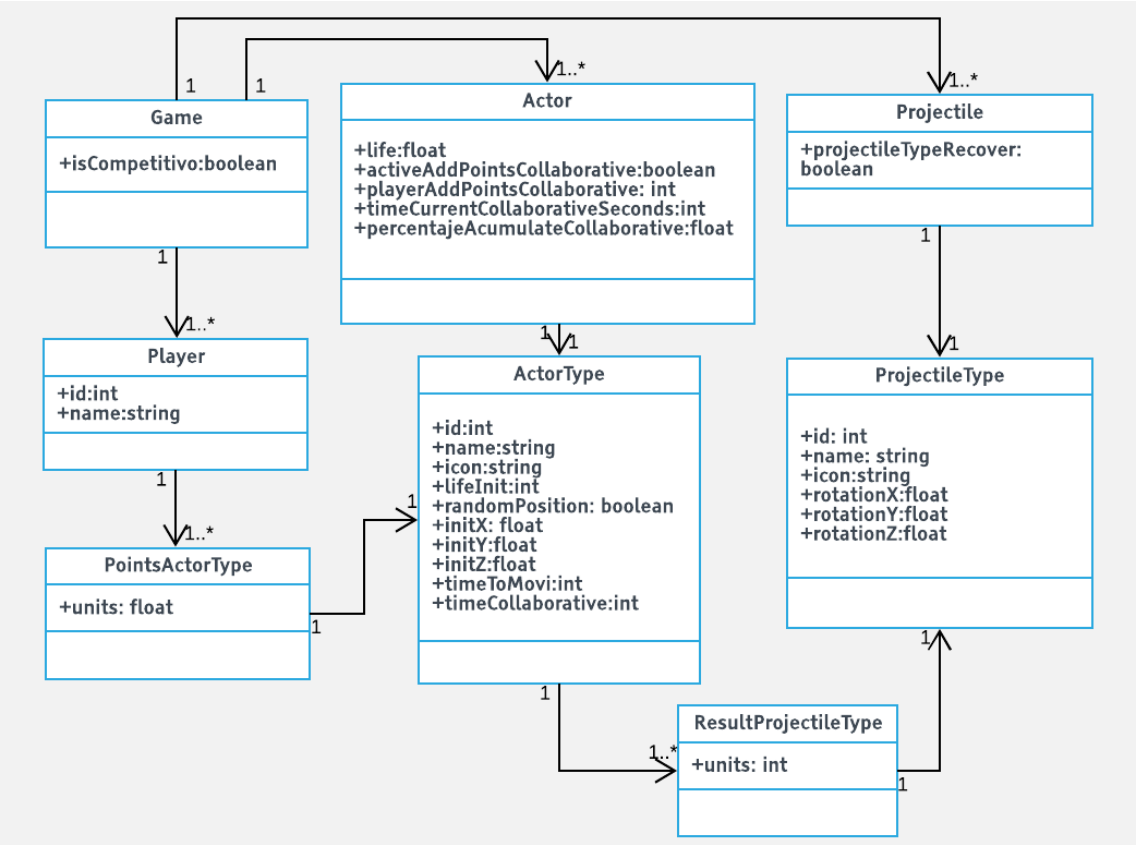


Ilustración 1: Diagrama de clases

A continuación se detalla el propósito de cada clase y sus atributos:

- **Game**: Representa toda la lógica del juego. El juego podrá ejecutarse en la modalidad de competitivo o colaborativo representado por el campo isCompetitivo. Tendrá asociado una lista de actores que son los objetos a capturar y una lista de proyectiles. Adicional, para cada instancia del juego se tendrá la lista de jugadores que están actualmente conectados.
- **Player**: representa a un jugador que tiene como atributos el identificador y su nombre. Cada jugador tendrá una lista de PointsActorType que llevara el control de sus puntos obtenidos por cada ActorType.
- **Actor**: son los objetos a capturar que tiene un nivel de vida representado por el atributo life en una escala de 0 a 1, donde el nivel de 0 representa la captura de personaje. Los campos restantes son utilizados para la modalidad de juego colaborativo:
    - activeAddPointsCollaborative: permite saber si el actor esta lista para la captura de forma colaborativa.

- playerAddPointsCollaborative: almacena el jugador que inicia la colaboración para controlar que la captura del personaje se realice de forma colaborativa.
- timeCurrentCollaborativeSeconds: temporizador del tiempo de colaboración para la captura del personaje.
- percentajeAcumulateCollaborative: almacena los puntos para al final de la colaboración sumar o restar la vida del personaje.

- **ActorType**: cada Actor tiene asociado un actorType definido por los siguientes atributos:
  - id: identificador del actorType
  - name: nombre del personaje
  - lifeInit: el número de puntos que tiene el personaje y corresponde al 100% de su nivel de vida.
  - randomPosition: define si el actor al inicio del juego debe ubicarse en una posición aleatoria.
  - initX, initY, initZ: si la posición no es aleatoria, se especifica en estos campos los valores del eje X, Y y Z que definen la posición.
  - icon: icono del personaje.
  - timeToMovi: define el tiempo en que debe moverse el personaje de su posición.
  - timeCollaborative: define el tiempo límite de colaboración para la captura del personaje.
  - ResultProjectileType: define la lista de proyectiles que pueden afectar al personaje y los puntos de daño.
- **Projectile**: Un proyectile puede ser de un solo tipo, de lanzamiento o de recuperación definido por el campo projectileTypeRecover. Si el valor es verdadero significa que el proyectil es de tipo recuperación.
- **ProjectileType**: cada Projectile tiene asociado un projectileType definido por los siguientes atributos:
  - id: identificar del projectileType
  - name: nombre del proyectil
  - icon: icono del proyectil
  - rotationX, rotationY, rotationZ: si se requiere que el proyectil se instancie con una rotación específica, caso contrario los campos tendrán por defecto el valor de 0.

Con este modelo lo que se consigue es que la lógica del juego sea parametrizable identificando fácilmente para el caso del competitivo los objetos capturados, los puntos totales obtenidos así como el ganador del juego. Para el caso del colaborativo los objetos capturados así como el puntaje total obtenido serán siempre igual para todos los jugadores.

4. Diseño e Implementación

Las especificaciones obtenidas en la fase de análisis son guías maestras que permiten iniciar con el desarrollo de una aplicación, es por ello que tanto los requerimientos, las historias de usuario y el modelo conceptual (ver capítulo 3) fueron la base para dar inicio con el proceso de diseño e implementación ya que detallan de forma clara el escenario de juego que se desea ejecutar. A continuación damos a conocer la tecnología utilizada, la interfaz de usuario, así como los componentes principales que se integran para tener como resultado un juego móvil multiplayer con realidad aumentada.

4.1. Tecnología utilizada

Nuestro juego está construido bajo dos tecnologías principales, Unity como plataforma de desarrollo de videojuegos y el SDK Placenote de realidad aumentada.

4.1.1. Motor de videojuegos

Unity es un motor de videojuegos multiplataforma que tiene soporte de compilación con diferentes tipos de plataformas como PC (Windows, GNU/Linux, OS X), dispositivos móviles (iOS, Android), Smart TV (tvOS, Android TV), dispositivos de realidad virtual (Oculus Rift, PlayStation VR) y actualmente es una de las competencias tecnológicas de mayor demanda [31]. Es por ello, que nuestro juego será desarrollado en Unity con la versión 2017.3.1.

4.1.2. Infraestructura de Red y Comunicación

Un juego enfocado a un escenario multijugador tiene que implementar una infraestructura de red para la comunicación entre los jugadores así como la sincronización de todos los objetos que intervienen en el mismo. Un paquete de Unity que implementa dicha infraestructura es Photon Unity Networking (PUN).

PUN permite implementar juegos multijugador en red y dispone de métodos de comunicación rápida que aseguran que todos los jugadores puedan encontrarse y jugar entre ellos, está basado en salas para que los objetos pueden sincronizarse y dispone de Remote Procedure Calls (RPC) que implica la llamada de métodos en clientes remotos [32].

Para disponer del paquete en nuestra aplicación de Unity, se debe obtener una AppId de los servidores de PUN y con ello proceder a registrarla en el campo de entrada

cuando se instala el paquete o en el archivo PhotonServerSettings posterior a su instalación.

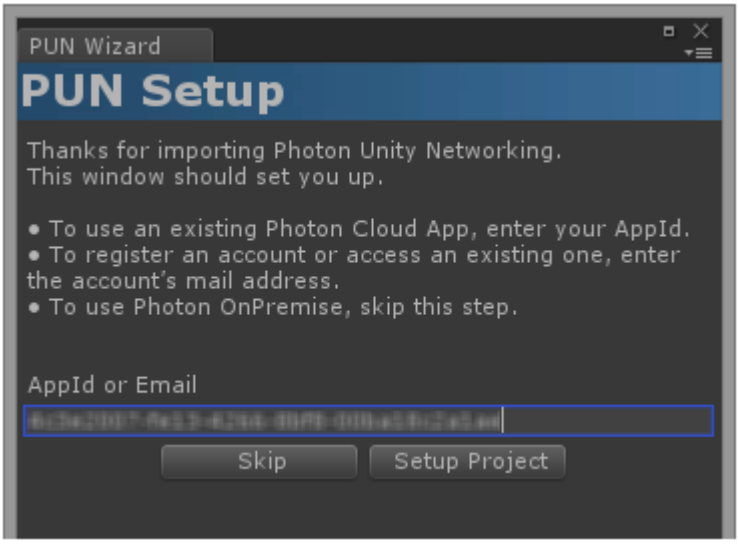


Ilustración 2: Photon Unity Networking (PUN)

Dado que Unity dispone de Pun, que es una solución para implementar un juego multijugador, este será utilizado para la comunicación entre los jugadores y la sincronización de los objetos (actores, proyectiles).

4.1.3. Plataforma de realidad aumentada

El SDK Placenote al igual que otras plataformas de realidad aumentada permite crear aplicaciones que colocan contenido digital a información física del mundo real. Una característica particular del Placenote es que no necesita GPS o marcadores para la geolocalización por lo que ambas alternativas tiene sus limitaciones respecto al espacio físico (ver capítulo 1.1). En su lugar, utiliza un sistema avanzado de visión por computadora que proporciona una capa de mapeo y posicionamientos visuales persistentes, lo que permite que podamos escanear cualquier ubicación física y convertirla en un plano para posicionar con precisión el contenido de realidad aumentada [33].

El SDK funciona en la nube y como requisito para integrarlo en una aplicación local se necesita crear una cuenta de desarrollador en el portal de Placenote a fin de obtener una clave de API.

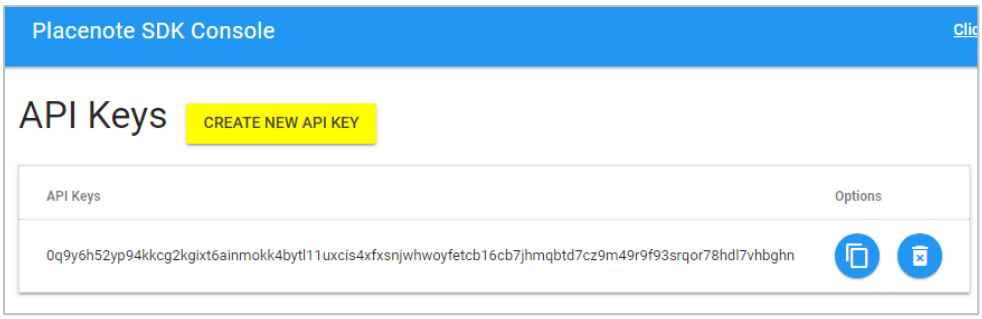


Ilustración 3: Placenote

Una limitación de la plataforma lanzada al mercado recientemente, es que solo está disponible para el desarrollo de iOS con las plataformas Unity o Native Scenekit, sin embargo los creadores anunciaron que el soporte de Android llegará pronto.

A pesar de su limitación multiplataforma, tiene una característica muy relevante para nuestro caso de estudio, que es poder construir una aplicación de realidad aumentada con geolocalización sin GPS o marcadores. En nuestro juego, Placenote permitirá que las posiciones de los objetos virtuales (actores y proyectiles) relativas al mundo real sean siempre persistentes en todos los jugadores.

4.1.4. Plataforma hardware de Destino

Al utilizar Placenote para la realidad aumentada, la plataforma de destino para nuestro juego es iOS 11 o superior.

4.2. Diseño de la Interfaz de usuario

El juego multijugador con realidad aumentada cumple con todos los requerimientos especificados en el capítulo 3.1 y para su demostración, a continuación se da a conocer la interfaz junto a las historias de usuario a las que se hace hincapié (ver capítulo 3.2).

Configuración del juego

El juego móvil dispone de una interfaz principal que permite registrar datos informativos y de configuración como, el nombre del jugador (1), tipo de modalidad a ejecutar (2) y selección del mapa (3), para posterior proceder a iniciar el juego (4). La lista de mapas son obtenidos del servidor Placenote (5).

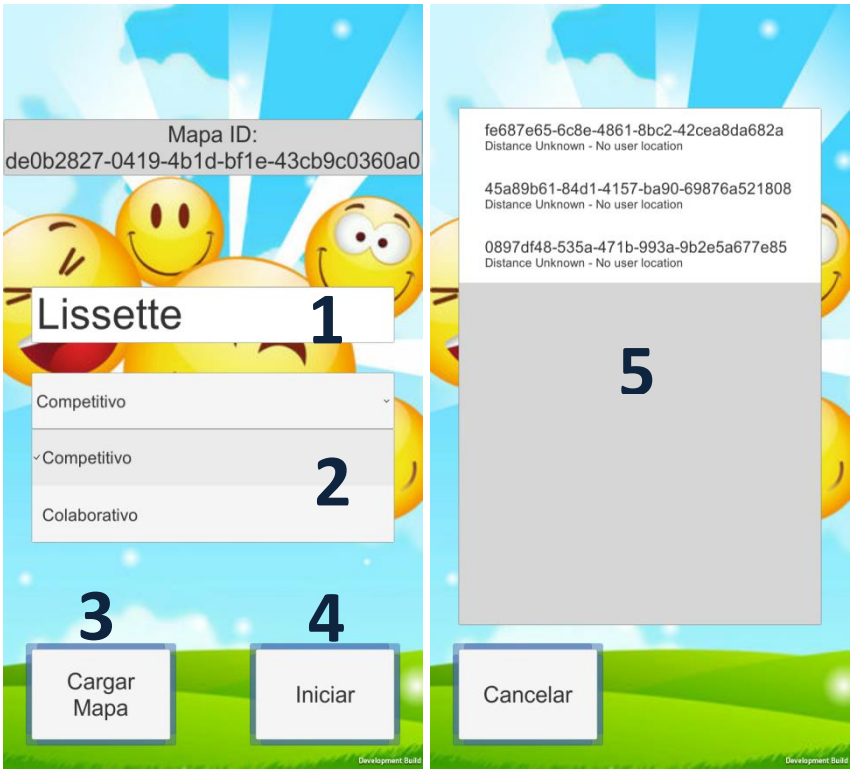


Ilustración 4: Interfaz gráfica, pantalla inicial

Como paso previo a la selección del mapa, se debe escanear el espacio físico de juego (habitación, aula) y almacenarlo en los servidores del Placenote. El SDK proporciona un app gratuita que dispone de las funcionalidades para la creación y carga de mapas en sus servidores[3], lo que permite que nuestra aplicación solo consuma su API para obtener y seleccionar el mapa.

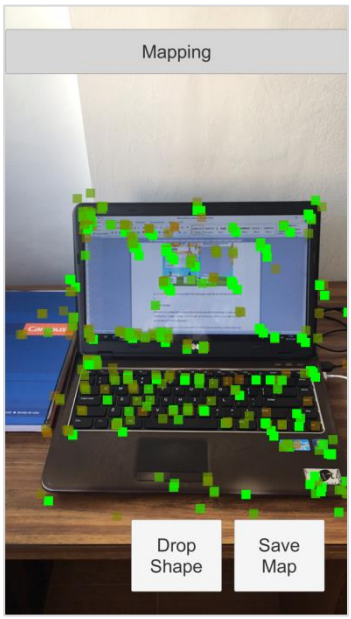


Ilustración 5: Interfaz gráfica, app de Placenote

[3] https://placenote.com/install/

La ilustración 4 representa la pantalla inicio del juego y satisface las historias de usuario 1 y 2.

Escena del juego

Posterior a la configuración se procede a iniciar el juego, lo que equivale a nivel de implementación a cargar el mapa en la escena y unirse a una sala de juego gestionada por PUN (multijugador).

Los personajes a capturar son instanciados en red de acuerdo al archivo configuración (ver capítulo 4.3.1) y hacen referencia a los prefabs de tipo actor (ver capítulo 4.3.3.1). Estos personajes están ubicados en cualquier punto del entorno físico del juego (puntos del mapa), lo que permite que los jugadores con la cámara puedan acercarse al personaje y moverse en el entorno a fin de visualizar desde diferentes perspectivas a estos objetos 3D (1).

La interfaz dispone de un menú de proyectiles (1) de acuerdo al archivo configuración (ver capítulo 4.3.1), un botón que accede a la interfaz para mostrar información de los jugadores (2), puntos obtenidos del juego (3) y personajes capturados por el jugador (4).

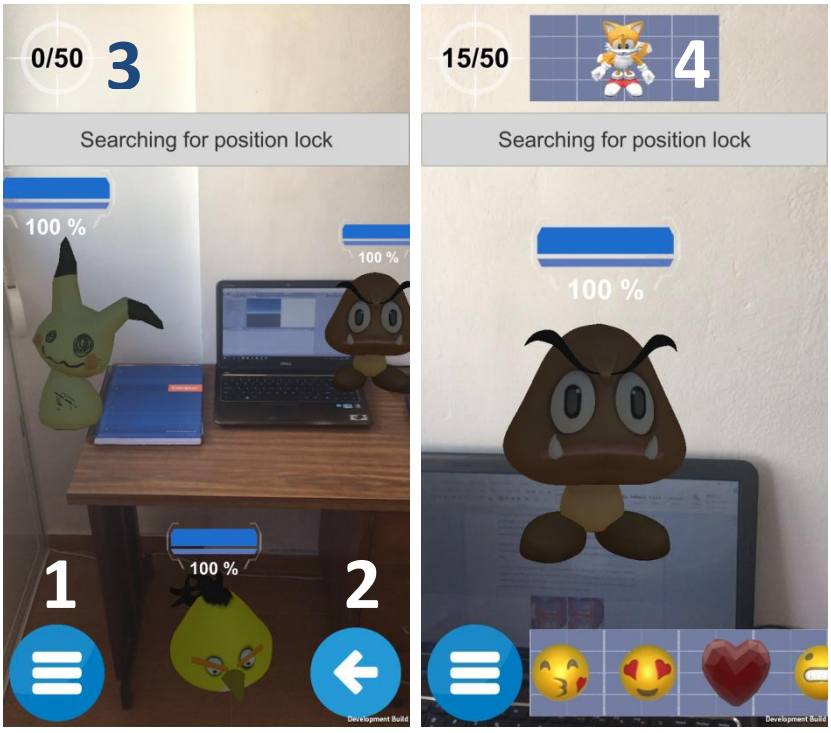


Ilustración 6: Interfaz gráfica, escena del juego

La ilustración 6 satisface las historias de usuario 3, 4, 8, 9 y 19.

Personajes a capturar

Los personajes al inicio del juego se posicionan en un punto ya se aleatorio y o no, especificado en el archivo configuración (ver capítulo 4.3.1). Al transcurrir el tiempo, cada personaje se mueve a un punto aleatorio del espacio físico del juego, donde, el límite de tiempo para que se mueva está definido por el archivo configuración.

Cada personaje tiene asociado una interfaz 2D, que muestra el nivel de vida actual, valor que se reduce o aumenta solo cuando recibe el impacto de un proyecto y si este produce daño.

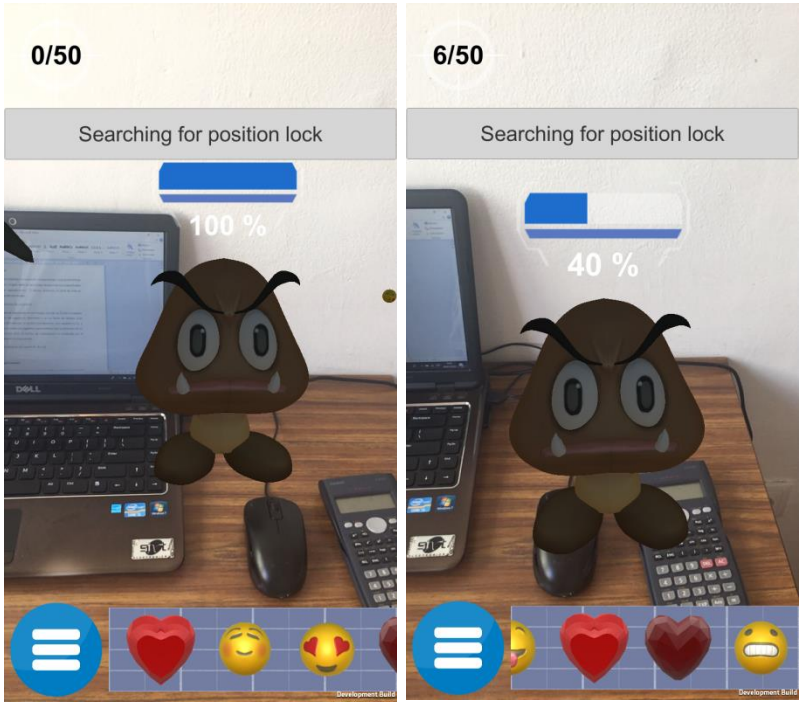


Ilustración 7: Interfaz gráfica, nivel de vida del personaje

La creación de los personajes y su movimiento es gestionado por el jugador master (jugador que entra primero a la sala de juego), y este se encarga de sincronizar con los jugadores restantes, con lo cual tanto las posiciones y el nivel de vida de cada personaje siempre serán iguales en todas las escenas de los jugadores.

La ilustración 7 satisface las historias de usuario 5, 6 y 7.

Proyectiles a lanzar

El jugador al seleccionar un proyectil del menú proyectiles (1), este se instancia en red (en todos los jugadores) como un objeto 3D que hace referencia a los prefabs de tipo proyectil (ver capítulo 4.3.3.2).

El proyectil se posicionado en dirección central a la cámara (2), sigue el movimiento de la cámara y es lanzado en dirección a donde está apuntando la cámara (3), lanzamiento que inicia cuando el jugador topa el proyectil. Cada proyectil solo puede ser lanzado por el jugador que lo creo, sin embargo los jugadores restantes pueden ver el proyectil y su disparo.

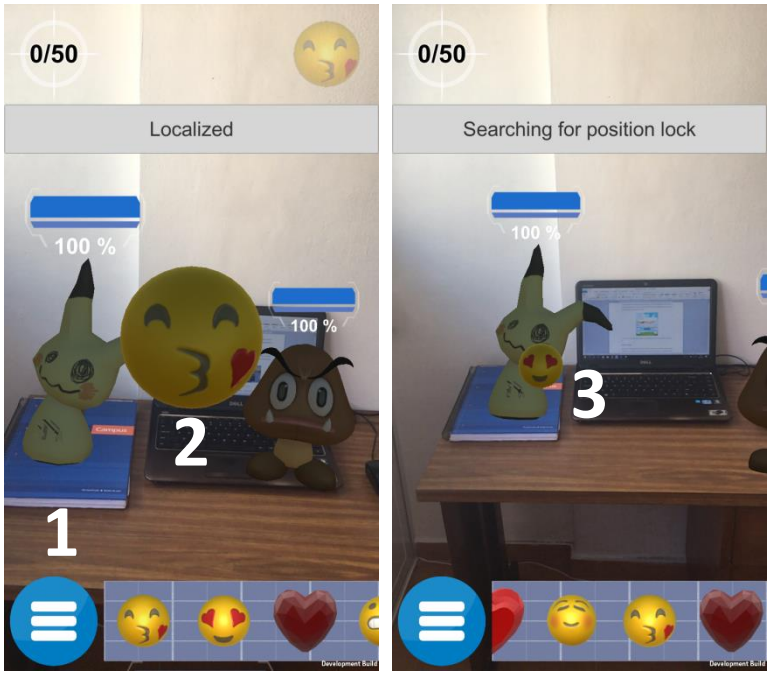


Ilustración 8: Interfaz gráfica, selección y lanzamiento del proyectil

La ilustración 8 satisface las historias de usuario 10, 11, 12 y 18.

Cada proyectil al colisionar con un personaje, produce un daño de acuerdo a los especificado en el archivo configuración (ver capítulo 4.3.1). El lanzamiento de un proyectil lo elimina de la lista de proyectiles, que son todos aquellos disponibles para ejecutar un disparo (1). Esta acción hace que el jugador local disponga de su lista de proyectiles a recuperar (2), proyectiles que son ubicados de forma aleatoria en el espacio físico del juego con cambio de posición cada cierto tiempo (3). El jugador para recuperarlos con ayuda de la cámara tiene que buscar a cada objeto (4) y colisionarlo (toparlo), lo que

significa que vuelven a estar disponibles en el menú. Los proyectiles de tipo recuperación solo son visibles y accesibles por el jugador local.

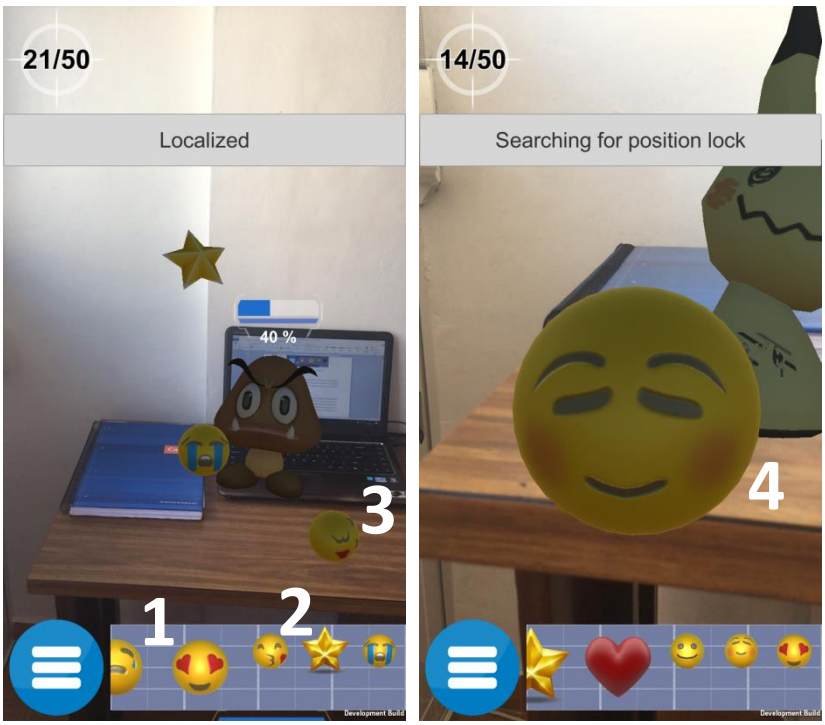


Ilustración 9: Interfaz gráfica, proyectiles de tipo recuperación

La ilustración 9 satisface las historias de usuario 13, 14, 15, 16 y 17.

Daño Competitivo y Colaborativo

En el juego competitivo cuando se impactan un proyectil y un personaje, el proyectil afecta con puntos positivos, negativos o ningún daño al personaje de acuerdo a lo especificado en el archivo configuración (ver capítulo 4.3.1). Si afecta, entonces el nivel de vida es actualizado en la interfaz gráfica del personaje (ver ilustración 7), satisfaciendo la historia de usuario número 21.

En el juego competitivo la lógica de captura de un personaje consiste en recibir el impacto de dos proyectiles originados de jugadores diferentes (1) y en un límite de tiempo (2). Este tiempo de colaboración está definido por el archivo configuración (ver capítulo 4.3.1), y es inicializado y sincronizado en todos los jugadores para notificar que la obtención de un personaje de forma colaborativa inicio. El tiempo de colaboración es inicializado por el primer proyectil que se impacta en el personaje.

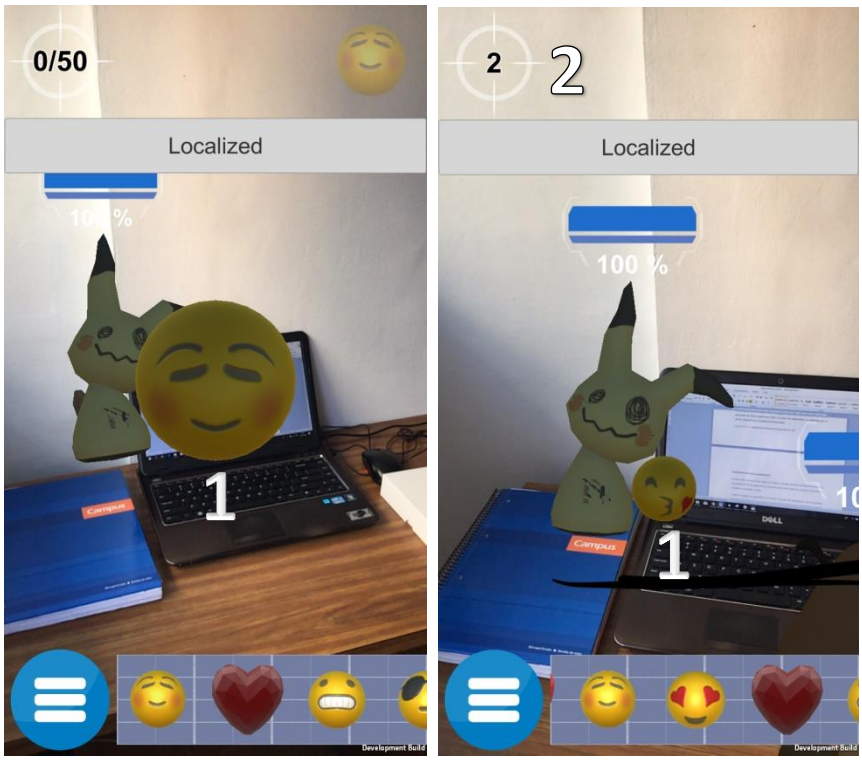


Ilustración 10: Interfaz gráfica, tiempo colaborativo

La ilustración 10 satisface las historias de usuario 24, 25 y 26.

Finalización del juego y participantes

Cuando todos los personaje, llegan a su vida 0, el juego termina mostrando una interfaz que detalla la lista de jugadores (1), los personajes capturados (2), así como la mención si el jugador es ganador o no(3).

Cuando el juego es competitivo, la interfaz muestra los participantes con sus puntos obtenidos (4), mientras que cuando es colaborativo, todos ganan, mostrando los puntos totales obtenidos por la colaboración (5).

La interfaz de usuario definida en la ilustración 11 puede ser invocada en cualquier momento del juego (ver ilustración 6) y satisface las historias de usuario 20, 22, 23, 27.

Ilustración 11: Interfaz gráfica, información del juego

Finalmente la aplicación tiene un sonido principal que se reproduce durante todo el juego y otros efectos musicales cuando se realiza acciones como: lanzar un proyectil, impacto del proyectil con un personaje, daño positivo y negativo del personaje, inicio de la colaboración, sonidos que son complemento para captar la atención de los niños.

4.3. Diseño e Implementación de la lógica del juego multijugador en Unity

Nuestro juego en Unity esta creado en una escena llamada Conexión, la cual implementa la interfaz de usuario diseñado en un Canvas y diversos GameObjects: PlacenoteCameraManager, PlacenoteMap, MainThreadTaskQueue que son prefabricados del SDK Placenote, GameObjectConfig, GameObjectRoom, GameObjectGame que manejan funcionalidades generales del juego y el GameObjectAudioGame para la reproducción del audio principal del juego.

Ilustración 12: Escena Conexión

La siguiente figura proporciona un mayor detalle de la estructura interna de las clases y componentes del sistema de juego.

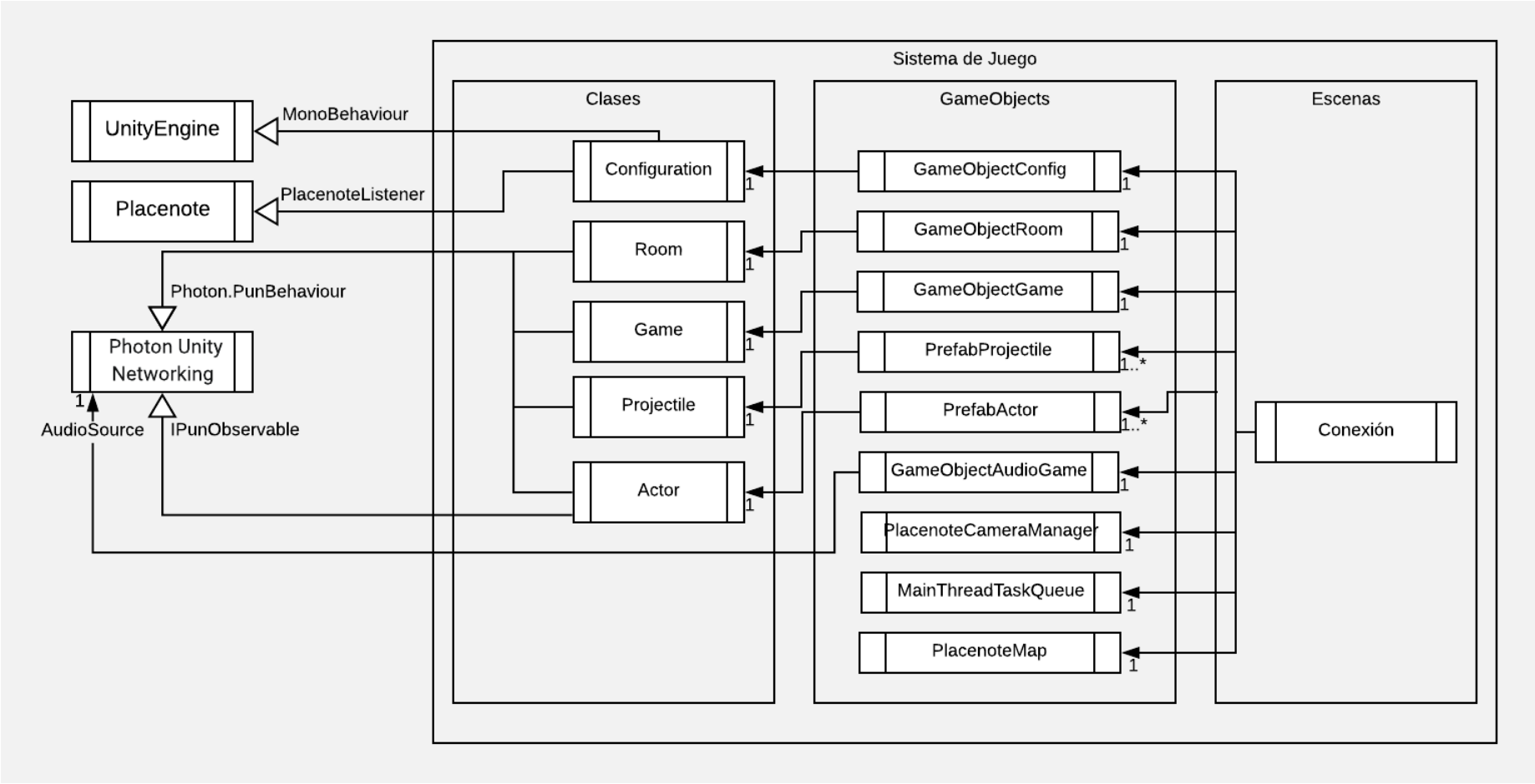


Ilustración 13: Diagrama de clases y componentes del Sistema de Juego

Los componentes PlacenoteCameraManager, PlacenoteMap y MainThreadTaskQueue permiten incorporar en nuestro juego el escenario de realidad aumentada. Una acción obligatoria que se realizó en el prefabricado PlacenoteCameraManager fue asignar al componente LibPlacenote nuestra clave de desarrollador (ver capítulo 4.1.3).

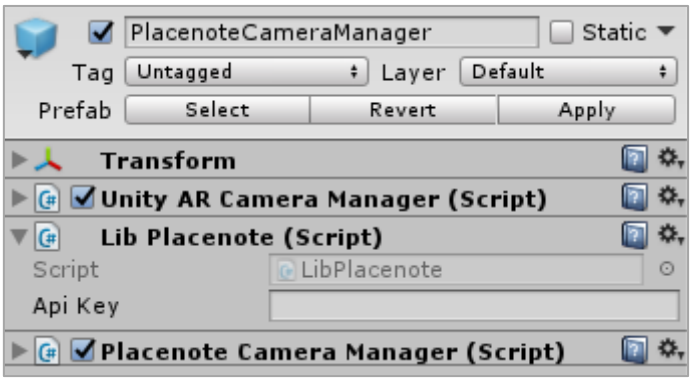


Ilustración 14: Configuración Placenote

Debido que nuestra aplicación utiliza PUN para el juego multijugador se procedió a obtener un AppId para registrarlo en el componente PhotonServerSettings (ver capítulo 4.1.2).

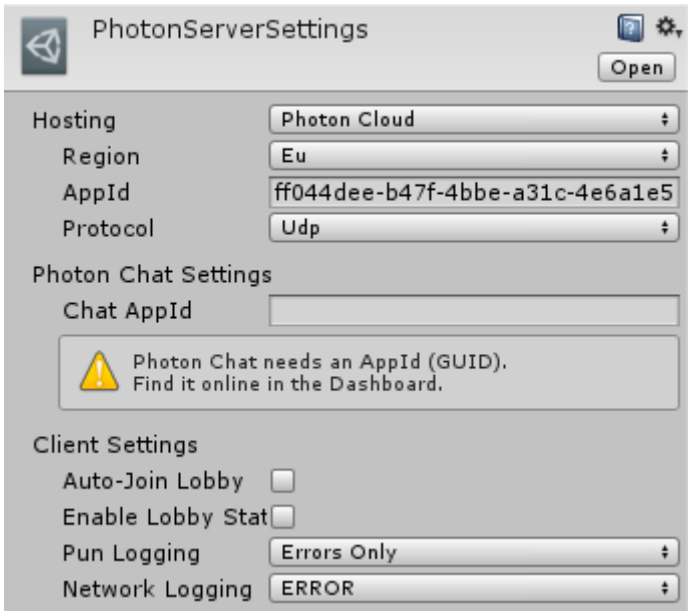


Ilustración 15: Configuración PUN

4.3.1. Configuración inicial del juego

El diagrama de clases definido en la fase de análisis tiene como principal objetivo permitir que la lógica del juego sea totalmente configurable a fin de producir

ambientes de juego diferentes, por ejemplo cuidado personal, hábitos alimenticios, emociones, etc., siendo el de emociones el escenario para nuestro caso de estudio.

Para habilitar dicha parametrización, se creó un archivo de inicialización llamado configuracion.json (ver anexo 1) a fin permitir al usuario almacenar diversas opciones de juegos, donde cada juego debe especificar su configuración con la siguiente estructura:

```
{"games": [
  {
    "id": 1,
    "name": "emociones",
    "actorsType": [
      { "id": 1,
        "name": "emociones_actor_enojado_01",
        "lifeInit": 10,
        "randomPosition": true,
        "icon": "emociones_actor_enojado_01",
        "timeToMovi": 20,
        "timeCollaborative": 10,
        "resultsType": [
          {"projectileType": 1,"units": 8},
          {"projectileType": 2,"units": 2}
        ]
      },
      { "id": 2,
        "name": "emociones_actor_enojado_02",
        "lifeInit": 10,
        "randomPosition": true,
        "icon": "emociones_actor_enojado_02",
        "timeToMovi": 20,
        "timeCollaborative": 10,
        "resultsType": [
          {"projectileType": 1,"units": 8},
          {"projectileType": 2,"units": 4}
        ]
       }
    ],
    "projectilesType": [
      { "id": 1,
        "name": "emociones_projectile_corazon_rojo",
        "icon": "emociones_projectile_corazon_rojo",
        "rotationX": 0,"rotationY": 0,"rotationZ": 0},
      { "id": 2,
        "name": "emociones_projectile_sonrisa_cubo",
        "icon": "emociones_projectile_sonrisa_cubo",
        "rotationX": 0,"rotationY": 0,"rotationZ": 0}
  }
 ]
}
```

Ilustración 16: Configuración inicial del juego

Cada juego definido en el archivo configuración debe contemplar los siguientes atributos:

- **id**: identificador para acceder a su configuración.
- **name**: nombre descriptivo del juego.
- **actorsType**: contiene la lista de personajes o actores que deberán ser capturados por el jugador. Cada actorType tiene su id, name, lifeInit, randomPosition, icon, timeToMovi, timeCollaborative y una lista resultsType que contiene los proyectiles que afectaran la vida de este actorType, donde cada ítem contiene el id del projectilType y units que son los puntos de daño, positivos o negativos.
- **projectilesType**: contiene la lista de proyectiles que pueden ser lanzados a los personajes. Cada projectileType contiene el id, name, icon, rotationX, rotationY y rotationZ.

El archivo es leído al iniciar la aplicación y de acuerdo al juego seleccionado se construye la lista de personajes y proyectiles, dando así al usuario la libertad de diseñar y configurar el juego según el escenario que desea ejecutar. En este caso, la ilustración 16 contiene parte de la configuración de nuestro juego enfocado a las emociones.

Finalmente, la organización de datos del archivo de configuración se rige bajo la estructura de nuestro modelo conceptual, el cual está dado por las siguientes clases:

```
public class ProjectileType{
    public int id;
    public string name;
    public string icon;
    public float rotationX;
    public float rotationY;
    public float rotationZ;
 }

public class ActorType{
    public int id;
    public string name;
    public int lifeInit;
    public bool randomPosition;
    public float initX;
    public float initY;
    public float initZ;
    public string icon;
    public int timeToMovi;
    public int timeCollaborative;
    //<id_ProjectileType, units>
    public Dictionary<string, int> resultsProjectileType;
 }
```

Ilustración 17: Configuración del juego, clases

4.3.2. GameObjects implementados

Se crearon 4 GameObjects en la escena conexión con el fin de separar las funcionalidades relacionadas a Placenote e interfaz de configuración, PUN e interfaz del juego, lógica general del juego y la reproducción del audio. A continuación se detalla por cada ítem algunas funciones principales.

GameObjectConfig: El GameObject tiene asignado el script Configuración (ver anexo 2) el cual configura la escena de juego con Placenote (ver capítulo 4.1.3) y maneja la interfaz de usuario del panel panelConfig.

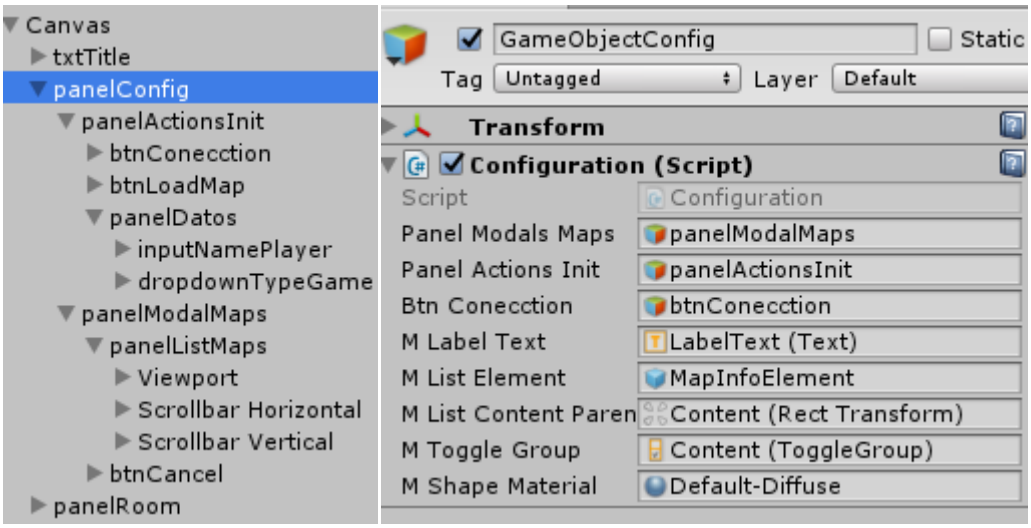


Ilustración 18: GameObjectGame

Algunas características y funcionalidades principales que implementa el script son:

- Hereda de la interfaz PlacenoteListener de Placenote para iniciar sesión y acceder a los mapas que fueron previamente guardados en su API.
- OnLoadMapClicked(), función que descarga el mapa seleccionado e invoca al método Room.roomInstance.conecction() que es el encargado de inicializar el juego multijugador con PUN (script Room).
- getPointRandomMap(), función que es invocada desde otros scripts para retornar un punto aleatorio del mapa que está actualmente cargado.

GameObjectRoom: El GameObject tiene asignado el script Room (ver anexo 3), el cual gestiona la configuración de PUN (ver capítulo 4.1.2) para propiciar un juego multijugador y controla la interfaz de usuario del panel panelRoom.

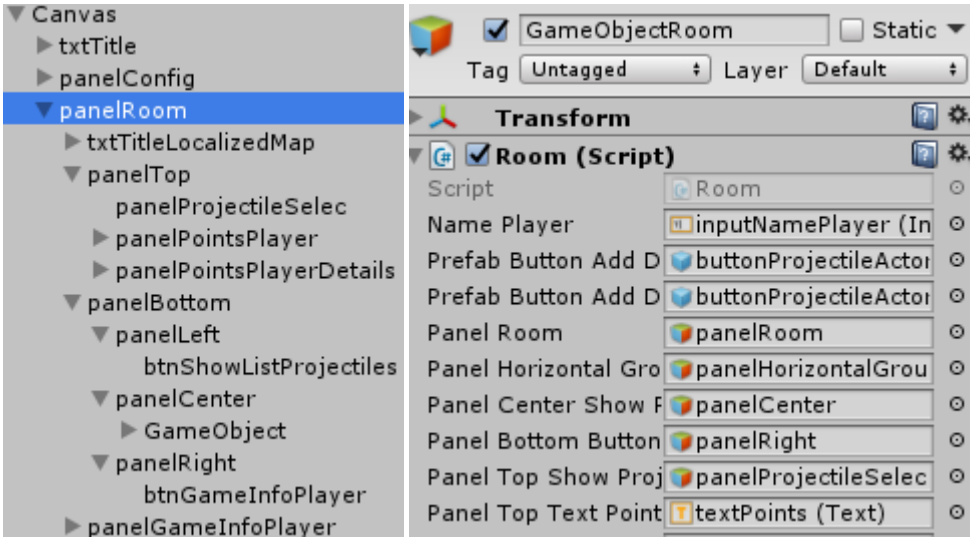


Ilustración 19: GameObjectRoom

Algunas características y funcionalidades principales que implementa el script son:

- Hereda de la clase Photon.PunBehaviour de PUN para conectarse con el servidor, crear una sala de juego si no existe así como unirse y salir de una sala de juego ya existente.
- OnJoinedRoom(): función de PUN que se ejecuta cuando un jugador ingresa a una sala de juego e invoca al método Game.gameInstance.startGame () definida en el script Game. Cuando un jugador se conecta al servidor PUN para unirse a una sala de juego y no existe, PhotonNetwork se encarga de gestionar su creación. El room actualmente está definido para un límite de 4 jugadores.

```
PhotonNetwork.CreateRoom(null, new RoomOptions() { MaxPlayers
= this.maxPlayersPerRoom }, null);
```

Ilustración 20: PhotonNetwork, créate Room

- Update(): la función Update es una función que está definida para ser ejecutada constantemente. Por ello esta función implementa las siguientes funcionalidades:
    - Escucha constantemente si el usuario lanza un rayo y si este impacta a un objeto de tipo proyectil (cuando el usuario topa el proyectil). Si el proyectil es de tipo recuperación o disparo entonces se invocan los métodos necesarios para que el usuario pueda recuperarlo o lanzarlo.
    - Verifica constantemente si el juego termina.

- Dispone de funciones referentes a la interfaz de usuario como habilitar y deshabilitar los paneles de proyectiles y actores, mostrar el puntaje del jugador, mostrar la lista de jugadores, otros.

GameObjectGame: El GameObject tiene asignado el script Game (ver anexo 4) el cual propicia todas las funciones para iniciar, controlar y finalizar el juego.

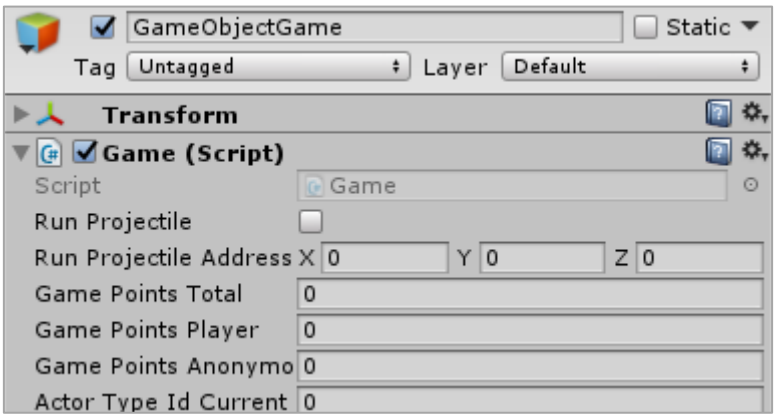


Ilustración 21: GameObjectGame

Algunas características y funcionalidades principales que implementa el script son:

- El script hace referencia a la clase Game de nuestro modelo (ver ilustración 1).
- Hereda de la clase Photon.PunBehaviour de PUN para obtener la lista de jugadores de la habitación e instanciar en Red los personajes y proyectiles.
- startGame (): función principal del script, el cual es invocado cuando un jugador ingresa a la sala de juego. Este método se encarga de inicializar las variables, cargar la lista de jugadores actuales (List<Player> players) y crear la escena del juego.

```
public class Player{
         public int id;
         public string name;
         //<id_ActorType, units>
        public Dictionary<int, float> pointsActorType;
}

foreach (PhotonPlayer newPlayer in PhotonNetwork.playerList){
  Player player = new Player ();
  player.id = newPlayer.ID;
  player.name = newPlayer.NickName;
  player.pointsActorType = new Dictionary<int, float>();
  players.Add (player);
}
```

Ilustración 22: función startGame

- getConfigurationGame(int gameId): método que se encarga de leer el archivo configuration.json y cargar el juego cuyo id sea igual al parámetro gameId. Al cargar el juego, se dispone de una lista de Personajes y Proyectiles. La lista de personajes es utiliza para instanciar con PUN los prefabs cuando se inicia el juego mientras que la lista de proyectiles es utilizada para construir el menú de proyectiles.

```
List<ProjectileType> projectilesType;
List<ActorType> actorsType;
```

Ilustración 23: Lista personajes y proyectiles

- createConfigurationInstantiate(): en la sala de juego se crea en red los prefabs de tipo personaje o actor (ver capítulo 4.3.3.1) de acuerdo a los datos que fueron previamente cargados (actorsType). Estos personajes 3D son comunes para todos los jugadores, por ende se crean como propiedad de la sala del juego. En PUN para que los objetos 3D sean propiedad de la sala, deben ser creados por el jugador Master, que corresponde al primer jugador que ingresa a la sala. Para los jugadores restantes, el jugador master es aquel que se encarga de la sincronización de estos objetos.

```
if (PhotonNetwork.isMasterClient){
  for (int i = 0; i < actorsType.Count; i++){
    float x, y, z;
    Vector3 position;

    if (actorsType[i].randomPosition){
      position = Configuration.configurationInstance
                 .getPointRandomMap ();
    }else{
      x = actorsType[i].initX;
      y = actorsType[i].initY;
      z = actorsType[i].initZ;
      position = new Vector3(x, y, z);
    }

    object[] instantiationData = new object[1];
    instantiationData[0] = actorsType[i].id;
    GameObject goActor = PhotonNetwork.InstantiateSceneObject(
                         actorsType[i].name, position,
                         Camera.main.transform.rotation,
                         0, instantiationData);
    goActor.GetComponent<Actor>().actorType = actorsType[i];

  }
```

Ilustración 24: función createConfigurationInstantiate

Un GameObject de tipo personaje creado con PhotonNetwork.InstantiateSceneObject hace referencia en nuestro modelo conceptual a la clase Actor (goActor.GetComponent<Actor>) el cual tiene asociado un actorType.

- showProjectilePlayer(): se crea en Red el prefab de tipo proyectil (ver capítulo 4.3.3.2) y se ubica en una posiciona central a la cámara. El objeto proyectil es creado con PhotonNetwork.Instantiate lo que significa que solo puede ser manipulado (disparado) por el jugador que lo creó, sin embargo los jugadores remotos pueden visualizar su posición y disparo.

```
GameObject projectileSelecObject = PhotonNetwork.Instantiate(
projectileType.name, positionProjectile,
rotationProjectile, 0, instantiationData);
projectileSelecObject.GetComponent<Projectile>().projectileType
= projectileType;
projectileSelecObject.GetComponent<Projectile>().projectileTypeR
ecover = false;
```

Ilustración 25: función showProjectilePlayer

Un GameObject de tipo proyectil hace referencia en nuestro modelo conceptual a la clase Projectile (projectileSelecObject.GetComponent<Projectile>) el cual tiene asociado el projectileType que es de tipo disparo (projectileTypeRecover = false).

- runProjectilePlayer(): lanza el proyectil en dirección frontal a la cámara y se invoca al método createProjectileTypeRecover() para crear en la escena de juego un nuevo proyectil de tipo recuperación.

```
GameObject proyectileTypeRecoverObj =   Instantiate(
  Resources.Load(proyectileTypeRecover.name, typeof(GameObject)),
  position, Camera.main.transform.rotation) as GameObject;
proyectileTypeRecoverObj.GetComponent<Projectile>().projectileType
= proyectileTypeRecover;
proyectileTypeRecoverObj.GetComponent<Projectile>().projectileTypeR
ecover = true;
```

Ilustración 26: fución createProjectileTypeRecover

El GameObject proyectil de tipo disparo también hace referencia en nuestro modelo conceptual a la clase Projectile.

GameObjectAudioGame: Implementa el componente Audio Source de Unity para reproducir el audio superMarioBrosTemaPrincipal cuando se inicia el juego.

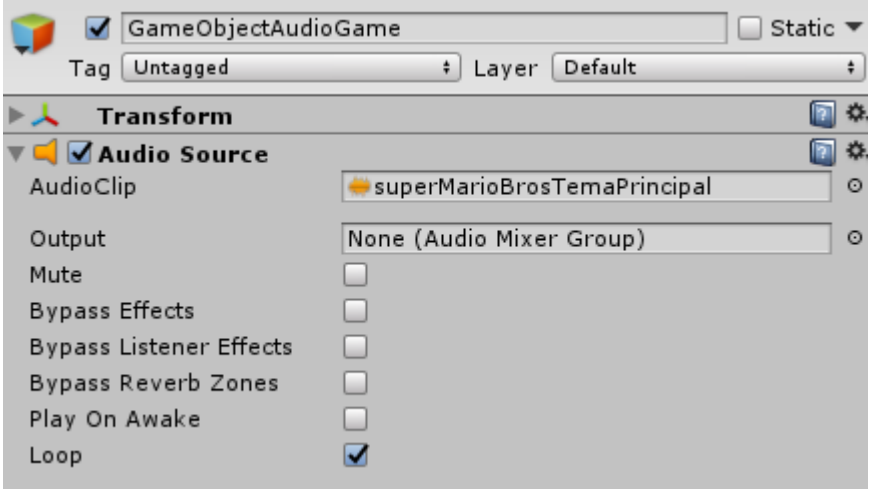


Ilustración 27: GameObjectAudioGame

4.3.3. Prefabs implementados

El archivo de configuración contiene dos objetos principales de juego, como son los actores y proyectiles con los que el jugador debe interactuar. Estos objetos 3D son creados en tiempo de ejecución (script Game, ver anexo 4) y por ende su inicialización está dada por prefabs de Unity que son GameObjects predefinidos y que pueden ser reutilizables durante todo el juego.

De acuerdo a nuestro juego de emociones, se detalla a continuación los prefabs creados en Unity y que están definidos en el archivo configuración.json.

4.3.3.1. Actor

Para nuestro juego se definieron 6 actores que representan un tipo específico de emoción, dos enojados (emociones_actor_enojado_01, emociones_actor_enojado_02), dos tristes (emociones_actor_triste_01, emociones_actor_triste_02) y dos alegres (emociones_actor_feliz_01, emociones_actor_feliz_02), donde cada uno tiene un nivel de vida representado en una escala de 0 a 1.

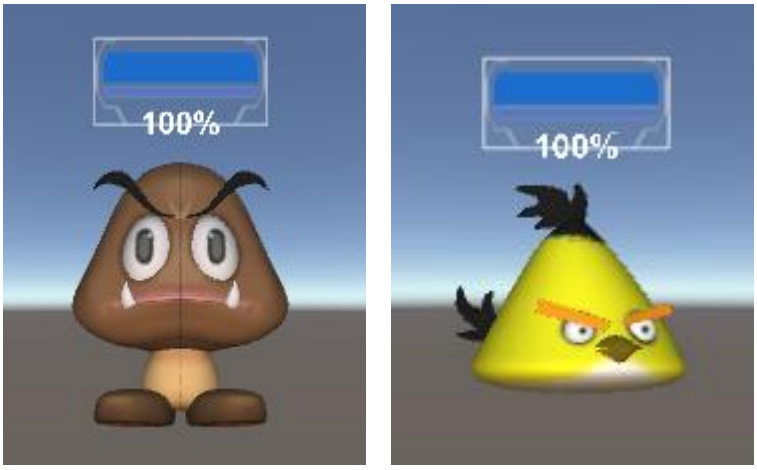


Ilustración 28: Emociones_actor_enojado_01 y 02

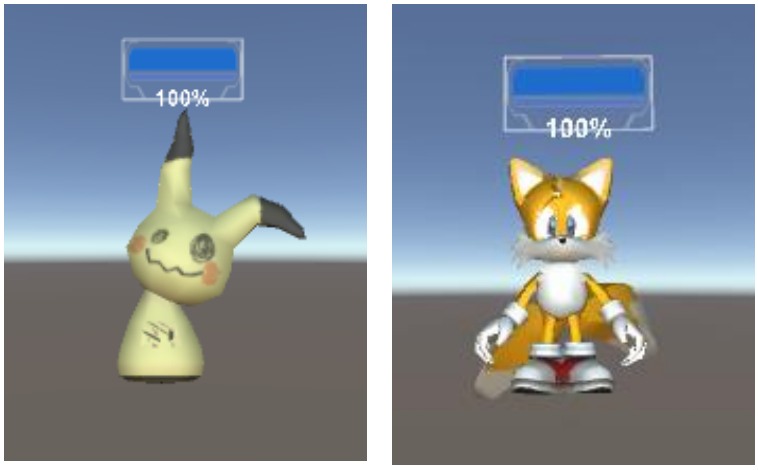


Ilustración 29: Emociones_actor_triste_01 y 02

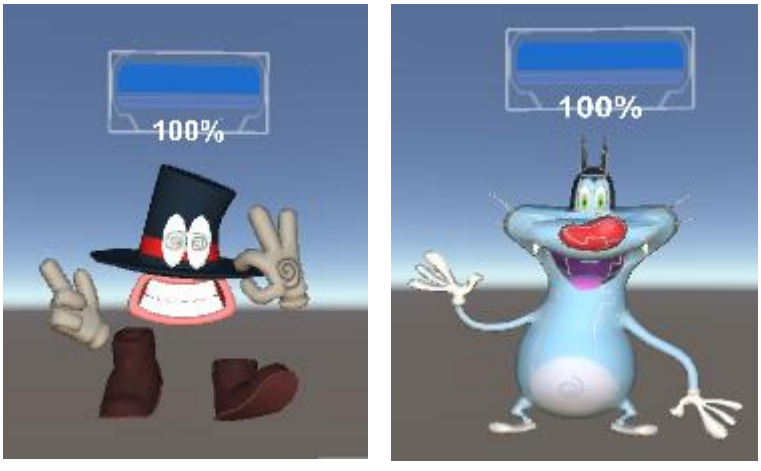


Ilustración 30: Emociones_actor_alegre_01 y 02

Cada Actor prefabricado es un objeto 3D que tienen asociado un script Actor el cual implementa funcionalidades para controlar su comportamiento. A nivel de

interfaz, el objeto contiene un canvas para mostrar el nivel de vida del personaje representado por el atributo público Life.

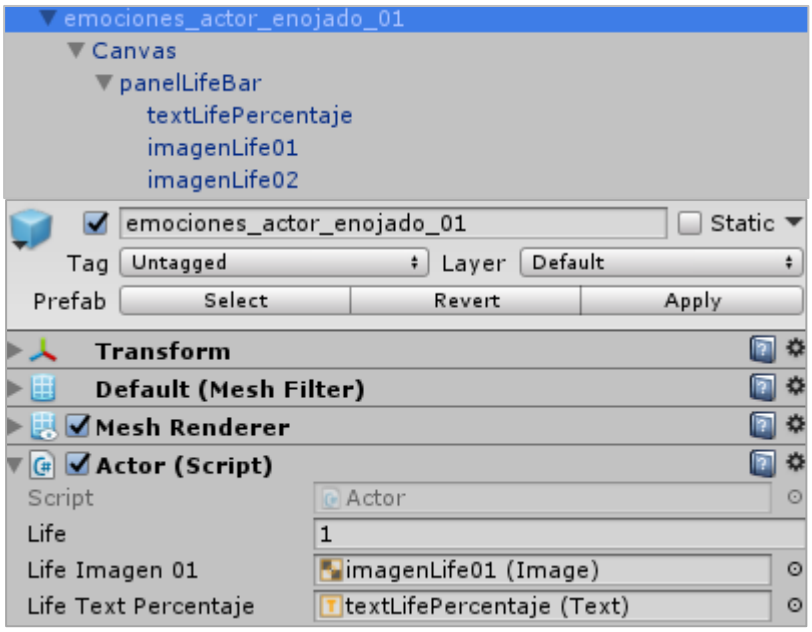


Ilustración 31: Prefab Actor

El script Actor (ver anexo 5) hace referencia a la clase Actor del modelo definido en la ilustración 1 y hereda de la clase Photon.PunBehaviour e interface IPunObservable del paquete PUN para su sincronización con todos los jugadores (ver capítulo 4.1.2).

```
public class Actor : Photon.PunBehaviour, IPunObservable{

    public ActorType actorType;
    public float life = 1;
    bool activeAddPointsCollaborative = false;
    int playerAddPointsCollaborative = -1;
    int timeCurrentCollaborativeSeconds = 0;
    float percentajeAcumulateCollaborative = 0;

    void Update(){……}

    void updateLife(float percentaje, int playerID){……}

    void updateLifeCollaborativeMaster(float percentaje,
    int playerID) {……}

    void initCollaborative{……}

    void OnCollisionEnter(Collision col) {……}

    public void OnPhotonSerializeView(PhotonStream stream,
    PhotonMessageInfo info) {……}
}
```

Ilustración 32: Prefab Scritp Actor

Se detallan a continuación los métodos del script que implementan todas las funcionalidades referentes al comportamiento que debe tener un Actor o Personaje:

Update: Los objetos Actores al estar definidos como propiedad de la sala de juego solo pueden ser manipulados por el jugador maestro quien es el primero que ingresa a la sala. El jugador maestro puede realizar acciones como moverlo, destruirlo y además es el encargado de sincronizar dichas acciones con los jugadores restantes. Mencionando dicha característica, el método update se encarga de las siguientes funcionalidades:

- Inicializa el atributo actorType solo para aquellos usuarios que ingresan a la sala después del jugador maestro.
- Muestra el nivel de vida en el canvas de acuerdo al campo life.
- Lo elimina de la escena si la vida es igual a 0 (life==0).
- Crea un temporizador y lo mueve al personaje al transcurrir un tiempo. El tiempo límite está dado por el atributo timeToMovi del objeto actorType que tiene asociado el actor (actorType.timeToMovi), donde la posición nueva del personaje corresponde a un punto aleatorio del mapa relativo al mundo real.
- Inicializa y culmina el temporizador para la captura del personaje de forma colaborativa.

UpdateLife: es una llamada de procedimiento remoto (RPC) disponible por Pun para sincronizar la vida del personaje en todos los jugadores.

UpdateLifeCollaborativeMaster: es una llamada de procedimiento remoto (RPC) disponible por PUN y que puede ser ejecutada solo por el jugador maestro para iniciar la captura del personaje de forma colaborativa. Al finalizar la colaboración, este método invoca al RPC updateLife para sincronizar en todos los jugadores los puntos de colaboración obtenidos.

OnPhotonSerializeView: método de Pun que es invocado varias veces por segundo para poder escribir y leer datos de sincronización. Entre los datos que necesitamos sincronizar son life, timeCurrentCollaborativeSeconds, activeAddPointsCollaborative y el actorTypeIdCurrentCollaborative.

OnCollisionEnter: detecta el impacto con un objeto. Al detectarse la colisión con un objeto de tipo proyectil que es ejecutado por el jugador local y si este perjudica al personaje, entonces se procede a destruirlo de la escena y se invoca al RPC de acuerdo a la modalidad de juego que se está ejecutando:

- Competitivo: se invoca al método updateLife para actualizar la vida del personaje en todos los jugadores.
- Colaborativo: se invoca al método updateLifeCollaborativeMaster para que el jugador master inicie la colaboración.

Los RPC reciben como argumento principal el porcentaje de daño (lifePercentaje), el cual es calculado por los puntos que tiene el proyectil (actorType.resultsProjectileType) referente a los puntos totales que tiene el actor (actorType.lifeInit).

```
int units = actorType.resultsProjectileType[
            gameObjectProjectile.projectileType.id.ToString()
            ];
float lifePercentaje = units / (float)actorType.lifeInit;
```

Ilustración 33: Prefab Scritp Actor, porcentaje de daño

Finalmente para hacer que el objeto siempre sincronice la posición y sus datos con todos los jugadores, se agregó los componentes PhotonView y PhotonTransformView de Pun como muestra la siguiente ilustración:

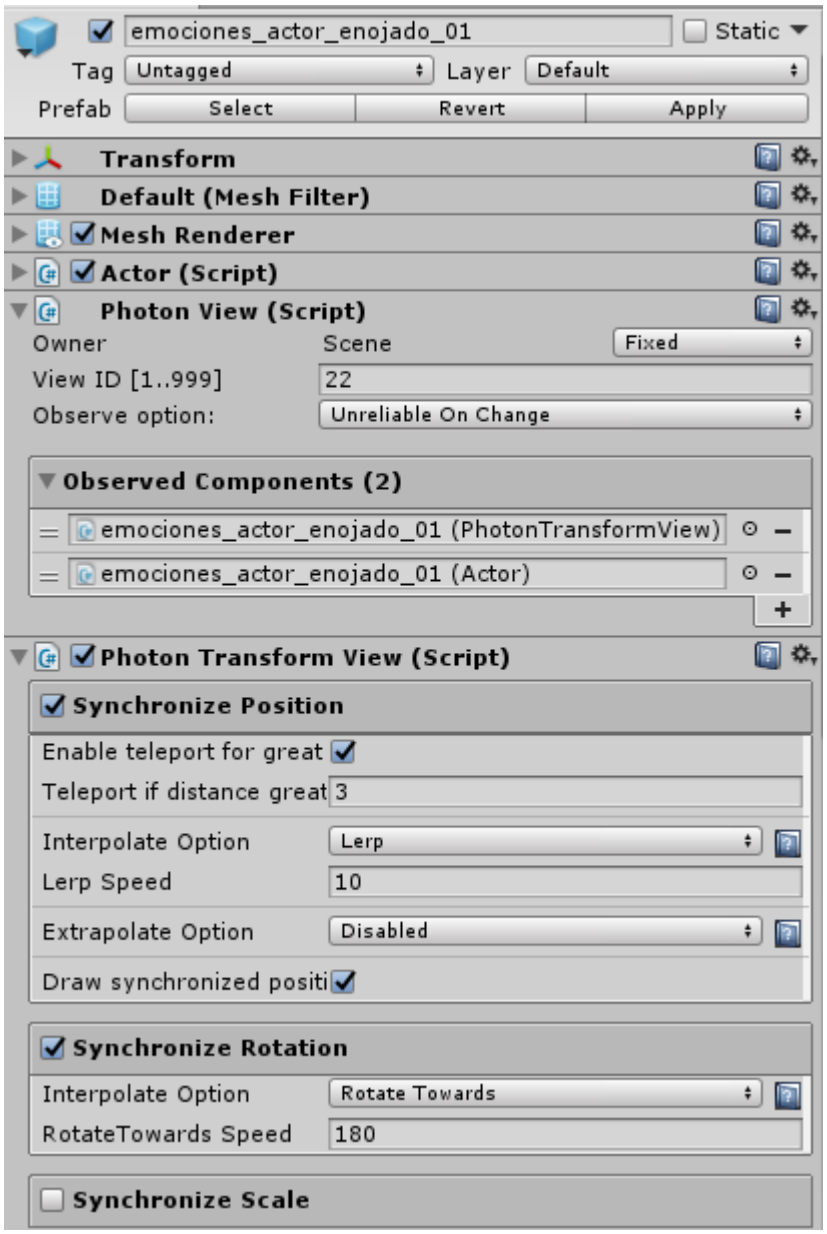


Ilustración 34: Prefab Actor, componentes Photon Pun

4.3.3.2. Projectile

El juego de emociones contiene 21 proyectiles que afectan a cada personaje de acuerdo a lo definido en el archivo configuracion.json (atributo resultsType de cada actorType).

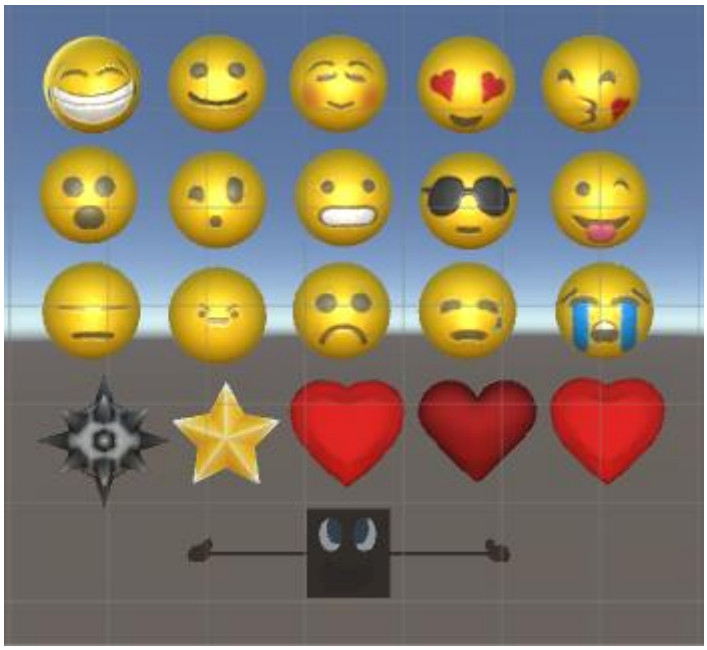


Ilustración 35: Proyectiles

Cada proyectil pueden afectar a cada personaje de diferentes formas, ya sea sumando puntos, reduciendo puntos o ningún daño, siendo para el juego de emociones lo siguientes efectos:

- **Puntos positivos**: los proyectiles que expresen sonrisas, corazones, besos, estrella, bomba afectan de forma negativa al personaje, es decir reducirá la vida del personaje (0, nivel mínimo de vida) a fin que el jugador obtenga la mayor cantidad de puntos. Los puntos disponibles del personaje se reducirán de acuerdo al campo units que tiene asignado el proyectil en la lista de resultsType (ver ilustración 16).
- **Puntos negativos**: los proyectiles que expresen llanto incrementan el nivel de vida de los personajes (1, nivel máximo) y además estos puntos se restaran del total de puntos acumulados que tiene el jugador hasta ese momento. Los puntos del personaje se incrementaran de acuerdo al campo units que tiene asignado el proyectil en la lista de resultsType (ver ilustración 16).
- **Sin efecto**: Pueden existir algunos proyectiles que no tienen ningún efecto sobre estos personajes y son todos aquellos que no se encuentren en la lista de resultsType para cada actorType (ver ilustración 16).

Cada Proyectil prefabricado tiene asociado el script Projectile el cual hace referencia a la entidad Proyectile del modelo de clases definido en la ilustración 1.

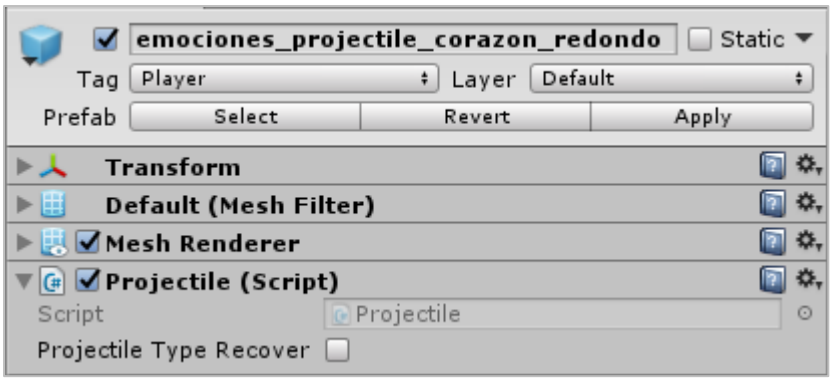


Ilustración 36: Prefab Projectile

El script Projectile (ver anexo 6) además de hacer referencia a la clase Projectile, hereda de Photon.PunBehaviour de Pun para la sincronización de los objetos en red (ver capítulo 4.1.2).

```
using UnityEngine;
namespace ExitGames.Upv.ProyectoTfm
{
    public class Projectile : Photon.PunBehaviour
    {
        public ProjectileType projectileType;
        public bool projectileTypeRecover;

        void Update()
        {
         ..............
        }
    }
}
```

Ilustración 37: Prefab Script Projectile

El script tiene como objetivo ejecutar el disparo del proyectil en dirección a donde está apuntando la cámara y hacer que el objeto se visualice siempre en el centro de la cámara. Esta implementación se encuentra en el método update y se ejecuta solo para aquellos proyectiles que no son de tipo recuperación.

```
if (projectileTypeRecover == false)
{
  // Inicializa el projectileType cuando es propiedad de los usuarios
  // remotos
  if (projectileType == null){
    object[] instantiationData = photonView.instantiationData;
    if (instantiationData != null){
      int idProjectileType = (int)instantiationData[0];
      projectileType = Game.gameInstance.getProjectileTypeById
                       (idProjectileType);
     }
 }

  // Cuando el projectile es propiedad del jugador local.
  if (photonView.isMine){

    // Se dispara en dirección donde está la cámara
    if (Game.gameInstance.runProjectile){
      Vector3 address = Game.gameInstance.runProjectileAddress
                        *Time.deltaTime;
      transform.position += address;

    // Se mueve el objeto con la cámara
    } else {
    Vector3 positionProjectile = Camera.main.transform.position +
                            Camera.main.transform.forward * 0.4f;
    Quaternion rotationProjectile = Camera.main.transform.rotation;
    transform.position = positionProjectile;
    transform.rotation = rotationProjectile;

       //Rotación de acuerdo al archivo de configuración
       if (projectileType != null)
       {
         transform.Rotate(new Vector3(projectileType.rotationX,
                                      projectileType.rotationY,
                                      projectileType.rotationZ));
       }
    }
  }
}
```

Ilustración 38: Prefab Script Projectile, método Update

Finalmente para hacer que la posición del objeto este siempre sincronizado con todos los jugadores, se agregó los componentes PhotonView y PhotonTransformView de Pun habilitando las opciones como ilustra la siguiente imagen:

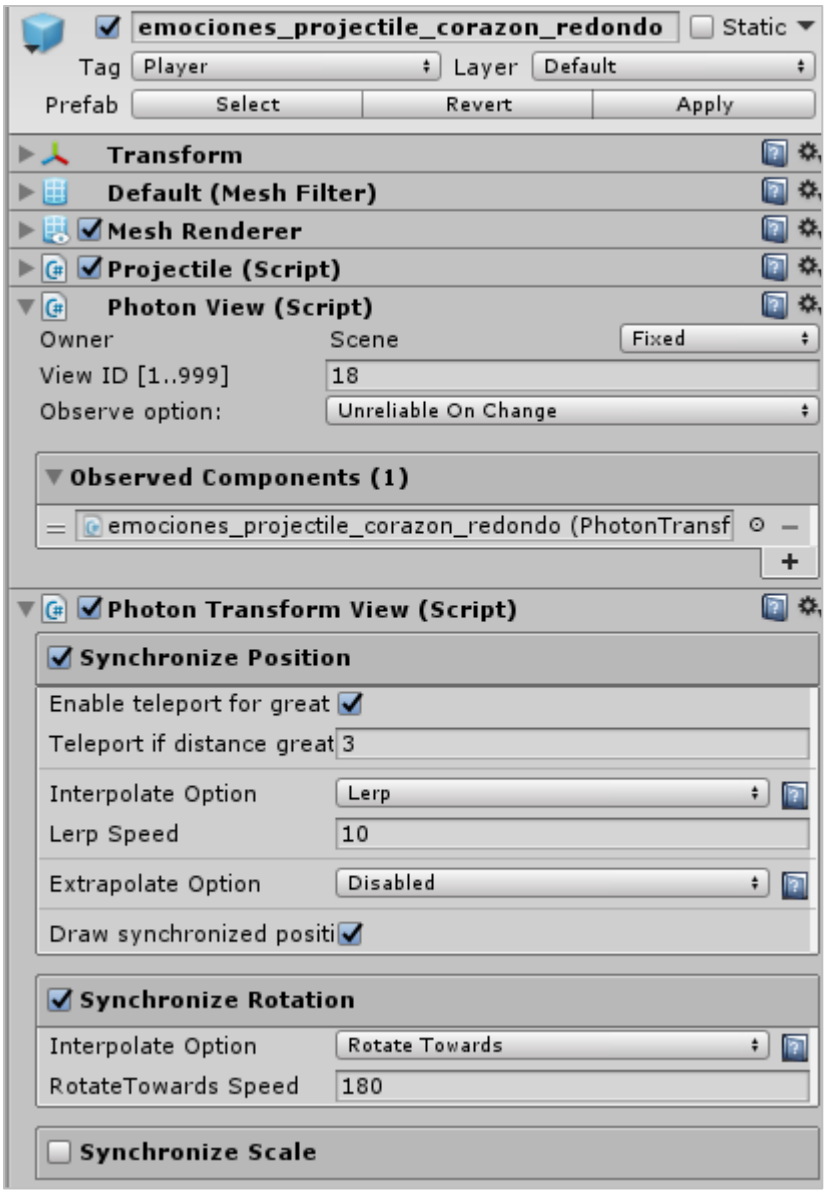


Ilustración 39: Prefab Projectile, componentes Photon Pun

5. Evaluación experimental

Con el objetivo de comparar las dos modalidades y así evaluar la aplicabilidad del juego se realizó la evaluación experimental con un grupo de niños de primaria. Previo al experimento se procedió a escanear el lugar para disponer del mapa el cual es un requisito indispensable para dar inicio con el juego (ver capítulo 4.2). El juego de emociones dispuso de solo 4 personajes a capturar, 2 enojados y 2 tristes (ver ilustración 28 y 29) debido al tiempo disponible que se tenía para realizar el experimento.

5.1. Descripción de las tareas

El experimento se realizó por parejas de niños invirtiendo un tiempo estimado de 15 minutos por cada pareja. Las tareas de cada pareja fue ejecutar el juego en las dos modalidades disponibles:

- *Competitivo:* Cada niño en función de sus lanzamientos de proyectiles debía obtener la mayor cantidad de puntos y personajes, para en función de ello ganar o empatar, caso contrario, si obtenía puntos menores al de su contrincante perdía el juego.
- *Colaborativo:* cada pareja de niños tenía que ponerse de acuerdo sobre el personaje a capturar, para posterior a ello en un tiempo establecido lanzar el proyectil hacia este personaje. La culminación exitosa de este juego tenía como requisito principal que los niños puedan establecer una comunicación para capturar de forma colaborativa todos los personajes empleando el menor tiempo posible. En este juego, todos los participantes son ganadores.

Con el objetivo de obtener las opiniones de cada niño respecto a la experiencia obtenida por cada modalidad de juego, se aplicó un cuestionario tras finalizar cada iteración (competitivo y colaborativo). El cuestionario consta de 7 preguntas detallas a continuación:

Preguntas	**Escala**
1. ¿Cuánto te divertiste en el juego?	1. Nada, 2. Poco, 3. Algo, 4. Bastante y 5.Mucho
2. ¿Qué tan fácil fue manejar el juego?	1. Súper difícil, 2. Difícil, 3. Normal, 4. Fácil y 5. Súper fácil
3. ¿Cuánto te gustaría jugar nuevamente en clase a este juego?	1. Nada, 2. Poco, 3. Algo, 4. Bastante y 5.Mucho
4. ¿Cuánto te gustaría jugar otra vez fuera de clase a	1. Nada, 2. Poco, 3. Algo, 4. Bastante y 5.Mucho

este juego?	
5. ¿Cuánto te gustaría jugar a este juego con amigos?	1. Nada, 2. Poco, 3. Algo, 4. Bastante y 5.Mucho
6. ¿En qué asignatura de tu clase te gustaría jugar el juego?	Respuesta libre
7. ¿Qué cambiarías en el juego para que te guste más?	Respuesta libre

Tabla 5: Cuestionario para evaluar cada modalidad de juego

Al finalizar el juego en las dos modalidades, se aplicó a cada niño un último cuestionario que tiene como objetivo comparar las dos versiones de juego para respaldar la credibilidad de los datos del cuestionario anterior.

Preguntas	**Escala**
1. ¿Con qué versión del juego se te hizo más fácil jugar?	1. Compitiendo 2. Colaborando 3. Ambas
2. ¿Con qué versión del juego crees que jugaste mejor?	
3. ¿Con qué versión del juego te divertiste más?	
4. ¿Con qué versión del juego te gustaría jugar en clase?	
5. ¿Con qué versión del juego te gustaría jugar fuera clase?	

Tabla 6: Cuestionario para comparar las modalidades del juego

A más de aplicar los cuestionarios a los niños, se utilizó una plantilla observacional para obtener información sobre el estado anímico, actividad física, interacción social, uso de la herramienta y comentarios de la actividad respecto al uso del juego para cada modalidad (ver tabla 7). En este proceso, participaron dos observadores que tenían la responsabilidad de evaluar en tres instantes dados, al inicio, intermedio y tras finalizar el juego (Evl.1, Evl.2, Evl.3) [34].

Evento	**Escala**	**Niño**	**Evl.1**	**Evl.2**	**Evl.3**
1. Afecto	0. Aburrido/triste 1. Afecto plano 2. Alegre 3. Eufórico	**A**			
		B			
2. Actividad física	1. Nada de movimiento 2. Movimiento casi nulo 3. Muestra movimiento 4. Movimiento muy activo	**A**			
		B			
3. Interacción social	1. Juega individual 2. Responde la interacción 3. Dirige la interacción 4. Produce un juego colaborativo	**A**			
		B			
4. Uso de la herramienta	1. No presta interés 2. Explora pasivamente	**A**			

	3. Explora con interés 4. Explora y propone nuevas ideas	**B**			
5. Satisfacción	1. Comentarios negativos 2. Ningún comentario	**A**			
	3. 1 comentario positivo 4. > de un comentario positivo	**B**			

Tabla 7: Plantilla Observacional para cada modalidad del juego

Finalmente, al juego colaborativo se aplicó una rúbrica que permite observar la comunicación efectuada [35] por cada pareja de niños (ver tabla 8). Cada ítem tiene un código de calidad de la colaboración definido como: -2 muy mal, -1 mal, 0 neutro, +1 bien, +2 muy bien.

Dimensión	**Niño A**	**Niño B**
1. Mantener la comprensión mutua.		
2: Gestión del diálogo		
3. Conjunto de información		
4. Alcanzando el consenso		
5. División de tareas		
6. Gestión del tiempo		
7. Coordinación técnica		
8. Interacción recíproca		
9. Orientación de tareas individuales		

Tabla 8: Plantilla para la observación de la comunicación en el juego colaborativo

5.2. Participantes

Los participantes en el proceso de evaluación fue un grupo de 38 niños de quinto curso de primaria del colegio Vicente Gaos con edades comprendidas entre 9 y 11 años. La evaluación se realizó en dos secciones de trabajo, agrupando 24 niños en la primera sección y 14 niños en la segundo sección teniendo un total de 19 parejas. En el experimento cada pareja alterno la modalidad de juego, competitiva y colaborativo y viceversa, a fin te tener mayor precisión en los datos.

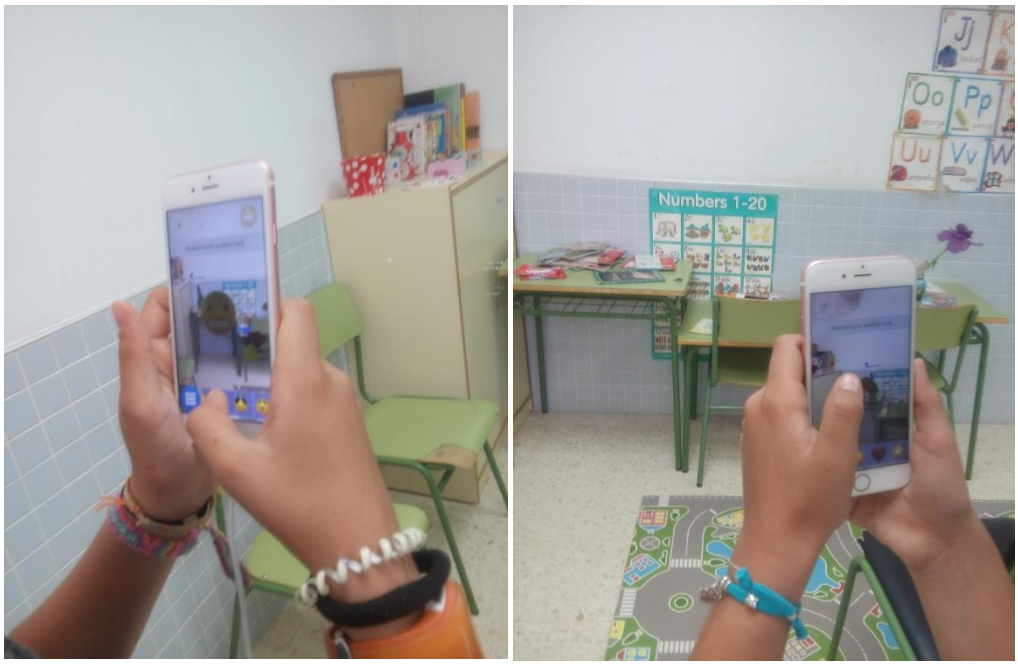


Ilustración 40: Evaluación experimental, pareja de niños

5.3. Resultados

Todos los cuestionarios y plantillas observacionales aplicados en el experimento nos permitieron obtener datos cuantitativos y cualitativos relacionados a la aplicabilidad de juego desarrollado. Estos resultados se detallan en las siguientes dos secciones:

5.3.1. Resultados de los cuestionarios

Cada niño realizo tres cuestionarios, el primero y segundo referente a evaluar cada modalidad de juego (ver tabla 5) y el tercero para comparar las dos modalidades de juego, competitivo vs colaborativo (ver tabla 6).

Cuestionario Competitivo y Colaborativo

La pregunta número 1 planteado como ¿Cuánto te divertiste en el juego? da como resultado que el 97% de los niños se divirtieron bastante y mucho en la modalidad del competitivo, mientras que en la modalidad del colaborativo se obtiene un 100%.

Escala	Frecuencia Competitivo	% Competitivo	Frecuencia Colaborativo	% Colaborativo
1. Nada	0	0%	0	0%
2. Poco	0	0%	0	0%
3. Algo	1	3%	0	0%

4. Bastante	13	34%	15	39%
5. Mucho	24	63%	23	61%
Total	**38**	**100%**	**38**	**100%**

Tabla 9: Resultados pregunta Nro. 1, modalidad del juego

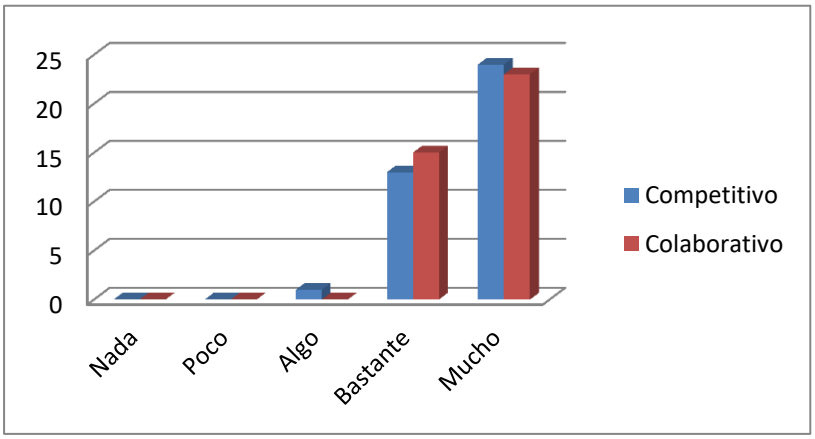


Ilustración 41: Resultados pregunta Nro. 1, modalidad de juego

En la pregunta número 2 ¿Qué tan fácil fue manejar el juego?, el 82% de los niños consideran que el manejo del juego es fácil y súper fácil para la modalidad del competitivo, mientras que en la modalidad del colaborativo se obtiene un 71%.

Escala	Frecuencia Competitivo	% Competitivo	Frecuencia Colaborativo	% Colaborativo
1. Súper difícil	0	0%	0	0%
2. Difícil	0	0%	1	3%
3. Normal	7	18%	10	26%
4. Fácil	19	50%	16	42%
5. Súper fácil	12	32%	11	29%
Total	38	100%	38	100%

Tabla 10: Resultados pregunta Nro. 2, modalidad de juego

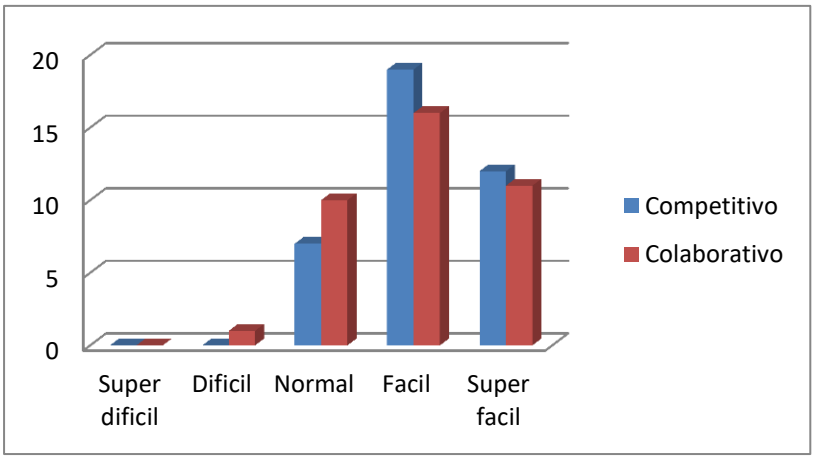


Ilustración 42: Resultados pregunta Nro. 2, modalidad de juego

En la pregunta número 3 ¿Cuánto te gustaría jugar nuevamente en clase a este juego?, para la modalidad del competitivo el 95% de los niños consideran que les gustaría entre la escala de bastante y mucho, mientras que en la modalidad del colaborativo para ambas escalas se obtiene un 92%.

Escala	Frecuencia Competitivo	% Competitivo	Frecuencia Colaborativo	% Colaborativo
1. Nada	0	0%	0	0%
2. Poco	0	0%	0	0%
3. Algo	2	5%	3	8%
4. Bastante	5	13%	7	18%
5. Mucho	31	82%	28	74%
Total	38	100%	38	100%

Tabla 11: Resultados pregunta Nro. 3, modalidad de juego

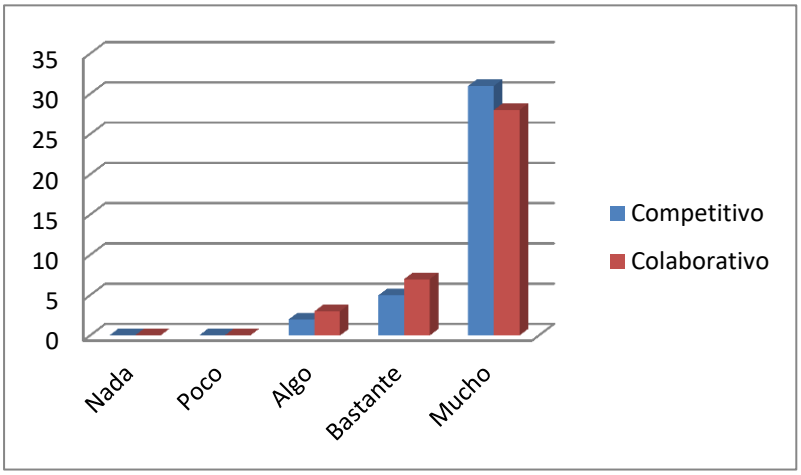


Ilustración 43: Resultados pregunta Nro. 3, modalidad de juego

En la pregunta número 4 ¿Cuánto te gustaría jugar otra vez fuera de clase a este juego?, para la modalidad del competitivo el 97% de los niños consideran que les gustaría entre la escala de bastante y mucho, mientras que en la modalidad del colaborativo para ambas escalas se obtiene un 95%.

Escala	Frecuencia Competitivo	% Competitivo	Frecuencia Colaborativo	% Colaborativo
1. Nada	0	0%	0	0%
2. Poco	0	0%	0	0%
3. Algo	1	3%	2	5%
4. Bastante	15	39%	13	34%
5. Mucho	22	58%	23	61%
Total	38	100%	38	100%

Tabla 12: Resultados pregunta Nro. 4, modalidad de juego

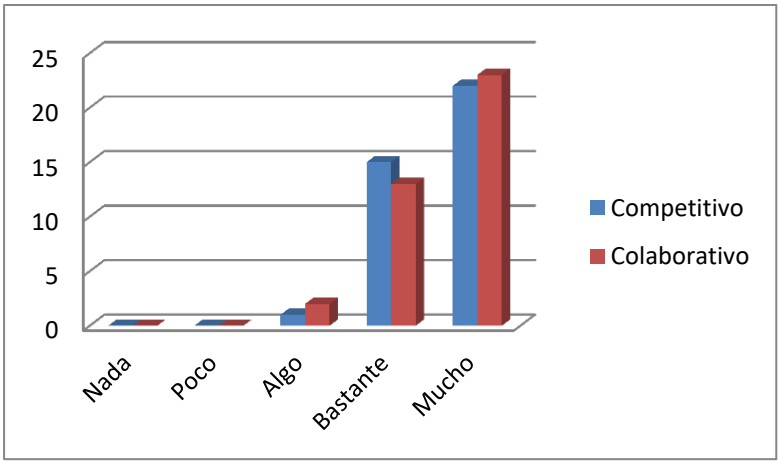


Ilustración 44: Resultados pregunta Nro. 4, modalidad de juego

En la pregunta número 5 ¿Cuánto te gustaría jugar a este juego con amigos?, para la modalidad del competitivo el 98% de los niños consideran que les gustaría entre la escala de bastante y mucho, mientras que en la modalidad del colaborativo para ambas escalas se obtiene un 100%.

Escala	Frecuencia Competitivo	% Competitivo	Frecuencia Colaborativo	% Colaborativo
1. Nada	0	0%	0	0%
2. Poco	0	0%	0	0%
3. Algo	1	3%	0	0%
4. Bastante	9	24%	7	18%
5. Mucho	28	74%	31	82%
Total	38	100%	38	100%

Tabla 13: Resultados pregunta Nro. 5, modalidad de juego

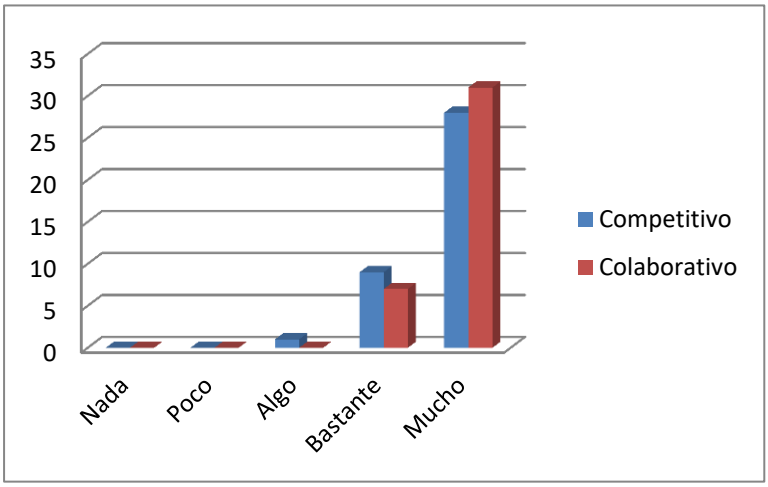


Ilustración 45: Resultados pregunta Nro. 5, modalidad de juego

La pregunta número 6 ¿En qué asignatura de tu clase te gustaría jugar el juego? dio como resultado que matemáticas, castellano, valenciano y sociales, son las asignaturas más solicitadas por los niños tanto para la modalidad de competitivo y colaborativo.

Asignaturas	Competitivo	Colaborativo
Matemáticas	13	10
Castellano	11	12
Valenciano	10	13
Sociales	9	7
Valores cívicos	3	3
Ingles	2	2
Naturales	2	5
Religión	1	1
Informática	1	1
E.F	1	0
Tutoría	0	1
Música	0	1
Total	**53**	**56**

Tabla 14: Resultados pregunta Nro. 6, modalidad de juego

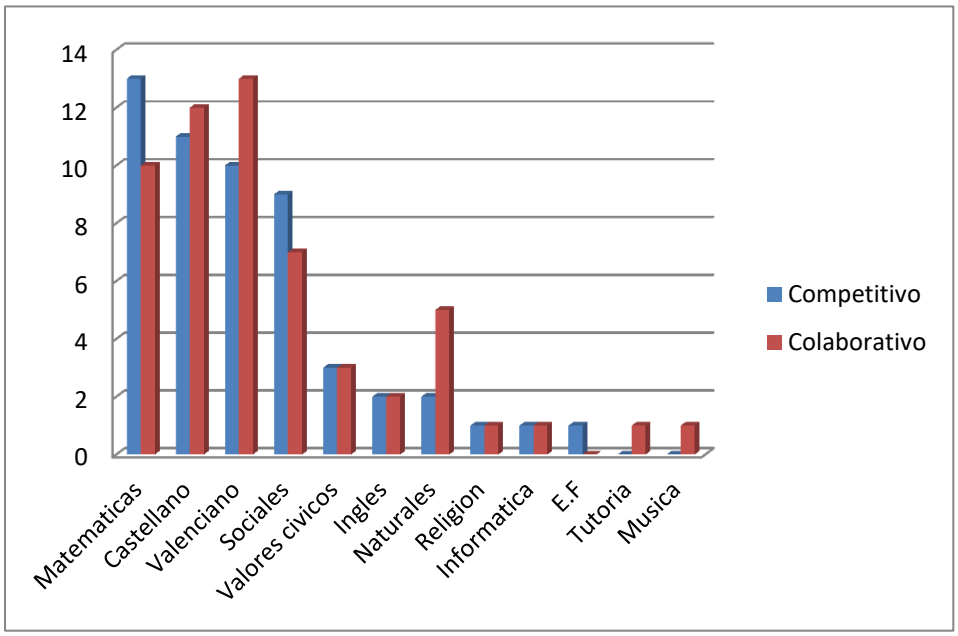


Ilustración 46: Resultados pregunta Nro. 7, modalidad de juego

La pregunta número 7 ¿Que cambiarias en el juego para que te guste más? dio como resultado algunas propuestas de mejora del juego, siendo la más popular para las dos modalidades de juego, que se incluyan más personajes de captura a fin de incrementar el tiempo de juego.

	Sugerencias	Comp.	Colab.
Proyectiles	Que se disponga de más proyectiles	3	2
	Proyectiles con efectos especiales o más brillantes	1	1
	Que hubiera pokeballs	1	1
	Que haya que lanzar en vez de tocar		1
	Toparle al proyectil 3 veces para recuperarlo	1	1
	Que el lanzamiento sea más rápido	1	1
	Que no se hicieran tan pequeños.	1	
Personajes	Que se incluyan más personajes	10	6
	Que no se muevan tanto	2	1
	Que estén por todo el aula	1	1
Juego general	Que hubiera más tiempo para jugar	1	2
	Estar con gafas de realidad virtual	1	2
	Que se pueda jugar con más personas		2
	Que se agregue más complejidad al juego	1	2
	Que sea más fácil el juego	1	
	3D		1

Tabla 15: Resultados pregunta Nro. 7, modalidad de juego

Para saber si los datos de las 5 primeras preguntas presentan una diferencia estadísticamente significativa en función de la modalidad de juego, se procedió aplicar el test de Wilcoxon obteniendo los siguientes resultados:

Pregunta	Competitivo	Colaborativo	p-valué
Pregunta Nro. 1	4.61	4.61	1.000
Pregunta Nro. 2	4.13	3.97	0.310
Pregunta Nro. 3	4.76	4.66	0.356
Pregunta Nro. 4	4.55	4.55	1.000
Pregunta Nro. 5	4.71	4.82	0.102

Tabla 16: Resultados del test Wilcoxon entre las dos modalidades de juego (*P < 0.05)

La tabla 16, da a conocer que ninguna pregunta presenta una diferencia estadísticamente significativa entre las dos modalidades de juego, competitiva vs colaborativo.

Cuestionario Competitivo vs. Colaborativo

En la pregunta número 1 ¿Con qué versión del juego se te hizo más fácil jugar?, el 50% manifestó la versión competitivo, el 34% la versión colaborativo y el 16% respondieron ambas alternativas.

Opciones	Frecuencia	Porcentaje
1. Compitiendo	19	50%
2. Colaborando	13	34%
3. Ambas	6	16%
Total	**38**	**100%**

Tabla 17: Resultados pregunta Nro. 1, modalidad competitivo vs colaborativo

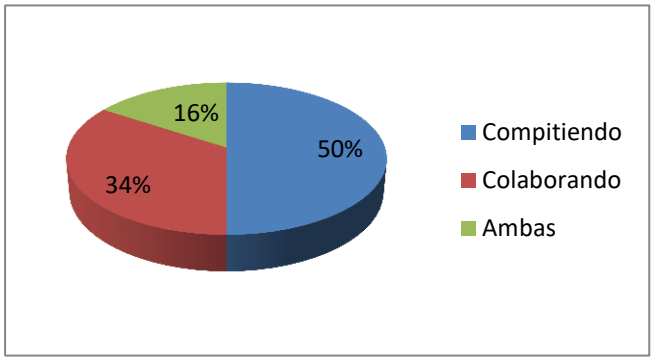


Ilustración 47: Resultados pregunta Nro. 1, modalidad competitivo vs colaborativo

En la pregunta número 2 ¿Con qué versión del juego crees que jugaste mejor?, el 53% se inclinó por la versión colaborativa, el 39% por la versión competitiva y el 8% respondieron ambas alternativas.

Opciones	Frecuencia	Porcentaje
1. Compitiendo	15	39%
2. Colaborando	20	53%
3. Ambas	3	8%
Total	**38**	**100%**

Tabla 18: Resultados pregunta Nro. 2, modalidad competitivo vs colaborativo

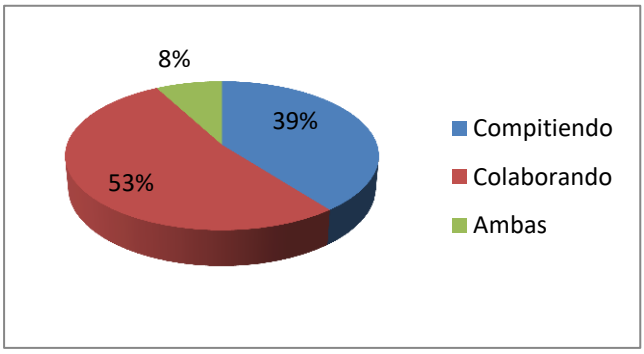


Ilustración 48: Resultados pregunta Nro. 2, modalidad competitivo vs colaborativo

En la pregunta número 3 ¿Con que versión del juego te divertiste más?, el 47% se inclinó por la versión colaborativa, el 40% por la versión competitiva y el 13% respondieron ambas alternativas.

Opciones	Frecuencia	Porcentaje
1. Compitiendo	15	40%
2. Colaborando	18	47%
3. Ambas	5	13%
Total	**38**	**100%**

Tabla 19: Resultados pregunta Nro. 3, modalidad competitivo vs colaborativo

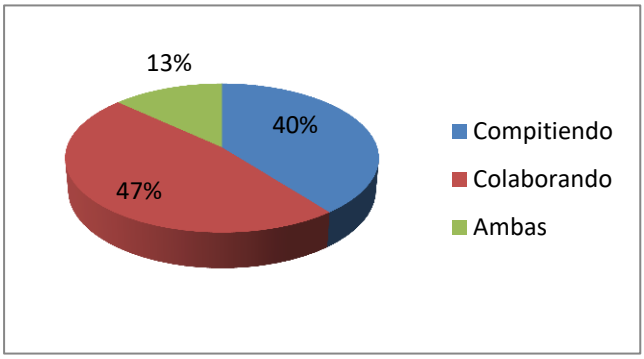


Ilustración 49: Resultados pregunta Nro. 3, modalidad competitivo vs colaborativo

La pregunta número 4 ¿Con qué versión del juego te gustaría jugar en clase? y la 5 ¿Con qué versión del juego te gustaría jugar fuera de clase?, reflejaron resultados iguales, donde, el 55% se inclinó por la versión competitiva, el 26% por la versión colaborativa y el 19% respondieron ambas alternativas.

Opciones	Frecuencia	Porcentaje
1. Compitiendo	21	55%
2. Colaborando	10	26%
3. Ambas	7	19%
Total	**38**	**100%**

Tabla 20: Resultados pregunta Nro. 4 y 5, modalidad competitivo vs colaborativo

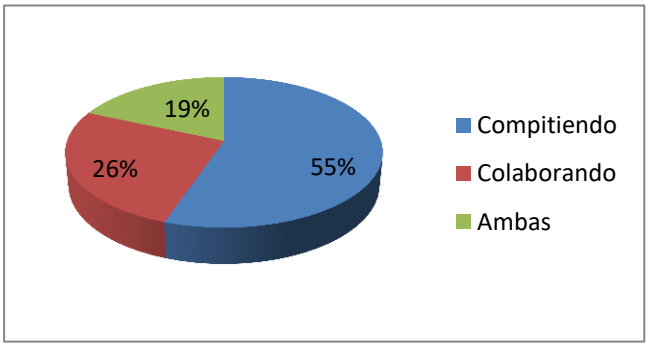


Ilustración 50: Resultados pregunta Nro. 4 y 5, modalidad competitivo vs colaborativo

5.3.2. Resultados observacionales

Cada observador completo tres plantillas por cada pareja de niños, las dos primeras evaluaron de forma independiente cada modalidad del juego (ver tabla 7) y la tercera evaluó para la versión colaborativa la comunicación abordada entre ellos (ver tabla 8).

Plantilla observacional Competitivo y Colaborativo

El afecto, como primer parámetro evaluado en la plantilla observacional para la versión competitiva dio como resultados que el 59% de los niños inician el juego con un afecto plano y el 68% finalizan entre alegres y eufóricos. Para la versión colaborativa el 50% de los niños inician con un afecto plano, a mediados del juego la escala alegre incrementa al 63% y el 83% expresan entre alegría y euforia tras finalizar el juego.

	Competitivo			Colaborativo		
Escala	Eval. 1	Eval.2	Eval.3	Eval. 1	Eval.2	Eval.3
Aburrido/triste	2	0	0	2	0	2
Afecto plano	45	33	24	38	19	11
Alegre	28	42	49	33	48	47
Eutrófico	1	1	3	3	9	16
Total	**76**	**76**	**76**	**76**	**76**	**76**

Tabla 21: Resultados observacionales, afecto

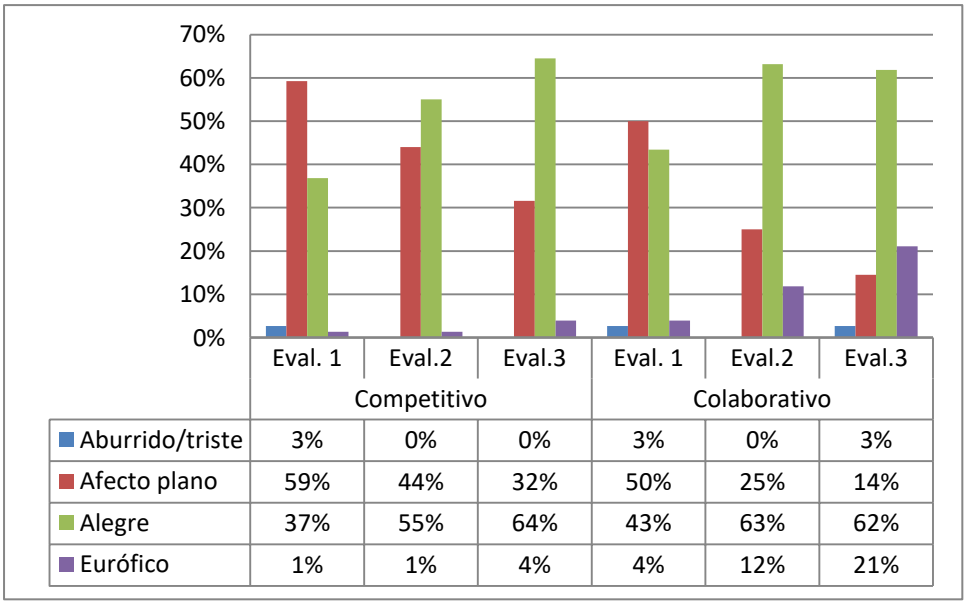


Ilustración 51: Resultados observacionales, afecto

Los resultados del parámetro actividad física indican para la versión competitiva que gran parte de los niños tienen movimiento casi nulo, con un porcentaje del 64% al inicio, a mediados del juego el porcentaje disminuye al 53% y termina con el 49%. Para la versión colaborativa el 70% de los niños inician con movimiento casi nulo, a mediados del juego el movimiento casi nulo sigue sobresaliendo con un 47% y tras finalizar el juego la escala mostrar movimiento sobresale con un 43%. En la ilustración 52 se puede destacar que un grupo de niños inician con movimiento casi nulo y con el transcurso del juego culminan mostrando movimiento, haciendo mayor hincapié en la modalidad el juego colaborativo.

	Competitivo			Colaborativo		
Escala	Eval. 1	Eval.2	Eval.3	Eval. 1	Eval.2	Eval.3
Nada de movimiento	15	14	11	10	13	6
Movimiento casi nulo	49	40	37	53	36	31
Muestra movimiento	12	22	27	13	22	33
Movimiento muy activo	0	0	1	0	5	6
Total	**76**	**76**	**76**	**76**	**76**	**76**

Tabla 22: Resultados observacionales, actividad física

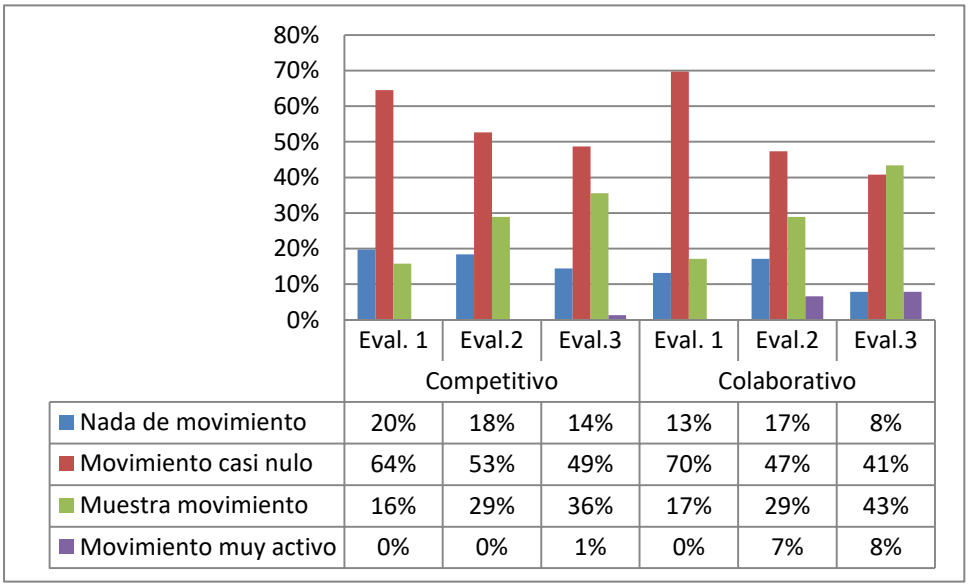


Ilustración 52: Resultados observacionales, actividad física

Los resultados del parámetro interacción social expresan para la versión competitiva que el 97% de los niños juegan individualmente al inicio, el 94% continúan jugando de esta forma y tras finalizar el juego, el porcentaje sube al 100%. Para la versión colaborativa los niños producen un juego colaborativo con el 53%, 51% y 75% en el tiempo del inicio, intermedio y final del juego.

	Competitivo			Colaborativo		
Escala	Eval. 1	Eval.2	Eval.3	Eval. 1	Eval.2	Eval.3
Juega individualmente	74	72	76	4	6	4
Responde la interacción	2	2	0	20	15	8
Dirige la interacción	0	0	0	12	16	7
Produce un juego colaborativo	0	2	0	40	39	57
Total	**76**	**76**	**76**	**76**	**76**	**76**

Tabla 23: Resultados observacionales, interacción social

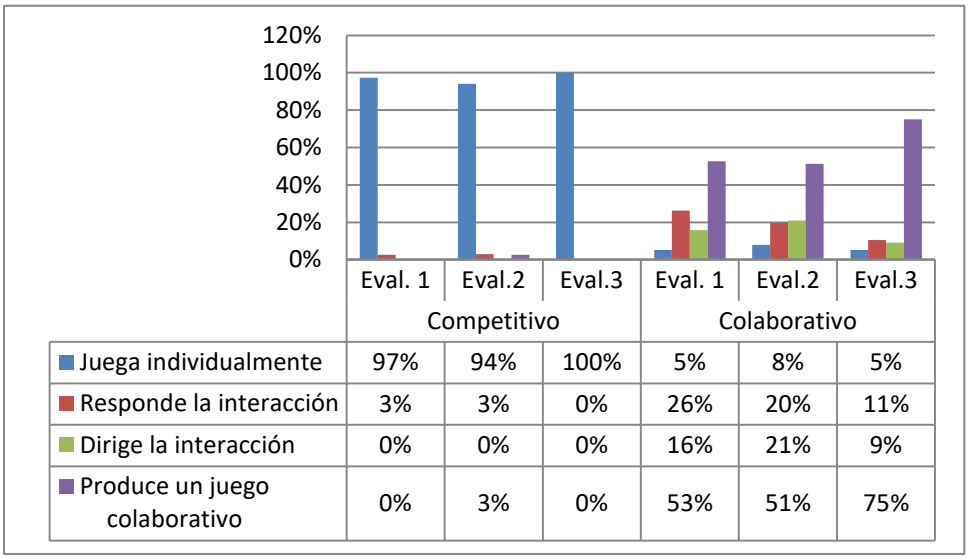
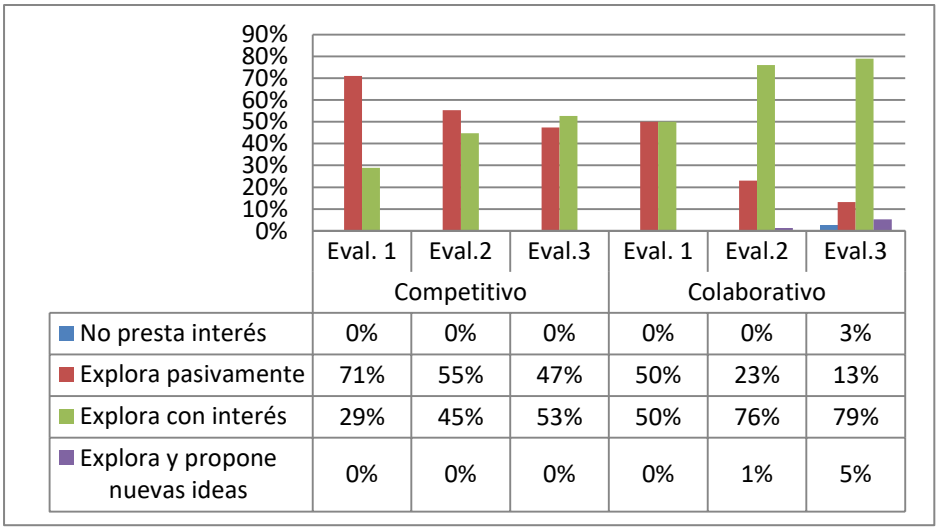


Ilustración 53: Resultados observacionales, interacción social

Los resultados del uso de la herramienta expresan para la versión competitiva que al inicio del juego el 71% de los niños exploran la herramienta de forma pasiva, a mediados del juego la exploración pasiva continua sobresaliendo con un 55% y tras finalizar el juego el 53% exploran la herramienta ya con interés. En la versión colaborativa al inicio del juego existe un empate del 50% en la exploración pasiva y con interés, en el tiempo intermedio y tras finalizar el juego los niños ya exploran la herramienta con interés en un porcentaje del 76 y 79%.

	Competitivo			Colaborativo		
	Eval. 1	Eval.2	Eval.3	Eval. 1	Eval.2	Eval.3
No presta interés	0	0	0	0	0	2
Explora pasivamente	54	42	36	38	17	10
Explora con interés	22	34	40	38	58	60
Explora y propone nuevas ideas	0	0	0	0	1	4
Total	**76**	**76**	**76**	**76**	**76**	**76**

Tabla 24: Resultados observacionales, uso de la herramienta



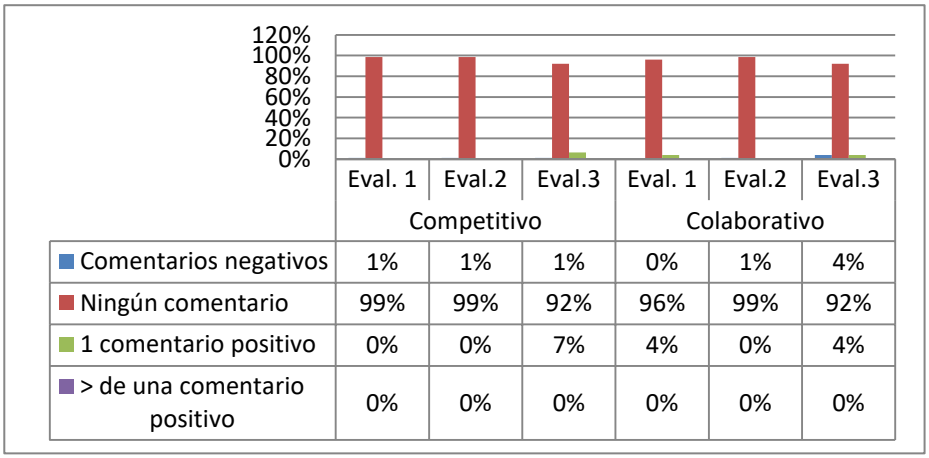
	Competitivo			Colaborativo		
No presta interés	0%	0%	0%	0%	0%	3%
Explora pasivamente	71%	55%	47%	50%	23%	13%
Explora con interés	29%	45%	53%	50%	76%	79%
Explora y propone nuevas ideas	0%	0%	0%	0%	1%	5%

Ilustración 54: Resultados observacionales, uso de la herramienta

El parámetro satisfacción respecto a emitir comentarios sobre la actividad para la versión competitivo da como resultado que los niños no emiten ningún comentario con el 99%, 99% y 92% en el tiempo del inicio, intermedio y final del juego. En la versión colaborativa de forma similar los niños no emiten ningún comentario con el 96%, 99% y 92%.

	Competitivo			Colaborativo		
	Eval. 1	Eval.2	Eval.3	Eval. 1	Eval.2	Eval.3
Comentarios negativos	1	1	1	0	1	3
Ningún comentario	75	75	70	73	75	70
1 comentario positivo	0	0	5	3	0	3
> de una comentario positivo	0	0	0	0	0	0
Total	**76**	**76**	**76**	**76**	**76**	**76**

Tabla 25: Resultados observacionales, satisfacción



	Competitivo			Colaborativo		
Comentarios negativos	1%	1%	1%	0%	1%	4%
Ningún comentario	99%	99%	92%	96%	99%	92%
1 comentario positivo	0%	0%	7%	4%	0%	4%
> de una comentario positivo	0%	0%	0%	0%	0%	0%

Ilustración 55: Resultados observacionales, satisfacción

Para proceder a identificar si los datos anteriores presentan una diferencia estadísticamente significativa referente a la modalidad de juego, se realizó como paso previo, una prueba de validez interna inter-jueces llamada Índice Kappa para evaluar la concordancia observada entre los dos examinadores (observador 1 y 2).

Kappa	**Grado de acuerdo**	**Valor obtenido**
< 0	sin acuerdo	
0 - 0,2	insignificante	
0,2 - 0,4	bajo	
0,4 - 0,6	moderado	
0,6 - 0,8	bueno	
0,8 - 1	muy bueno	0,87

Tabla 26: Resultados índice Kappa, observación factores competitivo y colaborativo

El valor 0,87 puntualiza que los datos registrados por el observador uno y dos tiene un grado muy bueno de concordancia. A continuación, aplicando el test de Wilcoxon se obtiene para cada parámetro el valor p que define la diferencia significativa entre la modalidad de juego competitivo y colaborativo.

Parámetro	**Competitivo**	**Colaborativo**	**p-valué**
Afecto	1.5570	1.7900	0.000 *
Actividad física	1.1012	1.2678	0.026 *
Interacción social	0.0437	2.2853	0.000 *
Uso de la herramienta	1.4213	1.7200	0.000 *
Satisfacción	1.0087	1.0087	1

Tabla 27: Resultados del test Wilcoxon de la plantilla observacional (*P < 0.05)

A través de la tabla 27 se concluye que el parámetro afecto, actividad física, interacción social y uso de la herramienta presenta una diferencia estadísticamente significativa entre la modalidad de juego competitiva y colaborativo, por lo cual, utilizando la diferencia de medias se enfatiza que estos 4 parámetros tienen mejores resultados para la versión colaborativa.

Plantilla observacional de la comunicación para la versión Colaborativa

El primer parámetro Mantener la compresión mutua evaluado en la versión del juego colaborativo arrogo que el 69% de los niños realizaron contribuciones compresibles hacia sus compañeros, dividido en el 58% y 11% para la compresión buena y muy buena.

Escala	Frecuencia	Porcentaje
muy mal	0	0%
mal	10	13%
neutro	14	18%
bien	44	58%
muy bien	8	11%
Total	**76**	**100%**

Tabla 28: Resultados observacionales, Mantener la compresión mutua

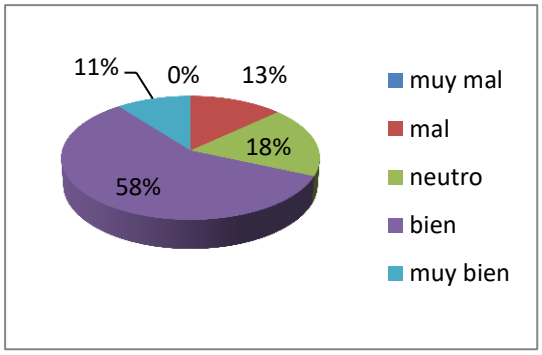


Ilustración 56: Resultados observacionales, Mantener la compresión mutua

Los resultados del parámetro Gestión de diálogo en el juego colaborativo expresan que el 75% de los niños mantuvieron un flujo de comunicación entre bien y muy bien, siendo 59% y 16% respectivamente.

Escala	Frecuencia	Porcentaje
muy mal	4	5%
mal	10	13%
neutro	5	7%
bien	45	59%
muy bien	12	16%
Total	**76**	**100%**

Tabla 29: Resultados observacionales, Gestión de diálogo

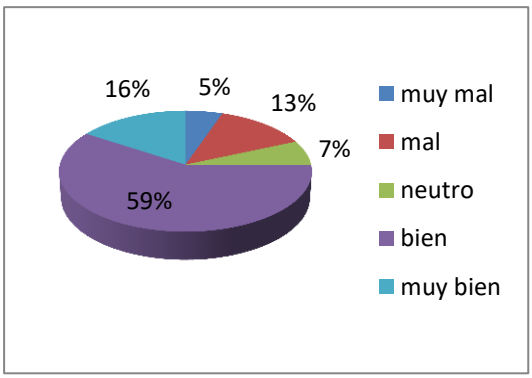


Ilustración 57: Resultados observacionales, Gestión de diálogo

Los resultados del parámetro Conjunto de Información en el juego colaborativo expresan que el 57% de los niños se comunicaron para obtener información relevante del juego, dividido en el 37% y 20% para las dos escalas máximas (bien y muy bien).

Escala	Frecuencia	Porcentaje
muy mal	8	10%
mal	4	5%
neutro	21	28%
bien	28	37%
muy bien	15	20%
Total	**76**	**100%**

Tabla 30: Resultados observacionales, Conjunto de Información

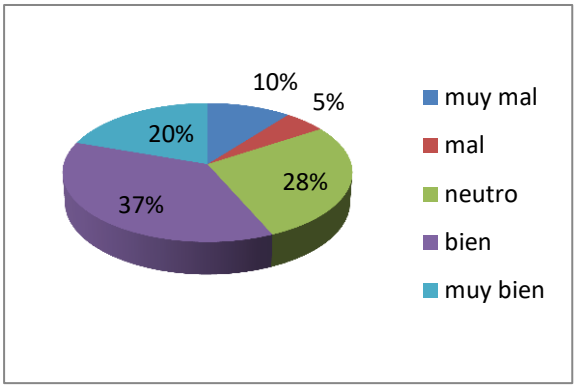


Ilustración 58: Resultados observacionales, Conjunto de Información

Los resultados del parámetro Alcanzando el consenso en el juego colaborativo expresan que el 47% de los niños intercambiaron argumentos para acordar una solución, dividido en el 29% y 18% para las dos escalas máximas (bien y muy bien).

Escala	Frecuencia	Porcentaje
muy mal	6	8%
mal	14	18%
neutro	20	27%
bien	22	29%
muy bien	14	18%
Total	**76**	**100%**

Tabla 31: Resultados observacionales, Alcanzando el consenso

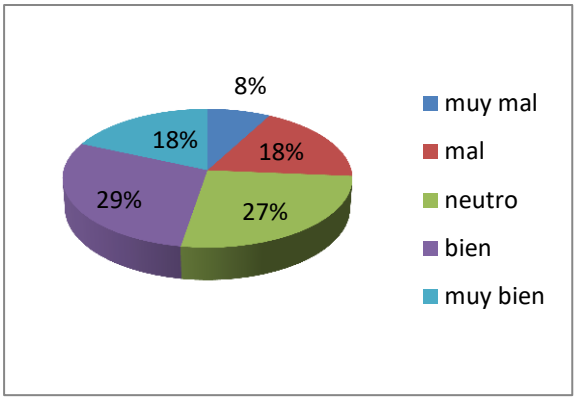


Ilustración 59: Resultados observacionales, Alcanzando el consenso

Los resultados del parámetro División de tareas en el juego colaborativo expresan que el 59% de los niños se asignaron sub-tareas de forma equitativa para llegar a un bien común que es finalizar el juego, dividido en el 46% y 13% para las dos escalas máximas (bien y muy bien).

Escala	Frecuencia	Porcentaje
muy mal	4	5%
mal	4	5%
neutro	23	31%
bien	35	46%
muy bien	10	13%
Total	**76**	**100%**

Tabla 32: Resultados observacionales, División de tareas

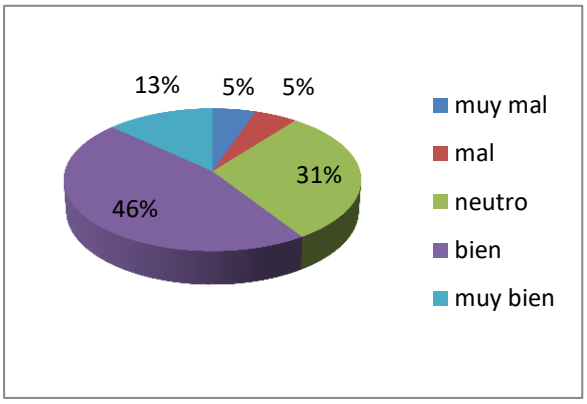


Ilustración 60: Resultados observacionales, División de tareas

Los resultados del parámetro Gestión del tiempo en el juego colaborativo expresan que el 37% de los niños supervisaron el tiempo para tratar de finalizar el juego en el menor tiempo posible, dividido en el 29% y 8% para las dos escalas máximas (bien y muy bien).

Escala	Frecuencia	Porcentaje
muy mal	6	8%
mal	12	16%
neutro	30	39%
bien	22	29%
muy bien	6	8%
Total	**76**	**100%**

Tabla 33: Resultados observacionales, Gestión del tiempo

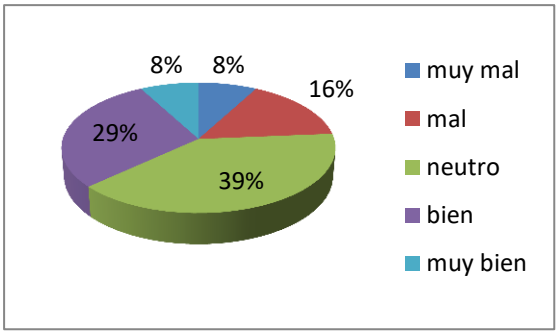


Ilustración 61: Resultados observacionales, Gestión del tiempo

Los resultados del parámetro Coordinación técnica en el juego colaborativo expresan que el 78% de los niños se comunicaron para ejecutar de forma sincronizada las funcionalidades disponibles del juego (captura del personaje), dividido en el 54% y 24% para las dos escalas máximas (bien y muy bien).

Escala	Frecuencia	Porcentaje
muy mal	4	5%
mal	2	3%
neutro	11	14%
bien	41	54%
muy bien	18	24%
Total	**76**	**100%**

Tabla 34: Resultados observacionales, Coordinación técnica

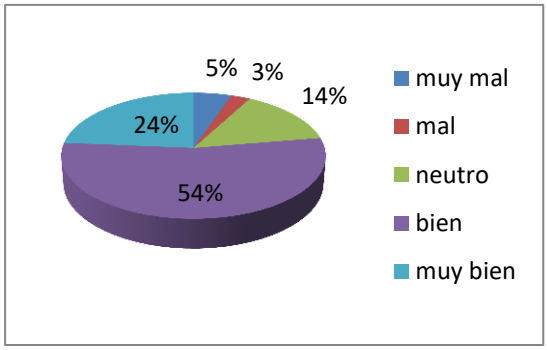


Ilustración 62: Resultados observacionales, Coordinación técnica

Los resultados del parámetro Interacción recíproca en el juego colaborativo expresan que el 59% de los niños se animaron entre ellos y contribuyeron con opiniones y perspectivas para finalizar el juego, dividido en el 33% y 26% para las dos escalas máximas (bien y muy bien).

Escala	Frecuencia	Porcentaje
muy mal	8	11%
mal	4	5%
neutro	19	25%
bien	25	33%
muy bien	20	26%
Total	**76**	**100%**

Tabla 35: Resultados observacionales, Interacción recíproca

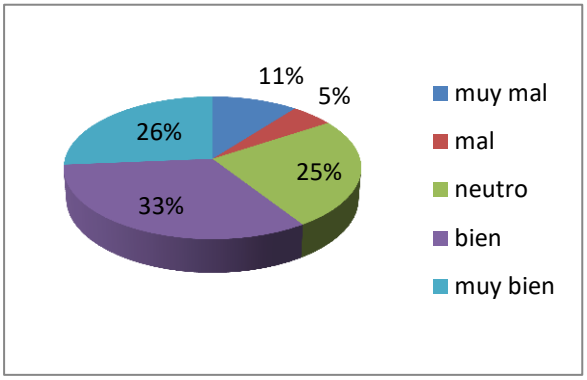


Ilustración 63: Resultados observacionales, Interacción recíproca

El ultimo parámetro Orientación de tareas individuales dio como resultado en el juego colaborativo que el 95% de los niños participaron activamente en la búsqueda de la solución para finalizar el juego, dividido en el 57% y 38% para las dos escalas máximas (bien y muy bien).

Escala	Frecuencia	Porcentaje
muy mal	0	0%
mal	2	3%
neutro	2	3%
bien	43	57%
muy bien	29	38%
Total	**76**	**100%**

Tabla 36: Resultados observacionales, Orientación de tareas individuales

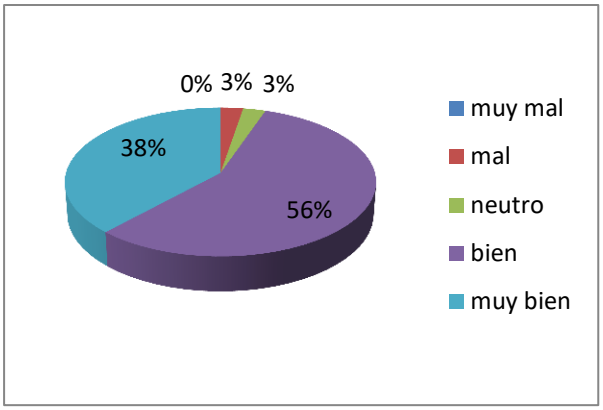


Ilustración 64: Resultados observacionales, Orientación de tareas individuales

Al igual que la rúbrica observacional anterior se procedió a calcular el índice Kappa para evaluar la concordancia observada entre los dos examinadores (observador 1 y 2).

Kappa	Grado de acuerdo	Valor obtenido
< 0	sin acuerdo	
0 - 0,2	insignificante	
0,2 - 0,4	bajo	
0,4 - 0,6	moderado	
0,6 - 0,8	bueno	0,68
0,8 - 1	muy bueno	

Tabla 37: Resultados índice Kappa, observación de la comunicación

El valor 0.68 señala que los datos registrados por el observador uno y dos tiene un grado moderado de concordancia.

5.4. Discusión

A continuación se plantea la discusión de los resultados obtenidos tanto de los cuestionarios como de las plantillas observacionales.

5.4.1. Resultados de los cuestionarios

De forma general las dos versiones del juego competitivo y colaborativo arrojaron resultados altos de satisfacción lo que implica que las dos modalidades de juego captaron la atención de los niños.

Los resultantes del primer cuestionario que valoran de forma independiente cada versión de juego (ver tabla 5) son respaldados por el segundo cuestionario que comparar dichas versiones (ver tabla 6), por lo que se concluye, el juego más fácil para los niños fue el competitivo, con una diferencia no muy alta referente al colaborativo,

más de la mitad de la población jugó mejor en la colaboración, sin embargo, las dos modalidades favorecieron de forma positiva su estado anímico ya que se divirtieron en los dos escenarios. Respecto al manejo del juego con una diferencia pequeña la facilidad se inclina para el juego competitivo al igual que volver a jugar con sus amigos tanto dentro y fuera de clases.

Se puede enfatizar que existen tres unidades de clase en que les gustaría a los niños nuevamente jugar, como es matemáticas, castellano y valenciano, unidades que fueron muy nombradas por los niños en ambas versiones.

En cuanto a las opiniones de los niños obtenidos del cuestionario, estas expresaron sugerencias valiosas tanto de mejora del juego como de satisfacción. Entre las más destacadas están que se disponga de más proyectiles y personajes de captura, que se admita jugar con más personas y que se amplié el tiempo y espacio físico del juego, mencionando que para el experimento el espacio físico y el tiempo total del juego que esta dado en función de los personajes de captura se limitó debido a la disponibilidad de tiempo.

Con los resultados obtenidos y con las pruebas estadísticas realizadas (ver tabla 16) se concluye que no existe una diferencia significativa entre las dos modalidades de juego, sin embargo de forma independiente cada modalidad de juego fue muy interesante para cada niño. La ausencia de la diferencia estadística pudo haberse dado por varios factores que incidieron para que los resultados sean similares en ambas versiones, entre ellos, el número de población y el tiempo disponible. Sería interesante como trabajo futuro poder realizar el experimento con más niños en un espacio físico extenso como una habitación u aula completa y disponer de más personajes para incrementar el tiempo total de juego.

5.4.2. Resultados observacionales

Los tabla 27 demuestra que el afecto, la actividad física, interacción social y uso de la herramienta tiene una diferencia estadística entre las dos modalidades de juego y confirman que:

- Ambas modalidades de juego incrementan en la mayoría de los niños su estado anímico, iniciando con un afecto plano y culminan el juego entre alegres y eufóricos. Sin embrago, la colaboración influye de forma más positiva en el afecto del niño (ver tabla 21).

- El espacio físico disponible en el juego pudo haber limitado a los niños a realizar más movimiento. A pesar de ello y comparado con el competitivo, la colaboración con mayor frecuencia incentiva a los niños a moverse en el transcurso del juego (ver tabla 22).
- Los niños jugando de forma competitiva se inclinan por concentrarse individualmente para alcanzar la meta, que es ganar el juego. Caso contrario sucede con la colaboración, que influye desde el inicio del juego a que los niños establezcan una comunicación para conseguir un objetivo común y por lo tanto el triunfo o la derrota sea alcanzada en conjunto (ver tabla 23).
- La modalidad colaborativa motiva a los niños a explorar la herramienta con más interés (ver tabla 24). Esto puede darse por que en la colaboración se requiere que ambas partes manipulen de forma sincronizada las funcionalidades del juego, lo que puede aumentar su curiosidad por la herramienta.

La satisfacción de los niños medible por comentarios de la actividad son similares en cada modalidad de juego (ver tabla 25), por ende no existe una diferencia significativa. La mayoría de los niños no emitieron ningún comentario, resultados que pueden ser mejorados por dos factores, el espacio y tiempo de duración del juego. Al ampliar los dos factores, se captaría de mejor forma la atención de los niños para motivarlos hablar y también los observadores tendrían más tiempo para captar los comentarios y diferenciarlos de los tres instantes dados, al inicio, intermedio y final del juego.

Finalmente en la versión colaborativa respecto a la comunicación se concluye:

- En una escala considerable los niños realizaron contribuciones comprensibles y crearon un flujo ordenado de comunicación (ver tabla 28 y 29), lo que implica que mantuvieron una buena comunicación entre ellos para en equipo alcanzar el éxito del juego.
- La agrupación de información, el nivel consenso, la división de tareas y la interacción reciproca no tuvieron gran protagonismo en el juego (ver tabla 30, 31, 32 y 35), factores que pueden ser mejorados al aumentar la complejidad del mismo.
- La terminación del juego colaborativo no depende del tiempo, sino de la captura de los personajes, lo que influenció en los niños a que no supervisen el tiempo (ver tabla 33). Una alternativa a ello para obtener resultados más precisos, es el

crear una competición entre parejas de niños, donde la pareja ganadora será aquella que termine el juego en el menor tiempo.

- La coordinación técnica tuvo un protagonismo muy bueno en los niños (ver tabla 34), ya que les motivo a utilizar sus conocimientos y habilidades para manipular y coordinar las funcionalidades principales de la colaboración.
- La orientación de tareas individuales tiene gran relevancia en la colaboración (ver tabla 36), ya que casi toda la población participo en el juego de forma activa mostrando interés en sus tareas y disfrutando del trabajo cooperativo involucrado.

6. Conclusiones y trabajos futuros

Tras finalizar el presente proyecto, se describe las conclusiones obtenidas:

- Los juegos competitivos/colaborativos con realidad aumentada son intrínsecamente satisfactorios en los niños, ya que generan en ellos emociones positivas como, entusiasmo, felicidad y curiosidad, entre otros, factores que mejoran el estado anímico del participante, lo que ayudará a aumentar el grado de implicación de los niños respecto a una asignatura.
- La versión de juego colaborativo promueve con mayor impacto el afecto emocional, la actividad física, la interacción social y el interés comparado al competitivo. Este puede darse por la lógica que tiene el juego para completarlo, que consiste en forzar a los niños a colaborar de forma sincronizada para la captura de los personajes.
- El juego colaborativo representa una alternativa viable para la adquisición de habilidades de comunicación, ya que esta, elimina el juego individualizado y motiva a los niños a crear diálogos e interactuar entre ellos para conseguir un objetivo común.
- Nuestro prototipo de juego no está limitado por los objetos que intervienen en la escena como personajes y proyectiles, lo que significa que el usuario tiene la libertad de diseñar el juego con los objetos 3D según su escenario.
- La tecnología de realidad aumentada sin marcadores está preparada para implementar escenarios de juego colaborativo con aplicaciones educativas

Finalmente se da conocer los trabajos futuros como ampliación del proyecto:

- Para tener mayor precisión de los resultados, se podría realizar una evaluación experimental que involucre mayor población de niños, mayor tiempo en la duración del juego y con un espacio físico más amplio. El incremento de los dos últimos factores, permitirán en los niños captar de mejor forma su atención para motivarlos a expresar comentarios y también facilitará a los observadores en el proceso de evaluación la captura de datos para los tres instantes dados, al inicio, a mediados y final del juego.
- Ampliar la aplicación para que disponga de más funcionalidades como, se implemente varias opciones de juego, que el número de participantes del juego sea configurable, se agregue efectos especiales, se aumente la complejidad de juego mediante niveles, otros.

- Estudiar la aplicación de esta tecnología en el campo de la ludoterapia hospitalaria dado que permitiría realizar juegos con realidad aumentada en las propias habitaciones hospitalarias sin necesidad de incorporar marcadores a las mismas.
- Estudiar el impacto la aplicación de esta tecnología en la efectividad del aprendizaje en un contexto real con usuarios en edad infantil.

7. Bibliografía

[1] F. Garcia, S. Jurdi, J. Jaen, and V. Nacher, “Evaluating a tactile and a tangible multi-tablet gamified quiz system for collaborative learning in primary education,” *Comput. Educ.*, vol. 123, pp. 65–84, Aug. 2018.

[2] S. García Sastre, M. Idrissi Cao, A. Ortega Arranz, and E. Gómez Sánchez, “Uso de la colaboración y la gamificación en MOOC: un análisis exploratorio,” *RIED. Rev. Iberoam. Educ. a Distancia*, vol. 21, no. 2, p. 263, Mar. 2018.

[3] J. He *et al.*, “The research and application of the augmented reality technology,” in *2017 IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, 2017, pp. 496–501.

[4] J. Fombona Cadavieco, M. Á. Pascual Sevillano, and M. F. Ferreira Amador, “Realidad aumentada, una evolución de las aplicaciones de los dispositivos móviles,” *Pixel-Bit. Rev. Medios y Educ.*, pp. 197–210, 2012.

[5] R. S. C. Espinosa and J. L. Eguia, “Gamificación en aulas universitarias,” *Bellaterra: Institut de la Comunicació*, 2016.

[6] M. Tsay and M. Brady, “A Case Study of Cooperative Learning and Communication Pedagogy: Does Working in Teams Make a Difference?,” *J. Scholarsh. Teach. Learn.*, vol. 10, no. 2, pp. 78–89, 2012.

[7] S. Deterding, D. Dixon, R. Khaled, and L. Nacke, “From Game Design Elements to Gamefulness: Defining ‘Gamification,’” in *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments*, 2011, pp. 9–15.

[8] S. A. Kocadere and Ş. Çağlar, “The design and implementation of a gamified assessment,” *J. e-learning Knowl. Soc.*, vol. 11, no. 3, pp. 85–99, 2015.

[9] S. Irshad and D. Rohaya Bt Awang Rambli, “User experience of mobile augmented reality: A review of studies,” in *2014 3rd International Conference on User Science and Engineering (i-USEr)*, 2014, pp. 125–130.

[10] C. Prendes Espinosa, “Realidad aumentada y educación: análisis de experiencias prácticas,” *Pixel-Bit. Rev. Medios y Educ.*, pp. 187–203, 2015.

[11] J. Barroso Osuna and O. M. Gallego Pérez, “La realidad aumentada y su aplicación en la educación superior,” *Rev. del Salomé, Rev. Dominic. Educ.*, vol. 1, pp. 111–127, 2016.

[12] N. F. Gazcón, J. I. Larregui, and S. M. Castro, “La Realidad Aumentada como complemento motivacional,” *Rev. Iberoam. Tecnol. en Educ. y Educ. en Tecnol.*, vol. 0, no. 17 SE-Artículos originales, pp. 7–15, Jun. 2016.

[13] E. D. M. Pérez and L. Villalustre Martínez, “Juegos perceptivos con realidad aumentada para trabajar contenido científico,” *Educ. Formação Tecnol. - ISSN 1646-933X*, vol. 10, no. 1, pp. 36–46, 2017.

[14] A. Pérez and M. C. Jerez Mesquida, “Estudio de APPs de realidad aumentada para su uso en campos de aprendizaje en un entorno natural,” *Edutec. Rev. Electrónica Tecnol. Educ.*, vol. 0, no. 62, p. 19--33 (a370), 2018.

[15] J. L. Vega Hernández, B. A. Rico Jiménez, and C. Hernández Nava, “Aplicación móvil para ayudar al aprendizaje de niños autistas,” *Res. Comput. Sci.*, no. 75, pp. 103–111, 2014.

[16] R. M. Nubia, G. R. Fabian, R. A. Wilson, and P. B. Wilmer, “Development of a mobile application in augmented reality to improve the communication field of autistic children at a Neurorehabilitar Clinic,” in *2015 Workshop on Engineering Applications - International Congress on Engineering (WEA)*, 2015, pp. 1–6.

[17] H. Ivana and M. Romero Pazmiño, “Uso de nuevas tecnologías TICS - Realidad Aumentada para tratamiento de niños TEA un diagnóstico inicial,” *Rev. CienciAmérica*, vol. 6, no. 3, pp. 131–137, 2017.

[18] R. M. Martín Sabarís and G. Brossy Scaringi, “La realidad aumentada aplicada al aprendizaje en personas con Síndrome de Down: un estudio exploratorio,” La Laguna, Tenerife, Jun. 2017.

[19] R. Xaime Rivas, “BandApp: Monitorización del ritmo respiratorio con realidad aumentada,” Sep. 2016.

[20] Z. Barsom, M. Graafland, and M. Schijven, “Systematic review on the effectiveness of augmented reality applications in medical training,” *Surg. Endosc.*, vol. 30, no. 10, pp. 4174–4183, Oct. 2016.

[21] J. Ejarque García, “Realidad aumentada y virtual en entornos hospitalarios,” Universidad Politécnica de Catalunya, 2017.

[22] N. B. Mora, I. C. Sánchez, and R. E. Fernández, “Aplicación de la Realidad Aumentada para la asistencia en enseñanza médica,” Universidad Complutense de Madrid, 2017.

[23] I. Bonilla Gutiérrez, M. O. Mendoza Gutiérrez, and L. J. Hernández Sierra, “Integración de una Interfaz de Realidad Aumentada para Estimulación Visual de los Usuarios en Sistemas Robóticos para Terapia Ocupacional,” *Memorias del Congr. Nac. Ing. Biomédica*, vol. 4, no. 1, pp. 154–157, 2017.

[24] “Una App con Realidad Aumentada para fomentar el compromiso de trabajadores y clientes,” *Esmartcity, todo sobre ciudades inteligentes*, 2018. [Online]. Available: https://www.esmartcity.es/2018/01/18/app-realidad-aumentada-fomentar-compromiso-trabajadores-clientes.

[25] “Juglar, realidad aumentada educativa.” [Online]. Available: http://www.esmuik.es/juglareducativa/.

[26] W. Knight, “Connectivity Microsoft Researchers Are Working on Multi-Person Virtual Reality,” 2015. [Online]. Available: https://www.technologyreview.com/s/542341/microsoft-researchers-are-working-on-multi-person-virtual-reality/.

[27] J. C. Vázquez Parra, A. Encinas Soto, and E. García Béjar, “Herramientas para desarrollar el reconocimiento ético en estudiantes universitarios: Una experiencia de aprendizaje con Pokémon Go,” *Revista de Investigación Apuntes Universitarios*, p. 15, 2017.

[28] K. Werbach and D. Hunter, *For the Win: How game thinking can revolutionize Your business*, Wharton Di. 2012.

[29] E. Szewkis *et al.*, “Collaboration within large groups in the classroom,” *Int. J. Comput. Collab. Learn.*, vol. 6, no. 4, pp. 561–575, 2011.

[30] C. Keith, *Agile Game Development with Scrum*, Education,. 2010.

[31] “Unity, plataforma de desarrollo de juegos.” [Online]. Available: https://unity3d.com/es.

[32] “Photon Unity Networking, paquete de Unity para juegos multijugador.” [Online]. Available: https://doc.photonengine.com/en-us/pun/current/getting-started/pun-intro.

[33] “Placenote, SDK de realidad aumentada.” [Online]. Available:

https://placenote.com/documentation/.

[34] R. González Barrón, I. Montoya Castilla, M. M. Casullo, and J. Bernabéu Verdú, “Relación entre estilos y estrategias de afrontamiento y bienestar psicológico en adolescentes,” *Revista Anual de Psicología*, vol. 14, no. 2, pp. 363–368, 2002.

[35] A. Meier, H. Spada, and N. Rummel, “A rating scheme for assessing the quality of computer-supported collaboration processes,” *Int. J. Comput. Collab. Learn.*, vol. 2, no. 1, pp. 63–86, 2007.

Anexo 1: Archivo de inicialización configuracion.json


```
{"games": [
  {
    "id": 1,
    "name": "emociones",
    "actorsType": [
      {
        "id": 1,
        "name": "emociones_actor_enojado_01",
        "lifeInit": 10,
        "randomPosition": true,
        "icon": "emociones_actor_enojado_01",
        "timeToMovi": 20,
        "timeCollaborative": 10,
        "resultsType": [
          {"projectileType": 1,"units": 8},
          {"projectileType": 2,"units": 2},
          {"projectileType": 3,"units": 4},
          {"projectileType": 4, "units": 4},
          {"projectileType": 5, "units": 8},
          {"projectileType": 6,"units": 4},
          {"projectileType": 7,"units": 6},
          {"projectileType": 8,"units": 6},
          {"projectileType": 9,"units": 8},
          {"projectileType": 10,"units": 2},
          {"projectileType": 11,"units": 2},
          {"projectileType": 13,"units": 8},
          {"projectileType": 17,"units": 4},
          {"projectileType": 19,"units": -2},
          {"projectileType": 20,"units": -2},
          {"projectileType": 21,"units": 6}
        ]
      },
      {
        "id": 2,
        "name": "emociones_actor_enojado_02",
        "lifeInit": 10,
        "randomPosition": true,
        "icon": "emociones_actor_enojado_02",
        "timeToMovi": 20,
        "timeCollaborative": 10,
        "resultsType": [
          {"projectileType": 1,"units": 8},
          {"projectileType": 2,"units": 2},
          {"projectileType": 3,"units": 4},
          {"projectileType": 4, "units": 4},
          {"projectileType": 5, "units": 8},
          {"projectileType": 6,"units": 4},
          {"projectileType": 7,"units": 6},
          {"projectileType": 8,"units": 6},
          {"projectileType": 9,"units": 8},
          {"projectileType": 10,"units": 2},
          {"projectileType": 11,"units": 2},
          {"projectileType": 13,"units": 8},
          {"projectileType": 17,"units": 4},
          {"projectileType": 19,"units": -2},
          {"projectileType": 20,"units": -2},
          {"projectileType": 21,"units": 6}
        ]
      },
```

```
{
  "id": 3,
  "name": "emociones_actor_triste_01",
  "lifeInit": 15,
  "randomPosition": true,
  "icon": "emociones_actor_triste_01",
  "timeToMovi": 15,
  "timeCollaborative": 10,
  "resultsType": [
    {"projectileType": 1,"units": 8},
    {"projectileType": 2,"units": 2},
    {"projectileType": 3,"units": 4},
    {"projectileType": 4, "units": 4},
    {"projectileType": 5, "units": 8},
    {"projectileType": 6,"units": 4},
    {"projectileType": 7,"units": 6},
    {"projectileType": 8,"units": 6},
    {"projectileType": 9,"units": 8},
    {"projectileType": 10,"units": 2},
    {"projectileType": 11,"units": 2},
    {"projectileType": 13,"units": 8},
    {"projectileType": 17,"units": 4},
    {"projectileType": 19,"units": -2},
    {"projectileType": 20,"units": -2},
    {"projectileType": 21,"units": 6}
  ]
},
{
  "id": 4,
  "name": "emociones_actor_triste_02",
  "lifeInit": 15,
  "randomPosition": true,
  "icon": "emociones_actor_triste_02",
  "timeToMovi": 15,
  "timeCollaborative": 10,
  "resultsType": [
    {"projectileType": 1,"units": 8},
    {"projectileType": 2,"units": 2},
    {"projectileType": 3,"units": 4},
    {"projectileType": 4, "units": 4},
    {"projectileType": 5, "units": 8},
    {"projectileType": 6,"units": 4},
    {"projectileType": 7,"units": 6},
    {"projectileType": 8,"units": 6},
    {"projectileType": 9,"units": 8},
    {"projectileType": 10,"units": 2},
    {"projectileType": 11,"units": 2},
    {"projectileType": 13,"units": 8},
    {"projectileType": 17,"units": 4},
    {"projectileType": 19,"units": -2},
    {"projectileType": 20,"units": -2},
    {"projectileType": 21,"units": 6}
  ]
},
{
  "id": 5,
  "name": "emociones_actor_feliz_01",
  "icon": "emociones_actor_feliz_01",
  . . .
},
{
```

```
    "id": 6,
    "name": "emociones_actor_feliz_02",
    "icon": "emociones_actor_feliz_02",
    . . .
  }
],
"projectilesType": [
  {
    "id": 1,
    "name": "emociones_projectile_corazon_redondo",
    "icon": "emociones_projectile_corazon_redondo",
    "rotationX": 0,"rotationY": 0,"rotationZ": 0
  },
  {
    "id": 2,
    "name": "emociones_emoji_02",
    "icon": "emociones_emoji_02",
    "rotationX": 0,"rotationY": 180,"rotationZ": 0
  },
  {
    "id": 3,
    "name": "emociones_emoji_01",
    "icon": "emociones_emoji_01",
    "rotationX": 0,"rotationY": 180,"rotationZ": 0
  },
  {
    "id": 4,
    "name": "emociones_emoji_07",
    "icon": "emociones_emoji_07",
    "rotationX": -90, "rotationY": 180,"rotationZ": 0
  },
  {
    "id": 5,
    "name": "emociones_projectile_corazon_rojo",
    "icon": "emociones_projectile_corazon_rojo",
    "rotationX": 0,"rotationY": 0,"rotationZ": 0
  },
  {
    "id": 6,
    "name": "emociones_emoji_09",
    "icon": "emociones_emoji_09",
    "rotationX": -90,"rotationY": 180,"rotationZ": 0
  },
  {
    "id": 7,
    "name": "emociones_emoji_21",
    "icon": "emociones_emoji_21",
    "rotationX": -90,"rotationY": 180,"rotationZ": 0
  },
  {
    "id": 8,
    "name": "emociones_emoji_22",
    "icon": "emociones_emoji_22",
    "rotationX": -90,"rotationY": 180,"rotationZ": 0
  },
  {
    "id": 9,
    "name": "emociones_projectile_corazon_tallado",
    "icon": "emociones_projectile_corazon_tallado",
    "rotationX": 0,"rotationY": 0,"rotationZ": 0
  },
```

```
{
  "id": 10,
  "name": "emociones_emoji_18",
  "icon": "emociones_emoji_18",
  "rotationX": -90,"rotationY": 180,"rotationZ": 0
},
{
  "id": 11,
  "name": "emociones_emoji_11",
  "icon": "emociones_emoji_11",
  "rotationX": -90,"rotationY": 180,"rotationZ": 0
},
{
  "id": 12,
  "name": "emociones_emoji_05",
  "icon": "emociones_emoji_05",
  "rotationX": -90,"rotationY": 180,"rotationZ": 0
},
{
  "id": 13,
  "name": "emociones_projectile_bomba",
  "icon": "emociones_projectile_bomba",
  "rotationX": 0,"rotationY": 0,"rotationZ": 0
},
{
  "id": 14,
  "name": "emociones_emoji_13",
  "icon": "emociones_emoji_13",
  "rotationX": -90,"rotationY": 180,"rotationZ": 0
},
{
  "id": 15,
  "name": "emociones_emoji_12",
  "icon": "emociones_emoji_12",
  "rotationX": -90,"rotationY": 180,"rotationZ": 0
},
{
  "id": 16,
  "name": "emociones_emoji_06",
  "icon": "emociones_emoji_06",
  "rotationX": -90,"rotationY": 180,"rotationZ": 0
},
{
  "id": 17,
  "name": "emociones_projectile_sonrisa_cubo",
  "icon": "emociones_projectile_sonrisa_cubo",
  "rotationX": 0,"rotationY": 0,"rotationZ": 0
},
{
  "id": 18,
  "name": "emociones_emoji_15",
  "icon": "emociones_emoji_15",
  "rotationX": -90,"rotationY": 180,"rotationZ": 0
},
{
  "id": 19,
  "name": "emociones_emoji_17",
  "icon": "emociones_emoji_17",
  "rotationX": -90,"rotationY": 180,"rotationZ": 0
},
{
```

```
        "id": 20,
        "name": "emociones_emoji_19",
        "icon": "emociones_emoji_19",
        "rotationX": -90,"rotationY": 180, "rotationZ": 0
      },
      {
        "id": 21,
        "name": "emociones_projectile_estrella",
        "icon": "emociones_projectile_estrella",
        "rotationX": 0,"rotationY": 0,"rotationZ": 0
      }
    ]
  }
 ]
}
```

Anexo 2: Script Configuration.cs

```
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.XR.iOS;
using System.Runtime.InteropServices;
using Newtonsoft.Json.Linq;
using Newtonsoft.Json;
using UnityEngine.UI;

namespace ExitGames.Upv.ProyectoTfm
{

public class Configuration : MonoBehaviour, PlacenoteListener{

	//Serializer Paneles
	[SerializeField] GameObject panelModalsMaps;
	[SerializeField] GameObject panelActionsInit;
	[SerializeField] GameObject btnConecction;
	[SerializeField] Text mLabelText;
	[SerializeField] Text mLabelTextMapLoad;
	[SerializeField] GameObject mListElement;
	[SerializeField] RectTransform mListContentParent;
	[SerializeField] ToggleGroup mToggleGroup;
	[SerializeField] Material mShapeMaterial;

	//Placenote, arkit
	private UnityARSessionNativeInterface mSession;
	private bool mFrameUpdated = false;
	private UnityARImageFrameData mImage = null;
	private UnityARCamera mARCamera;
	private bool mARKitInit = false;
	private List<ShapeInfo> shapeInfoList = new List<ShapeInfo> ();
	private List<GameObject> shapeObjList = new List<GameObject> ();
	private BoxCollider mBoxColliderDummy;
	private SphereCollider mSphereColliderDummy;
	private CapsuleCollider mCapColliderDummy;

	//Placenote, mapa seleccionado
	private LibPlacenote.MapInfo mSelectedMapInfo;
	private string mSelectedMapId {
		get {
			return mSelectedMapInfo != null ?
			mSelectedMapInfo.placeId :
			null;
		}
	}

	//Placenote, puntos del mapa
	private LibPlacenote.PNFeaturePointUnity[] mapPoints;

	static public Configuration configurationInstance;

	//*******************************
	// Inicializacion de Placenote
	//*******************************
	void Start ()
	{
		configurationInstance = this;
		panelModalsMaps.SetActive (false);
		btnConecction.SetActive (false);
```

```
        //Inicialización de placenote
        Input.location.Start ();
        mSession =
UnityARSessionNativeInterface.GetARSessionNativeInterface ();
        UnityARSessionNativeInterface.ARFrameUpdatedEvent +=
        ARFrameUpdated;
        StartARKit ();
        //FeaturesVisualizer.EnablePointcloud ();
        LibPlacenote.Instance.RegisterListener (this);
}

private void ARFrameUpdated (UnityARCamera camera)
{
        mFrameUpdated = true;
        mARCamera = camera;
}

private void InitARFrameBuffer ()
{
        mImage = new UnityARImageFrameData ();

        int yBufSize = mARCamera.videoParams.yWidth *
        mARCamera.videoParams.yHeight;
        mImage.y.data = Marshal.AllocHGlobal (yBufSize);
        mImage.y.width = (ulong)mARCamera.videoParams.yWidth;
        mImage.y.height = (ulong)mARCamera.videoParams.yHeight;
        mImage.y.stride = (ulong)mARCamera.videoParams.yWidth;

        int vuBufSize = mARCamera.videoParams.yWidth *
        mARCamera.videoParams.yWidth/2;
        mImage.vu.data = Marshal.AllocHGlobal (vuBufSize);
        mImage.vu.width = (ulong)mARCamera.videoParams.yWidth/2;
        mImage.vu.height = (ulong)mARCamera.videoParams.yHeight/2;
        mImage.vu.stride = (ulong)mARCamera.videoParams.yWidth;

        mSession.SetCapturePixelData (true, mImage.y.data,
        mImage.vu.data);
}

void Update ()
{
        if (mFrameUpdated) {
                mFrameUpdated = false;
                if (mImage == null) {
                        InitARFrameBuffer ();
                }

                if (mARCamera.trackingState ==
                ARTrackingState.ARTrackingStateNotAvailable) {
                        return;
                } else if (!mARKitInit) {
                        mARKitInit = true;
                        mLabelTextMapLoad.text = "ARKit Initialized";
                }

                Matrix4x4 matrix = mSession.GetCameraPose ();
                Vector3 arkitPosition =
                PNUtility.MatrixOps.GetPosition (matrix);
                Quaternion arkitQuat =
                PNUtility.MatrixOps.GetRotation (matrix);
```

```
            LibPlacenote.Instance.SendARFrame (mImage,
            arkitPosition, arkitQuat,
            mARCamera.videoParams.screenOrientation);
      }
}

private void StartARKit ()
{
      mLabelTextMapLoad.text="Initializing ARKit";
      Application.targetFrameRate = 60;
      ARKitWorldTrackingSessionConfiguration config = new
      ARKitWorldTrackingSessionConfiguration ();
      config.planeDetection = UnityARPlaneDetection.Horizontal;
      config.alignment =
      UnityARAlignment.UnityARAlignmentGravity;
      config.getPointCloudData = true;
      config.enableLightEstimation = true;
      mSession.RunWithConfig (config);
}

public void OnPose (Matrix4x4 outputPose, Matrix4x4 arkitPose)
{}


public void OnStatusChange (LibPlacenote.MappingStatus
prevStatus, LibPlacenote.MappingStatus currStatus)
{
      if (currStatus == LibPlacenote.MappingStatus.RUNNING &&
      prevStatus == LibPlacenote.MappingStatus.LOST) {
            mLabelText.text="Localized";
            LoadShapesJSON (mSelectedMapInfo.userData);
      } else if (currStatus == LibPlacenote.MappingStatus.RUNNING
      && prevStatus == LibPlacenote.MappingStatus.WAITING) {
            mLabelText.text="Mapping";
      } else if (currStatus == LibPlacenote.MappingStatus.LOST) {
            mLabelText.text="Searching for position lock";
      } else if (currStatus ==
      LibPlacenote.MappingStatus.WAITING) {
            if (shapeObjList.Count != 0) {
                  ClearShapes ();
            }
      }
}

//*******************************
// Acciones de carga de objetos cuando se crea el mapa
//*******************************
private GameObject ShapeFromInfo(ShapeInfo info)
{
      GameObject shape = GameObject.CreatePrimitive
      ((PrimitiveType)info.shapeType);
      shape.transform.position = new Vector3(info.px, info.py,
      info.pz);
      shape.transform.rotation = new Quaternion(info.qx, info.qy,
      info.qz, info.qw);
      shape.transform.localScale = new Vector3 (0.05f, 0.05f,
      0.05f);
      shape.GetComponent<MeshRenderer> ().material =
      mShapeMaterial;
      return shape;
}
```

```
private void ClearShapes () {
      foreach (var obj in shapeObjList) {
            Destroy (obj);
      }
      shapeObjList.Clear ();
      shapeInfoList.Clear ();
}


private void LoadShapesJSON (JToken mapMetadata)
{
      ClearShapes ();
      if (mapMetadata is JObject && mapMetadata ["shapeList"] is
      JObject) {
            ShapeList shapeList = mapMetadata
            ["shapeList"].ToObject<ShapeList> ();
            if (shapeList.shapes == null) {
                  Debug.Log ("no shapes dropped");
                  return;
            }

            foreach (var shapeInfo in shapeList.shapes) {
                  shapeInfoList.Add (shapeInfo);
                  GameObject shape = ShapeFromInfo(shapeInfo);
                  shapeObjList.Add(shape);
            }
      }
}

//*******************************
// Acciones de carga de mapas
//*******************************
public void OnListMapClick ()
{
      if (!LibPlacenote.Instance.Initialized()) {
            Debug.Log ("Tfm, SDK not yet initialized");
            ToastManager.ShowToast ("Tfm, SDK not yet
            initialized", 2f);
            return;
      }

      foreach (Transform t in mListContentParent.transform) {
            Destroy (t.gameObject);
      }

      panelModalsMaps.SetActive (true);
      panelActionsInit.SetActive (false);

      LibPlacenote.Instance.ListMaps ((mapList) => {
            foreach (LibPlacenote.MapInfo mapId in mapList) {
                  if (mapId.userData != null) {
                        Debug.LogError (mapId.userData.ToString
                        (Formatting.None));
                  }
                  AddMapToList (mapId);
            }
      });
}
```

```
void AddMapToList (LibPlacenote.MapInfo mapInfo)
{
      GameObject newElement = Instantiate (mListElement) as
      GameObject;
      MapInfoElement listElement =
      newElement.GetComponent<MapInfoElement> ();
      listElement.Initialize (mapInfo, mToggleGroup,
      mListContentParent, (value) => {
            OnMapSelected (mapInfo);
      });
}

void OnMapSelected (LibPlacenote.MapInfo mapInfo)
{
      mSelectedMapInfo = mapInfo;
      mLabelTextMapLoad.text = "Mapa ID: " + mSelectedMapId;
      btnConecction.SetActive (true);
      panelActionsInit.SetActive (true);
      panelModalsMaps.SetActive (false);
}

public void OnLoadMapClicked ()
{
      if (!LibPlacenote.Instance.Initialized()) {
            Debug.Log ("SDK not yet initialized");
            ToastManager.ShowToast ("SDK not yet initialized",
            2f);
            return;
      }

      mLabelTextMapLoad.text = "Loading Map ID: " +
      mSelectedMapId;
      LibPlacenote.Instance.LoadMap (mSelectedMapId,
            (completed, faulted, percentage) => {
                  if (completed) {
                        LibPlacenote.Instance.StartSession ();
                        // Puntos aleatorios del mapa
                        mapPoints = LibPlacenote.Instance.GetMap
                        ();
                        if (mapPoints != null) {
                              //Se une el usuario al Room
                              panelActionsInit.SetActive (false);
                              Room.roomInstance.conecction ();
                              mLabelTextMapLoad.text = "Map ID: "
                              + mSelectedMapId;
                              mLabelText.text = "Loaded ID: " +
                              mSelectedMapId;
                        } else {
                              mLabelTextMapLoad.text = "Loaded
                              ID: " + mSelectedMapId + " vacio";
                              OnExitClick ();
                        }

                  } else if (faulted) {
                        mLabelTextMapLoad.text = "Failed to load
                        ID: " + mSelectedMapId;
                  }
            }
      );
}
```

```
        public void OnCancelClick ()
        {
              panelModalsMaps.SetActive (false);
              panelActionsInit.SetActive (true);
        }

        public void OnExitClick ()
        {
              LibPlacenote.Instance.StopSession ();
              panelActionsInit.SetActive (true);
        }

        public Vector3 getPointRandomMap(){
              Vector3 pointRandom = new Vector3();
              if (mapPoints != null) {
                    int random = UnityEngine.Random.Range(0,
                    mapPoints.Length-1);
                    pointRandom.x= mapPoints [random].point.x;
                    pointRandom.y = mapPoints [random].point.y;
                    pointRandom.z = -mapPoints [random].point.z;

              }
              return pointRandom;
        }
}
}
```

Anexo 3: Script Room.cs

```
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

namespace ExitGames.Upv.ProyectoTfm
{
public class Room : Photon.PunBehaviour
{
	//Configuración de Photon
	byte maxPlayersPerRoom = 4;
	string gameVersion = "1";
	bool isConnecting = false;

	//Prebafricados y paneles
	public InputField namePlayer;
	public GameObject prefabButtonAddDinamico;
	public GameObject prefabButtonAddDinamicoRecover;

	public GameObject panelRoom;
	public GameObject panelHorizontalGroupProjectile;
	public GameObject panelCenterShowProjectiles;
	public GameObject panelBottomButtonGameInfoPlayer;
	public GameObject panelTopShowProjectileSelec;

	public Text panelTopTextPoints;
	public GameObject panelCenterShowActorsTypePlayer;
	public GameObject panelHorizontalGroupActorsTypePlayer;

	public GameObject panelGameInfoPlayer;
	public GameObject panelGameInfoPlayerButtonCancel;
	public GameObject panelGameInfoPlayerHeadWinner;
	public GameObject panelGameInfoPlayerActors;
	public GameObject panelGameInfoPlayerLists;
	public GameObject prefabPlayerAddDinamico;
	public Text textGameInfoPlayerResult;
	public Text textGameInfoPlayerHeadWinner;
	public Dropdown dropdownTypeGame;

	//Variables cuando finaliza el juego
	private bool isFinalizedGame;

	//Variables de los sonidos del juego
	AudioSource audioSourceGame;
	AudioSource audioSourceAction;

	public GameObject audioSourceGameGameObject;
	public AudioClip sonidoRunProjectile;
	public AudioClip sonidoRecoverProjectile;
	public AudioClip sonidoProjectileWithoutEffect;
	public AudioClip sonidoProjectileWitEffect;
	public AudioClip sonidoEndGame;
	public AudioClip sonidoBackgroundGame;
	public AudioClip sonidoInitCollaborative;

	static public Room roomInstance;

	void Start ()
	{
		roomInstance = this;
```

```
        audioSourceAction = GetComponent<AudioSource>();
        audioSourceGame =
        audioSourceGameGameObject.GetComponent<AudioSource>();
        panelRoom.SetActive (false);
}

//*******************************
// Configuración de Photon
//*******************************

// Inicia la conexión con el servidor de Photon siempre y cuando
este seleccionado un mapa
public void conecction()
{
        isConnecting = true;
        if (PhotonNetwork.connected)
        {
                PhotonNetwork.JoinRandomRoom();
        }
        else
        {
                PhotonNetwork.ConnectUsingSettings(gameVersion);
        }
}

// Ingresa cuando se ha conectado al servidor
public override void OnConnectedToMaster()
{
        if (isConnecting)
        {
                PhotonNetwork.JoinRandomRoom();
        }
}

// Ingresa cuanfo falla la unión a una sala existente
public override void OnPhotonRandomJoinFailed(object[]
codeAndMsg)
{
        PhotonNetwork.CreateRoom(null, new RoomOptions() {
        MaxPlayers = this.maxPlayersPerRoom }, null);
}

// Ingresa cuando nos unimos a una sala
public override void OnJoinedRoom()
{
        PhotonNetwork.playerName = namePlayer.text;
        if (dropdownTypeGame.value == 0)
        {
                Game.gameInstance.updateTypeGame(true);
                textGameInfoPlayerResult.text = "Competitivo";
        }
        else
        {
                Game.gameInstance.updateTypeGame(false);
                textGameInfoPlayerResult.text = "Colaborativo";
        }
        // Inicializa la configuración del juego
        isFinalizedGame = false;
        Game.gameInstance.startGame ();
        panelCenterShowProjectiles.SetActive(false);
        panelTopShowProjectileSelec.SetActive(false);
```

```
        panelCenterShowActorsTypePlayer.SetActive (false);
        panelGameInfoPlayer.SetActive (false);
        panelBottomButtonGameInfoPlayer.SetActive (true);
        panelRoom.SetActive(true);
        audioSourceGame.Play();
}

// Método que sale de la habitación invoca por el botón
public void exitRoom()
{
        PhotonNetwork.LeaveRoom();

}

//Método que se invoca cuando se sale de una habitación
public override void OnLeftRoom()
{
        isConnecting = false;
        Configuration.configurationInstance.OnExitClick ();
        panelRoom.SetActive(false);

        foreach (Transform t in
        panelHorizontalGroupProjectile.transform) {
                Destroy (t.gameObject);
        }

        foreach (Transform t in
        panelHorizontalGroupActorsTypePlayer.transform) {
                Destroy (t.gameObject);
        }

        GameObject[] projectilesGame =
        GameObject.FindGameObjectsWithTag("Player");
        foreach (GameObject projectile in projectilesGame)
        {
                if(projectile.GetComponent<Projectile>().
                projectileTypeRecover == true)
                {
                        Destroy(projectile);
                }
        }
        audioSourceGame.Stop();
}

//*******************************
// Lanzamiento de proyectiles
//*******************************

//Saber cuándo el lanzado y recuperado un proyectil
void Update () {
        if (PhotonNetwork.connected && panelRoom.GetActive())
        {
                if (Input.GetMouseButtonDown(0))
                {
                        RaycastHit hit;
                        Ray ray = Camera.main.ScreenPointToRay
                        (Input.mousePosition);
                        if (Physics.Raycast(ray, out hit))
                        {
                                if (hit.collider.GetComponent
                                <Projectile>())
```

```
                        {
                              if(hit.collider.GetComponent
                              <Projectile>().
                              projectileTypeRecover == true)
                              {
                                    Game.gameInstance.
                                    removeProjectilesTypeRecover(
                                    hit.collider.
                                    GetComponent<Projectile>().
                                    projectileType);
                                    playAudio(2);
                                    Destroy(
                                    hit.collider.gameObject);
                                    addButtonProjectiles();
                              }else {
                                    if (Game.gameInstance.
                                    projectileSelec != null &&
                                    Game.gameInstance.
                                    runProjectile == false &&
                                    hit.collider.GetComponent
                                    <PhotonView>().isMine)
                                    {
                                          Game.gameInstance.
                                          runProjectilePlayer();
                                          addButtonProjectiles();
                                    panelTopShowProjectileSelec.
                                    SetActive(false);
                                    }
                              }
                        }
                  }
            }

            bool finalize = Game.gameInstance.finalizeGameRoom();
            if (finalize == true && isFinalizedGame == false)
            {
                  playAudio(5);
                  isFinalizedGame = true;
                  printInfoGame(true);
                  panelGameInfoPlayerButtonCancel.
                  SetActive(false);
                  panelGameInfoPlayer.SetActive(true);
            }
      }
}

// Contruye todo el menu de los projectiles
public void addButtonProjectiles()
{
      foreach (Transform t in
      panelHorizontalGroupProjectile.transform)
      {
            Destroy(t.gameObject);
      }

      foreach (ProjectileType projectileType in
      Game.gameInstance.projectilesType)
      {
            addButtonProjectile(projectileType, false);
      }
```

```
        foreach (ProjectileType projectileType in
        Game.gameInstance.projectilesTypePlayerRecover)
        {
              addButtonProjectile(projectileType, true);
        }
}

// Agrego un projectile al menu de projectiles
public void addButtonProjectile(ProjectileType projectileType,
bool projectileTypeRecover)
{
        GameObject projectileButton;
        if (projectileTypeRecover == false)
        {
              projectileButton =
              Instantiate(prefabButtonAddDinamico);
              projectileButton.GetComponent<Button>().
              onClick.AddListener(() =>
              buttonProjectileClicked(projectileType));
        }else{
              projectileButton =
              Instantiate(prefabButtonAddDinamicoRecover);
        }

        Sprite projectileIcon = getIconSprite(projectileType.icon);
        projectileButton.GetComponent<Image>().sprite =
        projectileIcon;
        projectileButton.transform.SetParent(
        panelHorizontalGroupProjectile.transform);

}

// Cuando se hace click en los botones de los projectiles
void buttonProjectileClicked(ProjectileType projectileType)
{
        if (Game.gameInstance.projectileSelec != null)
        {
              if(Game.gameInstance.projectileSelec.id ==
              projectileType.id)
              {
                    return;
              }
        }
        Game.gameInstance.showProjectilePlayer(projectileType);
        Sprite projectileIcon = getIconSprite(projectileType.icon);
        panelTopShowProjectileSelec.GetComponent<Image>().sprite =
        projectileIcon;
        panelTopShowProjectileSelec.SetActive(true);
}


// Muestra y oculta el menu de los projectiles
public void showMenuProjectiles()
{
        if (panelCenterShowProjectiles.GetActive ()) {
              panelBottomButtonGameInfoPlayer.SetActive (true);
        } else {
              addButtonProjectiles();
              panelBottomButtonGameInfoPlayer.SetActive (false);
        }
        panelCenterShowProjectiles.SetActive(
```

```
        !panelCenterShowProjectiles.GetActive());
    }

    //*******************************
    // Puntos del Juego e información para el jugador local
    //*******************************
    //Actualiza en la interfaz los puntos obtenidos del jugador
    public void updatePointsText(string text){
        panelTopTextPoints.text = text;
    }

    //Muestra y oculta el panel de los objetos Actor que ha obtenido
    el jugador
    public void showhideActorsTypePlayer()
    {
        Button[] buttons = panelCenterShowActorsTypePlayer.
        GetComponentsInChildren<Button> ();
        if (buttons.Length>0) {
            panelCenterShowActorsTypePlayer.SetActive(
            !panelCenterShowActorsTypePlayer.GetActive());
        }
    }

    //Agrega al panel de actorsTypePlayer una nuevo actorPlayer
    public void addIconActorsTypePlayer(ActorType actorType)
    {
        Sprite actorIcon = getIconSprite(actorType.icon);
        GameObject actorButton =
        Instantiate(prefabButtonAddDinamico);
        actorButton.GetComponent<Image>().sprite = actorIcon;
        actorButton.transform.SetParent(
        panelHorizontalGroupActorsTypePlayer.transform);
    }

    // Muestra y oculta el panel que detalla los puntos de los
    jugadores
    public void showhideGameInfoPlayer()
    {
        if (!panelGameInfoPlayer.GetActive ()) {
            printInfoGame (false);
            panelGameInfoPlayerButtonCancel.SetActive(true);
        }
        panelGameInfoPlayer.SetActive(
        !panelGameInfoPlayer.GetActive());
    }

    // Construye la interfaz de los jugadores y sus puntos
    public void printInfoGame(bool finalize){

        foreach (Transform t in
        panelGameInfoPlayerActors.transform) {
            Destroy (t.gameObject);
        }

        foreach (Transform t in panelGameInfoPlayerLists.transform)
        {
            Destroy (t.gameObject);
        }

        List<Player> playerWinners = new List<Player> ();
        if (finalize == true) {
```

```
        playerWinners =
        Game.gameInstance.getPlayerWinnerGameRoom ();
}

if (Game.gameInstance.actorsTypePlayer.Count > 0)
{
        panelGameInfoPlayerActors.SetActive(true);
        foreach (ActorType actorType in
        Game.gameInstance.actorsTypePlayer)
        {
                Sprite actorIcon =
                getIconSprite(actorType.icon);
                GameObject actorButton =
                Instantiate(prefabButtonAddDinamico);
                actorButton.GetComponent<Image>().sprite =
                actorIcon;  actorButton.transform.SetParent(
                panelGameInfoPlayerActors.transform);
        }
}
else
{
        Sprite actorIcon =
        getIconSprite("player_actors_null");
        GameObject actorButton =
        Instantiate(prefabButtonAddDinamico);
        actorButton.GetComponent<Image>().sprite = actorIcon;
        actorButton.transform.SetParent(
        panelGameInfoPlayerActors.transform);
}

foreach (Player item in Game.gameInstance.players)
{
        GameObject prefabPlayer =
        Instantiate(prefabPlayerAddDinamico);
        string txtPoints = "";
        string txtIcon = "player_no_winner";
        if (Game.gameInstance.isCompetitive == true)
        {
                float points =
                Game.gameInstance.getPointsPlayer(item);
                txtPoints =
                System.Math.Ceiling(points).ToString();

        }
        if (finalize == true) {
                Player isWinner = playerWinners.Find
                (playerWinner => playerWinner.id == item.id);
                if (isWinner != null) {
                        txtIcon = "player_winner";
                }
        }
        prefabPlayer.GetComponent<PlayerItem>
        ().playerName.text = (item.name=="") ?
        item.id.ToString(): item.name;
        prefabPlayer.GetComponent<PlayerItem>
        ().playerPoints.text = txtPoints;
        prefabPlayer.GetComponent<PlayerItem>
        ().playerIcon.sprite = getIconSprite (txtIcon);
        prefabPlayer.transform.SetParent(
        panelGameInfoPlayerLists.transform);
}
```

```
        if (Game.gameInstance.isCompetitive == true)
        {
              if (Game.gameInstance.gamePointsAnonymous > 0)
              {
                    GameObject prefabPlayer =
                    Instantiate(prefabPlayerAddDinamico);
                    prefabPlayer.GetComponent<PlayerItem>().
                    playerName.text = "Anónimo";
                    prefabPlayer.GetComponent<PlayerItem>().
                    playerPoints.text = System.Math.Ceiling(
                    Game.gameInstance.gamePointsAnonymous).
                    ToString();
                    prefabPlayer.GetComponent<PlayerItem>().
                    playerIcon.sprite =
                    getIconSprite("player_no_winner");
                    prefabPlayer.transform.SetParent(
                    panelGameInfoPlayerLists.transform);
              }
        }
        else
        {
              GameObject prefabPlayer =
              Instantiate(prefabPlayerAddDinamico);
              prefabPlayer.GetComponent<PlayerItem>().
              playerName.text = "Total de puntos";
              prefabPlayer.GetComponent<PlayerItem>().
              playerPoints.text =
              Game.gameInstance.gamePointsPlayer
              + "/" + Game.gameInstance.gamePointsTotal;
                    prefabPlayer.GetComponent<PlayerItem>().
              playerIcon.sprite = getIconSprite(finalize == true ?
              "player_winner": "player_no_winner");
                    prefabPlayer.transform.SetParent(
              panelGameInfoPlayerLists.transform);
        }

        if (finalize == true) {
              Player isWinnerLocal = playerWinners.Find
              (playerWinner => playerWinner.id ==
              PhotonNetwork.player.ID);
              if (isWinnerLocal != null &&
              Game.gameInstance.isCompetitive) {
                    textGameInfoPlayerHeadWinner.text =
                    "FELICIDADES! GANASTES";
              }else if(isWinnerLocal != null &&
              !Game.gameInstance.isCompetitive) {
                    textGameInfoPlayerHeadWinner.text =
                    "FELICIDADES! FIN DE JUEGO";
              } else {
                    textGameInfoPlayerHeadWinner.text =
                    "SUERTE PARA LA PRÓXIMA";
              }
              panelGameInfoPlayerHeadWinner.SetActive (true);

        } else {
              panelGameInfoPlayerHeadWinner.SetActive (false);
        }
  }
```

```
        //*******************************
        // Métodos comunes
        //*******************************
        // Retorna un icon de tipo sprite
        public Sprite getIconSprite(string nombre_icon){
              Sprite spriteIcon = Resources.Load<Sprite>("Icons/" +
              nombre_icon);
              return spriteIcon;
        }

        // Ejecuta el audio
        public void playAudio(int opcAudioRun)
        {
              if (opcAudioRun == 1)
              {
                    audioSourceAction.clip = sonidoRunProjectile;
              }
              else if (opcAudioRun == 2)
              {
                    audioSourceAction.clip = sonidoRecoverProjectile;
              }
              else if (opcAudioRun == 3)
              {
                    audioSourceAction.clip =
                    sonidoProjectileWithoutEffect;
              }
              else if (opcAudioRun == 4)
              {
                    audioSourceAction.clip = sonidoProjectileWitEffect;
              }
              else if (opcAudioRun == 5)
              {
                    audioSourceAction.clip = sonidoEndGame;
              }
              else if (opcAudioRun == 6)
              {
                    audioSourceAction.clip = sonidoInitCollaborative;
              }

              audioSourceAction.Play();
        }

}
}
```

Anexo 4: Script Game.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System.IO;
using Newtonsoft.Json.Linq;
using System.Linq;

namespace ExitGames.Upv.ProyectoTfm
{
public class Game : Photon.PunBehaviour
{
	public bool isCompetitive;
	public List<Player> players;

	// Prefabs del juego
	List<ActorType> actorsType;
	public List<ProjectileType> projectilesType;
	public List<ProjectileType> projectilesTypePlayerRecover;

	// Proyectil del juego
	float timeCurrentProjectilesRecover;
	float timeLimitProjectilesRecover;
	public bool runProjectile;
	public Vector3 runProjectileAddress;
	public ProjectileType projectileSelec;

	// Puntos del juego
	public int gamePointsTotal;
	public float gamePointsPlayer;
	public float gamePointsAnonymous;
	public List<ActorType> actorsTypePlayer;

	// Juego colaborativo
	public int actorTypeIdCurrentCollaborative;

	static public Game gameInstance;

	void Start()
	{
		gameInstance = this;

	}

	void Update () {

		if (PhotonNetwork.connected){
			if (projectilesTypePlayerRecover.Count > 0) {
				timeCurrentProjectilesRecover +=
				Time.deltaTime;
				if (timeCurrentProjectilesRecover >
				timeLimitProjectilesRecover){
					timeCurrentProjectilesRecover = 0;
					updatePositionProjectilesTypeRecover ();
				}
			}

		}

	}
```

```
//*******************************
// Control del juego
//*******************************
public void startGame()
{
      if (PhotonNetwork.connected)
      {
            gamePointsPlayer = 0;
            gamePointsTotal = 0;
            gamePointsAnonymous = 0;
            actorsTypePlayer=new List<ActorType>();
            //Inicializa los jugadores actuales
            players = new List<Player> ();
            foreach (PhotonPlayer newPlayer in
            PhotonNetwork.playerList){
                  Player player = new Player ();
                  player.id = newPlayer.ID;
                  player.name = newPlayer.NickName;
                  player.pointsActorType = new Dictionary<int,
                  float>();
                  players.Add (player);
            }
            // Inicializa la configuración del juego
            runProjectile = false;
            projectileSelec = null;
            timeCurrentProjectilesRecover = 0;
            timeLimitProjectilesRecover = 15;
            projectilesTypePlayerRecover = new
            List<ProjectileType>();
            projectilesType = new List<ProjectileType>();
            actorsType = new List<ActorType>();
            actorTypeIdCurrentCollaborative = -1;
            getConfigurationGame(1);
            createConfigurationInstantiate();
            updatePointsActorTypePlayerLocal ();
      }

}

// Busca el projectileType por id en la lista asignada al game
public ProjectileType getProjectileTypeById(int
idProjectileType)
{
      ProjectileType projectileType = projectilesType.Find(item
      => item.id == idProjectileType);
      return projectileType;
}

// Busca el actorType por id en la lista asignada al game
public ActorType getActorTypeById(int idActorType)
{
      ActorType actorType = actorsType.Find(item => item.id ==
      idActorType);
      return actorType;
}

// Obtiene la configuración del juego que se va a ejecutar de
acuerdo al archivo configuration.json
void getConfigurationGame(int gameId)
{
```

```
string filePath =
Path.Combine(Application.streamingAssetsPath,
"configuration.json");
string data = "";
if (filePath.Contains("://")){
      WWW www = new WWW(filePath);
      while (!www.isDone) { }
      data = www.text;
}else{
      data = File.ReadAllText(filePath);
}

JObject jObject = JObject.Parse(data);
IList<JToken> gamesJtoken =
jObject["games"].Children().ToList();
foreach (JToken gameJtoken in gamesJtoken)
{
      if ((int)gameJtoken["id"] == gameId)
      {
            IList<JToken> actorsTypeJtoken =
            gameJtoken["actorsType"].Children().ToList();
            foreach (JToken actorTypeJtoken in
            actorsTypeJtoken)
            {
                  List<JToken> resultsTypeJtoken =
                  actorTypeJtoken["resultsType"].
                  Children().ToList();
                  Dictionary<string, int>
                  resultsProjectileType =
                  new Dictionary<string, int>();
                  foreach (JObject resultTypeJtoken in
                  resultsTypeJtoken)
                  {
                        resultsProjectileType.Add(
                        resultTypeJtoken["projectileType"].
                        ToString(),
                        (int)resultTypeJtoken["units"]);
                  }
                  ActorType actorType =
                  actorTypeJtoken.ToObject<ActorType>();
                  actorType.resultsProjectileType =
                  resultsProjectileType;
                  actorsType.Add(actorType);
                  gamePointsTotal += actorType.lifeInit;
            }
            IList<JToken> projectilesTypeJtoken =
            gameJtoken["projectilesType"].
            Children().ToList();
            foreach (JToken projectileTypeJtoken in
            projectilesTypeJtoken)
            {
                  ProjectileType projectileType =
                  projectileTypeJtoken.ToObject
                  <ProjectileType>();
                  projectilesType.Add(projectileType);
            }

            break;
      }
}
```

```
}

//Crea los prefats de acuerdo al juego seleccionado asociados al
Room y no al jugador
void createConfigurationInstantiate()
{
      if (PhotonNetwork.isMasterClient){
            for (int i = 0; i < actorsType.Count; i++){
                  float x, y, z;
                  Vector3 position;
                  if (actorsType[i].randomPosition){
                        position =
                        Configuration.configurationInstance.
                        getPointRandomMap ();
                  }else{
                        x = actorsType[i].initX;
                        y = actorsType[i].initY;
                        z = actorsType[i].initZ;
                        position = new Vector3(x, y, z);
                  }
                  // Cuando el prefabs es propiedad del room
                  object[] instantiationData = new object[1];
                  instantiationData[0] = actorsType[i].id;
                  GameObject goActor =
                  PhotonNetwork.InstantiateSceneObject(
                  actorsType[i].name, position,
                  Camera.main.transform.rotation, 0,
                  instantiationData);
                  goActor.GetComponent<Actor>().actorType =
                  actorsType[i];
            }
      }
}

// Muestra el objeto a disparar en la camara
public void showProjectilePlayer(ProjectileType projectileType)
{
      if (projectileType != null){
            projectileSelec = projectileType;
            restarProjectilePlayer();
            Vector3 positionProjectile = C
            Camera.main.transform.position +
            Camera.main.transform.forward * 0.4f;
            Quaternion rotationProjectile =
            Camera.main.transform.rotation;
            // Cuando el prefabs es propiedad del jugador local
            object[] instantiationData = new object[1];
            instantiationData[0] = projectileType.id;
            GameObject projectileSelecObject =
            PhotonNetwork.Instantiate(
            projectileType.name, positionProjectile,
            rotationProjectile, 0, instantiationData);
            projectileSelecObject.
            GetComponent<Projectile>().projectileType =
            projectileType;


            projectileSelecObject.
            GetComponent<Projectile>().projectileTypeRecover =
            false;
      }
```

```
}

// Dispara el objeto mostrado en la camara
public void runProjectilePlayer()
{
      bool existsProjectile = verifyExistsProjectilePlayer();
      if (existsProjectile){
            Room.roomInstance.playAudio(1);
            runProjectileAddress = Camera.main.transform.forward;
            runProjectile = true;
            ProjectileType proyectileTypeRecover =
            getProjectileTypeById(projectileSelec.id);
            if (proyectileTypeRecover != null){
StartCoroutine(createProjectileTypeRecover(
                  proyectileTypeRecover));
                  addProjectilesTypeRecover(
                  proyectileTypeRecover);
            }

            projectileSelec = null;
      }
}

// Creao el projectile a recuperar pasando un determinado tiempo
IEnumerator createProjectileTypeRecover(ProjectileType
proyectileTypeRecover)
{
      yield return new WaitForSeconds(5);
      Vector3 position = Configuration.configurationInstance
      .getPointRandomMap ();
      GameObject proyectileTypeRecoverObj =
      Instantiate(Resources.Load(proyectileTypeRecover.name,
      typeof(GameObject)), position,
      Camera.main.transform.rotation) as GameObject;
      proyectileTypeRecoverObj.
      GetComponent<Projectile>().projectileType =
      proyectileTypeRecover;
      proyectileTypeRecoverObj.GetComponent<Projectile>().
      projectileTypeRecover = true;
      proyectileTypeRecoverObj.GetComponent<Transform>().
      Rotate(new Vector3(proyectileTypeRecover.rotationX,
      proyectileTypeRecover.rotationY,
      proyectileTypeRecover.rotationZ));
}

// Agrega el projectile type a la lista de projectiles a
recuperar
public void addProjectilesTypeRecover(ProjectileType
projectileType)
{
      ProjectileType existInRecover =
      projectilesTypePlayerRecover.Find(item => item.id ==
      projectileType.id);
      if (existInRecover == null){
            projectilesTypePlayerRecover.Add(projectileType);
            projectilesType.Remove(projectileType);
      }
}

// Agrega el projectile type a la lista de projectiles a
disparar
```

```
public void removeProjectilesTypeRecover(ProjectileType
projectileType)
{
      ProjectileType proyectileTypeRecover =
      projectilesTypePlayerRecover.Find(item => item.id ==
      projectileType.id);
      if (proyectileTypeRecover != null){
            projectilesTypePlayerRecover.Remove(
            proyectileTypeRecover);
            projectilesType.Add(proyectileTypeRecover);
      }
}

// Reiniciar disparo, por lo cual debe exitir un solo prefab
para ser diparado
void restarProjectilePlayer()
{
      runProjectile = false;
      GameObject[] projectilesGame =
      GameObject.FindGameObjectsWithTag("Player");
      foreach (GameObject projectile in projectilesGame){
            if (projectile.GetComponent<Projectile>().
            projectileTypeRecover != true){
                  PhotonView view = projectile.GetComponent
                  <PhotonView>();
                  if (view.isMine){
                        PhotonNetwork.Destroy(view);
                  }
            }

      }
}

// Retorna true si existe un objeto disparo asociado al jugador
local
bool verifyExistsProjectilePlayer()
{
      GameObject[] projectilesGame =
      GameObject.FindGameObjectsWithTag("Player");
      foreach (GameObject projectile in projectilesGame){
            if (projectile.GetComponent<Projectile>().
            projectileTypeRecover != true){
                  PhotonView view = projectile.GetComponent
                  <PhotonView>();
                  if (view.ownerId == PhotonNetwork.player.ID){
                        return true;
                  }
            }
      }
      return false;
}

// Actualiza cada cierto tiempo la posición de los projectiles
de tipo recuperacion
public void updatePositionProjectilesTypeRecover(){
      GameObject[] projectilesGame =
      GameObject.FindGameObjectsWithTag("Player");
      foreach (GameObject projectileGO in projectilesGame){
            if (projectileGO.GetComponent<Projectile>().
            projectileTypeRecover == true){
```

```
                  Vector3 newPosition =
                  Configuration.configurationInstance.
                  getPointRandomMap ();
                  projectileGO.transform.position = newPosition;
            }
      }
}

//*******************************
// Puntos del Juego
//*******************************
//Cuando el actorType llega a la vida de cero, se valida que
jugador captura a este, que es el que tiene mayor puntaje
public void addActorTypePlayer(ActorType actorType){
      if (isCompetitive)
      {
            int playerSelecId = -1;
            float points = -1;
            foreach (Player player in players){
                  if(player.pointsActorType.ContainsKey(
                  actorType.id)){
                        if (playerSelecId == -1){
                              playerSelecId = player.id;
                              points = player.pointsActorType[
                              actorType.id];
                        }
                        else if (player.pointsActorType[
                        actorType.id] > points){
                              playerSelecId = player.id;
                              points = player.pointsActorType[
                              actorType.id];
                        }
                  }
            }

            if (playerSelecId != -1){
                  bool uniqueActorTypePlayer = true;
                  List<Player> uniquePlayers =
                  players.FindAll(item => item.id !=
                  playerSelecId);
                  foreach (Player player in uniquePlayers){
                        if (player.pointsActorType.ContainsKey(
                        actorType.id)){
                              if (player.pointsActorType
                              [actorType.id] == points){
                                    uniqueActorTypePlayer =
                                    false;
                                    break;
                              }
                        }
                  }

                  if (playerSelecId == PhotonNetwork.player.ID &&
                  uniqueActorTypePlayer == true){
                        ActorType actorTypePlayer =
                        actorsTypePlayer.Find(
                        item => item.id == actorType.id);
                        if (actorTypePlayer == null){
                              actorsTypePlayer.Add(actorType);
                              Room.roomInstance.
                              addIconActorsTypePlayer(actorType);
```

```
                        }
                    }
                }
            }else{
                ActorType actorTypePlayer =
                actorsTypePlayer.Find(item => item.id ==
                actorType.id);
                if (actorTypePlayer == null){
                    actorsTypePlayer.Add(actorType);
                    Room.roomInstance.addIconActorsTypePlayer(
                    actorType);
                }
            }

        }

        // Almacena los puntos acumulados por cada jugador y por cada
        actor
        public void addPointsActorTypePlayer(int playerId, int
        playerActorId, float points){
            foreach (Player player in players){
                if((isCompetitive == true && player.id == playerId)
                || isCompetitive == false){
                    if (player.pointsActorType.
                    ContainsKey(playerActorId))
                    {
                        float sumaPoints =
                        player.pointsActorType[playerActorId]
                        + points;
                        if (sumaPoints <= 0) {
                            player.pointsActorType.Remove(
                            playerActorId);
                        }else{
                            player.pointsActorType[
                            playerActorId] += points;
                        }
                    }else{
                        if (points > 0) {
                            player.pointsActorType.Add(
                            playerActorId, points);
                        }
                    }
                }
            }
            updatePointsActorTypePlayerLocal();
        }

        // Actualiza los puntos totales del jugador local
        public void updatePointsActorTypePlayerLocal(){
            gamePointsPlayer = 0;
            Player player = players.Find(item => item.id ==
            PhotonNetwork.player.ID);
            if (player != null) {
                gamePointsPlayer = getPointsPlayer (player);

            }
            Room.roomInstance.updatePointsText
            (System.Math.Ceiling(gamePointsPlayer)
            + "/"+gamePointsTotal);
        }
```

```
//Veifica si debe finalizar el juego
public bool finalizeGameRoom(){
      Actor[] actorsGame = GameObject.FindObjectsOfType<Actor>();
      if (actorsGame.Length == 0){
            return true;
      }
      return false;
}

//Obtiene los ganadores del juego, si existe
public List<Player> getPlayerWinnerGameRoom(){
      List<Player> playerWinners = new List<Player>();
      float playerWinnerPoints = 0;
      foreach (Player player in players)
      {
            float pointsPlayer = getPointsPlayer (player);
            if (pointsPlayer > 0) {
                  if (playerWinners.Count==0) {
                        playerWinners.Add(player);
                        playerWinnerPoints = pointsPlayer;

                  } else if (pointsPlayer > playerWinnerPoints) {
                        playerWinners.Clear();
                        playerWinners.Add(player);
                        playerWinnerPoints = pointsPlayer;
                  }
            }
      }

      if (playerWinners.Count == 1) {
            List<Player> playersEqualPoints =
            players.FindAll(item =>
            (item.id != playerWinners[0].id &&
            getPointsPlayer(item) == playerWinnerPoints));
            if (playersEqualPoints != null){
                  playerWinners.AddRange(playersEqualPoints);
            }
      }
      return playerWinners;
}

//Retorna los puntos para un jugador especifico
public float getPointsPlayer(Player player){
      float pointsPlayer = 0;
      foreach(KeyValuePair<int, float> entry in
      player.pointsActorType){
            pointsPlayer += entry.Value;
      }
      return pointsPlayer;
}

// Actualiza el id para saber cual actor type esta en
colaboración para ser capturado
public void updateActorTypeIdCurrentCollaborative(int
idActorType)
{
      actorTypeIdCurrentCollaborative = idActorType;
}

// Inicializa el tipo de juego
public void updateTypeGame(bool typeGame)
```

```
        {
              isCompetitive = typeGame;
        }

        public override void OnPhotonPlayerConnected(PhotonPlayer
        newPlayer)
        {
              Player existe = players.Find(item => item.id ==
              newPlayer.ID);
              if (existe == null) {
                    Player player = new Player ();
                    player.id = newPlayer.ID;
                    player.name = newPlayer.NickName;
                    player.pointsActorType = new Dictionary<int,
                    float>();
                    players.Add (player);
              }
        }

        public override void OnPhotonPlayerDisconnected(PhotonPlayer
        newPlayer)
        {
              Player player = players.Find(item => item.id ==
              newPlayer.ID);
              if (player != null) {
                    float pointsAnonymous = getPointsPlayer (player);
                    if (pointsAnonymous > 0) {
                          gamePointsAnonymous += pointsAnonymous;
                    }
                    players.Remove (player);
              }
        }
}
}
```

Anexo 5: Script Actor.cs

```
using UnityEngine;
using UnityEngine.UI;

namespace ExitGames.Upv.ProyectoTfm
{
public class Actor : Photon.PunBehaviour, IPunObservable
{
	public ActorType actorType;
	public float life = 1;
	public bool activeAddPointsCollaborative = false;
	public int playerAddPointsCollaborative = -1;
	public int timeCurrentCollaborativeSeconds = 0;
	public float percentajeAcumulateCollaborative = 0;

	// Manejar un temporizador
	float timeCurrent = 0;
	float timeCurrentCollaborativeInc = 0;
	float timeCurrentCollaborativeLim = 1;

	// Colaborativo, sincronizar con usuarios remotos
	int actorTypeIdCurrentCollaborative = -1;

	// Muestra información por interfaz
	public Image lifeImagen01;
	public Text lifeTextPercentaje;

	void Start()
	{
	}

	void Update()
	{
		if ((actorType != null && actorType.id == 0) ||
		(actorType == null))
		{
			object[] instantiationData =
			photonView.instantiationData;
			if (instantiationData != null)
			{
				int idActorType = (int)instantiationData[0];
				actorType = Game.gameInstance.getActorTypeById(
				idActorType);
			}
		}

		lifeImagen01.fillAmount = life;
		float porcentaje = life * 100;
		lifeTextPercentaje.text = porcentaje.ToString("0") + " %";

		if (PhotonNetwork.isMasterClient)
		{
			if (life == 0)
			{
				PhotonNetwork.Destroy(photonView);
				return;
			}
			timeCurrent += Time.deltaTime;
			if (timeCurrent > actorType.timeToMovi)
			{
```

```
                timeCurrent = 0;
                Vector3 address =
                Configuration.configurationInstance.
                getPointRandomMap ();
                transform.position = address;
            }

            if (Game.gameInstance.isCompetitive == false)
            {
                if (activeAddPointsCollaborative == true)
                {
                    if (timeCurrentCollaborativeSeconds == 0)
                    {
                        initCollaborative();
                    }
                    else
                    {
                        timeCurrentCollaborativeInc +=
                        Time.deltaTime;
                        if (timeCurrentCollaborativeInc >
                        timeCurrentCollaborativeLim)
                        {
                            timeCurrentCollaborativeInc =
                            0;
                        timeCurrentCollaborativeSeconds--;
                        }
                        Room.roomInstance.updatePointsText(
                        timeCurrentCollaborativeSeconds.
                        ToString());
                    }
                }

                actorTypeIdCurrentCollaborative = Game.
                gameInstance.actorTypeIdCurrentCollaborative ;
            }

        }
    }


    [PunRPC]
    void updateLife(float percentaje, int playerID)
    {
        if (life - percentaje >= 0 && life - percentaje <= 1 &&
        percentaje!=0) {
            life = life - percentaje;
            float points = percentaje * actorType.lifeInit;
            Game.gameInstance.addPointsActorTypePlayer (playerID,
            actorType.id, points);
            if (life == 0) {
                Game.gameInstance.addActorTypePlayer
                (actorType);
            }
            if((Game.gameInstance.isCompetitive == true &&
            playerID == PhotonNetwork.player.ID) ||
            Game.gameInstance.isCompetitive == false)
            {
                if (percentaje > 0)
                {
                    Room.roomInstance.playAudio(4);
                }
```

```
                else
                {
                        Room.roomInstance.playAudio(3);
                }
            }

        }
}

[PunRPC]
void updateLifeCollaborativeMaster(float percentaje, int
playerID)
{
      if (PhotonNetwork.isMasterClient)
      {
            if (life > 0)
            {
                  if (activeAddPointsCollaborative == false &&
                  Game.gameInstance.
                  actorTypeIdCurrentCollaborative == -1) {
                        Game.gameInstance.
                        updateActorTypeIdCurrentCollaborative
                        (actorType.id);
                        activeAddPointsCollaborative = true;
                        playerAddPointsCollaborative = playerID;
                        percentajeAcumulateCollaborative =
                        percentaje;
                        timeCurrentCollaborativeSeconds =
                        actorType.timeCollaborative;
                        timeCurrentCollaborativeInc = 0;
                        Room.roomInstance.playAudio (6);

                  } else if (activeAddPointsCollaborative == true
                  && Game.gameInstance.
                  actorTypeIdCurrentCollaborative
                  == actorType.id) {

                        if (playerID !=
                        playerAddPointsCollaborative) {
                              float lifePercentaje =
                              percentajeAcumulateCollaborative
                              + percentaje;
                              float lifeDiferencia = life -
                              lifePercentaje;
                              if (lifeDiferencia < 0) {
                                    lifePercentaje = life;
                              } else if (lifeDiferencia > 1) {
                                    lifePercentaje = 1 - life;
                              }
                              if (life - lifePercentaje >= 0 &&
                              life - lifePercentaje <= 1
                              && lifePercentaje != 0) {
                                    PhotonView photonView =
                                    PhotonView.Get (this);
                                    photonView.RPC ("updateLife",
                                    PhotonTargets.All,
                                    lifePercentaje,
                                    PhotonNetwork.player.ID);
                              }

                              initCollaborative ();
```

```
                        } else {
                              Game.gameInstance.
                        updateActorTypeIdCurrentCollaborative
                              (actorType.id);
                              playerAddPointsCollaborative =
                              playerID;
                              percentajeAcumulateCollaborative =
                              percentaje;
                              timeCurrentCollaborativeSeconds =
                              actorType.timeCollaborative;
                              timeCurrentCollaborativeInc = 0;
                              Room.roomInstance.playAudio (6);
                        }
                  } else {
                        Room.roomInstance.playAudio(3);
                  }

            }
      }
}

void initCollaborative()
{
      activeAddPointsCollaborative = false;
      playerAddPointsCollaborative = -1;
      timeCurrentCollaborativeSeconds = 0;
      timeCurrentCollaborativeInc = 0;
      percentajeAcumulateCollaborative = 0;
      Game.gameInstance.updateActorTypeIdCurrentCollaborative(-
      1);
      Game.gameInstance.updatePointsActorTypePlayerLocal();
}

public void OnPhotonSerializeView(PhotonStream stream,
PhotonMessageInfo info)
{
      if (stream.isWriting)
      {
            stream.SendNext(this.life);
            stream.SendNext(
            this.timeCurrentCollaborativeSeconds);
            stream.SendNext(this.activeAddPointsCollaborative);
            stream.SendNext(
            this.actorTypeIdCurrentCollaborative);
      }
      else
      {
            int auxIdCurrentCollaboratibe =
            actorTypeIdCurrentCollaborative;
            this.life = (float)stream.ReceiveNext();
            this.timeCurrentCollaborativeSeconds =
            (int)stream.ReceiveNext();
            this.activeAddPointsCollaborative =
            (bool)stream.ReceiveNext();
            this.actorTypeIdCurrentCollaborative =
            (int)stream.ReceiveNext();
            if (Game.gameInstance.isCompetitive == false &&
            actorType != null)
            {
```

```
                        if(actorTypeIdCurrentCollaborative ==
                        actorType.id)
                        {
                              if (activeAddPointsCollaborative == true)
                              {
                                    Room.roomInstance.updatePointsText
                                    (timeCurrentCollaborativeSeconds.
                                    ToString());
                                    if (timeCurrentCollaborativeSeconds
                                    == actorType.timeCollaborative)
                                    {
                                    Room.roomInstance.playAudio(6);
                                    }
                              }
                              else
                              {
                                    Game.gameInstance.
                                    updatePointsActorTypePlayerLocal();
                              }
                        }else if(auxIdCurrentCollaboratibe ==
                        actorType.id)
                        {
                              Game.gameInstance.
                              updatePointsActorTypePlayerLocal();
                        }
                  }
            }
      }

      void OnCollisionEnter(Collision col)
      {
            if (PhotonNetwork.connected) {
            if (col.gameObject.GetPhotonView()){
            if (col.gameObject.GetPhotonView().isMine &&
            Game.gameInstance.runProjectile == true){
            if(col.gameObject.GetComponent<Projectile>()){

                  Projectile gameObjectProjectile =
                  col.gameObject.GetComponent<Projectile>();
                  if (gameObjectProjectile.
                  projectileTypeRecover != true)
                  {
                        if(actorType.resultsProjectileType.
                        ContainsKey(gameObjectProjectile.
                        projectileType.id.ToString()))
                        {
                              if (life > 0)
                              {
                              int units = actorType.
                              resultsProjectileType
                              [gameObjectProjectile.
                              projectileType.
                              id.ToString()];
                              float lifePercentaje = units /
                              (float)actorType.lifeInit;
                              float lifeDiferencia = life -
                              lifePercentaje;

                                    if (lifeDiferencia < 0){
                                                lifePercentaje = life;
```

```
                                    }else if (lifeDiferencia > 1) {
                                          lifePercentaje = 1 - life;
                                    }

                                    if (life - lifePercentaje >= 0 &&
                                    life - lifePercentaje <= 1
                                    && lifePercentaje != 0)
                                    {
                                          if (col.gameObject.
                                          GetPhotonView().isMine)
                                          {
                                          PhotonNetwork.Destroy(
                                          col.gameObject.
                                          GetPhotonView());
                                          }

                                          PhotonView photonView =
                                          PhotonView.Get(this);

                                          if (Game.gameInstance.
                                          isCompetitive == true) {
                                                photonView.RPC(
                                                "updateLife",
                                                PhotonTargets.All,
                                                lifePercentaje,
                                                PhotonNetwork.player.
                                                ID);
                                          }else{
                                                photonView.RPC(
                                    "updateLifeCollaborativeMaster",
                                                PhotonTargets.
                                                MasterClient,
                                                lifePercentaje,
                                                PhotonNetwork.player.
                                                ID);
                                          }

                                          return;
                                    }
                              }
                        }

                        Room.roomInstance.playAudio(3);

                        }

                  }
            }
            }
            }
            }
      }
}
```

Anexo 6: Script Projectile.cs

```
using UnityEngine;

namespace ExitGames.Upv.ProyectoTfm
{
public class Projectile : Photon.PunBehaviour
{
      public ProjectileType projectileType;
      public bool projectileTypeRecover;

      void Update()
      {
            if (projectileTypeRecover == false){
                  if ((projectileType != null &&
                  projectileType.id == 0) ||
                  (projectileType == null)){
                        object[] instantiationData =
                        photonView.instantiationData;
                        if (instantiationData != null)
                        {
                              int idProjectileType =
                              (int)instantiationData[0];
                              projectileType =
                              Game.gameInstance.getProjectileTypeById
                              (idProjectileType);
                        }

                  }

                  if (photonView.isMine){
                        if (Game.gameInstance.runProjectile){
                              Vector3 address =
                              Game.gameInstance.runProjectileAddress *
                              Time.deltaTime;
                              transform.position += address;
                        }else{
                              Vector3 positionProjectile =
                              Camera.main.transform.position +
                              Camera.main.transform.forward * 0.4f;
                              Quaternion rotationProjectile =
                              Camera.main.transform.rotation;
                              transform.position = positionProjectile;
                              transform.rotation = rotationProjectile;

                              if (projectileType != null &&
                              projectileType.id != 0){
                                    transform.Rotate(new
                                    Vector3(projectileType.rotationX,
                                    projectileType.rotationY,
                                    projectileType.rotationZ));
                              }
                        }
                  }
            }
      }
}
}
```