Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Diseño e implementación de juegos serios para realizar pruebas psicológicas basadas en estímulo-respuesta

Proyecto Final de Carrera

Ingeniería Informática

Autor: Giovanny Javier Tipantuña Toapanta

Director: José Vicente Benlloch Dualde

Codirector: Lenin Guillermo Lemus Zúñiga

Septiembre 2015

Resumen

Una de las tendencias actuales es el uso de juegos serios en diferentes áreas con propósitos igualmente diferentes como reducir el tiempo de aprendizaje de los empleados, eliminar el miedo al riesgo al estar en un entorno simulado, obtener datos sobre la interacción con el participante, etc. En este proyecto estudiamos el uso de juegos serios para realizar pruebas cognitivas pues el área de la psicología experimental es un área interesada en transformar algunas de las pruebas clásicas a un formato multimedia que haga uso de las potencialidades de las nuevas tecnologías, es decir, que facilite la automatización de los procesos de captura, permita la modificación de las variables independientes del experimento de forma rápida, y ofrezca entornos simulados más atractivos y acordes a las expectativas de las nuevas generaciones.

Como parte del proyecto se implementa y evalúa una aplicación que pretende servir como herramienta de apoyo en estudios sobre el acceso léxico. Ésta permite la realización de una tarea de decisión léxica a través de un juego serio, la automatización de la captura de los datos y el almacenamiento en una base de datos para su posterior análisis. El juego serio consiste en la búsqueda y selección de varios libros dentro de una estantería.

Finalmente veremos que las conclusiones arrojan que los juegos serios utilizados para realizar pruebas cognitivas son un aliciente para los participantes por lo que se puede incrementar el número de sujetos que realicen ésas pruebas. Así mismo veremos cómo el uso de las nuevas tecnologías nos permite modificar las variables independientes de forma rápida.

Palabras clave: juegos serios, psicología experimental, validez ecológica, tiempo de reacción, Unity, Stroop, tarea de decisión léxica.

Abstract

One of the most current trends is the use of serious games in different areas with different purposes such as: the reduction of training time for employees, to avoid the fear of risk thanks to a simulated environment, to obtain data from the interaction with the participant, etc. In this project we study the usage of serious games to do cognitive tests, provided that the psychological area is interested in converting some of the classical tests into a multimedia format and thus, taking advantage of the new technologies. Therefore, these conversions would facilitate the automation of capture processes, a quick modification of the independent variables, and would provide us with more attractive simulated environments in line with expectations of new generations.

As part of the project an application is implemented and assessed in order to serve as a support tool in studies of lexical access. This permits the realization of a lexical decision task through a serious game, automating data capture and storing in a database for further analysis. In particular, the serious game consists of the search and selection of several books in a bookcase.

As conclusions we will see that using serious games to perform cognitive tests are an incentive for participants so the number of contributors may be increased. Also we will see how the use of new technologies allows us to modify the independent variables quickly.

Keywords: serious games, experimental psychology, ecological validity, reaction time, Unity, Stroop, lexical decision task.

Índice de contenidos

Índice de figuras

Índice de tablas

1 Introducción

El avance de las herramientas software ha hecho posible que muchos de los procesos que se realizaban manualmente se hayan automatizado, e incluso ha permitido modificar los contextos en los que se realizaban, para así obtener resultados más permanentes y fiables.

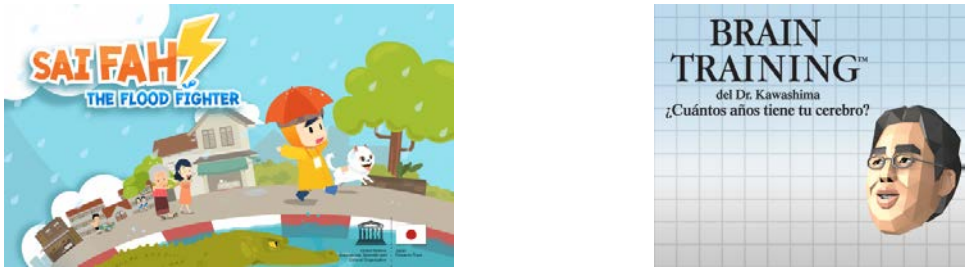
Uno de los campos de mayor actualidad donde se hace realidad lo anterior es el llamado juegos serios (*serious games*), que intenta aprovechar los beneficios de los videojuegos ya sea para hacer más efectivo un proceso educativo, o para la obtención de información a partir de la interactividad con el jugador. Actualmente, existen varios ejemplos de juegos serios que han llegado a tener muchos usuarios por el crecimiento del uso de dispositivos móviles en la población, desde juegos que enseñan a sobrevivir a catástrofes naturales como es el caso de *Sai fah: The Flood Fighter* [1], hasta aquellos que pretenden recuperar o mejorar habilidades cognitivas a través de ejercicios mentales como *Brain Training del Dr. Kawashima* [2].

En *Sai fah: The Flood Fighter* los retos a ser superados por el jugador pueden desarrollarse en un contexto anterior a una inundación, presentando características particulares como fuertes precipitaciones en la ciudad. Así mismo, los retos pueden desarrollarse en un contexto posterior a la inundación mostrando escenas determinadas como crecidas de ríos, animales fuera de su hábitat, etc. A la vez existen retos que se desarrollan al mismo tiempo que se produce la inundación en los que, por ejemplo, se pide al usuario elegir artículos de primera necesidad en un supermercado, seguir las órdenes de un agente de seguridad, etc. En todos ellos el propósito principal es proporcionar a los jugadores conocimientos que ayuden a reducir los riegos ante eventuales catástrofes naturales reales, empleando principalmente la técnica de ensayo-error. Debido a la cantidad de situaciones de peligro generadas como consecuencia de un desastre natural, las lecciones dadas durante el desarrollo de los retos de este videojuego van desde evitar serpientes hasta cortar la electricidad para reducir riesgos de electrocución. En definitiva, el jugador debe satisfacer los puntos de revisión establecidos en cada mapa del juego antes de pasar al siguiente. Esta iniciativa nace como respuesta ante las inundaciones producidas en 2011 en Tailandia (OmniumGames, 2014).

[1] *Sai Fah: The Flood Fighter* es un proyecto apoyado por el gobierno de Japón y la UNESCO. Para más detalles podemos acudir al enlace web: <http://www.floodfighterthegame.com/en/about>. [Consultado: Junio 2015]

[2] *Brain Training del Dr. Kawashima* fue un juego exitoso para la consola portable *Nintendo DS* ® lanzado en 2006 que ofrecía ejercicios verbales y manuales para estimular la mente. Para más detalles podemos consultar el siguiente enlace web: <https://www.nintendo.es/Juegos/Nintendo-DS/Brain-Training-del-Dr-Kawashima-Cuantos-anios-tiene-tu-cerebro--270627.html>. [Consultado: Junio 2015]

En *Brain Training del Dr. Kawashima* se empleaban técnicas como la escritura en la pantalla para recibir las respuestas, así como el reconocimiento del habla, dando al usuario una nueva manera de jugar que en 2006 resultó ser exitosa. Gracias al incremento del uso de videoconsolas portables como *Nintendo DS* ™, este videojuego ofrecía una forma novedosa y cómoda de entretenerse y a la vez realizar ejercicios mentales. Una de las principales características que atrajo a los usuarios fue la posibilidad de medir, en base a los resultados obtenidos y en contraste con datos estándares de pruebas de agilidad mental, la edad estimada de su cerebro, así como llevar un seguimiento de los resultados propios (Nintendo, 2015). En junio de 2015 se realizó el lanzamiento de la versión actualizada de este videojuego para la consola portable *Wii U* ™ y cuyos detalles se pueden hallar en la web oficial de *Nintendo* ®.



(a) (b)

Figura 1-1: Pantallas iniciales de los videojuegos *Sai fah: The Flood Fighter* (a) y *Brain Training del Dr. Kawashima* (b).
Imagen (a) tomada del sitio web oficial http://www.floodfighterthegame.com [Consultado: Junio 2015]
Imagen (b) tomada del sitio web oficial https://www.nintendo.es [Consultado: Junio 2015]

Como vemos, las nuevas tecnologías permiten la implementación de juegos serios que tienen distintos propósitos como instruir o mejorar las capacidades del usuario de forma entretenida. Son empleados en diferentes áreas como, por ejemplo, en el sector empresarial para instruir a empleados, en defensa para simular escenarios de peligro, en psicología experimental como apoyo en algunos estudios, etc. La mayoría de estos videojuegos presentan escenarios simulados que dan la posibilidad al jugador de experimentar situaciones específicas de forma virtual. Con ello se consigue reducir el coste económico de instruir, de obtener datos de la interacción con el usuario, o de mejorar las capacidades del jugador. Ya que instruirlo de forma natural puede requerir más recursos económicos y materiales, así como obtener datos de forma manual puede introducir imprecisiones. Además, la motivación de los jugadores por obtener éxito en el juego hace que los jugadores se comprometan con su formación.

1.1 Objetivos

Este proyecto se enmarca dentro del proceso de transformación de determinadas pruebas psicológicas basadas en estímulo-respuesta, desde un formato de papel hacia un formato multimedia, teniendo cada una de ellas sus particularidades. Para abordar estas transformaciones deberemos, en primer lugar, realizar una toma de contacto con las herramientas software buscando maneras de emitir estímulos y capturar respuestas, y, en segundo lugar, tratando de incrementar el grado de validez ecológica.

Los objetivos que abordaremos a lo largo de este proyecto son:

- Estudio de los inconvenientes derivados del uso del tiempo de reacción como variable de medición en estudios de psicología experimental.
- Evaluación de algunas herramientas software para implementar juegos serios que miden el tiempo de reacción. Se pretende valorar el nivel de conocimientos y la cantidad de recursos necesarios derivados del uso de una determinada plataforma mediante la implementación breve y sencilla de la prueba de *Stroop*.
- Implementación de una aplicación multiplataforma que permita realizar una tarea de decisión léxica (TDL) y almacenar los datos recogidos para su análisis. Se pretende elevar el grado de validez ecológica mediante gráficos 3D y el uso de una base de datos para el almacenamiento. La TDL consistirá en la búsqueda y selección de varios libros colocados en una estantería, representando así una escena cotidiana sencilla.

En estas líneas de trabajo la variable principal a medir es el tiempo de reacción, de forma que los conceptos adquiridos en la primera serán tenidos en cuenta al implementar la prueba de *Stroop*, y las maneras encontradas de automatizar la captura en ésta servirán para abordar la misma labor en la aplicación de la tarea de decisión léxica. Veremos que además se pueden medir otras variables como la tasa de aciertos para arrojar más conocimientos acerca del comportamiento de los participantes, incluso el orden de selección en algunos casos.

2 Juegos serios como herramienta para pruebas cognitivas

Los juegos serios han ido llenando el vacío que las nuevas tecnologías habían dejado en áreas que requieren la formación o aprendizaje de determinadas habilidades por parte de los usuarios. Esto es así gracias a una de sus características principales que es ayudar en la mejor compresión de procesos complejos. A ello se suma el incremento del uso de dispositivos móviles así como la mejora de las capacidades del hardware de los mismos, lo cual ha hecho que por un lado el número de usuarios potenciales de aplicaciones de formación y juegos serios se haya incrementado, y por otro lado que las aplicaciones para dispositivos móviles puedan hacer uso de más recursos del hardware y el software de estos dispositivos como, por ejemplo, los gráficos 3d o el acceso a internet a través de redes inalámbricas.

En el ámbito de la psicología experimental un gran número de juegos serios basan sus objetivos principales en los objetivos de estudios propios de este ámbito. La intención de estos estudios es recopilar información que puede ser útil para diagnóstico, prevención o recuperación de capacidades cognitivas, donde sus principales fuentes de medición son las expresiones de la cara, las expresiones corporales, los tiempos de reacción y los valores en la sociedad como, por ejemplo, aquellos valores que son objeto de estudio en experimentos que intentan medir el grado de empatía de un participante ante situaciones de terceras personas.

2.1 Contexto sobre la salud mental

Uno de los últimos informes de la Organización Mundial de la Salud (OMS, 2009) indica que en el año 2006 la población con 60 años o más superaba los 600 millones, dejando como resultado que este número haya triplicado al de 1950 y duplicado al de 1980. Este incremento ha llamado la atención de los gobiernos ya que se prevé que este número alcance los 2000 millones antes del año 2050, con una tasa de crecimiento de 2.6% mientras que la tasa de crecimiento de la población es de 1.2%. La Figura 3-1 nos muestra un resumen del crecimiento de los porcentajes aquí mencionados.

Como consecuencia de este incremento los casos de pacientes con deterioro cognitivo o *demencias* [3] como el *Alzheimer* [4] ha aumentado de manera importante. Todo ello ha

[3] Término para describir síntomas que son el resultado de varios trastornos que afectan al cerebro.

[4] Es un tipo de demencia en el que las células nerviosas dejan de funcionar perdiendo conexiones con otras neuronas.

provocado que médicos y psicólogos hayan establecido estrategias para realizar tratamientos y busquen nuevos métodos que arrojen datos hasta ahora desconocidos o que permitan agilizar la investigación en este campo.

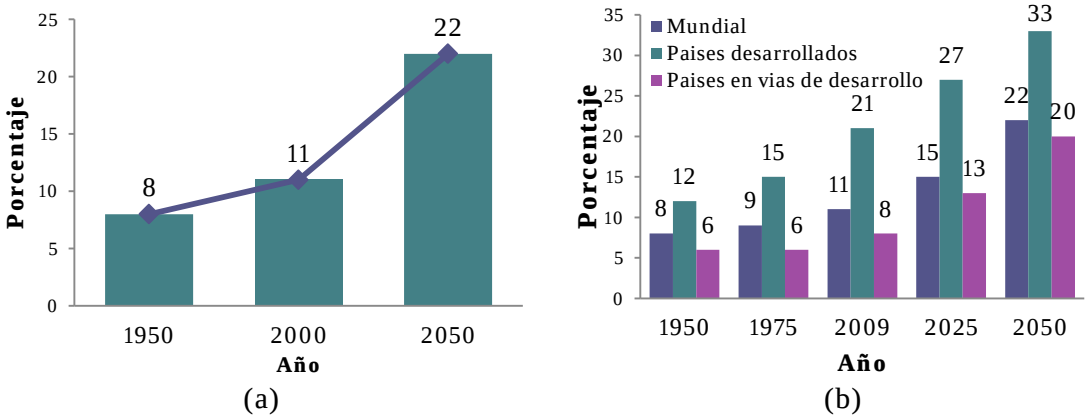


Figura 3-1: Proporción de población de 60 o más años (a) en el mundo entre 1950 y 2050, y (b) por regiones de desarrollo en el mundo entre 1950 y 2050.

Por ejemplo, para diagnosticar la demencia se emplean varias estrategias donde es necesario descartar otras condiciones que pueden causar síntomas parecidos. En esta búsqueda, la realización de un diagnóstico a tiempo permite el tratamiento temprano de los síntomas y en ocasiones una recuperación de las habilidades del paciente. Por ello algunas de las técnicas más empleadas por los médicos para ayudar a identificar la demencia según (NINDS, 2010) son:

- El historial médico del paciente.
- El examen físico.
- Las evaluaciones neurológicas.
- Las pruebas cognitivas y neurosicológicas.
- La escanografía del cerebro.
- Las pruebas de laboratorio.

En el mismo sentido nuestra labor en este proyecto se encamina hacia el apartado de pruebas cognitivas ya que es aquí donde se realizan innovaciones para medir diversos parámetros como la memoria, las habilidades relacionadas con el funcionamiento mental, los tiempos de reacción, etc. Además, el poder disponer de nuevas formas de realizar estas mediciones es un punto relevante ya que en algunos casos es difícil detectar los síntomas de la demencia, pues a menudo los pacientes con esta enfermedad no son conscientes de ello o incluso pueden negar estar afectados debido a su estado emocional.

Es en la tarea de aportar nuevas formas de recoger los datos donde presentarle un juego serio a un participante puede permitirnos enmascarar procesos que de otra manera podrían resultarle poco atractivos. Así mismo, la disponibilidad de herramientas software con las que la implementar estas aplicaciones permite optimizar la captura de

respuestas, eliminando en la medida de lo posible, el porcentaje de error que solía introducirse cuando se utilizaban instrumentos de medición manuales.

Sin embargo, en esta medición de las reacciones del participante se deben tener en cuenta algunos conceptos propios de la psicología experimental, ya que a pesar de realizar correctamente la captura de los datos, las fases de diseño del experimento, o la de evaluación de los datos recogidos pueden contener errores.

Como habíamos comentado una de las principales fuentes de medición de las respuestas de los participantes es el tiempo de reacción. Por ello en el apartado siguiente realizaremos un recorrido por algunos de los métodos de medición de esta variable y mencionaremos algunas aplicaciones que la usan para medir procesos cognitivos. Así mismo mencionaremos varios inconvenientes que se pueden encontrar durante su captura o análisis. Estos conceptos que son propios del área de la psicología experimental deben ser tenidos en cuenta para una correcta interpretación de los datos recogidos.

2.2 Tiempos de reacción

Los vínculos que se han descubierto entre los tiempos de reacción y las manifestaciones mentales son relevantes puesto que han contribuido a dar una demostración objetiva. Además, el estudio de los tiempos de reacción permite establecer subprocesos mentales en la ejecución de una tarea.

A día de hoy, el tiempo de reacción es la *variable dependiente* [5] más empleada en la experimentación en psicología cognitiva. De acuerdo con (Sternberg, Reaction-Time Experimentation, 2004) esto es debido a varias características que mencionamos a continuación:

- Permite medir los fenómenos mentales cuando el cerebro aún funciona correctamente y no sólo cuando se encuentra afectado por algún desorden.
- Incluso cuando las respuestas no están determinadas completamente por el estímulo, el tiempo para iniciar una respuesta puede ser un indicador más sensible que el subproceso de elección de la respuesta.
- Permite revelar la organización temporal de los procesos mentales.

Para realizar estas mediciones se han desarrollado varios métodos que se pueden separar en tres:

[5] En psicología experimental una variable dependiente es la variable de estudio que se quiere medir. Es aquella que demuestra una relación causal cuando al cambiar los valores de las variables independientes también cambia su valor.

- El método sustractivo.
- El método de los factores aditivos.
- Modelos alternativos contemporáneos.

2.2.1 Método sustractivo

El *método sustractivo* (Donders, 1868) establece una manera de realizar mediciones a la duración de procesos mentales basándose en el análisis de los tiempos de reacción fisiológicos. La idea base de este método es que el proceso de realizar una determinada acción o tarea es un conjunto de varios subprocesos cognitivos que pueden reflejarse en la duración total de dicha ejecución. Por consiguiente, el tiempo total es la suma de los subprocesos, por lo que Donders clasificó tres tipos de tareas para ayudar a extraer el valor aislado de la duración de cada subproceso, las cuales mencionamos a continuación:

- Tipo A: Tiempo de reacción simple en dos etapas. Detección de estímulo y emisión de respuesta.
- Tipo B: Tiempo de reacción disyuntivo. Pueden aparecer dos estímulos y cada uno con cuatro etapas. Detección del estímulo, discriminación del estímulo, selección de la respuesta y emisión de la respuesta.
- Tipo C: Tiempo de reacción de elección. Pueden aparecer dos estímulos pero sólo se responde a uno en tres etapas. Detección del estímulo, identificación de la respuesta y emisión de la respuesta.

De esta forma, Donders conseguía establecer la duración de los procesos mentales sustrayendo el tiempo de reacción simple al tiempo de reacción disyuntivo. La Tabla 3-1 nos muestra un esquema del funcionamiento de este método.

Comparación de diferentes tareas

A: TR simple	*Estímulo*	Detección del estímulo	Emisión de respuesta	*Respuesta*		
B: TR disyuntivo	*Estímulo 1* *Estímulo 2*	Detección de estímulo	Discriminación del estímulo	Selección de la respuesta	Emisión de la respuesta	*Respuesta 1* *Respuesta 2*
C: TR de elección	*Estímulo 1* *Estímulo 2*	Detección del estímulo	Discriminación del estímulo	Emisión de la respuesta	*Respuesta 1* *Respuesta 2*	

Discriminación del estímulo & Selección de respuesta	=	TR(B) – TR(A)
Selección de respuesta	=	TR(B) – TR(C)
Discriminación del estímulo	=	TR© – TR(A)

Tabla 3-1: Tabla de comparación de tareas usando el método sustractivo de Donders para el cálculo del tiempo de reacción.

2.2.2 Método de los factores aditivos

El *método de los factores aditivos* presentado por (Sternberg, The Discovery of Processing Stages. Extensions of Donders' Method, 1969), expone que no son las etapas las que se suman si no los factores que intervienen en el proceso que se realiza en cada etapa. Argumenta que hay etapas que dependen de un solo factor y otras que dependen de varios.

Para aquellas que dependen de un solo factor, sus tiempos de reacción se pueden sumar. En cambio, para aquellas en las que intervienen varios factores el tiempo de reacción no es aditivo y por tanto se debe emplear un diseño factorial.

El hecho de hacer la distinción entre dependencia de un factor o dependencia de varios factores, como se explica en (Sternberg, Reaction-Time Experimentation, 2004), es debido a que el tiempo de reacción es la duración de varios procesos mentales, y cuando se diseña un experimento se debe encontrar uno o varios factores que influencien sólo al subproceso que es objeto de estudio y no a los otros.

Este método no pretende obtener la duración de las etapas sino arrojar respuestas sobre cuáles son las etapas que participan en la actividad mental entre la presentación del estímulo y la respuesta mediante el estudio de los factores que influyen en dichas etapas.

Según este método se puede decir que la intensidad del estímulo influye en la reacción y la etapa que se ve afectada por el nivel de intensidad es la que se corresponde con el proceso de detección del estímulo. Así mismo, el tipo de respuesta (manual o verbal) puede afectar a la duración temporal de una etapa, en cuyo caso, la etapa se corresponde con el proceso de ejecución de la respuesta. La Figura 3-2 resume el modelo de Sternberg.

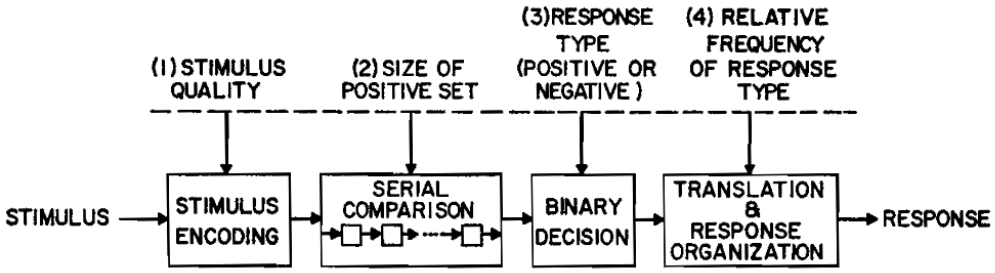


Figura 3-2: Etapas del método de los factores aditivos de Sternberg.

Imagen tomada del documento de **(Sternberg, The Discovery of Processing Stages. Extensions of Donders' Method, 1969)**.

2.2.3 Uso en tareas de decisión léxica

La tarea de decisión léxica es empleada para arrojar conocimientos sobre los subprocesos mentales que se producen durante la lectura de palabras. Actualmente es una de las tareas más usadas en los estudios sobre acceso léxico visual. Fue utilizada por Rubenstein en 1970 y consiste en presentar al participante, de forma escrita u oral, un conjunto de palabras que está compuesto por palabras y no-palabras. Las denominadas no-palabras son aquellas que cumplen reglas de formación de palabras del lenguaje que se está usando de apoyo en el estudio, como por ejemplo el español, pero que no existen en él, así como palabras imposibles. Tras la presentación del estímulo el participante debe decidir si se trata de una palabra o una no-palabra. Es en esta recepción de la respuesta donde el tiempo de reacción puede arrojar más datos en este tipo de estudios junto a la tasa de aciertos y otras variables de medición.

Ejemplo de lo que hemos mencionado antes es el proyecto *ScienceXL* [6] que fue desarrollado recientemente por el grupo de Dufau et al. (2011), en el que se implementó una aplicación para aumentar el conocimiento sobre la capacidad de leer. En este proyecto los investigadores mostraron cómo las tareas de decisión léxica (TDL) pueden extenderse hasta las nuevas tecnologías en forma de juego a través de una aplicación, sobre todo hacia plataformas móviles como *iPad*, *iPhone*, *iPod*, etc. Además, el proyecto de Dufau et al (2011) se caracterizó por su carácter internacional y repercusión sobre el uso de los tiempos de reacción como variable que refleja procesos lectores, principalmente a través de uso de tareas de decisión léxica. La Figura 3-3 nos muestra un ejemplo del aspecto de esta aplicación.

[6] Se puede obtener más información acerca del proyecto y sus colaboradores, así como videos de demostración y enlaces de descarga de la aplicación, en su web oficial: <http://www.sciencexl.org/>

Figura 3-3: Ejemplo de tarea de decisión léxica (TDL) desarrollada en *ScienceXL* en 2011.

El participante debe decidir si lo que se le presenta es una palabra o pseudopalabra presionando la tecla "sí" o "no". Estos tiempos de reacción proporcionan información subyacente a procesos cognitivos básicos.

2.2.4 Uso en aplicaciones web

Una de las tendencias de más actualidad para recopilar datos en estudios de psicología es usar las ventajas derivadas del incremento del acceso a internet mediante aplicaciones disponibles a través de páginas web. Algunos de éstos abordan la captura y el análisis de los tiempos de reacción.

Estos estudios son posibles gracias a la compatibilidad de las aplicaciones con el amplio número de navegadores web que existen actualmente, así como gracias a sus complementos. En dichos estudios se evitan los retardos temporales debidos a la transmisión por la red, de manera que las aplicaciones o programas hechos para realizar las mediciones primero se descargan, luego se ejecutan en el lado del cliente, generalmente con la ayuda de un complemento del navegador, y finalmente, en algunos casos, se envían los resultados a través de la red para ser almacenados en un servidor. Sin embargo el uso del navegador o de sus complementos como soporte para ejecutar las aplicaciones introduce retardos en la captura de las respuestas. Estos inconvenientes serán descritos en el apartado de problemas en la captura.

A modo de ejemplo de estas aplicaciones web podemos mencionar, de los varios ejemplos que se pueden hallar investigando a través de internet, uno destinado a mostrar al usuario el efecto *Stroop* mediante la realización de varias pruebas de lectura con configuraciones diferentes. En él se presentan nuevos escenarios y configuraciones para medir el efecto *Stroop* que difieren del experimento original, el cual se basaba en presentar estímulos compuestos por color y palabra. Estas configuraciones van desde presentar conjuntos con un número determinado de palabras que puede diferir del

número que esas palabras están describiendo, por lo que se puede inducir a error a la hora de identificar el número correcto de palabras que se encuentran en el conjunto, hasta mostrar animales con una palabra encima que también puede diferir. Estas pruebas pretenden dar a conocer esta interferencia en la mente del usuario mediante la presentación de los tiempos de reacción. Primero se le presenta una imagen con configuración que no tiene errores, y luego una imagen con una configuración que sí los contiene. El usuario puede comprobar que su tiempo de identificación correcta de todos los elementos varía. Esta información se le muestra tras pulsar el botón de finalización. En el caso de la segunda configuración el tiempo para identificar correctamente es notablemente mayor que en el primer caso. Esta aplicación utiliza programación en el lenguaje *Html* para, por ejemplo, capturar el evento de pulsación del botón y programación en el lenguaje *JavaScript* para medir el tiempo de reacción y mostrarlo en un mensaje dentro de la ventana. La figura Figura 3-4 nos muestra el aspecto de estas pruebas que se pueden realizar accediendo a la página web.

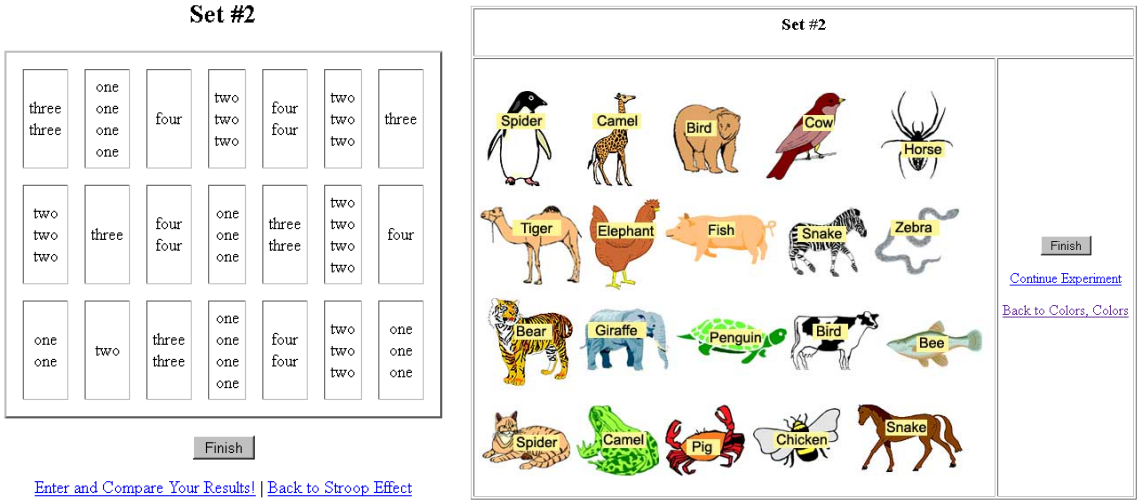


Figura 3-4: Capturas de pantalla de las aplicaciones web que realizan la prueba de *Stroop* con escenarios diferentes y midiendo el tiempo de reacción. (a) El participante debe identificar correctamente el número de palabras de cada conjunto. (b) El usuario debe identificar correctamente todos los animales.

Imágenes tomadas durante la ejecución de las aplicaciones en su sitio web <https://faculty.washington.edu/chudler/words.html>, apartado *New Stroop Test*. [Consultado: Julio 2015]

Como vemos, por un lado el tiempo de reacción es una de las herramientas más empleadas en los estudios de psicología relacionados con la salud mental, y por otro, la existencia de nuevas tecnologías al alcance del ciudadano común ha permitido el surgimiento de proyectos como el de *ScienceXL*, o la implementación de aplicaciones web enfocadas al estudio de los procesos cognitivos. Este tipo de trabajos intentan abordar la transformación de determinadas pruebas psicológicas hacían un formato que sea más amistoso y haga uso de las oportunidades que las nuevas tecnologías ofrecen.

El incremento de las capacidades de los dispositivos también ha jugado un papel importante, pues a día de hoy es posible hacer uso de características que antes no poseían como, por ejemplo, interfaces gráficas de usuario, gráficos 3d, acceso a internet, mayor velocidad de procesamiento, etc. Estas características facilitan la automatización de procesos repetitivos y reducen el error que se cometía al medir de forma manual.

Por ejemplo, nos permiten automatizar la captura de los tiempos de reacción y con ello arrojar nuevos datos o datos más precisos sobre estudios que ya se habían realizado, así como realizar comparativas e intentar determinar si realmente aportan facilidades a los experimentos de psicología experimental.

Sin embargo se deben tener en cuenta los inconvenientes que conlleva hacer uso del tiempo de reacción como variable de medición, pues durante la captura o durante el análisis de los datos recogidos, se pueden cometer errores. En el apartado siguiente mencionaremos algunos de los inconvenientes.

2.2.5 Problemas en la medición

Actualmente, el modelo de Sternberg es una de las herramientas analíticas más exitosas en psicología para el tratamiento del tiempo de reacción, sin embargo existen problemas a la hora del diseño y evaluación de resultados en este tipo de experimentos.

Por ejemplo, uno de los problemas que surgen en la fase de diseño de un experimento es que no se pueden evitar los efectos de la práctica ya que los tiempos de reacción disminuyen con la práctica. Así mismo, puede que no seamos capaces de evitar pensamientos alternativos durante el cumplimiento de una tarea.

Por otro lado, también existen problemas relacionados con la fase de tratamiento de los datos recogidos. Un ejemplo de ello es el que se expone en (Perea, 1999), donde se explica que recortar de la muestra los datos extremos, cuyas causas pueden deberse a distracciones del participante o fallos de los instrumentos de medición, puede provocar interpretaciones desviadas.

A pesar que las estimaciones de la media son sesgadas se emplean para comparar diferentes *condiciones experimentales* [7] y no para conocer el tiempo de reacción real. Sin embargo, cuando el tamaño se reduce a un tamaño considerado pequeño debido a la eliminación de puntuaciones “erróneas” o extremas, se puede dar lugar a que el número de datos por condición sea diferente al haber valores perdidos.

[7] Una condición experimental es una combinación de valores de variables independientes (ej.: tiempo de exposición de un estímulo) que permiten observar variaciones en una o varias variables dependientes (ej.: tiempo de reacción) en un experimento.

Existen otros problemas relacionados con ajustar la distribución de los tiempos de reacción como se explican en (Zandt, 2000). Por ejemplo, ajustar un modelo únicamente a la media de los tiempos de reacción puede enmascarar detalles importantes de los datos, los cuales examinando la distribución al completo sí serían revelados.

También existen inconvenientes en las tareas que emplean el tiempo de reacción como variable de medición para examinar los procesos léxicos cognitivos subyacentes al proceso de lectura. Las dos tareas más destacables son la tarea de decisión léxica (TDL) y la tarea de denominación o "naming". En un experimento de TDL, los participantes tienen que decidir si los estímulos son palabras, por ejemplo, pulsando una tecla. En el caso de la tarea de denominación (que es muy similar a la TDL), los participantes tienen que responder a los estímulos mediante su nombramiento. Estas tareas podrían estar contaminadas por otros procesos tales como la pronunciación o la falta de validez ecológica, como se explica en el estudio de (D'Amico, Devescovi, & Bates, 2001) y en (Bates, 2001; Paap, McDonald, Schvaneveldt, y Noel, 1987), es decir, son entornos muy artificiales.

En el presente trabajo intentaremos incrementar el grado de validez ecológica en una tarea de decisión léxica, pues se pretende mostrar al participante un escenario que podría desarrollarse en cualquier librería a la hora de buscar un libro. Esto se aborda en apartados posteriores referentes al trabajo realizado.

Como vemos existen diversos problemas en el diseño del experimento y en la evaluación de los datos recogidos, ante los cuales se están desarrollando soluciones y mejoras. Por otro lado, también existen inconvenientes en la captura de los tiempos de reacción que están relacionados con el tipo de tecnología que se emplea como herramienta de ayuda en la captura de los tiempos de reacción. En el apartado siguiente haremos un recorrido por algunos de los inconvenientes encontrados en el proceso de captura al emplear las nuevas tecnologías.

2.2.6 Problemas en la captura

La evolución de los sistemas basados en microprocesadores y su uso en estudios científicos ha ido mostrando inconvenientes relacionados directamente con el hardware del que están compuestos los nuevos dispositivos. Por ello debemos mencionar algunas de las interferencias que se pueden producir durante la captura de los tiempos de reacción debido a las características del funcionamiento del hardware interno o los complementos software empleados.

Ejemplo de estos problemas son los que se encontraron en estudios que utilizaban pantallas de tubo de rayo catódico [8] (*CRT* por sus siglas en inglés) y pretendían medir el tiempo de reacción, como los que se describen en (E. Lincoln & M. Lane, 1980). Según este estudio un problema importante se podría producir debido al proceso de barrido secuencial de las tramas horizontales en la pantalla. Al ser la frecuencia de sincronización de 60 Hz, se requiere 16.76 ms para barrer la pantalla completa. Sin embargo, las velocidades con las que trabajan los circuitos del computador usado para generar y almacenar la información de la imagen, son mucho más altas. De forma que el reloj que mide el tiempo de reacción puede empezar su cuenta antes que la imagen de estímulo realmente aparezca en la pantalla.

Así mismo, se explica que la relevancia de este error de retardo en la presentación del estímulo depende de la magnitud de la varianza en el error, y que generalmente hay una variación muy grande entre los tiempos de reacción de los participantes. Para determinar la pérdida de potencia estadística de forma más precisa, su experimento variaba el tiempo entre estímulos de forma aleatoria y uniforme, y concluyeron que incluso manipulando esta variable independiente entre los sujetos, la pérdida de potencia debida al error por el retardo del CRT, es decir, debida al retardo temporal en la presentación del estímulo, es muy pequeña.

En el caso de las aplicaciones web que miden el tiempo de reacción también existen problemas de interferencias en la captura. Ante lo cual se han realizado algunos estudios que revelan el grado de relevancia de estos problemas. Ejemplo de ello es el estudio de (Schubert, Murteira, Collins, & Lopes, 2013). En él se mencionan algunos de los problemas derivados del uso de complementos del navegador como los *Applets* de *Java*, los programas escritos en lenguaje *Javascript*, o las aplicaciones que se ejecutan sobre el complemento *Flash*, entre otros. Este estudio muestra que *Flash*, *Java*, y *JavaScript* pueden utilizarse para medir tiempos de reacción y obtener mediciones aceptables. Sin embargo éstos incrementan el ruido en las mediciones en comparación con las aplicaciones que han sido programadas con código nativo de la máquina o que han sido compiladas para ejecutarse con una determinada arquitectura de microprocesador y un determinado sistema operativo.

A pesar de las interferencias en las mediciones debido al uso de los complementos del navegador sobre el que se ejecutan las aplicaciones web, existen ventajas que justifican su uso. Una de ellas es que las aplicaciones implementadas mediante lenguajes de programación interpretados como *JavaScript* o *ActionScript* (lenguaje empleado para implementar aplicaciones *Flash*) no dependen del sistema operativo ni de la arquitectura de la máquina, por lo que el potencial número de usuarios no se ve limitado. En su lugar dependen del complemento del navegador web que es capaz de interpretar dicha aplicación. Por ejemplo, los applets de Java en definitiva son

[8] Estas pantallas utilizan un sistema de barrido en el que un haz de electrones con intensidad variable recorre líneas en horizontal y golpea puntos en la pantalla, brillando éstos durante unos milisegundos debido al componente de fósforo.

estructuras de tipo *byte-code* que son interpretados por una máquina virtual java, las aplicaciones Flash son ejecutadas por el complemento de navegador correspondiente que generalmente es el de *Adobe® Flash® Player*, etc. Así mismo, como revela el estudio antes mencionado, los retardos derivados del uso de estos lenguajes y complementos son pequeños y no influyen significativamente en los resultados adquiridos. Por ejemplo, en el caso de las aplicaciones Flash, el complemento introduce un retardo 30 ms, como se describe en el estudio mencionado anteriormente, pero no añade un incremento en la desviación estándar. Por lo que se pueden usar para implementar programas que midan el tiempo de reacción, los cuales pueden ser embebidos en páginas web con la ayuda del lenguaje *Html* y así permitir el acceso a un mayor número de participantes y recopilar una gran cantidad de datos.

En el presente proyecto se intentarán abordar los pasos básicos en la creación de una aplicación que pueda ser usada en un entorno web, con la intención de experimentar las facilidades e inconvenientes que algunos entornos de desarrollo proporcionan en la implementación de aplicaciones que miden el tiempo de reacción. Lo cual será abordado en apartados posteriores referentes al trabajo realizado.

Finalmente, existen otros conceptos que deben tenerse en cuenta de forma importante. Nos referimos a las características que, a grandes rasgos, deben poseer los juegos serios para poder enfocarlas a nuestras aplicaciones de pruebas psicológicas. Por ejemplo, el público al que está destinado un determinado juego o prueba, la edad, la complejidad de la prueba, etc. Éstos y otros conceptos serán abordados en el siguiente apartado.

2.3 Características de los juegos serios

Según (Felix Navarro, Lawrence, Garcia Marin, & Sax, 2011) existen puntos necesarios en el desarrollo de un juego serio que deben tenerse en cuenta para que éstos sean correctamente interpretados por los usuarios, así como para que cumpla el objetivo para el que fue creado. Estos puntos los mencionamos a continuación.

- Se deben proporcionar instrucciones claras y precisas sobre cómo se debe jugar.
- Los objetivos del juego deben ser alcanzables ya que en caso contrario el jugador se puede desmotivar.
- Se deben tener en cuenta las capacidades físicas y mentales de los participantes.
- Las instrucciones deben presentarse antes y no durante el juego.
- Se debe evitar exigir demasiada memorización o complejidad cognitiva innecesaria ya que se busca que el usuario vea el juego como un entretenimiento y debemos evitar frustraciones.
- La historia del juego debe resultar atractiva para el jugador.

Uno de los métodos empleados en juegos serios para el adiestramiento del jugador es el método de ensayo-error, ya que de esta forma los conceptos adquiridos por el usuario son más permanentes y al ser una simulación no suponen un riesgo humano o de recursos de una empresa.

Así mismo, como se describe en el estudio de (Sánchez Gómez, 2007), un punto a tener en cuenta a la hora de la creación de un *serious game*, es que los objetivos deben adaptarse al nivel cognitivo del público al que va dirigido. En este sentido, si trabajamos con personas adultas debemos saber que el aprendizaje no se realiza de la misma forma que en etapas previas a la etapa adulta de la persona. Es por ello, que los conceptos nuevos se deben introducir de forma que no contrasten con sus experiencias e ideas previas.

Para estas adaptaciones, que en nuestro caso adquieren relevancia ya que pretendemos implementar juegos serios con los que realizar pruebas cognitivas, se deben tener cuenta los procesos mentales que las diferentes escenas de nuestro juego, así como los elementos presentes en ellas, pueden generar en el participante en el momento de su visualización. Como describe (Vorderer, 2000), debemos considerar los siguientes puntos que hacen referencia a los procesos cognitivos de los contenidos del juego y de los modelos mentales del mismo:

- Debe haber consistencia entre el experimento y las habilidades que son objeto de análisis.
- El juego debe tener elementos que permitan al participante seguir la historia del mismo mediante sus diferentes sentidos como la vista, el oído, incluso el idioma, con el objetivo de que no se sienta perdido en ningún momento.
- Se debe evitar la exigencia de conocimientos previos ya que esto reduce el número de usuarios potenciales. Por ejemplo, las personas de la tercera edad tienen dificultades en el uso de dispositivos digitales.
- Los juegos deben ser sencillos e independientes.

En cuanto a la organización de las fases clave para el desarrollo de un juego serio, podemos destacar tres principales: análisis del contexto, diseño y desarrollo, y evaluación. A continuación describiremos las ideas principales a llevar a cabo en las distintas fases, según (Sánchez Gómez, 2007).

- Análisis del contexto: Se debe realizar con la intención de delimitar los objetivos, identificar el público de destino, y establecer un plan de diseño acotado. Establecer los objetivos permitirá delimitar lo que se desea lograr y marcará un conjunto de recursos que dependerá de la intención del juego.
- Diseño y desarrollo: Lo primero a tener en cuenta para un buen plan de diseño son las variables que no son heterogéneas en los participantes como el sexo, la edad, o la educación. De esta forma la interpretación del juego por parte del usuario presentará el mínimo de inconvenientes. Así mismo, el diseñador debe

anticipar comportamientos del usuario y estados de la aplicación para asegurar que la navegabilidad y la interfaz sean intuitivas. Por otro lado, se debe organizar la información empleando correctamente elementos visuales (colores, fondos, textos, animaciones, etc.), así como elementos de control (botones, temporizadores, puntos de control en la escena, etc.).

- Evaluación: En esta fase convendría realizar pruebas reales con voluntarios para recoger fortalezas y debilidades de la aplicación, así como pedir una evaluación de expertos relacionados con el área que aborda el juego, es decir, en nuestro caso habría que ver si la prueba cognitiva no excede los límites para los que ha sido creada. Los inconvenientes deberán ser corregidos o minimizados hasta satisfacer una correcta usabilidad adaptando menús, escenas, etc.

Como vemos, para explotar la mayoría de las bondades que un juego serio puede aportar a nuestro propósito de recopilar datos en pruebas psicológicas, será necesario tener en cuenta algunos puntos clave como la sencillez en las pruebas, la claridad en las instrucciones en pruebas no supervisadas, o la facilidad de intuición durante la realización de la prueba para no perturbar el tiempo de reacción. Teniendo en cuenta esos puntos, los escenarios del juego pueden aportar un mayor atractivo para los pacientes o usuarios, pues a pesar que este tipo de juegos dejan en segundo plano la diversión y tienen como objetivo principal la formación o la extracción de datos, se presentan como un reto de contacto con las nuevas tecnologías en la sociedad digital actual.

En el apartado siguiente mencionaremos algunos de los juegos serios que han sido implementados para diferentes propósitos como la ayuda en el aprendizaje de la anatomía humana, la realización de pruebas cognitivas para recopilar datos, entre otros.

2.4 Uso de juegos serios en diferentes áreas

Entre los principales beneficios aportados por los juegos serios podemos encontrar que reducen la sensación de gravedad frente a errores y fracasos e invitan a la participación activa. Ayudando así en el desarrollo integral de las personas. Un ejemplo de esto se puede observar en las personas mayores que utilizan las redes sociales, o aquellos que usan juegos en los que entran en relación con otros usuarios, pues adquieren o recuperan pautas de relación y convivencia como se describe en el estudio de (Marcano, 2008).

Es por ello, que las áreas en las que se emplean o se están empezando a utilizar este tipo de juegos son variadas. Para corroborar el incremento de las áreas en las que se están

desarrollando proyectos en las que los juegos serios toman protagonismo, vamos a mencionar algunos ejemplos a continuación.

En el sector empresarial las expectativas sobre los beneficios del uso de videojuegos para entrenamiento de sus empleados son altas, ya que sirven como herramienta efectiva para la mejora de determinadas habilidades como el manejo de software, habilidades de relaciones interpersonales, etc. De modo que quien está siendo instruido actúa participativamente.

Ejemplo de ellos es *Lego® Serious Play®*, el cual consiste en una metodología de estrategia en tiempo real y un conjunto de materiales. Los participantes trabajan de forma cooperativa y, mediante la construcción de escenarios, pueden identificar conocimientos, actitudes y habilidades. Además les permite entender la manera de pensar del resto de participantes, así como sentirse en un entorno más cómodo en el que compartir su punto de vista. Se emplean piezas de *Lego®* pues éstas representan una forma menos intimidante de afrontar una determinada tarea y asumir riesgos en un contexto de simulación. En la Figura 3-5 podemos observar una imagen de ejemplo del uso de esta metodología y materiales.

Figura 3-5: Ejemplo de uso de *Lego® Serious Play®* en el sector empresarial. La compañía *Strategic Play Group Ltd*. investiga maneras de romper el ciclo de la pobreza y proporcionar una vida familiar más sana enfocándose en individuos y familias que han sufrido pérdida de empleo, pobreza o enfermedad.

Imagen tomada de <http://seriousplaypro.com>, Sitio web dedicado a mostrar el uso de *Lego® Serious Play®* [Consultado: Julio 2015].

El sector de la salud por su parte se ha visto beneficiado de los simuladores de entrenamiento para estudiantes como se detalla en (Marcano, 2008) y en (Sun Center of Excellence for Visual Genomics & Department of Biological Sciences, 2009), donde *CAVEman* [9] se presenta como un atlas anatómico pudiendo el usuario navegar y ver diferentes perspectivas de los órganos. Como se observa en la Figura 3-6, esta

[9] Se pueden encontrar un breve resumen de las características acerca de *CAVEman* en el siguiente enlace: <http://techdigest.tv/2007/05/caveman_provide.html>

herramienta proporciona a los estudiantes mayor precisión para moverse por el área de trabajo y mayor coordinación ojo-mano.

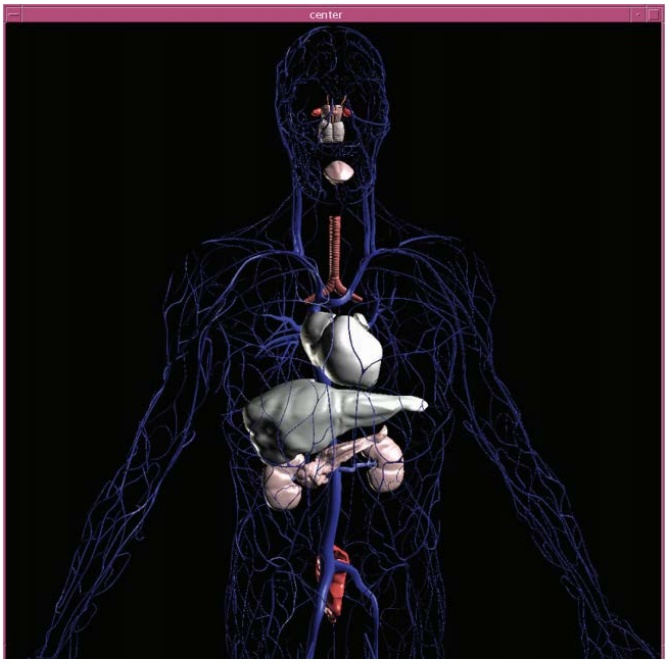


Figura 3-6: Visualización 3D de un cuerpo con *CAVEman*.

Imagen Tomada del documento de **(Sun Center of Excellence for Visual Genomics & Department of Biological Sciences, 2009)**.

En el área de la psicología experimental también se han desarrollado proyectos en los que se emplean juegos serios para el tratamiento de problemas cognitivos. Como es el caso de la aplicación *Brain Challenge Eval* de (Montañana Aliaga & Lemus Zúñiga), cuyas principales pantallas de juego se pueden observar en la Figura 3-7. Esta aplicación posee las siguientes pruebas cognitivas:

- Búsqueda visual: Es una prueba que consiste en la búsqueda de la letra B dentro de un conjunto de letras pudiendo aparecer en diferentes posiciones. Se mide el tiempo que tarda el usuario en encontrarla.
- *Stroop*: Es una prueba basada en el experimento desarrollado por John Ridley Stroop. Como se explica en el artículo *Stroop Effect* de (Wikipedia, 2015), el método utiliza el hecho de que el proceso de análisis semántico del texto es automático mientras que identificar el color del texto requiere capacidades cognitivas, lo cual puede inducir a errores cuando no hay correspondencia entre texto y color.
- *Probe dot*: Es una prueba basada en el método creado por MacLeod, Mathews y Tata en 1986. En esta adaptación se requiere en primer lugar que el usuario fije la atención en el centro de la pantalla, luego aparece una flecha indicándole dónde debe fijar su atención hasta que aparezca una estrella. Se mide el tiempo que tarda el usuario en decidir en qué lugar (izquierdo o derecho) apareció la estrella para detectar en qué lado se encuentra el *foco atencional* del sujeto.

- *Digit Span*: Es una prueba cuyo objetivo es medir cuántos ítems, en este caso dígitos, puede repetir una persona en el orden correcto inmediatamente después de que la aplicación los muestra. La longitud aumenta con los aciertos del usuario.

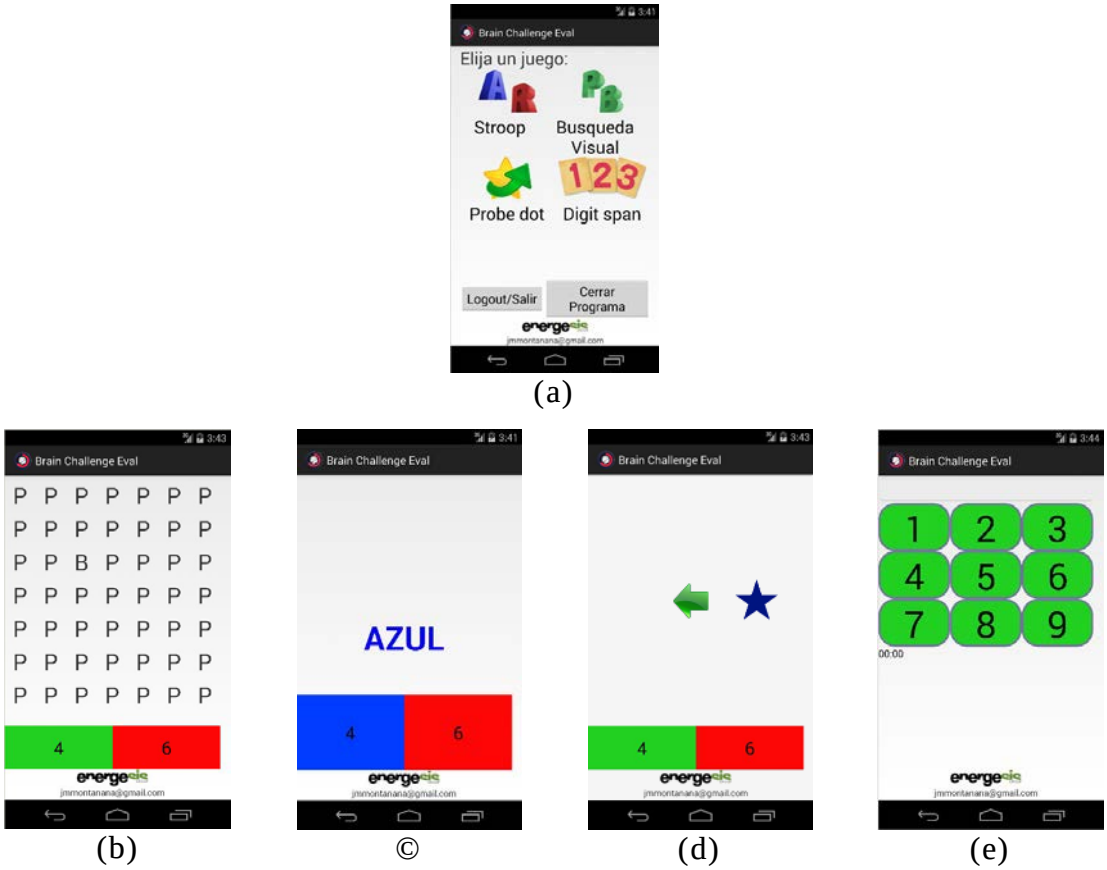


Figura 3-7: Capturas de pantalla de la aplicación móvil *BrainChallengeEval*, (a) Menú principal de juego inteligente, (b) juego *Digit span*, (c) juego *Stroop*, (d) juego Búsqueda visual, (e) juego *Probe dot*.

En el área de la educación también existen ejemplos de interés como el caso de *Aprende y juega con EA* [10], proyecto dedicado al estudio y exposición de resultados del uso didáctico de los videojuegos.

Incluso el área militar se beneficia de los juegos y simuladores. Tal es el caso de *America's Army* [11], desarrollado como herramienta para captar y entrenar soldados demostrando que mejora el aprendizaje, la velocidad de reacción, y la comunicación, sin dejar de lado que ofrece una mayor seguridad ya que los fallos o errores no conllevan pérdidas humanas o de material costoso. En la Figura 3-8 podemos observar una muestra del aspecto de los gráficos 3d de este videojuego.

[10] Se pueden consultar más detalles de nuevos proyectos y resultados de *Aprende y juega con EA*, así como ver algunas imágenes de participantes y algunos videos relacionados en el sitio web oficial: <http://www.aprendeyjuegaconea.com/>

[11] Algunos videos de demostración de *America's Army* así como las últimas características añadidas al juego se pueden hallar en el sitio web oficial: <http://www.americasarmy.com>

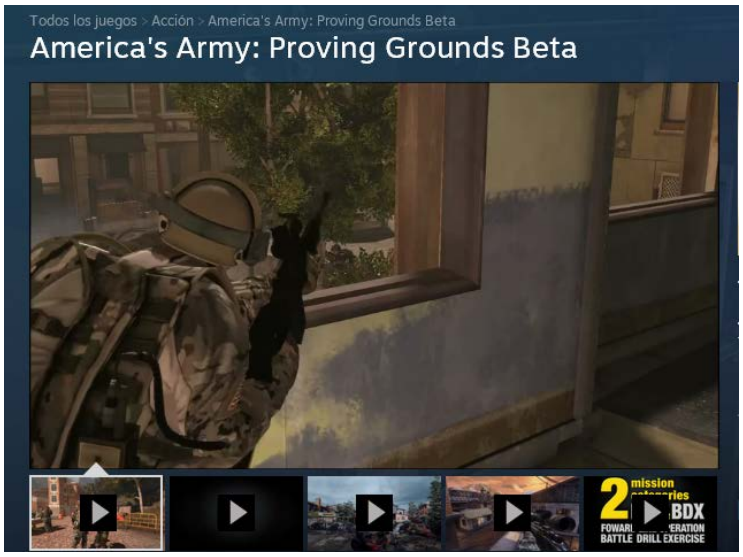


Figura 3-8: Captura de pantalla del video de demostración de escenas del videojuego *America's Army*, disponible en el sitio web de la tienda en línea *Steam*.

Video disponible en: <http://store.steampowered.com/app/203290>[Consultado: Julio 2015]

Como vemos, el número de áreas de uso de los juegos serios sigue creciendo y los propósitos para los que se crean son variados. Cada vez más se pone de manifiesto la necesidad de encontrar nuevas formas o instrumentos afines a los avances de las tecnologías de la información y que también consigan llenar las expectativas de las nuevas generaciones. Nuestro camino se dirige entonces hacia el área de la salud mental, concretamente hacia los juegos serios relacionados con la realización de pruebas cognitivas y la captura de tiempos de reacción que posteriormente podrían ser analizados.

2.5 Conclusiones

Los estudios que hemos descrito en los apartados anteriores revelan algunos de los problemas al usar los tiempos de reacción, como los problemas de retardos temporales introducidos por características propias del hardware o del software, los cuales, como hemos visto, no limitan la captura ni el análisis de los tiempos de reacción recogidos. En el caso de la presentación del estímulo, sabemos que actualmente la tecnología más utilizada son las pantallas de cristal líquido de transistores de película fina (*TFT-LCD* por sus siglas en inglés). Las cuales presentan una frecuencia de refresco de la imagen que, generalmente, es de 60 Hz, por lo que aún existe un retardo en la presentación de la imagen estímulo. Sin embargo, como mencionamos en apartados anteriores, este retardo no es significativo y los datos recogidos pueden analizarse.

Por otro lado, la proliferación de herramientas para desarrollar aplicaciones web ha dado pie a que muchos estudios de psicología experimental hayan creado versiones multimedia de experimentos ya realizados, pues a pesar de los inconvenientes como la no supervisión de la realización de la prueba, los retardos introducidos debidos al software, o las diferentes características de los dispositivos con los que es posible acceder a una misma aplicación, etc., la gran cantidad de datos que se pueden recoger, así como la poca influencia de los retardos, justifican la creación de estos experimentos en versión multimedia para su uso a través de internet.

Tras haber mencionado los problemas que pueden existir tanto en el diseño del experimento, en la captura de los tiempos de reacción, o en el análisis de los datos recogidos, tenemos que mencionar que solventar estos problemas requiere la realización de tareas específicas y por tanto están más allá del propósito de este proyecto, el cual pretende analizar de forma genérica los inconvenientes derivados del uso del tiempo de reacción como variable de estudio, encontrar las facilidades que algunos entornos de desarrollo ofrecen para la creación de juegos serios, y abordar la implementación de un juego serio que ayude a elevar el grado de validez ecológica en la realización de las pruebas.

Las características en las que se debe poner hincapié son aquellas que se derivan de las condiciones realización de las pruebas cognitivas. Es decir, puesto que durante la realización de una prueba cognitiva, el participante puede verse expuesto a interferencias tanto internas como externas a la aplicación, buscamos minimizar las interferencias internas que pueden perturbar el tiempo de reacción mediante el diseño de escenarios sencillos, intuitivos y que intenten elevar el grado de inmersión del participante. La minimización de las interferencias externas como el ruido de la sala, deben ser controlados por supervisores de la prueba en la medida de lo posible, cuando se requiere que los datos recogidos contengan un mínimo de errores debidos a interferencias externas.

Es por ello que la mayoría de aplicaciones destinas a realizar pruebas cognitivas presentan escenarios sumamente sencillos con la intención de no llamar la atención con elementos como, por ejemplo objetos animados innecesarios que se mueven por el escenario, cambios bruscos de colores al pasar de una escena a otra, escenarios no intuitivos, etc. Otra condición importante en la realización de estas pruebas es que primero se debe instruir al participante sobre la tarea a realizar, ya sea con texto en la aplicación o de forma supervisada por una persona. Incluso, en algunos casos debido a la complejidad de la prueba, es conveniente la realización de ensayos para la familiarización del participante en el uso de la aplicación. Así en una sesión posterior la captura de los tiempos de reacción se obtiene con menos errores debidos al desconocimiento del uso de la aplicación por parte del participante.

Actualmente, son muchos los ejemplos de videojuegos que presentan un nivel elevado de detalle en sus escenarios debido al uso de gráficos 3d, técnicas de iluminación,

texturas, teselado, etc. Por ello en el presente proyecto buscamos una manera de emplear herramientas para la generación de escenarios en 3d con la intención de reducir las distracciones del participante y elevar el grado de atención. Abordamos la creación de un escenario generado con gráficos 3d que tiene la intención de elevar el interés del participante durante la realización de la prueba y mantener lo atento al presentarle una escena que puede ocurrirle en su día a día.

Como vemos muchas de las características propias de los videojuegos, como escenarios sumamente elaborados, efectos especiales, etc., deben ser utilizados de forma coherente cuando se emplean para la presentación de una prueba cognitiva. Un ejemplo de contraste lo podemos hallar comparando los juegos serios *Sai fah: The Flood Fighter* de la Figura 1-1 y *Brain Challenge Eval* de la Figura 3-4. En el primero se puede observar el uso de muchas de las técnicas de diseño propias de videojuegos de entretenimiento aplicadas a un juego serio en el que la intención es dar lecciones de supervivencia al jugador mediante ensayo-error. Mientras que el segundo, presenta escenarios sencillos en los que la intención es retar al participante a responder lo más rápido posible. Como consecuencia, algunas de estas técnicas de animación que dan vida propia a objetos del escenario no se utilizan en muchas de las aplicaciones que miden el tiempo de reacción para mantener consistencia en los datos recogidos.

3 Evaluación de herramientas software

Para experimentar las facilidades de diseño e implementación que se pueden encontrar al utilizar una determinada herramienta software para crear juegos serios basados en los conceptos descritos en el apartado “Juegos serios como herramienta para pruebas cognitivas”, consideramos la implementación de la prueba cognitiva de *Stroop* como una labor que nos acercará a estas experiencias. La intención es dar luz a las tareas de programación de la emisión de estímulos, la captura de las respuestas, la generación de aleatoriedad, la medición de los tiempos de reacción, etc. Posteriormente estas experiencias se aplicarán en el desarrollo de la aplicación de la tarea de decisión léxica.

El motivo principal de la elección de esta prueba es que reúne algunas características referentes al diseño del experimento comunes a otras pruebas cognitivas que miden las respuestas de los participantes. A la vez la sencillez del funcionamiento del escenario agiliza su implementación. A continuación resumimos algunas de éstas características.

- Variable independiente manipulada: en nuestro caso la correspondencia color-palabra.
- Variable dependiente: en nuestro caso el tiempo de reacción (también podría ser la tasa de aciertos).
- El error de medida no tiene gran influencia en la validez interna del experimento: en nuestro caso los retardos debido al software son despreciables frente a los tiempos de reacción y la cantidad de capturas realizadas.
- Neutralización de variables extrañas mediante el reparto equitativo de su influencia a través de los niveles de variable manipulada: en nuestro caso repartimos un igual número de elementos en el conjunto de estímulos con discrepancias entre color y palabra así como en el conjunto sin discrepancias.
- Aleatoriedad para conseguir una distribución uniforme de las variables extrañas o fuentes de confusión: en nuestro caso mezclamos de forma aleatoria los estímulos del conjunto sin discrepancias con los del conjunto con discrepancias y los guardamos en una lista.
- Constancia en la presentación: en nuestro caso no existen pantallas intermedias entre la recepción de la respuesta y la presentación del siguiente estímulo.

En cuanto a las herramientas seleccionadas podemos decir que buscamos en ellas características determinadas: que faciliten el diseño del escenario, que permitan compartir contenido a través de internet de forma rápida, que ayuden a incrementar el grado de validez ecológica, y que permitan exportar la aplicación para su uso en dispositivos móviles. Por ello seleccionamos tres tecnologías: *Flash®*, *VRML/X3D*, *y Unity®*. En las conclusiones de este aparado se mencionarán las características en las

que cada una se destaca, así como al finalizar cada implementación se mencionarán las ventajas e inconvenientes encontrados.

Puesto que existen varias maneras de medir el efecto *Stroop* en el siguiente apartado vamos a delimitar las características de nuestra versión en formato multimedia buscando una implementación rápida y sencilla. En los apartados posteriores describiremos cada herramienta y algunos detalles destacables sobre el proceso de implementación de la prueba.

3.1 Prueba de Stroop a implementar

En primer lugar tenemos que mencionar que el efecto de *Stroop* descrito en 1935 por *John Ridley Stroop*, se refiere a la dificultad que presentan los observadores a la hora de eliminar información que puede ser irrelevante y que entra en conflicto con otra información durante la realización de una tarea. Esta interferencia, al igual que describimos en el apartado de ejemplos en la aplicación de (Montañana Aliaga & Lemus Zúñiga) y cuyo aspecto se observa en la Figura 3-7, se puede experimentar con la prueba de reconocimiento del color tras presentar un estímulo color-palabra y a la vez midiendo el tiempo de reacción.

Actualmente son varios los ejemplos que podemos encontrar de esta prueba en formato multimedia y que se encuentran disponibles en la red, como el que describimos en el apartado “Uso en aplicaciones web” referente al uso del tiempo de reacción y que se puede observar en la Figura 3-4. La mayoría pretenden revelar a los usuarios que acceden a ellos, los errores o las interferencias que se producen en su mente cuando se lee la palabra en lugar de acertar el color, dejando de manifiesto la inmediatez con la que se realiza el proceso de lectura, mientras que el proceso de decisión resulta ser más complejo.

Observamos también que un gran número de estas aplicaciones para realizar la prueba de *Stroop* disponibles en sitios web tan solo pretenden hacer que el usuario descubra este efecto, es decir, proporcionarle información. Nuestra implementación va en la misma línea ya que buscamos hacer un recorrido superficial por los elementos multimedia presentes en una determinada herramienta que nos ayuden en la creación de esta prueba. El almacenamiento y análisis de los datos recogidos no serán abordados dejando esta labor para ser abordada en la implementación de la tarea de decisión léxica de apartados posteriores.

3.1.1 Escenario

Las versiones multimedia de la prueba de *Stroop* que vamos a describir están basadas en una interfaz sencilla donde el punto importante es mostrar al participante estímulos conformados por un color y una palabra. En definitiva se trata de presentar texto en dos dimensiones cuyas letras tiene color de relleno. Tras la emisión del estímulo se mide el tiempo de reacción a través de la recepción de los eventos de pulsación de los botones de colores.

Estos elementos multimedia que permiten la interacción entre usuario y aplicación en algunos casos pueden ser construidos de forma rápida con la ayuda de un editor de escenarios presente en la herramienta software. Además pueden ser manipulados mediante código para cambiar propiedades como el color, el tipo de fuente, el contenido, la posición, etc.

Como sabemos la presentación de texto en pantalla es una funcionalidad explotada desde hace mucho tiempo en muchas aplicaciones para transmitir instrucciones a los usuarios, por lo que es fácil encontrar elementos multimedia que faciliten su uso y manipulación mediante la modificación en tiempo de ejecución de las propiedades. Lo mismo ocurre con los botones, son elementos empleados recurrentemente para controlar (*avanzar, retroceder, iniciar, terminar, confirmar, etc.*) el flujo de ejecución mediante la interacción con el usuario.

La organización espacial del escenario presentará en la parte superior un estímulo color-palabra, y la parte inferior contendrá los botones de colores para ser pulsados. Los colores de relleno de los botones disponibles del escenario se deben corresponden fielmente con los colores de relleno que pueden aparecer en los estímulos. Así mismo, el significado semántico de las cadenas de caracteres almacenadas como parte del estímulo también se deben corresponder únicamente con los colores posibles que las letras pueden tener, es decir, no podemos tener una cadena de caracteres "*negro*" si las letras de los estímulos nunca van a tener este color. En el mismo sentido se debe vigilar que las cadenas de caracteres ocupen el mismo espacio y no cambien su tamaño o tipo de fuente de un estímulo a otro pues esto puede distraer al participante afectando su tiempo de reacción.

La Figura 4-1 nos muestra un ejemplo de la organización que tiene el escenario para realizar la prueba de *Stroop* en versión multimedia. La sencillez que presenta se debe a que se deben minimizar las posibles distracciones del participante limitando el uso de características disponibles en el entorno que, en casos en los que se busque llamar la atención del usuario con elementos vistosos, darían más dinamismo al escenario, como por ejemplo, las animaciones, la interpolación de movimiento, el diseño más elaborado de los elementos presentes, etc.

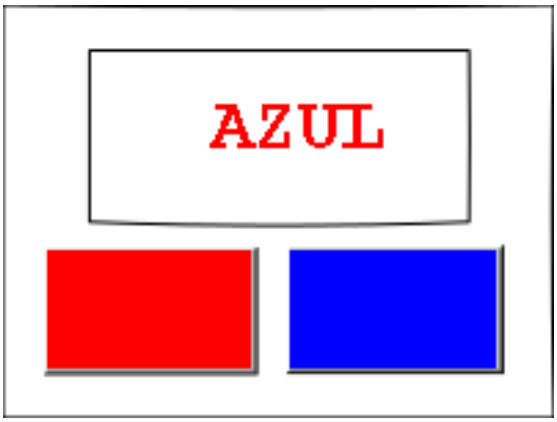


Figura 4-1: Ejemplo de organización del escenario para realizar la prueba de *Stroop* empleando estímulos conformados por color- palabra y botones de los colores correspondientes para capturar las respuestas.

3.1.2 Funcionamiento

El funcionamiento interno en cuanto al control de la interactividad, en líneas generales, se basa en la recepción de los eventos de pulsación de los botones. Al inicio de la ejecución de la aplicación se generan los estímulos que se van a ir presentando y se mezclan de forma aleatoria los estímulos sin discrepancias con los estímulos con discrepancias, en igual proporción. De esta forma, son los eventos de pulsación de los botones los que disparan la ejecución de las funciones que realizan el control de la interacción y la captura de los tiempos de reacción. La Figura 4-2 nos muestra brevemente un diagrama de transición de estados para aclarar este funcionamiento. Destacamos algunas de las tareas que se llevan a cabo internamente cuando ya se ha iniciado la prueba: capturar los instantes de emisión y recepción, determinar el tiempo de reacción, determinar si es un acierto, guardar y mostrar el siguiente estímulo.

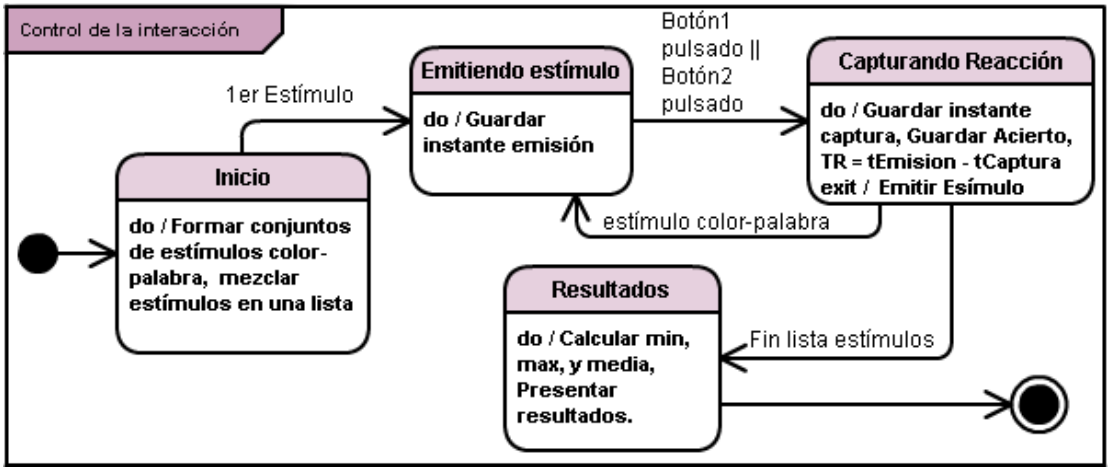


Figura 4-2: DTE del control de la interacción con el usuario que se realiza internamente en la aplicación que permite realizar la prueba de *Stroop*. Este diagrama de transición de estados intenta mostrar brevemente la manera en la que se capturan los tiempos de reacción en las aplicaciones desarrolladas en la toma de contacto con determinadas herramientas software. Las flechas nos indican la transición de un estado a otro al darse la aparición de un evento el cual es indicado por la etiqueta de la flecha.

Este funcionamiento es abordado con los elementos multimedia disponibles y lenguaje de programación adecuado en cada una de las plataformas elegidas. En algunos casos encontraremos que existe una similitud notable en el comportamiento de algunos elementos, estructuras de datos y objetos predefinidos de los diferentes entornos para medir el tiempo de reacción y generar aleatoriedad.

En los apartados siguientes pasaremos a describir las herramientas software y los pasos seguidos en el diseño del escenario y la programación del control de la interactividad en cada una de ellas.

3.2 Flash®

La tecnología *Flash®* ha sido durante mucho tiempo la más empleada en sitios web para añadir a éstos diferentes funcionalidades como publicidad, decorados, formularios, etc., lo cual hacía que sean más atractivos. Actualmente su presencia se mantiene a pesar de la aparición de otras tecnologías que intentan suplirla en áreas como la reproducción de video en los navegadores, como es el caso de *Html5*. Además su presencia en videojuegos de entretenimiento en línea y su uso para la reproducción de video por medio de difusión bajo demanda (*streaming*), entre otros usos, aún es común.

Uno de sus ventajas principales es que solventa el problema de la compatibilidad con los navegadores ya que, ante el problema recurrente en el diseño web sobre la manera independiente en la que cada navegador presenta los contenidos, *Flash®* asegura que la presentación de los mismos sea exactamente como se crearon. Esta característica se da gracias a la manera en la que se complementan dos herramientas propias de esta tecnología, es decir, por un lado *Adobe® Flash®* permite crear aplicaciones multimedia y por otro *Adobe® Flash® Player* las reproduce.

Esta tecnología es conocida por el uso de gráficos vectoriales que, a diferencia de los gráficos basados en *raster*, mantienen la calidad al ampliar el tamaño de una imagen a voluntad, cosa que no ocurre con las imágenes *raster* (*bmp*, *jpeg*, *png*, *etc.*) las cuales al ampliarse demasiado suelen presentar el efecto de *aliasing* o degradación en su visualización. Un ejemplo de aplicación que usa imágenes vectoriales es *Google® Maps* la cual al ampliar el tamaño de una zona del mapa no presenta degradación en la visualización.

Por otro lado esta tecnología posee un lenguaje propio para la programación de aplicaciones conocido como *ActionScript* y se trata de una interfaz de programación de aplicaciones (*API* por sus siglas en inglés) que reúne un conjunto de objetos destinados a facilitar la animación como *Button*, *MovieClip*, *Sprite*, *Scene*, *Timer*, *etc.* Actualmente esta *API* puede ser utilizada en sus versiones *2.0* y *3.0* las cuales tienen la

programación orientada a objetos como una de sus características principales para que las aplicaciones sean más eficientes. Las diferencias entre las versiones dan puntos a favor y en contra que pueden justificar el uso de una u otra versión. Por ejemplo la versión 3.0 tiene mejoras en cuanto al control de eventos al tener un solo sistema que los controla, mientras que la versión 2.0 permite asignar código directamente a un objeto del escenario dando mayor intuición al programar el comportamiento de los objetos. En el mismo sentido la versión 3.0 incluye clases de objetos que permiten programar funcionalidades presentes en los dispositivos móviles como el uso del acelerómetro para mover objetos, la entrada táctil, etc.

Para corroborar el uso de *Flash* podemos mencionar a modo de ejemplo el juego serio *Re-mision 2*. El cual es una colección de juegos en línea que ayuda a jóvenes y niños con cáncer a luchar contra su enfermedad. Éste ha tenido un gran impacto en el área de la salud y ha sido merecedor de varios premios pues ayuda a estos pacientes a mejorar su estado emocional. En esta colección de juegos las protagonistas son las diferentes formas de terapia que se presentan en forma de personaje animado que destruye a las células cancerígenas. En la Figura 4-3 podemos observar el aspecto de este videojuego.

Figura 4-3: Captura de pantalla del juego serio en línea *Re-mision 2* que ayuda a niños y jóvenes con cáncer a tener un estado emocional más positivo frente a las terapias.
Captura realizada en el sitio web oficial: <http://www.re-mission2.org/games/>. [Consultado: Julio 2015]

Como vemos esta tecnología nos ofrece una manera rápida de compartir aplicaciones que pueden ser accedidas por los usuarios a través de internet. Para ello existen herramientas que permiten el desarrollo de este tipo de aplicaciones. Nosotros nos centraremos en uno de los entornos de desarrollo integrado que ya tiene una trayectoria de uso en la creación de aplicaciones *Flash*. A continuación mencionaremos las características más importantes que éste posee.

3.2.1 Adobe® Flash® Professional

Adobe® Flash® Professional es un entorno de desarrollo integrado de la compañía *Adobe® Systems Incorporated* que usa la tecnología de *Adobe® Flash®* comúnmente conocida como *Macromedia Flash* o *Shockwave Flash* y ampliamente usada en videojuegos de entretenimiento en línea.

El entorno permite integrar imágenes en fondos, decorados, botones personalizados, etc. Así mismo, permite integrar en la escena otros tipos de elementos multimedia como, por ejemplo, el texto, el audio, los videos, la difusión bajo demanda o comportamientos como arrastra y soltar. De forma la creación de escenas es rápida.

Actualmente *Adobe® Flash® Professional* proporciona herramientas para exportar las aplicaciones desarrolladas hacia otras plataformas[12], entre ellas *Html5* y *WebGL*, con lo que se pretende que el uso de la tecnología *Flash* se mantenga. En el mismo sentido, a pesar que *Android* e *iOS*, las dos plataformas móviles con mayor número de usuarios, no ofrecen soporte directo para la ejecución de aplicaciones *Flash*, el entorno de desarrollo integrado en su versión más actualizada, *Adobe® Flash® Professional CC®*, sí permite exportar aplicaciones para ejecutarse en esta plataformas. Para ello se basa en "incluir" el soporte *AIR (Adobe Integrated Runtime)* [13] sobre el que se ejecuta *Flash* junto con la aplicación desarrollada. Así mismo, ofrece posibilidad de que la aplicación no lleve incluido este *runtime* si no que durante la instalación se descargue de la tienda, *GooglePlay* en el caso de *Android*, reduciendo así el espacio ocupado por la aplicación.

Entrando en los detalles de la interfaz del entorno podemos decir que éste presenta un espacio de trabajo dinámico pudiendo el usuario desplegar y esconder paneles de herramientas. Algunos de ellos están destinados a facilitar el diseño de escenas, asistir en la escritura de código, organizar los objetos del proyecto, etc.

En la Figura 4-4 se puede observar el aspecto visual de la configuración por defecto de la interfaz del entorno, donde podemos encontrar las paneles principales: el editor de línea de tiempo, el editor de escenas, el panel de propiedades, filtros, y parámetros de un objeto seleccionado, el panel de colores para el diseño de figuras manualmente, y el panel de la biblioteca de objetos presentes en el proyecto.

[12] El tutorial del siguiente enlace detalla los pasos para publicar un fichero ejecutable *apk* de *Android* usando *Adobe ® Flash Professional CS6*: <https://www.youtube.com/watch?v=En14YZt-e8k>. [Consultado: Julio 2015]

[13] El funcionamiento es similar al *Java Runtime Environment* necesario para ejecutar aplicaciones de *Java*. En este caso la intención es que las aplicaciones desarrolladas para ejecutarse sobre *AIR* no dependan del sistema operativo del dispositivo.

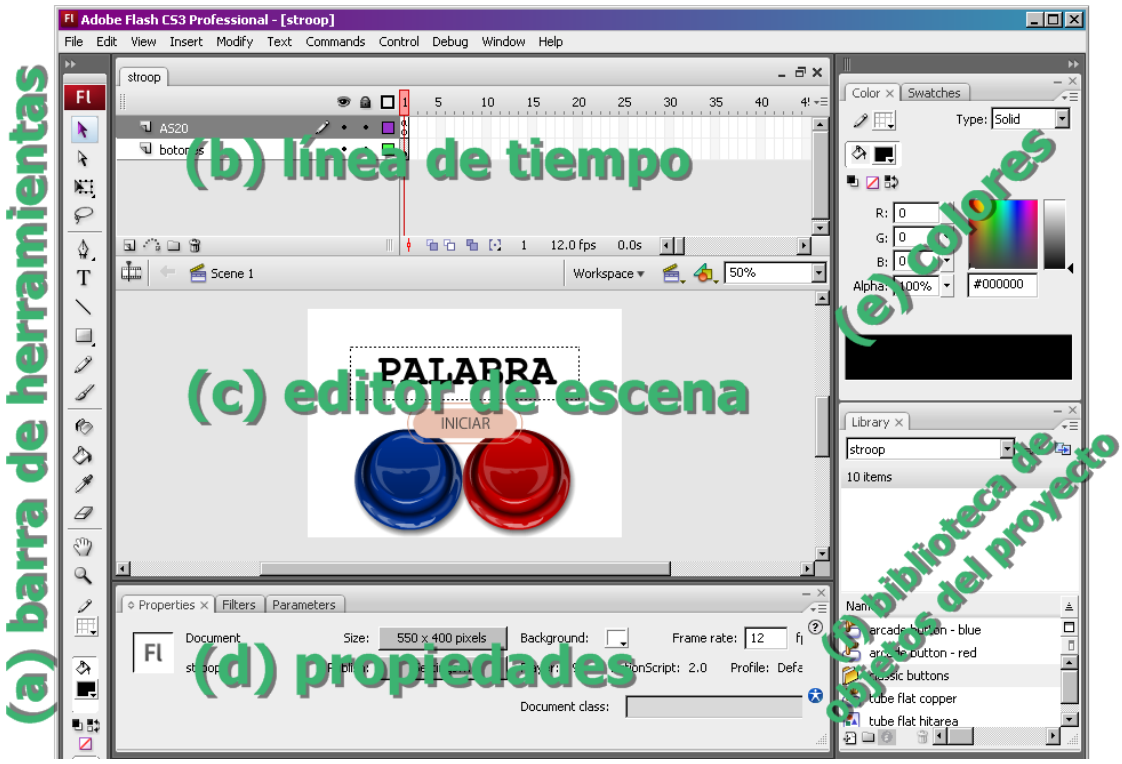


Figura 4-4: Captura de pantalla de la interfaz de usuario por defecto del entorno de desarrollo *Adobe® Flash® CS3 Professional*. Se muestran las siguientes ventanas: (a) barra de herramientas, (b) línea de tiempo, (c) editor de escenas, (d) panel de propiedades, filtros y parámetros de un objeto seleccionado, (e) panel de colores para el dibujado, (f) biblioteca de objetos presentes en el proyecto.

Puesto que el propósito de este proyecto es emplear de forma rápida algunas de las herramientas disponibles en este entorno para nuestro cometido de crear una prueba de *Stroop*, algunas de las herramientas, que en otras aplicaciones multimedia pudieran ser de interés, quedarán fuera de nuestro recorrido. Nos centraremos en la línea de tiempo y la animación mediante interpolación.

El panel de línea de tiempo permite una organización jerárquica por capas. La Figura 4-5 (parte superior) nos muestra un ejemplo de ello, donde los elementos colocados en un fotograma perteneciente a una capa superior se visualizarán por encima de los colocados en un fotograma perteneciente a una capa inferior. A su vez permite la reorganización de capas moviendo hacia arriba o hacia debajo de toda la línea de fotogramas de la capa. El funcionamiento es, básicamente, como la creación de una película conformada por escenas y fotogramas, donde el concepto fundamental a tener en cuenta es que la reproducción se realiza gracias al avance de una cabeza lectora principal que recorre los fotogramas principales de la escena, los cuales a su vez pueden contener objetos que tienen sus propios fotogramas.

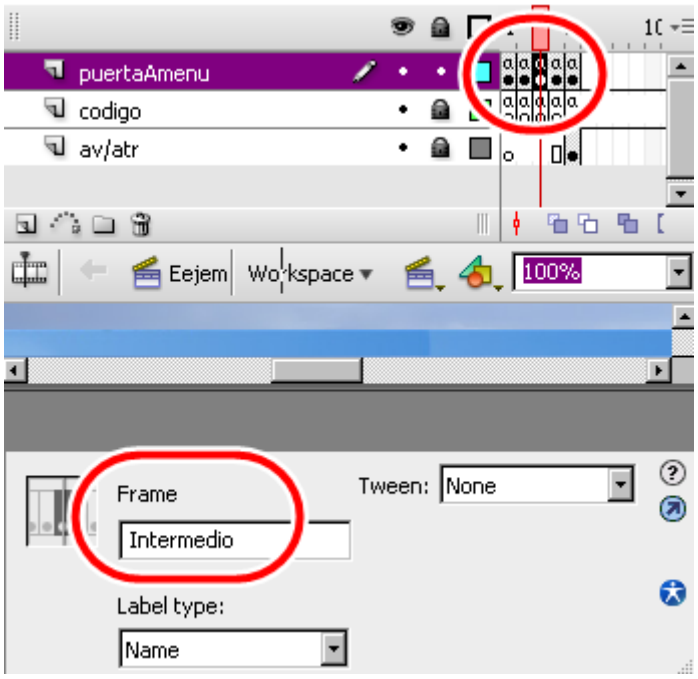


Figura 4-5: Captura de pantalla que muestra un ejemplo de etiquetado de un fotograma y organización por capas en *Adobe® Flash® CS3 Professional*. El tercer fotograma de la capa "*puertaAmenu*" se etiqueta como "*Intermedio*". Este nombre de etiqueta permite saltar a este fotograma desde otro referenciándolo por su etiqueta.

Para el control del flujo de la reproducción el entorno permite establecer etiquetas o nombres de fotograma. De esta forma podemos usar instrucciones en *script* para "saltar" de un fotograma a otro mediante el nombre de etiqueta. Lo mismo ocurre con las escenas, podemos controlar el orden de reproducción de éstas indicando que al llegar la cabeza lectora a un fotograma determinado, se debe saltar, por ejemplo, al primer fotograma de otra escena. Incluso se puede usar este mecanismo para controlar los objetos del escenario mediante código. La Figura 4-5 (parte inferior) nos muestra un ejemplo de etiquetado.

En cuanto a la interpolación de propiedades de un objeto podemos decir que el entorno permite crear la interpolación entre dos fotogramas clave, de forma que el tamaño, el color o la posición, entre otros parámetros, varía del valor dado en el fotograma inicial al valor dado en el fotograma final. Incluso usando el panel de edición de escena (letra c en la Figura 4-4) tenemos la posibilidad de establecer un recorrido que debe seguir el objeto. Para ello se puede usar una capa espacial denominada capa guía o *GuideLayer* en la que podemos crear la ruta deseada mediante el trazado de líneas de forma manual en el escenario. Los objetos a desplazarse se anclan al punto de inicio y de fin del trazo. La Figura 4-6 muestra varios instantes capturados que dejan ver los cambios producidos en un objeto gracias a la interpolación.

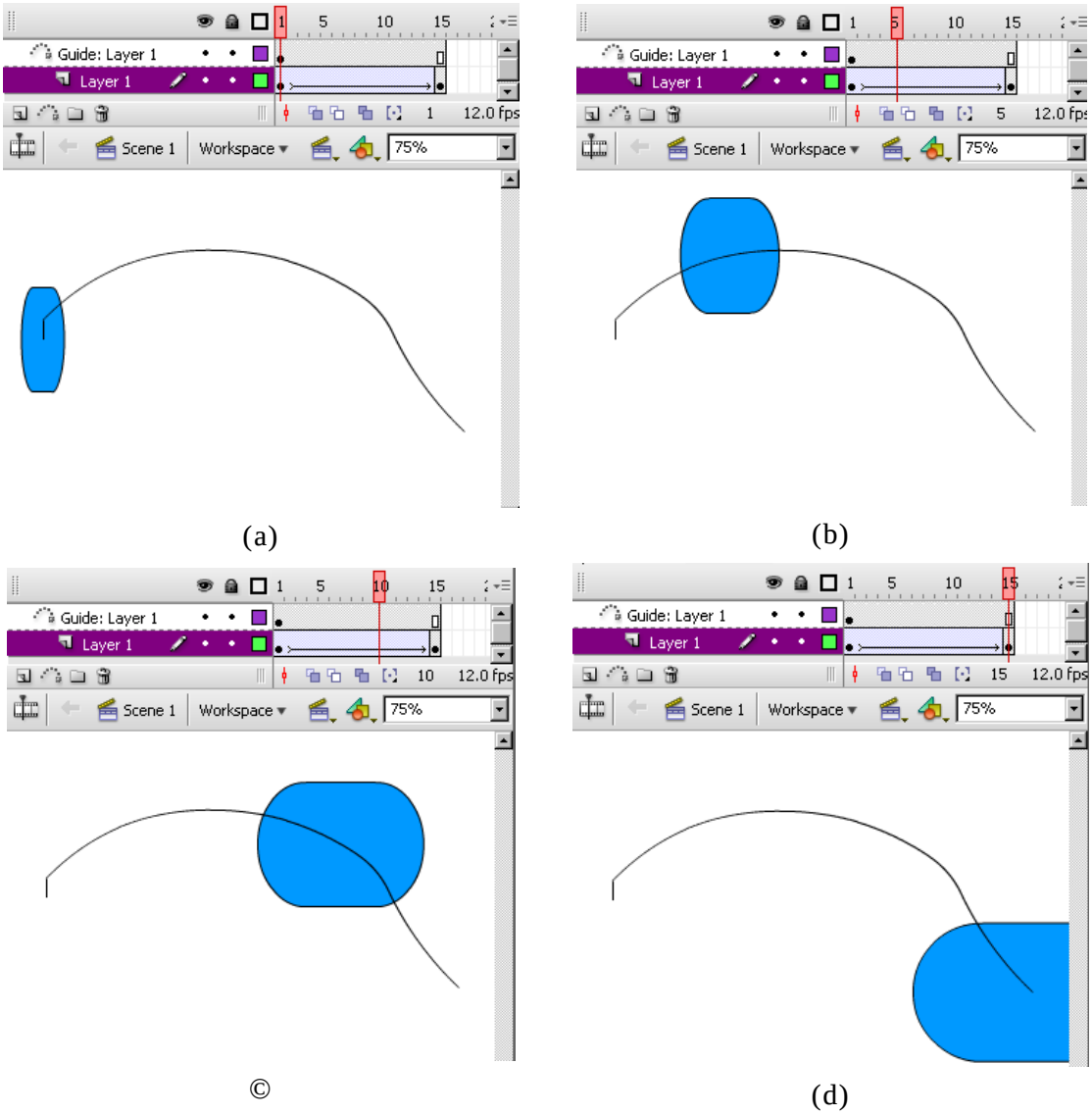


Figura 4-6: Capturas de pantalla que muestran un ejemplo de interpolación de propiedades (posición, forma, y tamaño) de un objeto en la línea de tiempo para generar animaciones y su visualización en el panel de edición de escena de *Adobe® Flash® CS3 Professional*. Trazamos una línea de guía para el movimiento. (a) Captura en el fotograma 1. (b) Captura en el fotograma 5. (c) Captura en el fotograma 10. (d) Captura en el fotograma 15.

Tras mencionar las características principales en el siguiente apartado vamos a describir algunos de los conceptos propios de la programación con el lenguaje *ActionScript* utilizado en la implementación de la prueba de *Stroop* en conjunción con este entono.

3.2.2 Implementación

La implementación de la prueba de *Stroop* llevada a cabo se ha divido en dos partes: el diseño del escenario y el control por código. La primera contempla 3 zonas o áreas visuales en las que se colocan los objetos y se les asigna etiquetas, y la segunda contempla el uso de funciones y clases de *ActionScript 2.0* para el control de la interacción mediante las etiquetas asignadas.

3.2.2.1 Escenario

Para el diseño del escenario nos hemos servido de algunos de los objetos que la *API* de *ActionScript 2.0* y el entorno nos ofrecen: principalmente los objetos *TextField* y *Button*.

En primer lugar en **el área del estímulo** (parte superior del escenario) hemos empleado un objeto de tipo *TextField*. El cual, en resumen, es un cuadro en el que se incluye el texto y puede presentar 3 configuraciones: estático, dinámico, y de entrada. El estático sirve únicamente para mostrar texto en una determina área del escenario. El dinámico permite modificar propiedades como el tipo de fuente, el color, el tamaño, la justificación, etc. El de entrada sirve principalmente cuando se quiere establecer un campo que debe ser rellenado por el usuario y recoger esa información como, por ejemplo, un campo en el que se debe introducir una contraseña de acceso.

Nos hemos servido del texto dinámico al que le hemos asignado una etiqueta mediante la cual podremos modificar sus propiedades. Además en los parámetros de este objeto hemos tenido la posibilidad de aplicar, entro otros, un filtro *anti-aliasing*[14] enfocado a la lectura y no a la animación.

En segundo lugar en **el área de la captura** de las respuestas (parte inferior del escenario) nos hemos servido de uno de los elementos que se pueden crear de manera rápida y fácil con este entorno. Se trata del objeto *Button* el cual tiene un funcionamiento igualmente basado en línea de tiempo con la salvedad de que tan solo presenta cuatro fotogramas. Éstos simbolizan los cuatro eventos que un botón puede tener en esta *API*: ser pulsado, ser soltado, tener el puntero en su área, o ser golpeado por otro objeto.

Nosotros nos centramos en los tres primeros ya que han sido los que han facilitado la captura de las respuestas para medir el tiempo de reacción. Estos tres fotogramas establecen sus nombres como *"Up"*, *"Over"*, y *"Down"*. Puesto que se trata de fotogramas, éstos permiten usar todas las opciones de edición de un fotograma normal.

[14] Con este filtro el reproductor *Adobe® Flash® Player* representará el texto sin problemas de degradación en la visualización evitando el conocido efecto "escalera" o *pixelado* al ampliar el tamaño del reproductor.

Por ello se pueden editar los fondos del escenario y no solo la forma del botón. Incluso es posible incorporar un fichero de sonido a uno de estos fotogramas y se reproducirá cuando el fotograma se reproduzca.

El comportamiento de los eventos de un botón queda resumido a continuación:

- Cuando el puntero del ratón se coloca encima se lanza la reproducción del fotograma *Over*.
- Cuando es pulsado se lanza la reproducción del fotograma *Down*.
- Cuando es soltado se lanza la reproducción del fotograma *Up*.

Sin embargo el entorno nos ofrece también objetos predefinidos almacenados en una biblioteca, la biblioteca de objetos comunes. Aquí podemos encontrar botones, clases, e interacciones. La mayoría de los botones pueden presentar un aspecto visual que no deseamos (anticuado, demasiado llamativo, etc.) o tal vez quisiéramos mejorar uno de sus fotogramas eliminando o añadiendo elementos, en ese caso acudimos a la edición de los fotogramas añadiendo o eliminando objetos y capas como mencionamos anteriormente.

En nuestro caso nos hemos servido de tres botones de colores de la biblioteca de objetos comunes a los cuales les asignamos etiquetas que servirán para programar el control de la recepción de los eventos de los botones. Los botones seleccionados no presentaban texto en su interior, como suele ser habitual, y estaban disponibles en los tres colores que requeríamos para la prueba: rojo, verde y azul.

Finalmente en **el área de los resultados**, que es un escenario o pantalla que aparecerá al finalizar las capturas, hemos empleado un *TextField* de tipo dinámico en el que mostramos una traza de las iteraciones, el mínimo, el máximo y la media de los tiempos de reacción. Además en este caso, hemos empleado la clase *Graphics* para la representación gráfica de un histograma que da una idea de la distribución de los tiempos capturados, lo cual será descrito en el siguiente apartado ya que el dibujado se realiza mediante instrucciones de esta clase.

En la Figura 4-7 podemos ver el aspecto visual de la organización de los elementos multimedia en las diferentes pantallas de la aplicación de la prueba de *Stroop* una vez exportada como película *Flash*.

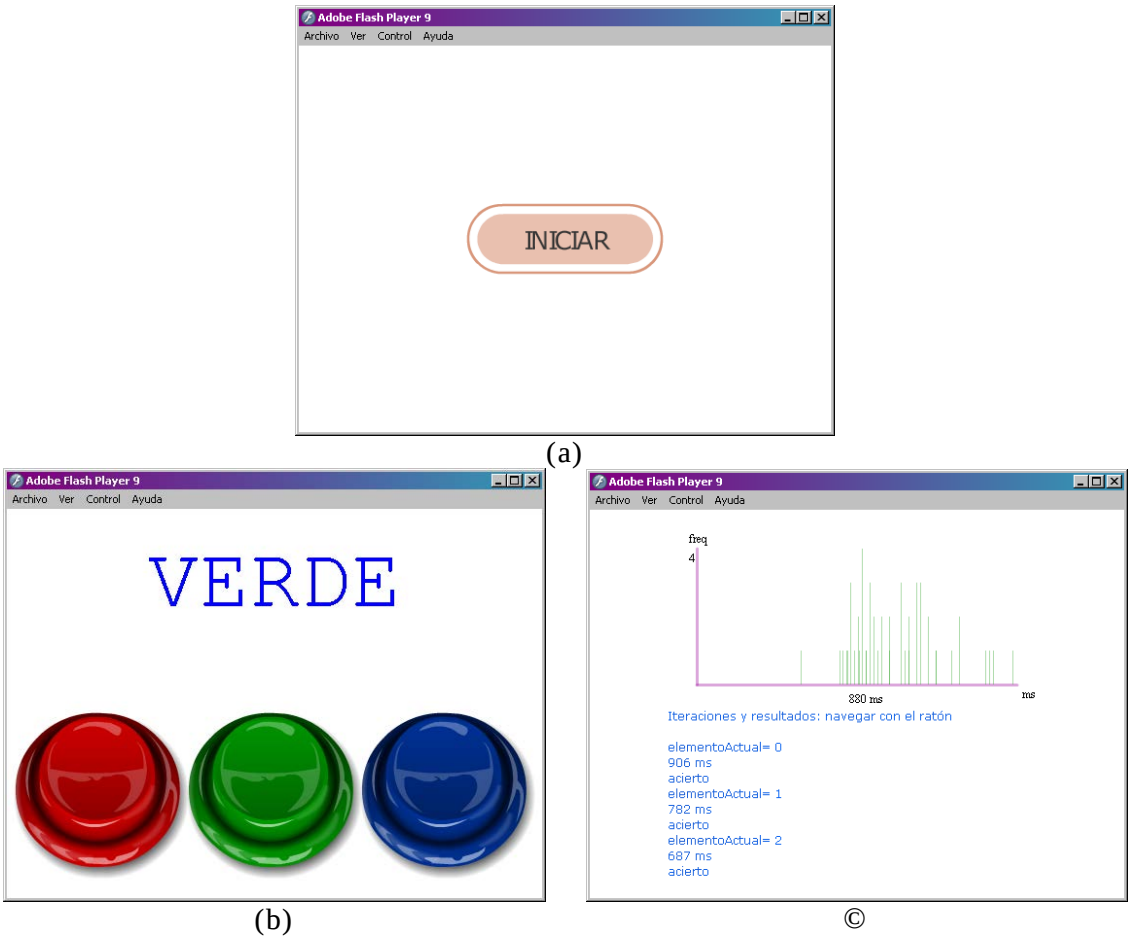


Figura 4-7: Capturas de pantalla de la aplicación implementada con *Flash* y *ActionScript* para realizar la prueba de *Stroop*[15]. (a) Pantalla inicial: construcción de estímulos y asignación de aleatoriedad. (b) Pantalla de emisión de estímulos y captura de respuestas: se almacenan los instantes de emisión y pulsación de los botones. (c) Pantalla de resultados: se calculan el mínimo, el máximo, la media de los tiempos de reacción y se dibuja un histograma.

Una vez configurado el escenario de forma manual el siguiente paso es el control de los objetos mediante código así como la previa configuración de estímulos en cuanto a su color y la cadena de caracteres contenida, entre otras configuraciones. En el siguiente apartado describimos estos pasos.

3.2.2.2 Control con ActionScript

La segunda parte de la implementación es el control de la aplicación, en definitiva, de los objetos presentes en la pantalla, mediante código. Para esta tarea hemos utilizado dos paneles que por defecto se encuentran ocultos en el espacio de trabajo: el panel de acciones y el panel de salida por consola.

[15] Actualmente la aplicación que permite realizar la prueba de *Stroop* implementada con *Flash* y *ActionScript* se encuentra disponible en el siguiente enlace: <http://giotitoa.esy.es/pfc/stroop/flash/stroop.html>. [Julio 2015]

El panel de acciones integra una ventana de acceso rápido a métodos y funciones de las clases de *ActionScript*, una ventana de acceso rápido a los *script* colocados en otros fotogramas y una ventana de asistente de código. En la Figura 4-8 podemos observar el aspecto del panel de acciones cuyas características principales son: permite verificar la sintaxis, resalta funciones y palabras reservadas, posee historial de acciones, e indica el nombre de capa, número de fotograma y número de línea del *script*.

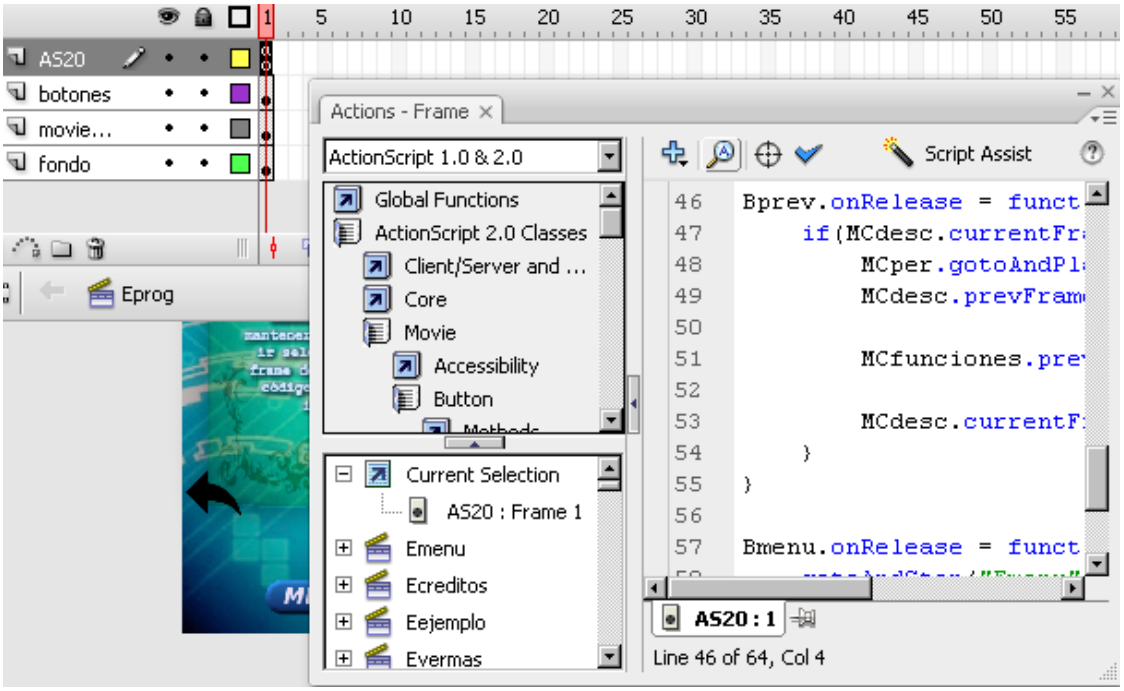


Figura 4-8: Captura de pantalla que muestra un ejemplo de uso del panel de acciones de *Adobe® Flash® CS3 Professional* para escribir código en un *script*. El script corresponde al fotograma "*1*" de la capa "*AS20*" de la imagen.

El panel de salida por su lado ha permitido ir encontrando errores e ir obteniendo trazas de las iteraciones durante la implementación. Para obtener estas salidas en texto hemos empleado la función "*trace*" que permite imprimir por consola una cadena de caracteres que se la pasa como argumento. Este panel no es más que una ventana típica de salida por consola que se despliega cuando se ejecuta una instrucción que imprime texto por consola.

Una vez hemos mencionado los paneles que agilizan la programación vamos a mencionar los conceptos propios del lenguaje *ActionScript 2.0* con el que se ha programado esta prueba de *Stroop*. Éstos los mencionaremos a continuación de forma separada para cada una de las tareas que la aplicación lleva a cabo internamente.

Construcción de estímulos: *ActionScript 2.0* permite construir estructuras de datos de tipo *Array* cuyos elementos pueden ser de cualquier tipo (*String*, *float*, *Number*, *Object*, *Button*, *MovieClip* *etc.*), incluso de tipo *Array*. Esto da una gran facilidad para construir estímulos conformados por un valor numérico que representan el color de los

pixeles y una cadena de caracteres. De forma que nuestro estímulo color-palabra viene a ser un *Array* con dos elementos, uno de tipo *Number* y uno de tipo *String*, cuyos valores dependerán de si se quiere que el estímulo tenga discrepancia o no. Por ejemplo para un estímulo sin discrepancia se establecería los valores `[0x00FF0000`[16]`, "ROJO"]`. Al final tenemos un *Array* de estímulos (un *Array* cuyos elementos son *Array*) que contiene todos los estímulos posibles que se van presentar en la prueba de tal forma que existe el mismo número de estímulos sin discrepancia que con discrepancia.

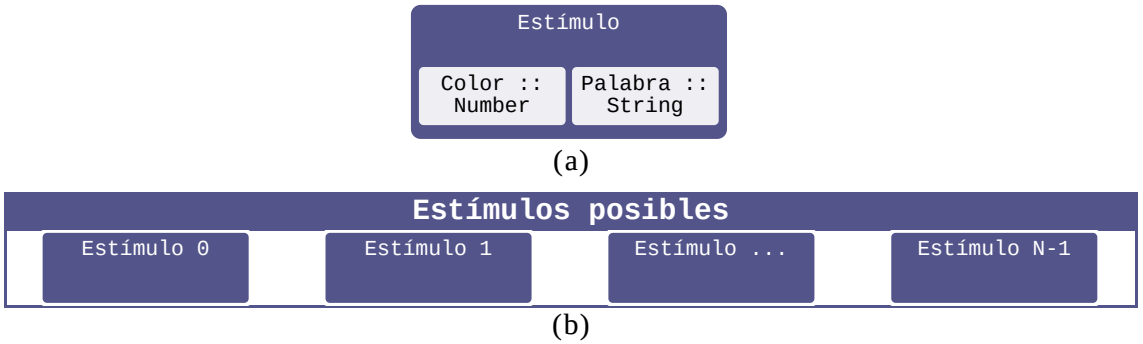


Figura 4-9: Configuración de los tipos de datos que conforman un estímulo color-palabra (a) y Array de estímulos posibles (b) en nuestra prueba de *Stroop*.

Aleatoriedad: los elementos tipo *Array* disponen de métodos como *push* para encolar un nuevo elemento mientras que el paquete de funciones matemáticas *Math* nos proporciona un gran número de operaciones entre las cuales encontramos la función *random*. Ésta función genera valores aleatorios en coma flotante en el intervalo [0,1] que, mediante operaciones matemáticas, se pueden convertir en valores que estén en el intervalo entre la primera y última posición del *Array* de estímulos posibles. De forma que la tarea a realizar es copiar los "estímulos" en otro *Array* usando la función *random* para distribuirlos aleatoriamente. De este último iremos extrayendo uno a uno los estímulos para presentarlos en la pantalla.

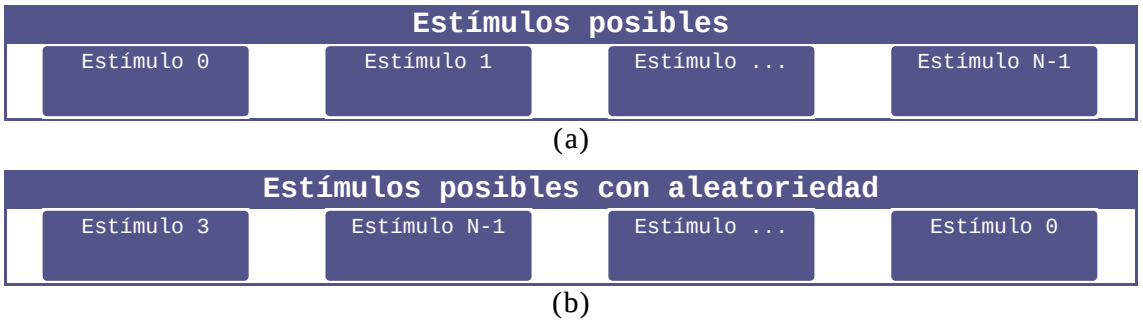


Figura 4-10: Ejemplo del resultado de aplicar la función *random* para conseguir aleatoriedad en la lista de estímulos que se van a presentar en nuestra prueba de *Stroop*. (a) *Array* original. (b) *Array* con elementos en posiciones aleatorias.

[16] En *ActionScript 2.0* los valores de colores se expresan en hexadecimal. De forma que asignando un valor hexadecimal mediante código a la propiedad "*color*" de un objeto se modifica el color del mismo.

Emisión de estímulos: según el número de estímulo en curso se comprueba si la el *Array* de estímulos posibles está vacío en cuyo caso se pasa a calcular los resultados. En otro caso modifican las propiedades del *TextField* para que aparezca el siguiente estímulo. Es decir, se consultan los valores de color y palabra del estímulo en el *Array* de estímulos posibles y se asigna éstos a las propiedades "*textColor*" y "*text*" del *TextField* presente en el escenario. El siguiente paso consiste en capturar el instante de emisión. Para ello se emplea el método *getTime* de la clase *Date* el cual nos devuelve el número de **milisegundos** pasados desde el 1 de enero de 1970. Finalmente la aplicación se queda a la espera de recibir un evento de pulsación de alguno de los botones, los cuales lanzarán el proceso de captura de la reacción e informarán a éste del color pulsado.

Captura de las reacciones: al iniciarse este proceso se captura el instante con el método *getTime*. Luego se calcula el tiempo de reacción en milisegundos como la diferencia entre el instante de emisión y el instante de captura y se encola en un *Array* junto con su correspondiente color pulsado. Es decir el *Array* "*reacciones*" contendrá elementos del tipo `[1454, 0x00FF0000]` que servirán para hacer un recuento posterior y calcular la tasa de aciertos. El último paso es lanzar nuevamente el proceso de emisión indicándole que se debe consultar el siguiente estímulo.

Eventos de los botones: los botones disponen de cuatro eventos como habíamos comentado en el apartado Escenario. A nosotros nos interesa únicamente el evento de *pulsado* ya que debemos lanzar el proceso de captura y el siguiente proceso de emisión sin esperar a que el botón sea *soltado*. Para ello al evento *onPress* del botón referenciado por su etiqueta le asignamos una función cuya única tarea es iniciar el proceso de captura pasándole el valor numérico del color del botón pulsado.

Resultados: el proceso se inicia una vez que se ha terminado de recorrer el *Array* de estímulos posibles. Se realizan los cálculos necesarios sobre el *Array* de reacciones y el de estímulos para obtener el mínimo, el máximo, la media de tiempos de reacción y el número de aciertos. Se incluye estos datos al texto de iteraciones que se ha ido acumulando y finalmente se presenta todo a través del *TextField* dispuesto para ello. Además en este caso se realiza el dibujado de un histograma para permitir observar visualmente los tiempos capturados. Para ello se acude a la clase *Graphic* presente en *ActionScript* la cual dispone de métodos para dibujar líneas, poli líneas, polígonos, etc. En nuestro caso fue de interés el uso de las funciones *moveTo* y *lineTo* con las que se traza los ejes y las barras del histograma. Así mismo, la creación de los títulos de los ejes se realiza con un objeto de tipo *TextField* dentro de la película *Flash* empleando la función *createTextField*.

Control de escenas: para el control del paso de una pantalla a otra, se ha empleado el atributo "*visible*" de los objetos, de forma que en la pantalla de inicio tan sólo se deja "ver" el botón de inicio. En la pantalla de la prueba se muestran los botones y el cuadro de texto superior. Y en la de resultados sólo se muestra el cuadro de texto de resultados inferior y el dibujado del histograma.

Como vemos la implementación de esta prueba empleando esta plataforma ha tenido varios apartados que nos han permitido conocer las principales herramientas disponibles en él. Tras esta toma de contacto con el entorno pasaremos a describir las ventajas e inconvenientes.

3.2.3 Ventajas e inconvenientes

A continuación resumimos las ventajas:

- La combinación entre la edición manual de los objetos y el control de los mismos mediante *script*, permite reducir el tiempo de desarrollo.
- Permite saltar entre escenas ante un determinado evento, lo cual se puede utilizar para desarrollar etapas de una aplicación separadas y unirlas mediante puertas de enlace (generalmente botones).
- La existencia de un depurador para eliminar errores facilita en gran medida la corrección de las aplicaciones en las primeras etapas de programación.
- La información en la red sobre el uso específico de herramientas de este entorno así como de funciones presentes en *ActionScript* es abundante. La guía de referencia de *ActionScript*, los tutoriales y los foros de discusión ayudan en la resolución de inconvenientes. (Esto se debe a que durante años la tecnología *Flash* fue empleada exhaustivamente en el desarrollo web).
- Existen clases, en la versión 3.0, que permiten hacer uso de las propiedades del hardware de los dispositivos móviles. Tal es el caso de la clase *Accelerometer* [17] del paquete *flash.sensors*, que se puede emplear para mover objetos de la escena leyendo los valores del sensor con métodos de esta clase.
- Existe soporte para la conexión con bases de datos. Un ejemplo de ello es la clase *SQLConnection*, que permite leer e insertar valores.

A continuación resumimos los inconvenientes:

- Se precisa tener conocimientos de programación relacionados con lenguajes de scripting y orientado a objetos, como *JavaScript*, *C#*, o similares. Ya que en ocasiones precisamos realizar tareas específicas como almacenar los tiempos de

[17] Se puede ver un video que muestra un ejemplo de ejecución de una aplicación en un dispositivo móvil con plataforma *Android* que emplea el acelerómetro para mover un objeto, en el siguiente enlace: <https://www.youtube.com/watch?v=IOFnhqyoiSo>. [Consultado: Julio 2015]

reacción, esperar eventos, conectar con una base de datos, gestionar la aleatoriedad, etc.

- No ofrece soporte para crear entornos 3D navegables por el usuario. En su lugar ofrecen herramientas para crear "efectos 3D"[18] sobre clips de película (objeto *MovieClip*). Se trata de una técnica 2.5D que depende de un punto de fuga.
- La dependencia del *plug-in* de *Adobe® Flash® Player* o del *runtime AIR* es un punto en contra frente a otras herramientas que permiten exportar las aplicaciones a un formato que no dependa de complementos.
- El coste económico al ser una herramienta de autor puede ser un punto en contra frente a entornos gratuitos.

3.3 VRML/X3D

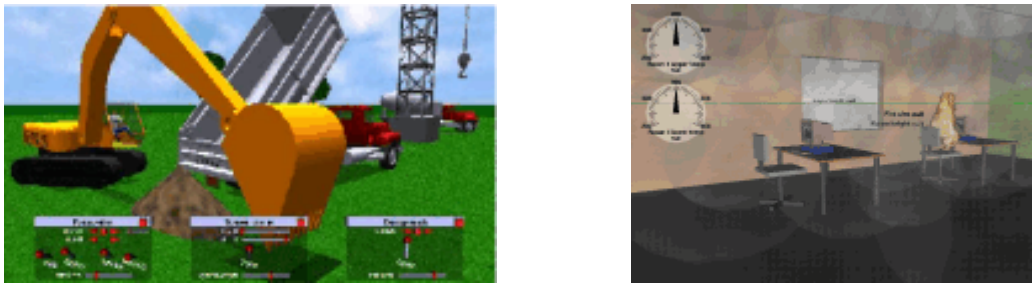
VRML (*Virtual Reality Modelling Language*) es un lenguaje de modelado de realidad virtual que forma parte de los estándares desarrollados por el *Web3D Consortium* y es la base sobre la que se ha desarrollado la versión sucesora *X3D* (*eXtensible3D*). En líneas generales, *X3D* añade la codificación *XML* a *VRML* así como *shaders*, geo-localización, y otros aspectos de vanguardia para dar soporte a diferentes áreas como medicina, sistemas de información geográfica (*GIS* por sus siglas en inglés), diseño asistido por computador (*CAD*), aplicaciones multiusuario, etc. Si bien *VRML* empezó siendo un formato de fichero y un lenguaje para describir formas y entornos interactivos, *X3D* permitió adaptarlo para una mejor integración con el resto de tecnologías web.

Esta tecnología fue creada como un lenguaje simple para construir mundos virtuales para internet en los que el usuario puede moverse e interactuar con los objetos presentes. Con ésta podemos crear animaciones, sonido espacial, representar rápidamente primitivas básicas o formas complejas, visualizar modelos 3D provenientes de otras herramientas de tipo *CAD*, etc. Así mismo ofrece facilidades para gestionar la interacción con el usuario ya que existen por un lado nodos de tipo *sensor* y por otro, nodos de tipo *interpolador* los cuales pueden conectarse para animar objetos. Éstos y otro gran número de nodos permiten elevar el grado de abstracción al programar mundos virtuales. Incluso se pueden crear funcionalidades muy complejas cuyos resultados se pueden ver reflejados en pantalla gracias a la existencia de un nodo de tipo *Script* que permite incrustar en su interior fragmentos de código *Java* o *JavaScript*.

[18] Se pueden observar ejemplos de esta técnica en la página web de *Adobe® Flash® Professsional*: <https://helpx.adobe.com/es/flash/using/3d-graphics.html>. [Consultado: Julio 2015]

El uso de *VRML/X3D* está presente en áreas en las que se precisan ambientes virtuales para simular tareas específicas. A continuación mencionaremos algunos ejemplos.

- Realidad virtual en proyectos de construcción: la necesidad de encontrar nuevas formas de transmitir la información de los procesos de construcción, como por ejemplo los documentos de las fases del proyecto y los planos CAD 2D, para instruir y mejorar las capacidades de los empleados, puede satisfacerse con esta tecnología, ya que gracias a los gráficos tridimensionales y las capacidades que ésta posee se pueden explorar virtualmente áreas de una determinada construcción, simular comportamientos de materiales y estructuras, permitir el acceso multiusuario a través de una interfaz web, simular el manejo de maquinaria pesada, etc. La Figura 4-11 nos muestra un ejemplo de ello.
-



(a) (b)

Figura 4-11: Ejemplo de uso de *VRML/X3D* en procesos de construcción. (a) Modelo de excavadora: simula el movimiento, colisiones y una articulación controlada por el usuario. (b) Simulación de un sistema de ventilación.

Imágenes tomadas de: <http://rebar.ecn.purdue.edu/ect/links/technologies/internet/default.aspx>. [Consultado: Julio 2015]

- Realidad virtual en el ámbito académico: las nuevas tecnologías permiten crear nuevas formas de transmitir conocimientos de forma más precisa y amena. Un ejemplo lo podemos ver en la Figura 4-12 que presenta una aplicación dedicada a abordar conceptos de la geometría descriptiva utilizando un ambiente virtual que permite al usuario cambiar el punto de vista para visualizar mejor los objetos, manipular elementos geométricos, etc. Este tipo de aplicaciones proporcionan un mejor entendimiento de lo que se suele ver reflejado en papel (dos dimensiones), como en este caso el abatimiento de rectas sobre planos de proyección.

Figura 4-12: Captura de pantalla que muestra un ejemplo de la ejecución de la aplicación "Geometría descriptiva en realidad virtual" que emplea *VRML* en el ámbito académico.

Capturada realizada en el sitio web:
<http://webdelprofesor.ula.ve/ingenieria/pfaraujo/personal/Geometr%EDa%20Descriptiva%20en%20%20VRML/Inicio/Inicio.html>. [Consultado: Julio 2015]

- Realidad virtual para la simulación de procesos mecánicos: el problema de acceso a sistemas mecánicos específicos en el mundo real hace que el aprendizaje de los mismos se dificulte. Es aquí donde *VRML/X3D* puede ser de utilidad al permitir incluir piezas modeladas en 3D dentro del mundo virtual y dotarlas de comportamientos dinámicos, facilitando la intuición del funcionamiento de procesos mecánicos concretos y reduciendo el coste económico. Un ejemplo de ello lo podemos ver en la Figura 4-13.

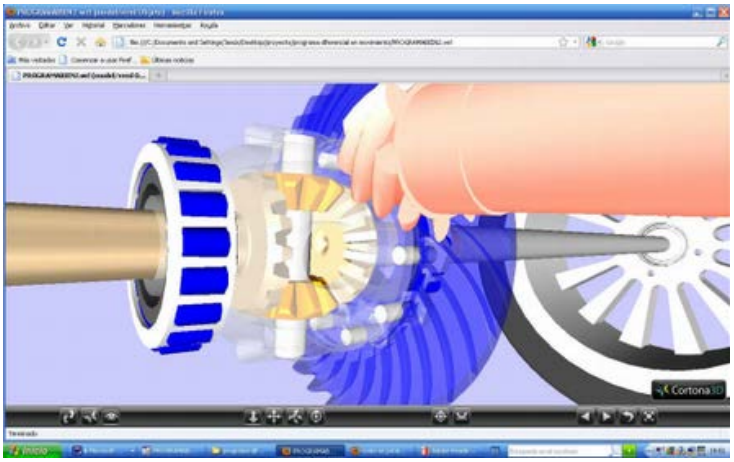


Figura 4-13: Captura de pantalla que muestra un diferencial mecánico simulado en *VRML*.

Imagen tomada de: <http://e-archivo.uc3m.es/handle/10016/16630>. [Consultado: julio 2015]

Como vemos esta tecnología de realidad virtual facilita la creación de entornos de simulación interactivos que pueden ser usados a través de internet. En nuestro caso nos enfocaremos en aplicarla para crear un videojuego que permita realizar la prueba de *Stroop*. Así podremos conocer las facilidades que nos ofrece a la hora de configurar el experimento, emitir los estímulos, controlar la recepción de las respuestas, compartir a través de internet, etc.

En cuanto a las especificaciones del lenguaje, tanto *VRML* como *X3D* son estándar *ISO* de la web y proporcionan una infraestructura que permite la rápida creación de los mundos virtuales que puede observarse en un navegador web con un *plug-in* o complemento apropiado. Al tratarse de la misma infraestructura pasar de *VRML* a *X3D*, en cuanto a los conceptos de la jerarquía de nodos con los que se crean los mundos, es sencillo salvo que se deben añadir los conceptos propios del lenguaje de marcas extensible (*XML)*. Sin embargo resulta más intuitivo para quienes se inician en el uso de esta tecnología empezar con *VRML* pues permite centrarse en conocer y explotar las propiedades internas de los nodos dejando para un último paso la conversión a *X3D*, la cual suele ser de interés en aplicaciones que precisan dar soporte de conexión con bases de datos. En la Tabla 4-1 podemos corroborar la similitud que hemos mencionado.

Notación X3D

```
<?xml version="1.0" encoding="UTF-8"?>
<X3D profile="Immersive" version="3.0">
<head/>
 <!--Un cilindro rojo-->
 <Scene>
  <Shape>
   <Appearance>
    <Material diffuseColor='1.0 0.0 0.0'/>
   </Appearance>
   <Cylinder height='3' radius='1'/>
  </Shape>
 </Scene>
</X3D>
```

Notación VRML

```
#VRML V2.0 utf8
#Un cilindro rojo
Shape {
 appearance Appearance {
  material Material {
   diffuseColor 1.0 0.0 0.0
  }
 }
 geometry Cylinder{
  height 3
  radius 1
 }
}
```

Tabla 4-1: Ejemplo de similitud en la jerarquía de nodos usando *X3D* y *VRML*. Ambas notaciones definen un una primitiva básica dentro de una escena: un cilindro de color rojo, altura 3 y radio 1.

X3D posee una gran flexibilidad ya que, al respetar las especificaciones de *XML*, permite crear una estructura jerárquica de componentes, lo cual da la posibilidad de que se añadan innovaciones que surjan en el área de los gráficos 3D enfocados a entornos web. Además existen varias *API* que pretenden facilitar la integración de *X3D* con otras tecnologías. Ejemplo de ello son *Ajax3D*[19] y *Xj3D*[20]. La primera combina *SAI*, que es la interfaz de acceso a escenas de *X3D*, con el uso del *DOM*, que es una estructura de objetos que genera el navegador al cargar un documento web cuyos, y métodos de peticiones a servidor como *XMLHttpRequest* y *createX3DFromURL*. La segunda es un conjunto de herramientas escrito en *Java* y de código abierto que permite incorporar gráficos de *X3D* en aplicaciones *Java*.

Puesto que además se trata de un formato de fichero tan sólo se necesita un editor de texto plano para escribir la organización de los nodos y las instrucciones necesarias en *JavaScript* para programar el comportamiento de los objetos del mundo virtual.

[19] Podemos consultar algunos detalles en: <http://edutechwiki.unige.ch/en/AJAX3D>.

[20] Una breve descripción podemos encontrar en: <http://www.xj3d.org/tutorials/xj3d_application.html>.

Por otro lado también existen editores de mundos virtuales que permiten crear escenas y animaciones de forma manual. Sin embargo los fragmentos de código *JavaScript* que generalmente éstos generan para animar los objetos, acaba siendo ininteligible cuando queremos retocar o añadir funciones específicas, como en nuestro caso, para medir el tiempo de reacción, configurar los estímulos, etc. Por tanto en el apartado de implementación de la prueba de *Stroop* se mencionarán los pasos seguidos sin utilizar este tipo de editores y haciendo uso de la documentación en línea de las especificaciones del lenguaje y del visor *Cortona3D Viewer*, el cual veremos que nos facilita la corrección de errores y las pre-visualizaciones durante el desarrollo del mundo virtual.

3.3.1 Cortona3D Viewer

Cortona3D Viewer[21] es un visor de mundos virtuales escritos en *VRML* que se ofrece como *plug-in* para navegadores web y que puede ser utilizado de forma libre para uso académico, personal y no comercial. Éste interpreta las instrucciones del fichero y permite visualizar las formas 3D, navegar, e interactuar con los sensores de la escena así como usar herramientas del *plug-in* para la navegación (barra de herramientas en la parte inferior de la Figura 4-12).

Incorpora una consola que puede emitir mensajes de salida cuando encuentra un error de sintaxis indicando el nombre del fichero, el número de línea y una descripción del error. Incluso puede servir, al igual que hicimos con la función *trace* en el apartado "Control con ActionScript", para emitir mensajes de salida y llevar un control de la ejecución.

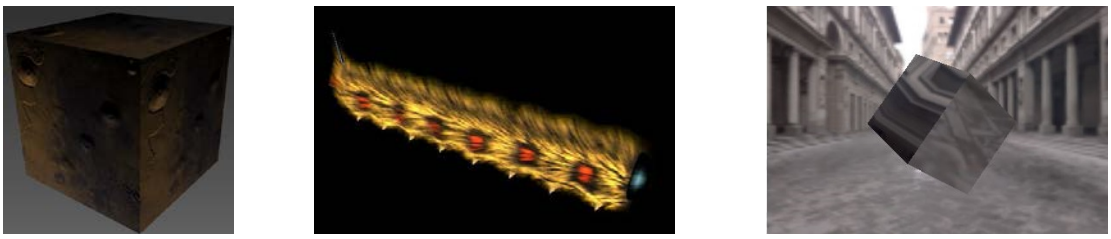
Este *plug-in* tiene parámetros establecidos por defecto que permiten la navegación del usuario. Sin embargo cuando se requiere tomar el control de ciertas características para restringir el movimiento de la cámara, el desplazamiento por la escena, el menú de opciones desplegable, etc., debemos especificarlo en los correspondientes nodos (*NavigationInfo, WorldInfo, etc.*) referentes a la información del mundo virtual y modificar algunos parámetros del *plug-in* embebido en una página web a través de la etiqueta *embed*[22] de *Hmtl*.

[21] Una descripción de las características del *plug-in* así como el acceso a su descarga se pueden hallar en la web oficial: <http://www.cortona3d.com/cortona3dviewer>. [Consultado: Julio 2015]

[22] Podemos encontrar más detalles de los parámetros de *Cortona3D Viewer* que se pueden manipular para restringir algunas propiedades de éste, en el siguiente enlace: <http://www.parallelgraphics.com/developer/products/cortona/html>. [Consultado: Julio 2015]

Además nos suministra una serie de nodos que incrementan las capacidades de *VRML*. Estas extensiones[23] no forma parte del estándar y por tanto solo pueden visualizarse con el navegador de *Cortona3D Viewer*. Puesto que su uso puede ser de interés para crear escenas más realistas vamos a mencionar algunos de éstos.

- *BumpMap*: permite añadir mapas de normales a las texturas de un objeto dando un efecto de "relieve".
- *NavigationInfo*: permite especificar el centro de rotaciones cuando se utiliza el tipo de navegación *Examine* y la distancia al plano de recorte delantero.
- Paneles y texto *Html*: permiten crear etiquetas 2D o paneles de información en la ventana 3D para visualizar texto formateado.
- *KdbSensor*: genera eventos basados en la entrada del teclado. (Junto a *JavaScript* puede permitir programar el control mediante teclado de la aplicación o de un objeto animado).



(a) (b) ©

Figura 4-14: Capturas de pantalla que muestran ejemplos de uso de las extensiones (nodos) de *VRML* en *Cortona3D Viewer*. (a) *BumpMap*. (b) *FragmentShader*. (c) *CubeEnvironment*.

Tras describir las características principales de esta herramienta en el siguiente apartado vamos a describir la manera de unir nodos para construir los botones y los estímulos así como el control de la prueba mediante código *JavaScript*.

3.3.2 Implementación

De la misma forma que en el apartado de Flash® la implementación de la prueba de *Stroop* llevada a cabo se ha divido en dos partes: el diseño del escenario con nodos de primitivas 3D básicas y el control por código con fragmentos de código *JavaScript* dentro de un nodo de tipo *Script*. La notación elegida es *VRML*97 (la última versión y la más cercana a *X3D*), ya que pretendemos dar una vista genérica sobre las posibilidades de esta tecnología de realidad virtual aplicada a la creación de un videojuego.

[23] Más detalles y ejemplos de las extensiones que permite *Cortona3d Viewer* podemos verlas en: <http://www.parallelgraphics.com/developer/products/cortona/extensions/>. [Consultado: Julio 2015]

Tanto el escenario como el comportamiento de la prueba se han creado empleando la técnica de prototipado, es decir, la creación de nuevos tipos de nodos definidos como *Proto* en los que *VRML/X3D* permite encapsular un mundo virtual o varios objetos animados para ser usados como parte de otros nodos. En éstos tenemos la posibilidad de crear una interfaz de entrada y salida en la que podemos especificar el tipo de campos (*MFStrinfg, SFBool, SFVec3f, etc*) que podrán ser consultados y modificados desde el exterior, o el tipo de eventos de entrada o de salida que queremos capturar o enviar. En la Tabla 4-2 podemos observar un ejemplo de interfaz de un nodo *Proto* para nuestra prueba de *Stroop*.

```
PROTO protoStroop [
       eventIn SFBool iniciar
       eventOut SFBool terminado
       eventOut MFString textoResultadosListo
       field SFVec3f tamBoton 2 2 0
       field SFFloat tamPalabra 1
       exposedField SFVec3f trasladoBotonRojo 1 0 0
       exposedField SFVec3f trasladoBotonVerde 0 0 0
       exposedField SFVec3f trasladoBotonAzul -1 0 0
       exposedField SFVec3f trasladoPalabra 0 2 0
       exposedField SFColor colorRespuestaRojo 1 0 0
       exposedField SFColor colorRespuestaVerde 0 1 0
       exposedField SFColor colorRespuestaAzul 0 0 1
       exposedField SFColor colorPalabra 1 1 1
       exposedField MFString letrasPalabra [”PALABRA”]
]{
    #Body
}
```

Tabla 4-2: Ejemplo de la interfaz de un nodo *Proto* en *VRML* en el que encapsulamos la prueba de *Stroop*. Los valores de los campos *exposedField* pueden ser establecidos externamente en el momento de crear una instancia de este *Proto*, El evento de entrada *iniciar* sirve para iniciar el proceso, los eventos *terminado* y *textoResultadosListo* sirven para informar de la finalización y transmitir los resultados al exterior.

3.3.2.1 Escenario

Para la construcción del escenario nos hemos servido de algunos de los nodos que nos permiten construir los botones de respuestas y emitir los estímulos, es decir, hemos incluido en un nodo de agrupamiento *Group* varios nodos principalmente de tipo *Text*, *Shape*, y *TouchSensor*.

En primer lugar en **el área del estímulo** se ha empleado un nodo de tipo *Shape* para incluir en éste una forma de tipo *Text* la cual permite observar texto 3D en la escena, cuyo contenido será manipulado mediante código para mostrar los estímulos color-palabra. Además para transmitir *feedback* visual al participante se ha creado una breve animación que hace notar el paso entre un estímulo y el siguiente ya que, en ocasiones debido a la aleatoriedad, se puede dar la situación de que se presenten dos estímulos del mismo tipo (idéntico color e idéntica palabra) de forma consecutiva. Puesto que la creación de animaciones es sencilla y puede ser de interés en la Figura 4-15 mostramos un esquema del funcionamiento básico del que resumimos lo siguiente: un nodo de tiempo *TimeSensor* emite *ticks* o fracciones de tiempo una vez que se activa, un nodo interpolador *PositionInterpolator* recibe *ticks* de tiempo y emite valores (*SFVec3f*, tres

valores en coma flotante) de posición interpolados entre valores clave preestablecidos, y las rutas *Route* permiten conectar un nodo emisor con varios nodos receptores mediante el encaminamiento de eventos.

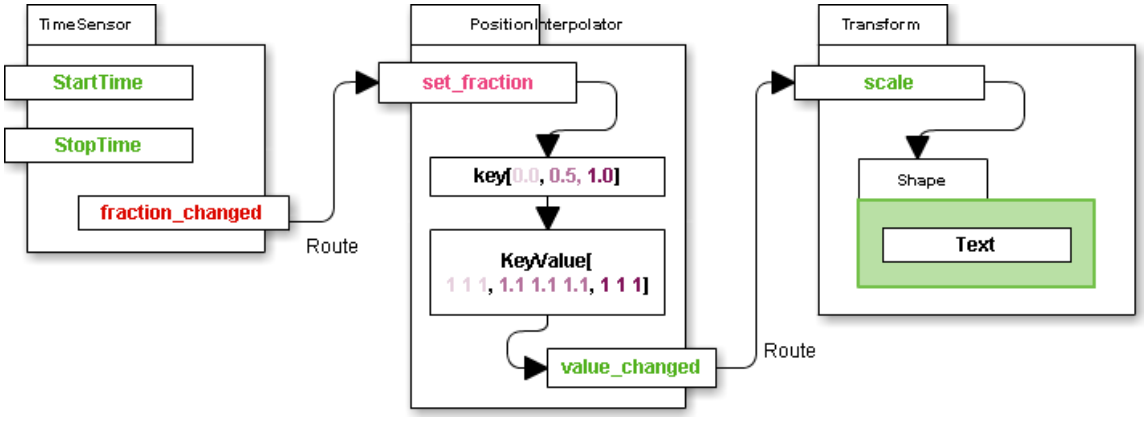


Figura 4-15: Diagrama de que muestra un ejemplo de conexión entre nodos para crear una animación básica en *VRML*. El nodo *Text* del ejemplo incrementará su tamaño un 10% en (x, y, z) durante una fracción de tiempo y volverá a su tamaño original, ya que al nodo *Transform* que lo contienen le llegan valores de tipo *SFVec3f* que modifican la escala.

En segundo lugar en **el área de la captura** de las respuestas se han empleado tres nodos de tipo *Shape* para incluir en ellos las formas de los botones (*Box*) y las imágenes de textura en el nodo de apariencia *Appearance* que todo nodo *Shape* tiene como hijo. En *Appearance* podemos cargar imágenes con canal alfa, por lo que se pueden incluir imágenes de botones con un fondo transparente, como en nuestro caso tres imágenes de botones con colores diferentes. Una vez definidas la forma se añade como hermano de ésta un nodo de tipo sensor de tacto *TouchSensor* el cual aplica de forma automática a sus nodos hermanos la propiedad de que puedan ser "tocados" por el usuario. Este sensor será de utilidad para medir los tiempos de reacción ya que emite eventos de tipo *SFBool* en el momento de su activación.

Finalmente **en el área de resultados** se emplea un nodo *Text* externo al nodo prototipo que contiene la prueba de *Stroop*. La manera de transmitir los resultados desde el nodo *Script* que realiza las capturas de los tiempos de reacción hasta el *Text* externo es mediante el envío de un evento de tipo *MFString* (un *Array* de varios *String*) que contiene toda la información recogida sobre los tiempos, los aciertos, el mínimo, el máximo y la media, ya que los eventos en *VRML* pueden ser de varios tipos: *MFString*, *SFBool*, *SFvec3f*[24], *etc*.

En la Figura 1-1 podemos ver el aspecto visual de las diferentes pantallas de la aplicación de la prueba de *Stroop* ejecutada sobre el *plug-in* de *Cortona3D Viewer*.

[24] Las iniciales *SF* definen un tipo simple *SingleField* y *vec3f* hace alusión a una estructura de 3 valores numéricos en coma flotante. Para más detalles de los tipos de eventos de los nodos podemos consultar la documentación en línea en la web oficial: <http://www.web3d.org/documents/specifications/14772/V2.0/index.html>. [Consultado: Julio 2015]

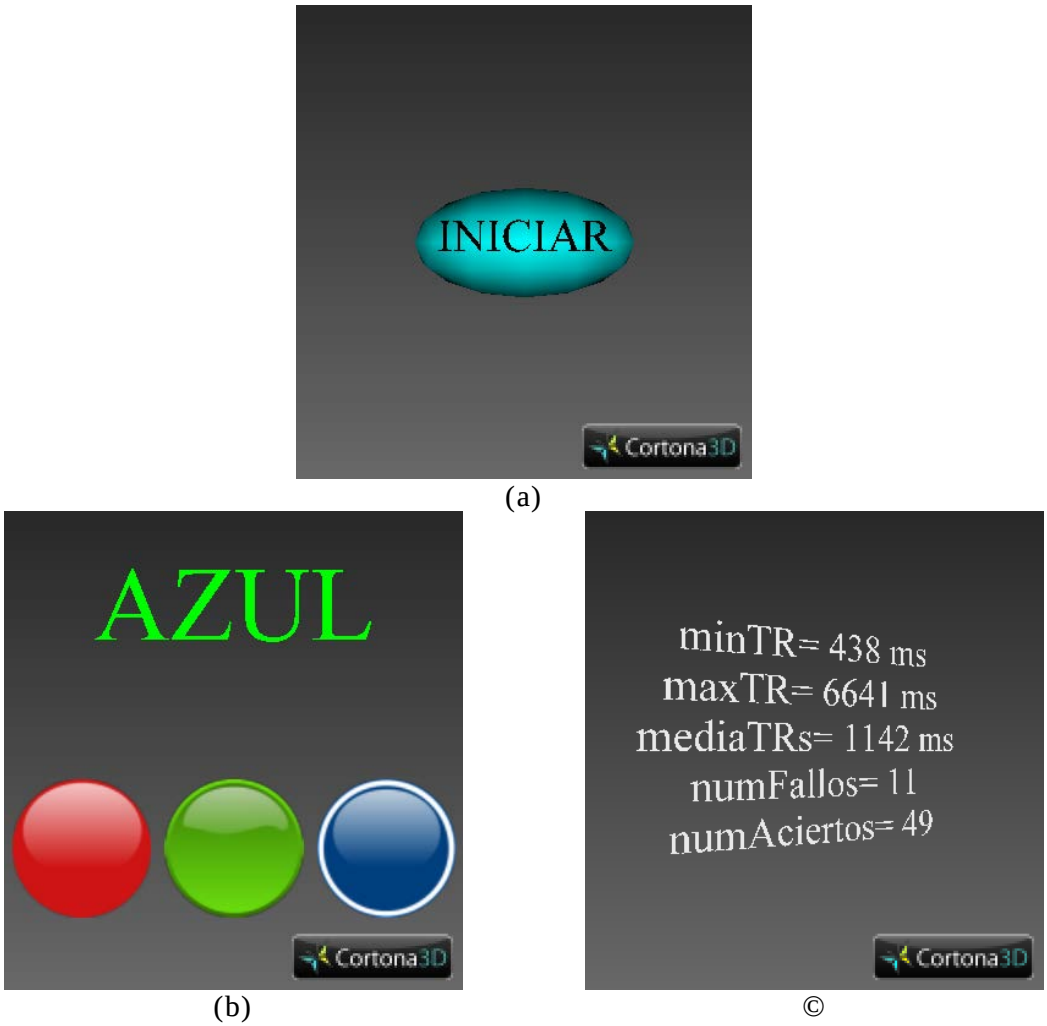


Figura 4-16: Capturas de pantalla de la aplicación implementada con *VRML* y *JavaScript* para realizar la prueba de *Stroop*[25]. (a) Pantalla inicial: construcción de estímulos y asignación de aleatoriedad. (b) Pantalla de emisión de estímulos y captura de respuestas: se almacenan los instantes de emisión y pulsación de los botones (c) Pantalla de resultados: se calculan el mínimo, el máximo, la media de los tiempos de reacción.

Finalizada la parte de creación de los objetos en la escena pasamos a describir en el siguiente apartado los pasos para configurar el experimento: generación de estímulos posibles, aleatoriedad, captura de respuestas, etc.

3.3.2.2 Control con JavaScript

Para el control de la aplicación nos hemos servido de un nodo de tipo *Script* en el que hemos configurado el experimento con lenguaje *JavaScript*. El funcionamiento, en líneas generales, que tiene este nodo es que se puede crear una interfaz de entrada y salida al igual que describimos para el nodo *Proto*. Además el nodo *Script* permite

[25] Actualmente la aplicación que permite realizar la prueba de *Stroop* implementada con *VRML* y *JavaScript* se encuentra disponible en el siguiente enlace: <http://giotitoa.esy.es/pfc/stroop/vrml/stroop.html>. [Julio 2015]

otras posibilidades ya que se pueden declarar, entre otros, campos *field* no accesibles desde fuera que sirven a modo de variables globales, así como declarar campos de tipo *SFNode* en los que podemos hacer referencia a cualquier nodo creado previamente en el fichero y de esta forma manipular sus propiedades. Los eventos en este nodo están acompañados de una marca de tiempo *timeStamp* por lo que se pueden recibir eventos de los sensores y en consecuencia ejecutar funciones específicas que realicen la captura de los tiempos.

La configuración de la prueba con *JavaScript* ha permitido observar la similitud de las capacidades de éste con *ActionScript* descrito en apartados previos de forma que las diferentes tareas que se llevan a cabo en la prueba se implementan de una forma casi idéntica. A continuación mencionaremos brevemente las adaptaciones realizadas.

Construcción de estímulos: disponemos de la función *initialize* que permite inicializar variables, ya que ésta se ejecuta en el momento en el que se carga el fichero *VRML*. La manera de construir los estímulos color-respuesta es la misma que en *ActionScript*, es decir, mediante el uso de un *Array* compuesto de un *String* y un *SFColor* (estructura de tres valores en coma flotante). Se genera una lista de todos los estímulos posibles.

Aleatoriedad: la generación de la aleatoriedad tiene los mismos principios descritos en "Control con ActionScript", es decir, empleamos el método *Math.Random* para crear una lista con los estímulos en posiciones aleatorias.

Emisión de estímulos: el procedimiento es similar. Tomamos estímulos de la lista y los presentamos. La diferencia está en la menara de insertar los valores en el nodo *Text* para mostrar el color y la palabra. Esto se realiza, como habíamos comentado, gracias a un campo de la interfaz del nodo *Script* que permite hacer referencia al nodo *Text* donde queremos introducir la palabra y el color. Hablamos del campo *SFNode*. En cuanto a la captura del instante de emisión y recepción debemos decir que se realiza transmitiendo las marcas de tiempo *timeStamp* que llegan con los eventos de los botones. Una vez recibida la respuesta, establecemos el tiempo de la nueva emisión como el tiempo de la última recepción. Con ello, podemos ir calculando el tiempo de reacción como la diferencia entre el instante de emisión y el instante de recepción. Además con este funcionamiento al crear un objeto de la clase *Date* podemos pasarle el *timeStamp*, el cual está expresado en **segundos** que han pasado desde el 1 de Enero de 1970, multiplicado por 1000 para así obtener las mediciones en milisegundos al usar el método *getTime* de esta clase.

Captura de las reacciones: el funcionamiento es similar. Se captura el instante, se calcula el tiempo de reacción y se comprueba si se trata de un acierto para lo cual se deben comparar los colores componente a componente (r, g, b) del botón pulsado y del estímulo. Los resultados se almacenan en una lista y finalmente se llama a la función de emisión para emitir el siguiente estímulo.

Eventos de los botones: declarando un evento de entrada de tipo *SFBool* en la interfaz del nodo *Script* podemos recibir en éste la activación del sensor de tacto *TouchSensor* del botón y en consecuencia llamar a la función de captura pasándole el color y la marca de tiempo.

```
function botonRojoPulsado(e, ts){
 if(e == true)
  capturarReaccion(new SFColor(1,0,0), ts); //rojo
}
```

Tabla 4-3: Ejemplo de función en *JavaScript* que atiende la llegada de un evento *SFBool* proveniente de uno de los sensores. La función recibe como parámetros de entrada el evento "e" y la marca de tiempo "ts". Un evento de entrada *SFBool* declarado en la interfaz del nodo *Script* en *VRML/X3D* puede ser *true* o *false*.

Resultados: una vez finaliza la lista de estímulos, se calculan los resultados y se insertan los resultados en formato texto en el campo correspondiente de la interfaz del nodo *Script*. Una vez construido el texto se envía hacia fuera del *Proto* (que contiene toda la prueba de *Stroop*) un evento *SFBool* y un evento *MFString*, en los que informamos del fin de la prueba y transmitimos los resultados calculados en formato texto ya que entre nodos *VRML* sólo podemos usar ciertos tipos entre los cuales no existen *Array*. Éstos serán capturados por nodos externos y se mostrarán en una escena final.

Control de escenas: para el control de escenas podemos usar un nodo de tipo *Switch* que permite agrupar varios nodos hijos y mostrar uno a la vez. Podemos colocar nuestro *Proto* de la prueba de *Stroop* como uno de sus hijos, y colocar otros nodos que sirvan para generar las pantallas de inicio y resultados. De forma que mediante código *JavaScript* podemos modificar el hijo mostrado por el nodo *Switch* y así generar "saltos" entre escenas.

Como vemos utilizar la tecnología de realidad virtual para construir un videojuego puede ser en algunos apartados rápida teniendo los conocimientos de las propiedades de algunos nodos. Tras esta toma de contacto pasaremos a describir las ventajas e inconvenientes.

3.3.3 Ventajas e inconvenientes

A continuación mencionamos las ventajas:

- *VRML/X3D* son herramientas de código abierto que están libres del pago de regalías por su uso, lo cual puede ser interesante para crear aplicaciones grandes no comerciales para su uso a través de internet.
- *X3D* (tecnología sucesora de *VRML*) es un estándar que aún permanece en mantenimiento y evolución y que permite añadir otro tipo de servicios web como la conexión a bases de datos.

- Existen bibliotecas que permiten su integración en otras tecnologías como las aplicaciones *Java*.
- Existen varios *plug-in* visores 3D de ficheros *VRML* para navegadores así como algunos destinados a dar soporte a la visualización de dichos ficheros en dispositivos móviles, entre ellos los dos más destacables *BSContact* y *Cortona3D*.
- La programación de mundos virtuales basada en el agrupamiento jerárquico de nodos eleva el grado de abstracción.
- Dotar de interactividad a algunos de los objetos de la escena se puede hacer rápidamente con nodos de tipo sensor.
- Puesto que es el visor3D el que se encarga de comunicarse con la tarjeta gráfica, al redimensionar el tamaño de la ventana del *plug-in*, los elementos 3D que está enfocando la cámara se redimensiona correctamente. En nuestro caso, al definir los botones como formas planas de 2 dimensiones dentro de un mundo 3D, el redimensionado es igualmente correcto.
- Se pueden programar comportamientos complejos y ver los resultados con gráficos 3D empleando lenguajes de apoyo como *Java* o *JavaScript*.
- Al existir *plug-in* soportados por la mayoría de navegadores se incrementa el número de usuarios potenciales.
- Algunos de las compañías que desarrollan los *plug-in* ofrecen *API* de desarrollo para incorporar el *plug-in* directamente como parte de otras aplicaciones.

A continuación mencionamos los inconvenientes:

- *VRML* está enfocado a mundos virtuales y dependemos mucho del *plug-in* que interpreta estos ficheros. Al existir varios cada uno puede presentar extensiones que no son interpretadas por el resto.
- Para realizar una escena estática como en nuestro caso, es necesario acudir a la documentación del el *plug-in* con el que vamos a querer que se visualice. Ya que por defecto, los *plug-in* que interpretan estos ficheros y "rasterizan" los gráficos 3D, ofrecen opciones como la navegación con el ratón, botones para enfocar diferentes partes del escenario, botones para reposicionar la cámara, despliegue de un menú de opciones del *plug-in* con el botón derecho, etc.
- Al pedirle al usuario que se instale un *plug-in* perdemos un gran porcentaje de potenciales usuarios que no desean la instalación.
- El contenido del fichero *.wrl* cargado por el *plug-in* es visible lo cual limita el uso para aplicaciones comerciales.
- *VRML* por si solo ofrece soporte para implementar mundos virtuales sencillos, sin embargo, si queremos crear comportamientos más complejos tenemos que recurrir a *Java* o *JavaScript*, lo cual incrementa el número de conocimientos necesarios.

3.4 Unity®

Como se explica en la página web oficial "*Unity®* es una plataforma de desarrollo flexible y poderosa para crear juegos y experiencias interactivos 3D y 2D multiplataforma". Actualmente se encuentra en la versión cinco la cual presenta mejoras destacables como la integración de *Canvas* para crear interfaces de usuario de forma manual, así como nuevas opciones en el editor de escenas que permiten el re-escalado de objetos de una forma más simple. Ésta se ofrece en dos ediciones: para uso personal y para uso profesional[26].

Se trata de un motor de videojuegos que presenta un entorno de desarrollo con herramientas de edición de escenas y de ayuda a la programación de las aplicaciones. El motor gráfico utiliza *Direct3D* (en *Windows*), *OpenGL* (en *Mac* y *Linux*), *OpenGL ES* (en *Android* y *iOS*), e interfaces propietarias (*Wii*), mientras que el motor de audio emplea *FMOD* para generar sonido espacial 3D.

A pesar de estar dirigido principalmente a la creación de videojuegos su uso se encuentra presente en diferentes áreas ya que la opción de exportar las aplicaciones a diferentes plataformas, entre ellas las de dispositivos móviles cuyo uso sigue incrementándose, atrae a muchos desarrolladores que ven el tiempo de desarrollo reducido. Además les permite mantener el código de las aplicaciones en un solo entorno desde el que se puede ir añadiendo pequeñas adaptaciones para el uso de características del hardware de los dispositivos móviles como el acelerómetro, la entrada táctil, etc.

Este motor se integra con *Monodevelop* del proyecto *Mono*[27] para dar soporte a la programación basada, principalmente, en *scripting*, permitiendo al desarrollador elegir el lenguaje de implementación entre *JavaScript*, *Boo*, y *C#*. Con lo que se amplía el número de potenciales programadores.

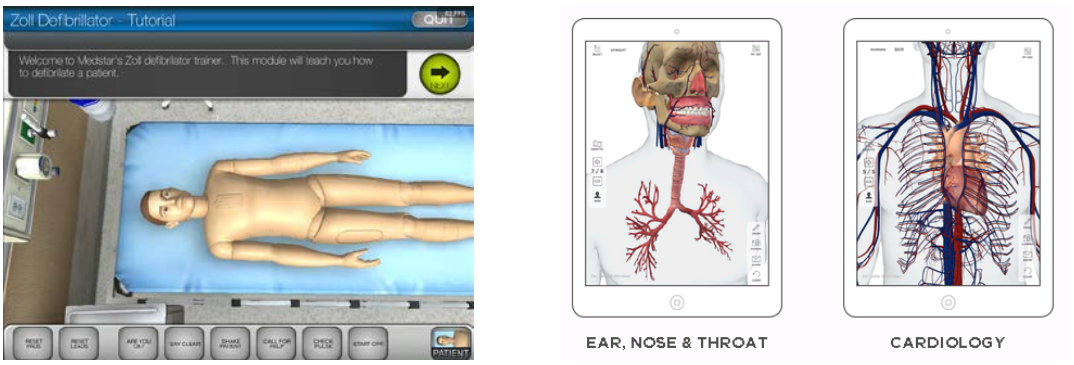
Otra de las características destacables para reducir el tiempo de desarrollo de las aplicaciones con esta herramienta es la existencia de una tienda de *Assets*. Esto significa que podemos encontrar animaciones, modelos 3D, *script*, texturas, materiales, bibliotecas de funciones, etc., listos para ser incorporados como parte de nuestros proyectos en *Unity®*. El funcionamiento de esta tienda es mediante la compra digital.

Si bien el uso de *Unity®* se encuentra en videojuegos de entrenamiento cabe destacar que su uso es igualmente elevado en otro tipo de aplicaciones, muchas de las cuales

[26] La versión profesional incluye algunas herramientas que la versión personal no posee por ejemplo O *render* a textura, determinación de cara oculta, iluminación global y efectos de post-procesamiento. Para más detalles se puede consultar la web oficial: < http://unity3d.com/es/5>. [Consultado: Julio 2015]

[27] *Monodevelop* es un entorno de desarrollo integrado libre y gratuito de *.NET framework* que presenta facilidades como el manejo de clases, verificación de sintaxis, depurador integrado, etc. Podemos consultar más detalles en su web oficial: <http://www.monodevelop.com/>. [Consultado: Julio 2015]

están enfocas a presentar entornos de simulación. En la Figura 4-17 podemos observar el aspecto de algunas aplicaciones creadas con *Unity*®.



(a) (b)

Figura 4-17: Ejemplos de aplicaciones que han sido creadas empleando *Unity*®. (a)*Virtual Zoll Defibrillator Trainer*: es un programa de entrenamiento diseñado para enseñar a enfermeros y enfermeras cómo deben reanimar a un paciente durante una emergencia cardíaca. (b) *CoherentRx anatomy*: son aplicaciones desarrolladas por médicos para médicos. Cada aplicación presenta la visualización de anatomía específica que agiliza las consultas médicas en varias especialidades.

Imágenes tomadas de los sitios web oficiales: (a) <http://sitel.org/digital-media-services/>, (b) <https://www.coherentrx.com/>. [Consultado: Julio 2015]

3.4.1 Editor

La interfaz gráfica del editor de *Unity*® nos presenta una serie de herramientas separadas en paneles o vistas, los cuales pueden redimensionarse para editar escenas de forma más cómoda. Incluso se ofrece configuraciones de la interfaz en las que es posible observar al mismo tiempo la pre-visualización de la escena en una ventana y explorar áreas de la escena en otra, lo cual sirve de ayuda para observar el comportamiento de aquellos objetos que no están siendo enfocados por la cámara principal de la escena. En la Figura 4-18 podemos observar un ejemplo de esta interfaz.

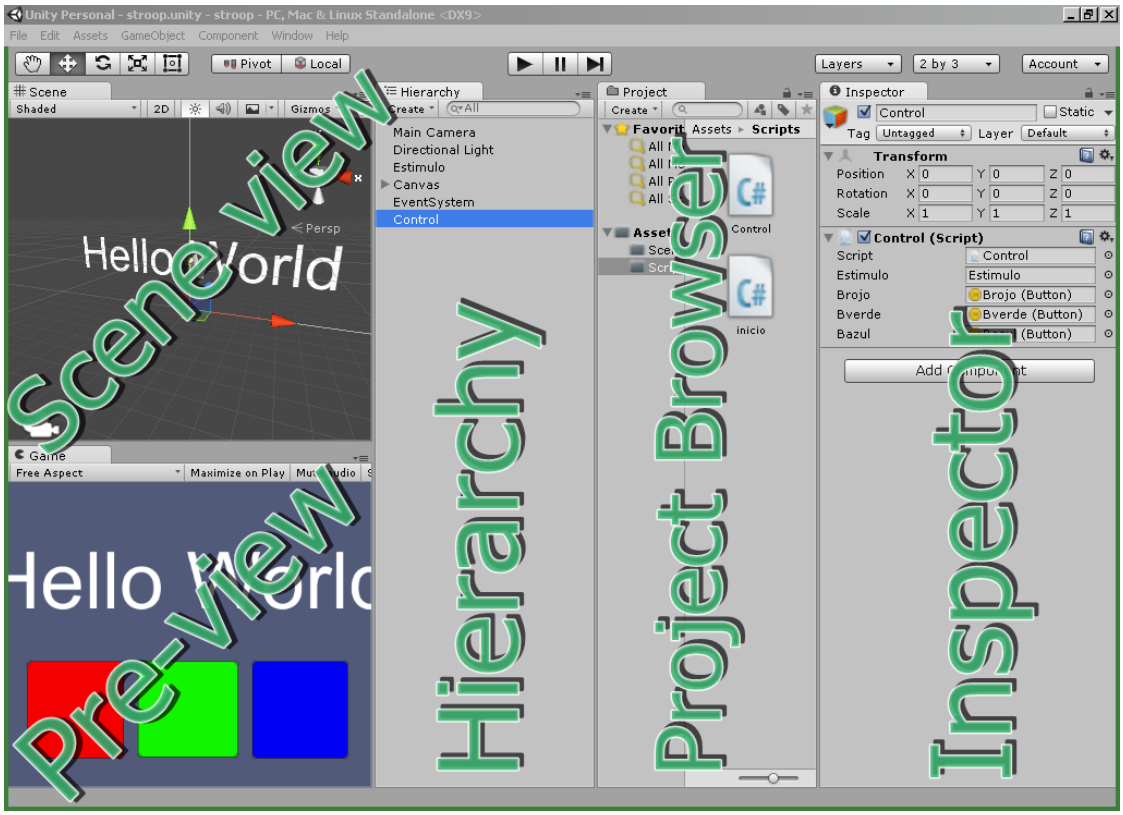


Figura 4-18: Captura de pantalla que muestra un ejemplo de configuración de los paneles de la interfaz de *Unity®*. *Scene view*, *Preview*, *Hierarchy*, *Project Browser*, e *Inspector*.

La interfaz está conformada, principalmente, por los siguientes paneles: *Project Browser*, *Hierarchy*, *Inspector*, y *Scene View*.

Project Browser es el panel navegador de proyecto que permite organizar todos los *Assets*[28] o activos mediante carpetas. Los resultados de la organización en esta área se verán reflejados en el directorio correspondiente en el que se encuentre creado el proyecto en el disco duro local. Incluso es posible añadir nuevos elementos tanto desde el editor, por ejemplo arrastrando y soltando una imagen, como incluyéndolo directamente en el directorio. Este panel también permite abrir los *script* con doble clic, tras lo cual se abrirá el *Monodevelop*.

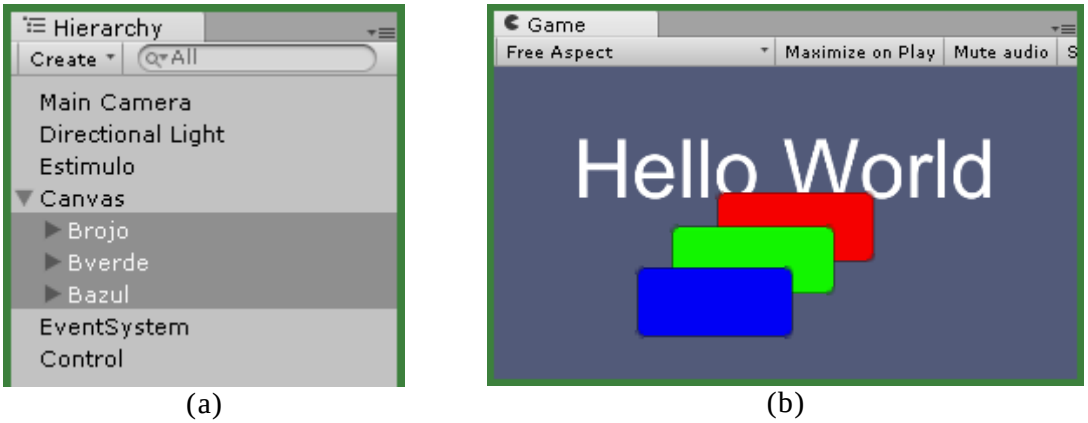
Scene View es el panel en el que se visualiza la escena, y posee una serie de opciones seleccionables para hacer que la edición de los objetos en la escena sea más cómoda. Por ejemplo se puede marcar la visualización en alámbrico, desactivar la visualización de luces, activar la visualización de niebla, etc. Es interesante mencionar que es posible hacer que la cámara de este panel enfoque un determinado que no vemos en la escena a primera vista seleccionándolo en el panel *Hierarchy*. Este panel además permite la navegación por la escena pudiendo rotar el escenario, cambiar a una vista isométrica, mostrar rápidamente la visualización de alzada, lateral o inferior, entre otras opciones.

[28] Los *Assets* o activos en *Unity®* pueden ser objetos de tipo *Scene*, *Game Object*, *Camera*, *Ligth*, *etc.* Es decircasi todos.

Inspector es el panel en el que se presentan todas las propiedades y las componentes de un objeto seleccionado. En éste podemos introducir los valores de parámetros como las dimensiones, la posición, etc., y añadir componentes de diferentes tipos a un *Game Object* seleccionado en panel *Hierarchy*. Por ejemplo podemos incluir efectos de leyes de partículas, cuerpo rígido para simular los efectos de la gravedad, ficheros audio, etc. Entre estos componentes existe uno importante que es el componente *Script* que será donde introduciremos las instrucciones para controlar el comportamiento de un objeto o la comunicación con otros objetos de la escena. Es el más importante de los paneles pues en todo momento podemos observar los cambios en los valores de los parámetros del objeto seleccionado así como ajustarlos para observar resultados en la escena, y sobre todo porque será aquí donde visualizaremos las variables públicas declaradas en los *script* adjuntos. Incluso podemos asignar a dichas variables públicas objetos de la escena arrastrándolos y soltándolos en dichos campos.

Hierarchy es el panel que permite organizar los objetos mediante jerarquía padre-hijo de forma que la ejecución se realiza de arriba a abajo según la organización dada a los objetos en este panel. Por ejemplo podemos crear un *Game Object* vacío al que denominamos, por ejemplo, *InterfazGrafica* y en el que podemos incluir los objetos de la clase *UI* que la versión 5 incorpora, como los objetos *Canvas*, botones, *slider*, *toggle*, etc.

El caso del *Canvas* nos va a servir para explicar el orden de ejecución según la jerarquía: al incluir objetos de tipo *UI* se crea un *Canvas* dentro del cual podemos organizar, por un lado el orden de renderizado, y por otro el orden de atención de los eventos de los elementos *UI* contenidos en el *Canvas*. Para aclarar este concepto podemos observar la Figura 4-19 donde se muestra un ejemplo del renderizado jerárquico y el orden de prioridad en el que se atenderán los eventos.



(a) (b)

Figura 4-19: Capturas de pantalla que muestran un ejemplo de los resultados de la organización de los objetos en el panel *Hierarchy* de *Unity®*. (a)Prioridad de renderizado de los botones en este orden: rojo, verde y azul. (b)Pantalla de pre-visualización de la escena: el último en "renderizarse" es el azul por eso "solapa" a los otros botones.

Otro punto destacable de la nueva *API* es que se pueden emplear anclas para que los objetos se redimensionen automáticamente en función del ancho y alto de la pantalla del dispositivo en el que se ejecute la aplicación, permitiendo a los desarrolladores realizar un diseño adaptativo que no requiere de esfuerzo adicional para que la interfaz de usuario se adapte a varios dispositivos. Dependiendo de la organización de la anclas y del padre al que pertenezcan los elementos *UI*, éstos se harán más estrechos o más altos, ante lo cual debemos tener en cuenta que en algunos casos el contenido más interno como texto de los botones también debe ser ajustado ya que por defecto tiene un tamaño de fuente fijo. La Figura 4-20 nos muestra un ejemplo de ello.

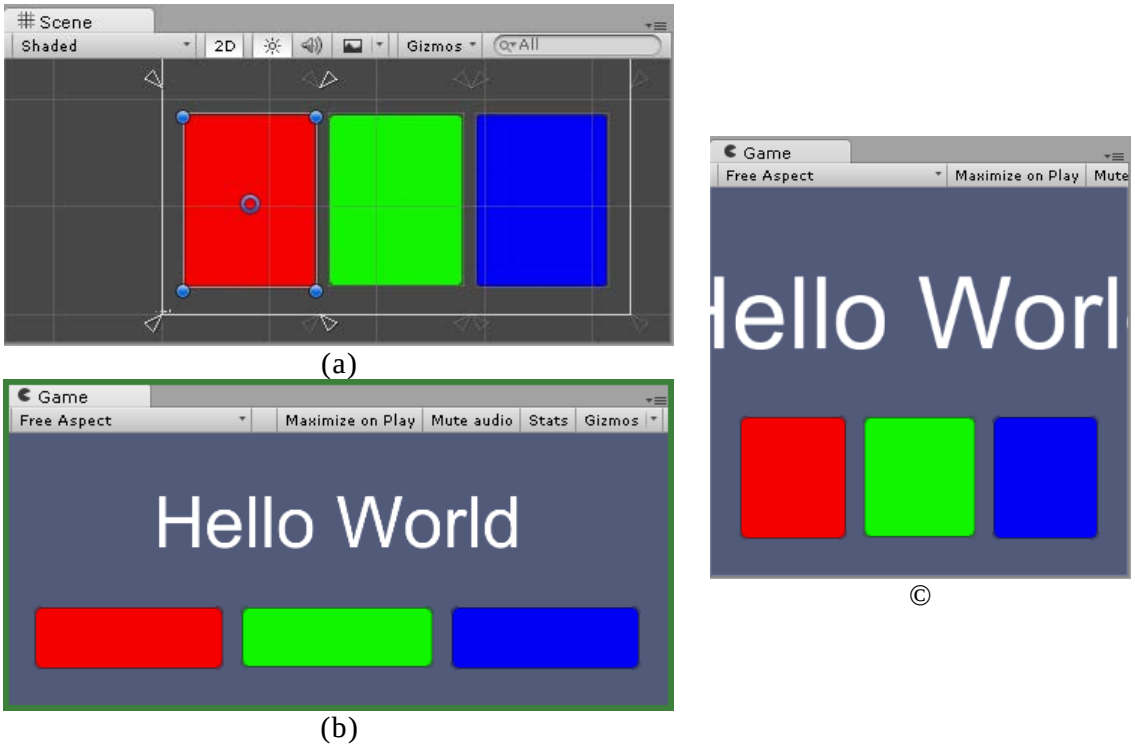


Figura 4-20: Capturas de pantalla que muestran un ejemplo del redimensionado de los objetos dentro de un *Canvas* gracias a la configuración de las anclas en *Unity*® versión 5. (a) Distribución de las anclas (triángulos) del botón rojo. (b) Pre-visualización con pantalla ancha. (c) Pre-visualización con pantalla alta.

Finalmente es importante resaltar que *Unity*® permite pre-visualizar y modificar escenas durante la pre-visualización sin embargo las acciones que se realicen, como por ejemplo, la modificación del tamaño o la posición de un objeto, no serán tenidas en cuenta. Para ello se debe detener la pre-visualización y modificar los valores.

3.4.2 Scripting

En cuanto a la programación gracias a la integración con *Monodeveop* la depuración de los *script* se agiliza. Sin embargo debemos mencionar algunos conceptos fundamentales para entender es el flujo de ejecución que sigue el motor de *Unity®*. En nuestro caso vamos a desatacar algunas funciones principales que responden a eventos de ejecución: *Start, Update, LateUpdate, FixedUpdate*, y *OnGui*.

Las funciones *Start* de todos los *script* de un mismo *Game Object* se ejecutan (interpretan) en paralelo una sola vez al principio del flujo de ejecución. Es dentro de esta función donde se deben inicializar las variables a ser utilizadas posteriormente en la función *Update*.

Las funciones *Update* de todos los *script* se ejecutan en cada fotograma. Lo cual significa que la frecuencia con la que se ejecutan depende del *Time step* fijado para el proyecto (modificable en las preferencias del proyecto). En nuestro caso, en el que queremos medir el tiempo de reacción, podríamos pensar que esta dependencia traerá problemas, sin embargo veremos más adelante que existe una clase que permite medir el tiempo con mayor precisión. Es en esta función donde se programan casi todos los comportamientos de los objetos.

Las funciones *LateUpdate* de todos los *script* se ejecutan una vez que todas las funciones *Update* han terminado de llamarse. Esto sirve para organizar el flujo de ejecución de nuestras aplicaciones y optimizar el número de llamadas a funciones específicas. Por ejemplo en los casos en los que queremos programar el comportamiento de cámaras seguidoras, es decir, cámaras que en función de los movimientos y desplazamientos del personaje enfocan adecuadamente la escena.

Las funciones *FixedUpdate* son llamadas con un *framerate* (tasa de fotogramas por segundo) fijo para las clases que heredan de *MonoBehaviour* (clase de la que toman herencia todos los *Game Objec*t en *Unity®*). Esta función debe ser utilizada en lugar de *Update* cuando se realiza tratamientos sobre cuerpos rígidos *RigidBody*. Por ejemplo, cuando se añade una fuerza a un *RigidBody*, tenemos que aplicar dicha fuerza a una velocidad constante dentro de *FixedUpdate* en lugar de cada fotograma dentro de *Update*. Además podemos mencionar que es posible definir el *Time step* con el que se llama a *FixedUpdate* desde las preferencias del proyecto (*Edit->ProjectSettings->Time->FixedTimestep*). El valor por defecto es 0.02.

Las funciones *OnGUI* se ejecutan (son llamadas) varias veces por fotograma y gestionan el dibujado de botones y otros elementos de la clase *GUI* (diferente de la clase *UI*). Estos elementos de interfaz gráfica pertenecen a las *API* de las versiones 4.6 en adelante. Éstos presentan facilidades para ser manipulados mediante código. Por ejemplo con un bucle podemos crear un *ScrolView* de botones en vertical, ir cambiando sus colores en función de la interacción con el usuario, esconderlos, etc. Tanto los

elementos de la clase *UI* como los de la clase *GUI* tienen sus puntos a favor. Los de la clase *UI* su fácil edición manual, los de la clase *GUI* la creación de interfaces complejas interactivas, entre otras características. Además al haber sido esta última la que lleva más historia de uso podemos encontrar más tutoriales que nos pueden ayudar a solventar problemas.

Además podemos mencionar algunos detalles referentes a la programación que reducen el tiempo de desarrollo entre los que destacamos el uso de variables de tipo *public* que son visibles en el inspector. Esto permite asignar de forma manual (arrastrar y soltar) objetos desde el panel *Hierarchy* hasta una de las casillas que representa a la variable declarada visible en el panel *Inspector*. Por ejemplo, declarando una variable de tipo *Button* en las primeras líneas del script, podremos luego asignarle el botón correspondiente como se observa en la Figura 4-21.

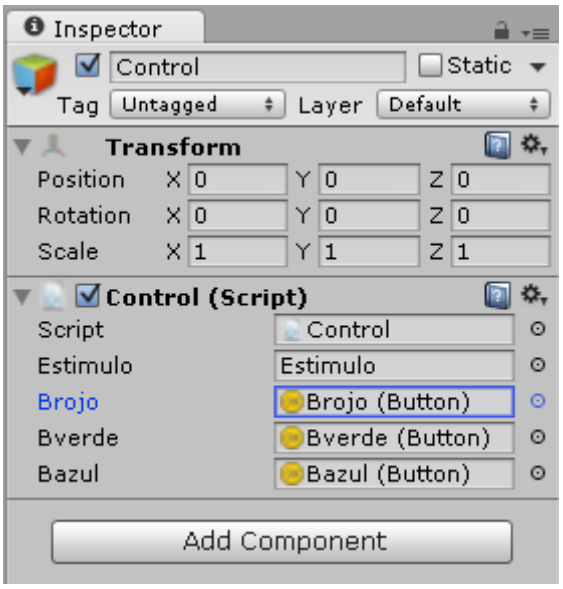


Figura 4-21: Captura de pantalla que muestra un ejemplo de asignación de un objeto *Button* a una variable publica de un *script* en el panel *Inspector* de *Unity®*.

Lo mismo podemos realizar para manipular un *Game Object* mediante código. Es decir, una vez dentro del script podremos acceder a las propiedades y métodos del *Game Object* usando la variable pública a la que se adjunta. En nuestro caso en la variable pública *Estimulo* hemos adjuntado un *Game Object* que es una primitiva *3D Text* como se observa en la Figura 4-21.

Para el control de errores podemos emplear la clase *Debug* que permite emitir por consola un registro de errores empleando el método *Log*. Para ello es necesario que el parámetro que se le pasa a *Log* sea un *String* por lo que el método *toString*, que casi todas las clases poseen en la *API* de *Unity®*, servirá para ello. El método *Log* junto con la consola de salida del entorno sirven para localizar rápidamente líneas dentro del *script*, ya que con solo hacer clic en el texto de salida accedemos de forma automática a la línea específica del *script* en *Monodevelop*.

3.4.3 Implementación

La implementación de la prueba de *Stroop* se ha realizado una parte con el panel *Scene View* en el que hemos incorporado primitivas que *Unity®* nos ofrece como el *3D Text* y el *Canvas*, los cuales servirán para emitir los estímulos y capturar las respuestas, y por otra empleando *Monodevelop* para la depuración de los *script*.

3.4.3.1 Escenario

Para configurar el escenario se ha usado el inspector de propiedades y el editor de escena.

En el área del estímulo hemos incluido un *3D Text* al que le hemos modificado las propiedades como el *Anchor* o anclaje (en definitiva el centro de gravedad) del objeto estableciéndolo a *Middle center*. Con esto conseguimos que las trasformaciones (rotaciones, traslaciones) que se le apliquen se realicen en función de este punto.

Para que todas las cadenas de caracteres de la lista de estímulos tiene se visualicen sin desplazamientos hemos establecido el parámetro *Alingment* con valor *Center*. Mientras que para reducir el *aliasing* de las letras hemos incrementado el tamaño de la fuente hasta encontrar que su visualización sea correcta y luego reducido el *Character size* hasta que el tamaño sea el que queremos. La Figura 4-22 nos muestra un ejemplo de esta solución.

(a) (b)

Figura 4-22: Capturas de pantalla que muestran un ejemplo de reducción del *aliasing* en un *3D Text* modificando el *Character Size* de la fuente en *Unity®*. (a) *3D Text* con fuente con tamaño 12 y *Character size* con tamaño 1. (b) *3D Text* con fuente con tamaño 100 y *Character size* con tamaño 0.1.

En el área de la captura de las respuestas del participante, hemos empleado objetos de tipo *UI*, es decir tres *Game Object Button* para construir los botones: rojo, verde y azul. El modo de establecer los colores ha sido igualmente utilizando el inspector de propiedades en el que hemos establecido el color en el componente *Image* que cada *Button* tiene.

Una vez configurado el escenario pasamos a editar el *script* de control de la aplicación. Para ello definimos variables públicas de tipo *Button* para asignar los botones desde el inspector, y una variable publica para el *3D Text* donde presentaremos los estímulos. La declaración de variables se debe realizar fuera de las funciones *Start* y *Update* para que su ámbito esté presente en todas las funciones del script. Las variables que se declaren dentro de la función *Start* o *Update* sólo estarán activas dentro de dichas funciones.

Como vemos la creación de escenas en *Unity®* es muy sencilla gracias a que nos permite incluir primitivas básicas, las cuales en nuestro caso, bastan para crear un escenario donde emitir estímulos y capturar reacciones. Sin embargo si se quisiera incorporar modelos 3D provenientes de otras aplicaciones como *Blender* o *3DMax*, *Unity®* ofrece soporte para cargar ficheros provenientes de estas herramientas de modelado.

Finalmente el aspecto de la aplicación implementada lo podemos ver en la Figura 4-23 que muestra varias pantallas.

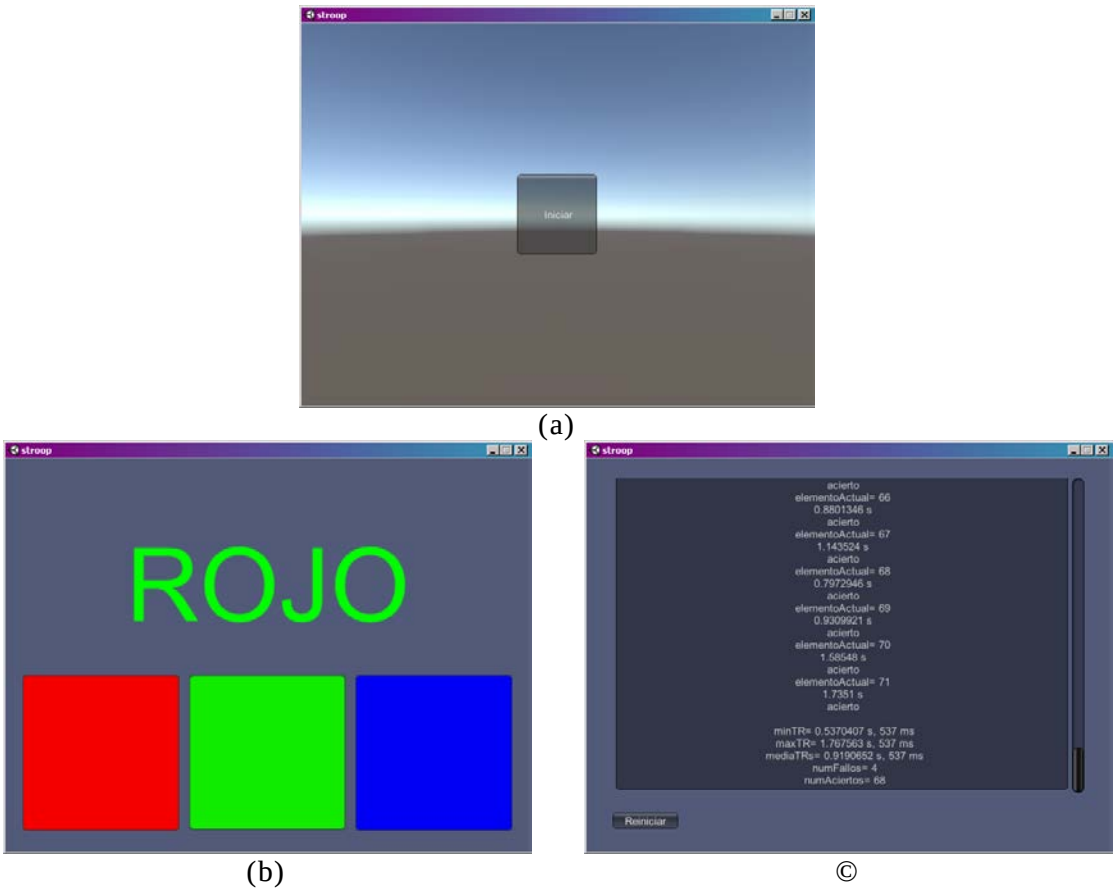


Figura 4-23: Capturas de pantalla de la aplicación implementada con *Unity®* y *C#* para realizar la prueba de *Stroop*[29]. (a) Pantalla inicial: construcción de estímulos y asignación de aleatoriedad. (b) Pantalla de emisión de estímulos y captura de respuestas: se almacenan los instantes de emisión y pulsación de los botones (c) Pantalla de resultados: se calculan el mínimo, el máximo, la media de los tiempos de reacción.

[29] Actualmente la aplicación que permite realizar la prueba de *Stroop* implementada con *Unity®* y *C#* se encuentra disponible en el siguiente enlace: <http://giotitoa.esy.es/pfc/stroop/unity/stroop.html>. [Julio 2015]

3.4.3.2 Control con C#

Para configurar la prueba de *Stroop* hemos creado un *Game Object* vacío al que denominamos *Control* en el cual hemos añadido el *script* que se gestiona la configuración de los estímulos, la captura de las respuestas, el cálculo de los resultados, etc. Hemos hecho uso de variables públicas en las que adjuntamos manualmente los botones y el *3D Text*. Con éstas hemos podido aplicar el método *getComponent* que todos los *Game Object* poseen, para buscar determinadas componentes. Por ejemplo para encontrar la malla 3D del objeto *3D Text* y así poder cambiar su contenido:

- `TextMesh tm = Estímulo.getComponent<TextMesh> ();`

A continuación vamos a mencionar algunos detalles de cada una de las tareas internas que realiza la aplicación para gestionar la prueba.

Construcción de estímulos: *C#* nos ofrece la posibilidad de usar listas dinámicas[30] lo cual nos va a facilitar la construcción de listas de estímulos color-palabra. Esto es así gracias a que la clase *List<T>* está creada como una plantilla pudiendo almacenar elementos de cualquier tipo el cual se le pasa como argumento cuando se crea una nueva lista dinámica (parámetro *T*). Además al ser una lista dinámica ofrece métodos como *Add* para añadir un elemento, *Sort* para ordenarlos, *Find* para buscar un objeto dentro de la lista, etc.

Es en la construcción del tipo *T* que define los parámetros de nuestro estímulo (un color y un palabra) donde podemos construir una clase que contenga dichos parámetros, es decir un dato de tipo *String* y un dato de tipo *Color* (tipo de dato disponible en *Unity®* para manipular colores *rgb*).

Aleatoriedad: la manera de generar la lista aleatoria en esta plataforma ha sido empleando el método *Range* de la clase *Random* que nos permite obtener valores enteros entre un valor mínimo y valor máximo (enteros) que se le pasa como parámetro. Con este método hemos generado la aleatoriedad copiando los elementos de la lista original a otra. Es en el proceso de creación de la lista de reacciones donde es de especial interés la creación de una estructura que agrupe tres valores: tiempo de reacción, color pulsado y acierto. Esta clase puede ser pasada como parámetro al constructor de lista dinámica para construir una que permita encolar este tipo de elementos. Lo cual nos servirá posteriormente para consultar y calcular los resultados.

En ocasiones puede ser interesante tener una función que nos genere la aleatoriedad en función de la presencia de una semilla y en otras no. Sin embargo en *C#* podemos sobrecargar una función para que admita parámetros opcionales, de tal forma que la

[30] Podemos encontrar varios ejemplos comparativos de listas dinámicas en *JavaScript* y *C#* en el siguiente enlace: <http://wiki.unity3d.com/index.php/Choosing_the_right_collection_type>. [Consultado: Julio 2015]

misma función puede llamarse con o sin los parámetros opcionales. Esto se consigue en el momento de la definición de la función donde establecemos los parámetros de entrada opcionales tras los que no lo son. El siguiente ejemplo es un amuestra de ello:

- `public int[] randomIndices(int nIndicesADevolver, int rangoDeCeroAN, int semilla=-1){…};`

Eventos de los botones: en *Unity®* podemos hacer uso del evento *onClick* y añadirle a éste un oyente no persistente o *listener* que realice una llamada a una determinada función. Es decir, podemos añadir un *listener* que llame a nuestra función *capturarReaccion* la cual iniciará todo el proceso de captura al igual que explicamos en "Control con *ActionScript*" y "Control con *JavaScript*". Sin embargo cabe mencionar que esta adición del *listener* se realiza una sola vez en la función *Start* y para poder pasar el color pulsado es necesario delegar la llamada, es decir, invocar al método *delegate* y dentro de éste llamar a la función *capturarReacción* pasando el valor de color como parámetro. Esto se debe hacer así dado que el *listener* solo recibe parámetros de tipo *call-back*, es decir nombres de función, y *delegate* nos soluciona el conflicto.

Emisión de estímulo: el primer punto a resolver es la manera de cambiar las propiedades del *3D Text*. Para ello podemos hacer uso de la clase *Color*, que ya hemos mencionado, para asignar valores a la propiedad *color* del objeto *3D Text* colocado en la parte superior de nuestra escena. Además esta clase facilita la lectura de código ya que presenta atributos como, por ejemplo, *Color.red* que establece los valores *rgb* para definir un color rojo. Puesto que *Color* es un grupo de 3 valores en coma flotante podemos usar el operador de comparación == que está sobre cargado para comparar las componentes de los colores. En cuanto a la cadena de caracteres del estímulo podemos manipular la propiedad *text* del *3D Text* para asignar allí la palabra a mostrar.

Captura de las reacciones: para la medición de los tiempos de reacción hemos hecho uso del método *realtimeSinceStartup* de la clase *Time* el cual nos da los segundos que han pasado desde que la aplicación empezó a ejecutarse en la máquina. Es decir se trata de una medida local que presenta mayor precisión que las obtenidas con la función *getTime* usada en *ActionScript* y *JavaScript*, ya que, como habíamos mencionado, dicha función retorna el número de milisegundos pasados desde el 1 de Enero de 1970. Es decir es una medida relativa. *realtimeSinceStartup* incluso nos ofrece ventajas sobre las opciones de modificación del *Time step* de las preferencias del proyecto, puesto que las medidas hechas con este método no se ven afectadas cuando el *Time step* cambia, lo cual suele hacerse para efectos de aceleración y reducción del número de fotogramas por segundo. Las medidas que nos da este método están expresadas en segundos.

Resultados: para poder calcular el mínimo, el máximo, la media de los tiempos de reacción y las tasa de aciertos, nos servimos igualmente, como ya mencionamos, de una lista dinámica cuyos elementos son de tipo *reaccion*, una clase creada para reunir tres atributos: color pulsado, tiempo de reacción, y acierto. Las listas dinámicas de *C#* nos

permiten emplear métodos propios para ordenar los elementos como *Sort*, al cual podemos pasarle un *call-back*. Por ello creamos una lista dinámica cuyos elementos son de tipo *reaccion* y la denominamos *reacciones*. Finalmente para usar el método de ordenación *Sort* en ésta, delegamos el *call-back* a una función que va ordenando cada elemento de la lista según el atributo *tiempoDeReaccion* (valor en coma flotante de los segundos capturados). Puesto que ha sido interesante la manera de crear una función delegada que establece el modo de ordenación de los elementos mostramos un ejemplo breve:

```
reacciones.Sort(delegate(claseReaccionx, claseReaccion y){
  return x.tiempoDeReaccion.CompareTo(y.tiempoDeReaccion); //de menor a mayor
});
```

Control de escenas: en *Unity®* podemos indicar mediante instrucciones que dada una condición como, por ejemplo, el final del juego se debe "saltar" a otra escena en la que se presentan los resultados. Las instrucciones necesarias para este control del flujo de la reproducción de las escenas las podemos resumir en las siguientes:

- `Aplicacion.LoadLevel(indice_escena);` con la que podemos ir a cualquier escena presente en los *Assets*.
- `Application.Quit();` con la que finalizamos la aplicación.

Así mismo, podemos eliminar los objetos de la pantalla empleando el método *Destroy* de la clase *Game Object*. En nuestro caso al estar todos los botones dentro del *Canvas*, basta con destruir el *Canvas* para que todos sus hijos sean destruidos.

Finalmente debemos mencionar que es necesario que las escenas estén incluidas como parte de la "construcción" cuando se va a exportar la aplicación.

3.4.4 Ventajas e inconvenientes

A continuación mencionamos algunas de las ventajas encontradas:

- Permite crear aplicaciones en 2D y 3D.
- El funcionamiento de arrastras y soltar para incluir objetos en la escena hace que el diseño sea sumamente sencillo. Esto reduce el tiempo de desarrollo comparado con otras herramientas.
- La posibilidad de programar los *script* en *JavaScript*, *C#*, o *Boo*, junto con las clases de la *API* permite elevar el grado de abstracción.
- Permite realizar animaciones[31] basadas en línea de tiempo (en la versión 5).

[31] Podemos encontrar detalles de cómo crear animaciones en: <http://docs.unity3d.com/es/current/Manual/animeditor-AnimationEvents.html>. [Consultado: Julio 2015]

- La versión 5 se integra perfectamente con *DirectX*, permitiendo ver efectos gráficos avanzados.
- Soporta algunas técnicas de gráficos avanzados como los mapas de normales y los mapas de luz.
- Permite incorporar modelos 3D complejos creados con otras herramientas, permitiendo para opciones como la reducción de puntos de las mallas, etc.
- Dispone de clases para la conexión con servidores y la comunicación con bases de datos (Ej.: *XMLDocument, WWW, etc.*).
- La clase *XMLDocument* permite comunicar aplicaciones de distintas plataformas, sin que importe el origen de los datos. Esto será de gran ayuda para conectarse a un servidor.
- La clase *Coroutine* permite crear co-rutinas cuya ejecución se realiza en paralelo o en segundo plano permitiendo que la aplicación no se quede detenida.
- Permite la portabilidad de las aplicaciones a casi todas las plataformas: móviles y tabletas (*iOS*, *Android*, *Windows Phone* y *BlackBerry*), navegador web (con *Unity Web Player*, *HTML5*, y *WebGL*), consolas (*Xbox*, *PlayStation* y *Wii*) y escritorio (*Windows*, *Mac* y *Linux*).
- La documentación es extensa y aborda todos los detalles de las clases.
- Existen tutoriales para principiantes y profesionales que abordan todos los pasos de desarrollo de un videojuego. Algunos de éstos están en formato video y otros en texto con imágenes.
- El tiempo de aprendizaje de *Unity®* se reduce al integrar el editor gráfico, los lenguajes de programación basados en *scripting* y la abundante documentación y tutoriales.
- La existencia de la tienda *Asset Store* permite que algunos desarrolladores vendan sus productos a otros usuarios de Unity®
- Existen *SDK* para *Unity3®* con las que realizar reconocimiento del habla, integrar gafas de realidad virtual que pueden elevar el grado de inmersión, etc.
- Integración de la *SDK* de *Android* con la que construir un fichero *.apk*.
- *Android* permite probar ejecuciones de depuración para probar el comportamiento de la aplicación en desarrollo directamente en el dispositivo.

A continuación mencionamos algunos de los inconvenientes encontrados:

- La publicación de aplicaciones comerciales creadas con este motor de videojuegos está sujeta al pago de regalías.
- Precisa conocimientos de programación de lenguajes orientados a objetos y basados en *scripting*.

3.5 Conclusiones

A lo largo de este apartado hemos visto cómo algunos entornos de desarrollo permiten reducir el tiempo de desarrollo de las aplicaciones. Sin embargo la mayoría precisan tener unos conocimientos mínimos de programación, y principalmente de programación orientada objetos.

Estas experiencias han dejado notar las posibilidades que cada una de estas herramientas puede proporcionarnos para crear aplicaciones con las que medir el tiempo de reacción. Los conceptos de programación y uso de funciones específicas como *getTime* o *realTimeSinceStartUp* nos han dado las primeras ideas para capturar los tiempos de reacción.

En el mismo sentido nos han permitido notar la similitud en las maneras de manipular los objetos para cambiar sus propiedades como en nuestro caso para mostrar una palabra con un determinado color. Por ejemplo, en el caso de *Adobe® Flash® Professional* podemos asignar nombres de instancia (etiquetas) a los objetos y así modificarlos mediante código. Lo mismo ocurre en *Unity®* que nos permite adjuntar objetos a variables públicas del *script* con las que acceder a las propiedades y componentes y modificarlos. El caso de *VRML/X3D* es similar ya que se basa en referenciar los nombres de los nodos y manipularlos dentro de un nodo de tipo *Script*.

Los casos que presentan editores de escena, como *Adobe® Flash® Professional* y *Unity®*, sin duda agilizan el diseño permitiendo centrarse en la parte de programación de comportamientos de los objetos, la interacción con el usuario, el comportamiento de interfaces gráficas, animaciones, etc. Sin embargo se debe valorar el objetivo de las aplicaciones y la plataforma de destino ya que el uso de este tipo de herramientas está sujeto al pago de licencias que pueden ser un punto en contra.

El caso de *VRML/X3D* podría solventar el problema económico al ser una herramienta de código abierto y que presenta gráficos 3D y nodos que facilitan la creación de objetos interactivos en la escena. Sin embargo precisa unos conocimientos elevados que pueden incrementar el tiempo de aprendizaje y desarrollo.

Los casos de las tecnología *Flash®* y *VRML/X3D* han permitido observar que se pueden crear aplicaciones para ser compartidas rápidamente a través de internet. Sin embargo la dependencia del *plug-in* que interpreta las aplicaciones es un punto en contra ya que en algunas plataformas no son soportados de forma directa. En el caso de *VRML/X3D* las diferentes extensiones que cada *plug-in* incluye dificultad la compatibilidad entre éstos. Esto puedo ser solucionado con *Unity®* ya que, además de permitir exportar a plataformas *Windows*, *Linux*, *MacOS*, *Android*, etc., también permite exportar las aplicaciones a *WebGL*, *Html5* o incluso empleando el *plug-in* web propio *Unity® Web Player*.

En cuanto a las similitudes que hemos podido experimentar al usar diferentes lenguajes de programación basados en *scripting*, podemos decir que éstos permiten una gran versatilidad para construir estructuras de datos de casi cualquier tipo que consideremos. Ejemplo de ello son los *Array* sin tipo que tanto *JavaScript* como *ActionScript* permiten crear, y las listas dinámicas de *C#*.

En la Tabla 4-4 mostramos un resumen de características encontradas en las herramientas evaluadas enfocadas a nuestro propósito de crear aplicaciones para medir los tiempos de reacción.

	Flash®	***VRML/X3D***	***Unity®***
Editor para el diseño de escenarios	Sí	Sí (generan código farragoso)	Sí
Modelos 3D	No	Sí	Sí
Efectos de propiedades físicas	Sí	No	Sí
Mapas de normales	No	Sí	Sí
Lenguaje de programación	*ActionScript 2.0* y *3.0*	*Java* o *JavaScript*	*JavaScript*, *C#*, y *Boo*
Funciones para medir el tiempo de reacción	*getTime* (valores enteros que expresan los milisegundos)	*getTime* (valores enteros que expresan los milisegundos al pasar el *timeStamp*)	*realTimeSinceStartUp* (valores en coma flotante que expresan los segundos)
Funciones para generar aleatoriedad	*Math.random*	*Math.random*	*Random.range*
Dispositivos para la recepción de las respuestas	Teclado, ratón, entrada táctil, acelerómetros, etc.	Teclado y ratón.	Teclado, ratón, entrada táctil, acelerómetros, etc.
Clases para manejar documentos *XML*	Sí	Sí (en *X3D*)	Sí
Clases para conexión con servidores	Sí	Sí (en *X3D*)	Sí
Portabilidad a dispositivos móviles	Si (incluyendo AIR)	Sí (algunos *plug-in* están soportados en plataformas móviles)	Sí (incluyendo las bibliotecas de funciones necesarias)
Asisten de código	Sí (integrado en *Adobe® Flash® Professional*)	Sí (herramientas de autor sujetas al pago de licencias)	Sí (*Monodevelop* libre y gratuito)
Depurador de código para la corrección de errores	Sí (integrado en *Adobe® Flash® Professional*)	Sí (herramientas de autor sujetas al pago de licencias)	Sí (*Monodevelop* libre y gratuito)

Tabla 4-4: Resumen de características encontradas en las herramientas evaluadas enfocadas a nuestro propósito de crear aplicaciones para medir los tiempos de reacción y eleva el grado de validez ecológica.

Como conclusión principal tenemos que decir que entre las herramientas evaluadas *Unity®* nos ofrece más ventajas puesto que nos permite exportar la aplicación a diferentes plataformas, reducir el tiempo de diseño de las escenas, depurar los *script* con la ayuda de *Monodevelop*, etc. De este entorno cabe destacar el reducido tiempo de aprendizaje y el elevado grado de abstracción al programar el comportamiento de los *Game Object* con *C#*. Ya que por un lado las clases de la *API* disponen de un gran número de métodos que permiten al programador dejar de lado la implementación de tareas repetitivas, como por ejemplo la ordenación de los elementos de las listas, y por otro la existencia de primitivas 3D propias de *Unity®* facilitan la creación de escenarios sencillos. Además tenemos la posibilidad de incorporar texturas, mapas de normales,

etc., que mejoran la creación de escenas realistas que nos pueden servir para incrementar el grado de validez ecológica en la tarea de decisión léxica.

Otra ventaja que podemos encontrar es que, gracias a las posibilidades de portar la aplicación a diferentes plataformas y la existencia de clases para gestionar las conexiones a un servidor como *WWW* y *XMLDocument*, permiten crear aplicaciones que posean un comportamiento de almacenamiento en la nube. Es decir, podemos crear aplicaciones para varias plataformas de tal manera que todas tengan la posibilidad de conectarse al mismo servidor para, por ejemplo, almacenar y consultar datos. Esto puede ser utilizado cuando queremos recopilar una gran cantidad de datos o cuando se quiere permitir el acceso multiusuario. En nuestro caso lo podemos emplear para almacenar datos de la interacción con el usuario como el tiempo de reacción, el orden de selección, la configuración del escenario, etc.

4 Aplicación para realizar tareas de decisión léxica a través de un juego serio

Este apartado recoge las conclusiones y resultados finales del diseño y la implementación de una aplicación multiplataforma que permite realizar tareas de decisión léxica a través de un juego serio y almacenar los datos en una base de datos para su análisis. Incluye las descripciones de los trabajos realizados en las distintas fases de desarrollo como el análisis de requisitos, el diseño de la plataforma de juego, la descripción del prototipo desarrollado, un informe de los experimentos realizados y la presentación de algunos resultados del análisis de los datos recogidos, de entre los cuales destacamos: la verificación de la reducción del tiempo de reacción (TR) debido al efecto de la práctica y la aplicación de la aleatoriedad en varias variables independientes como el orden, el color, o la posición, para minimizar los efectos de la memorización en los TR.

La tarea de decisión léxica que vamos a abordar a lo largo de estos apartados reúne las mismas características que describimos para la prueba de *Stroop* en el apartado "Evaluación de herramientas software" referentes al diseño del experimento. Además presenta el mismo funcionamiento de las pruebas psicológicas basadas en estímulo-respuesta que usan el tiempo de reacción como una de las variables dependientes a analizar. En nuestro caso incluso podremos arrojar datos referentes a la interacción del participante como el orden de selección. A continuación mostramos un resumen de las características de configuración del experimento para la tarea de decisión léxica en la que presentamos escena con una estantería en la que se incluyen varios libros, la cual se describirá con detalle más adelante.

- Variables manipuladas en los sujetos: en nuestro caso la edad y el sexo.
- Variables independientes manipuladas en la prueba: el color, el orden dentro de la repisa, y la posición (inferior, superior) dentro de la estantería.
- Variables dependientes: en nuestro caso el tiempo de reacción, la tasa de aciertos, y el orden de selección.
- El error de medida no tiene gran influencia en la validez interna del experimento: en nuestro caso con número de objetivos a seleccionar veremos que la media de los tiempos se sitúa por encima de 1 segundo, por lo que los retardos del *Hardware* o *Software* de los dispositivos, que es del orden de milisegundos, no influye significativamente en los tiempos recogidos.
- Aleatoriedad para conseguir una distribución uniforme de las variables extrañas o fuentes de confusión: en nuestro caso aplicamos aleatoriedad a los colores, el orden dentro de la repisa, y la posición (repisa superior y repisa inferior) de los libros en la estantería.
- Neutralización de variables extrañas mediante el reparto equitativo de su influencia a través de los niveles de variable manipulada: en nuestro caso

veremos que es posible emplear una semilla con la que repetir la configuración aleatoria del escenario (colores, orden, y posiciones de los libros) entre diferentes sujetos basándonos en repetir la misma configuración según el número de sesiones realizadas.

- Validez ecológica: veremos que se emplean mapas de normales e iluminación para dar mayor realismo al escenario.

La herramienta elegida para el desarrollo de la aplicación ha sido *Unity®* ya que ésta nos permite tanto diseñar el escenario con primitivas 3D como depurar los *script* con *Monodevelop*. Además nos permite abordar uno de los puntos clave de la aplicación que es la posibilidad de portarla a varias plataformas con lo que se incrementaría las posibilidades para recoger datos y almacenarlas en una misma base de datos. Para encontrar más razones de la elección podemos recurrir al apartado "Conclusiones" anterior o al apartado "Implementación" de la prueba de *Stroop* con *Unity®* donde describimos algunas de las facilidades que nos ofrece esta herramienta.

4.1 Ámbito y alcance de la aplicación

El destino de esta aplicación es intentar proporcionar una herramienta alternativa que emplee las nuevas tecnologías para abordar el estudio de los subprocesos mentales que se producen durante la lectura de palabras de una forma más atractiva y amistosa para los participantes. Para ello se propuso la realización de la tarea de decisión léxica a través de un juego serio que a la vez permita elevar el grado de validez ecológica de la tarea.

Como explicamos en el apartado "Uso en tareas de decisión léxica" sobre el uso del tiempo de reacción como variable dependiente en estudios de psicología experimental, la tarea de decisión léxica consiste en ir presentando al participante, de forma escrita u oral, una serie de palabras provenientes de un conjunto de palabras compuesto por palabras y no-palabras. Tras la presentación el participante debe decidir si se trata de una palabra o una no-palabra.

Su uso está vinculado principalmente a estudios de psicología sobre el acceso léxico los cuales pretenden abordar maneras de prevenir o diagnosticar, por ejemplo, los problemas de acceso léxico que surgen debido a la vejez o los que sufren las personas que padecen algún tipo de dislexia.

Como sabemos para leer palabras hacen falta conocimientos acerca de éstas. Estos conocimientos se almacenen en lo que se ha denominado léxico interno y que se relaciona con la memoria a largo plazo. Estos almacenes son: el léxico visual, el léxico auditivo, el léxico fonológico y el sistema semántico.

Nuestro juego serio se relaciona por tanto con el léxico visual y el sistema semántico, de los cuales podemos decir que el léxico visual está formado por las representaciones visuales que poseemos de las palabras tras haberlas visto suficientes veces, y el sistema semántico es el almacén que contiene los significados de las palabras que se debe activar al leer o decir una palabra.

Algunas aplicaciones permiten realizar estas tarea sin embargo presentan escenarios con un aspecto demasiado sintético y en muchos casos tan solo presentan texto en pantalla, lo cual hace que los datos se contaminen al perder validez ecológica como explicamos en el apartado "Problemas en la medición".

Por ello proponemos un juego serio con gráficos 3D que permita elevar el grado de validez ecológica de la tarea, es decir, un videojuego que permita recrear en una escena que simule una situación cotidiana sencilla que ayude a que los participantes olviden que están realizando una prueba cognitiva en la que se están recogiendo datos sobre su interacción.

La aplicación desarrollada está compuesta de los siguientes **elementos principales**:

- Un escenario que muestra una estantería con varios libros de diferentes colores y diferentes títulos en sus tapas construidos con gráficos 3D en base a parámetros predefinidos: número de libros por repisa, número de repisas, número de estanterías, conjunto de títulos, conjunto de títulos objetivo, identificador del participante y número de sesión a realizar.
- Un grupo de funcionalidades internas que gestiona la configuración del escenario en base a determinados parámetros para cada participante, así como la interacción con el participante durante la prueba.
- Un servicio de almacenamiento centralizado que permite almacenar los datos de las sesiones realizadas en diferentes dispositivos en una misma base de datos.
- Una interfaz gráfica sencilla e intuitiva para facilitar la introducción de datos personales de los supervisores y los participantes, así como para navegar por las diferentes pantallas de la aplicación.

Actualmente se dispone de dos versiones de la aplicación: una para estudiar la reducción del tiempo de reacción por el efecto de la práctica, y otra para contrarrestar los efectos de la memorización (de posiciones, colores, etc.) mediante la aplicación de aleatoriedad.

Los participantes voluntarios para la validación de la aplicación y la captura de los datos fueron amigos y familiares de quien presenta este proyecto. Los cuales no presentaban ningún tipo de demencia o problemas mentales y tampoco dificultades en la lectura. Así mismo las sesiones se realizaron de forma supervisada controlando que no existan interferencias externas como el ruido y que los participantes no tuvieran problemas en el uso de los dispositivos de captura (ratón). Los experimentos se

realizaron de forma que las dos versiones de la aplicación almacenen sus datos en bases de datos diferentes para su análisis por separado. En ambos experimentos participaron los mismos sujetos con el mismo número de sesiones.

Durante la fase de evaluación se comprobó la atracción que producen los videojuegos en personas jóvenes así como en personas mayores y el afán de superación (reducción del tiempo de reacción) que surge en los participantes cuando la configuración del escenario es siempre aleatoria. Así mismo se comprobó que los retardos que se pudieran introducir en los TR capturados debido al *Hardware* de la máquina eran insignificantes frente a los tiempos que tardaban los participantes en seleccionar tres libros.

Por último debemos mencionar que el prototipo de esta aplicación está construido de forma modular con opciones de escalabilidad, lo cual puede permitir añadir futuras ampliaciones de forma relativamente rápida para facilitar el trabajo de los supervisores y los psicólogos en cuanto a la configuración del experimento: añadir más repisas, más libros, cambiar la orientación de los libros, presentar varias estanterías, restringir la gama de colores, etc.

4.2 Requisitos

La primera tarea abordada para el desarrollo de la aplicación consistió en hacer un estudio de los requisitos que en primera instancia debía cumplir la aplicación antes de entrar en la fase de análisis y diseño. El objetivo es delimitar el problema abordado que es la creación de una aplicación para realizar tareas de decisión léxica a través de un juego serio, para lo cual debemos definir las características de ésta, los actores involucrados, y los requisitos no funcionales.

Las características generales que deben tener la aplicación son:

- Debe permitir configurar el experimento.
- Debe construirse como herramienta de apoyo que facilite la captura de datos en pruebas supervisadas.
- Debe restringir el acceso no autorizado.
- Los datos deben almacenarse de forma consistente para facilitar el análisis posterior.
- Debe contemplar diferentes perfiles de participantes (edad, sexo, etc.).
- Los elementos del escenario deberán ser perfectamente identificables por cualquier participante.
- Se debe construir con opciones de escalabilidad y opciones de modificación para facilitar ampliaciones o mejoras como las que habría que hacer para abordar el

estudio del deterioro cognitivo con el mismo escenario restringiendo el acceso en un período de tiempo.

4.2.1 Actores

Supervisor: será la persona responsable (posiblemente un psicólogo) de controlar la correcta realización de la prueba verificando aspectos como el estado emocional del participante, el ruido externo la sala, etc. Deberá crearse y almacenarse en la base de datos antes de permitirle el acceso.

Participante: será la persona que realice la tarea de decisión léxica a través del juego serio buscando y seleccionando determinados libros de la estantería mediante la identificación por los títulos. Deberá crearse y almacenarse en la base antes de permitirle el acceso.

Servidor: será la entidad que se encargue de la recepción de inserciones, consultas y actualizaciones en la base de datos.

Analista: será la persona que realice el análisis de los datos recogidos en la base de datos.

4.2.2 Características

La búsqueda de características se dividió en cuatro partes de las cuales la primera y la más importante sirvió para identificar las características principales que debía tener el **escenario** del juego serio, ya que éste es el centro atracción para los participantes y nos debe permitir modificar elementos y añadir nuevos funcionalidades de forma escalable para configurarlo en función de los parámetros predefinidos del experimento. Esta parte se corresponde con la emisión de estímulos que hemos tratado en apartados previos. A continuación resumimos los resquicitos principales encontrados para el escenario:

- Debe aumentar la validez ecológica con gráficos 3D.
- Debe permitir que casi todos los parámetros del experimento sean modificables: posiciones, colores, títulos, tamaños, etc.
- Debe responder (redibujarse) en función de la interacción con el participante.
- Debe diseñarse de forma que permita la escalabilidad.
- Debe generarse únicamente en el momento de inicio de la prueba
- Debe configurarse en función de los parámetros del experimento (aleatoriedad, coherencia, etc.).
- No debe tener decorados adicionales que no tengan función alguna.

La segunda parte abordó la identificación de las fuentes de captura así como la **gestión de la interacción** con el usuario, es decir, la identificación de las maneras de capturar las reacciones y la secuencia de pasos para guardar los datos y redibujar el escenario. A continuación mencionamos algunas características encontradas:

- Las fuentes de captura deben facilitar realizar la tarea y no entorpecerla.
- Las capturas se deben automatizar de tal forma que con cada captura se emita un nuevo estímulo (redibujado del escenario).
- No se deben realizar conexiones con el servidor durante la captura de respuestas. Éstas se harán al finalizar la sesión.
- Las modificaciones del escenario tras una captura no deben distraer al participante.
- Las capturas deben realizarse todas de la misma manera para que los pequeños retardos se distribuyan uniformemente en los datos recogidos.

En la tercera parte se contempló la **gestión de la comunicación de la aplicación con una base de datos** para almacenar los datos recogidos de una sesión realizada y consultar parámetros para la configuración del escenario por participante. A continuación resumimos las características:

- Se deben crear varias capas para gestionar las comunicaciones de la aplicación con la base de datos.
- Un servidor que atienda peticiones hará de intermediario entra aplicación y base de datos.
- La aplicación deberá contemplar los fallos de conexión y actuar en consecuencia, invalidando los datos o reconectándose.
- La estructura de la base de datos debe poseer una organización basada en la sesión: fecha, participante, clics, libros seleccionados, tiempos de reacción, etc.
- La estructura de la base de datos debe permitir ciertas restricciones como no introducir duplicados, no borrar datos de sesiones ya realizadas, etc.

La cuarta parte abordó el análisis de la incorporación de funcionalidades que faciliten la labor de los supervisores de la prueba mediante una **interfaz gráfica de usuario** intuitiva y amigable.

- La organización de los elementos debe ser, en la medida de lo posible, igual en todas las pantallas para dar mayor rapidez de aprendizaje en el uso.
- Debe contemplar la emisión de *feedback* visual al usuario: cambio de colores, resaltados, menú desplegable, etc.
- Debe permitir distinguir las diferentes pantallas: colores de fondo.
- Debe poseer una navegación secuencial: selección supervisor, selección participante, selección títulos, realización de la prueba, resultados.
- Debe permitir al usuario volver atrás.

- Debe trabajar conjuntamente con el apartado que gestiona las conexiones a la base de datos: mostrar listas descargadas en pantalla, comprobar contraseña, etc.

Finalmente una última parte analizó los **requisitos no funcionales** que se debían cumplir:

- Operatividad: la plataforma (aplicación + servidor + base de datos) debería estar disponible las 24 horas del día, los 7 días de la semana y durante los
- Protección de datos: los datos personales de los usuarios del sistema deberían estar protegidos de acuerdo a lo establecido en la Ley Orgánica de Protección de datos.
- Portabilidad: la plataforma debería desarrollarse pensando que puede ser usada en varias plataformas.
- Escalabilidad: la plataforma debería crearse con una arquitectura escalable, de tal forma que tenga la capacidad de permitir en el futuro el desarrollo de nuevas funcionalidades, modificar o eliminar funcionalidades después de su construcción y puesta en marcha inicial.

4.3 Análisis y diseño

La segunda fase del desarrollo de la aplicación abordó el análisis y diseño del sistema propuesto. La plataforma en su conjunto está formada por cuatro grandes apartados: escenario, gestión de la interacción con el participante, gestión de las conexiones a la base de datos, e interfaz gráfica de usuario. En los siguientes apartados vamos a mencionar las características diseñadas.

4.3.1 Escenario del juego serio

La idea principal de este juego serio es que debe presentar una escena lo más realista posible en la que se muestre una estantería de madera con varios libros colocados con diferentes características: color, posición, título del libro. El participante debe buscar y seleccionar varios libros objetivo cuyos títulos se le habrán indicado previamente. Con esto pretendemos dar una manera alternativa de realizar las tareas de decisión léxica al tratarse de una búsqueda e identificación de títulos objetivo, ya que el funcionamiento es similar a la tarea que presenta palabras y no-palabras que habíamos comentado en el apartado “Uso en tareas de decisión léxica”.

4.3.1.1 Estantería

Las características identificadas para la escena que simula la estantería de madera y los libros se basaron principalmente en elevar la validez ecológica[32] y a la vez se permitir la reconfiguración de parámetros como el número de libros por repisa, el número de repisas, etc.

El boceto principal de este escenario tenía la intención de presentar una réplica de una estantería real con todos sus componentes de tal forma que el participante observe una escena cotidiana y realista. En la Figura 5-1 podemos observar el boceto original del escenario del que destacamos los siguientes puntos:

- La estantería sería una réplica de una estantería real conteniendo elementos como tablas laterales, tabla trasera, tabla superior, varias repisas, y dos puertas en la parte inferior.
- Cada repisa contendría un mismo número de libros.
- Todos los libros serían de iguales dimensiones.
- Todos los títulos de los libros tendrían la misma orientación.
- La mesa adyacente contendría siluetas de libros objetivo que serían usadas a modo de recordatorio para el participante.
- Para elevar el grado de realismo se usarían texturas y mapas de normales[33] acordes al objeto. Para las tablas textura de madera y mapas de normales que simulen las vetas de la madera. Para los libros colores diferentes y mapas de normales que simulen las hojas y las tapas.
- Se le permitiría al participante girar ligeramente la cámara que enfoca la escena para simular una búsqueda más dinámica.

[32] La validez ecológica del experimento es un tipo de validez interna que analiza el entorno de prueba y determina cuánto influye en el comportamiento.

[33] Los mapas de normales son un conjunto de vectores en tres dimensiones que permiten simular relieve o hendiduras en la superficie de un modelo 3D en función del ángulo entre éstos, el vector *Up* de la cámara, y el vector de dirección de las fuentes de iluminación de la escena. Éstos se emplean para reducir el coste espacial y de cómputo ya que permite crear efectos de iluminación en lugar de crear objetos con más vértices.

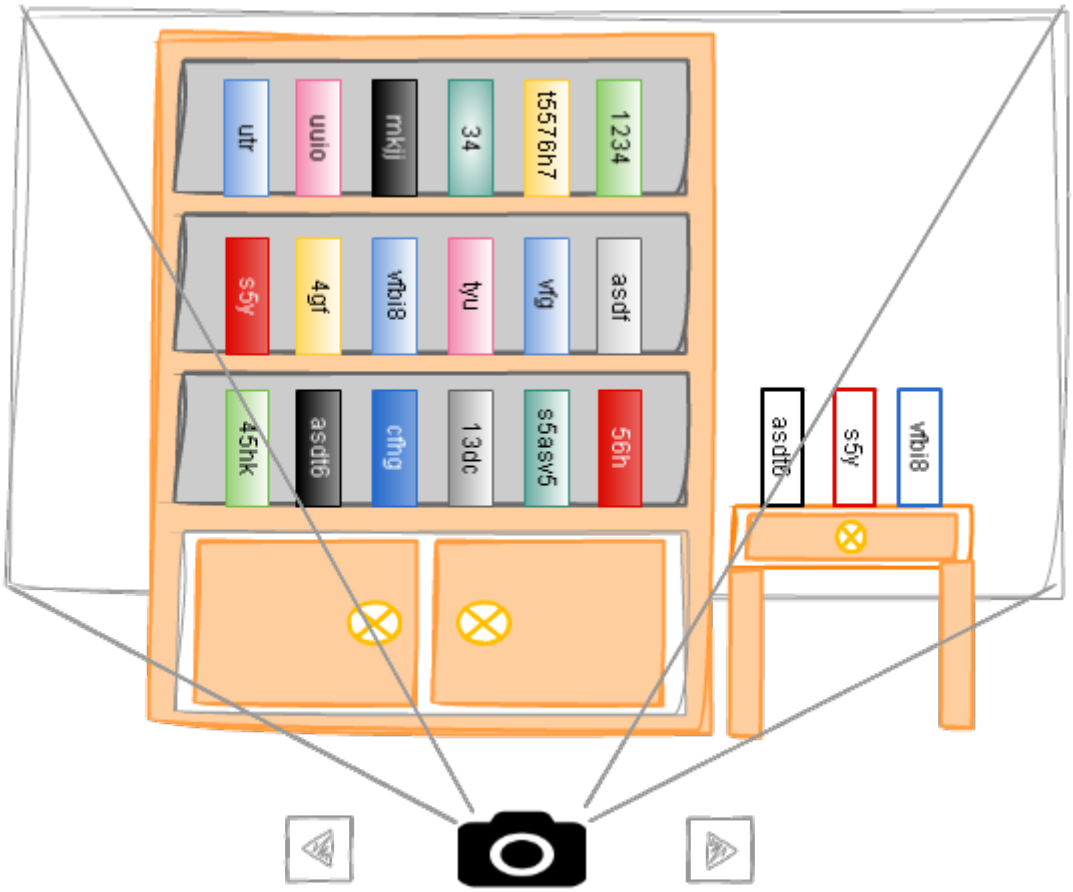


Figura 5-1: Boceto de la configuración del escenario (estantería, mesa, libros y cámara). Análisis y diseño del escenario del juego serio para realizar tareas de decisión léxica.

Para abordar la construcción de este escenario que debía cumplir la característica de ser re-configurable, se decidió que la construcción de los principales elementos como las tablas, las repisas, y los libros, debía hacerse mediante código dejando pequeños detalles como la creación de materiales (madera con textura mapas de normales, tapas de libros con colores y mapas de normales, hojas, etc.) para realizarse de forma manual y que luego sean aplicados a los objetos de la forma que explicamos en la Figura 4-21 del apartado “Scripting” de *Unity*®.

Por ello se realizó una lista de los **parámetros** que la aplicación debía permitir que sean **modificables**:

- Número de compartimentos por repisa.
- Número de libros por repisa.
- Número de libros por repisa <= Número de compartimentos disponibles.
- Número de repisas.
- Número de títulos.
- Número de títulos objetivo.
- Posiciones de los libros dentro del espacio interno de la repisa.
- Compartimentos vacíos.

- Orientación del libro (vertical, horizontal).
- Orientación de los títulos dentro de la tapa del libro (vertical izquierda, vertical derecha).
- Tipo de fuente de los títulos.
- Material (textura y mapa de normales) de las tablas.
- Material (color y mapa de normales) de las tapas de los libros.
- Material (textura) de las hojas de los libros.
- Dimensiones de los elementos principales: repisa, compartimentos, libros. (Modificables con límites de corte para evitar solapamientos)

Así mismo se contempló la posibilidad de que en futuras ampliaciones los libros tengan propiedades físicas que podrán servir para simular la caída de la estantería hacia una canasta, etc. así podrían ir recogiendo libros de varias estanterías.

4.3.1.2 Repisa

La construcción de los objetos de la repisa está conformada por dos grupos de espacio: el espacio interno y el espacio externo. El espacio interno abarca el espacio propiamente de los espacios de los compartimentos y los libros sobre la repisa, y el espacio externo abarca tanto el espacio interno como el espacio ocupado por las tablas adyacentes. Para entender mejor la organización requerida podemos observar las figuras: Figura 5-2 y Figura 5-3.

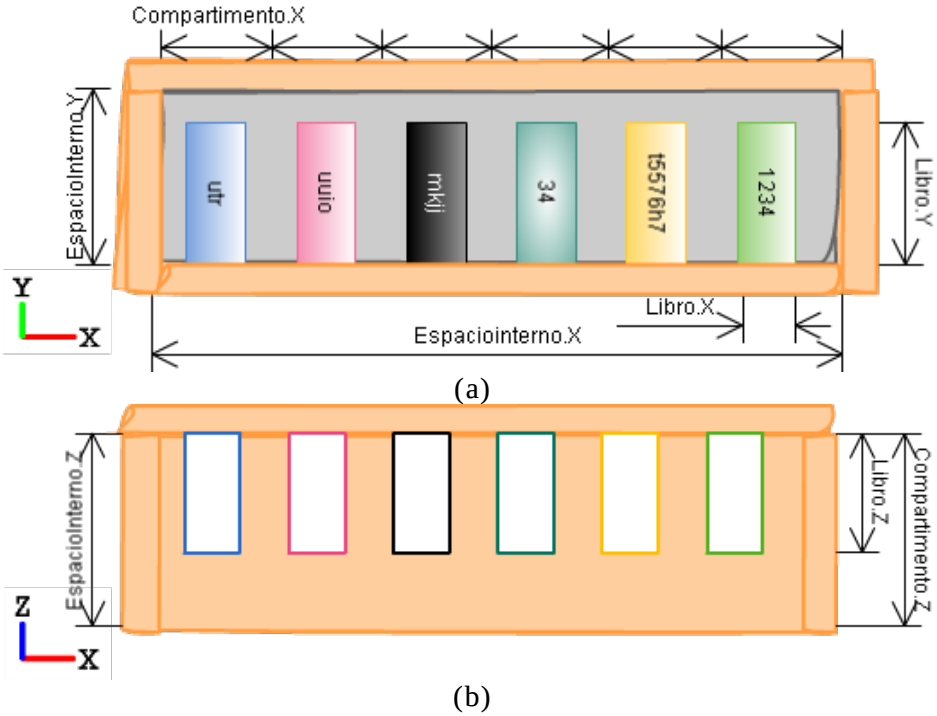


Figura 5-2: Boceto de las dimensiones del espacio **interno** de una repisa. (a) Dimensiones en XY. (b) Dimensiones en XZ. Análisis y diseño del escenario del juego serio para realizar tareas de decisión léxica.

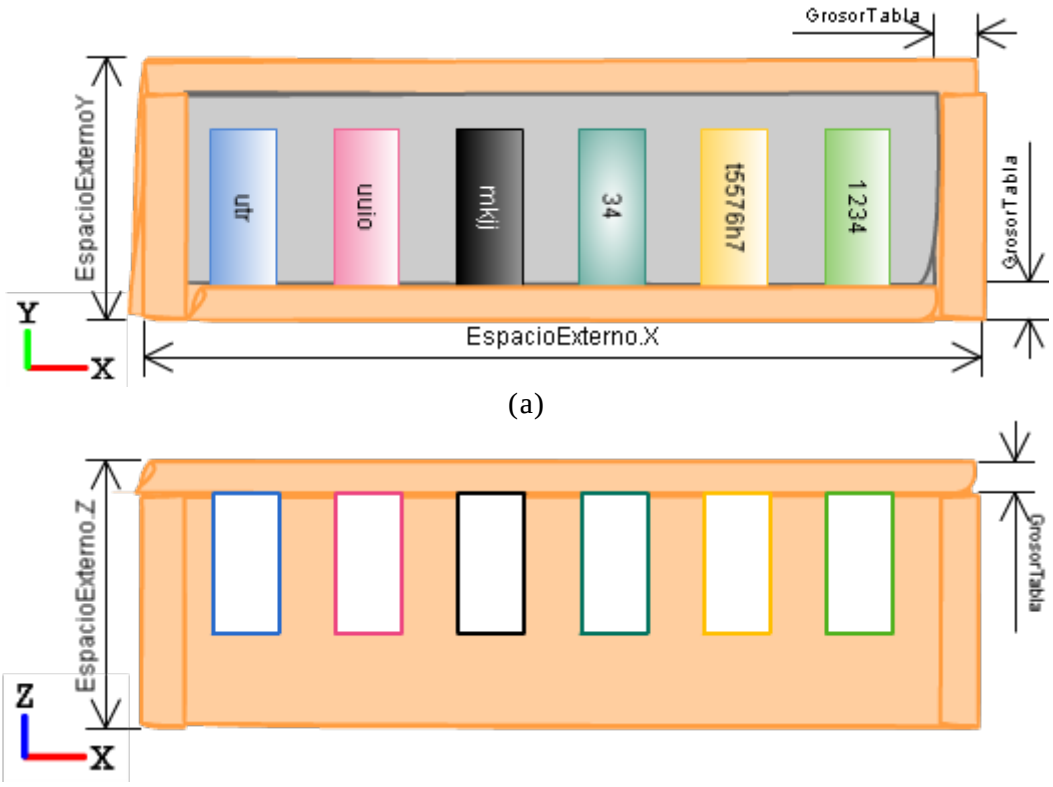


Figura 5-3: Boceto de las dimensiones del espacio **externo** de una repisa. (a) Dimensiones en XY. (b) Dimensiones en XZ. Análisis y diseño del escenario del juego serio para realizar tareas de decisión léxica.

Para la construcción se debe tener en cuenta que los objetos poseen un **centro pivote** u origen local desde el que se "miden" las unidades de tamaño otorgadas, por tanto se deberán realizar las operaciones matemáticas necesarias para, en función de dicho centro, colocar correctamente los objetos sin solapamientos. Esto puede evitar problemas futuros cuando se añadan propiedades físicas a los libros como asignarle un cuerpo rígido *RigidBody* que emplea detección de colisiones y que puede dar un comportamiento anómalo si los objetos se crean ya colisionados.

Para evitar este potencial problema futuro así como para distribuir uniformemente los elementos se propone la implementación de algunas funciones que realicen los siguientes trabajos:

- Calcular las dimensiones del compartimento en base a las dimensiones del espacio y el número de compartimentos requerido.
- Calcular las dimensiones del libro en base a las dimensiones del compartimento de tal forma que se permita que haya suficiente espacio entre libros como para evitar que el participante haga clic en dos libros adyacentes.
- Calcular las posiciones de los centros de libros en base a la altura y el grosor de la tabla base (inferior) de la repisa sobre la que se van a colocar y las dimensiones del libro. Esto debe permitir colocar los libros casi pegados a la

tabla trasera y con una ligera elevación de décimas sobre la cara superior de la tabla base.

La Figura 5-4 servirá para aclarar la manera realizar los cálculos necesarios para colocar los objetos en base a los centros.

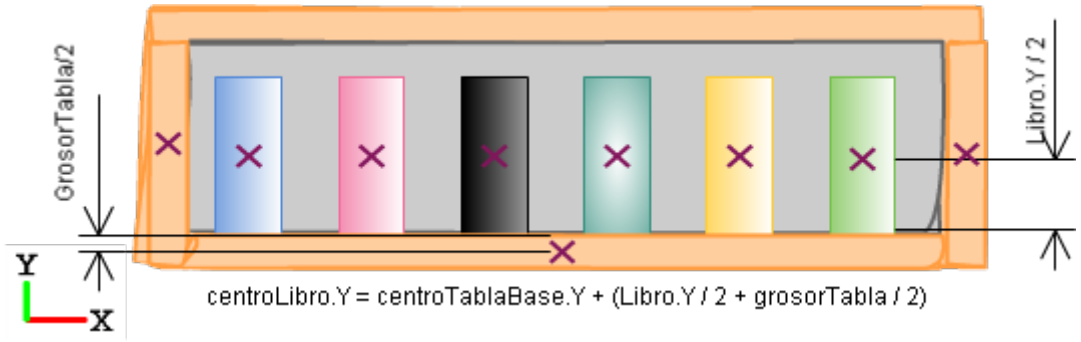


Figura 5-4: Boceto que muestra un ejemplo del cálculo de las posiciones de los libros dentro del espacio interno de la repisa en base a los **centros de los objetos**. Los centros se observan marcados con una X de color violeta. Análisis y diseño del escenario del juego serio para realizar tareas de decisión léxica.

4.3.1.3 Libro

La creación de este elemento debe tener en cuenta las características de la tarea, es decir, puesto que nuestros estímulos van a ser libros éstos deben contener todos los elementos posibles para parecerse lo máximo posible a un libro real. El punto principal es que debe existir una tapa que contenga un título que identifique el libro y que ésta se encuentre a la vista del participante. El color puede asignarse directamente a la primitiva 3D que en definitiva será un cubo.

La manera de agrupar el cubo base, las tapas y el título se hará siguiendo el mismo mecanismo basado en los centros de los objetos, es decir, al cubo base se le adjunta dos planos laterales donde colocaremos las texturas de las tapas y un plano en vertical que simulara la tapa en la que se coloca el título.

Esto se realizará de esta manera ya que cuando se aplica una textura a una primitiva 3D ésta se aplica en todas las caras de la primitiva, sin embargo nosotros requerimos simular las diferentes partes del libro con diferentes características: tapas con color y textura, tapa con título y hojas. Por lo que es mejor adjuntar planos laterales a un cubo base que actuará como padre y del que dependerán todos los objetos en cuanto a su posición tamaño, orientación, y existencia. En las Figura 5-5 y Figura 5-6 podemos observar la manera en la que se colocarán los planos laterales sobre el cubo base para simular las tapas de libro, así como la adición de texturas, mapas de normales y texto 3D a éstas.

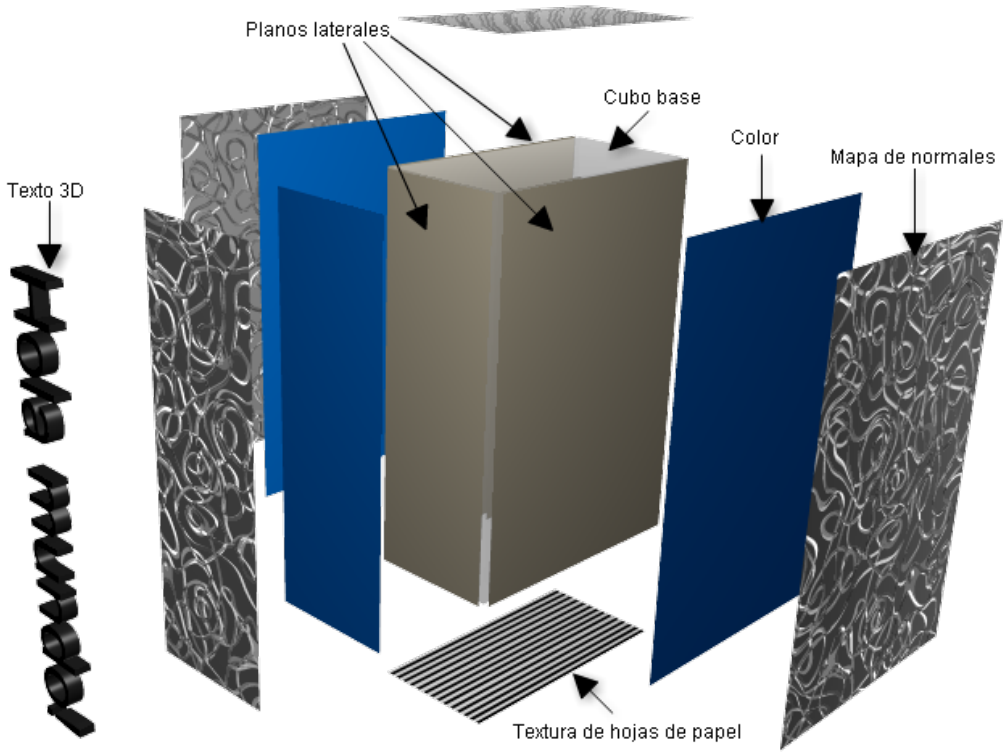


Figura 5-5: Boceto que muestra la manera en la que se que se colocan los planos laterales sobre el cubo base para simular las tapas de libro, así como la adición de **texturas**, **mapas de normales** y **texto 3D** a éstas. Análisis y diseño del escenario del juego serio para realizar tareas de decisión léxica.

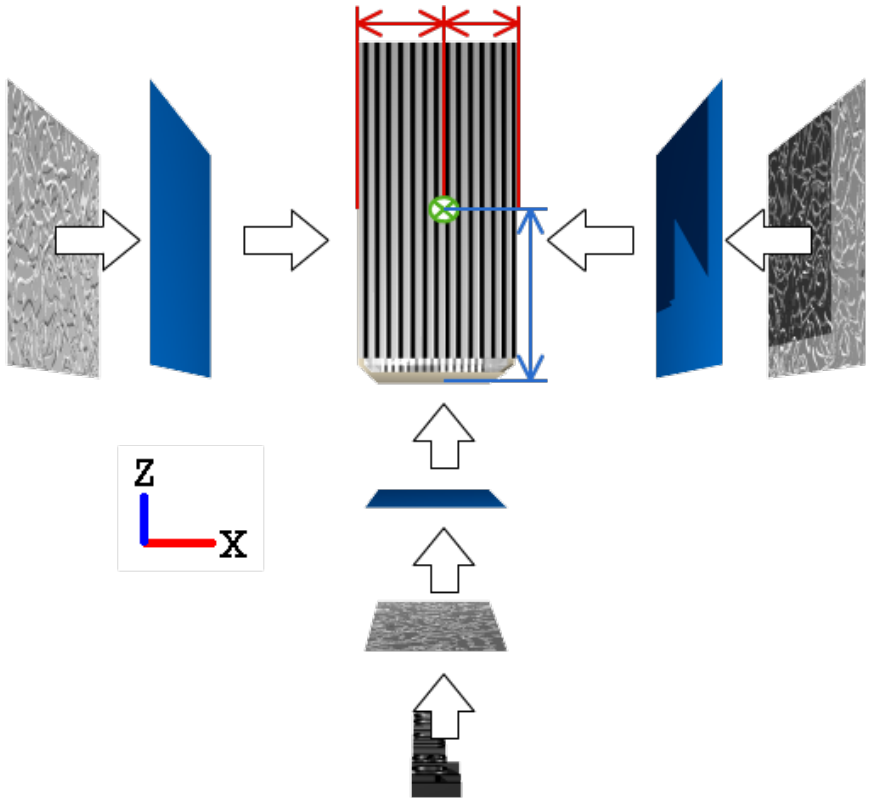


Figura 5-6: Boceto que muestra un ejemplo de cálculo de las **posiciones de los componentes** (planos que simulan las tapas) del libro en base a los **centros de los objetos**. Análisis y diseño del escenario del juego serio para realizar tareas de decisión léxica.

Se debe tener cuenta, como hemos dicho, que la creación se debe hacer de forma jerárquica padre-hijos de tal forma que cuando se proceda a destruir un libro se destruyan todos sus hijos: tapas lateras, tapa frontal y título.

4.3.2 Interacción con el participante

Abordando la búsqueda de posibles fuentes de captura de las respuestas que el motor de *Unity®* nos permite usar la clase *Input* para gestionar los eventos de entrada de varias fuentes, es decir, con la misma clase podemos usar diferentes métodos para los eventos del teclado, del ratón, del acelerómetro de los dispositivos móviles, la entrada táctil, entre otros. Es en este punto donde la separación entre los libros toma relevancia ya que en los dispositivos móviles se emplea la entrada táctil y esta separación ayudará a evitar errores de selección.

Una vez halladas las posibles fuentes de captura seleccionamos la entrada del ratón (o la entrada táctil) para pasar al siguiente paso en el que analizamos la manera de capturar los eventos de entrada para, en consecuencia, modificar el estado del escenario.

Puesto que buscamos que el participante no se distraiga con animaciones complejas, como podría ser el uso de un *RigibBody* con el que simular la caída del libro de la estantería, que podrían influir en su tiempo de reacción, encontramos que la mejor manera de mostrar el resultado del clic era mediante la desaparición del libro.

El proceso de captura de las respuestas a implementar deberá cumplir las siguientes funcionalidades:

- Crear una estructura donde almacenar: tiempo de reacción, libro seleccionado, y acierto.
- Las funcionalidades de este proceso se activan únicamente cuando se está realizando la prueba.
- Atender únicamente a eventos destinados a seleccionar libros y no a otros objetos, por lo que habrá que encontrar la correspondencia entre posición del clic (o toque) dentro de la pantalla en dos dimensiones con la posición de un libro en el escenario 3D.
- Guardar los títulos de los libros seleccionados, el tiempo de reacción para cada selección, el acierto o fallo, y el orden de selección.
- Actualizar los parámetros de la estantería y las repisas tras una selección: decrementar el número de libros presentes en la repisa, decrementar el número de libros presentes en la estantería.
- Emitir *feedback* visual para que el participante se entere de que su evento de clic se ha producido así evitaremos que mantenga pulsado el botón.

Para aclarar el orden en el que se realizarán estas funcionalidades mostramos el diagrama de transición de estados de la Figura 5-7.

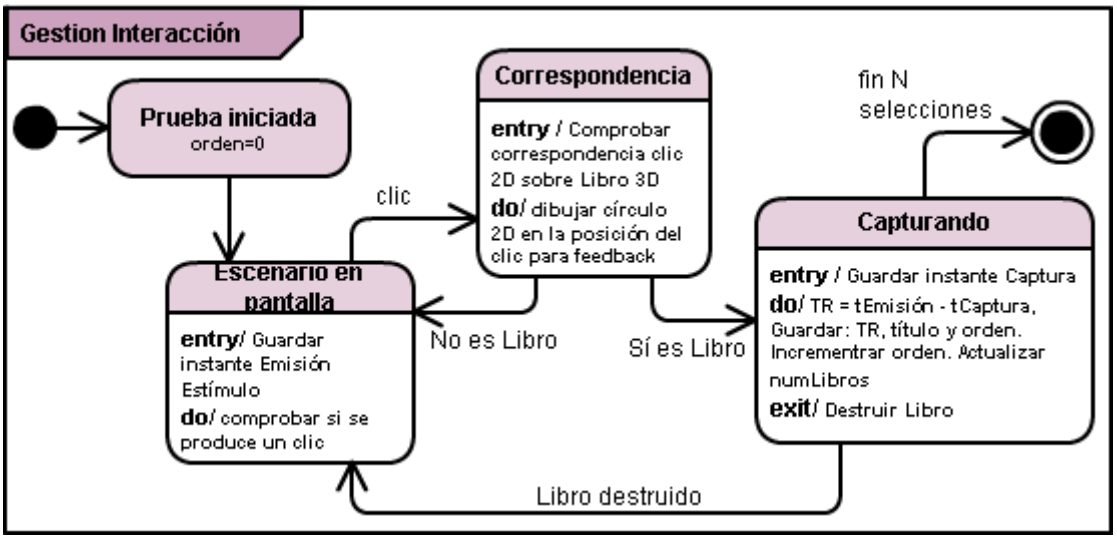


Figura 5-7: DTE del gestión de la interacción de la aplicación con el participante durante la realización de la prueba. Análisis y diseño de la gestión de la interacción de la aplicación con el participante.

4.3.3 Comunicación con la base de datos

Para la comunicación con la base de datos debemos contemplar la característica de almacenamiento centralizado que queremos que la aplicación posea. Por este motivo tenemos que recurrir al uso de un servidor que ayude a la gestión de las inserciones y actualizaciones de datos en la base de datos, teniendo que organizar las operaciones de comunicación mediante capas.

Es decir en algunos casos vamos a necesitar que la aplicación sea capaz de enviar una consulta al servidor, que el servidor acepte o rechace la consulta, extraiga los datos de la base de datos, retorne los datos hacia la aplicación, y que la aplicación los reciba, compruebe si no hay errores y los almacene en una estructura de datos para ser usados, por ejemplo, para colocar los títulos en las tapas.

Así mismo esta comunicación servirá para mantener la coherencia al realizar las pruebas ya que, por ejemplo, podremos consultar los mismos títulos de los libros en cada inicio de sesión y mostrarlos a diferentes participantes. Así mismo servirá para guardar datos personales de los participantes como el *alias* de usuario, la contraseña, la edad y el sexo, y de esta forma podremos crear funcionalidades como la restricción de acceso por contraseña, la configuración del escenario (títulos, posiciones de los libros, etc.) en base a algún parámetro almacenado, etc.

La independencia entre la base de datos colocada en un servidor externo y la aplicación nos va a permitir que la aplicación se distribuya en dispositivos diferentes y que la

información que se recoja se agrupe en un mismo sitio. Con esto conseguiríamos recopilar una mayor cantidad de datos con los que realizar análisis más ajustados a la realidad de las reacciones de los participantes.

La base de datos será la estructura de apoyo cuya organización en tablas debe permitir almacenar toda la información de interés de realización de la prueba y debe crearse de forma que permita añadir nuevos atributos de forma escalable. Las tablas principales que como mínimo debería poseer la base de datos así como las relaciones entre éstas las mostramos en la Figura 5-8.

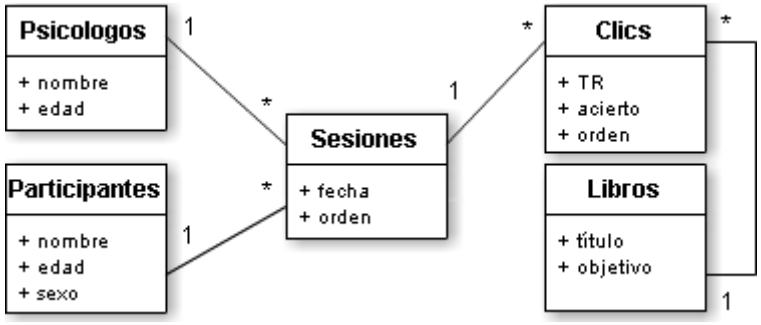


Figura 5-8: Diagrama de tablas de la base de datos. Análisis y diseño de la comunicación de la aplicación con la base de datos.

Para la gestión de las comunicaciones entre aplicación y base de datos debe existir una capa intermedia que realice las tareas apropiadas. En esta capa será ocupada por un servidor externo que responda a las consultas de la aplicación.

La programación del comportamiento específico requerido en el servidor se debe realizar con algún lenguaje propio de éstos. Por ello encontramos que el lenguaje *PHP* posee una gran versatilidad y potencia en cuanto a la gestión de diferentes tareas relacionadas con los entornos web: crear páginas *Html*, configurar la manera en la que se atienden las peticiones al servidor, comunicarse con bases de datos, enviar resultados de salida a los host que realizan peticiones al servidor, etc. En la Figura 5-9 mostramos un esquema de la organización por capas de los elementos que intervienen en las conexiones de la aplicación con la base de datos.

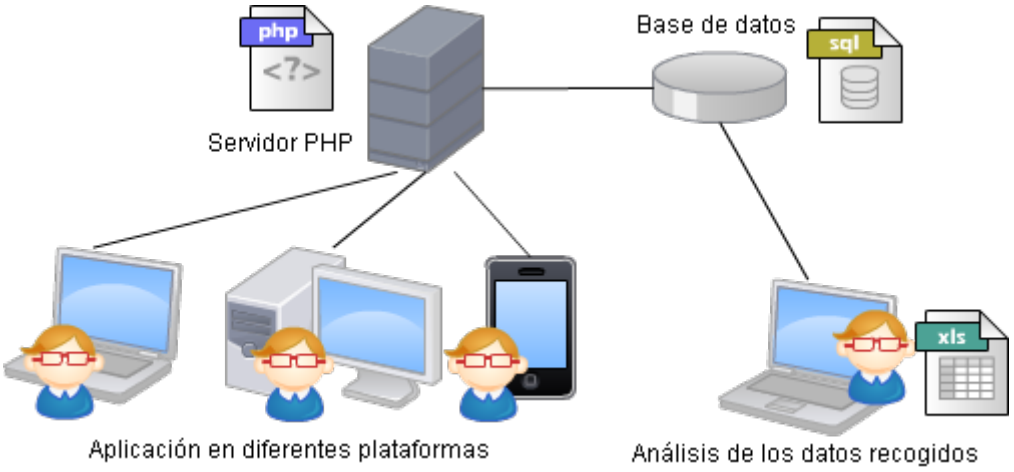


Figura 5-9: Esquema de la organización por capas de los elementos involucrados en la comunicación de la aplicación con la base de datos. Análisis y diseño de la comunicación de la aplicación con la base de datos.

Finalmente para organizar los instantes en los que se debe permitir la conexión con el servidor con el objetivo de no realizar tareas adicionales durante la realización de la prueba, mostramos en la Figura 5-10 un diagrama *BPMN* que muestra la evolución de las peticiones al servidor cuando se inicia una nueva sesión de un participante existente. Se muestra la interacción de la aplicación con el supervisor y el participante, y la comunicación con la base de datos.

El resto de comunicaciones atendidas por el servidor en los casos de la creación de un nuevo usuario, un nuevo participante, o un nuevo libro, se realizarán de forma similar a las seis primeras tareas del servidor que se observan en el diagrama de la Figura 5-10 que hemos mencionado. Dichas inserciones de nuevos datos se harán desde cada uno de los paneles (panel supervisores, panel participantes, panel títulos) en los que se introducirán los valores apropiados. Por ejemplo para insertar un nuevo participante se deberá introducir nombre, edad y sexo, y lanzar la petición al servidor mediante la pulsación de algún botón, provocando el inicio del proceso del servidor.

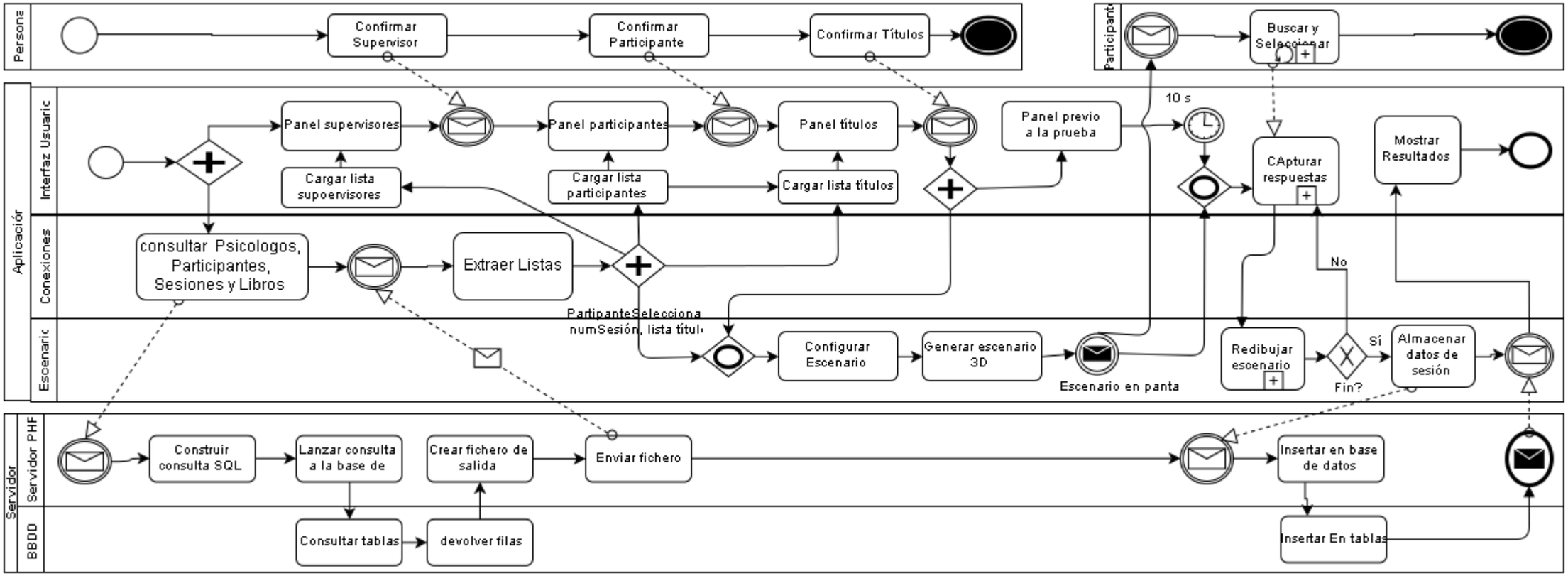


Figura 5-10: Diagrama *BPMN* del proceso de realización de una nueva sesión de la tarea de decisión léxica de un participante existente. Se muestra la evolución de las peticiones al servidor, y la interacción de la aplicación con el supervisor y el participante. Se puede observar que durante la realización de la prueba no se realizan conexiones con el servidor.

4.3.4 Interfaz gráfica de usuario

Para la creación de la interfaz de usuario, es decir, para la creación de las diferentes pantallas que presentará la aplicación es necesario en primer lugar diseñar grafos de navegación que clarifiquen las opciones de navegabilidad que se van a implementar. Éstos permitirán observar de forma rápida las formas de pasar de un apartado de la aplicación al siguiente así como observar las posibilidades de retroceso o cancelación de introducción de datos, entre otros aspectos.

La Figura 5-11 muestra un grafo cuyos nodos serán las secciones principales de la interfaz de usuario que agruparán varias pantallas de opciones: insertar nuevo, cancelar, acceder, etc.

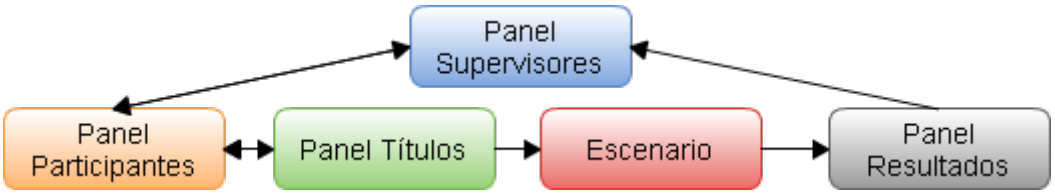


Figura 5-11: Grafo de **navegación general** en el que se presentan las principales secciones de la interfaz gráfica de usuario. El color del nodo identifica el color de fondo de todas las pantallas de una misma sección. Análisis y diseño de la interfaz gráfica de usuario.

A continuación vamos a mostrar los grafos de las diferentes secciones de la interfaz de usuario.

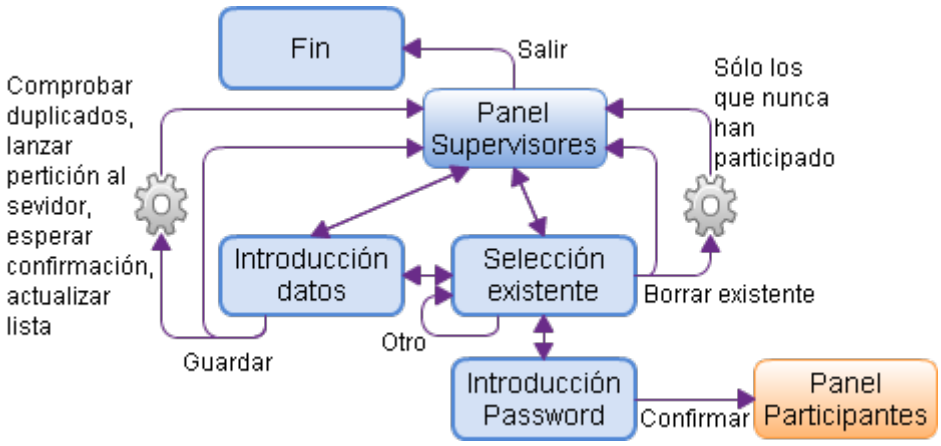


Figura 5-12: Grafo de navegación de la sección **Panel Supervisores**. Análisis y diseño de la interfaz gráfica de usuario.

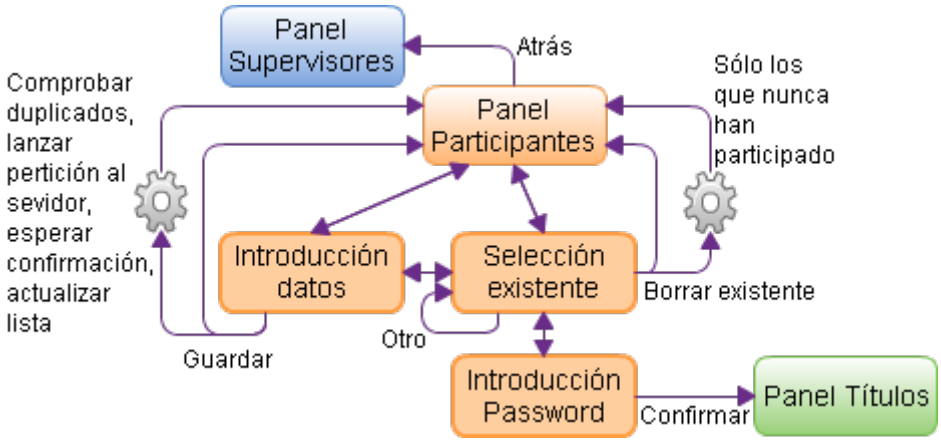


Figura 5-13: Grafo de navegación de la sección **Panel Participantes**. Análisis y diseño de la interfaz gráfica de usuario.

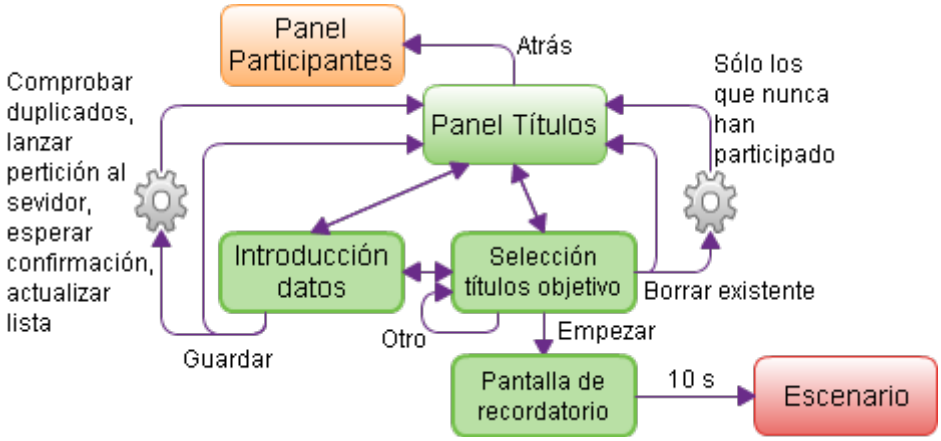


Figura 5-14: Grafo de navegación de la sección **Panel Títulos**. Análisis y diseño de la interfaz gráfica de usuario.

Para evitar que el usuario se distraiga durante la prueba en la sección **Escenario** se deshabilitará la interfaz gráfica de usuario, activándola únicamente cuando la sesión haya finalizado tras cumplirse las N capturas.

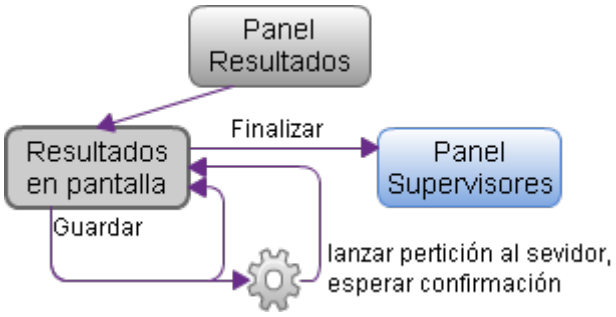


Figura 5-15: Grafo de navegación de la sección **Panel Resultados**. Análisis y diseño de la interfaz gráfica de usuario.

4.4 Prototipo implementado

Tras haber diseñado la aplicación entramos en la fase de implementación en la que se han empleado varias herramientas con las que se programaron las diferentes funcionalidades de los componentes que permiten la correcta ejecución de la aplicación en diferentes dispositivos y la comunicación con la base de datos.

A continuación mostramos una lista de las principales **herramientas software empleadas**:

- *Script* implementados con lenguaje *PHP* para atender las peticiones al servidor.
- Consultas (consultas, inserciones, borrados y actualizaciones) en la base de datos escritas con lenguaje *MySQL*.
- Fichero de resultados de la petición al servidor escrito con lenguaje *XML*.
- Base de datos creada con *PHPMYADMIN*.
- Aplicación implementada con lenguaje en *C#*.
- Gestión del escenario mediante código empleando métodos de construcción, destrucción, y modificación de algunas primitivas 3D de *Unity®*.
- Imágenes de texturas, mapas de normales, y materiales retocados con *Unity®*.
- Listas dinámicas en las que se almacenan los datos descargados.
- Programación orientada a objetos para construir las clases en las que se almacena la información propia del experimento (clase estantería, clase repisa, clase libro, etc.).

4.4.1 Escenario del juego serio

El escenario del juego serio de la aplicación posee todas las características descritas en el apartado "Escenario del juego serio" del apartado "Análisis y diseño" salvo algunas restricciones o modificaciones debidas a inconvenientes encontrados, como por ejemplo el efecto de *aliasing* que se presenta cuando el texto 3D de *Unity®* se encuentra lejos de la cámara. Por lo que hemos tenido que realizar las siguientes **restricciones** para que el escenario se observe correctamente:

- La cámara enfoca únicamente dos repisas para que todos los libros, títulos, texturas, mapas de normales y colores sean perfectamente visibles. El objetivo es que todos los libros estén a la vista para no entorpecer el tiempo de reacción.
- El giro de la cámara se limita a 10 grados a izquierda o derecha.

- Como resultado del problema del *aliasing* no se contempla la mesa con las siluetas de los libros de recordatorio ni los cajones de estantería ya que para enfocar todo la cámara tendría que alejarse más.
- El número máximo de libros por repisa se limita a 8 debido a que debe haber una separación entre libros y al haber demasiados se harían más estrechos.
- El número máximo de caracteres de los títulos colocados en las tapas se limita a 25 ya que el texto 3D de las tapas que simula los títulos se re-escala en función del número de caracteres del título para ajustarse a las dimensiones de la tapa, por lo que a más caracteres más se reduciría el tamaño haciéndose menos legible que el resto.
- La generación del escenario según determinados parámetros (número de repisas, número de libros, conjunto de títulos, etc.) se realiza mediante código para lo cual se ha diseñado una jerarquía de clases que permite con un solo método al que le pasamos los parámetros, construir todo el escenario.

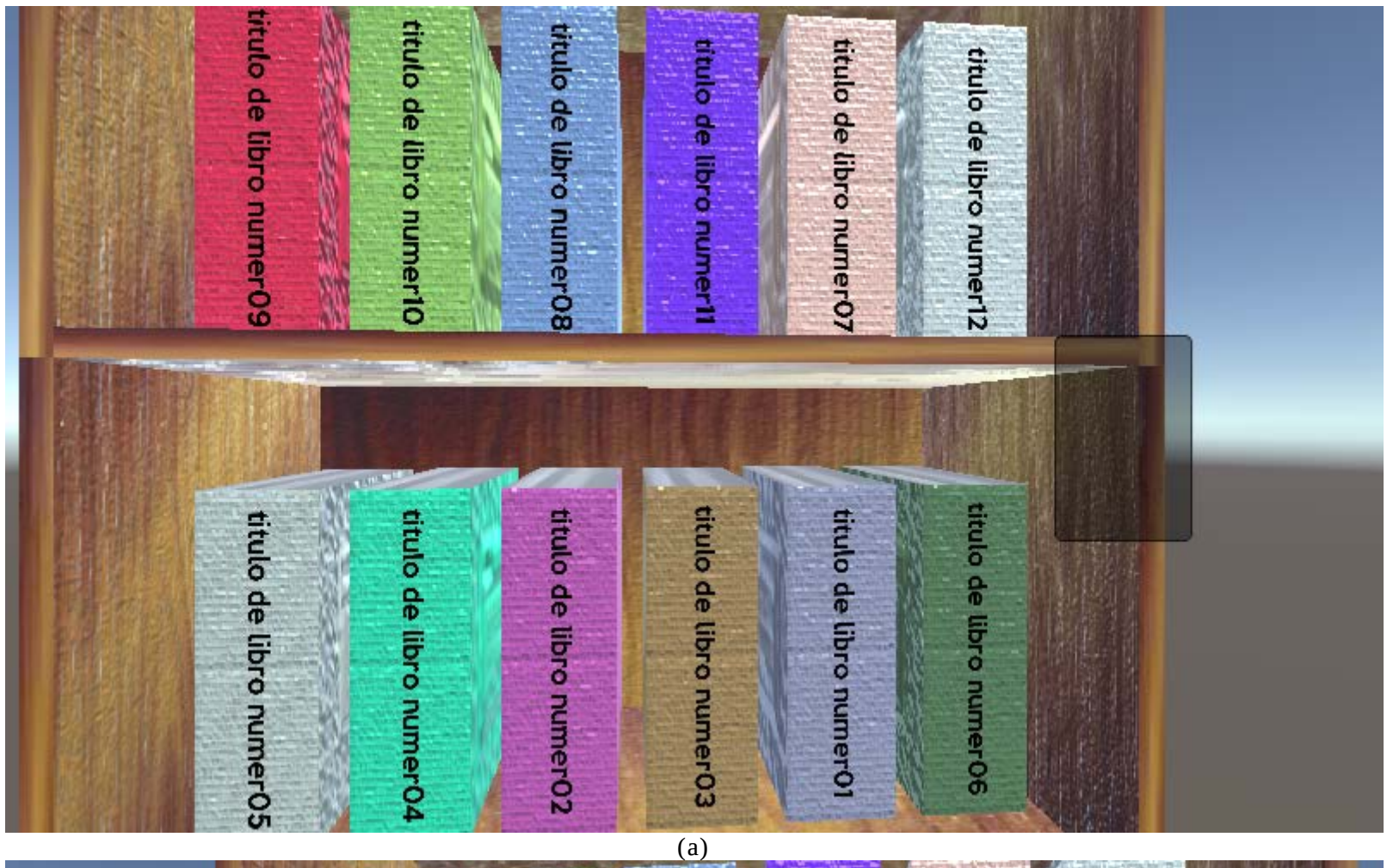
Un ejemplo del aspecto visual que presenta el escenario podemos observarlo en la Figura 5-16.

El funcionamiento del **control de la cámara** contiene los siguientes puntos:

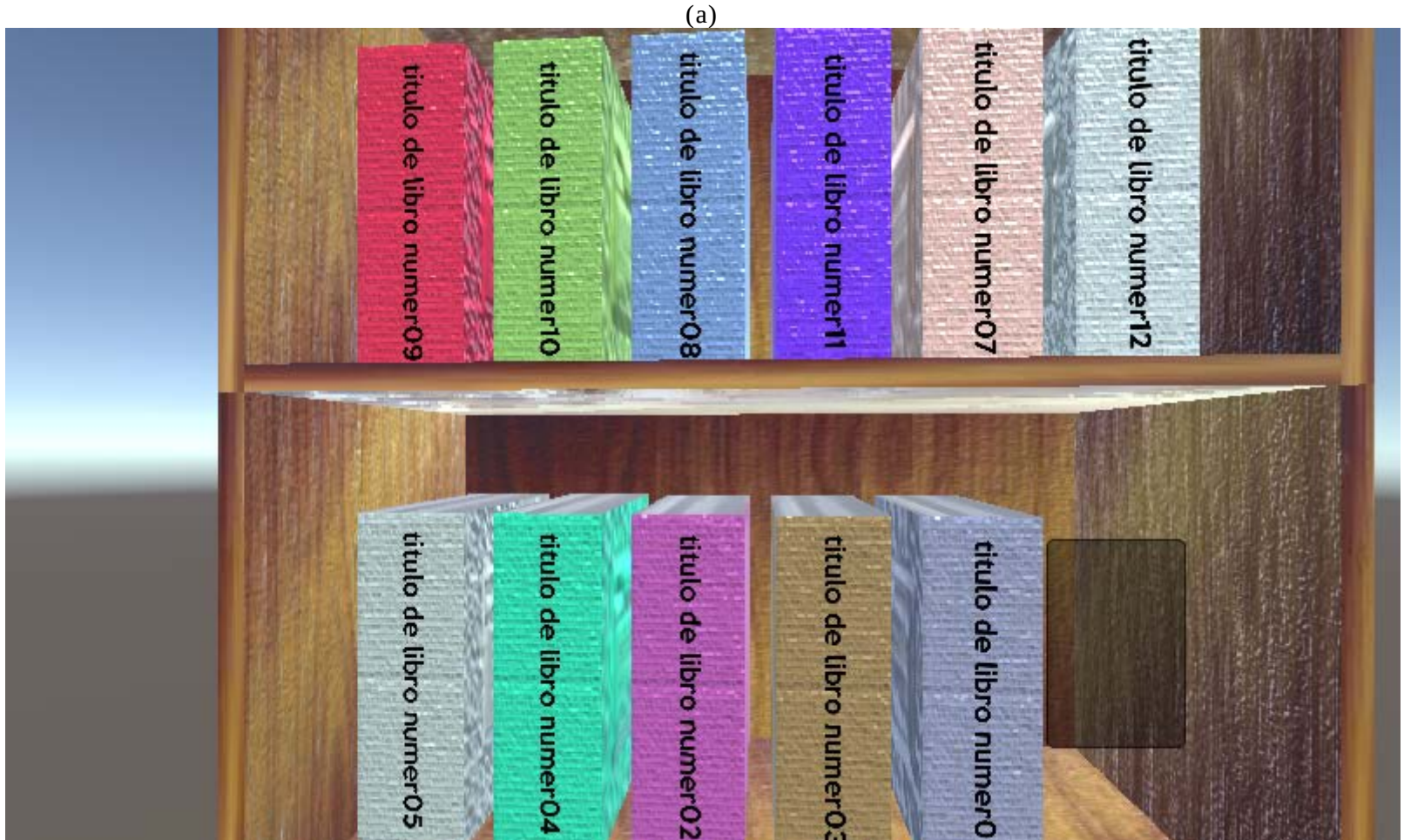
- El movimiento de la cámara sólo se activa tras haber pasado un tiempo predefinido (alrededor de 400 ms) una vez que se ha pulsado el botón. Esto intenta evitar que se mueva la cámara por error al seleccionar un libro.
- Tras activarse el movimiento la cámara responde al movimiento a izquierda o derecha del ratón.
- El movimiento a izquierda o derecha se basa en un vector de dirección entre 2 posiciones del ratón en diferentes instantes.
- El movimiento se desactiva cuando se suelta el botón.
- Durante el movimiento la cámara se mueve de forma constante a izquierda o derecha con un *step* en el ángulo de giro.
- No hay movimiento vertical.

Para la construcción de los objetos del escenario se crearon estructuras de datos que encapsularan diferentes elementos. Por ejemplo para crear un libro se creó una estructura que incluyera el cubo base (un *Game Object*), los planos laterales, el texto3D, los planos en los que se coloca la textura, las texturas, los mapas de normales, la cadena de caracteres que se muestra en el texto 3D, las dimensiones de cada objeto, etc.

Cada una de estas clases posee métodos de construcción y destrucción así como para posicionar elementos dentro de su área lo cual permite colocar por separado los libros de diferentes repisas, las propias repisas, etc., así como generar aleatoriedad (espacio interno de la repisa), cambiar colores, etc. Para aclarar mejor la organización jerárquica



(a)



(b)

Figura 5-16: Captura de pantalla del escenario del juego serio de la aplicación implementada. Parámetros establecidos para esta captura: 6 libros por repisa, 2 repisas, conjunto de 12 títulos. (a) Se ha girando la cámara hasta su límite derecho. (b) Se ha girado la cámara hasta su límite izquierdo y se ha seleccionado un libro provocando su desaparición. En ambas el *feedback* visual de la recepción de los clics del ratón se observan con un cuadro gris semitransparente.

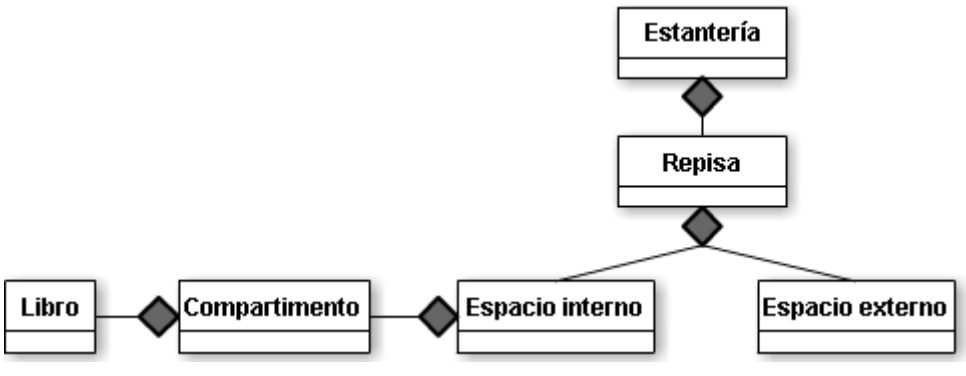


Figura 5-17: Diagrama de clases para la creación, destrucción y modificación de los elementos del escenario del juego serio de la aplicación implementada. Para reconfigurar el escenario basta con llamar al método de destrucción de la estantería y llamar al método de construcción con los nuevos parámetros.

de las clases creadas en la Figura 5-17 mostramos un diagrama de clases para la generación del escenario de la que destacamos que la clase *Estantería* es la que recibe los valores de los parámetros de la prueba y la que se comunica con el resto de clases.

La configuración del experimento, es decir, la configuración de las posiciones, colores, conjunto de títulos, conjunto de títulos objetivo, etc., se abordó mediante la creación de métodos en las clases creadas para construir el escenario que aparecen en la figura anterior. Sin embargo se debe aclarar una de las funcionalidades más destacables que es la generación de la aleatoriedad.

Como sabemos debemos mantener la consistencia en las realizaciones de las pruebas entre los diferentes participantes. Para ello se debe repetir la misma configuración del escenario con diferentes participantes. Sin embargo para cada participante se debe generar una configuración diferente en cada sesión.

Esta tarea se abordó con la clase *Random* que describimos en el apartado "Control con C#" la cual nos permite generar **números aleatorios** en un intervalo dado. En este caso además hemos hecho uso de otra de las funcionalidades que posee esta clase, *Random.seed* que da la posibilidad de usar una semilla con la que generar pseudo-aletoriedad. Es decir una semilla que permite repetir la generación de los mismos números aleatorios. Con esto podemos repetir la configuración del escenario para diferentes participantes dejando el comportamiento de la configuración de los libros en la estantería con un comportamiento pseudo-aletorio.

Se decidió que el valor de la **semilla** se correspondiera con el número de sesión de la última sesión realizada por un determinado participante. Esto permite que se repita la misma configuración pseudo-aleatoria para todos los participantes que realicen el mismo número de sesión.

Dada la jerarquía de clases creada para la construcción del escenario dicha semilla debe pasar por las diferentes clases para, por ejemplo, distribuir los libros en una misma repisa, otorgar un color aleatorio a un libro, colocar los títulos en las tapas, etc.

Para aclarar la manera en la que se transmite la semilla por las diferentes clases mostramos la Tabla 5-1.

```
GestionsEscenario.cs
|
+---generarEscenario(semilla, ...);
      |
      +---Estanteria(semilla, ....);
         |
         +---Repisa(semilla, ...);
             |
             +---eInterno.crearLibros(semilla + idRep, ...);
             +---eInterno.randomIndicesCompartimentos(semilla, ...);
             +---...
```

Tabla 5-1: Transmisión de la semilla para construir la estantería en función del número de sesión. La pseudo-aleatoriedad que nos da el uso de una semilla permite repetir la configuración (colores, posiciones, títulos, etc.) del escenario.

Así mismo se debe tener en cuenta que la colocación de las cadenas de caracteres (títulos) en el texto 3D de las tapas se hará en el momento en el que se haya terminado de descargar y extraer los títulos. En ese caso también se usa la semilla para distribuir los títulos aleatoriamente por repisa y por estantería. Se deja este último paso para el final puesto que, generalmente, la descarga tarda más de lo que tarda la aplicación en generar el resto de elementos del escenario. Hasta que se realice este último paso los títulos permanecen con una cadena de caracteres por defecto.

4.4.2 Interacción con el participante

La implementación de la gestión de la interacción de la aplicación con el participante incluye dos apartados: la correspondencia entre posición 2D del clic y el objeto 3D de las escena, y la captura de los datos (tiempos de reacción, orden de clics y aciertos).

Para encontrar la **correspondencia entre el clic 2D y el objeto 3D seleccionado** hemos hecho uso de la técnica de trazado de rayos. En *Unity®* disponemos de la clase *Physics* que nos permite trazar un rayo entre objetos al cual le podemos decir que trace un rayo desde un punto de origen con el método *Raycast*. A éste le podemos indicar un punto de origen, un vector de dirección y una distancia máxima tras lo cual nos retornará un valor *Bool* indicando si el rayo trazado ha colisionado con algún objeto.

Para que este trazado de rayos nos permita colisionar con un libro ha sido necesario incluir en la clase *Libro*, mencionada en el apartado anterior, un objeto de tipo *Collider*[34] que abarque el área de todos los elementos del libro de tal forma que al colisionar el rayo permita identificar el libro, calcular el tiempo de reacción, almacenar los datos de la selección (TR, libro seleccionado, orden del clic, número de compartimento, número de repisa), actualizar el número de libros disponibles, y destruir el libro seleccionado.

Para explicar mejor el funcionamiento del trazado del rayo mostramos un ejemplo en la Figura 5-18 que muestra un ejemplo de colisión de un rayo trazado desde la cámara con un libro de la estantería cuando el participante pincha o toca la pantalla.

En cuanto a las **capturas de los tiempos de reacción** hemos hecho uso del método *realTimeSinceStarUp* de la clase *Time* que habíamos comentado en el apartado "Control con C#". Las capturas se realizan conforme a lo que describimos en el sub-apartado "Interacción con el participante" del apartado "Análisis y diseño". Podemos observar el proceso interno en la Figura 5-7 ya mostrada en dicho sub-apartado, que al realizarse siempre igual permite distribuir uniformemente los posibles retardos debidos al hardware.

[34] Un objeto *Collider* en *Unity®* es un componente que se añade a un *Game Object* y que permite capturar eventos de colisión del *Game Object*. Se trata de una forma geométrica no visible en la escena final que delimita el área de colisión con el resto de *Collider* de otros objetos de la escena pudiendo ser ésta mayor o inferior al área de dicho objeto.

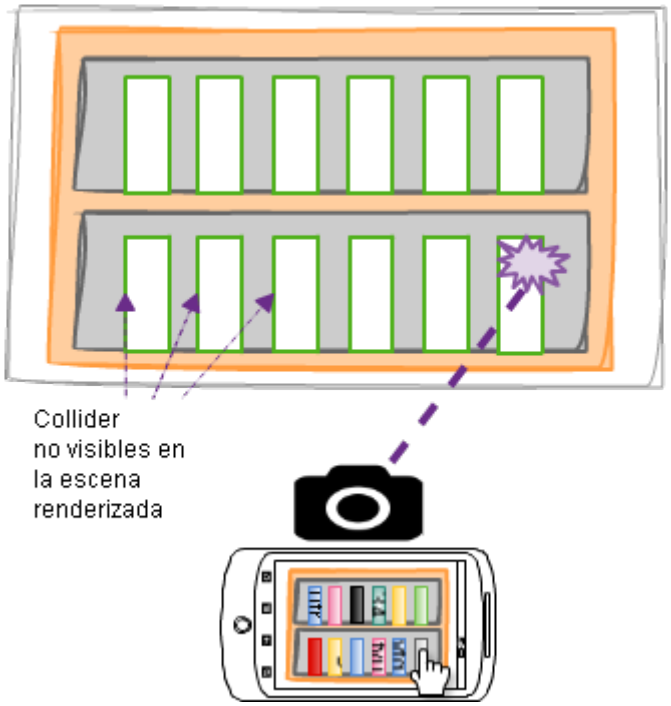


Figura 5-18: Ejemplo del uso del método *Raycast* para identificar un *Libro* mediante el trazado de un rayo desde la cámara y la colisión con el *Collider* del *Libro*. Tras la colisión se almacenan los datos oportunos y se destruye el libro.

4.4.3 Comunicación con la base de datos

En la implementación de la gestión de las conexiones al servidor se abordaron tres apartados: creación de la base de datos con *PHPMYDMIN*, creación de *script PHP* que atienda las peticiones llegadas desde la aplicación y devuelva un fichero *XML*, y creación de funciones específicas en la aplicación para consulta, borrado, actualización y extracción de los datos recibidos en el fichero *XML*.

A continuación mostramos un diagrama que representa la manera en la que viaja la información entre aplicación, el servidor y la base de datos cuando se realiza una petición al servidor:

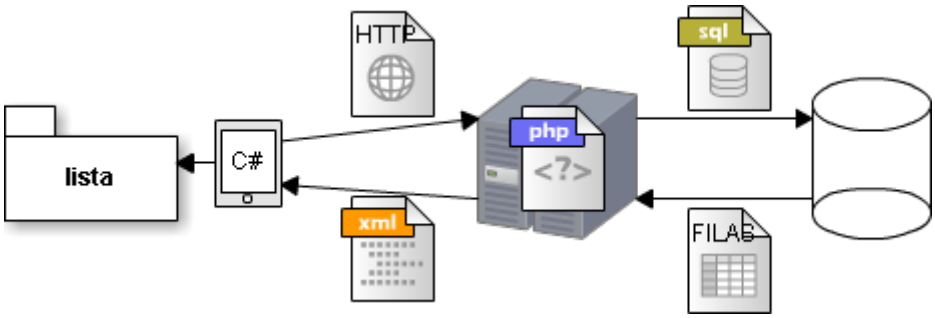


Figura 5-19: Diagrama que representa la manera en la que viaja la información entre aplicación, el servidor y la base de datos cuando se realiza una petición al servidor.

La **base de datos** de la primera versión de la plataforma se creó con el motor *InnoDB* que nos proporciona *PHPMYADMIN* por defecto. Este motor de base de datos permite establecer restricciones a la inserción, actualización o borrado. En nuestro caso hemos hecho uso de estas restricciones para impedir borrar libros, supervisores, participantes y clics, que ya hayan formado parte de la realización de una sesión. Así mismo se ha establecido que los identificadores de cada tabla se autoincrementen de forma automática con cada inserción de nueva fila. Esto nos permite dejar que el motor de base de datos gestione dichos valores de los identificadores y consultarlos desde la aplicación. La Figura 5-20 nos muestra la estructura de las tablas y la organización final de la base de datos.

Se puede observar en la base de datos final que se ha incorporado una tabla denominada *uso*. Ésta fue incorporada con la intención de poder almacenar la configuración del escenario de cada sesión de un participante. De esta forma podríamos más adelante realizar un análisis del comportamiento de los participantes para observar si la organización influye de alguna manera en las respuestas del participante. Observaremos en el apartado de resultados que efectivamente con dicha tabla podemos determinar el comportamiento recurrente que presentaban la mayoría de participantes al iniciar la prueba.

A continuación vamos a resumir algunos detalles que permitirán entender mejor el comportamiento de la base de datos de la figura Figura 5-20 que hemos mencionado:

- Ejemplo de **multiplicidades**: un registro (fila) de la tabla *sesion* sólo puede relacionarse con cero o un registro de la tabla *participante*, un *participante* puede relacionarse con cero, uno o varios registros de la tabla *sesion*.
- Se otorga **índice único** a los siguientes atributos: *nombre* de la tabla *psicologo*, *nombre* de la tabla *participante*, *fechaYHora* de la tabla *sesion*, y *titulo* de la tabla *libro*.
- El índice único de un atributo no permite que haya varios registros (filas) con el mismo valor en dicho atributo. Por ejemplo la tabla *libro* no puede tener dos valores de *titulo* repetidos.
- Se establece la **restricción de borrado** para los siguientes atributos: {*id_psicologo*, *id_participante*} de la tabla *sesion*, {*id_sesion*, *id_libro*} de la tabla *pick*, {*id_sesion*, *id_libro*} de la tabla *uso*.
- La restricción de borrado *ONDELETE RESTRICT* otorgada a un atributo no permite que se borren registros de la tabla a la que hace referencia dicho atributo. Por ejemplo: No se puede borrar un *participante* que esté siendo referenciado desde la tabla *sesion* mediante la clave ajena *id_participante*.
- Las **claves primarias** (*id* en todas las tablas) tienen **autoincremento**.

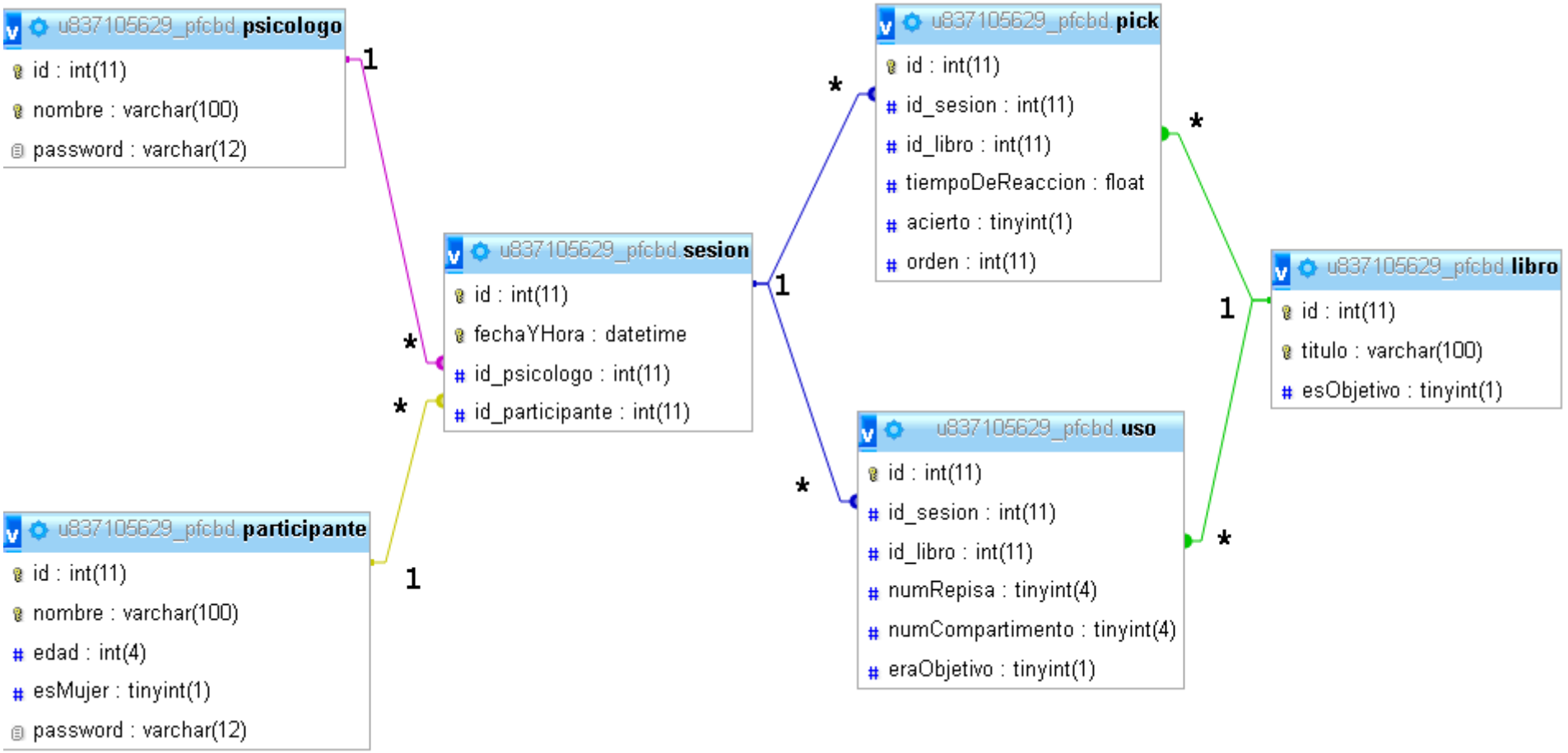


Figura 5-20: Diagrama de tablas de la base de datos para almacenar la información referente a las sesiones de la tarea de decisión léxica realizadas por los participantes en la aplicación implementada.

Para las **peticiones** al servidor hemos empleado tres clases que nos facilitan esta labor: *Coroutine*, *WWW*, y *XMLDocument*.

La clase *Coruotine* nos sirve para crear hilos de ejecución paralelos al flujo de ejecución del motor de *Unity®* de tal forma que podemos realizar funcionalidades como esperar la descarga de un fichero sin tener que detener el resto de tareas que realiza la aplicación. Dentro de una función ejecutada como co-rutina se puede congelar el flujo de ejecución del hijo empleando la función *yield* a la que le podemos especificar el número de segundos de espera. De esta forma eliminamos la dependencia de los retardos de la transmisión.

La clase *WWW* nos sirve para lanzar las peticiones al servidor y recibir respuestas del mismo. Ésta permite incluir una *URL* a la que se debe realizar la conexión. En nuestro caso incluimos en dicha *URL* la dirección del *script PHP* que atiende las peticiones de inserción de una determinada tabla. Para ello hacemos uso del paso de parámetros por la *URL*. Esto significa que podemos construir la *URL* como una cadena de caracteres en la que incluimos, además de la dirección del *script PHP* al que lanzar la petición, los campos y los valores de dichos campos para que el servidor los recoja. Para clarificar este funcionamiento mostramos la Tabla 5-2.

```
insertarEnBD(
  servidor.url + "?nombre=" + nuevo + "&edad=" + nuevaEdad.ToString() +
  &esMujer=" + nuevoEsMujer.ToString() + "&password=" + nuevaPass
);
```

Tabla 5-2: Ejemplo del paso de parámetros y valores al servidor a través de la URL desde la aplicación implementada.

La clase *XMLDocument* nos permite crear una estructura de datos con opciones de manipulación de nodos *XML*, es decir podemos extraer el valor (entero, cadenas de caracteres, *boolean*, etc.) de un determinado nodo presente en la estructura *XML* creada. A esta clase le podemos pasar como parámetro de construcción una cadena de caracteres que puede ser todo el texto devuelto por el servidor. Es decir el servidor habrá construido una cadena de caracteres en la que figura toda la estructura del fichero *XML* resultado de la petición.

En la Tabla 5-3 podemos ver un ejemplo del formato que se ha establecido para los ficheros *XML* (contenido en una sola cadena de caracteres) que **retorna** el servidor. El formato para el resto de peticiones es similar.

```
<?xml version="1.0" encoding="UTF-8" ?>
<Participantes>
        <Participante id="1" nombre="gett" edad="23" esMujer="1" password="asdf"/>
        <Participante id="2" nombre="dact" edad="23" esMujer="0" password="5fde"/>
        <Participante id="3" nombre="hctb" edad="41" esMujer="1" password="1234"/>
        <Participante id="4" nombre="srtc" edad="47" esMujer="0" password="5asj"/>
</Participantes>
```

Tabla 5-3: Ejemplo del formato del fichero *XML* que retorna el servidor tras una petición de consulta a la base de datos de los registros de la tabla *participante*. En la tabla se observan cuatro nodos *Participante* y cada uno de éstos con cinco atributos: *id*, *nombre*, *edad*, *esMujer*, y *password*. Aplicación implementada.

Una vez descargada la información el siguiente paso es extraer la información para convertirla a un formato que sea manipulable por la aplicación. Para ello hemos creado clases apropiadas para recoger la información proveniente de cada tabla y que está contenida en los nodos del *XML* que retorna el servidor. **Estas clases** sirven para crear listas dinámicas cuyos elementos sean de los tipos de estas clases. Éstas no poseen estructura jerárquica pero facilitan la extracción de los atributos de los nodos *XML*. Para más detalles sobre la creación de listas dinámicas podemos acudir al apartado “Control con C#” de la “Evaluación de herramientas software”.

Las atributos de las seis primeras clases creadas responden fielmente a los atributos de las seis tablas de la base de datos que se puede observar en la Figura 5-20 ya mostrada anteriormente.

La séptima clase se creó para almacenar en ésta valores que permitan controlar la conexión al servidor. Ésta incluye una **máquina de estados para saber el estado de la petición**, un valor del período de tiempo con el que se debe consultar el estado, y la dirección *URL* de cada petición (consulta, inserción y actualización). En la Figura 5-21 podemos observar un diagrama de transición de estados para controlar el ciclo de vida de una petición al servidor.

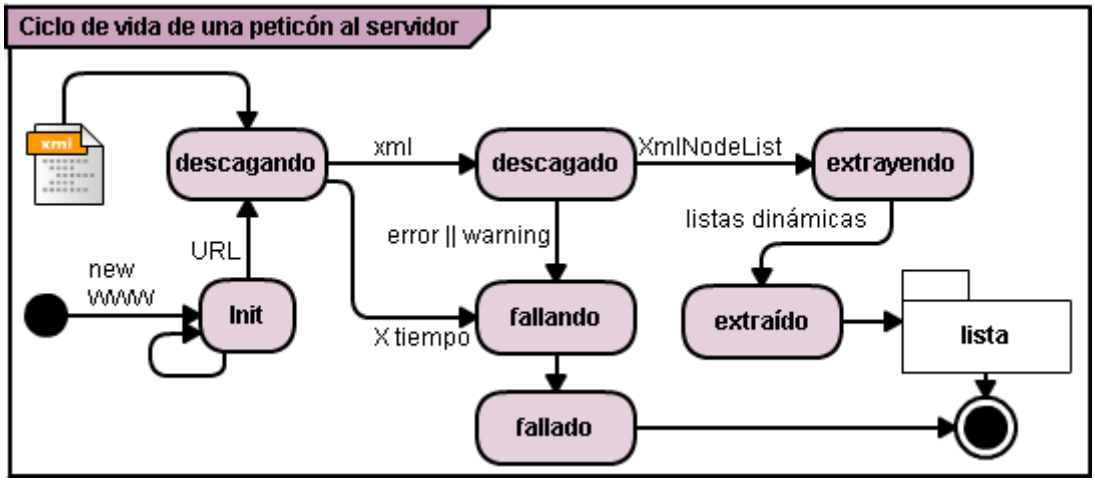


Figura 5-21: DTE de la máquina de estados que controla el ciclo de vida de una petición al servidor realizada internamente en la aplicación implementada.

4.4.4 Interfaz gráfica de usuario

Para la creación de la interfaz gráfica de usuario hemos cumplido las características detalladas en el sub-apartado “Interfaz gráfica de usuario” del apartado “Análisis y diseño”. Las clases empleadas de la clase *GUI* para esta interfaz dinámica han sido:

- *Button* para crear botones dinámicos en los que introducir los *alias* de los participantes, psicólogos, y títulos de los libros.
- *Button* para crear los botones de guardar, atrás, aceptar, etc.
- *TextField* para la entrada de datos.
- *Togle* para activar y desactivar el *scroller* horizontal de edades.
- *GUILayout* y *ScrolView* para generar un *scroller* vertical de botones para representar la lista de supervisores, participantes y libros ya almacenados.
- *GUISkin* en el que colocamos varios materiales a ser aplicados a los fondos, colores de relleno, colores de las letras, etc. Esto sirve para que los elementos de una misma pantalla tengan una gama de colores similar, y que el usuario pueda distinguir en qué pantalla se encuentra mediante dicha gama.
- Una máquina de estados para controlar el flujo de navegación y la interacción con el usuario, que responde fielmente a los grafos de navegación descritos en el sub-apartado “Interfaz gráfica de usuario” del apartado “Análisis y diseño”.

Un aspecto relevante a destacar es que los sistemas de coordenadas de los diferentes elementos que intervienen en la interfaz de usuario tienen orígenes de referencia diferentes. Por ello es necesario realizar reajustes con operaciones matemáticas para colocar correctamente los cuadros de los botones, de los fondos, del *scroller,* etc., en la pantalla según las dimensiones de éstos. En la Figura 5-22 mostramos un diagrama que muestra un ejemplo de readaptación de los orígenes de referencia de los sistemas de coordenadas de la pantalla, el ratón y el *GUI.Box* para mostrar el cuadro gris semitransparente de *feedback* visual que se muestra cuando el participante hace clic en un libro. Al origen de referencia **(0,0)** de la caja (*Box, Button, etc.*) del ejemplo se le ha aplicado las operaciones matemáticas necesarias para la correcta colocación centrada con la posición del ratón. El mismo tratamiento se realiza cuando se posicionan el resto de elementos de la interfaz gráfica teniendo en cuenta las dimensiones de la pantalla.

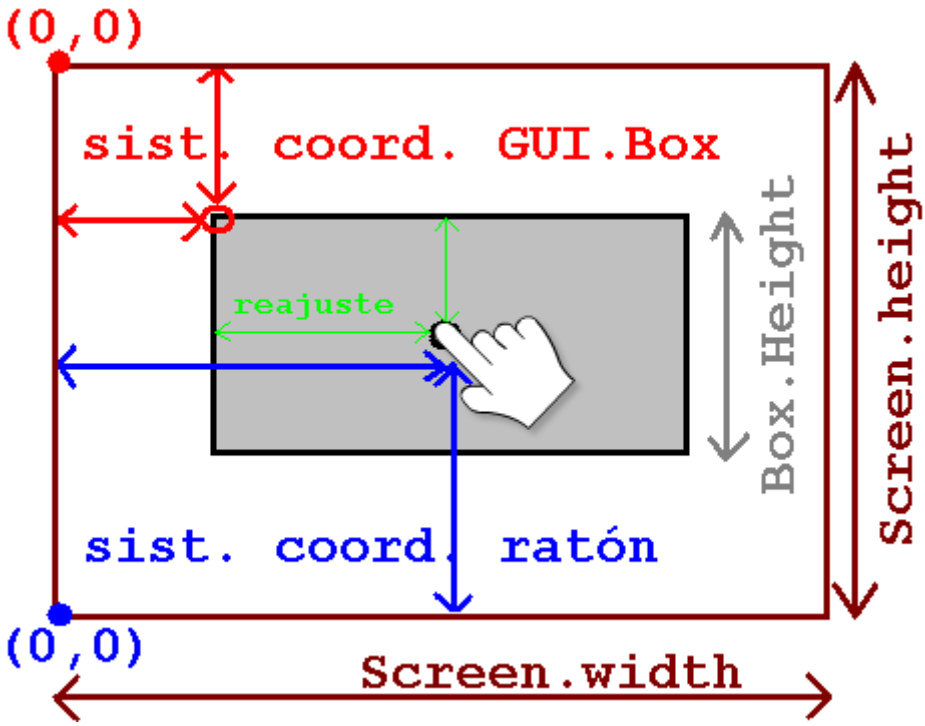
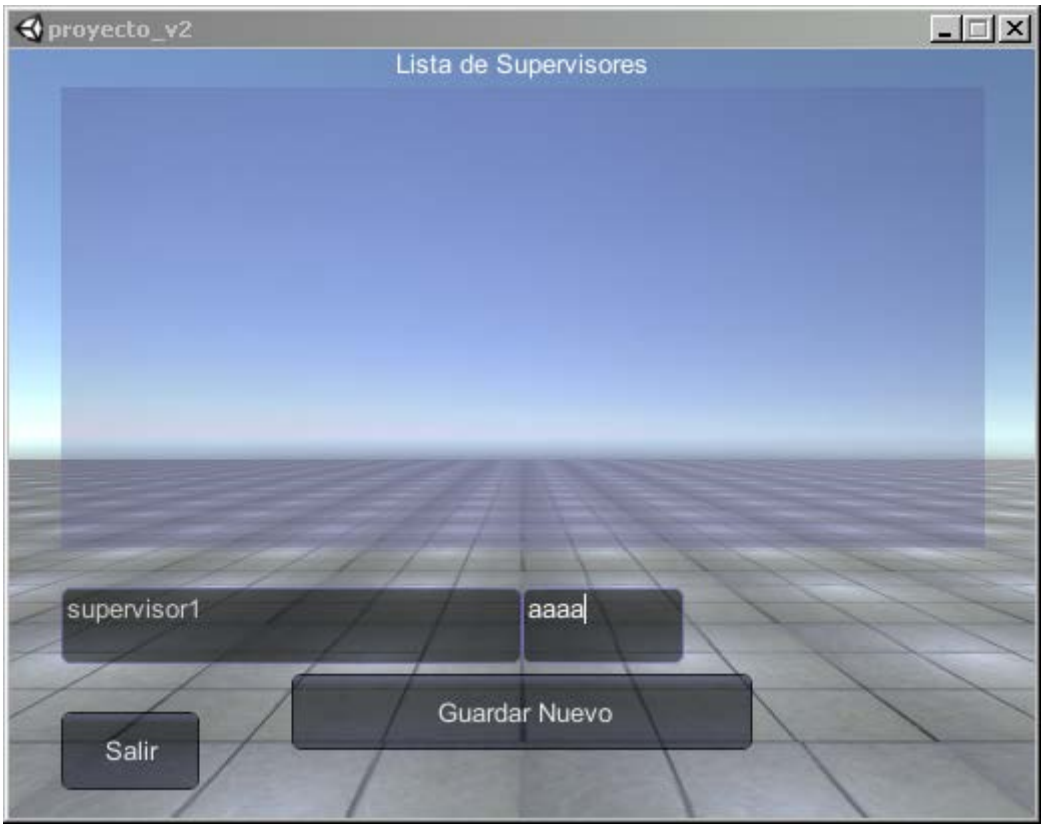


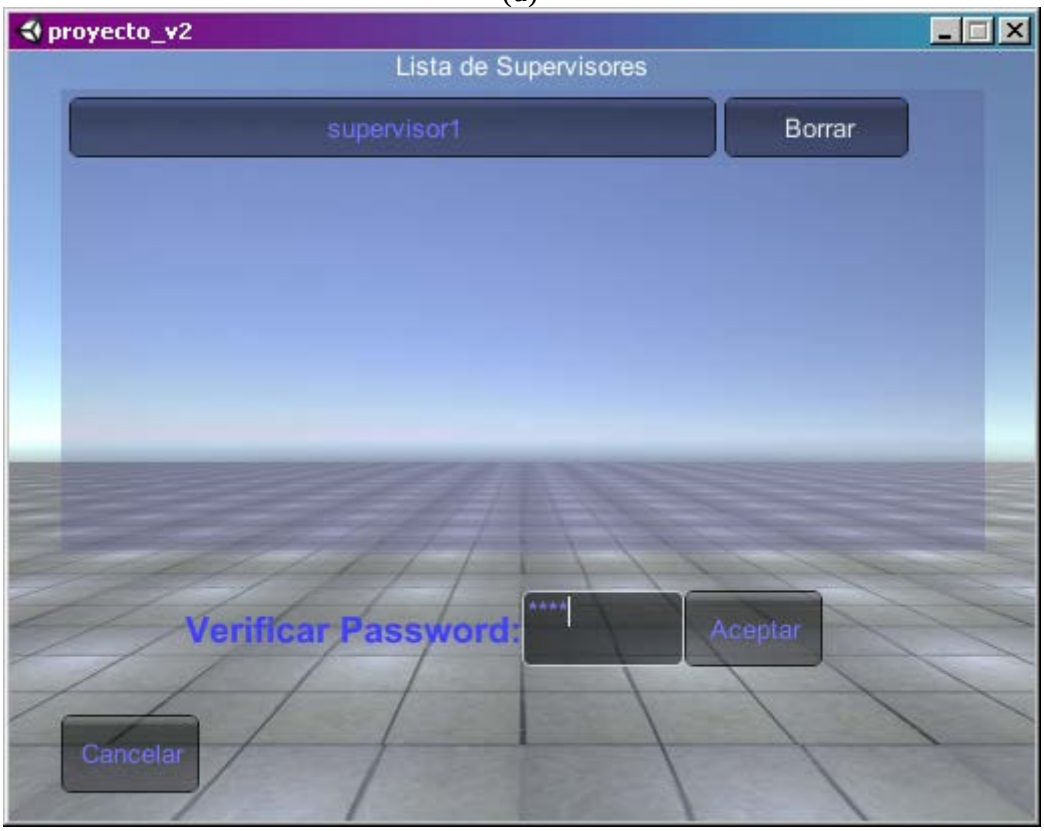
Figura 5-22: Diagrama que muestra un ejemplo de readaptación de los orígenes de referencia de los sistemas de coordenadas de la pantalla, el ratón y el *GUI.Box* para posicionar correctamente los elementos de la interfaz gráfica de usuario en la aplicación implementada.

La **navegabilidad** entre pantallas cumple todas las características detalladas en el sub-apartado “Interfaz gráfica de usuario” del apartado “Análisis y diseño”.

En cuanto al aspecto visual de las diferentes pantallas, las imágenes de la Figura 5-23 nos muestran un ejemplo del aspecto visual y la organización de los elementos de la interfaz en cada pantalla. El orden de presentación de las imágenes en dicha figura se corresponde con el orden de aparición de las pantallas cuando se realiza una sesión de un participante no existente.


proyecto_v2
Lista de Supervisores
supervisor1
aaaa
Guardar Nuevo
Salir

(a)


proyecto_v2
Lista de Supervisores
supervisor1
Borrar
Verificar Password:
****
Aceptar
Cancelar

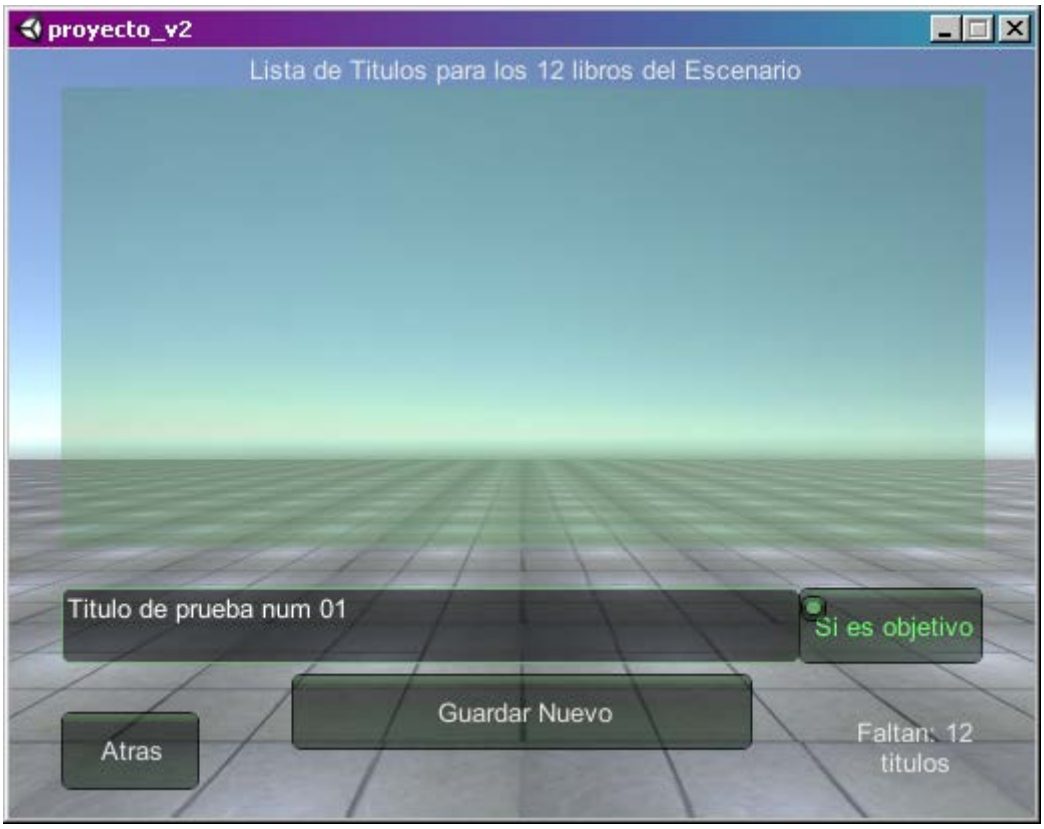
(b)


proyecto_v2
Lista de Participantes
participante1
aaaa
Hombre
Edad: 45
41
42
43
44
45
46
47
48
49
50
Atras

(c)


proyecto_v2
Lista de Participantes
participante1 45 Hombre
Borrar
Verificar Password:
****
Aceptar
Cancelar

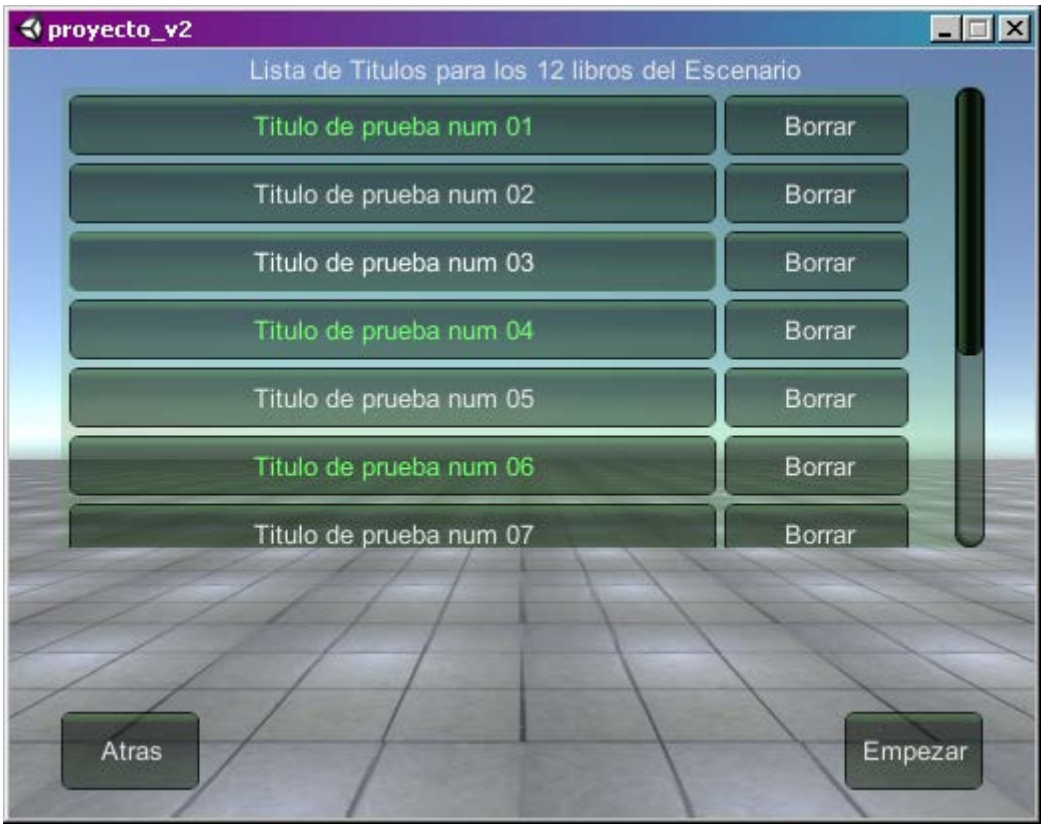
(d)


proyecto_v2
Lista de Titulos para los 12 libros del Escenario
Titulo de prueba num 01
Si es objetivo
Guardar Nuevo
Atras
Faltan: 12
titulos

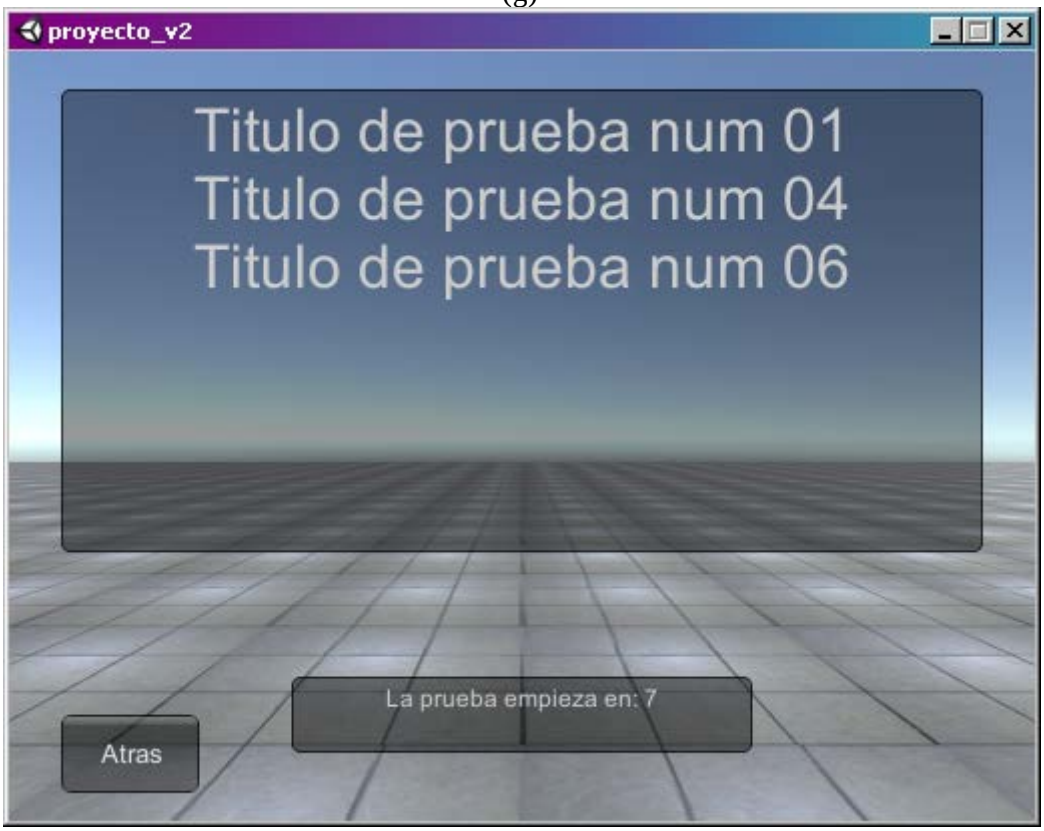
(e)


proyecto_v2
Lista de Titulos para los 12 libros del Escenario
Titulo de prueba num 01
Borrar
Titulo de prueba num 02
No es objetivo
Guardar Nuevo
Atras
Faltan: 11
titulos

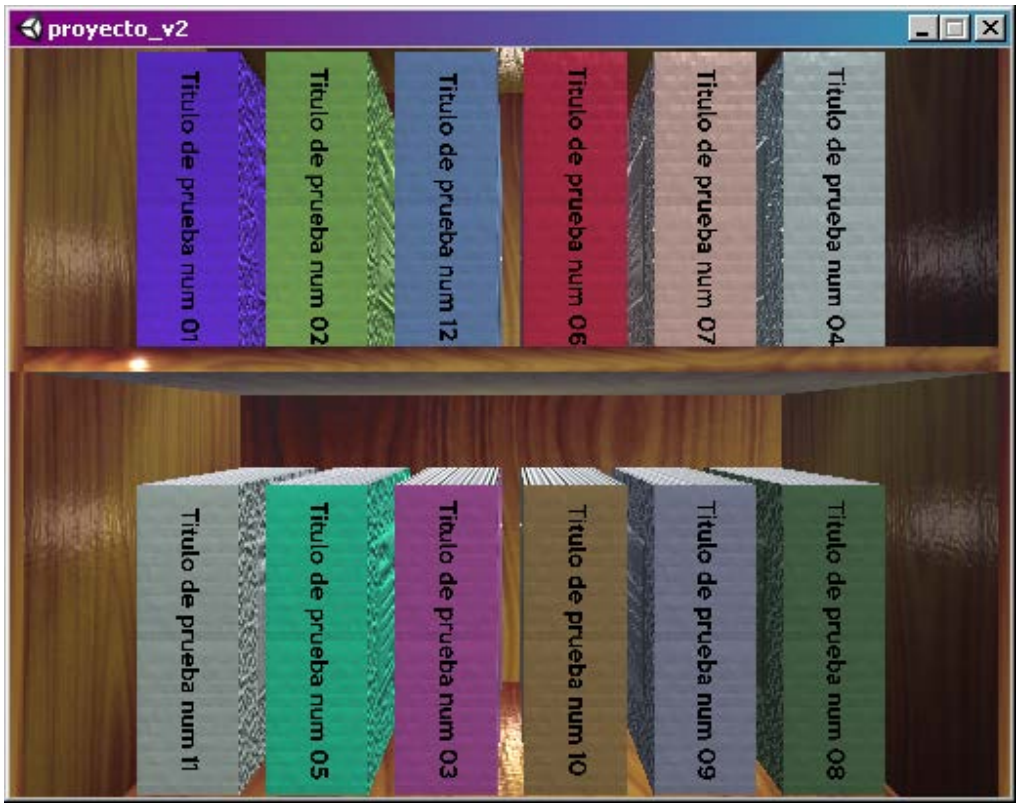
(f)


proyecto_v2
Lista de Titulos para los 12 libros del Escenario
Titulo de prueba num 01
Borrar
Titulo de prueba num 02
Borrar
Titulo de prueba num 03
Borrar
Titulo de prueba num 04
Borrar
Titulo de prueba num 05
Borrar
Titulo de prueba num 06
Borrar
Titulo de prueba num 07
Borrar
Atras
Empezar

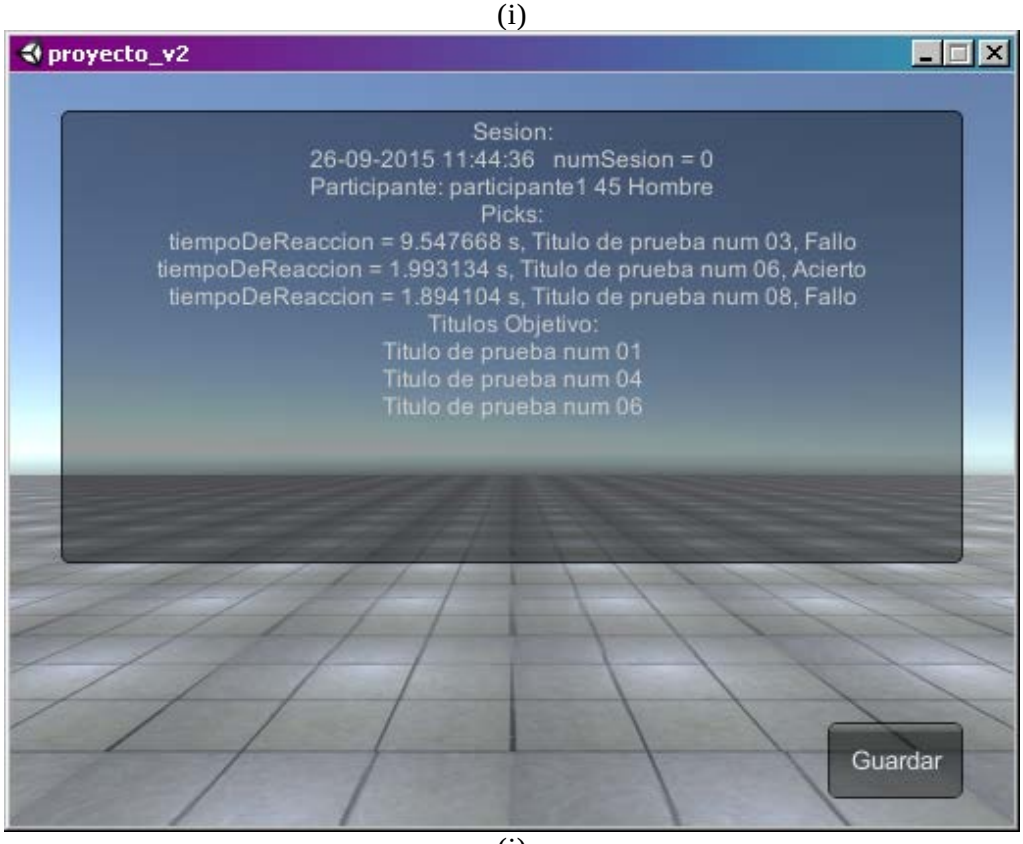
(g)


proyecto_v2
Titulo de prueba num 01
Titulo de prueba num 04
Titulo de prueba num 06
La prueba empieza en: 7
Atras

(h)



(i)



(j)

Figura 5-23: Capturas de pantalla que muestran de forma secuencial el aspecto de algunas de las pantallas de la interfaz gráfica de usuario de la aplicación implementada cuando se realiza una sesión de un usuario no existente. (a) Introducción de datos de nuevo supervisor. (b) Acceso de supervisor mediante contraseña. (c)Introducción de datos de nuevo participante. (d) Acceso de participante mediante contraseña. (e) Introducción del primer título. (f) Introducción de los siguientes títulos. (g) Lista de títulos completa. (h) Pantalla de recordatorio: 10 segundos. (i) Escenario. (j) Pantalla de resultados: guardar datos y finalizar. Al finalizar se vuelve a la pantalla (a).

Se puede observar un video de ejemplo de la realización de una sesión desde el principio hasta el final, así como un ejemplo de la recuperación de errores de conexión de la aplicación con el servidor disponible en el portal de videos *vimeo*[35].

4.4.5 Puesta en marcha y portabilidad

Tras la implementación del prototipo se puso en marcha la plataforma para validar que los tres componentes (aplicación, servidor y base de datos) se ejecutaban correctamente. Esta verificación no incluyó la realización de pruebas con participantes sino que se basó en ver el aspecto gráfico del escenario en diferentes dispositivos para comprobar el correcto enfoque de todos los elementos de la estantería en diferentes resoluciones de pantalla. Así mismo se pretendía encontrar los primeros inconvenientes derivados del uso de un servidor real y la exposición a los retardos debidos al tráfico de la red.

De dichos inconvenientes surgidos se pretendía encontrar soluciones y mejoras antes de pasar a la realización de experimentos con participantes reales.

La aplicación se ha exportado a las siguientes plataformas:

- *Windows* 32 y 64 bits.
- *Linux* 32 y 64 bits.
- *MacOS* 32 y 64 bits.
- *Android*.
- *Unity Web Player*.

La exportación de la aplicación a las plataformas[36] mencionadas pudo realizarse con el propio entorno de *Unity*® sin necesidad de la integración de bibliotecas de funciones adicionales, a excepción de la exportación a la plataforma *Android*. Para ésta es necesaria la incorporación de las bibliotecas correspondientes que permiten la compilación de la aplicación y la creación de un fichero *.apk* que sirve de instalador para dicha plataforma.

[35] Enlace al video de ejemplo de la realización de una sesión y la recuperación de errores de conexión de la aplicación implementada: <https://vimeo.com/138619493>. [Septiembre 2015]

[36] El siguiente tutorial muestra un video que explica algunos de los pasos para exportar la aplicación usando el panel de exportación *Build Settings* en *Unity*®: <http://unity3d.com/es/learn/tutorials/projects/space-shooter/building-the-game>. [Consultado: Septiembre 2015]

Android ofrece un kit de desarrollo de software[37] (*SDK* por sus siglas en inglés) gratuito para desarrolladores con el que se puede implementar aplicaciones destinadas a esta plataforma. El kit de desarrollo puede ser incorporado en diferentes entornos de desarrollo integrado (*IDE*) por lo que la incorporación a nuestro proyecto en *Unity®* ha sido relativamente ágil. Tras las descarga de la *SDK* tan sólo hay que indicarle la ruta en la que se ha descargado, configurar las propiedades de exportación, ajustar texturas, mapas de normales, el modo de renderizado y construir el fichero *.apk*.

Los dispositivos y sistemas operativos específicos en los que se ha comprobado la ejecución de la aplicación son los siguientes:

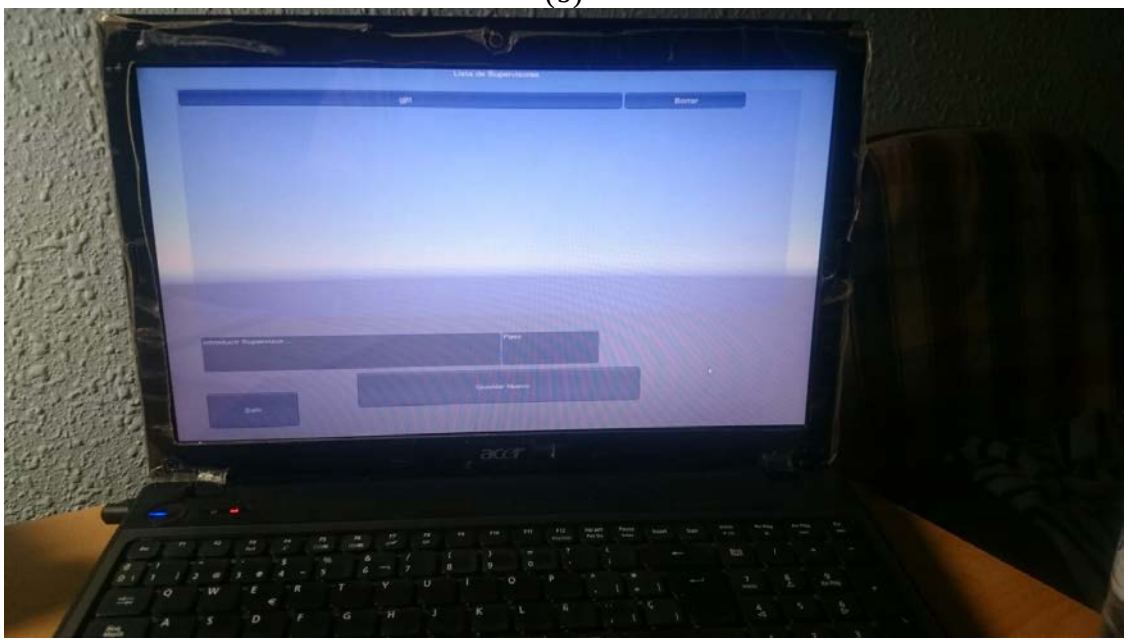
- Tableta *Airis OnePad x1100* con sistema operativo *Android 4.2*.
- *SmartPhone Samsung Galaxy S5* con sistema operativo *Android 4.4*.
- Ordenador portátil *Acer Aspire 5741* con sistema operativos *Windows 7* – 64 bits.
- Ordenador portátil *Mac Book Air* con sistema operativo *macOS* – 64 bits.

Las imágenes de la Figura 5-24 representan ejemplos de la ejecución en cada uno de los dispositivos.

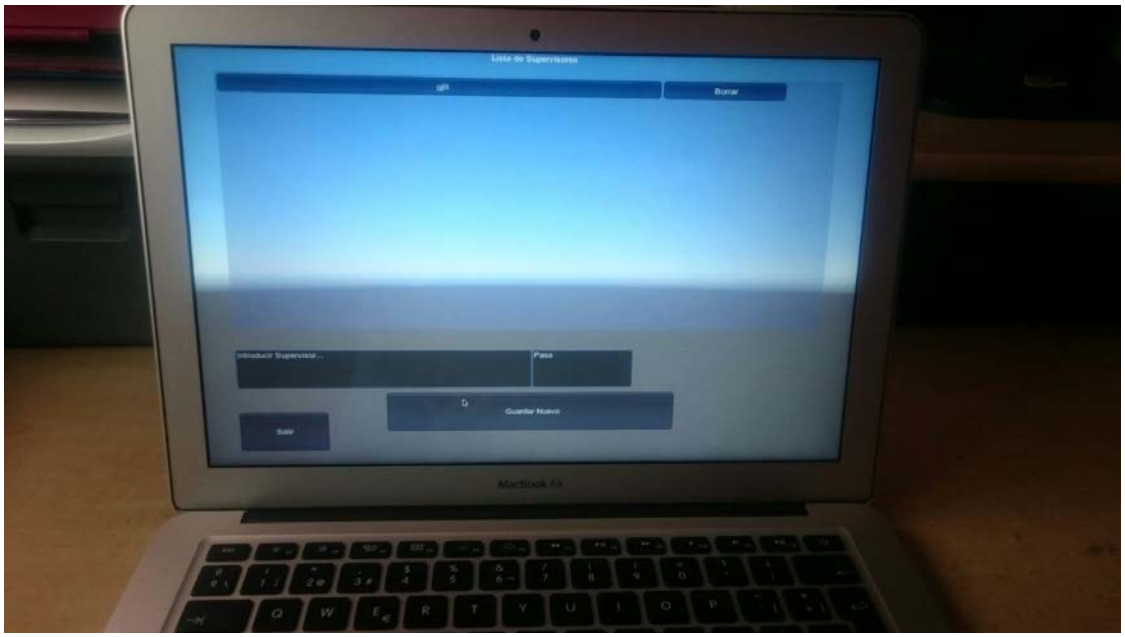
(a)

[37] Podemos obtener más información de la SDK así como acceder a su descarga en su sitio web oficial: <https://developer.android.com/sdk/index.html>. [Consultado: Septiembre 2015]

(b)



(c)



(d)

Figura 5-24: Imágenes que muestran ejecución de la aplicación en diferentes plataformas. (a) Tableta *Airis OnePad x110* con *Android 4.2*. (b) *Smatphone Samsung Galaxy S5* con *Android 4.4*. (c) Ordenador portátil con *Windows 7-64 bits*. (d) Ordenador portátil con *macOS-64 bits*.

El caso de la exportación al formato ***Unity® Web Player***[38] es de interés ya que requiere un paso más que el resto para la conexión al servidor, por ello vamos a mencionar algunos detalles de la exportación a dicho formato.

Cuando exportamos la aplicación a *Unity® Web Player* la herramienta de creación que existen en *Unity®* nos genera un fichero con extensión *.unity3d* y un fichero *.html* en el que se muestra de forma embebida la aplicación mediante la carga del fichero *.unity3d* en una página *Html*.

Para la correcta ejecución de la aplicación en esta plataforma es necesario tener en cuenta que el *plug-in* del que estamos hablando debe estar instalado en el navegador. Además para que la aplicación ejecutada con *Unity® Web Player* tenga acceso al servidor que atiende las peticiones de consultas a la base de datos, se precisa la existencia de un fichero de política de seguridad de conexión al servidor.

Esto se consigue en nuestro caso colocando un fichero de nombre *crossdomain.xml* en el servidor donde se encuentran los ficheros *.php* que atienden las peticiones. Es decir, si nuestros ficheros *.php* se encuentran en la dirección *URL* "*http://servidor.es/ficheros/consultaBD.php*" debemos colocar el fichero *crossdomain.xml* en la ruta "*http://servidor.es/ficheros/crossdomain.xml*". Este paso

[38] *Unity® Web Player* es una *plug-in* para navegadores web que permite ejecutar aplicaciones creadas con *Unity®* a través de internet. Podemos encontrar más detalles del *plug-in* en el siguiente en lace: **.

es necesario ya que la clase *WWW* que hemos empleado tiene restricciones[39] de seguridad al utilizarse en *Unity® Web Player*. Éste espera encontrar el fichero *croosdomain.xml* para evitar errores de acceso debido a restricciones de seguridad.

El contenido de *crossdomain.xml* para nuestro caso es sencillo, permite el acceso desde cualquier dominio y presenta la siguiente estructura:

```
<?xml version="1.0"?>
<cross-domain-policy>
<allow-access-from domain="*"/>
</cross-domain-policy>
```

Tabla 5-4: Ejemplo del formato de fichero *crossdomain.xml* que permite la comunicación de la aplicación ejecutada en *Unity® Web Player* con el servidor que atiende las peticiones de consultas a la base de datos.

Habiendo realizado los pasos anteriores (exportar, colocar el fichero *crossdomain.xml*, y tener los ficheros *.php* ya disponibles en diferentes *URL*), el último paso es colocar el fichero *.unity3d* y el fichero *.html* en un servidor. Con esto ya podremos acceder a la aplicación desde cualquier navegador a través de la *URL* que otorgue el servidor al fichero *.html*.

En cuanto a las conexiones del resto de plataformas con el servidor se comprobó que éstas se realizaban correctamente sin embargo vimos que era necesario crear una función que atendiera posibles fallos de conexión. Ésta atiende fallos típicos de un servidor y contempla la resolución de fallos mediante la reconexión para diferentes casos que surgieron al utilizar un servidor gratuito tales como el ancho de banda limitado, las respuestas que exceden un límite de tiempo, la no disponibilidad del servidor en determinadas horas, entre otros.

Un último punto a tener en cuenta sobre la portabilidad es que cada plataforma en la que se ejecute la aplicación hace uso de las correspondientes librerías gráficas presentes en dichas plataformas. Por ejemplo en el caso de *Android* se empleará *OpenGL*, mientras que en el caso de *Windows* se usará *DirectX*, etc. Dichas bibliotecas recrearán en la medida de lo posible la misma calidad en cuanto a detalles de las texturas, iluminación, sombras y mapas de normales. La calidad dependerá también de las opciones establecidas en las preferencias de exportación desde el panel *Build Settings* de *Unity®* en las que, por ejemplo para *Android*, se puede establecer que las dimensiones de las textura se reduzcan, se puede establecer un icono para identificar la aplicación instalada, se puede establecer la orientación (que en nuestro caso fue forzada para que se visualice la aplicación únicamente en modo panorámico), etc.

[39] Podemos encontrar más detalles del comportamiento de la clase WWW cuando se convierte la aplicación a formato *Unity® Web Player* en el siguiente enlace: <http://docs.unity3d.com/es/current/Manual/SecuritySandbox.html>. [Consultado: Septiembre 2015]

4.5 Experimentos y resultados

Tras la puesta en marcha y la verificación de que todos los elementos de la plataforma funcionaban correctamente de forma conjunta, se procedió a realizar pruebas con participantes reales. Dichas pruebas se separaron en dos grupos. Por un lado establecimos una versión de la aplicación en la que la configuración de los elementos del escenario fuera la misma para todos los participantes en todas las sesiones, y otra en la que dicha configuración dependiera del número de sesión a realizar por el participante.

El objetivo de estas dos versiones era, por un lado verificar la reducción de los tiempos de reacción (TR) debido al efecto de la práctica, y por otro lado comprobar el mantenimiento de los TR intentando eliminar el factor de la práctica introduciendo aleatoriedad. En ambos casos se pudo confirmar el crecimiento de los TR conforme aumenta la edad.

Como habíamos comentado en el apartado "Ámbito y alcance de la aplicación" existen problemas en el acceso léxico que pueden derivarse de los efectos del envejecimiento de una persona. Por ello seleccionamos un abanico de personas con diferentes edades y sexo de tal forma que los datos recogidos nos permitieran observar la evolución de los tiempos de reacción conforme aumenta la edad. En el mismo sentido se realizó un número de sesiones considerables ya que se pretendía que los valores medios fueran más precisos y así observar la evolución de los mismos conforme aumenta el número de sesiones realizadas. En la Tabla 5-5 podemos observar las características de los participantes.

Alias	Edad	Sexo	Alias	Edad	Sexo
2jict	7	Mujer	dact	23	Mujer
addt	8	Mujer	gett	23	Hombre
2klct	10	Hombre	3gpcd	23	Hombre
jhtt	11	Mujer	bhc	27	Mujer
jitt	13	Mujer	gjtt	27	Mujer
2dact	16	Mujer	gptb	37	Hombre
abdt	18	Hombre	cwdb	39	Mujer
3zbqt	21	Hombre	2mmtc	40	Hombre
4klls	21	Hombre	hctb	41	Hombre
3cmcd	21	Hombre	srtc	47	Mujer
jxdt	21	Mujer	2pecq	47	Mujer
3jmaf	22	Mujer	4obtb	48	Mujer
3daoo	22	Mujer			

Tabla 5-5: Características de los participantes de los experimentos.

Por otro lado gracias a la información retenida en la base de datos se pudo extraer otro tipo de información como por ejemplo la preferencia en la primera repisa seleccionada (inferior o superior), el orden de selección de los títulos objetivo, el tiempo entre clics, etc. Lo cual nos ha permitido experimentar de primera mano que se pueden hacer análisis más precisos cuantos más datos sobre la interacción de la aplicación con el participante se recojan. Incluso aquella información que a primera vista pareciera irrelevante puede ser de utilidad posteriormente en análisis más exhaustivos.

En el mismo sentido debemos mencionar que los tiempos de reacción (TR) recogidos se capturaron siguiendo el siguiente funcionamiento:

- El participante está correcta y cómodamente sentado frente al ordenador y con el ratón en la mano.
- La primera pantalla que observa es la pantalla previa al escenario en la que se le muestran los tres títulos objetivos duran 10 segundos.
- Cuando se muestra el escenario el participante debe buscar y seleccionar (hacer clic) en los libros objetivo.
- Cada selección muestra la desaparición del libro.
- No existen animaciones del escenario tras una selección.
- El **tiempo de reacción** está conformado por: tiempo de la búsqueda (segundos), tiempo de la selección (segundos) + posibles retardos del hardware (milisegundos).

Finalmente tenemos que mencionar que en este proyecto arrojamos los resultados de un primer análisis generalista y dejamos de manifiesto que el uso de una base de datos nos ha dado la posibilidad de analizar de forma rápida y ágil, con los conocimientos necesarios del lenguaje *SQL*, los datos recogidos para evaluar la evolución del TR (variable dependiente) con respecto a variables independientes como la edad, el sexo, el número de sesión, etc.

4.5.1 Efecto de la práctica

Las pruebas de este experimento se hicieron con las siguientes características:

- Personas de diferentes edades y sexos.
- Varias personas de la misma edad (en los casos que fueron posibles).
- 20 sesiones realizadas por cada participante.
- Sesiones supervisadas.
- La misma habitación para todas las sesiones.
- El mismo dispositivo para todas las sesiones.

- Títulos objetivo: Ángeles y demonios, Relato de un náufrago y Cien años de soledad.
- Dispositivo para la ejecución de la aplicación: ordenador portátil *Acer Aspire 5741* con sistema operativos *Windows 7* – 64 bits.
- Los tiempos de reacción capturados están expresados en segundos.

La configuración del escenario con la que se realizaron las pruebas es la que se observa en la Figura 5-25.

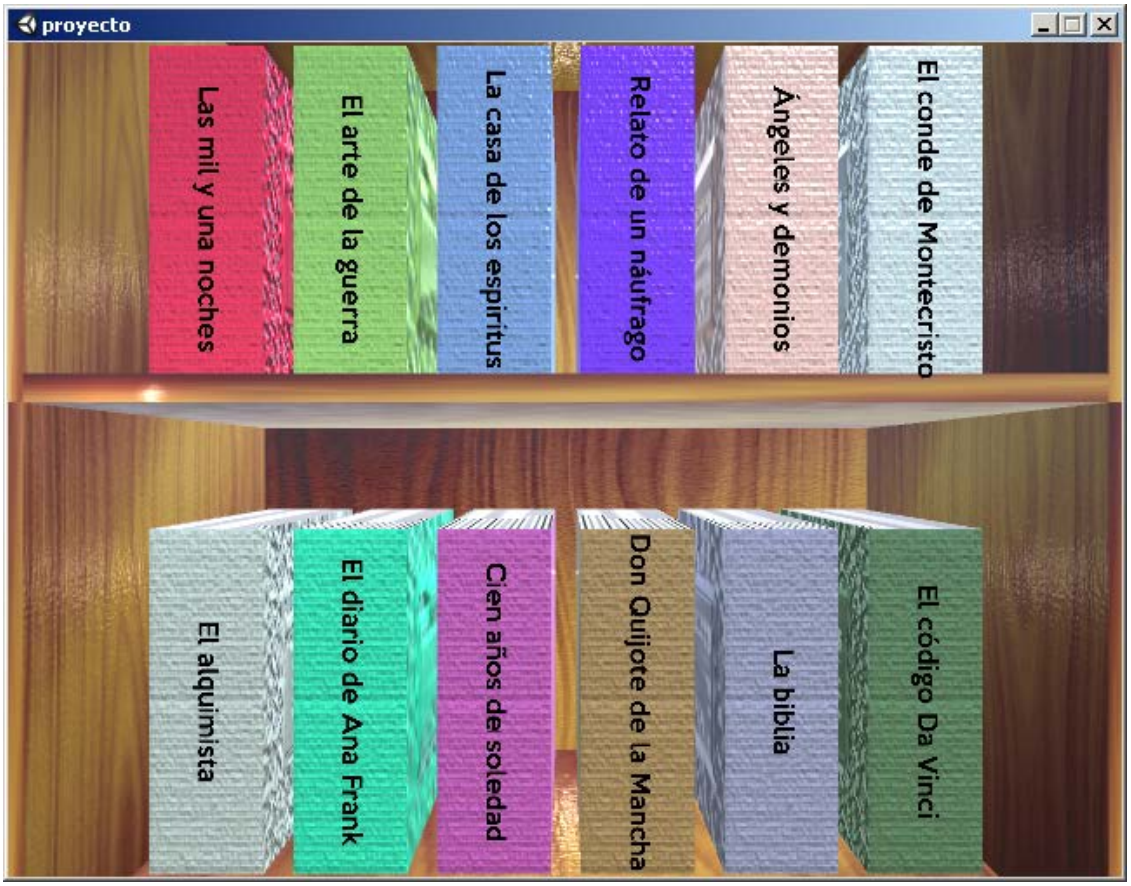


Figura 5-25: Configuración del escenario con las que se realizaron las pruebas del experimento que pretendía revelar la reducción del TR por el efecto de la práctica.

Como sabemos la primera sesión realizada por los participantes sería sin duda la más importante a la hora de verificar si el juego serio creado para realizar las tareas de decisión léxica se encontraba realmente dentro de los límites establecidos para esta prueba cognitiva. De las primeras sesiones realizadas por cada participante pretendíamos extraer las fortalezas y debilidades que presentaba el juego.

En estas primeras pruebas supervisadas experimentamos algunos sucesos que revelaron que la explicación del funcionamiento de la prueba quizá precisaría una realización de un ensayo previo para la familiarización del participante con el

escenario. Se encontró por ejemplo que un número reducido de los participantes que estaban realizando la prueba por primera vez preguntaron algo que se les había dicho con claridad, "¿Tengo que hacer clic en los libros?". Esto podría haber afectado a su tiempo de reacción en la primera sesión y por este motivo se realizaron 20 sesiones por cada participante buscando así que el cálculo de los tiempos de reacción medios fuera más preciso.

Se pudo observar también que los participantes que presentaron este inconveniente tenían un perfil similar: personas que hacen poco uso de los ordenadores y un uso igualmente menor de los videojuegos, por lo que el aspecto visual del escenario les pudo haber sorprendido.

Así mismo tras finalizar la primera sesión se les pidió a los participantes que valoraran el juego en cuanto a colores llamativos, preferencia de la primera repisa, la orientación de los libros, etc. A continuación vamos a mencionar algunas de las impresiones de los participantes y del supervisor recogidas tras realizar la primera sesión y posteriormente mencionaremos algunas correcciones que se tuvieron que hacer en la aplicación para que el resto de sesiones se realizaran adecuadamente.

Gett:

- Empezó a buscar por la repisa inferior.
- Los colores rojo y violeta llamaron su atención debido a su intensidad.
- Esperaba los títulos y los libros con una orientación para su lectura en horizontal.

Dact:

- Empezó a buscar por la repisa inferior.
- Esperaba los títulos y los libros con una orientación para su lectura en horizontal.
- Esperaba varias estanterías y opción de movimiento.

Hctb:

- Empezó a buscar por la repisa superior.
- Tuvo dificultad al buscar un libro debido a la confusión entre dos palabras con significado similar entre dos libros diferentes (un libro objetivo y un libro no objetivo). Las palabras eran “relato” y “diario” por lo que el participante se confundió.

Srtc:

- Empezó a buscar por la repisa superior.
- Esperaba los títulos con una orientación horizontal. Se apreció un notable giro de cuello para la lectura de los títulos.

- Pensaba que tenía que buscar los libros en el mismo orden en el que se le había presentado en la pantalla previa a la aparición del escenario.

Jhtt:

- Empezó a buscar por la repisa superior.
- Esperaba los títulos con una orientación horizontal. Se apreció un notable giro de cuello para la lectura de los títulos.

Jitt:

- Empezó a buscar por la repisa inferior.
- Se apreció tiempo excesivo al encontrar el último libro.

Jxdt:

- Empezó a buscar por la repisa superior.
- Esperaba los títulos con una orientación horizontal.
- Tuvo dificultad al buscar un libro debido a la confusión entre dos palabras con significado similar entre dos libros diferentes (un libro objetivo y un libro no objetivo). Las palabras eran “relato” y “diario” por lo que el participante se confundió.
- Los colores no le llamaron la atención.

Cwdb:

- Empezó a buscar por la repisa superior.
- Esperaba los títulos con una orientación horizontal.
- Tuvo dificultad al buscar un libro debido a la confusión entre dos palabras con significado similar entre dos libros diferentes (un libro objetivo y un libro no objetivo). Las palabras eran “relato” y “diario” por lo que el participante se confundió.
- El color violeta le llamó la atención.
- No esperaba que la cámara se moviera. Se pudo apreciar sorpresa en el participante al moverse la cámara lo cual pudo afectar a su tiempo de reacción.

Gptb:

- Empezó a buscar por la repisa superior.
- Esperaba los títulos con una orientación horizontal.
- No miraba los colores sino las letras.

Abdt:

- Empezó a buscar por la repisa superior.
- Esperaba los títulos con una orientación horizontal.
- El color rojo le llamó la atención.

Addt:

- Empezó a buscar por la repisa superior.

- Esperaba los títulos con una orientación horizontal. Se apreció un notable giro de cuello para la lectura de los títulos.
- Sólo miraba las letras.

2dact:

- Empezó a buscar por la repisa superior.
- Los colores no le llamaron la atención.
- Se apreció que el fallo en la selección de uno de los libros se produjo porque pretendía seleccionar todos muy rápido.
- Pensaba que tenía que buscar los libros en el mismo orden en el que se le había presentado en la pantalla previa a la aparición del escenario.

Bhc:

- Empezó a buscar por la repisa superior.
- Esperaba los títulos con una orientación vertical.
- Notable giro de cuello.
- El color violeta le llamo la atención.
- No esperaba moverse por el escenario.

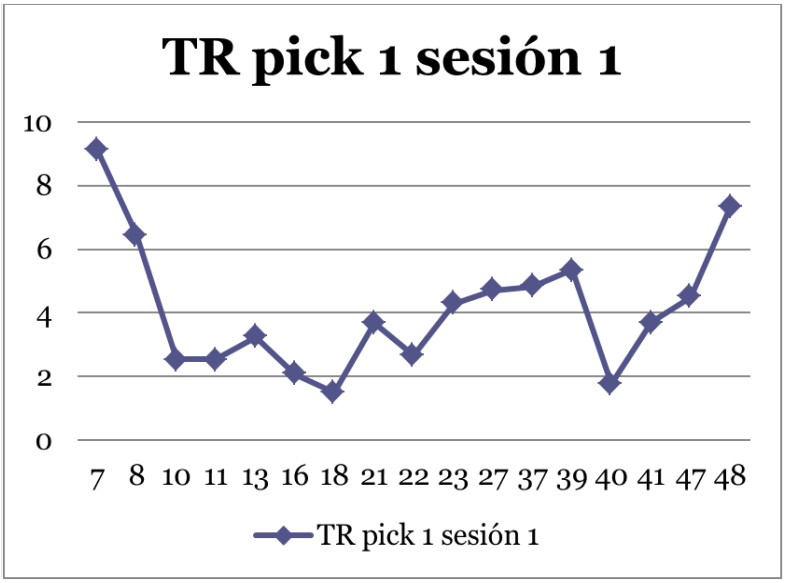
La corrección que se tuvo que realizar en la aplicación fue la referente al movimiento de la cámara que sorprendió a uno de los participantes como habíamos mencionado. Puesto que la recogida de datos contemplaba un conjunto de personas de diferentes edades con diferentes habilidades en el manejo de las nuevas tecnologías debíamos hacer que todas ellas estuvieran en igualdad de condiciones al interactuar con el escenario. Por tal motivo deshabilitamos la funcionalidad del movimiento de la cámara dejándolo disponible para una versión de la aplicación enfocada a participantes que no se vean sorprendidos o perjudicados por dicha posibilidad de movimiento tales como los jugadores habituales de videojuegos.

Por otro lado el hecho de que los participantes pensaran que debían seleccionar los libros en el mismo orden en el que se les presentaba los títulos objetivos en la pantalla previa de recordatorio se solventó indicándoles que podían seleccionarlos en cualquier orden. Sin embargo esto sacó a la luz que de tratarse de una prueba no supervisada los nuevos participantes presentaría el mismo inconveniente y quizá los datos recogidos responderían más al orden en el que se les presentaron los títulos objetivo que a la interacción propia de los participantes.

Así mismo se consideró que el resto de inconvenientes como la sorpresa al encontrar los títulos y los libros con orientación vertical se solventarían tras la primera sesión ya que esta permitiría a todos los participantes por un lado conocer el aspecto gráfico en

cuanto a los modelos 3D que simulan la estantería y los libros, la organización de los libros por estanterías, los colores, la iluminación, las texturas, etc., y por otro lado les permitiría entender el funcionamiento del juego serio y lo que sucede cuando se captura una de sus respuestas como es la desaparición del libro.

Puesto que consideramos que las primeras búsquedas son las que no están afectadas por el efecto de la práctica de los participantes, en la Figura 5-26 mostramos una tabla con los valores medios recogidos medidos en segundos para participantes con la misma edad y una gráfica que muestra el crecimiento del TR tardado conforme aumenta la edad en las primeras sesiones.



edad	TR pick 1 sesión 1
7	9,1262207
8	6,4708252
10	2,5386961
11	2,5336909
13	3,2457891
16	2,08252
18	1,5027159
21	3,6978512
22	2,665535
23	4,2976657
27	4,703476
37	4,837646
39	5,3695679
40	1,7607419
41	3,7169189
47	4,5000609
48	7,3128972

Figura 5-26: Gráfica que muestra el crecimiento del tiempo de reacción medio del primer pick en la primera sesión según aumenta la edad.

Los tiempos de reacción del participante de 7 años de edad en sus primeras sesiones, entre ellas la sesión uno mostrada en la figura anterior, se observa elevado por lo que parece oportuno mencionar algunos detalles.

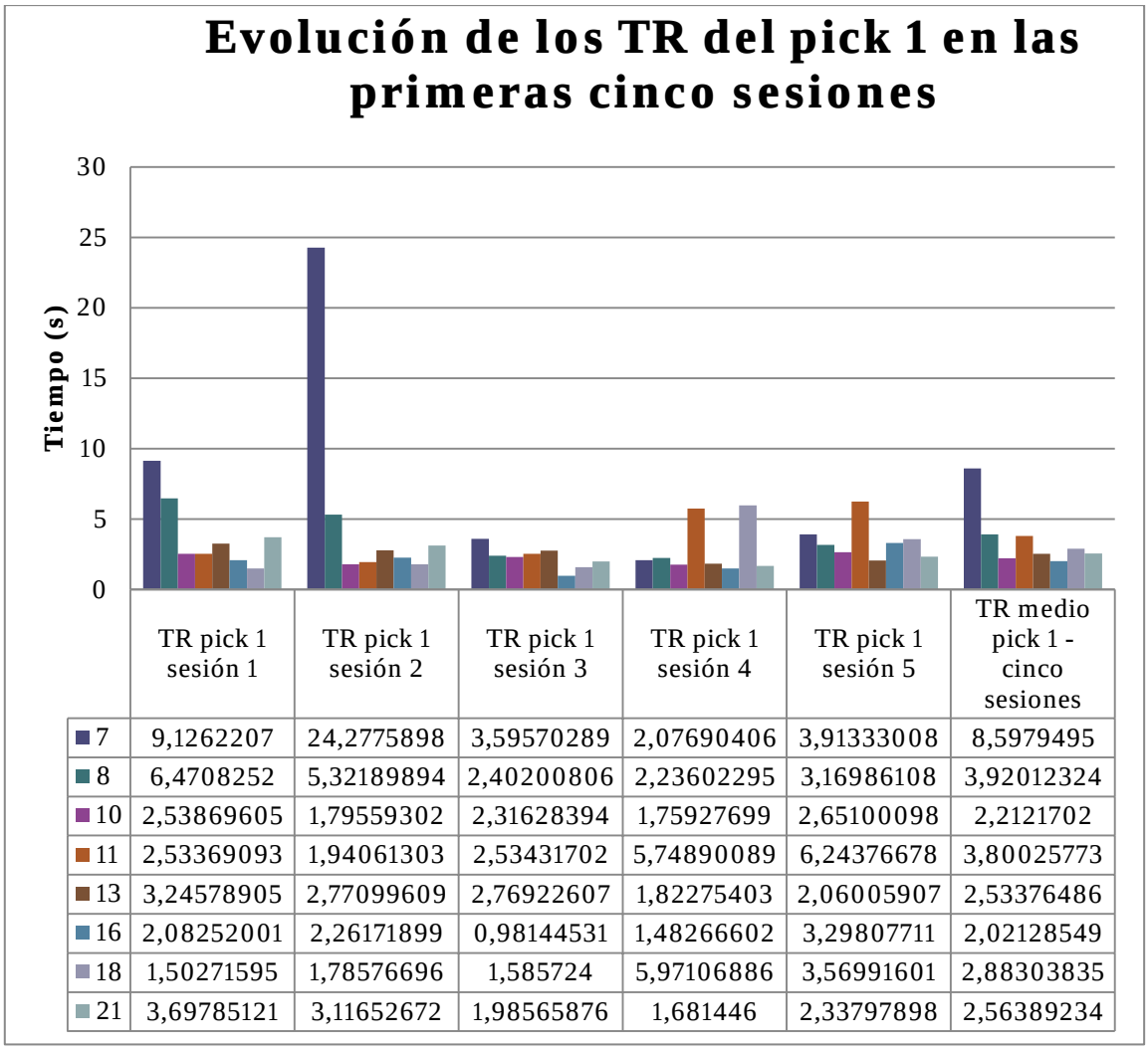
Durante la realización de la prueba de dicho participante se observó que el estado emocional del participante cambió al decirle que iba a realizar un aprueba mediante un videojuego. Este cambio emocional pudo haber sido la causa de que el participante no memorizara todos los títulos objetivo, pues el nerviosismo y la prisa por jugar eran notables en aspectos como los movimientos en las manos o la hiperactividad al moverse constantemente en la silla sentado frente al ordenador. Esto afectó a su tiempo de reacción pues solo recordaba uno de los títulos y los otros no los había memorizado, por lo que intentó recordar pero aun así falló al buscarlos. Observamos que esto se produjo sólo en la primera sesión de este participante y en el resto su estado era más tranquilo.

El problema podría haber sido debido a que se trata de una persona acostumbrada a utilizar videojuegos y la expectativa generada podría haberle puesto nervioso. Como sabemos hoy en día el repertorio de videojuegos a los que acceden los niños es muy grande y con contenido variado. Muchos de éstos presentan características que no son acordes a la edad del jugador. Por ejemplo, es común que algunos adultos caigan en la tentación de ofrecerle a un niño usar un videojuego para que se quede tranquilo durante unos minutos. Sin embargo al finalizar el juego el niño aún permanece inquieto. En el mismo sentido sabemos que actualmente se realizan estudios sobre este tipo de síntomas, ya que algunos de estos, podrían estar relacionados directamente con la exposición a programas de televisión o videojuegos, los cuales causan cambios fisiológicos en los niños como: expectación, alerta y aumento de adrenalina.

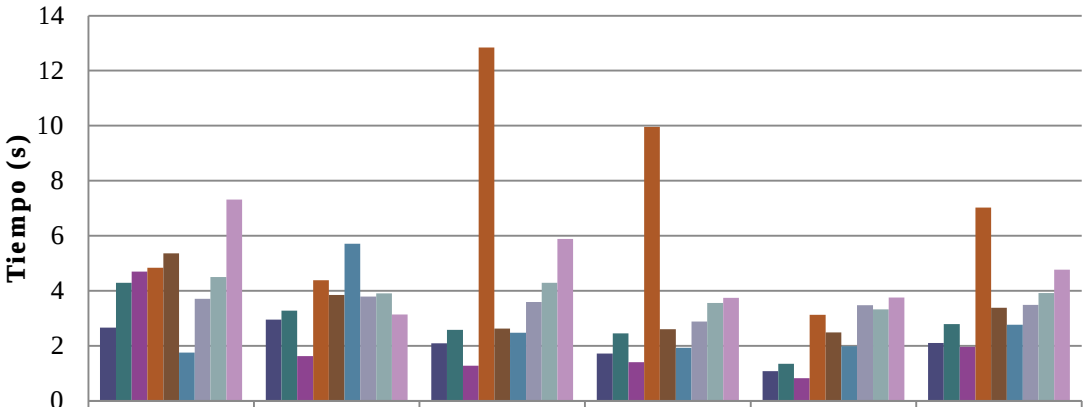
A modo de reflexión dejamos constancia de que quizá el uso de videojuegos de entretenimiento con contenido no acorde a la edad del jugador, en muchos de ellos contenido violento, podría hacer que se conviertan en participantes cuyos datos recogidos se desvíen del resto de participantes de mayor edad.

Este inconveniente en los datos recogidos fue solucionado igualmente con la repetición de la prueba 20 veces pero teniendo en cuenta el estado emocional de éste y el resto de participantes, así como observando que éstos presenten tranquilidad en las realizaciones de la prueba. Para comprobar que efectivamente el número de repeticiones nos permite balancear la media de los TR de cada participante podemos observar la Figura 5-27 que nos muestran la evolución (reducción) del tiempo de reacción expresado en segundos del primer *pick* en las primeras cinco sesiones.

Para entender la gráfica de dicha figura debemos mencionar que *pick i* representa una estructura que agrupa tres valores: tiempo de reacción capturado expresado en segundos, acierto o fallo, y el orden *i* (número de pick de entre los tres posibles para seleccionar los tres libros objetivo).



	TR pick 1 sesión 1	TR pick 1 sesión 2	TR pick 1 sesión 3	TR pick 1 sesión 4	TR pick 1 sesión 5	TR medio pick 1 - cinco sesiones
7	9,1262207	24,2775898	3,59570289	2,07690406	3,91333008	8,5979495
8	6,4708252	5,32189894	2,40200806	2,23602295	3,16986108	3,92012324
10	2,53869605	1,79559302	2,31628394	1,75927699	2,65100098	2,2121702
11	2,53369093	1,94061303	2,53431702	5,74890089	6,24376678	3,80025773
13	3,24578905	2,77099609	2,76922607	1,82275403	2,06005907	2,53376486
16	2,08252001	2,26171899	0,98144531	1,48266602	3,29807711	2,02128549
18	1,50271595	1,78576696	1,585724	5,97106886	3,56991601	2,88303835
21	3,69785121	3,11652672	1,98565876	1,681446	2,33797898	2,56389234



	TR pick 1 sesión 1	TR pick 1 sesión 2	TR pick 1 sesión 3	TR pick 1 sesión 4	TR pick 1 sesión 5	TR medio pick 1 - cinco sesiones
22	2,66553497	2,95280445	2,09350598	1,72671503	1,08691412	2,10509491
23	4,29766568	3,27677053	2,58784199	2,45643099	1,35092668	2,79392717
27	4,70347595	1,624695	1,28093129	1,41271996	0,82489771	1,96934398
37	4,83764601	4,38787794	12,8410902	9,95397854	3,13626099	7,03137074
39	5,36956787	3,85397291	2,63540602	2,60376	2,485641	3,38966956
40	1,76074195	5,70803785	2,48327589	1,92736804	1,99463701	2,77481215
41	3,71691895	3,79906797	3,59710693	2,88940406	3,47979689	3,49645896
47	4,50006092	3,91241443	4,29524249	3,56083691	3,32954061	3,91961907
48	7,31289721	3,14361596	5,88230896	3,74291992	3,76178002	4,76870441

Figura 5-27: Gráfica que muestra la evolución del tiempo de reacción del primer pick en las primeras cinco sesiones de los participantes con una misma edad.

Como vemos en la figura anterior se empieza a notar la reducción de los tiempos de reacción de los participantes conforme aumenta el número de sesiones que éstos realizan. Para arrojar una primera vista de este efecto vamos a mostrar en la Figura 5-28 la evolución del TR del *pick* 1 de 20 sesiones de los participantes más jóvenes.

numSesión	TR 7 años	TR 8 años	TR 10 años	TR 11 años	TR 13 años
0	6,0506999	8,1153555	2,43107104	2,3582559	3,2574464
1	13,917237	3,5738117	5,30873609	2,3157758	2,358134
2	2,298828	1,5457357	2,2998453	2,8233746	1,9712321
3	1,3716635	3,5799154	2,66320765	4,1673583	1,346456
4	3,6520998	2,246338	1,94803902	5,6451492	1,4249473
5	3,3919273	4,7966514	2,26838569	1,8116456	1,267741
6	2,2815143	2,7517904	2,01839737	2,521586	5,4153644
7	1,7352905	2,753825	1,63437412	1,0783793	10,17749
8	0,9738668	1,7954104	1,47882559	1,1314699	2,6643677
9	1,2396444	1,9018554	1,42932121	1,1703189	1,465495
10	1,1252035	1,2009329	1,77771238	0,7856852	1,3887227
11	1,919662	1,3341676	1,38308215	0,9436646	1,4661356
12	1,9324215	1,8900961	1,44667545	0,6705119	1,303843
13	1,0985156	1,0395915	0,99212122	0,6184895	0,7597046
14	1,562017	1,2451578	1,324748	0,5527547	0,9480386
15	1,7034046	1,0561524	0,85623183	0,6274619	0,914795
16	1,3768107	1,4554037	1,23064176	0,671051	0,9100953
17	0,9804077	1,0286715	1,00391507	2,0513279	0,8461305
18	1,1294456	0,9618937	1,5221887	0,6054384	0,990458
19	1,2248436	1,155337	1,33945667	0,5434572	0,7361042

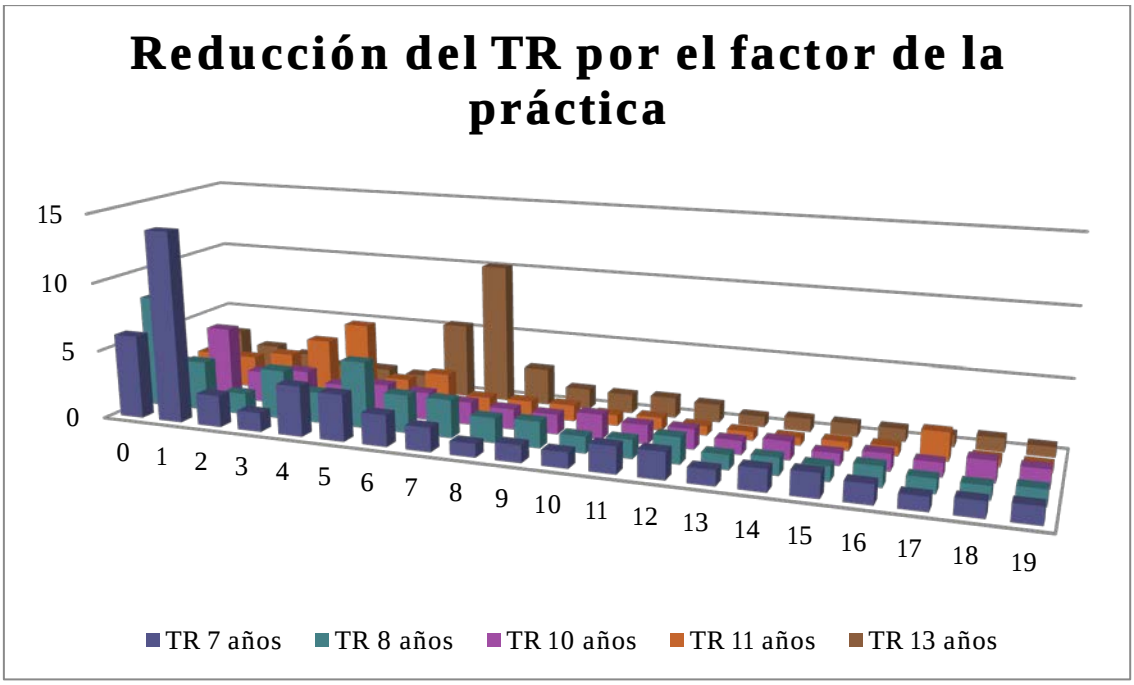


Figura 5-28: Gráfica que muestra la evolución (reducción) de los tiempos de reacción del primer *pick* por el efecto de la práctica de 20 sesiones de los participantes más jóvenes

Y para verificar finalmente la reducción del tiempo de reacción por el efecto de la práctica vamos a mostrar **como primer resultado** la media de los TR **de los tres pick** posibles de todos los participantes con la misma edad en función del número de sesión en la Figura 5-29.

numSesion	TR medio de todos los participantes
0	3,468009
1	3,3775626
2	2,3864796
3	2,2484211
4	2,0332761
5	2,544905
6	1,9160465
7	1,7548025
8	1,4813475
9	1,391824
10	1,4135411
11	1,4415941
12	1,3528842
13	1,2594659
14	1,3421641
15	1,0869622
16	1,1457795
17	1,1215489
18	1,0265397
19	1,0086217

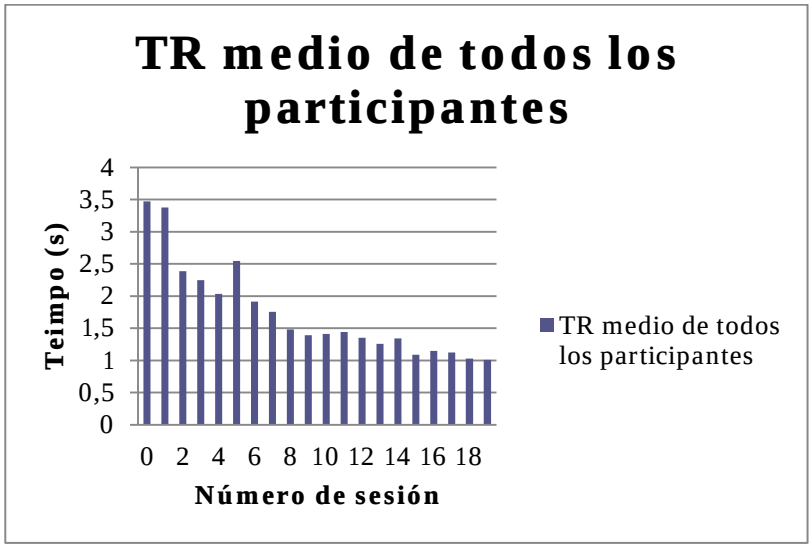


Figura 5-29: Gráfica que muestra la reducción del TR medio de **todos** los participantes por el efecto de la práctica de 20 sesiones.

Como **segundo resultado** vamos a mostrar las siguientes tablas en las que se puede observar que el tiempo de reacción medio de las 20 sesiones de participantes con una

misma edad evoluciona de forma creciente conforme aumenta la edad. Esto se observa en la Figura 5-30 en la que se destaca la **línea de tendencia** del tiempo de reacción medio de los tres *pick* de todos los participantes con una misma edad, marcada como una línea discontinua.

Edad	TR pick1	TR pick2	TR pick3	TR medio (pick 1,2, y 3)
7	3,863052	1,529832	2,371714	2,569951
8	2,840265	1,345176	2,628773	2,271405
10	2,118367	1,62688	1,783835	1,843027
11	2,07493	1,878823	1,01022	1,654658
13	2,425525	1,463044	2,353337	2,080635
16	1,885917	0,829905	1,18505	1,300291
18	1,508253	0,802145	0,937225	1,082541
21	1,954064	1,32013	1,360894	1,545959
22	1,214857	0,764453	0,833932	0,937748
23	2,173847	1,231859	1,535461	1,647776
27	1,055335	1,024464	0,637788	0,908115
37	4,508066	2,663949	2,121404	3,097806
39	2,257829	1,492082	2,483262	2,077724
40	2,089154	1,291341	1,875117	1,751871
41	3,021793	1,751565	1,913999	2,242786
47	2,93808	2,26856	2,439961	2,548867
48	2,548379	2,197647	1,444765	2,063597

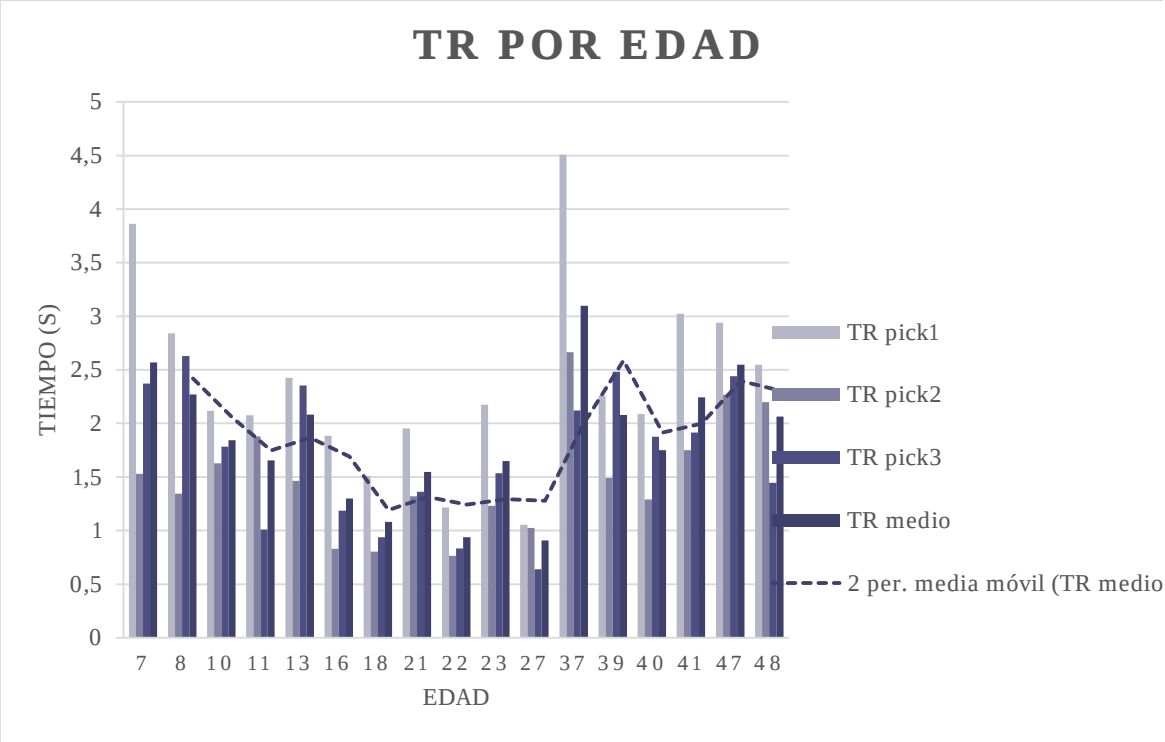


Figura 5-30: Gráfica que muestra el incremento en los TR medios de 20 sesiones conforme aumenta la edad.

A partir de la quinta sesión se pudo apreciar que los participantes se guiaban por los colores y la posición que habían reconocido en las sesiones previas. Es más se pudo apreciar que se dirigían directamente hacia los dos primeros libros que se encontraban colocados adyacentemente en la repisa superior. Esto puede ser corroborado con los histogramas de la Figura 5-31 que muestran la tasa de preferencia en el primer, segundo y tercer pick de todas las sesiones.

numRepisa	COUNT(`numRepisa`)	Tasa de selección de repisa en el pick 1	COUNT(`numRepisa`)	Tasa de selección de repisa en el pick 2	COUNT(`numRepisa`)	Tasa de selección de repisa en el pick 3
0	138	0,281632653	56	0,11428571	296	0,60408163
1	352	0,718367347	434	0,88571429	194	0,39591837

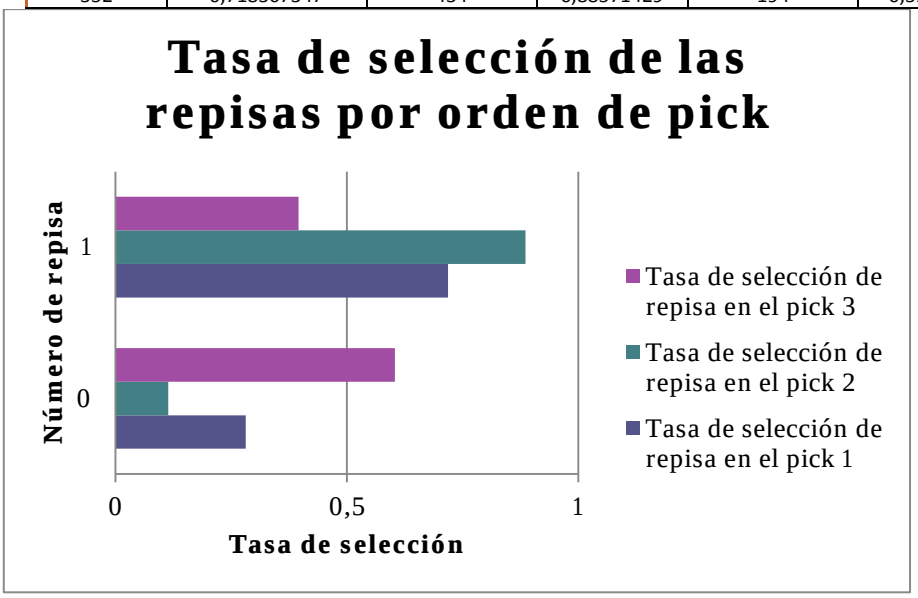


Figura 5-31: Gráfica que muestra la tasa de selección de las repisas de 20 sesiones de todos los participantes. Muestra una mayor preferencia en la repisa superior

Como vemos los participantes presentan una mayor disponibilidad para realizar los dos primeros pick (uno y dos) en la repisa uno (repisa superior). Esto puede ser debido a la memorización del escenario, puesto que el participante puede recordar que, debido a la replicación de la aleatoriedad del escenario para diferentes participantes, existen dos libros que están juntos y le puede resultar más rápido buscar y seleccionar aquellos dos de una pasada y dejar para el final el tercer libro. Esto se observa claramente en la gráfica anterior en la que la repisa cero (inferior) solo tiene mayor porcentaje cuando se trata del último pick, el pick 3.

4.5.2 Aplicación de aleatoriedad

En el segundo experimento nos enfocamos en medir el TR intentando minimizar el efecto de la memorización debido a la práctica. Para ello, hemos modificado algunos aspectos de la configuración del escenario.

Resumimos a continuación el funcionamiento actualizado de la aplicación con el que se intenta que los datos recogidos sean más exactos con referencia a la medición del tiempo de reacción únicamente de la tarea de decisión léxica:

- Los mismos participantes del experimento anterior realizan el mismo número de sesiones.
- 20 sesiones supervisadas.
- La configuración del escenario se basa en el número de sesión a realizar por el participante seleccionado, y por tanto cada participante va encontrando escenarios distintos en las diferentes sesiones.
- Dos participantes realizando el mismo número de sesión observarán la misma configuración del escenario (mismo número de libros, colores, distribución y títulos objetivo).
- La aleatoriedad se aplica a:
    - Colores de los libros en una misma repisa.
    - Colores de los libros en diferentes repisas.
    - Compartimento asignado al libro dentro de la estantería.
    - Títulos de las tapas de los libros en diferentes repisas.
- La opción de marcar/desmarcar títulos como objetivo está desactivada para tener consistencia en los datos recogidos.
- Títulos objetivo: Ángeles y demonios, Relato de un náufrago y Cien años de soledad.
- Dispositivo para la ejecución de la aplicación: ordenador portátil *Acer Aspire 5741* con sistema operativos *Windows 7* – 64 bits.
- Los tiempos de reacción capturados están expresados en segundos.

Podemos observar un ejemplo de la configuración aleatoria del escenario entre dos sesiones consecutivas de un participante en la Figura 5-32.

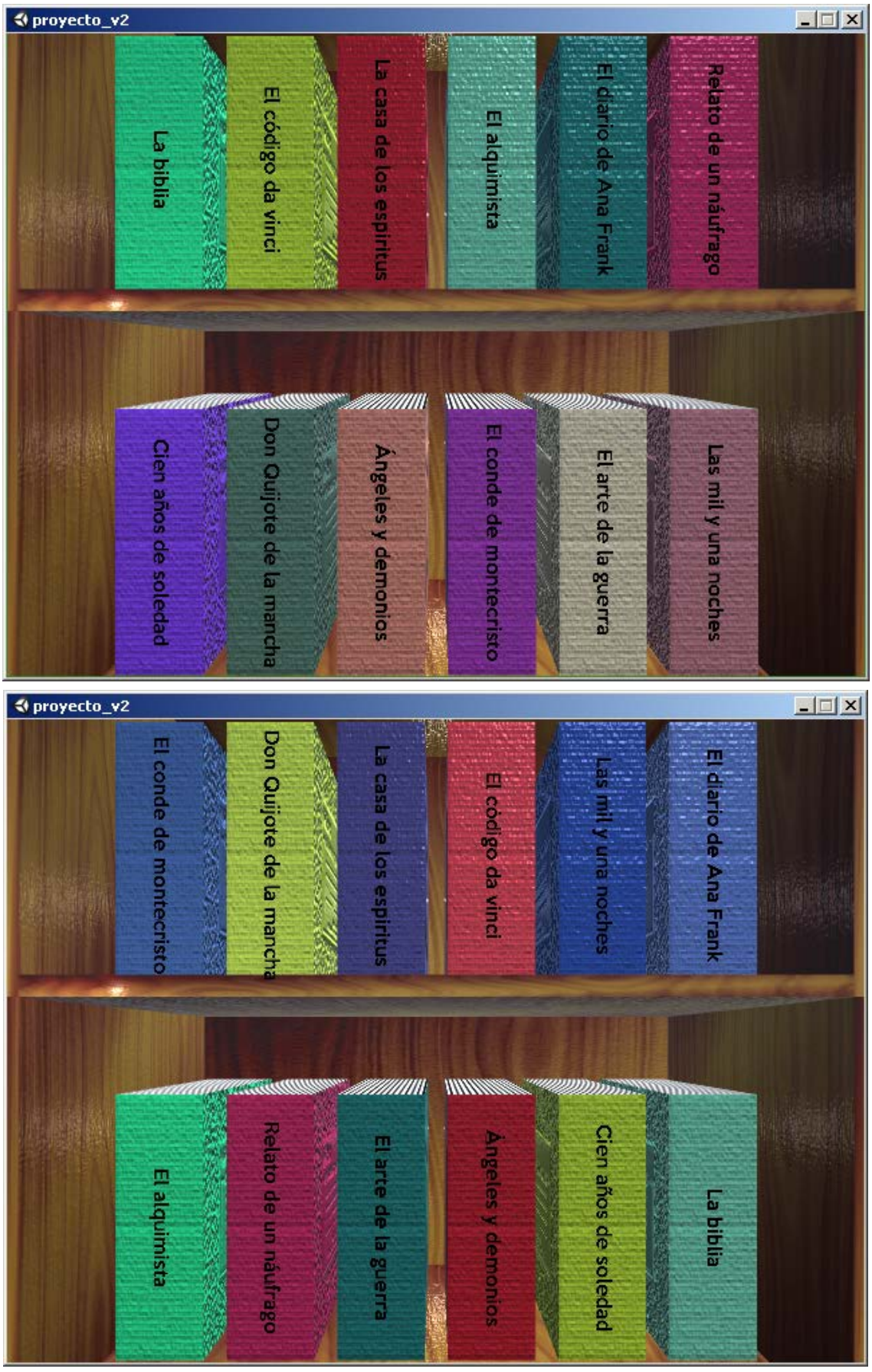


Figura 5-32: Ejemplo de la configuración aleatoria entre dos sesiones consecutivas de un mismo participante.

Las siguientes tablas sirven para corroborar el planteamiento de partida de aplicar aleatoriedad para minimizar los efectos de la práctica.

edad	TR pick 1 sesión 1 (segundos)
7	4,4521708
8	4,748291
10	7,1009521
11	1,616894
13	7,0393372
16	2,367691
18	3,4827881
21	4,7675646
22	1,2421609
23	3,7596935
27	2,9867705
37	5,184875
39	3,4000239
40	2,9655149
41	1,754761
47	3,0376894
48	3,2628779

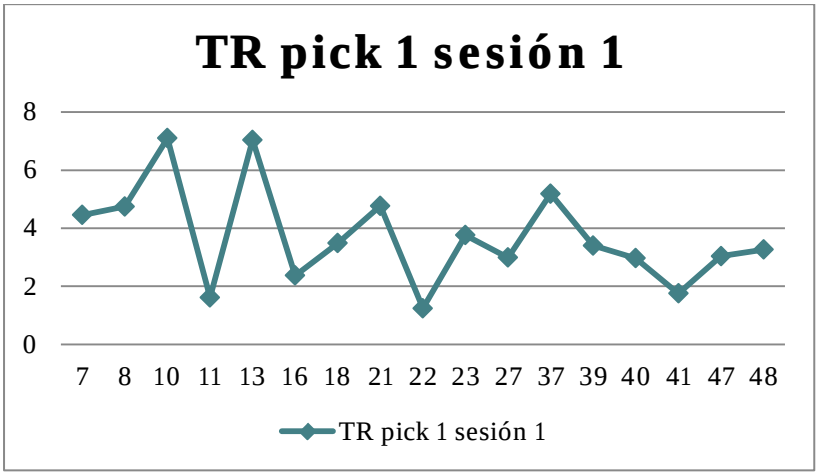
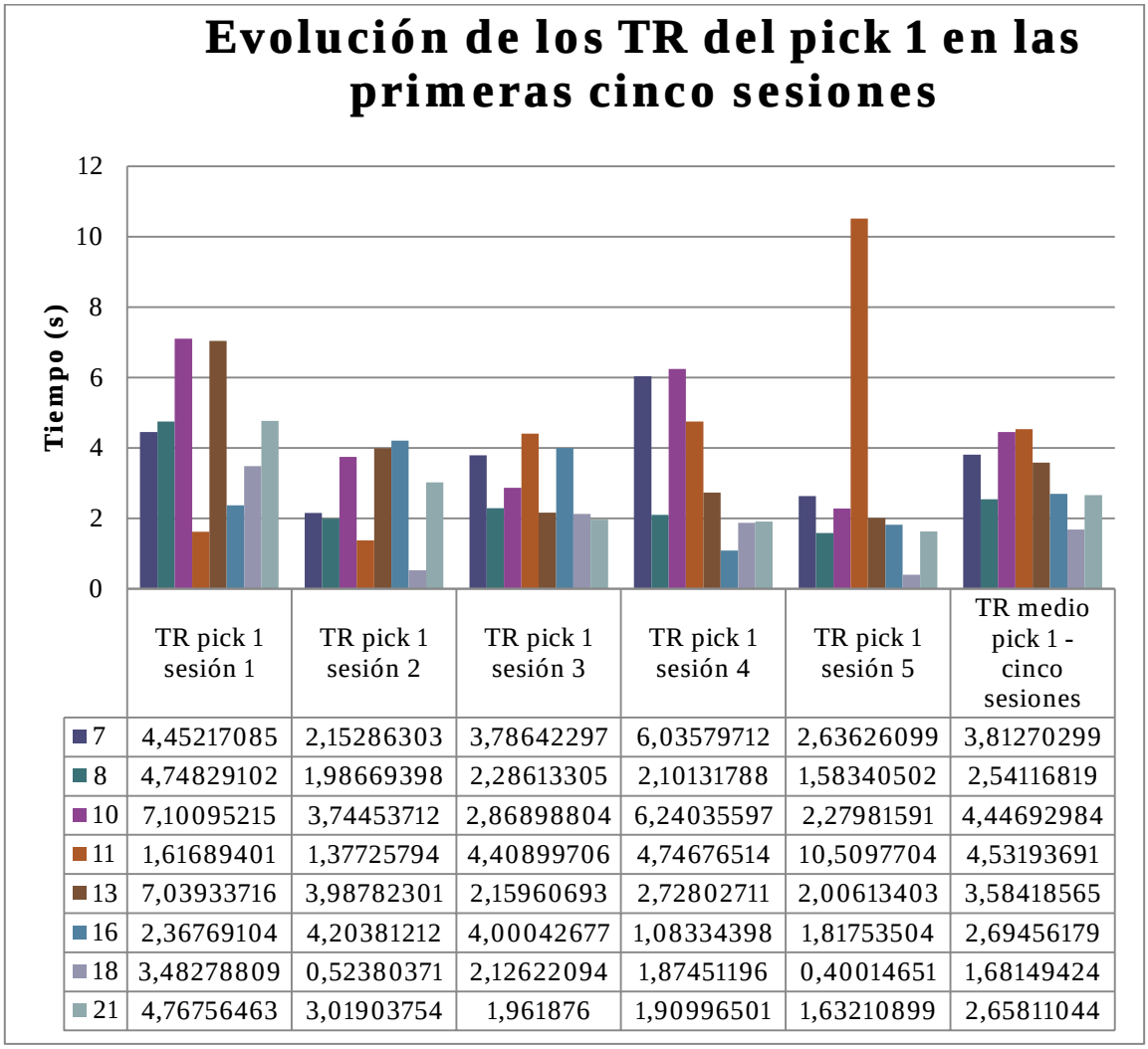
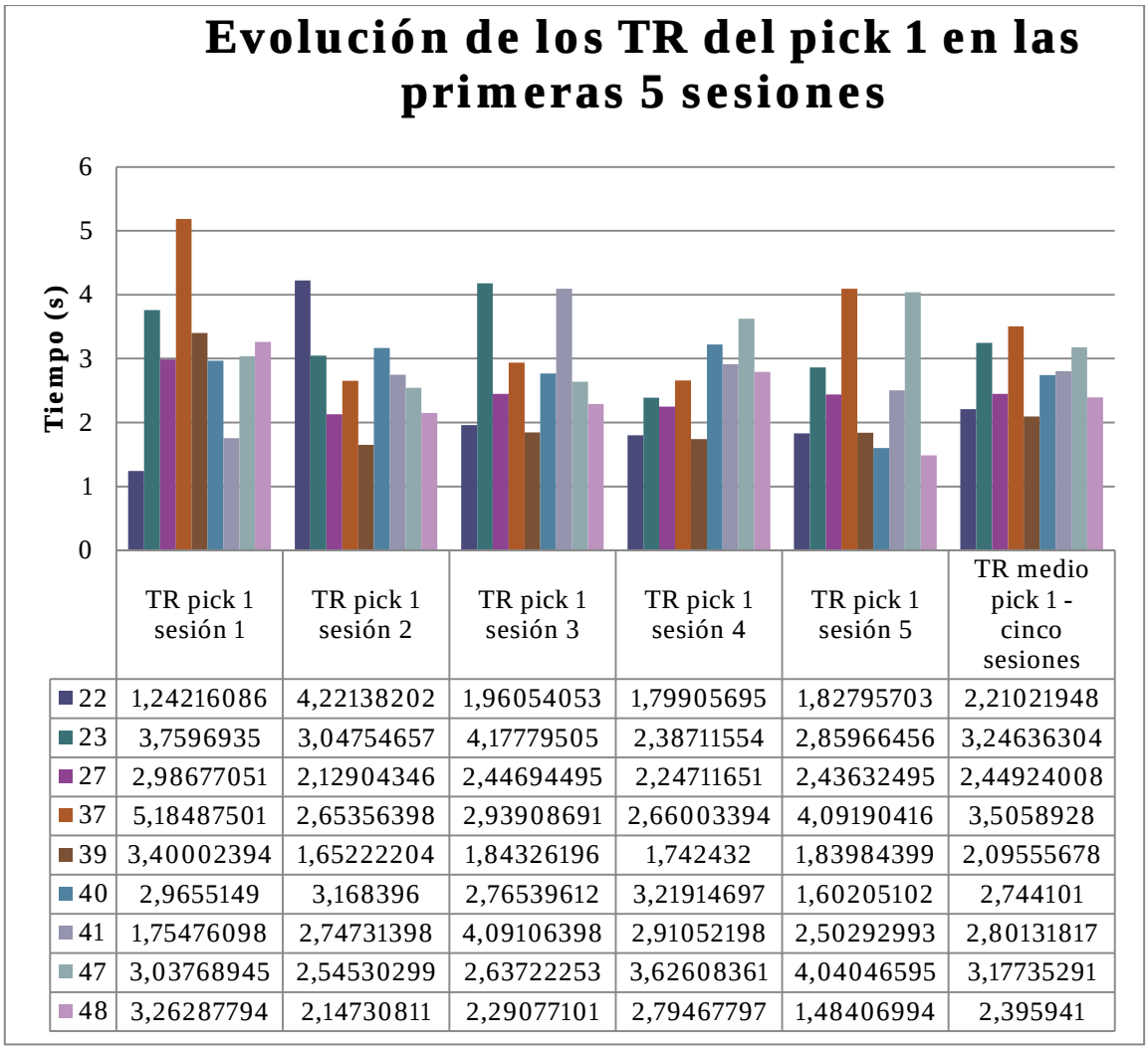


Figura 5-33: Tiempos de reacción del pick 1 en la sesión 1. Experimento que aplica la aleatoriedad para minimizar el efecto de la práctica.



	TR pick 1 sesión 1	TR pick 1 sesión 2	TR pick 1 sesión 3	TR pick 1 sesión 4	TR pick 1 sesión 5	TR medio pick 1 - cinco sesiones
7	4,45217085	2,15286303	3,78642297	6,03579712	2,63626099	3,81270299
8	4,74829102	1,98669398	2,28613305	2,10131788	1,58340502	2,54116819
10	7,10095215	3,74453712	2,86898804	6,24035597	2,27981591	4,44692984
11	1,61689401	1,37725794	4,40899706	4,74676514	10,5097704	4,53193691
13	7,03933716	3,98782301	2,15960693	2,72802711	2,00613403	3,58418565
16	2,36769104	4,20381212	4,00042677	1,08334398	1,81753504	2,69456179
18	3,48278809	0,52380371	2,12622094	1,87451196	0,40014651	1,68149424
21	4,76756463	3,01903754	1,961876	1,90996501	1,63210899	2,65811044

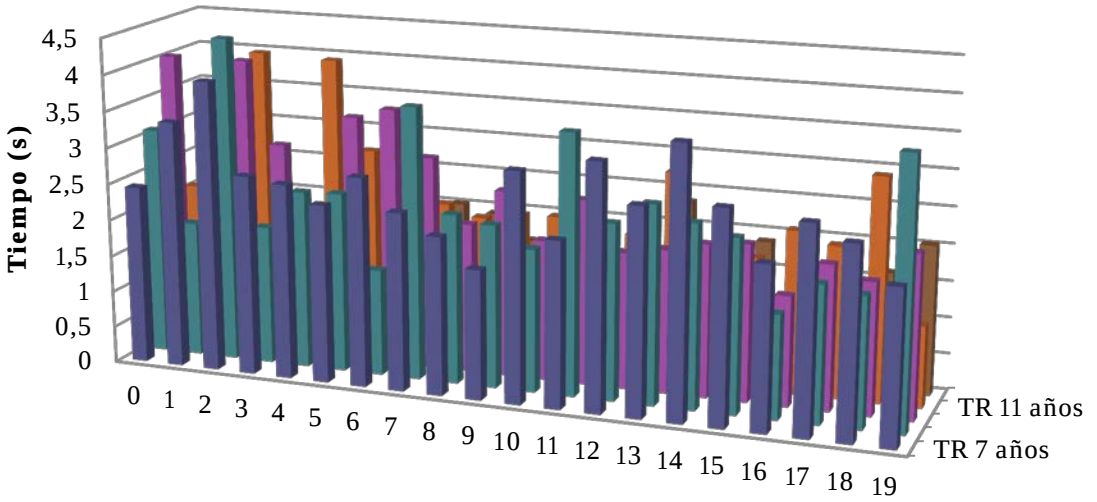


	TR pick 1 sesión 1	TR pick 1 sesión 2	TR pick 1 sesión 3	TR pick 1 sesión 4	TR pick 1 sesión 5	TR medio pick 1 - cinco sesiones
22	1,24216086	4,22138202	1,96054053	1,79905695	1,82795703	2,21021948
23	3,7596935	3,04754657	4,17779505	2,38711554	2,85966456	3,24636304
27	2,98677051	2,12904346	2,44694495	2,24711651	2,43632495	2,44924008
37	5,18487501	2,65356398	2,93908691	2,66003394	4,09190416	3,5058928
39	3,40002394	1,65222204	1,84326196	1,742432	1,83984399	2,09555678
40	2,9655149	3,168396	2,76539612	3,21914697	1,60205102	2,744101
41	1,75476098	2,74731398	4,09106398	2,91052198	2,50292993	2,80131817
47	3,03768945	2,54530299	2,63722253	3,62608361	4,04046595	3,17735291
48	3,26287794	2,14730811	2,29077101	2,79467797	1,48406994	2,395941

Figura 5-34: Gráfica que muestra la evolución del tiempo de reacción del primer pick en las primeras cinco sesiones. Experimento que aplica la aleatoriedad para minimizar el efecto de la práctica.

Como vemos en este experimento la tendencia ya no es a reducir el tiempo de reacción. Para verificarlo observamos las 20 sesiones de los participantes más jóvenes como hicimos en el experimento anterior.

numSesión	TR 7 años	TR 8 años	TR 10 años	TR 11 años	TR 13 años
0	2,4450453	3,123413	4,05420709	2,08934804	2,9411926
1	3,383606	1,8547363	2,37996936	1,71887199	2,023071
2	3,9780223	4,4586182	4,05997229	4,07531762	2,221049
3	2,7341614	1,8979083	2,92691032	2,11008708	1,5356444
4	2,6671346	2,438263	2,04295862	4,03955243	2,1978354
5	2,4338379	2,4593407	3,39369702	2,81224559	1,4287517
6	2,8557943	1,4600219	3,53705831	2,52158598	1,0900471
7	2,428335	3,7260842	2,92150903	2,15576174	2,0265299
8	2,1565346	2,3160297	2,05051668	2,00931803	1,9311727
9	1,7727456	2,2251997	2,5545857	2,1009115	1,5995076
10	3,1264759	1,9562176	1,92498916	2,1171877	1,7817384
11	2,2615358	3,5547283	2,53084326	2,24890153	1,9033203
12	3,3392284	2,3997393	1,87528487	1,97017435	2,2245281
13	2,8114781	2,703186	1,96455872	2,86531603	2,3420411
14	3,6671956	2,5027061	2,08695463	1,48388664	1,3442916
15	2,8822187	2,362447	2,14033004	1,8144937	1,8849054
16	2,2227428	1,419035	1,50600169	2,22635937	1,8456726
17	2,7893017	1,870402	1,98110966	2,07381201	1,5492578
18	2,5786029	1,778361	1,80649805	3,04235792	1,6276803
19	2,0935907	3,632568	2,23188271	1,11879983	2,0437417



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
TR 7 años	2,4	3,4	4	2,7	2,7	2,4	2,9	2,4	2,2	1,8	3,1	2,3	3,3	2,8	3,7	2,9	2,2	2,8	2,6	2,1
TR 8 años	3,1	1,9	4,5	1,9	2,4	2,5	1,5	3,7	2,3	2,2	2	3,6	2,4	2,7	2,5	2,4	1,4	1,9	1,8	3,6
TR 10 años	4,1	2,4	4,1	2,9	2	3,4	3,5	2,9	2,1	2,6	1,9	2,5	1,9	2	2,1	2,1	1,5	2	1,8	2,2
TR 11 años	2,1	1,7	4,1	2,1	4	2,8	2,5	2,2	2	2,1	2,1	2,2	2	2,9	1,5	1,8	2,2	2,1	3	1,1
TR 13 años	2,9	2	2,2	1,5	2,2	1,4	1,1	2	1,9	1,6	1,8	1,9	2,2	2,3	1,3	1,9	1,8	1,5	1,6	2

Figura 5-35: Gráfica que muestra la evolución (mantenimiento) de los tiempos de reacción al introducir aleatoriedad para minimizar el efecto de la práctica.

Incluso podemos comparar las evoluciones de un participante (jhtt) en concreto para reafirmar el hecho:

numSesión	TR 11 años experimento 1	TR 11 años experimento 2
0	2,3582559	2,089348
1	2,3157758	1,718872
2	2,8233746	4,0753176
3	4,1673583	2,1100871
4	5,6451492	4,0395524
5	1,8116456	2,8122456
6	2,521586	2,521586
7	1,0783793	2,1557617
8	1,1314699	2,009318
9	1,1703189	2,1009115
10	0,7856852	2,1171877
11	0,9436646	2,2489015
12	0,6705119	1,9701744
13	0,6184895	2,865316
14	0,5527547	1,4838866
15	0,6274619	1,8144937
16	0,671051	2,2263594
17	2,0513279	2,073812
18	0,6054384	3,0423579
19	0,5434572	1,1187998

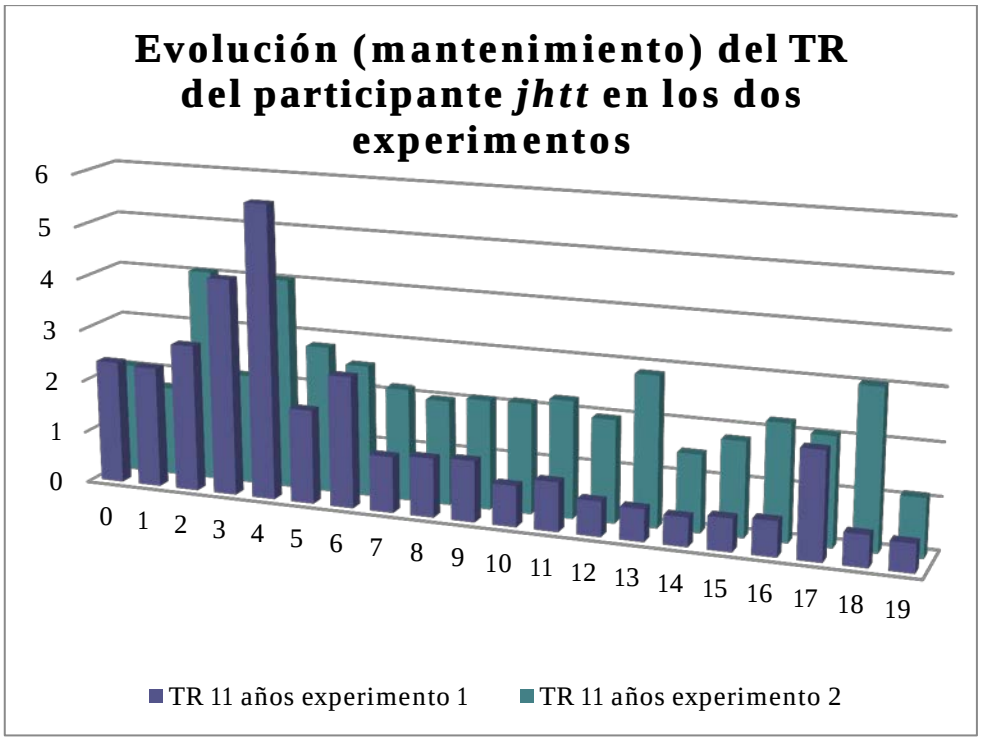


Figura 5-36: Gráfica que muestra una comparación de las evoluciones de los TR de un participante en los dos experimentos.

Por otro lado si analizamos la evolución del TR con respecto a la edad encontramos el mismo resultado que en el experimento anterior, el TR crece conforme aumenta la edad.

Edad	TR pick1	TR pick2	TR pick3	TR medio
7	3,0582764	2,4539513	2,6819105	2,7313794
8	2,4569476	2,7587953	2,3051079	2,5069503
10	3,0629674	1,7117662	2,6505596	2,4721238
11	3,1342176	1,5474425	2,0573069	2,2633972
13	2,7605248	1,3482777	1,5224943	1,8770989
16	2,5777107	2,1016246	1,499404	2,0595798
18	1,5786255	1,482483	1,3262695	1,4624593
21	2,4218395	1,8184801	1,6622886	1,9694211
22	2,2691036	1,8380712	1,9270403	2,011405
23	2,8192928	1,9685346	2,410958	2,3995951
27	2,3016042	1,51324	1,6580044	1,8242829
37	4,0465546	3,3369773	2,9431151	3,4422157
39	1,9802189	1,7283204	1,4909151	1,7331514
40	2,4890033	1,5753551	2,0679639	2,0441074
41	2,8698058	2,2610169	1,6870788	2,2726338
47	2,8599603	2,0247391	2,3934959	2,4260651
48	3,4322257	3,2840292	4,5808318	3,7656956

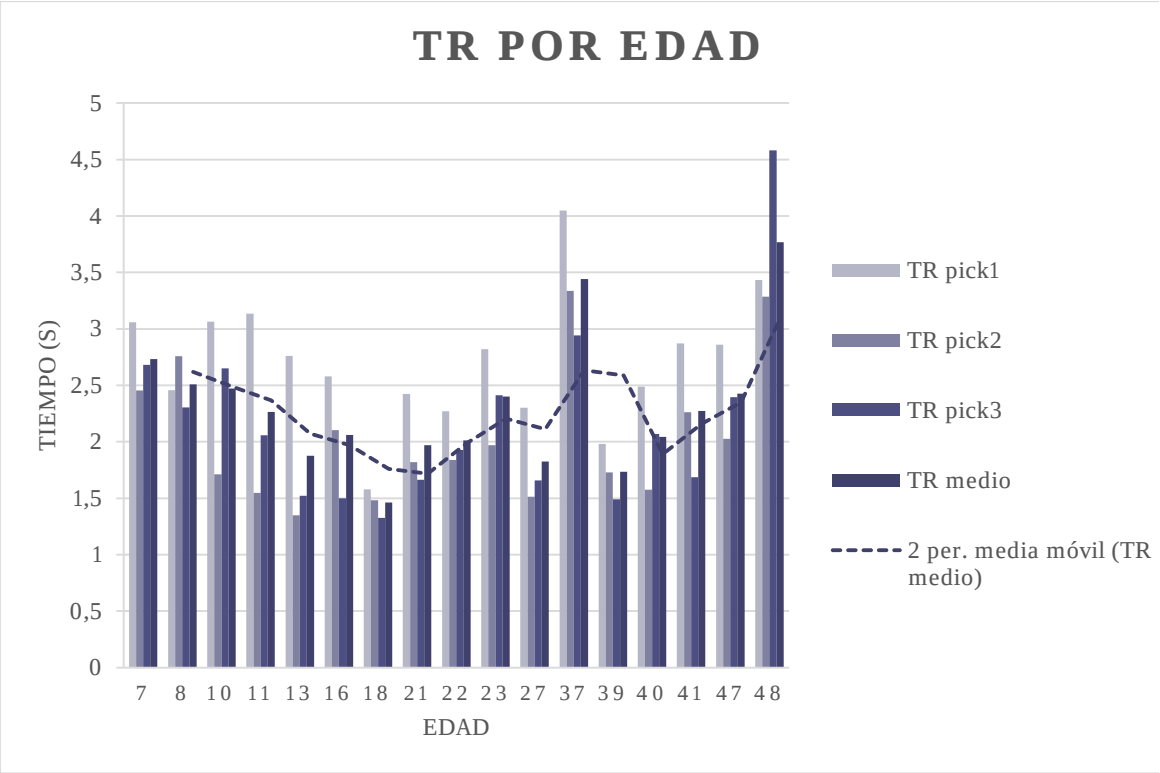


Figura 5-37: Gráfica que muestra que el TR se eleva conforme aumenta la edad en el experimento 2.

Y por último comprobamos si la preferencia de selección de la primera repisa ha cambiado.

numRepisa	COUNT(`numRepisa`)	Tasa de selección de repisa en el pick 1	COUNT(`numRepisa`)	Tasa de selección de repisa en el pick 2	COUNT(`numRepisa`)	Tasa de selección de repisa en el pick 3
0	138	0,286307054	227	0,47095436	288	0,59751037
1	344	0,713692946	255	0,52904564	194	0,40248963

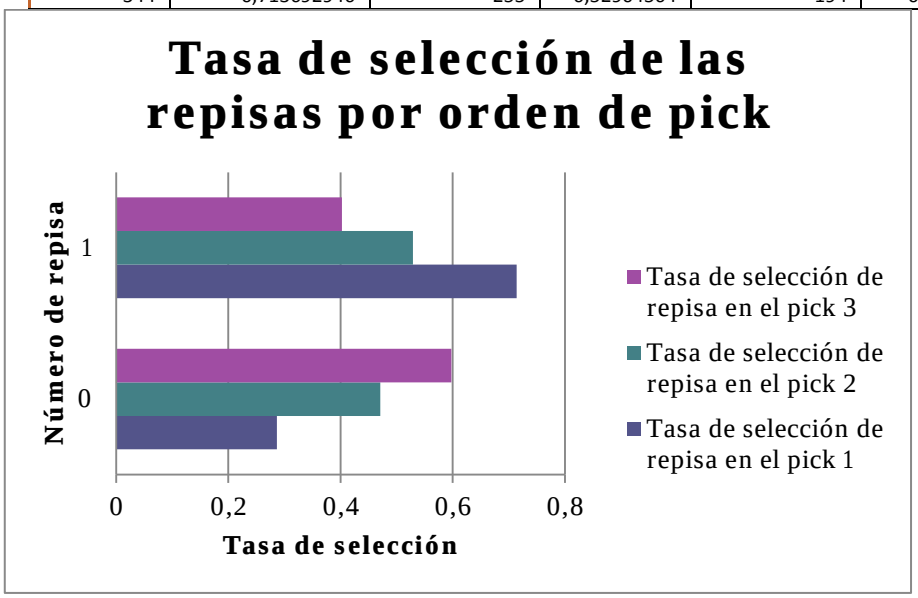


Figura 5-38: Gráfica que muestra la tasa de selección de las repisas de 20 sesiones de todos los participantes. Muestra un cambio en la preferencia de la repisa seleccionada en segundo lugar con respecto al experimento uno.

Como se observa en la gráfica la preferencia para la primera selección sigue siendo la repisa superior sin embargo en la segunda selección encontramos un cambio, ya que ambas repisas tienen una tasa de selección muy similar por lo que podemos identificar que el comportamiento de los participantes cambia al introducir la aleatoriedad.

5 Conclusiones y trabajos futuros

En este proyecto hemos realizado en primer lugar un estudio general de la problemática surgida al emplear las nuevas tecnologías en la realización de pruebas cognitivas en el ámbito de la psicología experimental. En particular se han analizado los inconvenientes derivados del uso del tiempo de reacción como variable dependiente en la realización de dichas pruebas.

A continuación hemos evaluado algunas herramientas software para implementar juegos serios que miden el tiempo de reacción. En concreto hemos valorado el nivel de conocimientos y la cantidad de recursos necesarios derivados del uso de tres plataformas concretas tales como *Flash*, *VRML/X3D*, y *Unity®*, para la implementación de una prueba clásica y relativamente simple como *Stroop*. Estas experiencias permitieron decidirnos por la tercera de las plataformas, *Unity®*, para el desarrollo de una aplicación bastante más compleja.

En concreto la aplicación implementada sirve para abordar el estudio sobre el acceso léxico de las personas y podría tener también un potencial uso en estudios sobre diagnóstico y prevención del deterioro cognitivo gracias a la facilidad de manipulación de las variables independientes.

Se contempla su uso como posible herramienta que ayude a la educación y la formación en psicología experimental ya que en última instancia, los usuarios de ésta tan sólo tendrán que recoger los datos con la aplicación y analizar los datos de la base de datos. Así mismo tiene como uno de sus objetivos ser una herramienta de apoyo en estudios de psicología por lo que no se pretende reemplazar al psicólogo sino más bien usar las bondades que un entorno en tres dimensiones y específicamente un juego serio puede aportar a dichos estudios.

Está creada con la intención de agilizar pruebas sobre tareas de decisión léxica de forma supervisada. Es el supervisor quien evaluará los parámetros del participante en cuanto al estado emocional, el ruido de la sala, etc. Si éste considera que las condiciones son adecuadas para realizar la sesión se procede a la realización de la prueba.

Este es uno de los motivos por los que la aplicación no se encuentra disponible para su uso libre a través de internet o para su descarga desde las tiendas en línea apropiadas. El otro motivo que se relaciona igualmente con los problemas derivados de no supervisar la realización de las pruebas es que se estaría permitiendo que cualquier persona elija cualquier edad y sexo para crear un nuevo participante, lo cual nos llevaría a obtener datos distorsionados ya que la información recogida no reflejaría fielmente las habilidades de los participantes.

Por ello las pantallas de la aplicación están creadas para facilitarle al supervisor la introducción de los datos así como una rápida realización de una sesión. El alias de los participantes así como las contraseñas pueden ser administrados por los supervisores y también recuperados de la base de datos.

La portabilidad a dispositivos móviles permite que los supervisores lleven la aplicación cómodamente a un entorno considerado apropiado para realizar la prueba.

En cuanto a las futuras ampliaciones que se podrían hacer consideramos la posibilidad de añadir a la plataforma (aplicación + servidor + base de datos) una capa nueva en la que se implementen las funciones necesarias para obtener valores medios, gráficas, ficheros *XML*, etc., directamente en la aplicación o en una página web *html* que lance las consultas a la base de datos y muestre las gráficas y los resultados, pero como habíamos comentado en los objetivos, el proyecto únicamente abordaba el almacenamiento y un breve análisis genérico de los datos mediante consultas *SQL*.

Otra posible ampliación derivada de la escalabilidad con la que ha sido creada la plataforma es la posibilidad de incorporar un apartado destinado a permitir al supervisor configurar el escenario en cuanto a número de libros, número de repisas, colores, orientación de los libros, activación del movimiento de la cámara, etc.

En cuanto a las modificaciones que podrían hacer más atractivo el juego serio encontramos la posibilidad de mostrar un gráfico que exprese la evolución de los tiempos de reacción medios al finalizar la sesión a modo de aliciente para el participante, ya que como pudimos comprobar en el segundo experimento, cuando el escenario es aleatorio el participante se siente motivado por reducir su TR. Esto podría ayudarnos a recopilar una mayor cantidad de datos ya que el interés del participante puede verse incrementado al querer mejorar los resultados.

Así mismo para aumentar el nivel de inmersión del participante así como para que los datos recogidos se vean expuestos aún menos por las interferencias externas a la prueba (ruido de la sala, distracciones, etc.), sería incorporar funcionalidades que permitan integrar gafas de realidad virtual con lo que tendríamos un entorno más controlado.

6 Bibliografía

D'Amico, S., Devescovi, A., & Bates, E. (2001). *Picture naming and lexical access in italian children and adults.* Recuperado el Julio de 2015, de CiteSeerX: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.139.6698&rep=rep1&type=pdf

E. Lincoln, C., & M. Lane, D. (Enero de 1980). *Reaction time measurement errors resulting from the use of CRT displays.* Recuperado el 25 de Junio de 2015, de Springer Link: http://link.springer.com/article/10.3758/BF03208326

Felix Navarro, K., Lawrence, E., Garcia Marin, J. A., & Sax, C. (2011). *"A Dynamic and Customisable Layered Serious Game Design Framework for Improving the Physical and Mental Health of the Aged and the Infirm.* Recuperado el 02 de Junio de 2015, de think Mind: http://www.thinkmind.org/index.php?view=instance&instance=eTELEMED+2011

Limpman, R., & NIST. (s.f.). *VRML Applications in Construction*. Recuperado el Agosto de 2015, de Purdue University. Division of construction engineering and management.: http://rebar.ecn.purdue.edu/ect/links/technologies/internet/vrml.aspx

Marcano, B. (3 de Noviembre de 2008). *Juegos serios y entrenamiento en la sociedad digital.* Recuperado el 30 de Mayo de 2015, de Revista Electrónica Teoría de la Educación: Educación y Cultura en la Sociedad de la Información. Vol. 9, nº 3. Universidad de Salamanca: http://www.usal.es/~teoriaeducacion/rev_numero_09_03/n9_03_marcano.pdf

Montañana Aliaga, J. M., & Lemus Zúñiga, L. G. (s.f.). *Brain Challenge Eval application on Android Market.* Obtenido de https://itunes.apple.com/es/app/id919963936?mt=8

NINDS. (12 de Febrero de 2010). *Demencias: Esperanza en la investigación*. Recuperado el 28 de Mayo de 2015, de National Institute of Neurological Disorders and Stroke: http://espanol.ninds.nih.gov/trastornos/las_demencias.htm

Nintendo. (25 de Junio de 2015). *Brain Training del Dr. Kawashima*. Recuperado el 26 de Junio de 2015, de Brain Training del Dr. Kawashima: ¿Cuántos años tiene tu cerebro?: https://www.nintendo.es/Juegos/Nintendo-DS/Brain-Training-del-Dr-Kawashima-Cuantos-anios-tiene-tu-cerebro--270627.html

OmniumGames. (1 de Septiembre de 2014). *Aprende a sobrevivir a una inundación con Sai Fah: The Fighter Flood*. Recuperado el 29 de Mayo de 2015, de Omnium Games The Game Journal: http://omniumgames.com/aprende-sobrevivir-una-inundacion-con-sai-fah-fighter-flood/

OMS. (2009). *World Population Ageing*. Recuperado el 29 de Mayo de 2015, de Organización Mundial de la Salud: http://www.un.org/esa/population/publications/WPA2009/WPA2009_WorkingPaper.pdf

Perea, M. (1999). *Tiempos de reacción y psicología cognitiva: Dos procedimientos para evitar el sesgo debido al tamaño muestral.* Recuperado el 01 de Junio de 2015, de Universistat de València: http://www.uv.es/~mperea/samplesize.pdf

Sánchez Gómez, M. (2007). *Buenas prácticas en la creación de serious games (Objetos de aprendizaje reutilizables).* Recuperado el 02 de Junio de 2015, de SPDECE '07 IV Simposio Pluridisciplinar sobre Diseño, Evaluación y Desarrollo de Contenidos Educativos Reutilizables: http://ceur-ws.org/Vol-318/

Schubert, T. W., Murteira, C., Collins, E. C., & Lopes, D. (21 de Junio de 2013). *ScriptingRT: A Software Library for Collecting Response Latencies in Online Studies of Cognition*. Recuperado el 28 de Junio de 2015, de Online Studies of Cognition. PLoS ONE 8(6): e67769. doi:10.1371/journal.pone.0067769: http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0067769

Sternberg, S. (1969). *The Discovery of Processing Stages. Extensions of Donders' Method*. Recuperado el 31 de Mayo de 2015, de Psychology Department. University of Pennsylvania: http://www.psych.upenn.edu/~saul/discoveryprocessingstages.pdf

Sternberg, S. (30 de Marzo de 2004). *Reaction-Time Experimentation*. Recuperado el 31 de Mayo de 2015, de Psychology Department. University of Pennsylvania: http://www.psych.upenn.edu/~saul/rt.experimentation.pdf

Sun Center of Excellence for Visual Genomics, U. o., & Department of Biological Sciences, U. o. (2009). *Spatiotemporal integration of molecular and anatomical data in virtual reality usign semantic mapping*. Recuperado el 02 de Junio de 2015, de http://www.dovepress.com/getfile.php?fileID=4602

Wikipedia. (11 de Junio de 2015). *Stroop Effect*. Obtenido de https://en.wikipedia.org/wiki/Stroop_effect

Zandt, T. V. (2000). *How to fit a response time distribution.* Recuperado el 01 de Junio de 2015, de Psychonomic Bulletin and Review, Vol. 7, No. 3, 01.09.2000, p. 424-462.: http://link.springer.com/article/10.3758/BF03214357