ESCUELA POLITÉCNICA SUPERIOR

UNIVERSIDAD DE ALMERÍA

DESARROLLO DE NUEVOS ENTORNOS VIRTUALES 3D PARA LA EVALUACIÓN DE MEMORIA ESPACIAL EN HUMANOS

Documento

PROYECTO FIN DE CARRERA

Para la obtención del título de

INGENIERO EN INFORMÁTICA

Autor

Armando Soria Albacete

Directores

Luis Iribarne Martínez

Antonio Moisés Espínola

ALMERÍA, 2009

Agradecimientos

En primer lugar me gustaría dar las gracias a José Manuel Cimadevilla Redondo y al Departamento de Neurociencia y Ciencias de la Salud. Sus estudios y su colaboración con el Departamento de Lenguajes y Computación, han hecho posible este proyecto.

También doy las gracias a Luis F. Iribarne Martínez y Antonio Moisés Espínola Pérez, mis directores de proyecto. Ellos me han guiado, ayudado y dado consejo durante la realización de este proyecto,

Por último, agradezco a Jésica su apoyo. Esos días de duro trabajo fueron más amenos gracias a tu compañía.

Armando Soria Albacete

Proyecto Fin de Carrera

Universidad de Almería. 2009

Prólogo

Este proyecto se enmarca dentro de un trabajo de investigación del Departamento de Neurociencia y Ciencias de la Salud de José Manuel Cimadevilla Redondo con el Departamento de Lenguajes y Computación.

El proyecto EVEMEH, Entornos Virtuales para la Evaluación de la Memoria Espacial en Humanos, es fruto de esa colaboración. Esta herramienta permite realizar simulaciones en entornos virtuales 3D, que posteriormente será usada para evaluar una capacidad muy importante del cerebro humano: la memoria espacial y la memoria de trabajo.

Este proyecto pretende crear otra herramienta que también permita realizar esas simulaciones. Se crearán nuevos entornos virtuales 3D, en los que también se podrá evaluar la memoria espacial y de trabajo.

Estos entornos virtuales 3D, estarán realizados con 3D Studio Max. Este potente programa tiene un uso muy extendido en el mundo del diseño en 3D y, además, permite exportar los modelos creados al formato adecuado para poder integrarlos posteriormente en nuestro entorno virtual 3D.

La interfaz gráfica del programa permitirá configurar varias opciones a la hora de la ejecución del programa, como pueden ser los objetos que aparecerán en la habitación o las posiciones de las zonas premiadas y guardar esas configuraciones. Tras la ejecución del programa se podrán ver los datos obtenidos además de las trayectorias que siguen los sujetos durante los experimentos.

El motor 3D usado se desarrollará utilizando Visual Studio y las APIs OpenGL y Fmod.

Palabras clave: Informática Gráfica, Motor 3D, Memoria espacial, Memoria de trabajo, OpenGL.

Contenido

Figuras

Tablas

Capítulo 1

Introducción y planteamiento

Resumen

Como en todo proyecto, antes de comenzar a desarrollarlo, se han de realizar una serie de estudios iniciales y estimaciones que se incluyen en este primer capítulo. El capítulo estará dividido a su vez en los siguientes apartados:

a) Identificación, planteamiento y justificación: En este apartado se realiza una investigación de los trabajos ya existentes, indicando referencias a la bibliografía, similitudes y extensiones que el proyecto hace de éstos.

b) Conceptos previos: En este apartado se lleva a cabo una descripción detallada de los conceptos relacionados con el proyecto, así como una descripción de aquellos programas software (CASE, IDE, algoritmos de propósito específico o general), hardware, modelos, técnicas y métodos relacionados con el proyecto.

c) Plan de proyecto: En este tercer apartado, se describirá la realización del plan de proyecto. Tras identificar los elementos y fuentes de información básicos, se realiza una descomposición funcional detallada y jerárquica del proyecto. Para la estimación de costes en cuanto duración y esfuerzo, se procederá al uso los modelos COCOMO básico e intermedio. Por último, para la planificación temporal, se usarán diagramas de Gantt y PERT.

1.1. Identificación, planteamiento y justificación

Muchos de los conocimientos y datos adquiridos en el laboratorio desde ensayos o estudios realizados con modelos animales respecto al conocimiento de la capacidad de memorización, el diagnóstico de lesiones cerebrales en su estructura responsable y su repercusión a nivel de conducta, eran imposibles de extrapolar a la investigación en seres humanos.

Dicha habilidad, la capacidad de memorización y orientación en el espacio, reside en el hipocampo, estructura cerebral cuyo funcionamiento es estudiado en modelos animales mediante técnicas comportamentales, donde las tareas de orientación espacial son las más utilizadas. Para ello, los expertos comparan las diferentes respuestas obtenidas frente a situaciones similares por roedores naturales y animales manipulados genéticamente, es decir, previamente se ha limitado la capacidad cerebral de la región hipocampal. Sin embargo, estos estudios no pueden ser aplicados en seres humanos por razones éticas obvias, por lo que su análisis es efectuado mediante técnicas neuropsicológicas clásicas.

Esta barrera, gracias al desarrollo de las nuevas tecnologías y sistemas de la información, se hace cada día más salvable. Existen numerosos casos en los que la informática sirve de herramienta para facilitar la labor de los investigadores a la hora de intentar entender el funcionamiento de la mente humana y, de esa manera, intentar comprender el por qué de los trastornos que se pueden sufrir y ayudar a curarlos o mitigar sus efectos.

Como ejemplos de algunos de estos casos, se pueden citar:

a) Uso de la realidad virtual para a ayudar a los pacientes a superar fobias y trastornos alimentarios.

b) Entornos virtuales inteligentes utilizados para el aprendizaje.

c) Entornos destinados a la mejora del tratamiento contra la violencia de género.

Otro ejemplo claro es la herramienta desarrollada por José Manuel Cimadevilla Redondo, miembro del departamento de Neurociencia y Ciencias de la Salud, en colaboración con el Departamento de Lenguajes y Computación, dirigido por Luis Fernando Iribarne Martínez.

El estudio comenzó como una colaboración con el doctor Robert Astur, de la Universidad de Yale (Conneticut), quien desarrolló la primera tarea de realidad virtual aplicada a la evaluación de la memoria espacial humana, basada en laberintos clásicos utilizados en experimentación animal como idea original.

Sin embargo, hubo que desarrollar una aplicación propia que se ajustara mejor a las necesidades de los estudios del grupo de investigación. Así fue cómo surgió la aplicación EVEMEH.

EVEMEH, acrónimo de Entornos Virtuales para la Evaluación de la Memoria Espacial en Humanos, es una aplicación que permite implementar en humanos las tareas efectuadas por los modelos animales en el laboratorio. Para ello, se crea un entorno virtual 3D que permite conocer cómo se comporta el ser humano en tareas que demandan memoria espacial. Cabe mencionar que la pérdida de orientación espacial es uno de los rasgos definitorios de la lesión hipocampal, poniéndose de manifiesto en las dificultades que algunos de nuestros mayores y pacientes con demencia demuestran en el día a día.

Este programa presenta una habitación cuadrada con una serie de objetos en las paredes que sirven de estímulos a la hora de orientarse. En el suelo de la habitación, aparecerán una serie de cofres que estará premiados o no. Para la evaluación y estudio de la capacidad y funcionamiento de la región hipocampal, el sujeto ha de recorrer la habitación para ir abriendo los cofres y encontrar los que estén premiados. En un primer ensayo, es de esperar tener que abrir todos los cofres. En los siguientes ensayos, han de recordarse las posiciones de los cofres premiados en función de los estímulos de las paredes, intentando cometer el menor número de errores posible.

Existen una serie de modificaciones sobre el proyecto EVEMEH. En ellas se eliminan estímulos de las paredes, hasta dejar 3, 2 o 1 estimulo en la habitación. Con esto se pretende estudiar la

memoria espacial de los sujeto con un número menor de estímulos que puedan usarse a la hora de orientarse y recordar donde se esconden las zonas premiadas.

Otra modificación existente sobre este programa elimina todos los estímulos existentes en las paredes de la habitación y coloca 3 pivotes de colores en medio de las cajas colocados formando un triangulo. En este caso, se estudia otra característica de la memoria espacial, ya que la orientación basándose en estímulos que hay en la pared y la orientación basándose en estímulos situándose sobre el suelo, utilizan distintas zonas y mecanismos del cerebro.

Recientemente se ha publicado un artículo [Cimadevilla *et al.*, 2008] en el que se usa el programa EVEMEH para evaluar la memoria espacial en 63 alumnos (30 chicos y 33 chicas). Tenían que encontrar 3, 5 y 7 cofres premiados de entre 16 en el menor tiempo posible durante 10 ensayos. Los resultados arrojan que los sujetos aprendieron la tarea, pero con diferentes cantidades de errores dependiendo de la dificultad de los ensayos. Otro dato de interés, es que las mujeres eran más lentas y con menos precisión que los hombres.

Continuando esta línea de investigación, surge la necesidad de recrear nuevos entornos virtuales y nuevas formas de seguir estudiando la capacidad y funcionamiento de la región hipocampal en pacientes humanos. Así es como surge este proyecto.

Además de toda la base comentada anteriormente, la otra base de este proyecto es la informática gráfica, y como tal, se ve influenciado por varias áreas de la misma. De todas las áreas, las que influyen con más peso son:

a) Diseño en 3D

b) Realidad virtual.

c) Motores 3D.

En los siguientes puntos se hablará de éstas áreas, definiéndolas, explicando cómo influyen en el proyecto y nombrando y explicando algunas herramientas pertenecientes a cada área.

1.1.1. Diseño en 3D

El término gráficos 3D por ordenador se refiere a trabajos de arte gráfico que fueron creados con ayuda de ordenadores y programas especiales 3D. En general, el término puede referirse también al proceso de crear dichos gráficos, o el campo de estudio de técnicas y tecnología relacionadas con los gráficos 3D.

Un gráfico 3D difiere de uno 2D principalmente por la forma en que ha sido generado. Este tipo de gráficos se origina mediante un proceso de cálculos matemáticos sobre entidades geométricas tridimensionales producidas en un ordenador, y cuyo propósito es conseguir una proyección visual en dos dimensiones para ser mostrada en una pantalla o impresa en papel.

En general, el arte de los gráficos 3D es similar a la escultura o la fotografía, mientras que el arte de los gráficos 2D es análogo a la pintura. En los programas de gráficos por ordenador esta distinción es a veces difusa: algunas aplicaciones 2D utilizan técnicas 3D para alcanzar ciertos efectos como iluminación, mientras que algunas aplicaciones 3D primarias hacen uso de técnicas 2D.

Los siguientes puntos detallan algunas herramientas del mercado destinadas a la creación y manejo de gráficos 3D, las cuales quedan resumidas en la Tabla 1.1:

Nombre	**Características principales**
LightWave 3D	Completo sistema de modelado, renderizado y animación. Se divide en *Modeler* y *Layout*. *Modeler* realiza el modelado del objeto. *Layout* renderizar, configura las cámaras, luces, etc.
Blender	Modelado y creación de gráficos tridimensionales. Distribución de menús y cámaras personalizados. Multiplataforma, libre, gratuito. Incluye motor de juegos 3D integrado.
AutoCAD	Parte orientada a la producción de planos. Conceptos de *espacio modelo* y *espacio papel* para separar las fases de diseño y dibujo en 2D y 3D. Permite exportar en varios formatos *.dwg*, *.dxf*
3D Studio Max	Creación de gráficos y animaciones 3D. Compatible solo con plataformas Windows. Muy usado en la creación de videojuegos y películas.

Tabla 1.1. Resumen de herramientas para la creación de gráficos 3D.

1.1.1.1. LightWave 3D

LightWave 3D [LightWave] es un completo sistema de modelado, renderizado y animación. Usado extensamente en producciones televisivas, efectos especiales en películas, desarrollo de videojuegos, impresión de gráficos y visualización, LightWave es el responsable de que muchos artistas hayan ganado Premios Emmy, muchos más que con cualquier otra aplicación 3D.

LightWave 3d se divide en dos subprogramas: *Modeler* y *Layout*; En *Modeler* se realiza el modelado del objeto con una filosofía orientada a capas; cada capa es una única malla, a diferencia

de otros paquetes de modelado, en LightWave no existen los objetos sino una malla generalizada en cada capa.

En *Layout* se realiza el *Rigging* o configuración del esqueleto de una malla, Dinámicas, FX, Render, configuración de cámaras y luces.

El editado de materiales y texturas se realiza tanto en *Modeler* como en *Layout* mediante el menú "Surface Editor".

Modeler y *Layout* están ligados en funcionamiento por medio de un programa llamado HUB, por ejemplo si tengo cargado en *Layout* el objeto hecho en *Modeler*, puedo modificarlo desde *Modeler* y los cambios aparecen en *Layout* automáticamente.

A continuación, la Figura 1.1 muestra una captura de *Modeler* de LightWave:

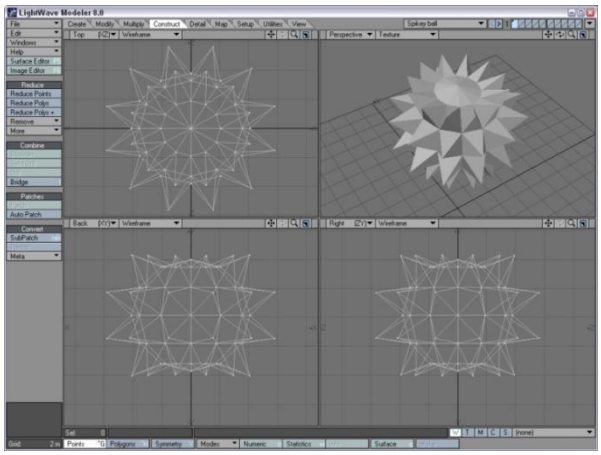


Figura 1.1. Captura de *Modeler* de LightWave 3D.

1.1.1.2. Blender

Blender es un programa multiplataforma, dedicado especialmente al modelado y creación de gráficos tridimensionales. Actualmente es compatible con todas las versiones de Microsoft Windows, Mac OS X, Linux, Solaris, etc.

Tiene una muy peculiar interfaz gráfica de usuario, que se critica como poco intuitiva, pues no se basa en el sistema clásico de ventanas; pero tiene a su vez ventajas importantes sobre éstas, como la configuración personalizada de la distribución de los menús y vistas de cámara.

Como características destacadas de Blender encontramos que es multiplataforma, libre, gratuito, con capacidad para una gran variedad de primitivas geométricas, incluyendo curvas, mayas poligonales, herramientas de animación, edición de audio y video, características interactivas para juegos como detección de colisiones, recreaciones dinámicas y lógicas. Puede descargarse desde su página web oficial [Blender].

Además, Blender acepta formatos gráficos como TGA, JPG, Iris, SGI, o TIFF. También puede leer ficheros *Inventor*. Incluye un motor de juegos 3D integrado, con un sistema de ladrillos

lógicos. Para más control se usa programación en lenguaje *Python*. Permite realizar simulaciones dinámicas para *softbodies*, partículas y fluidos. Incorpora también un sistema de partículas estáticas para simular cabellos y pelajes, al que se han agregado nuevas propiedades entre las opciones de *shaders* para lograr texturas realistas.

La Figura 1.2 muestra una captura del interfaz de Blender:

Figura 1.2. Captura de la interfaz de Blender.

1.1.1.3. AutoCAD

Autodesk AutoCAD es un programa de diseño asistido por ordenador para dibujo en 2D y 3D. Actualmente es desarrollado y comercializado por la empresa Autodesk.

Parte del programa AutoCAD está orientado a la producción de planos, empleando para ello los recursos tradicionales de grafismo en el dibujo, como color, grosor de líneas y texturas tramadas. AutoCAD, a partir de la versión 11, utiliza el concepto de *espacio modelo* y *espacio papel* para separar las fases de diseño y dibujo en 2D y 3D, de las específicas para obtener planos trazados en papel a su correspondiente escala. La extensión del archivo de AutoCAD es *.dwg*, aunque permite exportar en otros formatos (el más conocido es el *.dxf*). Maneja también los formatos *IGES* y *STEP* para manejar compatibilidad con otro software de dibujo.

Se puede encontrar más información en la web oficial [AutoCAD].

La interfaz de AutoCAD puede verse en la Figura 1.3:

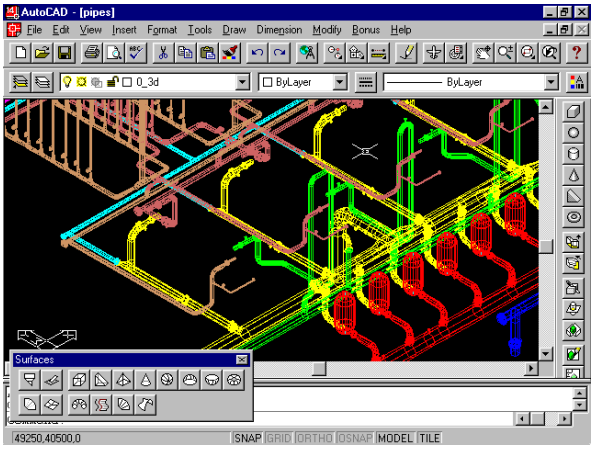


Figura 1.3. Captura del programa AutoCAD.

1.1.1.4. 3D Studio Max

A continuación, se profundiza en el programa usado en este proyecto para la creación de los modelos en 3D que forman parte del entorno virtual en el que se desarrollan los experimentos.

Autodesk 3D Studio Max [3DSMax] es un programa de creación de gráficos y animación 3D desarrollado por Autodesk Media & Entertainment. 3D Studio Max es uno de los programas de animación 3D más utilizados. Dispone de una sólida capacidad de edición, una omnipresente arquitectura de *plugins* y una larga tradición en plataformas Microsoft Windows. 3D Studio Max es utilizado en mayor medida por los desarrolladores de videojuegos, aunque también en el desarrollo de proyectos de animación como películas o anuncios de televisión, efectos especiales y en arquitectura. Este programa es uno de los mejores modeladores 3D masivo, más orientado a videojuegos, con el que se han hecho enteramente juegos como la saga *Tomb Raider* y la saga *Splinter Cell*, y la mayoría de los juegos de *Ubisoft*. Incluido el magnífico *Crysis* y *FarCry*.

Características destacadas:

a) Renderización con *Reveal*.

El sistema de renderización *Reveal*, nuevo en 3D Studio Max 2009, proporciona el control exacto necesario para refinar rápidamente las renderizaciones. Puede renderizar una escena entera excepto un determinado objeto, renderizar un único objeto o incluso una región específica del búfer de fotograma.

b) Mejoras de *Biped*.

En el nuevo flujo de trabajo de *Biped*, las manos de los personajes bípedos pueden comportarse como pies en relación con el plano del suelo. Esta nueva característica abrevia drásticamente los pasos necesarios para crear animaciones de cuadrúpedos.

c) Edición rápida de texturas UV.

Autodesk 3ds Max sigue liderando el mercado con avanzadas herramientas de mapeado fáciles de usar. La nueva función de mapeado de *spline* permite mapear objetos tubulares y tipo *spline*, como una carretera sobre un terreno. Además, las mejoras en los flujos de trabajo *Pelt* y *Relax* agilizan el desajuste UVW, para alcanzar el resultado deseado con menos pasos.

d) Compatibilidad con .NET en el SDK.

La compatibilidad con .NET en 3D Studio Max permite utilizar las eficaces API de interfaz de usuario de alto nivel de Microsoft para ampliar el software. El SDK de 3D Studio Max incluye código .NET de ejemplo y documentación que enseña cómo aprovechar este potente conjunto de herramientas de desarrollo.

e) *ProMaterials.*

3ds Max cuenta con una nueva biblioteca de materiales basados en la física y fáciles de utilizar para *mental ray*®. Con ellos se crean rápidamente superficies de construcción y diseño de uso frecuente, como pintura de pared (mate o brillo), vidrio y hormigón con acabado profesional.

f) Mejoras de iluminación fotométrica.

Autodesk 3ds Max admite nuevos tipos de luces de área (circular, cilíndrica), pre visualizaciones de red fotométrica en el cuadro de diálogo de examinar y en la interfaz de iluminación, así como mejor distribución focal y calidad fotométrica de campo próximo.

La Figura 1.4 ilustra la interfaz principal de 3D Studio:

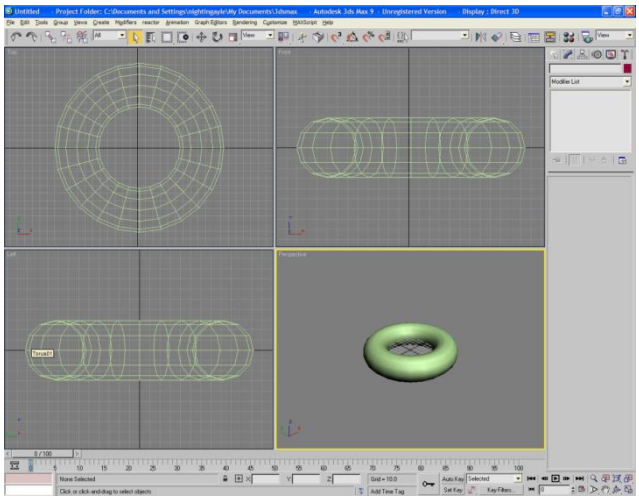


Figura 1.4. Captura de la interfaz principal de 3D Studio Max.

1.1.2. Realidad virtual.

Una definición de realidad virtual podría ser:

"Aquella tecnología informática que genera entornos tridimensionales con los que el sujeto interactúa en tiempo real, produciéndose de esa manera una sensación de inmersión semejante a la de presencia en el mundo real.", [Gutiérrez, 2002].

Tales entornos no tienen por qué limitarse a la reproducción de la realidad, sino que pueden construirse simulando casos imposibles en la realidad, que se adecúen a las necesidades concretas del entorno del estudio.

Las aplicaciones de la realidad virtual se hallan presentes en la actualidad en gran cantidad de campos, desde el entretenimiento hasta las simulaciones con las que se entrena en el manejo de diferentes máquinas, pasando por sistemas de visualización que ayudan a comprender complejos sistemas conceptuales matemáticos.

Los ejércitos cuentan desde hace décadas con simuladores de realidad virtual, en los que se entrena a los soldados en el manejo de la maquinaria bélica o en los que se simulan entornos en los que los soldados se ven en situaciones de gran tensión similares a las que pueden encontrar durante el trascurso de las posibles situaciones en las que puedan verse inmersos.

En medicina se dispone desde hace ya varios años de sistemas de realidad virtual mediante los que es posible aprender y practicar técnicas quirúrgicas, eliminando los riesgos de practicar con pacientes vivos y adaptando las condiciones en las que se realizan, simulando situaciones de tensión.

Hablemos ahora de psicología, que es el campo que nos interesa puesto que el fin último de este proyecto es el uso, por parte de psicólogos e investigadores, de la informática en el estudio de la memoria espacial y de trabajo.

Como se comentaba en la justificación de este proyecto, existen numerosos entornos virtuales que se usan en diferentes áreas de la psicología. A continuación se comentan algunos estudios que hacen uso de entornos virtuales aplicados a la psicología:

a) Tratamiento de la ansiedad ante los exámenes mediante la exposición a entornos de realidad virtual. En este estudio, se crearon 3 entornos virtuales, la casa del alumno, un medio de transporte y la universidad, en los que se producían conversaciones entre alumnos virtuales. Tras la exposición a estos entornos se medía la ansiedad subjetiva, la ansiedad-estado y el estado de ánimo deprimido. Tras realizar el tratamiento, se comprobó como los alumnos reducían su ansiedad e incluso algunos aumentaban su rendimiento académico, [Gutiérrez, 2002].

b) Realidad virtual en el tratamiento del miedo claustrofóbico. Este estudio intenta determinar la efectividad del uso de entornos virtuales en el tratamiento del miedo claustrofóbico. El tratamiento consiste en una serie de exposiciones en los entornos virtuales controladas. Tras el tratamiento, se

concluyó que el uso de estos entornos disminuía el miedo y la angustia en entornos cerrados de los pacientes y aumentaba la auto eficacia en entornos claustrofóbicos, [Botella *et al.*, 2000].

c) Realidad virtual en el tratamiento del miedo a volar: En este estudio, se somete a los pacientes a una serie de exposiciones en las que se muestra una ciudad vista desde el aire. Al principio de las sesiones, los niveles de ansiedad eran altos, pero disminuían progresivamente tras los primeros minutos de exposición, hasta llegar al valor 0, [Gutiérrez, 2002].

d) Realidad virtual en el tratamiento del miedo a hablar en público: se recreó un auditorio virtual con capacidad para 100 personas. Los pacientes participaban en una sesión semanal de 15 minutos durante 5 semanas en la que se veían enfrentados a su miedo. Tras las sesiones, se comprobó que las medidas subjetivas de ansiedad se redujeron y mejoraron las actitudes hacia situaciones en las que era necesario hablar en público [North *et al.*, 1998].

e) Realidad virtual en el tratamiento de estrés postraumático para víctimas de accidentes de tráfico: Se crearon entornos virtuales que simulaban la conducción para intentar reducir ese miedo. Se permite controlar aspectos climatológicos y condiciones de tráfico sin el riesgo que supone la conducción real. Se demostró que los resultados eran persistentes realizando las mismas simulaciones sobre pacientes 7 meses después [Wald *et al.*, 2000].

f) Realidad virtual en el tratamiento de aracnofobia: Se somete al paciente a una serie de ensayos, separados por un margen de tiempo, en los que se le anima a interactuar con una araña en un entorno virtual. El paciente interactúa a su ritmo, y hasta que no se alcanzaban niveles bajos de ansiedad, no se pasaba a situaciones más amenazadoras [Carling *et al.*, 1997] [Goetestam *et al.*, 1996] [Hollup SA, 1996].

Como curiosidad, cabe destacar un juego llamado LevelHead, un juego en el que se puede probar la memoria espacial 3D del jugador, mediante una innovadora interfaz. Si bien, aunque su finalidad no es estudiar ningún aspecto del funcionamiento de la memoria espacial humana, no cabe duda que es una aplicación original de la informática relacionada con la memoria espacial.

Funciona con una webcam y un pequeño cubo. Colocando el cubo frente a una webcam, se ve en la pantalla del ordenador cómo las caras del cubo muestran diferentes vistas 3D de un laberinto, compuesto por seis habitaciones con puertas y escaleras que las comunican. Se tienen 2 minutos para conseguir que un personaje virtual recorra el laberinto hasta la salida. Para conducir el personaje a través de las habitaciones, se debe girar el cubo frente a la webcam. En su página oficial, [LevelHead], se pueden ver vídeos del juego en cuestión.

1.1.3. Motores 3D.

Un motor 3D es una colección de estructuras, funciones y algoritmos utilizados para visualizar, después de realizar cálculos y transformaciones, objetos tridimensionales en una pantalla bidimensional.

Las secciones principales de un motor 3D son:

a) Ubicación de los datos de los objetos en estructuras.

b) Las transformaciones para posicionar los objetos en el mundo.

c) Renderizar la escena en una pantalla bidimensional.

Existen multitud de estándares y APIs que proporcionan la funcionalidad necesaria para la creación de un motor 3D, como pueden ser funciones para dibujar puntos, triángulos, etc., funciones para hacer rotaciones y traslaciones, etc.

En los siguientes apartados, se comentan algunos de los principales motores 3D existentes en el mercado.

1.1.3.1. OpenGL

OpenGL (Open Graphics Library) es una especificación estándar que define una API multilenguaje y multiplataforma para escribir aplicaciones que produzcan gráficos 2D y 3D. La interfaz consiste en más de 250 funciones diferentes que pueden usarse para dibujar escenas tridimensionales complejas a partir de primitivas geométricas simples, tales como puntos, líneas y triángulos. Fue desarrollada originalmente por Silicon Graphics Inc. (SGI) en 1992 y se usa ampliamente en CAD, realidad virtual, representación científica, visualización de información y simulación de vuelo. También se usa en desarrollo de videojuegos, donde compite con Direct3D en plataformas Microsoft Windows.

Fundamentalmente OpenGL [OpenGL] es una especificación, es decir, un documento que describe un conjunto de funciones y el comportamiento exacto que deben tener. Partiendo de ella, los fabricantes de hardware crean implementaciones, que son bibliotecas de funciones que se ajustan a los requisitos de la especificación, utilizando aceleración hardware cuando es posible. Dichas implementaciones deben superar unos test de conformidad para que sus fabricantes puedan calificar su implementación como conforme a OpenGL y para poder usar el logotipo oficial de OpenGL.

OpenGL tiene dos propósitos esenciales:

a) Ocultar la complejidad de la interfaz con las diferentes tarjetas gráficas, presentando al programador una API única y uniforme.

b) Ocultar las diferentes capacidades de las diversas plataformas hardware, requiriendo que todas las implementaciones soporten la funcionalidad completa de OpenGL (utilizando emulación software si fuese necesario).

El funcionamiento básico de OpenGL consiste en aceptar primitivas tales como puntos, líneas y polígonos, y convertirlas en píxeles. Este proceso es realizado por una pipeline gráfica conocida como la Máquina de estados de OpenGL. La mayor parte de los comandos de OpenGL o bien emiten primitivas a la pipeline gráfica o bien configuran cómo la pipeline procesa dichas

primitivas. Hasta la aparición de la versión 2.0 cada etapa de la pipeline ejecutaba una función prefijada, resultando poco configurable. A partir de la versión 2.0 algunas etapas son programables usando un lenguaje de programación llamado GLSL.

OpenGL es una API basada en procedimientos de bajo nivel que requiere que el programador dicte los pasos exactos necesarios para renderizar una escena. Esto contrasta con las API descriptivas, donde un programador sólo debe describir la escena y puede dejar que la biblioteca controle los detalles para representarla. El diseño de bajo nivel de OpenGL requiere que los programadores conozcan en profundidad la pipeline gráfica, a cambio de darles libertad para implementar algoritmos gráficos novedosos.

OpenGL ha influido en el desarrollo de las tarjetas gráficas, promocionando un nivel básico de funcionalidad que actualmente es común en el hardware comercial; algunas de esas contribuciones son:

a) Primitivas básicas de puntos, líneas y polígonos rasterizados.

b) Una pipeline de transformación e iluminación.

c) *Z-buffering*.

d) Mapeado de texturas.

e) *Alpha blending*.

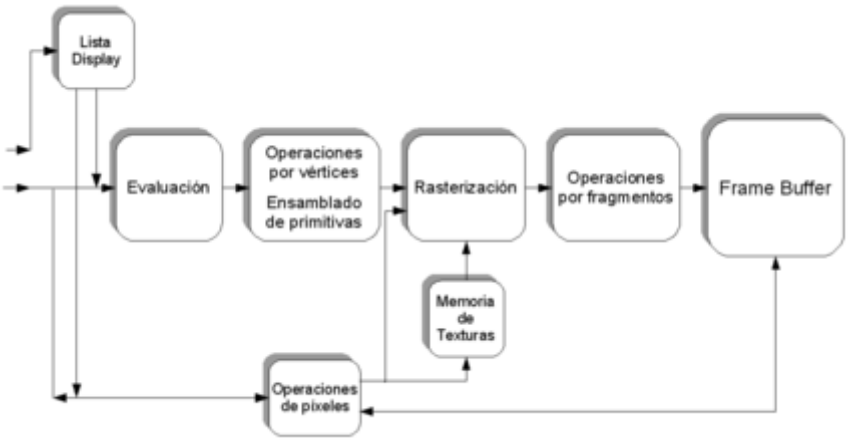


Figura 1.5. Proceso en la pipeline de gráficos.

Una descripción somera del proceso en la pipeline gráfica, visto en la Figura 1.5, podría ser:

1) Evaluación, si procede, de las funciones polinomiales que definen ciertas entradas, como las superficies *NURBS*, aproximando curvas y la geometría de la superficie.

2) Operaciones por vértices, transformándolos, iluminándolos según su material y recortando partes no visibles de la escena para producir un volumen de visión.

3) Rasterización, o conversión de la información previa en píxeles. Los polígonos son representados con el color adecuado mediante algoritmos de interpolación.

4) Operaciones por fragmentos o segmentos, como actualizaciones según valores venideros o ya almacenados de profundidad y de combinaciones de colores, entre otros.

5) Por último, los fragmentos son volcados en el *Frame buffer*.

Muchas tarjetas gráficas actuales proporcionan una funcionalidad superior a la básica aquí expuesta, pero las nuevas características generalmente son mejoras de esta pipeline básica más que cambios revolucionarios de ella.

Bibliotecas de utilidades

Se han programado varias bibliotecas externas que añaden características no disponibles en el propio OpenGL. Algunas de ellas son:

1) ***GLU***: GLU es el acrónimo de OpenGL Utility. Esta biblioteca está compuesta por una serie de funciones de dibujo de alto nivel que, a su vez, se basan en las rutinas primitivas de OpenGL y se suele distribuir normalmente junto a él.

Entre sus características podemos encontrar el mapeado entre pantalla y coordenadas, generación de texturas *mipmap*, dibujado de superficies cuádricas, NURBS, texelación de primitivas poligonales, interpretación de códigos de error de OpenGL, gran cantidad de rutinas de transformación para la creación de volúmenes de visualización y posicionado simple de la cámara, habitualmente de manera más sencilla que las rutinas que ofrece OpenGL. También posee primitivas para utilizar en aplicaciones OpenGL, como esferas, cilindros y discos.

Las funciones de GLU se reconocen con facilidad ya que todas comienzan con el prefijo glu. Por ejemplo *gluOrtho2D()*, que define una matriz en proyección ortográfica de dos dimensiones.

2) ***GLUT***: GLUT, OpenGL Utility Toolkit, es una librería de utilidades para programas OpenGL que principalmente proporciona diversas funciones de entrada/salida con el sistema operativo. Entre las funciones que ofrece se incluyen declaración y manejo de ventanas y la interacción por medio de teclado y ratón. También posee rutinas para el dibujado de diversas primitivas geométricas (tanto sólidas como en modo *wireframe*) que incluyen cubos, esferas y teteras. También tiene soporte para creación de menús emergentes.

Los dos objetivos de GLUT son para permitir la creación de código más portable entre diferentes sistemas operativos (GLUT es multiplataforma) y hacer OpenGL más simple. Introducirse en la programación con OpenGL utilizando GLUT conlleva normalmente sólo unas pocas líneas de código y hace innecesario el conocimiento de las API específicas de cada sistema operativo.

Todas las funciones de GLUT comienzan con el prefijo glut (por ejemplo, *glutPostRedisplay* indica que la ventana actual necesita ser redibujada).

3) ***GLUI***: Interfaz de usuario basada en GLUT. Proporciona elementos de control tales como botones, cajas de selección y *spinners*. Es independiente del sistema operativo, sustentándose en GLUT para manejar los elementos dependientes del sistema.

1.1.3.2. Direct3D

Direct3D es parte de DirectX [DirectX], una API propiedad de Microsoft disponible tanto en los sistemas Windows de 32 y 64 bits, como para sus consolas Xbox y Xbox 360 para la programación de gráficos 3D.

El objetivo de esta API es facilitar el manejo y trazado de entidades gráficas elementales, como líneas, polígonos y texturas, en cualquier aplicación que muestre gráficos en 3D, así como efectuar de forma transparente transformaciones geométricas sobre dichas entidades. Direct3D provee también una interfaz transparente con el hardware de aceleración gráfica.

Se usa principalmente en aplicaciones donde el rendimiento es fundamental, como los videojuegos, aprovechando el hardware de aceleración gráfica disponible en la tarjeta gráfica. El principal competidor de Direct3D es OpenGL, desarrollado por Silicon Graphics Inc.

Direct3D es uno de los múltiples componentes que contiene la API DirectX de Windows. Se le podría situar al nivel del GDI de Windows, presentando un nivel de abstracción entre una aplicación de gráficos 3D y los drivers de la tarjeta gráfica (véase gráfico adjunto). Con arquitectura basada en el COM de Microsoft, la mayor ventaja que presenta Direct3D frente al GDI es que Direct3D se comunica directamente con los drivers de pantalla, consiguiendo mejores resultados en la representación de los gráficos por pantalla que aquel.

En la Figura 1.6, se observa cómo Direct3D se sitúa entre la aplicación y los controladores del hardware.

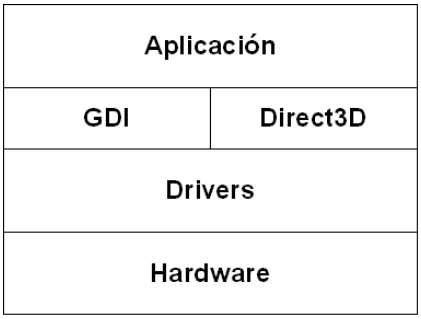


Figura 1.6. Esquema de funcionamiento de Direct3D.

Direct3D está compuesto por dos grandes APIs. El modo retenido y el modo inmediato. El modo inmediato da soporte a todas las primitivas de procesamiento 3D que permiten las tarjetas gráficas (*luces*, *materiales*, *transformaciones*, *control de profundidad*, etc). El modo retenido,

construido sobre el anterior, presenta una abstracción de nivel superior ofreciendo funcionalidades pre construidas de gráficos como jerarquías o animaciones. El modo retenido ofrece muy poca libertad a los desarrolladores, siendo el modo inmediato el que más se usa.

El modo inmediato de Direct3D trabaja fundamentalmente con los llamados dispositivos. Son los encargados de realizar la renderización de la escena. El dispositivo ofrece un interfaz que permite diferentes opciones de renderización. Por ejemplo un dispositivo *mono* permite la renderización en blanco y negro mientras que un dispositivo RGB permite el uso de colores.

Cada dispositivo tiene asociada una o más cadenas de intercambio o *swap chains*. Dichas cadenas están compuestas por varios buffers de superficies, considerando a una superficie como un conjunto de píxeles más todos los atributos asociados a cada uno de ellos como la profundidad, el color, la transparencia (canal alfa), etc.

La Figura 1.7, ilustra gráficamente las diferentes etapas del proceso de renderización:

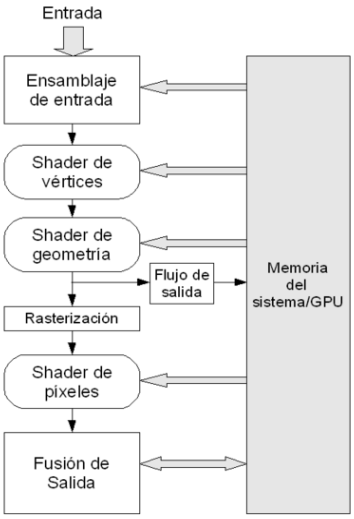


Figura 1.7. Etapas del proceso de renderización de Direct3D.

A continuación se explica cada una de las etapas del proceso de renderizado:

a) Ensamblaje de entrada: aporta los datos de entrada (líneas, puntos y triángulos).

b) Shader de vértices: se encarga de las operaciones de vértices (iluminación, texturas, transformaciones). Trata los vértices individualmente.

c) Shader de geometría: realiza operaciones con entidades primitivas (líneas, triángulos o vértices). A partir de una primitiva, el *shader de geometría* puede descartarla, o devolver una o más primitivas nuevas.

d) Flujo de salida: almacena la salida de la etapa anterior en memoria. Resulta útil para realimentar la pipeline con datos ya calculados.

e) Rasterización: convierte la imagen 3D en píxeles.

f) Shader de pixeles: operaciones con los píxeles.

g) Fusión de salida: se encarga de combinar la salida del *shader de pixeles* con otros tipos de datos, como los patrones de profundidad, para construir el resultado final.

Direct3D permite la reconfiguración de todas las etapas, aumentando considerablemente la flexibilidad de esta pipeline.

Las últimas versiones desarrolladas son:

a) **Direct3D 9.0** añadió una nueva versión del High Level Shader Language, soporte para HDR, renderización de múltiples objetos e indexación del buffer de vértices.

b) **Direct3D 10** consigue un aumento considerable en el rendimiento eliminando el llamado object overhead, que consiste en realizar llamadas entre la API y el driver usando la CPU. Dichas llamadas empezaban a producir un cuello de botella debido al incremento en la complejidad de las escenas y se han incluido medidas como los stateblocks de los shaders para poder referenciar grupos de variables del shader a través de un único ID en vez de enviar una a una entre otras cosas. Direct3D 10 incorpora además soporte del Shader Model 4, lo que significa que se puede hacer uso de los shaders de geometría. El principal problema de DirectX 10 es que sólo funciona con el sistema operativo Windows Vista, mientras que DirectX 9.0 trabaja con toda la familia de Windows a partir de Windows 98, debido a esto se rumorea que Microsoft planea o bien soportarlo en XP o bien sacar una versión especial de DirectX 9 que implemente algunas características. El SDK de DirectX 10 está disponible desde Febrero de 2007.

1.1.3.3. Renderman

Renderman Interface Specificacion, RISpec, es una API desarrollada por Pixar para transformar las escenas tridimensionales de sus producciones en imágenes digitales de alta calidad. Existe un protocolo de comunicación entre el programa de modelado y el de renderizado, necesario para poder comprender los comandos del programa y traducirlos a una imagen fotorealista.

Renderman [Renderman] es usado principalmente de dos formas:

a) Para crear largometrajes mediante la técnica de animación por ordenador, como en el caso de Toy Story, Buscando a Nemo o Monstruos S.A.

b) Para rodar fragmentos de mayor o menor duración en largometrajes de no animación. Estos fragmentos son, principalmente, efectos visuales digitales como los de Starwars o El Señor de los Anillos.

1.1.3.4. XNA

XNA Game Studio permite a estudiantes y a programadores de juegos, aficionado a crear juegos que usando .NET funcionen en Windows y su Xbox 360. El sistema XNA Framework es el sistema de las bibliotecas de .NET con las cuales los programadores disfrutaran construir sus juegos. Está disponible la primera versión beta desde agosto 30 del 2006.

Plataforma cruzada de desarrollo.

El objetivo fundamental es hacer fácil desarrollar los juegos. Usando el XNA Framework los juegos funcionan en Windows y el Xbox 360, de esta manera es fácil desarrollar juegos de Windows y una fiel versión de Xbox 360 de ella. Aun así existen funcionalidades distintas en una u otra plataforma pero el objetivo primordial fue unificarlas en la medida de lo posible.

Simplificar el desarrollo del juego.

El desarrollo de juegos es una tarea compleja hasta para el más experimentado programador, ¿cómo será para un simple aficionado?

La meta del XNA es crear juegos en "5 minutos", eliminado elementos de programación tales como:

a) Crear una ventana.

b) Enumerar los aparatos gráficos y sus características,

c) Crear dispositivos de representación 3D (Direct3D)

d) Manejar punteros y direcciones a objetos e interfaces

Estas tareas representan elevada complejidad. En XNA lo importante es enfocarse en crear un bueno código para un buen juego.

Capas de XNA.

El siguiente gráfico, Figura 1.8, resume las capas que posee XNA:

Figura 1.8. Capas de XNA.

1) Capa Plataforma.

La plataforma es la capa más baja y agrupa API nativo, enmascarado por XNA. A esta capa pertenecen las APIs:

Direct3D 9

XACT

XInput

Content

2) Capa Núcleo del Framework

El núcleo es la primera capa del XNA y proporciona la funcionalidad sobre las cuales las otras capas trabajan. Si se tuviera que asociar algún trabajo de DirectX, esta sería la capa.

Se encuentran aquí las demás partes agrupadas por funcionalidad:

a) ***Gráficos***. El sistema grafico actual del XNA framework (XNA GSE 2.0) se encuentra sobre DirectX 9. EL grupo desarrollador empezó el desarrollo desde MDX pero realizaron una reconstrucción total para hacerlas más fáciles utilizar y más constante con las pautas del diseño de .NET.

XNA framework carece de ayuda para la funciones fijas o funciones integradas a las tarjetas graficas T&L (como sombreado Pong y Goraund, Bump-maping, Luces tipo T&L Spot... point , etc). Esto a favor de los esquemas de con sombreadores personalizados (en otras palabras se requiere una buena tarjeta de video). Las razones para esta decisión es que el futuro de los gráficos de computadora en tiempo real como el "Direct3D 10" ya no ofrecerá soporte a las funciones fijas.

Se consideró más valioso lograr programar en PC y XBOX 360 de forma idéntica que la entender la forma que se realizan sus funciones internamente. También tiene una implicación sobre los programadores veteranos deben aprender a trabajar solo con sombreadores para evitar errores en la compilación.

Se ha integrado un objeto llamado BasicEffect (que es un objeto que maneja los efectos) más fácil utilizar que los dispuestos para DirectX bajo C++, y con características tales como luces y texturas, de modo que la experiencia de los "primeros 5 minutos" siga estando adentro alcance.

Con el BasicEffect se pueden conseguir efectos sorprendentes en pantalla muy rápidamente y fácilmente sin tener que realmente escribir un sombreador. Se brindando la facilidad de sombreadores y efectos, en el componente BasicEffect lo que permite comenzar a escribir tus propios sombreadores y efectos, utilizar estos directamente.

b) ***Audio*.** El audio en XNA se construyo sobre de XACT, o plataforma cruzada de audio creada para Windows y el Xbox 360.

La idea detrás de XACT es imitar los fundamentos de los sombreadores del Direct3D. Los autores pueden utilizar las herramientas de XACT para crear los "paquetes" de efectos sonoros y configurar cosas como el volumen, mezclarse (5.1 incluyendo), el etc. El desarrollador entonces toma el paquete, lo carga, y puede reproducir fácilmente sonidos por su nombre, no teniendo que preocuparse de los almacenamientos (buffers) intermediarios o de inicialización.

Un autor puede crear un sonido y llamarlo "Gran Explosión" de diversos archivos de WAV, que contenga tenga un efecto de LFE (de baja frecuencia), y se mezcle entre los otros canales. El programador no necesita saber ninguno de estos detalles; él apenas con llamar a la "Gran Explosión" este se reproduce.

c) ***Entradas***. En XNA no se necesita listar dispositivos, iniciarlos y leerlos por que todos los dispositivos están siempre listos para ser leídos, todo lo que se necesita es llamar al GetState del dispositivo para leer su estado. Se habilitaron entradas desde el teclado y el ratón para Windows.

En cuanto a mandos, el sistema de interacción se construyó sobre la librería XInput, que es la plataforma cursada que conduce al mando de Xbox 360, común para ambas plataformas.

d) ***Matemáticas***. Las matemáticas del XNA proporcionan los tipos de uso frecuente para el juego 2D y 3D.

XNA incluye un paquete de matemáticas vectoriales como por ejemplo Vector2, Vector3, Vector4, matriz, el plano, y el rayo. También están incluidos: limitación por volumen como BoundingBox, BoundingSphere y BoundingFrustum (ayudas para cálculos de colisiones en 3D desde la camara); estos incluyen los métodos para hacer pruebas de la intersección y de la contención.

e) ***Almacenamiento.*** El API Almacenamiento proporciona la manera que puedes leer y escribir datos del juego (tales como juegos de ahorro, cuentas altas, etc.) de una manera neutral. En Windows no es un reparo puesto que se poseen las librerías System.IO y los métodos del sistema que permiten la localización correcta para el usuario actual. En el Xbox 360, es necesario el estado del juego asociado a un perfil y a un dispositivo de almacenaje, tal como el disco duro o una unidad de memoria.

3) Capa Framework de extensión.

El foco principal de esta capa es hacer el desarrollo del juego más fácil. En la actualidad actualmente, esta capa tiene dos componentes principales:

1) ***El Modelo de Aplicación***. El propósito del modelo del uso alejar al programador de los problemas de la plataforma, para solo centrarse en escribir el juego. No hay que preocuparse al crear ventanas, manejar sus mensajes, crear contador de tiempo o un reloj; XNA proporciona todo esto al programador.

Se proporciona un componente primario llamado GraphicsDeviceManager (servicio para administración de gráficos) que se encarga de la creación y gestión de los dispositivos gráficos, de esta manera no es necesario escribir código para crear un dispositivo grafico DirectX. Igualmente en el Xbox 360 no son necesarias las ventanas así el desarrollo ahora unifico para ambas plataformas.

También se ha proporcionado un modelo básico de componente, que permite que incorporar y crear fácilmente GameComponents (Componentes de Juego) escritos por otras personas en tu juego. Lo que representa una gran para la creación de una biblioteca de componentes reutilizables.

2) ***Content Pipeline***. Es la propuesta de Microsoft en XNA para facilitar la administración y construcción de los contenidos en los juegos.

Todos los desarrolladores comerciales generan formatos específicos para manejar los contenidos de sus juegos y en este proceso no reducen la complejidad del desarrollo si no que la incrementan.

Veamos las razones:

a) Es necesario crear herramientas para procesar los contenidos (texturas, niveles, modelos 3D, música etc...) y utilizar estas herramientas para la creación de los datos, esto se da en un juego o grupo de juegos.

b) Además se deben crear elementos programación para cargar y mostrar los contenidos.

El Content Pipeline se ha desarrollado para cumplir con los siguientes objetivos:

Ser simple de utilizar

Ser Extensible.

Ser Adaptable.

El Content Pipeline está formado por las siguientes partes:

Importador

Cuando se agrega un contenido es necesario el importador, que es el responsable de tomar datos y de normalizarlos. El importador toma los archivos y las importa en Visual Express C#.

Modelo de objetos de documento (DOM)

Después de que el importador haya hecho su trabajo, los datos existen en formato DOM. Este término DOM se utiliza simplemente para representar una colección de clases o esquemas, y como los Modelo de objetos de documento XML contienen tipos de datos y estructuras especificas. Esto significa que los datos están representados en un formato conocido, por ejemplo una serie de vértices o de datos de textura (sin importar su procedencia). Para eliminar errores un archivo puede estar escrito al estilo XML lo que implica que el importador almacena solamente datos.

Procesador

El procesador es responsable de tomar datos del contenido tipo DOM y de crear un objeto en tiempo de ejecución. Este objeto puede ser tan simple como un modelo o tan complejo como los procesadores múltiples en un juego.

El Content Piepeline de XNA incluye algunos procesadores que para Modelos con texturas, Sprites y efectos (Materiales .FX).

Serializador

Se encarga de convertir o compilar los archivos en forma .xnb estándar de GSE. Lo que sucede en realidad en este paso es que el archivo es el archivo se escribe nuevamente (no necesariamente de forma compacta) y es trabajo del Cargador el recuperar la información escrita en el.

Cargador

El cargador es en realidad un método que debe ser colocado en el juego o el proyecto, se encarga de convertir el archivo .xnb en un objeto o clase (DOM) para que pueda ser utilizado en tiempo de ejecución.

4) Capa Juegos

De esta capa, XNA provee los kits de desarrollo. El código del juego en sí y el contenido es tarea del programador.

El Game Studio Express es realmente la combinación de:

- NET Compact Framework 2.0

- Microsoft Visual estudio C# Express y documentación MSDN Express

- Microsoft Visual estudio C# Express Service Pack 1

- Librerías y documentación de XNA Framework 2.0

1.2. Conceptos previos

En este apartado se dará una descripción detallada de los siguientes apartados:

a) Terminología.

b) Programas software (CASE, IDE, algoritmos de propósito general o específico) relacionados con el proyecto.

1.2.1 Terminología

En los siguientes aparatados se describen los principales términos necesarios para el entendimiento del proyecto, y que están relacionados con la psicología.

1.2.1.1. Memoria Espacial

La memoria espacial es crítica para el conocimiento y exploración de los distintos recursos a nuestro alrededor y está íntimamente relacionada con la orientación espacial. Nos permite recordar la posición de determinados objetos o estímulos que podrán ser usados como orientación a la hora de ubicarse en un contexto dado.

Todo esto puede entenderse mejor con el siguiente ejemplo: Cierra los ojos y responde a esta cuestión: ¿Dónde estás? Una respuesta podría ser: Sentado frente a un escritorio. Pero del mismo modo, una respuesta válida podría ser también: "A 5 metros de la puerta" o "En el sur de España". Naturalmente estas respuestas vienen desde la memoria, no de la percepción directa de la puerta o de España.

1.2.1.2. Memoria de Trabajo

La memoria a corto plazo, memoria mediata, memoria de trabajo o memoria funcional es la que guarda y procesa durante breve tiempo la información que viene de los registros sensoriales y actúa sobre ellos y sobre otros [Etchepareborda *et al.*, 2005].

Un estímulo, al ser atendido y percibido, se transfiere a la memoria de trabajo. Esta memoria nos capacita para recordar la información pero, es limitada y susceptible de interferencias. Esta vulnerabilidad del proceso le imprime un carácter de enorme flexibilidad, que nos permite estar siempre abiertos a la recepción de nueva información.

1.2.1.3. Estímulo

En psicología, se puede definir un estímulo como cualquier cosa que influya efectivamente sobre los aparatos sensitivos de un organismo viviente, incluyendo fenómenos físicos internos y externos del cuerpo.

En este proyecto, aparecerán distintos estímulos que influirán sobre la vista y oído de los sujetos a examen. Tendremos estímulos visuales colocados en la pared que ayudarán a orientarse al sujeto dentro de la habitación, sirviendo como punto de referencia para encontrar las zonas premiadas.

Al encontrar una zona premiada, el sujeto percibirá un estimulo visual en forma de mensaje comunicándole que ha encontrado una zona premiada y cuantas más le quedan por visitar, además de un estímulo sonoro. Cuando visite una zona, ésta aparecerá resaltada en verde en el suelo, siendo un estímulo visual más para poder orientarse y recordar dónde se encuentran las zonas premiadas para posteriores visitas.

1.2.2. Descripción de los programas software relacionados con el proyecto

1.2.2.1. Microsoft Visual Studio 6.0

Microsoft Visual Studio [VisualStudio] es un entorno de desarrollo integrado para sistemas Windows. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros.

Visual Studio permite a los desarrolladores crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET (a partir de la versión net 2002). Así se pueden crear aplicaciones que se intercomuniquen entre estaciones de trabajo, páginas web y dispositivos móviles.

1.2.3.1 COSMOS

Este programa es una herramienta de estimación y análisis de proyectos software, que proporciona a los desarrolladores una idea del tamaño, esfuerzo y planificación de su proyecto de software. Cosmos combina los puntos de función bien conocidos y los modelos COCOMO, así como un modelo *Rayleigh* de acumulación de personal propuesto por Lawrence Putnam. Estos tres modelos pueden ser usados independientemente o trabajar juntos. Con COSMOS [COSMOS], los usuarios pueden beneficiarse de un entendimiento de los cambios en los requerimientos del proyecto y en los recursos que producen impacto en el tamaño, esfuerzo y planificación del proyecto.

1.2.3.2. ArgoUML

ArgoUML es una aplicación para diseñar diagramas en UML (Unified Modeling Language) escrita en Java y publicada bajo la Licencia BSD Open Source. Dado que es una aplicación Java, está disponible en cualquier plataforma soportada por Java. Sin embargo, no es conforme completamente a los estándares UML y carece de soporte completo para algunos tipos de diagramas incluyendo los Diagrama de secuencia y los de colaboración.

1.2.3.3 Microsoft Project

Microsoft Project, programa de la suite Microsoft Office, es un software de administración de proyectos diseñado, desarrollado y comercializado por Microsoft para asistir a administradores de proyectos en el desarrollo de planes, asignación de recursos a tareas, dar seguimiento al progreso, administrar presupuesto y analizar cargas de trabajo.

El software Microsoft Office Project [MSProject] en todas sus versiones (la versión 2007 es la más reciente) es útil para la gestión de proyectos, aplicando procedimientos descritos en el Management Body of Knowledge del Project Management Institute.

1.2.3.4 WBS Chart Pro

Programa de administración de proyectos que permite mostrar y crear diagramas WBS que muestran la descomposición jerárquica de las tareas a realizar [WBSChart].

1.2.3.5 Eclipse

Eclipse [Eclipse] es un entorno de desarrollo integrado de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados (del inglés IDE), como el IDE de Java llamado *Java Development Toolkit* (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse (y que son usados también para desarrollar el mismo Eclipse).

1.3. Planificación

Un paso previo a la realización de un proyecto software de una envergadura considerable, es la realización de una serie de estudios para así estimar algunas métricas importantes como:

a) Líneas de código esperadas.

b) Esfuerzo en personas-mes.

c) Duración del proyecto.

d) Planificación temporal del proyecto.

Todas estas métricas no son del todo objetivas, pues se cuantifican de una manera aproximada. Sin embargo, son útiles para hacernos una idea de:

a) Número de personas necesarias para el proyecto.

b) Duración aproximada del proyecto.

c) Fechas límite e hitos en las entregas intermedias de las que se dispone.

Aplicando todo lo anterior, se pueden realizar valoraciones del estado actual del trabajo, comprobando si se cumplen los plazos fijados y los objetivos establecidos.

En la planificación del proyecto, se realizarán los siguientes pasos:

1º. Configurar el método de trabajo.

2º. Detectar los elementos y fuentes de información básicos que hacen falta para el desarrollo del proyecto.

3º. Delimitar el ámbito del software.

4º. Descomposición funcional detallada y jerárquica del proyecto.

5º. Obtener las primeras estimaciones: líneas de código y puntos de función.

6º. Estimación de costes en cuanto a duración y esfuerzo, aplicando para ello los modelos COMOMO básico e intermedio.

7º. Realizar una planificación temporal del proyecto.

1.3.1. Configurar el método de trabajo

Para el desarrollo del proyecto, se usará el modelo en espiral propuesto por Boehm, que es un modelo de proceso de software evolutivo que conjuga la naturaleza iterativa de la construcción de prototipos con los aspectos controlados y sistemáticos del modelo lineal secuencial, [Pressman *et al.*, 1997].

Al comienzo se evalúan las posibles alternativas de desarrollo, optando por la que menos riesgos acarree y se hace un ciclo de la espiral. Si se necesita añadir funcionalidad al software, se vuelven a evaluar las posibles alternativas de desarrollo y se realiza otra vuelta de la espiral.

La elección de este modelo se debe a que se parte de una versión anterior, de forma que puedo reutilizar parte del código y generar nuevos prototipos del programa mejorados. De esta forma, se seguirá pudiendo añadir nuevas funcionalidades en un futuro. Además, con este modelo se pueden obtener resultados rápidamente a través de los prototipos generados.

1.3.2. Detectar los elementos y fuentes de información básicos que hacen falta para el desarrollo del proyecto

En este apartado se ha realizado un análisis de la información necesaria para el desarrollo del proyecto:

a) Aplicaciones para la creación de gráficos en 3D.

b) APIs y motor gráfico creado para crear/simular mundos virtuales en 3D.

c) Comunicación entre el programa de modelado y el motor gráfico: formato de exportación.

d) Matemática del motor gráfico: técnicas de colisión, sentido de los ejes.

e) Aplicación de texturas.

f) Funcionamiento del programa: localización de zonas premiadas, ensayos.

g) Presentación de resultados: página web, trayectorias.

Las fuentes de información necesarias en el desarrollo del proyecto provendrán de diferentes fuentes:

1) Personas:

a) Directores de proyecto.

b) Ingenieros informáticos, ingenieros técnicos informáticos, Licenciados en Psicología.

2) Otros medios:

a) Bibliografía: programación, diseño en 3d, construcción de motores gráficos, estudios sobre memoria espacial, estudios sobre informática aplicada a psicología.

b) Enlaces: documentos digitales, foros, chats, etc.

1.3.3. Delimitar el ámbito del software

Antes de la planificación de un proyecto, se han de establecer el ámbito y los objetivos del proyecto, considerar las posibles alternativas e identificar las restricciones técnicas y de gestión.

Sin esto no se podrán obtener estimaciones razonables y precisas en cuanto al coste del proyecto. Tampoco una identificación realista de las tareas a realizar en el proyecto ni un plan de trabajo adecuado.

El ámbito identificará las funciones que llevará a cabo el software intentando delimitar esas funciones de forma cuantitativa. El ámbito describe la función, el rendimiento, las restricciones, las interfaces y la fiabilidad.

Descripción funcional

Se pretende crear una herramienta software capaz de realizar simulaciones en entornos virtuales 3D, que posteriormente será usada para evaluar una capacidad muy importante del cerebro humano: la memoria espacial y la memoria de trabajo.

Este entorno virtual estará formado por una habitación circular, de unos 8 metros de diámetro, en cuyo interior hay distribuidas 9 zonas. El tamaño de esas zonas puede variar entre pequeño, mediano o grande. La habitación junto con las zonas será diseñada en 3D Studio Max, diseñando además una habitación igual a la otra pero sin techo, necesaria para mostrar las trayectorias de los sujetos. Las paredes de la habitación estarán divididas en 12 partes iguales, de forma similar a las franjas horarias. En cada una de esas divisiones, podrán aparecer estímulos que ayuden a los sujetos a orientarse. Los estímulos también serán creados usando 3DS Max. Tanto los estímulos creados como los modelos de la habitación serán exportados en ".3DS". La configuración de la habitación podrá personalizarse mediante una serie de interfaces gráficas y almacenarse en archivos XML.

La aplicación constará de 2 tipos distintos de experimentos: los destinados a estudiar la memoria espacial y los destinados a estudiar la memoria de trabajo.

a) Experimentos de memoria espacial:

En los experimentos de memoria espacial, los estímulos y zonas premiadas son siempre los mismos a lo largo de los ensayos de que conste el experimento, variando solo la posición de salida del sujeto de forma aleatoria.

Para configurar los experimentos de memoria espacial, es necesario indicar los estímulos que aparecerán en cada una de las 12 divisiones de la habitación, y la localización de las zonas premiadas.

A la hora de lanzar un nuevo experimento, habrá que indicar el nombre del sujeto del experimento, el número de ensayos del experimento, la duración de los mismos, el tamaño de las zonas, la velocidad de desplazamiento, el modo de video y el archivo de configuración a usar.

El motor gráfico será capaz de leer todos estos datos y configurar los ensayos de forma adecuada. Permitirá también moverse por la habitación, permitiendo desplazamientos hacia adelante o rotaciones. Como consecuencia de ese movimiento, el sujeto encontrará zonas premiadas, recibiendo un mensaje indicando que ha encontrado una zona y cuantas le faltan por encontrar. Además, la zona encontrada aparecerá resaltada en el suelo con color verde. Los ensayos acaban cuando se encuentren todas las zonas premiadas o se acabe el tiempo establecido. Al finalizar un ensayo, se mostrará también un mensaje informativo.

Se añadirán además 2 ensayos adicionales al final del experimento. En el penúltimo ensayo no habrá *feedback*, o sea, que no se mostrará ningún mensaje al encontrar una zona ni se reproducirá ningún sonido ni se marcará la zona premiada en el suelo. Este ensayo se realiza para comprobar que el sujeto sabe realmente donde están las zonas premiadas. En el último ensayo, las zonas aparecerán marcadas en el suelo y si aparecerán los mensajes al encontrar las zonas. Este ensayo se realiza para comprobar que el sujeto sabe controlar el joystick.

Por último, se creará una página web con los resultados obtenidos mostrados de forma ordenada. Desde esta web, se permitirá la visualización de la trayectoria que han seguido los sujetos a lo largo de los distintos ensayos. Para ello, se muestra la vista en planta de la habitación y se muestran las trayectorias como líneas sobre el suelo de la habitación.

b) Experimentos de memoria de trabajo:

En los experimentos de memoria de trabajo, los estímulos serán siempre los mismos, pero para cada bloque habrá que configurar las posiciones de salida de los ensayos primero y segundo, la duración de los mismos, el tiempo entre un ensayo y el otro y las zonas premiadas para ese bloque.

Para configurar los experimentos de memoria de trabajo, será necesario indicar los estímulos que aparecerán en cada una de las 12 divisiones de la habitación y el número de bloques. Para cada uno de los bloques, habrá que indicar las posiciones de salida de los 2 ensayos, la duración de cada uno, el tiempo entre ensayos y las zonas premiadas del bloque.

A la hora de lanzar un nuevo experimento, se indicará el nombre del sujeto, el tiempo que trascurrirá entre bloques, el tamaño de las zonas, la velocidad de desplazamiento, el modo de video y el archivo de configuración a usar.

El motor gráfico será capaz de leer todos estos datos y configurar los ensayos de forma adecuada. Permitirá también moverse por la habitación, permitiendo desplazamientos hacia adelante o rotaciones. Como consecuencia de ese movimiento, el sujeto encontrará zonas premiadas, recibiendo un mensaje indicando que ha encontrado una zona y cuantas le faltan por encontrar. Además, la zona encontrada aparecerá resaltada en el suelo con color verde. Al comienzo y al final de cada bloque, aparecerá un mensaje informando del número de bloque y si se pasa a otro bloque. Los bloques acaban cuando acaban los 2 ensayos de que consta un bloque. Los ensayos acaban cuando se encuentren todas las zonas premiadas o se acabe el tiempo establecido. Al finalizar un ensayo, se mostrará también un mensaje informativo.

Por último, se creará una página web con los resultados obtenidos mostrados de forma ordenada. Desde esta web, se permitirá la visualización de la trayectoria que han seguido los sujetos a lo largo de los distintos ensayos. Para ello, se muestra la vista en planta de la habitación y se muestran las trayectorias como líneas sobre el suelo de la habitación.

Rendimiento

A la hora de diseñar los modelos con 3D Studio Max, se ha optado por usar el mínimo número posible de polígonos, intentando siempre que los modelos resulten realistas y atractivos. Al usar pocos polígonos, se consigue reducir el consumo de recursos, evitando así excesivos tiempos de carga y ralentizaciones en la ejecución del programa. A su vez, se ha intentado usar texturas de poco tamaño, menos de 1 Megabyte, para evitar también excesivos tiempos de carga. Además, para reducir los cálculos de colisiones, se han envuelto los objetos con una malla más simple que será con la que se colisionará realmente.

Interfaces

El programa consta de 3 interfaces. Una de ellas se usará para configurar los experimentos (crear nuevos experimentos para memoria espacial o de trabajo, crear configuraciones nuevas para los experimentos, ver experimentos anteriores), otra será la ventana donde se muestra el entorno 3D (la habitación con sus estímulos, etc.) y otra será la ventana donde se muestran las trayectorias recorridas por los sujetos.

Fiabilidad

Se espera una fiabilidad media ya que en caso de que ocurriera algún error, no conllevaría mayores problemas que el repetir el ensayo. Aún así se ha intentado que no existan errores algunos.

1.3.4. Descomposición funcional detallada y jerárquica del proyecto

Antes de empezar a diseñar y codificar una aplicación se ha de realizar una descomposición de toda la funcionalidad del mismo, para ir dividiendo poco a poco las distintas tareas que se deberán realizar. En la descomposición funcional, se ha separado por una parte todo lo relacionado a la creación de los modelos 3D, tanto de la habitación como de los estímulos que aparecen en ella. Por otro lado, se ha reunido lo referente a las interfaces, tanto la interfaz de ventanas como la interfaz gráfica del experimento en si como la de visualizar la trayectoria de los experimentos.

1. Estudio del modelado y exportación 3DS Max.
    a. Creación de la habitación circular.
        i. Comprobación del sistema de unidades.
        ii. Dibujar el diseño en 3D.
            i. Dibujar la habitación con forma circular.
            ii. Dibujar las zonas premiadas.
            iii. Colocación de las zonas en la habitación.
            iv. Nombrar correctamente los objetos.
        iii. Situar el modelo en el origen de coordenadas.
        iv. Aplicar texturas a los componentes del modelo.
        v. Extraer vista en planta del modelo.
        vi. Exportar ambos en formato .3ds.
    b. Creación de los estímulos.
        i. Comprobación del sistema de unidades.
        ii. Dibujar el diseño en 3D.
        iii Situar el modelo en base a una serie de restricciones.
        iv. Aplicar texturas a los componentes del modelo.
        vi. Exportar el modelo en formato .3ds.
2. Interfaces
    a. Configurar un experimento para memoria espacial.
        i. Seleccionar los estímulos y su posición.
        ii. Seleccionar las zonas premiadas.
        iii. Restablecer los estímulos y zonas.

iv. Guardar la configuración.

b. Configurar un experimento para memoria de trabajo.

i. Seleccionar los estímulos y su posición.

ii. Establecer el número de bloques.

iii. Seleccionar las zonas premiadas para cada bloque.

iv. Restablecer los estímulos y zonas.

v. Guardar la configuración.

c. Ver resultados de los experimentos.

i. Ver resultados de experimentos de memoria espacial.

1. Mostrar web con los datos del experimento.
2. Iniciar la vista de la trayectoria.

ii. Ver resultados de experimentos de memoria de trabajo.

1. Mostrar web con los datos del experimento.
2. Iniciar la vista de la trayectoria.

d. Crear nuevo experimento.

i. Nuevo experimento de memoria espacial.

1. Establecer el nombre del sujeto.
2. Establecer el número de ensayos.
3. Establecer el tiempo para cada ensayo.
4. Elegir la velocidad de desplazamiento.
5. Elegir el tamaño de las zonas.
6. Seleccionar el archivo de configuración.
7. Seleccionar modo de video.
8. Lanzar la aplicación.

ii. Nuevo experimento de memoria de trabajo.

1. Establecer el nombre del sujeto.
2. Establecer el tiempo entre bloques de ensayos.
3. Elegir la velocidad de desplazamiento.
4. Elegir el tamaño de las zonas.

5. Seleccionar el archivo de configuración.
6. Seleccionar modo de video.
7. Lanzar la aplicación 3D.

e. Aplicación 3D Experimentos.

i. Dibujar el interfaz

1. Iniciar el entorno gráfico en OpenGL.
2. Cargar archivo de configuración.
3. Dibujar los modelos 3D
1. Cargar el modelo en formato 3ds.
2. Representar los objetos.
4. Utilización de las texturas:
1. Cargar archivos ".bmp".
2. Mapear las texturas sobre los polígonos.

ii. Movimiento de la cámara.

1. Desplazar la cámara hacia delante.
2. Rotar la cámara a la izquierda.
3. Rotar la cámara a la derecha.

iii. Detección de colisiones.

1. Detección de colisiones con objetos.
2. Detección de colisiones con las zonas premiadas.

iv. Guardar datos del estudio.

1. Creación de la página web con los resultados.
2. Guardar configuración del experimento.
2. Guardar resultados de las trayectorias.
3 Creación de los scripts para la visualización de las trayectorias.

v. Cerrar aplicación.

1. Limpiar la interfaz.
2. Liberar recursos.

f. Aplicación 3D Trayectorias.

i. Dibujar el interfaz

1. Iniciar el entorno gráfico en OpenGL.
2. Cargar archivo de configuración.
3. Dibujar los modelos 3D.
1. Cargar el modelo en formato 3ds.
2. Representar los objetos.
4. Utilización de las texturas:
1. Cargar archivos ".bmp".
2. Mapear las texturas sobre los polígonos.

ii. Cargar las trayectorias

iii. Representar las trayectorias.

1. Dibujar trayectoria.
2. Ver paso a paso

iv. Movimiento de la cámara.

1. Acercar la cámara.
2. Alejar la cámara.
3. Desplazar la cámara a la derecha.
4. Desplazar la cámara a la izquierda.
5. Desplazar la cámara hacia arriba.
6. Desplazar la cámara hacia abajo.

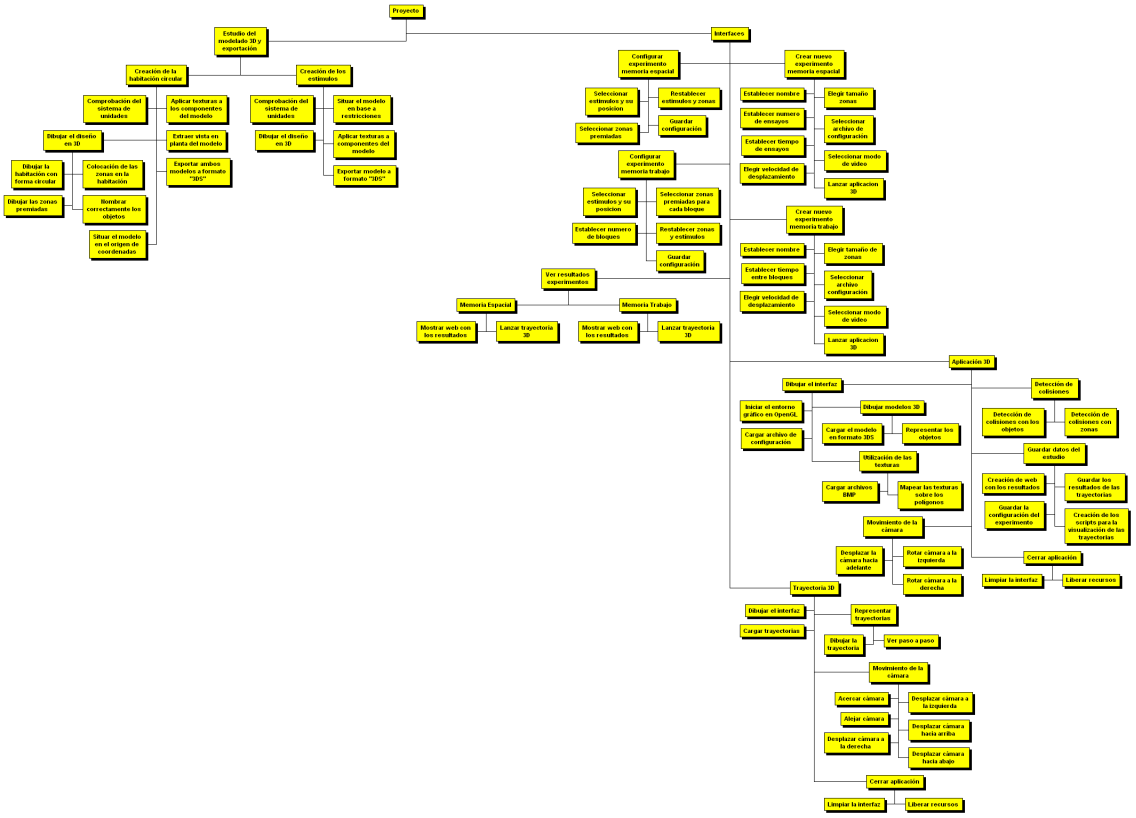
v. Cerrar aplicación.

1. Limpiar la interfaz.
2. Liberar recursos.

Con esta descomposición se comienza a diseñar el aspecto de la aplicación definitiva. Tomando la funcionalidad que ofrecerá la aplicación, se divide cada función hasta un límite oportuno, intentando dejar funciones lo más básicas posibles. Esta descomposición no es definitiva, pues seguramente alguna función habrá que descomponerla en otras más básicas.

El siguiente diagrama WBS muestra de forma gráfica y organizada jerárquicamente las funciones anteriormente expuestas, y permite tener una visión general de ellas y de cómo dependen unas de otras:

Aquí va el diagrama en A3



1.3.5. Puntos de función

Las métricas orientadas a la función son medidas indirectas del software y del proceso por el cual se desarrolla. Las métricas orientadas a la función se centran en la funcionalidad o utilidad del programa. Los puntos de función son obtenidos utilizando una relación empírica basada en medidas contables del dominio de la información del software y valoraciones subjetivas de la complejidad del software [Luque *et al*., 1999].

Se realizará una primera estimación, que dará un valor para el esfuerzo y duración del proyecto. Esta primera estimación puede ajustarse usando los valores de complejidad del software, tras los cuales se obtendrán nuevos valores para el esfuerzo y duración. Pero aún se puede ajustar más, usando los modelos COCOMO básico e intermedio.

La Figura 1.9, ilustra de forma gráfica el proceso de ajuste de los puntos de función:

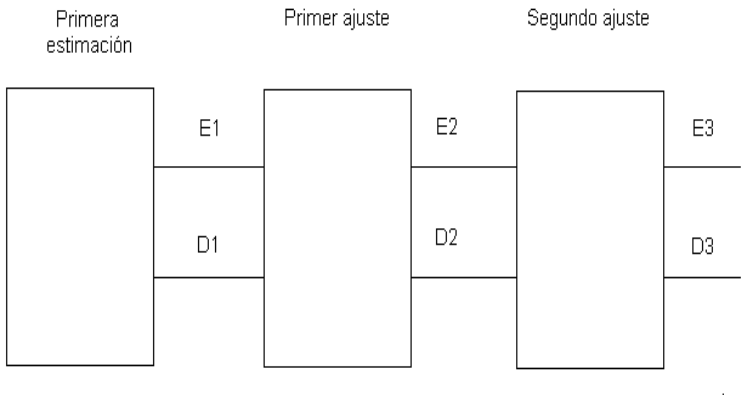


Figura 1.9. Proceso de ajuste de los puntos de función.

Para la realización de una estimación en puntos de función, se estiman cada una de las características del dominio de la información:

- *Número de entradas*: Se cuenta cada entrada de usuario que proporciona diferentes datos orientados a la aplicación.

- *Número de salidas*: Se cuenta cada salida que proporciona al usuario información orientada a la aplicación. En este contexto, la salida se refiere a informes, pantallas, mensajes de error. Los elementos de un reporte, no se cuentan de forma separada.

- *Numero de archivos*: Son los archivos que pueden ser parte de una base de datos o independientes.

- *Número de interfaces externas*: Son los archivos que se usan para transmitir información a otro sistema.

- *Número de peticiones*: Es una entrada interactiva que produce la generación de alguna respuesta del software en forma de salida interactiva.

Después se clasifica y pondera cada función por su nivel de complejidad: simple, media, compleja (Low, Average, High).

La imagen siguiente, Figura 1.10, muestra la estimación con el programa COSMOS:

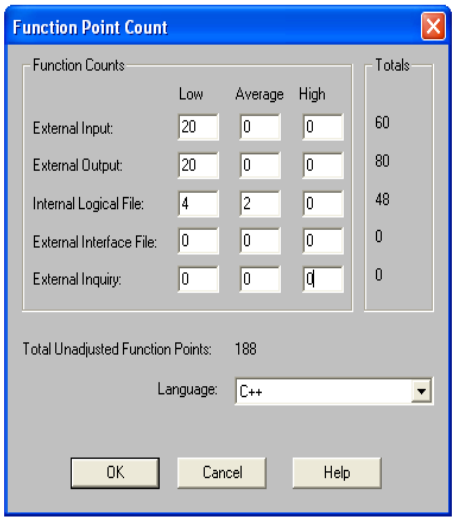


Figura 1.10. Estimación de los puntos de función usando COSMOS.

A la hora de realizar la estimación, se asignan valores a cada una de las tareas de la descomposición funcional y se realiza la suma de todos los valores. Puede observarse que predominan las entradas y salidas.

El número de puntos de función obtenidos es 188. Sin embargo todavía se pueden ajustar más parámetros, para poder realizar una estimación mucho más precisa. Estos parámetros son los 14 valores de complejidad del software y se puntúan según la Tabla 1.2:

#	**Concepto**	**Valor**
1	¿Requiere el sistema copias de seguridad y de recuperación fiables?	0
2	¿Requiere comunicación de datos?	0
3	¿Existen funciones de procesamiento distribuido?	0
4	¿Es crítico el rendimiento?	3
5	¿Se ejecutará el sistema en un entorno operativo existente y fuertemente utilizado?	3
6	¿Requiere entrada de datos interactiva?	2
7	¿Requiere la entrada de datos interactiva que las transacciones de entrada se lleven a cabo sobre múltiples pantallas u operaciones?	0
8	¿Se actualizan los archivos maestros de forma interactiva?	0
9	¿Son complejas las entradas, las salidas, los archivos o las peticiones?	3
10	¿Es complejo el procesamiento interno?	4
11	¿Se ha diseñado el código para ser reutilizable?	3

12	¿Están incluidas en el diseño la conversión y la instalación?	2
13	¿Se ha diseñado el sistema para soportar múltiples instalaciones en diferentes organizaciones?	0
14	¿Se ha diseñado la aplicación para facilitar los cambios y para ser fácilmente utilizada por el usuario?	4

Tabla 1.2. Valores de complejidad el software.

Cada uno de estos factores está evaluado de 0 a 5, siendo el significado:

0 → No está presente o no influye

1 → Influencia Mínima

2 → Influencia Moderada

3 → Influencia promedio

4 → Influencia significativa

5 → Influencia fuerte

El rendimiento es algo que se intentará sea el máximo posible, para evitar ralentizaciones en la ejecución del programa, por eso tiene una influencia moderada. Como el programa se ejecutará sobre plataformas Windows, que es un sistema operativo existente y fuertemente utilizado, también tendrá una influencia moderada a la hora de desarrollar el programa. El procesamiento a la hora de trabajar con vectores, colisiones, dibujar en pantalla los modelos, etc., será complejo, por ello, se le asigna una influencia significativa. El desarrollo se realizará teniendo en cuenta que las personas que lo usarán no tienen conocimientos informáticos avanzados, por lo que se intentará que la instalación y el funcionamiento del programa sean lo más sencillo e intuitivo posible. También se procuraré que haya facilidad a la hora de hacer cambios en el software y dar facilidades a la reusabilidad.

La salida que ofrece COSMOS tras el ajuste de los puntos de función es la siguiente:

Title:	**Proyecto Fin de Carrera**	
Prepared By:	**Armando Soria Albacete**	
Description:	**Desarrollo de nuevos entornos 3D para la evaluación de memoria espacial en humanos**	
Sensitivity Analysis Type:	**Adjusted Function Points**	
Unadj. Function Points:	**188.0**	
Value Adjustment Factor:	**0.89**	
Adj. Function Points:	**167.3**	**(167.3 - 167.3)**
Language:	**C++ (53 SLOC/FP)**	
Source Lines of Code:	**8868.0**	**(8868.0 - 8868.0)**

Figura 1.11. Captura de los datos arrojados por COSMOS tras el ajuste de los puntos de función.

Tras el ajuste de los puntos de función, se obtienen 167,3 que son unos pocos menos que los 188 iniciales. Usando C++ se estima que serán unas 8868 líneas de código con las que contará el programa.

1.3.5.1 Modelo COCOMO Básico

Este modelo trata de estimar, de una manera rápida y más o menos burda, la mayoría de proyectos pequeños y medianos. Se consideran tres modos de desarrollo en este modelo: orgánico, semiacoplado y empotrado.

Modo orgánico.

En este modo, un pequeño grupo de programadores experimentados desarrollan software en un entorno familiar. El tamaño del software varía de unos pocos miles de líneas (tamaño pequeño) a unas decenas de miles de líneas (medio), mientras que en los otros dos modos el tamaño varía de pequeño a muy grandes (varios cientos de miles de líneas). En este modo, al igual que en los otros, el coste se incrementa a medida que el tamaño lo hace, y el tiempo de desarrollo se alarga.

Se utilizan dos ecuaciones para determinar el esfuerzo de personal y el tiempo de desarrollo.

El Esfuerzo, que se expresa en personas-mes y KLDC es el tamaño expresado en miles de líneas de código.

$$E = 2{,}4 * KLDC^{1{,}05}$$

El tiempo de desarrollo, D, donde E se obtiene de la ecuación anterior y D es el tiempo de desarrollo en meses. Estas ecuaciones se obtuvieron por medio de ajustes de curvas realizado por Boehm en TRW sobre 63 proyectos.

$$D = 2{,}5 * E^{0{,}38}$$

Modo Empotrado.

En este modo, el proyecto tiene unas fuertes restricciones, que pueden estar relacionadas con el procesador y el interface hardware. El problema a resolver es único y es difícil basarse en la experiencia, puesto que puede no haberla.

Las estimaciones de tiempo y coste se basan en las mismas ecuaciones que en el modo orgánico, pero con diferentes constantes. Así, el Esfuerzo se da por:

$$E = 3{,}6 * KLDC^{1{,}20}$$

Y el tiempo de desarrollo por:

$$D = 2{,}5 * E^{0{,}32}$$

Modo Semiacoplado.

Es un modo intermedio entre los dos anteriores. Dependiendo del problema, el grupo puede incluir una mezcla de personas experimentadas y no experimentadas. Las ecuaciones son:

$$E = 3{,}0 * \ KLDC^{1{,}12}$$

$$D = 2{,}5 * E^{0{,}35}$$

Este proyecto entra en la categoría de proyectos orgánicos, es decir, para pocos programadores y unas decenas de miles de líneas de código.

Partiendo de las 8868 líneas de código estimadas mediante los puntos de función, se procede a calcular el esfuerzo y duración del proyecto:

$$E = 2{,}4 * \ KLDC^{1{,}05} = 2{,}4 * 8{,}9^{1{,}05} = 23{,}8 \ \frac{personas}{mes}$$

$$D = 2{,}5 * E^{0{,}38} = 2{,}5 * 23{,}8^{0{,}38} = 8{,}3 \ meses$$

El número de personas necesarias para completar el desarrollo en este tiempo será:

$$N = \frac{E}{D} = \frac{23{,}8}{8{,}3} = 2{,}8 \ personas$$

1.3.5.2. Modelo COCOMO intermedio

Este añade al modelo básico quince modificadores opcionales para tener en cuenta en el entorno de trabajo, incrementando así la precisión de la estimación. Para este ajuste, al resultado de la fórmula general se lo multiplica por el coeficiente surgido de aplicar los atributos que se decidan utilizar.

Cada atributo se cuantifica para un entorno de proyecto. Dependiendo de la calificación de cada atributo, se asigna un valor para usar de multiplicador en la fórmula.

El valor de cada atributo, de acuerdo a su calificación, se muestra en la siguiente tabla:

	Valor					
Atributo	Muy Bajo	Bajo	Nominal	Alto	Muy Alto	Extra Alto
Atributos de software						
Fiabilidad	0,75	0,88	1,00	1,15	1,40	-
Tamaño de la Base de datos	-	0,94	1,00	1,08	1,16	-
Complejidad	0,70	0,85	1,00	1,15	1,30	1,65
Atributos de hardware						
Restricciones de tiempo de ejecución	-	-	1,00	1,11	1,30	1,66
Restricciones de memoria virtual	-	-	1,00	1,06	1,21	1,56
Volatilidad de la maquina virtual	-	0,87	1,00	1,15	1,30	-
Tiempo de respuesta	-	0,87	1,00	1,07	1,15	-
Atributos del personal						

Capacidad de análisis	1,46	1,19	1,00	0,86	0,71	-
Experiencia en la aplicación	1,29	1,13	1,00	0,91	0,82	-
Calidad de los programadores	1,42	1,17	1,00	0,86	0,70	-
Experiencia en la máquina virtual	1,21	1,10	1,00	0,90	-	-
Experiencia en el lenguaje	1,14	1,07	1,00	0,95	-	-
Atributos del proyecto						
Técnicas actualizadas de programación	1,24	1,10	1,00	0,91	0,82	-
Utilización de herramientas de software	1,24	1,10	1,00	0,91	0,83	-
Restricciones de tiempo de desarrollo	1,23	1,08	1,00	1,04	1,10	-

Tabla 1.3. Tabla de modificadores del entorno de trabajo.

El significado de los atributos es el siguiente, según su tipo:

Atributos de software:

Fiabilidad: garantía de funcionamiento requerida al software. Indica las posibles consecuencias para el usuario en el caso que existan defectos en el producto. Para este atributo selecciono un valor bajo, pues si ocurre algún problema con la ejecución, ninguna vida correrá peligro y sólo obligará a ejecutarlo de nuevo.

Tamaño de la Base de datos: tamaño de la base de datos en relación con el tamaño del programa. Selecciono un valor bajo, pues no se usa una base de datos en la aplicación.

Complejidad: representa la complejidad del producto. Para este atributo, selecciono muy alto, pues la construcción de modelos, animación, cálculo de colisiones, etc., es un proceso bastante complejo.

Atributos de hardware:

Restricciones de tiempo de ejecución: limitaciones en el porcentaje del uso de la CPU. Elijo un valor nominal, pues no hay limitaciones en el porcentaje de uso de la CPU.

Restricciones de memoria virtual: limitaciones en el porcentaje del uso de la memoria. Selecciono un valor nominal pues no hay limitaciones.

Volatilidad de la maquina virtual: Selecciono un valor bajo.

Tiempo de respuesta: Elijo un valor muy alto, pues el rendimiento es prioritario en la aplicación.

Atributos de personal:

Capacidad de análisis: La considero muy alta ya que me considero preparado con los conocimientos adquiridos durante la carrera.

Experiencia en la aplicación: experiencia del personal en aplicaciones similares. Selecciono un valor nominal, puesto que utilizo tecnologías que conozco y otras que no.

Calidad de los programadores: También la considero muy alta por la misma razón que la primera.

Experiencia en la máquina virtual: Dejo un valor nominal.

Experiencia en el lenguaje: experiencia en el lenguaje de programación a usar. Valor alto, pues C++ es junto a Java el lenguaje más usado durante la carrera.

Atributos de proyecto:

Técnicas actualizadas de programación: uso de prácticas modernas de programación. Elijo muy alto, pues considero que utilizaré metodologías correctas en la programación.

Utilización de herramientas de software: uso de herramientas de desarrollo de software. Muy alto, pues el desarrollo del proyecto requiere el uso de múltiples herramientas.

Restricciones de tiempo de desarrollo: limitaciones en el cumplimiento de la planificación. Muy alto, pues es necesario entregar el producto en una fecha determinada.

Estos datos se recogen en la Tabla 1.4 a modo de resumen:

Atributo	Valor escogido
Fiabilidad	Bajo
Tamaño de la Base de datos	Bajo
Complejidad	Muy Alto
Restricciones de tiempo de ejecución	Nominal
Restricciones de memoria virtual	Nominal
Volatilidad de la maquina virtual	Bajo
Tiempo de respuesta	Muy Alto
Capacidad de análisis	Muy Alto
Experiencia en la aplicación	Nominal
Calidad de los programadores	Muy Alto
Experiencia en la máquina virtual	Nominal

Experiencia en el lenguaje	Alto
Técnicas actualizadas de programación	Muy Alto
Utilización de herramientas de software	Muy Alto
Restricciones de tiempo de desarrollo	Muy Alto

Tabla 1.4. Valores de ajuste del entorno.

Sabiendo que el programa se trata de un proyecto orgánico y que las líneas de código esperadas son 8868, que ya fueron calculadas con el modelo COCOMO básico, configuramos el modelo intermedio, como se muestra en la Figura 1.12:

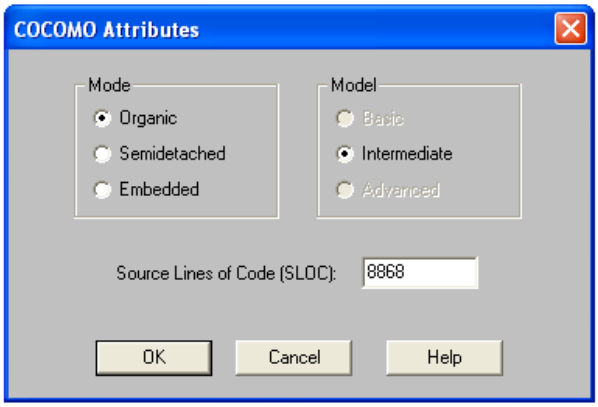


Figura 1.12. Configuración de los atributos de COCOMO

Los resultados obtenidos tras establecer los anteriores atributos son:

Title:	**Proyecto Fin de Carrera**
Prepared By:	**Armando Soria Albacete**
Description:	**Desarrollo de Nuevos Entornos 3D para la Evaluación de Memoria Espacial en Humanos**
Source Lines of Code:	**8868.0**
Nominal Effort:	**31.6 person months**
Adjusted Effort:	**13.6 person months**
Time to Develop:	**6.7 calendar months**

Figura 1.13. Captura de los datos arrojados por COSMOS.

Usando el modelo COCOMO intermedio, obtenemos que el esfuerzo nominal será de 31,6 personas/mes, pero tras ajustar los atributos del proyecto, obtenemos un esfuerzo de 13,6 personas/mes. El tiempo necesario para desarrollar el proyecto se estima en 6,7 meses.

Con estos datos se puede calcular el número de personas necesario para completar el desarrollo en ese tiempo:

$$N = \frac{E}{D} = \frac{13{,}6}{6{,}7} = 2{,}02\ personas$$

De casi 3 personas y 8 meses, se baja a 2 personas y casi 7 meses. Aún así, el hecho de salir 2 personas implica que tendré que esforzarme mucho.

1.3.6. Realizar una planificación temporal del proyecto

Como última tarea, se lleva a cabo una planificación para establecer el tiempo de finalización y los plazos que se han de ir cumpliendo para la consecución de los objetivos en el tiempo establecido. Para ello, se usarán diagramas de Gantt y Pert, que son populares herramientas gráficas que muestran en un gráfico el tiempo de dedicación previsto para las diferentes tareas a lo largo del tiempo [Sommerville,2005].

Usando los datos anteriormente calculados, se planifica una serie de tareas a realizar, la duración de las mismas y las posibles interdependencias entre ellas.

Las tareas planificadas son:

a) **Entrevista con los directores de proyecto y el cliente:** antes de comenzar, es necesario realizar una serie de entrevistas con los directores del proyecto, para identificar objetivos e intercambiar opiniones.

b) **Búsqueda bibliográfica**: es necesario familiarizarse con las herramientas que se usarán, así como buscar toda la información posible en cuanto a proyectos similares, elección de las herramientas, etc.

c) **Aprendizajes de las tecnologías**:

Entorno de desarrollo: Visual Studio 6

Lenguaje de programación: C++.

3DS MAX: programa usado en la creación de los modelos en 3D.

WBS Chart Pro: programa usado para la creación de los diagramas WBS.

ArgoUML: programa usado en la realización del modelado UML: diagrama de clases, de casos de uso, de actividades.

Eclipse: programa usado en la realización del modelado UML para crear los diagramas de secuencia mediante un plugin.

COSMOS: con esta aplicación se realizan las estimaciones de esfuerzo.

Microsoft Project: usado para la planificación temporal del proyecto.

d) **Diseño del proyecto**: modelado de las distintas partes que componen la aplicación.

e) C**odificación**: implementación en C++ de la aplicación según los modelos diseñados en la etapa anterior.

f) **Pruebas**: Tras codificar, se realizan una serie de pruebas para detectar posibles fallos y la posibilidad de realizar mejoras.

g) **Documentación**: documentar todo el trabajo realizado.

El inicio de las tareas es el día 1 de Septiembre de 2008. El plazo de entrega que tuve que cumplir era el 28 de Febrero de 2009. Para ese día, tenía que tener una versión operativa del programa con sus manuales de usuario lista para entregar. Tras la entrega, se le da al cliente un periodo de prueba tras el cual se realiza otra entrevista para informar de errores detectados por el cliente y posibles mejoras. Ese trabajo no se ve reflejado en la estimación temporal.

	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	Entrevistas con los directores de proyecto y el cliente	5 días	lun 01/09/08	vie 05/09/08	
2	Búsqueda bibliográfica	15 días	lun 01/09/08	vie 19/09/08	
3	Aprendizaje de las tecnologías	30 días	lun 01/09/08	vie 10/10/08	
4	Diseño del proyecto	15 días	lun 13/10/08	vie 31/10/08	3
5	Codificación	70 días	lun 03/11/08	vie 06/02/09	4
6	Pruebas	10 días	lun 09/02/09	vie 20/02/09	5
7	Documentacion	30 días	lun 23/02/09	vie 03/04/09	6

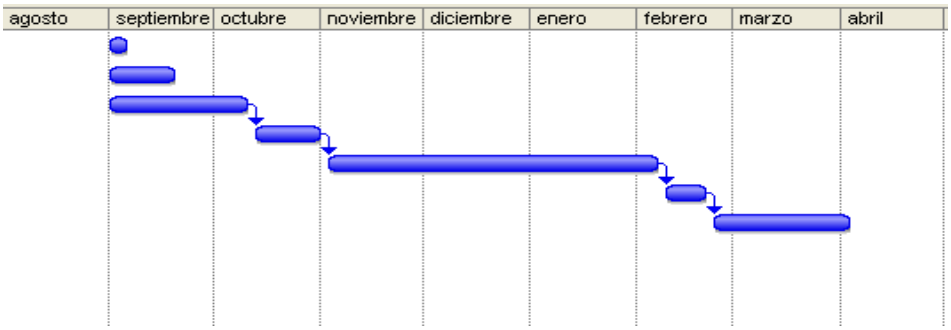


Figura 1.14. Diagrama de Gantt de la planificación del proyecto.

Se puede observar que la planificación comienza el día 01/09/2008 y acaba el 03/04/2009. Las 3 primeras tareas que son las entrevistas, búsqueda de bibliografía y aprendizaje de las tecnologías, comienzan a la vez, y pueden realizarse simultáneamente. Para las siguientes fases, es necesario completar estas 3 primeras tareas, y luego ir completando cada una antes de pasar a la siguiente.

La misma planificación la podemos ver en el diagrama PERT de la Figura 1.15:

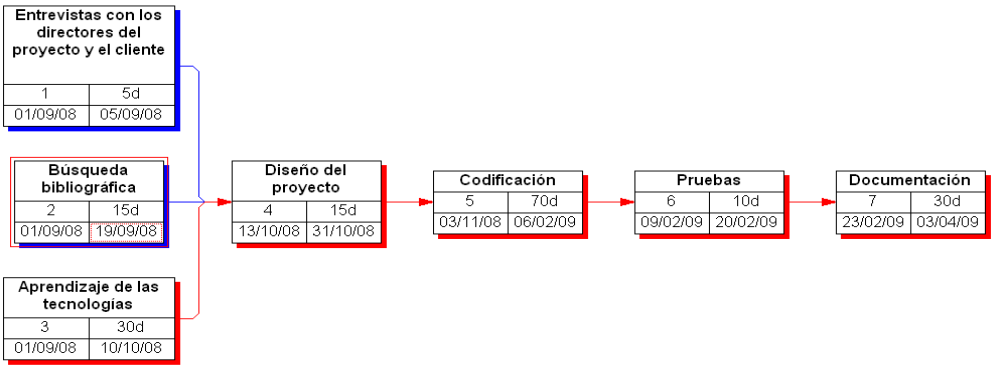


Figura 1.15. Diagrama PERT de la planificación del proyecto.

Capítulo 2

Metodología y resolución

Resumen

En este capítulo se realiza una descripción detallada del proyecto.

- En la primera sección se lleva a cabo una descripción de la arquitectura del sistema software.

- Diagramas de casos de uso, de clases, de secuencia y de actividades en UML.

- Descripción y detalle de los algoritmos de colisión.

2.1 Identificación, planteamiento y justificación

Se pretende crear una herramienta software capaz de realizar simulaciones en entornos virtuales 3D, que posteriormente será usada para evaluar una capacidad muy importante del cerebro humano: la memoria espacial y la memoria de trabajo.

Este entorno virtual estará formado por una habitación circular, de unos 8 metros de diámetro, en cuyo interior hay distribuidas 9 zonas. El tamaño de esas zonas puede variar entre pequeño, mediano o grande. La habitación junto con las zonas será diseñada en 3D Studio Max, diseñando además una habitación igual a la otra pero sin techo, necesaria para mostrar las trayectorias de los sujetos. Las paredes de la habitación estarán divididas en 12 partes iguales, de forma similar a las franjas horarias. En cada una de esas divisiones, podrán aparecer estímulos que ayuden a los sujetos a orientarse. Los estímulos también serán creados usando 3DS Max. Tanto los estímulos creados como los modelos de la habitación serán exportados en ".3DS". La configuración de la habitación podrá personalizarse mediante una serie de interfaces gráficas y almacenarse en archivos XML.

La aplicación constará de 2 tipos distintos de experimentos: los destinados a estudiar la memoria espacial y los destinados a estudiar la memoria de trabajo.

a) Experimentos de memoria espacial:

En los experimentos de memoria espacial, los estímulos y zonas premiadas son siempre los mismos a lo largo de los ensayos de que conste el experimento, variando solo la posición de salida del sujeto de forma aleatoria.

Para configurar los experimentos de memoria espacial, es necesario indicar los estímulos que aparecerán en cada una de las 12 divisiones de la habitación, y la localización de las zonas premiadas.

A la hora de lanzar un nuevo experimento, habrá que indicar el nombre del sujeto del experimento, el número de ensayos del experimento, la duración de los mismos, el tamaño de las zonas, la velocidad de desplazamiento, el modo de video y el archivo de configuración a usar.

El motor gráfico será capaz de leer todos estos datos y configurar los ensayos de forma adecuada. Permitirá también moverse por la habitación, permitiendo desplazamientos hacia adelante o rotaciones. Como consecuencia de ese movimiento, el sujeto encontrará zonas premiadas, recibiendo un mensaje indicando que ha encontrado una zona y cuantas le faltan por encontrar. Además, la zona encontrada aparecerá resaltada en el suelo con color verde. Los ensayos acaban cuando se encuentren todas las zonas premiadas o se acabe el tiempo establecido. Al finalizar un ensayo, se mostrará también un mensaje informativo.

Se añadirán además 2 ensayos adicionales al final del experimento. En el penúltimo ensayo no habrá *feedback*, o sea, que no se mostrará ningún mensaje al encontrar una zona ni se reproducirá ningún sonido ni se marcará la zona premiada en el suelo. Este ensayo se realiza para comprobar que el sujeto sabe realmente donde están las zonas premiadas. En el último ensayo, las zonas aparecerán marcadas en el suelo y si aparecerán los mensajes al encontrar las zonas. Este ensayo se realiza para comprobar que el sujeto sabe controlar el joystick.

Por último, se creará una página web con los resultados obtenidos mostrados de forma ordenada. Desde esta web, se permitirá la visualización de la trayectoria que han seguido los sujetos a lo largo de los distintos ensayos. Para ello, se muestra la vista en planta de la habitación y se muestran las trayectorias como líneas sobre el suelo de la habitación.

b) Experimentos de memoria de trabajo:

En los experimentos de memoria de trabajo, los estímulos serán siempre los mismos, pero para cada bloque habrá que configurar las posiciones de salida de los ensayos primero y segundo, la duración de los mismos, el tiempo entre un ensayo y el otro y las zonas premiadas para ese bloque.

Para configurar los experimentos de memoria de trabajo, será necesario indicar los estímulos que aparecerán en cada una de las 12 divisiones de la habitación y el número de bloques. Para cada uno de los bloques, habrá que indicar las posiciones de salida de los 2 ensayos, la duración de cada uno, el tiempo entre ensayos y las zonas premiadas del bloque.

A la hora de lanzar un nuevo experimento, se indicará el nombre del sujeto, el tiempo que trascurrirá entre bloques, el tamaño de las zonas, la velocidad de desplazamiento, el modo de video y el archivo de configuración a usar.

El motor gráfico será capaz de leer todos estos datos y configurar los ensayos de forma adecuada. Permitirá también moverse por la habitación, permitiendo desplazamientos hacia adelante o rotaciones. Como consecuencia de ese movimiento, el sujeto encontrará zonas premiadas, recibiendo un mensaje indicando que ha encontrado una zona y cuantas le faltan por encontrar. Además, la zona encontrada aparecerá resaltada en el suelo con color verde. Al comienzo y al final de cada bloque, aparecerá un mensaje informando del número de bloque y si se

pasa a otro bloque. Los bloques acaban cuando acaban los 2 ensayos de que consta un bloque. Los ensayos acaban cuando se encuentren todas las zonas premiadas o se acabe el tiempo establecido. Al finalizar un ensayo, se mostrará también un mensaje informativo.

Por último, se creará una página web con los resultados obtenidos mostrados de forma ordenada. Desde esta web, se permitirá la visualización de la trayectoria que han seguido los sujetos a lo largo de los distintos ensayos. Para ello, se muestra la vista en planta de la habitación y se muestran las trayectorias como líneas sobre el suelo de la habitación.

2.2 Usuarios y funciones del sistema.

En la aplicación habrá 3 usuarios: el sujeto, el investigador y el administrador. El administrador es el encargado de configurar todos los parámetros necesarios en los ensayos y de ver los resultados de los experimentos anteriores, el investigador se encarga de configurar y lanzar la aplicación y el sujeto se encarga de ejecutar la aplicación.

A continuación muestro las funciones que pueden realizar el administrador y una breve descripción:

1) Configurar el experimento de memoria espacial: permite al administrador establecer los estímulos que aparecerán en la habitación, el número de zonas premiadas y su ubicación, y el nombre con el que se guardará esa configuración.
2) Configurar el experimento de memoria de trabajo: permite al administrador establecer los estímulos que aparecerán en la habitación, el número de bloques de que constará el experimento, configurar los bloques y el nombre con el que se guardará esa configuración.
3) Configurar bloques: permite al administrador establecer la posición de salida para los 2 ensayos de que consta un bloque, el tiempo de cada ensayo, el número de zonas premiadas y su ubicación para cada bloque, y el tiempo entre los ensayos de cada bloque.
4) Limpiar configuración de memoria espacial: establece todos los estímulos a "Ninguno", el nombre de la configuración en blanco y todas las zonas premiadas a nulo.
5) Limpiar configuración de memoria de trabajo: establece todos los estímulos a "Ninguno", el nombre de la configuración en blanco y el número de bloques a 5.
6) Limpiar configuración de bloque: establece todas las zonas premiadas a nulo, las posiciones de inicio para ambos ensayos a 1, el tiempo de ambos ensayos a 60 y el tiempo entre ensayos a 5.
7) Guardar configuración de memoria espacial: crea un documento XML con la configuración seleccionada y con el nombre establecido por el administrador.
8) Guardar configuración de memoria de trabajo: crea un documento XML con la configuración seleccionada y con el nombre establecido por el administrador.
9) Ver resultados: el administrador puede ver los resultados obtenidos durante el experimento. El programa genera una web en la que aparece la información general del experimento: el

nombre del sujeto, el número de ensayos, el tiempo para cada ensayo, la velocidad de desplazamiento, el número de zonas premiadas, las zonas premiadas. Para cada uno de los ensayos aparecen el tiempo empleado en encontrar todas las zonas premiadas, la velocidad media de desplazamiento, la distancia recorrida y las zonas visitadas. Aparece además un link para ver las trayectorias.

10) Ver trayectoria: inicia la aplicación 3D mostrando una vista en planta de la habitación. Sobre la habitación, aparecerán en verde las zonas encontradas por el sujeto y una línea discontinua de color rojo que representa la trayectoria. Podrá ver paso a paso la trayectoria y hacer zoom.
11) Ver paso a paso la trayectoria: puede ver paso a paso la trayectoria que ha seguido un sujeto en un ensayo. Se mostrará paso a paso la línea discontinua que representa la trayectoria del sujeto.
12) Hacer zoom: el investigador puede hacer zoom sobre la vista de la trayectoria que siguió el sujeto durante el experimento.

El investigador podrá realizar las siguientes funciones:

1) Crear un nuevo experimento de memoria espacial: el investigador establece el nombre del sujeto que realizará el experimento, el número de ensayos de que constará el experimento, la duración de cada ensayo, el tamaño de las zonas premiadas, la velocidad de desplazamiento, el modo de video y el archivo de configuración a usar.
2) Crear un nuevo experimento de memoria de trabajo: el investigador establece el nombre del sujeto que realizará el experimento, el tamaño de las zonas premiadas, la velocidad de desplazamiento, el tiempo entre cada bloque, el modo de video y el archivo de configuración a usar.
3) Lanzar experimento de memoria espacial: lanza la aplicación 3D, pasándole la configuración del experimento.
4) Lanzar experimento de memoria de trabajo: lanza la aplicación 3D, pasándole la configuración del experimento.

A continuación muestro las funciones que puede realizar el otro usuario de la aplicación: el sujeto de experimento.

1) Desplazarse hacia adelante: el sujeto puede desplazarse (desplazar la cámara) hacia adelante por la habitación mediante el uso del teclado.
2) Rotar la cámara a la izquierda: el sujeto puede rotar la cámara a la izquierda.
3) Rotar la cámara a la derecha: el sujeto puede rotar la cámara a la derecha.
4) Descubrir zona premiada: como consecuencia de desplazarse, el sujeto descubrirá zonas premiadas en la habitación.

2.3 Representación en UML

El Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, *Unified Modeling Language*) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el OMG (Object Management Group). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables.

Es importante resaltar que UML es un "lenguaje" para especificar y no para describir métodos o procesos. Se utiliza para definir un sistema, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo.

Se puede aplicar en el desarrollo de software entregando gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como el Proceso Unificado Racional o RUP), pero no especifica en sí mismo qué metodología o proceso usar.

UML no puede compararse con la programación estructurada, pues UML significa Lengua de Modelación Unificada, no es programación, solo se diagrama la realidad de una utilización en un requerimiento. Mientras que, programación estructurada, es una forma de programar como lo es la orientación a objetos, sin embargo, la orientación a objetos viene siendo un complemento perfecto de UML, pero no por eso se toma UML sólo para lenguajes orientados a objetos

UML cuenta con varios tipos de diagramas, los cuales muestran diferentes aspectos de las entidades representadas.

En UML 2.0 hay 13 tipos diferentes de diagramas. Para comprenderlos de manera concreta, a veces es útil categorizarlos jerárquicamente, como se muestra en la Figura 2.1:

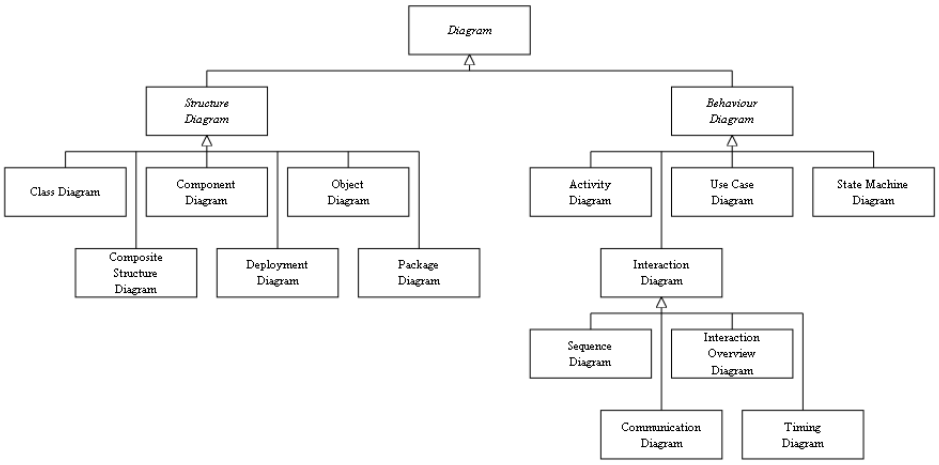


Figura 2.1. Diagramas UML organizados jerárquicamente.

2.3.1. Diagrama de casos de uso

Un diagrama de casos de uso es una representación gráfica de los actores y casos de uso del sistema, incluyendo sus interacciones. No son parte del diseño, sino parte del análisis, describiendo qué es lo que debe hacer el sistema.

El valor verdadero de un caso de uso reposa en dos áreas:

a) La descripción escrita del comportamiento del sistema al afrontar una tarea de negocio o un requisito de negocio. Esta descripción se enfoca en el valor suministrado por el sistema a entidades externas tales como usuarios humanos u otros sistemas.

b) La posición o contexto del caso de uso entre otros casos de uso. Dado que es un mecanismo de organización, un conjunto de casos de uso coherente, consistente promueve una imagen fácil del comportamiento del sistema, un entendimiento común entre el cliente/propietario/usuario y el equipo de desarrollo.

A continuación muestro el diagrama de casos de uso del proyecto. Se puede observar que las burbujas unidas directamente a los actores son las funciones que puede realizar directamente el actor, mientras que las burbujas unidas por <<include>> representan funciones internas desencadenadas por otras burbujas. Las burbujas apuntadas por una flecha discontinua indican dependencia de la burbuja de la cual sale la flecha. Para la realización del diagrama he usado la herramienta argoUML.

Aquí va el diagrama de casos de uso en A3

Vuelta del diagrama casos de uso

2.3.2. Diagrama de clases

Un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargaran del funcionamiento y la relación entre uno y otro.

A continuación muestro el diagrama de clases del proyecto:

Aquí va el diagrama de clases metido

Vuelta de pagina

Para entender este diagrama, es necesario explicar el papel que juega cada clase en el conjunto global del programa.

CBienvenida:

Clase que da la bienvenida al programa y muestra información del cliente. Esta ventana aparecerá siempre al inicio del programa. Hasta que no se cierre, no se mostrará la ventana principal del programa.

La Figura 2.2 muestra una captura de esta ventana:

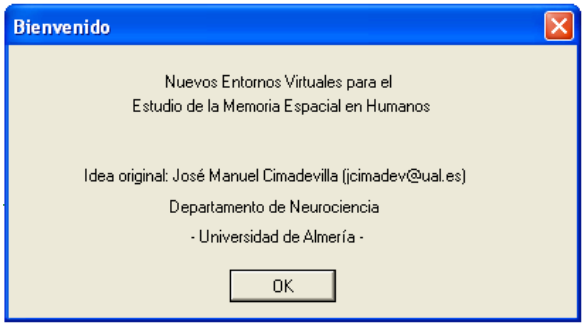


Figura 2.2. Captura de la ventana de bienvenida.

CAcercaDe

Clase que muestra información acerca del autor del programa. A continuación, se muestra una captura de la ventana en la Figura 2.3:

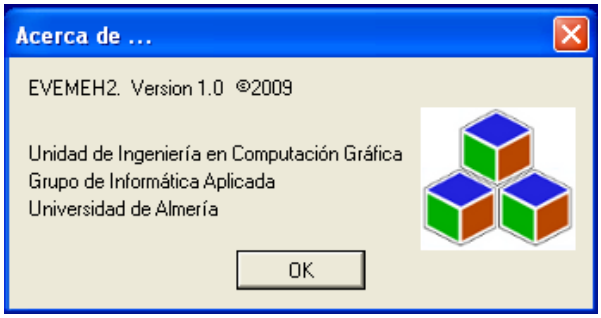


Figura 2.3. Captura de la ventana Acerca de…

CConfigurarExperimento.

Clase para la configuración de los experimentos para evaluar la memoria espacial. Consta de 12 menús desplegables, en los que aparece el nombre de los modelos almacenados en el ordenador. Para ello, tiene un puntero de tipo DIR al directorio "C:\EVEMEH2_9\Datos\Programa\Modelo3D\" que lee los nombres de los archivos y los añade a

los menús. De esta forma permite añadir nuevos modelos con solo añadirlos a ese directorio. Cada uno de esos 12 menús, representa las 12 zonas en las que se ha dividido la habitación y en las que pueden aparecer los estímulos. Por defecto aparece "Ninguno".

Consta también de una imagen que muestra la vista aérea de la habitación. En ella, se ven las 4 zonas de partida de los experimentos, las 12 divisiones para los estímulos y las posibles zonas premiadas, señaladas como círculos blancos. Al hacer clic sobre la imagen, en función de las coordenadas, se detecta si se ha pinchado sobre una zona premiada. Si es así, se cambia el valor de la variable asociada a cada una de las zonas premiadas, de verdadero a falso o viceversa. Aparecerá entonces un círculo con el borde verde si la variable asociada a esa zona es verdadera, o por el contrario, rojo si la variable asociada es falsa. Inicialmente todas las zonas tienen el valor falso.

El cuadro de texto que aparece es para introducir el nombre de la configuración. Hasta que no se introduzca algo en este cuadro, el botón "Guardar" estará deshabilitado. El botón "Reset" restablece todos los valores a sus valores iniciales.

Al pulsar "Guardar", se crea un archivo XML con el nombre introducido en el campo de texto. En él, se guardan los estímulos, guardando su posición y nombre del modelo, y las zonas premiadas.

Un ejemplo de archivo de configuración podría ser el mostrado en la Figura 2.4:

```
- <experimentoME>
  - <estimulos>
    - <estimulo>
        <zonaH>3</zonaH>
        <objeto>chimenea</objeto>
      </estimulo>
    - <estimulo>
        <zonaH>6</zonaH>
        <objeto>cuadro1</objeto>
      </estimulo>
    - <estimulo>
        <zonaH>9</zonaH>
        <objeto>losa</objeto>
      </estimulo>
    - <estimulo>
        <zonaH>12</zonaH>
        <objeto>lampara</objeto>
      </estimulo>
    </estimulos>
  - <zonasPremiadas>
      <zonaPremiada>1</zonaPremiada>
      <zonaPremiada>2</zonaPremiada>
      <zonaPremiada>6</zonaPremiada>
    </zonasPremiadas>
  </experimentoME>
```

Figura 2.4. Ejemplo de configuración de experimento de memoria espacial.

La Figura 2.5 muestra una captura de la ventana. Se puede observar en el Anexo 1.1:

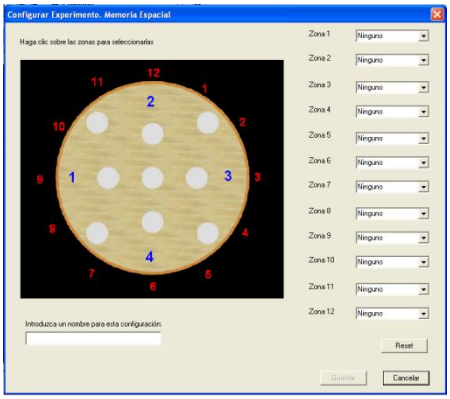


Figura 2.5. Captura de la ventana de configuración de experimentos de memoria espacial.

CConfigurarBloques

Clase para configurar un bloque de un experimento para evaluar la memoria de trabajo. Consta de 5 cuadros de texto, en los que se introducen las posiciones de salida de los 2 ensayos de que consta un bloque, la duración de cada uno de los 2 ensayos, y el tiempo entre ensayos.

Además aparece una imagen con la vista aérea de la habitación. En ella, se ven en azul las posiciones de salida, y círculos blancos que representan las zonas premiadas. Al hacer clic sobre la imagen, en función de las coordenadas, se detecta si se ha pinchado sobre una zona premiada. Si es así, se cambia el valor de la variable asociada a cada una de las zonas premiadas, de verdadero a falso o viceversa. Aparecerá entonces un círculo con el borde verde si la variable asociada a esa zona es verdadera, o por el contrario, rojo si la variable asociada es falsa. Inicialmente todas las zonas tienen el valor falso.

Al hacer clic en "Siguiente", se guarda la configuración del bloque en el archivo XML abierto por la clase que llama a esta. La Figura 2.6 muestra una captura de esta ventana, pudiéndose ver también en el Anexo 1.2:

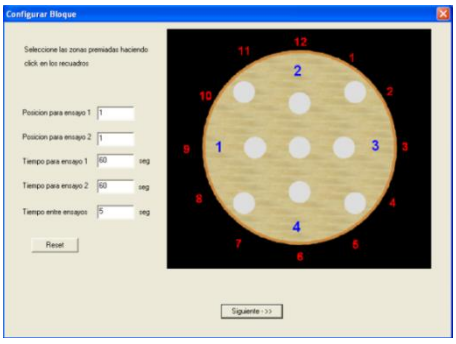


Figura 2.6. Captura de la ventana de configuración de bloques.

CConfigurarExperimentoTrabajo.

Clase para la configuración de los experimentos para evaluar la memoria de trabajo. Consta de 12 menús desplegables, en los que aparece el nombre de los modelos almacenados en el ordenador. Para ello, tiene un puntero de tipo DIR al directorio "C:\EVEMEH2_9\Datos\Programa\Modelo3D\" que lee los nombres de los archivos y los añade a los menús. De esta forma permite añadir nuevos modelos con solo añadirlos a ese directorio. Cada uno de esos 12 menús, representa las 12 zonas en las que se ha dividido la habitación y en las que pueden aparecer los estímulos. Por defecto aparece "Ninguno".

Consta también de una imagen que muestra la vista aérea de la habitación. En ella, se ven las 4 zonas de partida de los experimentos, las 12 divisiones para los estímulos y las posibles zonas premiadas, señaladas como círculos blancos. En este caso, sólo sirve de orientación a la hora de situar los estímulos en la habitación.

Los cuadros de texto que aparecen son para introducir el número de bloques del experimento y el nombre de la configuración. Hasta que no se introduzca algo en el cuadro del nombre, el botón "Configurar Bloques" estará deshabilitado. El botón "Reset" restablece todos los valores a sus valores iniciales.

Al pulsar "Configurar Bloques", se crea un archivo XML con el nombre introducido y se guardan los estímulos en él. Entonces esta clase crea tantas ventanas de configurar bloques como número de bloques se introdujo en el campo de texto. En cada una de las ventanas se leen las configuraciones y se guardan en el archivo XML.

Un ejemplo de archivo de configuración podría ser el mostrado en la Figura 2.7. Dicha imagen puede verse también en el Anexo 1.3.

```
- <experimentoMT>
  - <estimulos>
    - <estimulo>
        <zonaH>3</zonaH>
        <objeto>chimenea</objeto>
      </estimulo>
    - <estimulo>
        <zonaH>6</zonaH>
        <objeto>lampara</objeto>
      </estimulo>
    - <estimulo>
        <zonaH>9</zonaH>
        <objeto>chimenea</objeto>
      </estimulo>
    - <estimulo>
        <zonaH>12</zonaH>
        <objeto>lampara</objeto>
      </estimulo>
    </estimulos>
  - <bloques>
    - <bloque>
        <posicion1>1</posicion1>
        <posicion2>2</posicion2>
        <tiempo1>60</tiempo1>
        <tiempo2>60</tiempo2>
      - <zonasPremiadas>
          <zonaPremiada>2</zonaPremiada>
          <zonaPremiada>6</zonaPremiada>
        </zonasPremiadas>
        <tiempoEntreEnsayos>5</tiempoEntreEnsayos>
      </bloque>
    - <bloque>
        <posicion1>3</posicion1>
        <posicion2>4</posicion2>
        <tiempo1>60</tiempo1>
        <tiempo2>60</tiempo2>
      - <zonasPremiadas>
          <zonaPremiada>3</zonaPremiada>
          <zonaPremiada>5</zonaPremiada>
        </zonasPremiadas>
        <tiempoEntreEnsayos>5</tiempoEntreEnsayos>
      </bloque>
    </bloques>
</experimentoMT>
```

Figura 2.7. Ejemplo de configuración de experimento de memoria de trabajo.

La Figura 2.8, muestra una captura de esta clase, también recogida en el Anexo 1.4:

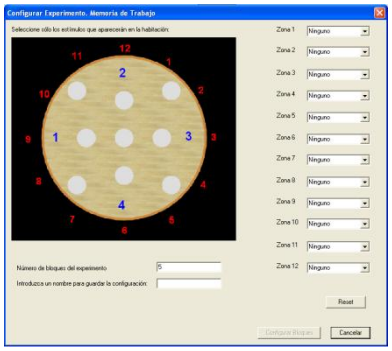


Figura 2.8. Captura de la ventana de configuración de experimentos de memoria de trabajo.

CExperimentosAnteriores

Clase que permite la elección de experimentos anteriores, tanto de memoria espacial, como de memoria de trabajo, para la visión de sus resultados. Consta de 2 menús desplegables que muestran los nombres de los archivos de datos almacenados en el equipo. Para ello, tiene un puntero de tipo DIR al directorio "C:\EVEMEH2_9\Datos\Resultados\Espacial" o "C:\EVEMEH2_9\Datos\Resultados\Trabajo", en función de si es el menú de memoria espacial o de memoria de trabajo, que lee los nombres de los archivos y los añade a los menús.

Al hacer clic en Ver, se lee el valor del menú desplegable y se llama a un visor de HTML, como Internet Explorer, para que muestre el archivo de datos.

La Figura 2.9 muestra una captura de esta clase. Esta captura aparece también en el Anexo 1.5.

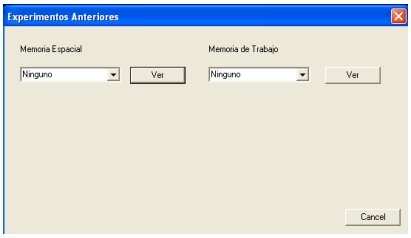


Figura 2.9. Captura de la ventana de ver experimentos anteriores.

CInterfaz

Clase que inicia el entorno gráfico para mostrar las ventanas.

CInterfazDlg

Clase principal encargada de crear y mostrar las ventanas correspondientes, según la opción elegida en el menú de la ventana. La Figura 2.10 o Anexo 1.6 muestra la interfaz principal:

Figura 2.10. Captura de la ventana principal.

CNuevoExperimento

Clase permite configurar un nuevo experimento para evaluar la memoria espacial. Esta clase es la que toma los atributos principales del experimento y lanza la aplicación 3D. Esta ventana consta de 3 cuadros de texto, en los que se introducen el nombre del sujeto del experimento, el número de ensayos y la duración de los mismos. También consta de una serie de radio buttons agrupados, que sirven para elegir: la velocidad de desplazamiento (rápida, normal o lenta), el tamaño de las zonas (grande, normal o pequeño) y el modo de video, que establece la resolución de la aplicación y si es en modo pantalla completa o ventana. Por último cuenta además con un menú desplegable en el que muestran los archivos de configuración para experimentos de memoria espacial almacenados en el ordenador. Para ello, tiene un puntero de tipo DIR al directorio "C:\EVEMEH2_9\Datos\Configuracion\Espacial" que lee los nombres de los archivos y los añade a los menús.

Al hacer clic en "Comenzar", se comprueba que el nombre del sujeto no esté repetido, que el número de ensayos sea al menos 1 y que la duración de los ensayos sea mayor de 0. Si se diera algún caso anterior, mostraría una ventana informativa de advertencia y no se lanzaría el programa. Si no, se leen los datos de los cuadros de texto, las variables asociadas a los radio buttons y el archivo de configuración, y se pasan como parámetros al ejecutable EVEMEHME.exe.

La Figura 2.11 muestra una captura de esta ventana, al igual que el Anexo 1.7:

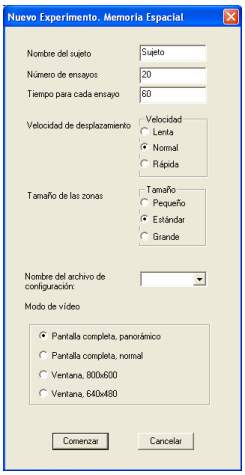


Figura 2.11. Captura de la ventana de crear un nuevo experimento de memoria espacial.

CNuevoExpTrabajo

Clase permite configurar un nuevo experimento para evaluar la memoria espacial. Esta clase es la que toma los atributos principales del experimento y lanza la aplicación 3D. Esta ventana consta de 2 cuadros de texto, en los que se introducen el nombre del sujeto del experimento y el tiempo a transcurrir entre bloques. También consta de una serie de radio buttons agrupados, que sirven para elegir: la velocidad de desplazamiento (rápida, normal o lenta), el tamaño de las zonas (grande, normal o pequeño) y el modo de video, que establece la resolución de la aplicación y si es en modo pantalla completa o ventana. Por último cuenta además con un menú desplegable en el que muestran los archivos de configuración para experimentos de memoria de trabajo almacenados en el ordenador. Para ello, tiene un puntero de tipo DIR al directorio "C:\EVEMEH2_9\Datos\Configuracion\Trabajo" que lee los nombres de los archivos y los añade a los menús.

Al hacer clic en "Comenzar", se comprueba que el nombre del sujeto no esté repetido y que el tiempo entre bloques sea al menos 0. Si se diera algún caso anterior, mostraría una ventana informativa de advertencia y no se lanzaría el programa. Si no, se leen los datos de los cuadros de texto, las variables asociadas a los radio buttons y el archivo de configuración, y se pasan como parámetros al ejecutable EVEMEHMT.exe.

La Figura 2.12 muestra una captura de esta ventana, al igual que el Anexo 1.8:

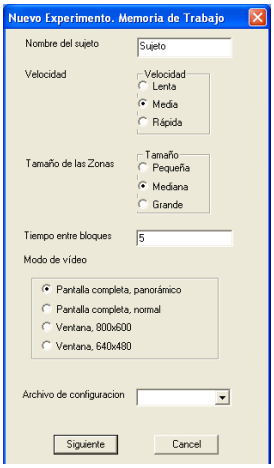


Figura 2.12. Captura de la ventana de crear un nuevo experimento de memoria de trabajo.

Cámara

Esta clase ofrece la funcionalidad necesaria para el manejo de la cámara. Tiene métodos para situar la cámara en un punto determinado y apuntando a una dirección determinada, otros métodos para rotar, desplazar o ladear la cámara. Además, se encarga de gestionar los eventos de pulsación de teclas. En esa gestión de eventos se incluye el ir guardando las trayectorias seguidas por el sujeto en un archivo de texto, ya que cada vez que se rota o se para y avanza se considera una nueva trayectoria.

Esfera

Esta clase contiene toda la lógica asociada al cálculo de colisiones. Para ello, tiene métodos para calcular la distancia de un punto a un plano, el ángulo que forman 2 vectores o para calcular el vector normal a un triangulo. También permite comprobar si existe intersección entre una línea y un plano y una línea y un polígono. Sirviéndose de esta funcionalidad, puede comprobar si existe colisión entre la esfera y el plano infinito de un polígono, o comprobar si existe colisión entre la esfera y el polígono en sí.

Principal

Es la clase que realiza la llamada a la clase **Juego3D**, en la cual está el bucle principal de la aplicación.

Juego3D

Contiene el bucle principal de la aplicación y es la que llama a las demás clases, controlando así los experimentos de memoria espacial. Antes de llegar al bucle principal, se ha de iniciar SDL para los gráficos y FSOUND para los sonidos, leer y cargar los modelos del archivo de configuración llamando a la clase **Pantalla**, guardar la configuración de los ensayos pasada como parámetros y leer las zonas premiadas desde el archivo de configuración. Se crea además el archivo HTML que contiene los resultados del experimento.

Dentro del bucle, se controlarán los eventos de teclado haciendo llamadas a **Cámara**, que también guardará la trayectoria que se sigua. Como consecuencia del movimiento de la cámara, se colisionará con las zonas premiadas, debiendo calcular si es una zona premiada o no. Para ir dibujando la escena, se harán llamadas a **Pantalla**. En el trascurso de los ensayos, se irán mostrando mensajes en pantalla llamando a **Texto**. Esta clase se encarga de ir creando nuevos ensayos y guardando los datos de cada uno.

Juego3D2

Contiene el bucle principal de la aplicación y es la que llama a las demás clases, controlando así los experimentos de memoria de trabajo. Antes de llegar al bucle principal, se ha de iniciar SDL para los gráficos y FSOUND para los sonidos, leer y cargar los modelos del archivo de configuración llamando a la clase **Pantalla**, guardar la configuración de los ensayos pasada como parámetros y leer la configuración de los bloques desde el archivo de configuración. Se crea además el archivo HTML que contiene los resultados del experimento.

Dentro del bucle, se controlarán los eventos de teclado haciendo llamadas a **Cámara**, que también guardará la trayectoria que se sigua. Como consecuencia del movimiento de la cámara, se colisionará con las zonas premiadas, debiendo calcular si es una zona premiada o no. Para ir dibujando la escena, se harán llamadas a **Pantalla**. En el trascurso de los ensayos, se irán mostrando mensajes en pantalla llamando a **Texto**. Esta clase se encarga de ir creando nuevos bloques y ensayos y guardando los datos de cada uno.

Trayectoria 1

Es la encargada de mostrar las trayectorias que siguen los sujetos durante los experimentos de memoria espacial. Funciona exactamente igual para trayectorias de experimentos de memoria espacial y experimentos de memoria de trabajo, ya que la forma de guardar y cargar las trayectorias es la misma. Contiene el bucle principal de la aplicación y es la que llama a las demás clases. Antes de llegar al bucle principal, se ha de iniciar SDL para los gráficos y FSOUND para los sonidos, leer y cargar los modelos del archivo de configuración llamando a la clase **Pantalla**, leer las zonas premiadas y trayectorias desde el archivo de trayectorias.

Dentro del bucle, se controlarán los eventos de teclado haciendo llamadas a **Cámara**. Para ir dibujando la escena, se harán llamadas a **Pantalla.** Las trayectorias se dibujan desde la misma clase con el método mostrarLineas()**.**

Trayectoria 2

Es la encargada de mostrar las trayectorias que siguen los sujetos durante los experimentos de memoria de trabajo. Funciona exactamente igual para trayectorias de experimentos de memoria espacial y experimentos de memoria de trabajo, ya que la forma de guardar y cargar las trayectorias es la misma. Contiene el bucle principal de la aplicación y es la que llama a las demás clases. Antes de llegar al bucle principal, se ha de iniciar SDL para los gráficos y FSOUND para los sonidos, leer y cargar los modelos del archivo de configuración llamando a la clase **Pantalla**, leer las zonas premiadas y trayectorias desde el archivo de trayectorias.

Dentro del bucle, se controlarán los eventos de teclado haciendo llamadas a **Cámara**. Para ir dibujando la escena, se harán llamadas a **Pantalla.** Las trayectorias se dibujan desde la misma clase con el método mostrarLineas()**.**

Modelo3DS

Clase que proporciona la funcionalidad necesaria para cargar los archivos “3DS”, los cuales contienen los modelos que se representarán en la aplicación y dibujarlos por pantalla. Consta de un método iniciar, que lee el número de objetos y texturas que contiene un modelo y reserva el espacio necesario. El método cargar, lee el archivo .3ds y va guardando los objetos y texturas que va leyendo en las estructuras destinadas a ello.

Estos modelos se pueden presentar en pantalla usando los métodos dibujar con o sin colisión. Ambos métodos realizan las mismas operaciones: cargan y activan las texturas y dibujan los triángulos mapeando las texturas sobre ellos. La diferencia es que en el dibujar con colisión, se comprueba si existe colisión entre la cámara y el objeto que se va a dibujar, en cuyo caso, se desplaza la cámara hacia atrás. Esta comprobación se realiza llamando a la clase **Esfera**.

Pantalla

Esta clase es la encargada leer el archivo de configuración los modelos que serán representados en la escena y de dibujar en pantalla esos modelos. Cuenta con métodos para contar cuántos modelos

contiene el archivo de configuración y reservar espacio para ellos. Permite también iniciar esos modelos, llamando a los métodos iniciar() y cargar() de la clase **Modelo3DS**.

Para dibujar los modelos en pantalla, llama a los métodos dibujar con colisión o sin colisión. Para el modelo 0, que será siempre el modelo de la habitación con las zonas premiadas, se llama al método dibujar colisión, pues en él están las zonas con las que hay que colisionar y la barrera que rodea a la habitación para que no se pueda salir de ella. Para los demás modelos, se rotan y trasladan según su posición en la habitación y se dibujan sin colisión, pues no es necesario ya que se colisionará con la barrera de la habitación y no contra ellos.

Texto

Esta clase es usada para representar texto en pantalla durante la ejecución del entorno 3D. Proporciona métodos para escribir texto en pantalla, y guardar y recuperar la matriz de proyección.

Textura

Esta clase ofrece la funcionalidad necesaria para cargar imágenes desde archivo y para aplicarlas a los objetos como texturas.

Vector

Esta clase proporciona representa un vector y todos los métodos necesarios para la manipulación de vectores como puede ser sumar, restar, módulo, producto vectorial, etc.

Vértice

Clase que representa un vértice mediante las 3 coordenadas del espacio.

Polígono

Clase que representa un polígono mediante sus 3 vértices.

Mapeado

Clase que representa el mapeado que sufre una textura al aplicarla sobre un polígono, definido por los valores u y v.

Objeto3D

Clase que representa un objeto tridimensional. El objeto constará de una serie de polígonos, los vértices de estos polígonos y los mapeados de las texturas de éstos.

2.3.3 Diagrama de secuencia

El diagrama de secuencia es uno de los diagramas más efectivos para modelar interacción entre objetos en un sistema. Un diagrama de secuencia muestra la interacción de un conjunto de objetos en una aplicación a través del tiempo y se modela para cada caso de uso. Mientras que el diagrama de casos de uso permite el modelado de una vista de negocio del escenario, el diagrama de secuencia contiene detalles de implementación del escenario, incluyendo los objetos y clases que se usan para implementar el escenario, y mensajes pasados entre los objetos.

Típicamente uno examina la descripción de un caso de uso para determinar qué objetos son necesarios para la implementación del escenario. Si tienes modelada la descripción de cada caso de uso como una secuencia de varios pasos, entonces puedes "caminar sobre" esos pasos para descubrir qué objetos son necesarios para que se puedan seguir los pasos. Un diagrama de secuencia muestra los objetos que intervienen en el escenario con líneas discontinuas verticales, y los mensajes pasados entre los objetos como flechas horizontales.

Los mensajes se dibujan cronológicamente desde la parte superior del diagrama a la parte inferior; la distribución horizontal de los objetos es arbitraria. En cada flecha, se coloca el nombre del método que está siendo llamado por un objeto en el otro. El método llamado, o invocado, pertenece a la definición de la clase instanciada por el objeto en la recepción final del mensaje.

Inicio de la aplicación y menú de administrador.

Al iniciar la aplicación, la ventana principal abre una ventana de bienvenida. Tras cerrar esta ventana de bienvenida, se muestra la ventana principal. La ventana principal consta de un menú en su parte superior. Este menú se divide en 3 partes: menú de administrador, menú de investigador y menú de ayuda. Desde el menú de administrador, se puede abrir la ventana de configuración de experimentos de memoria espacial, la ventana de configuración de experimentos de memoria de trabajo, la ventana para ver experimentos anteriores o también se puede salir de la aplicación.

Se puede apreciar mejor en la Figura 2.13:

Aquí va el A3 de diagrama secuencia administrador

Figura 2.13. Secuencia de las alternativas que ofrece el menú administrador.

Revés de la pagina

Crear archivos de configuración para experimentos de memoria espacial.

Al hacer clic sobre "Configurar Experimento. Memoria Espacial", la ventana principal crea y muestra la ventana de configuración de experimentos para memoria espacial. En esta ventana aparece un mapa de la habitación, visto desde arriba, en el que se seleccionan las zonas premiadas haciendo clic sobre ellas, y sirve también para ver dónde pueden ubicarse los estímulos en la habitación. Las zonas en las que no aparezca nada o en las que aparezca un círculo con el borde rojo son zonas no premiadas, y en las que aparezca un círculo verde son zonas premiadas. A la derecha de este mapa, aparecen una serie de menús desplegables. Haciendo clic en ellos y eligiendo el estimulo que se desee, ese estimulo aparecerá en la habitación en la zona correspondiente. El cuadro de texto que aparece bajo el mapa es para introducir el nombre que se usará para guardar la configuración. Hasta que no se introduzca algún carácter en este cuadro, el botón "Guardar" estará deshabilitado. Si no se eligiera ninguna zona o el nombre para el archivo de configuración estuviera repetido, al hacer clic en "Guardar", aparecerá un mensaje explicativo de advertencia y no se guarda la configuración. Al hacer clic sobre "Guardar", se creará el archivo de configuración con el nombre anteriormente introducido, y se guardará en él la posición de los estímulos y el estímulo en sí y las zonas premiadas. Tras esto, se muestra un mensaje indicando que se ha guardado el archivo correctamente, se cierra la ventana de configuración de experimentos para memoria espacial y se muestra la ventana principal.

En la Figura 2.14, se puede ver de forma gráfica la secuencia:

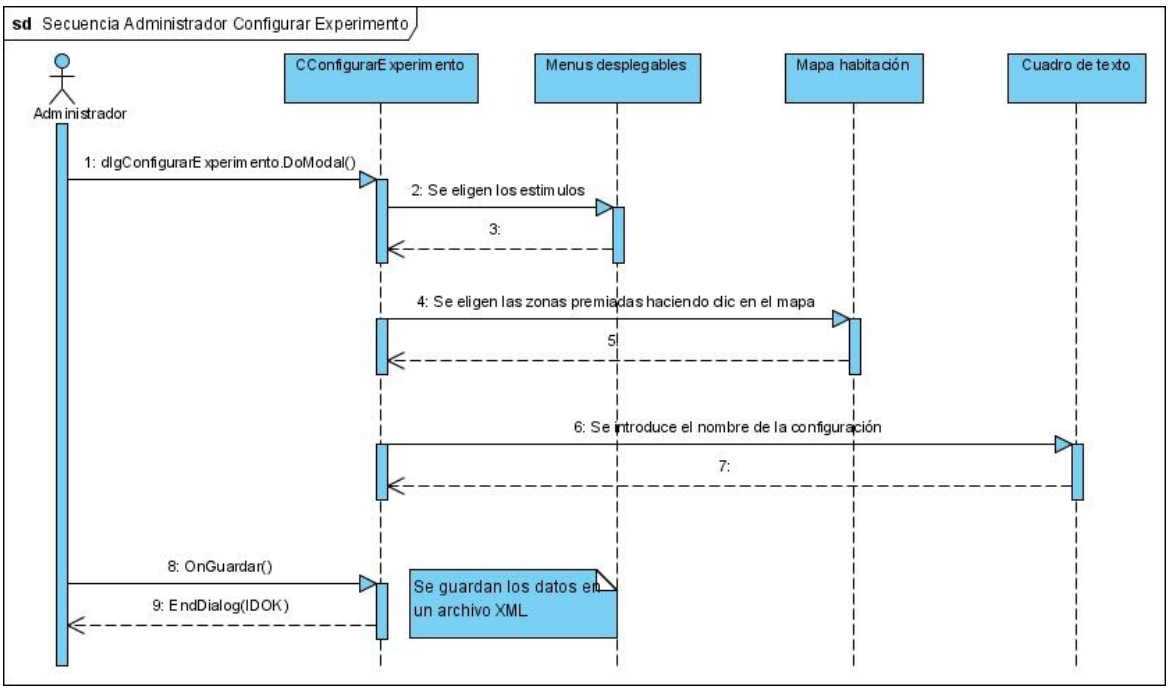


Figura 2.14. Diagrama de secuencia para crear un archivo de configuración para experimentos de memoria espacial.

Crear archivo de configuración para experimentos de memoria de trabajo.

Al hacer clic sobre "Configurar Experimento. Memoria Trabajo", la ventana principal crea y muestra la ventana de configuración de experimentos para memoria de trabajo. En esta ventana aparece un mapa de la habitación, visto desde arriba, que sirve para ver dónde pueden ubicarse los estímulos en la habitación. A la derecha de este mapa, aparecen una serie de menús desplegables. Haciendo clic en ellos y eligiendo el estimulo que se desee, ese estimulo aparecerá en la habitación en la zona correspondiente. Los cuadros de texto que aparece bajo el mapa sirven para introducir el nombre que se usará para guardar la configuración y el número de bloques de que contará el experimento. Hasta que no se introduzca algún carácter en el cuadro reservado para el nombre, el botón "Configurar bloques" estará deshabilitado. Si el número de bloques fuera menor o igual a 0, o el nombre para el archivo de configuración estuviera repetido, al hacer clic en "Configurar bloques", aparecerá un mensaje explicativo de advertencia. Al hacer clic sobre "Configurar bloques", se creará el archivo de configuración con el nombre anteriormente introducido, y se guardará en él la posición de los estímulos y el estímulo en sí.

Tras esto, se muestra la ventana de configuración de bloques tantas veces como número de bloques se introdujo anteriormente. En esta ventana aparece un mapa de la habitación visto desde arriba, en el que se seleccionan las zonas premiadas haciendo clic sobre ellas. Las zonas en las que no aparezca nada o en las que aparezca un círculo con el borde rojo son zonas no premiadas, y en las que aparezca un círculo verde son zonas premiadas. A la izquierda del mapa, aparecen una serie de cuadros de texto en los que se introducen las posiciones de salida de los 2 ensayos de que consta un bloque, el tiempo de cada ensayo y el tiempo entre ensayos. Si alguna posición fuera un número entero no comprendido entre 1 y 4, o algún tiempo fuera menor que 0, aparecerá un mensaje indicativo de advertencia al hacer clic en "Siguiente" y no se guarda la configuración. Al hacer clic en "Siguiente", se guarda los datos leídos de los cuadros de texto y las zonas premiadas, y se cierra la ventana.

La ventana de configuración de experimentos para memoria de trabajo, volverá a abrir otra ventana para configurar el bloque, hasta que se hayan configurado todos los bloques, entonces mostrará un mensaje indicando que se ha guardado correctamente y cerrará la ventana, dejando el programa en la ventana principal.

La Figura 2.15 ilustra todo el proceso:

A3 del secuencia administrador configurar exp trabajo

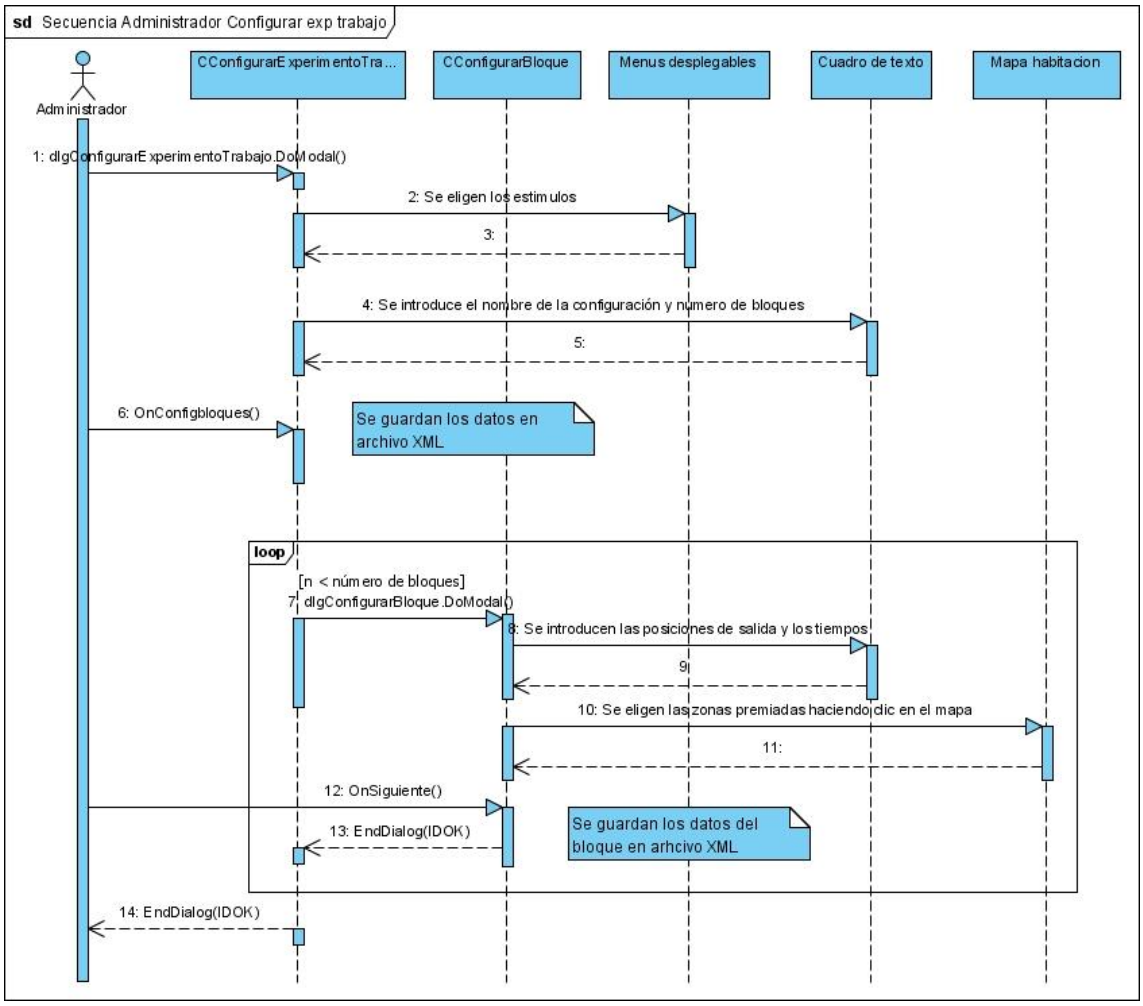


Figura 2.15. Diagrama de secuencia de crear configuración para experimentos de memoria de trabajo.

Revés de la pagina

Ver datos de los experimentos.

Al hacer clic sobre "Ver experimentos anteriores", la ventana principal crea y muestra la ventana para ver los datos de experimentos anteriores. Esta ventana consta de 2 menús desplegables, que muestran los nombres de los experimentos. Cada menú muestra un tipo de experimentos, el de la izquierda muestra los de memoria espacial y el de la derecha los de memoria de trabajo. Sin atender a qué menú escogemos, se hace clic en el menú y se elige el experimento a ver. Tras esto, se hace clic en "Ver". Al hacer esto, se lanza un visor de documentos HTML, que muestra los datos del experimento seleccionado del menú desplegable.

La Figura 2.16 muestra gráficamente el proceso:

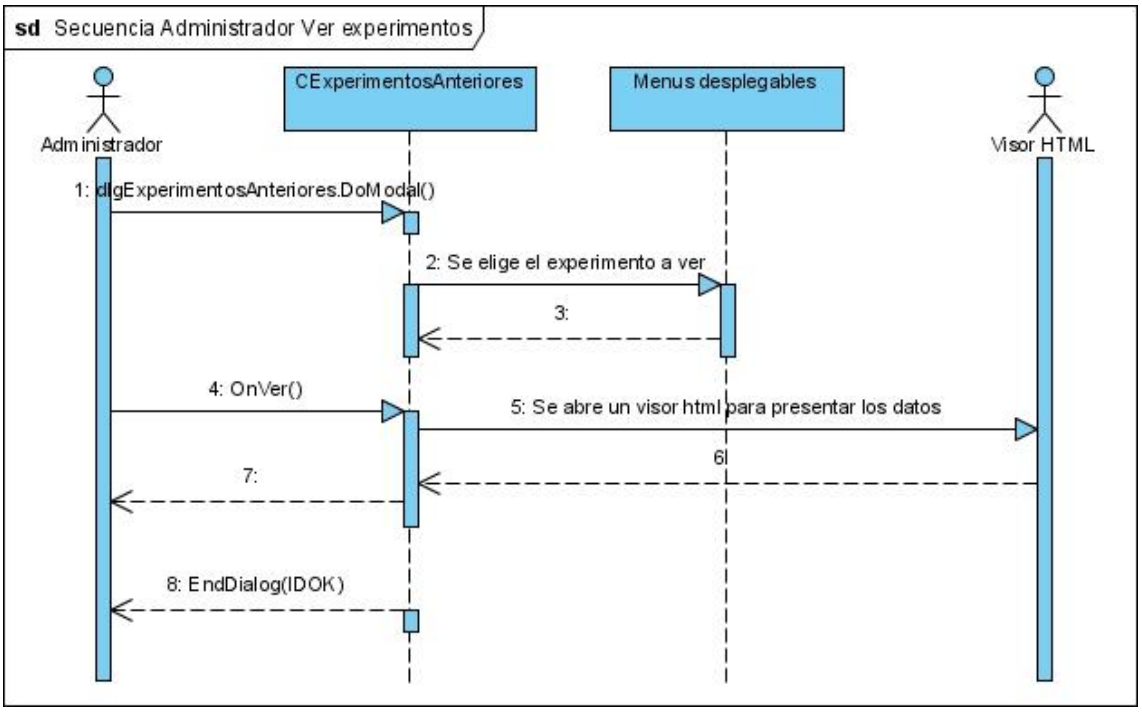


Figura 2.16. Diagrama de secuencia para ver experimentos anteriores.

Inicio de la aplicación y menú de investigador.

Al iniciar la aplicación, la ventana principal abre una ventana de bienvenida. Tras cerrar esta ventana de bienvenida, se muestra la ventana principal. La ventana principal consta de un menú en su parte superior. Este menú se divide en 3 partes: menú de administrador, menú de investigador y menú de ayuda. Desde el menú de investigador, se puede abrir una ventana para crear experimentos de memoria espacial y otra para crear experimentos de memoria de trabajo.

El diagrama de la Figura 2.17 ilustra la secuencia:

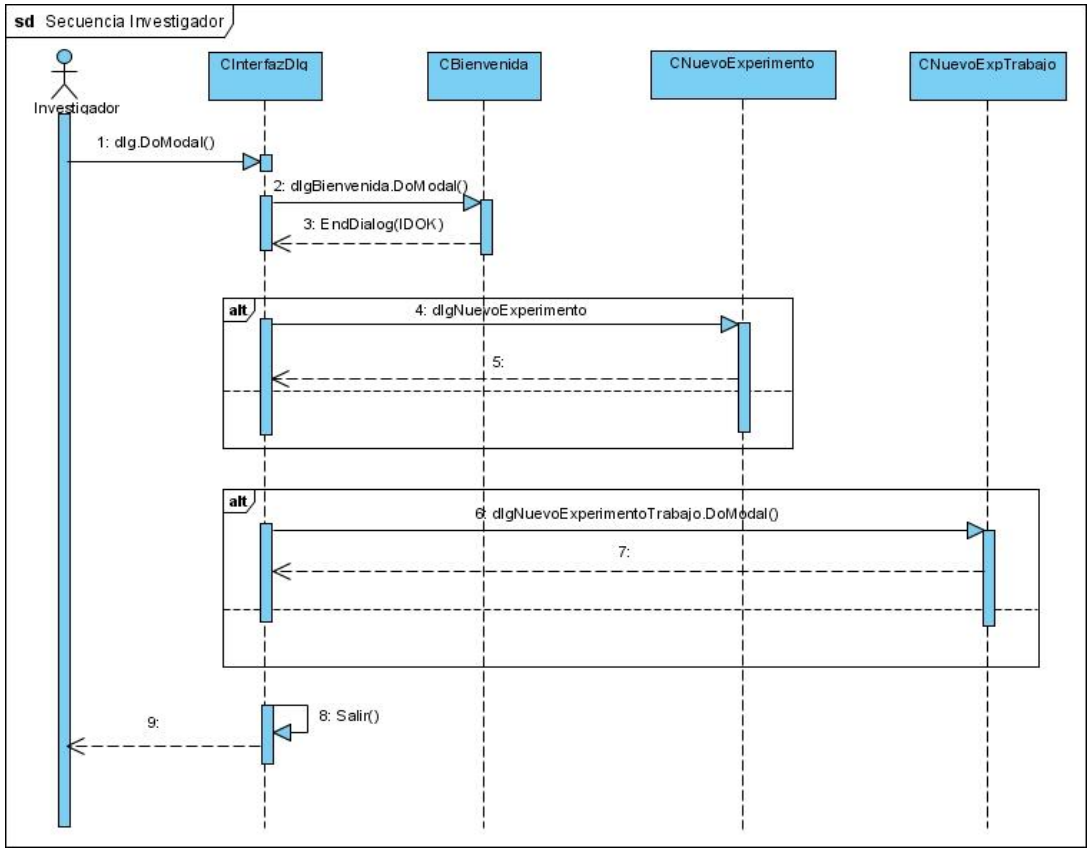


Figura 2.17. Diagrama de secuencia para el menú de investigador

Nuevo experimento de memoria espacial.

Al hacer clic sobre "Nuevo Experimento. Memoria Espacial", la ventana principal crea y muestra la ventana para crear un nuevo experimento de memoria espacial. Esta ventana consta de 3 cuadros de texto, en los que se introducen el nombre del sujeto del experimento, el número de ensayos y la duración de los mismos. También consta de una serie de radio buttons agrupados, que sirven para elegir: la velocidad de desplazamiento (rápida, normal o lenta), el tamaño de las zonas (grande, normal o pequeño) y el modo de video, que establece la resolución de la aplicación y si es en modo pantalla completa o ventana. Por último cuenta además con un menú desplegable en el que muestran los archivos de configuración para experimentos de memoria espacial almacenados en el ordenador.

Al hacer clic en "Comenzar", se comprueba que el nombre del sujeto no esté repetido, que el número de ensayos sea al menos 1 y que la duración de los ensayos sea mayor de 0. Si se diera algún caso anterior, mostraría una ventana informativa de advertencia y no se lanzaría el programa. Si no, se leen los datos de los cuadros de texto, las variables asociadas a los radio buttons y el archivo de configuración, y se pasan como parámetros al ejecutable EVEMEHME.exe. El diagrama de la Figura 2.18 ilustra todo el proceso:

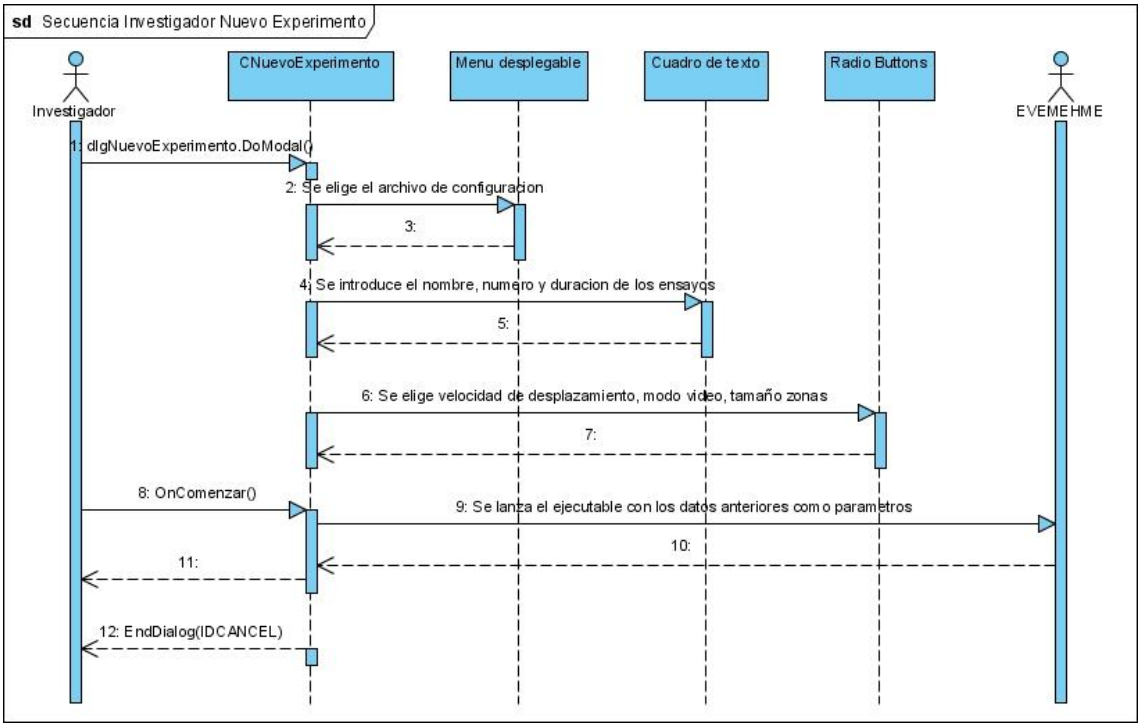


Figura 2.18. Diagrama de secuencia para crear un nuevo experimento de memoria espacial.

Nuevo experimento de memoria de trabajo.

Al hacer clic sobre "Nuevo Experimento. Memoria Trabajo", la ventana principal crea y muestra la ventana para crear un nuevo experimento de memoria de trabajo. Esta ventana consta de 2 cuadros de texto, en los que se introducen el nombre del sujeto del experimento y el tiempo a transcurrir entre bloques. También consta de una serie de radio buttons agrupados, que sirven para elegir: la velocidad de desplazamiento (rápida, normal o lenta), el tamaño de las zonas (grande, normal o pequeño) y el modo de video, que establece la resolución de la aplicación y si es en modo pantalla completa o ventana. Por último cuenta además con un menú desplegable en el que muestran los archivos de configuración para experimentos de memoria de trabajo almacenados en el ordenador.

Al hacer clic en "Comenzar", se comprueba que el nombre del sujeto no esté repetido y que el tiempo entre bloques sea al menos 0. Si se diera algún caso anterior, mostraría una ventana informativa de advertencia y no se lanzaría el programa. Si no, se leen los datos de los cuadros de texto, las variables asociadas a los radio buttons y el archivo de configuración, y se pasan como parámetros al ejecutable EVEMEHMT.exe.

La Figura 2.19 ilustra todo el proceso:

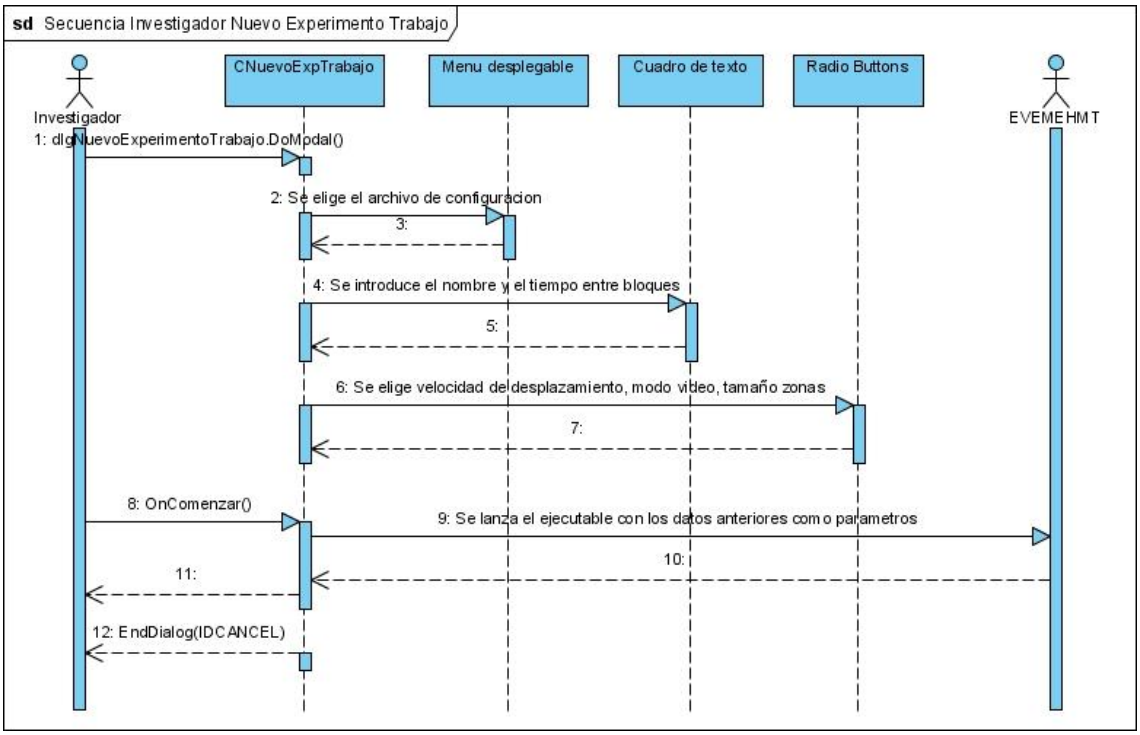


Figura 2.19. Diagrama de secuencia para crear un experimento de memoria de trabajo.

Inicio de la aplicación y menú de ayuda.

Al iniciar la aplicación, la ventana principal abre una ventana de bienvenida. Tras cerrar esta ventana de bienvenida, se muestra la ventana principal. La ventana principal consta de un menú en su parte superior. Este menú se divide en 3 partes: menú de administrador, menú de investigador y menú de ayuda. Desde el menú de ayuda, se puede abrir el manual de usuario del programa o una ventana que contiene información del programa.

La Figura 2.20 muestra la secuencia:

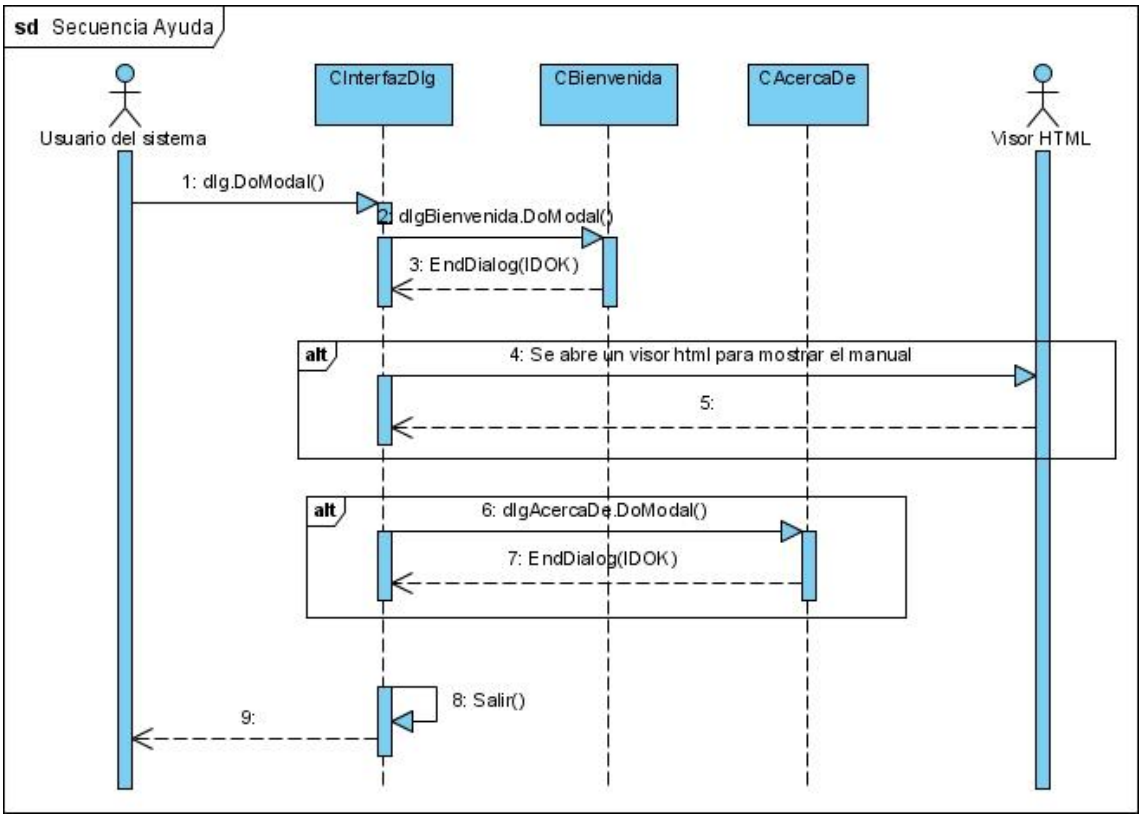


Figura 2.20. Diagrama de secuencia del menú de ayuda.

Iniciar aplicación 3D.

A continuación, se explica la secuencia de pasos que se realizan para iniciar la aplicación 3D y dejarla lista para ejecutar el bucle principal del programa.

La clase **principal** contiene el main(). En ella se reciben los parámetros introducidos al lanzar EVEMEHME o EVEMEHMT. Esta clase crea un objeto de tipo **Juego3D** y llama al método ejecutar, pasándole los parámetros recibidos al lanzar el ejecutable. El método ejecutar llama a su vez al método iniciarSDL(archivoEstimulos) también de **Juego3D**. Este método, inicia SDL, el modo de video, el tipo de letra que se usará en los mensajes, activa los sonidos, configura las texturas, inicia la textura que se usará para mostrar los mensajes por pantalla, activa la repetición de teclas y llama al método iniciar(c, archivoEstimulos) de la clase **Pantalla**.

El método iniciar de **Pantalla**, crea un puntero al archivo XML de configuración para recorrerlo recursivamente y leer el número de modelos que contiene. **Pantalla** reservará memoria para ese número de modelos. Una vez reservada la memoria, se pasa a iniciar y cargar cada uno de los modelos. Esto se hace desde la clase **Modelo3DS**.

Con el método iniciar de **Modelos3DS**, se lee el archivo .3ds y se reserva memoria para los objetos y texturas que contiene archivo. Con el método cargar, se guardan en objetos3d y texturas, cuya memoria acabamos de reservar, los objetos y texturas que contiene el archivo.

Tras iniciar y cargar los modelos, de **Modelo3DS** se vuelve a **Pantalla** y de **Pantalla** a **Juego3D**. Se llama entonces al método leerConfig() también perteneciente a **Juego3D**. LeerConfig() guarda el nombre del sujeto, el número de ensayos, la duración de los ensayos, el tamaño de las zonas y la velocidad de desplazamiento en variables propias de la clase. Para leer las zonas premiadas, se crea un puntero al archivo XML de configuración y se cuentan de forma recursiva las zonas premiadas que hay. Una vez contadas, se reserva espacio y se pasa a leer las zonas y guardarlas en las estructuras que tenemos para ellas. Una vez leídos y guardados todos los datos, se llama a CrearWeb(), también de Juego3D, que se encarga de guardar estos datos en un archivo HTML, que servirá después para ver los resultados del experimento.

La Figura 2.21 muestra de forma gráfica el proceso antes descrito.

A 3

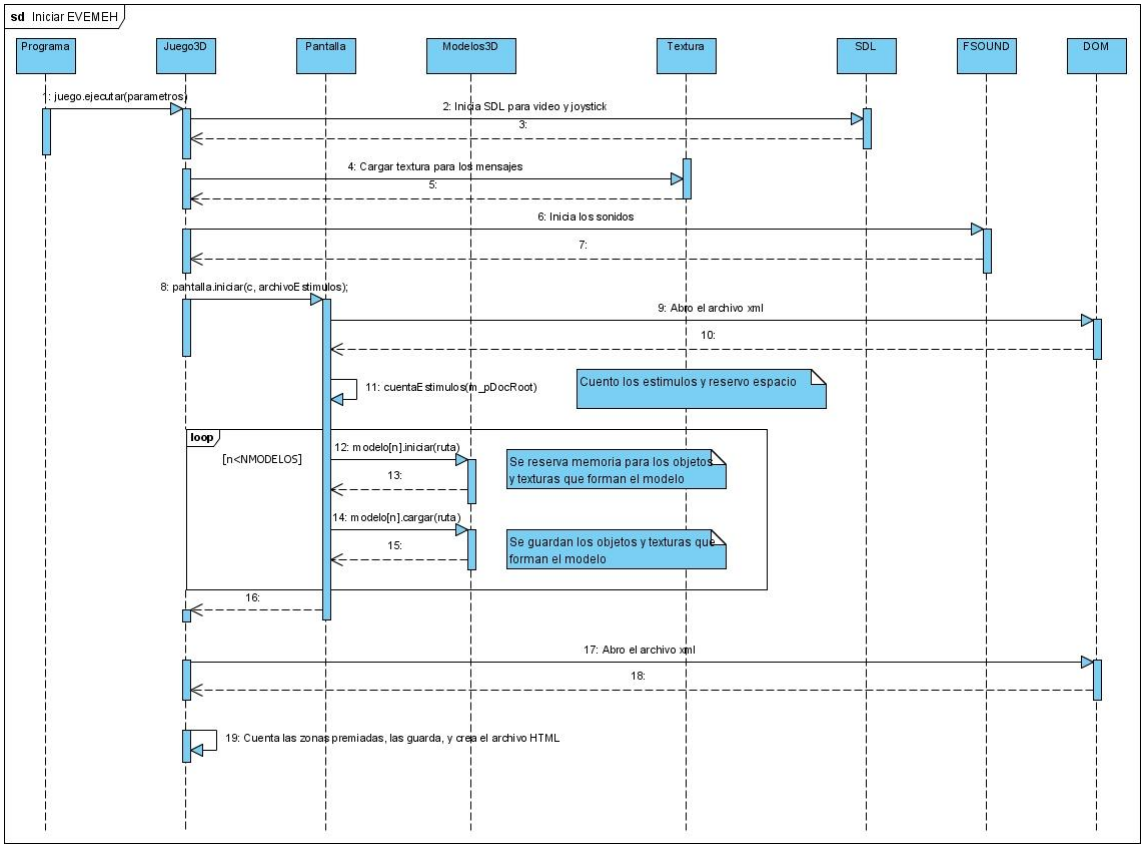


Figura 2.21. Diagrama de secuencia de iniciar aplicación 3D.

Revés de la pagina

Bucle principal del juego.

Se parte del apartado anterior en el que se dejaban los estímulos y las zonas cargadas.

El bucle principal se repite mientras el número de ensayos actual sea menor o igual al número de ensayos pasado como parámetro y no hayan finalizado todos los ensayos. Si fuera un experimento de memoria de trabajo, se comprobaría si es un experimento impar para leer la configuración del bloque, siendo el resto del proceso igual. Se llama a SDL_PollEvent() para capturar eventos. Una vez capturado el evento, se comprueba que tipo de evento es.

Si es pulsación de una tecla, se comprueba si es la tecla Up, y si lo es, se llama a descubirZona() de la misma clase y al método teclaAbajo($event,aumentarTiempoG) de la clase **Cámara**. Si no era la tecla Up, solo se llama a teclaAbajo($event,aumentarTiempoG).

Si es movimiento del joystick, se comprueba si es hacia adelante, y si lo es, se llama a descubirZona() de la misma clase y al método teclaAbajo($event,aumentarTiempoG) de la clase **Cámara**.Si al descubrirZona() se encontraran todas las zonas, comenzaría un nuevo ensayo. Si no era hacia adelante, solo se llama a teclaAbajo($event,aumentarTiempoG). Si es soltar una tecla, se llama a teclaArriba(&event.key.keysym) de la clase **Cámara**.

Una vez capturado el evento, se llama al método dibujarEscena() de la misma clase **Juego3D**. Si el tiempo actual fuera igual al tiempo de ensayo, comenzaría un nuevo ensayo, guardándose los datos del anterior en la web de resultados.

Una vez finalizados todos los ensayos, se llama al método finalizarTrayectoria() de la clase **Cámara**, que introduce los últimos datos en el archivo de trayectorias. Tras esto, solo queda finalizar SDL, los sonidos, el subsistema de joystick y todo lo que se hubiera abierto antes.

La Figura 2.22 muestra de forma gráfica el proceso antes descrito.

A3

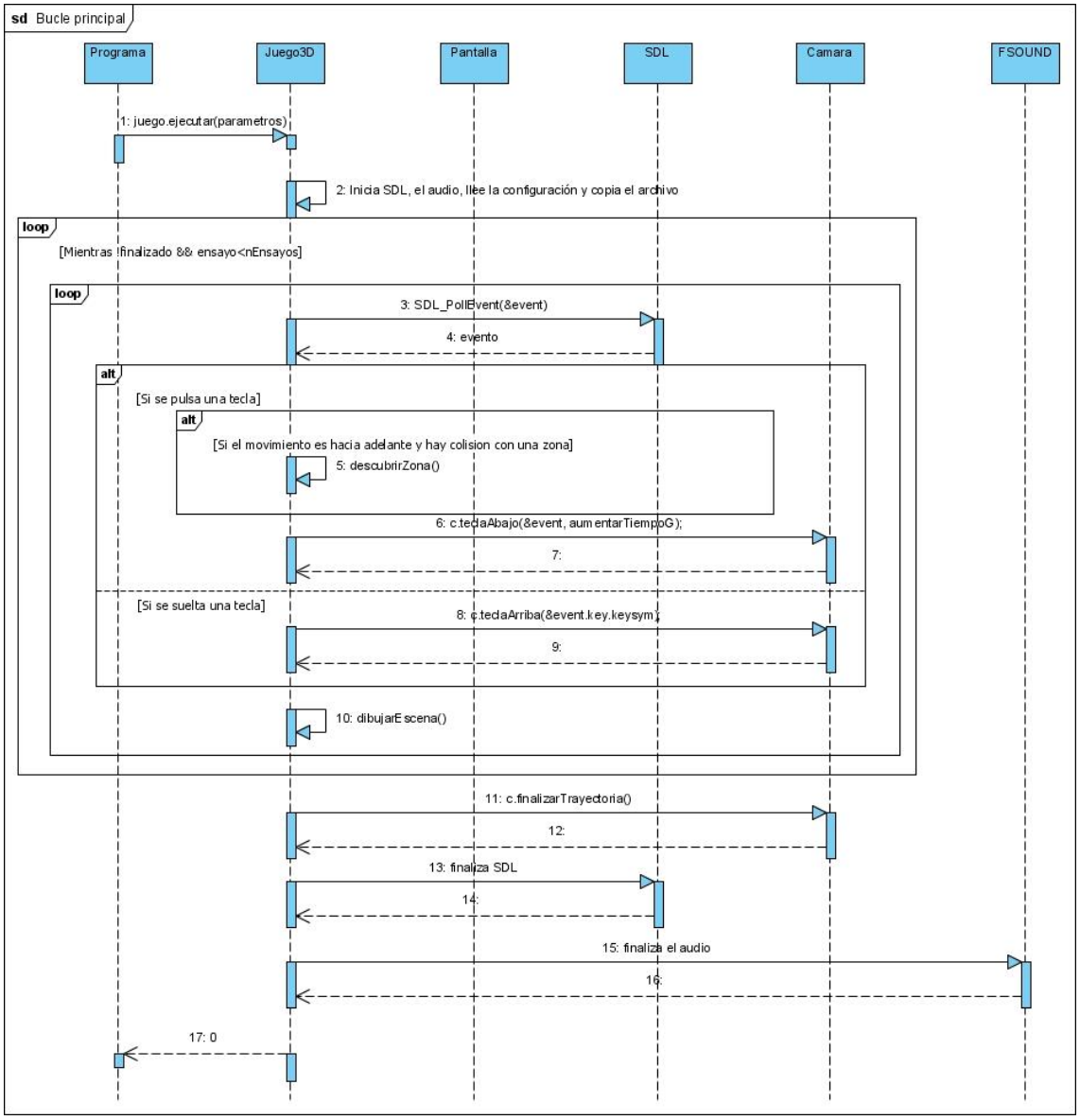


Figura 2.22. Diagrama de secuencias del bucle principal

Revés de la pagina

Dibujar la escena.

Para dibujar la escena es necesario calcular los fotogramas por segundo, que delimitarán la velocidad de la escena. Para ello, se llama al método calcularFPS(). Este método llama a SDL_GetTicks, para obtener el número de milisegundos transcurrido desde que se inició SDL. Con este tiempo, calculamos la velocidad de la escena, y además sirve para comprobar si el tiempo actual del ensayo es menor que el tiempo de ensayo establecido, en cuyo caso, se inicia un nuevo ensayo.

Tras calcular la velocidad de la escena, se limpia la pantalla y se actualiza hacia donde mira la cámara. Para ello se llama a los métodos actualizar() y mirar() de la clase **Cámara**. El método actualizar(), incluye el método comprobarMovimiento() que se encarga de desplazar o rotar la cámara. El método mirar, apunta hacia donde la cámara después de rotarla o desplazarla.

Una vez actualizada la cámara, se procede a dibujar los objetos 3d. Para ellos se llama al método dibujar() de la clase **Pantalla**. Para el modelo 0, que es la base de la habitación, se llama al método dibujarConColision() de la clase **Modelo3DS,** ya que en este modelo es en el que se encuentran las zonas premiadas con las que hay que colisionar para encontrarlas. DibujarConColisión() primero activa las texturas de los objetos, luego comprueba si hay colisión llamando al método detectarcolision() de la clase **Esfera**, y luego llama a dibujarTriángulos(), que lo que hace es leer y dibujar los triángulos que forman los objetos. Si hubo colisión, se devuelve true, sino false. Para los demás modelos, primero se rotan y trasladan y luego se dibujan sin tener en cuenta las colisiones con el método dibujarSinColision() también de la clase **Modelo3DS**. DibujarSinColision() activa las texturas de los objetos, y luego los dibuja.

Si hubo colisión, se retrocede la cámara llamando otra vez a actualizar() de la clase Cámara. Tras esto, se llama al método mostrarMensajes() que muestra por pantalla un mensaje si se dan una serie de condiciones. Por ejemplo, indica si se encontró una zona o acabó el ensayo.

Puede verse todo el proceso en la Figura 2.23:

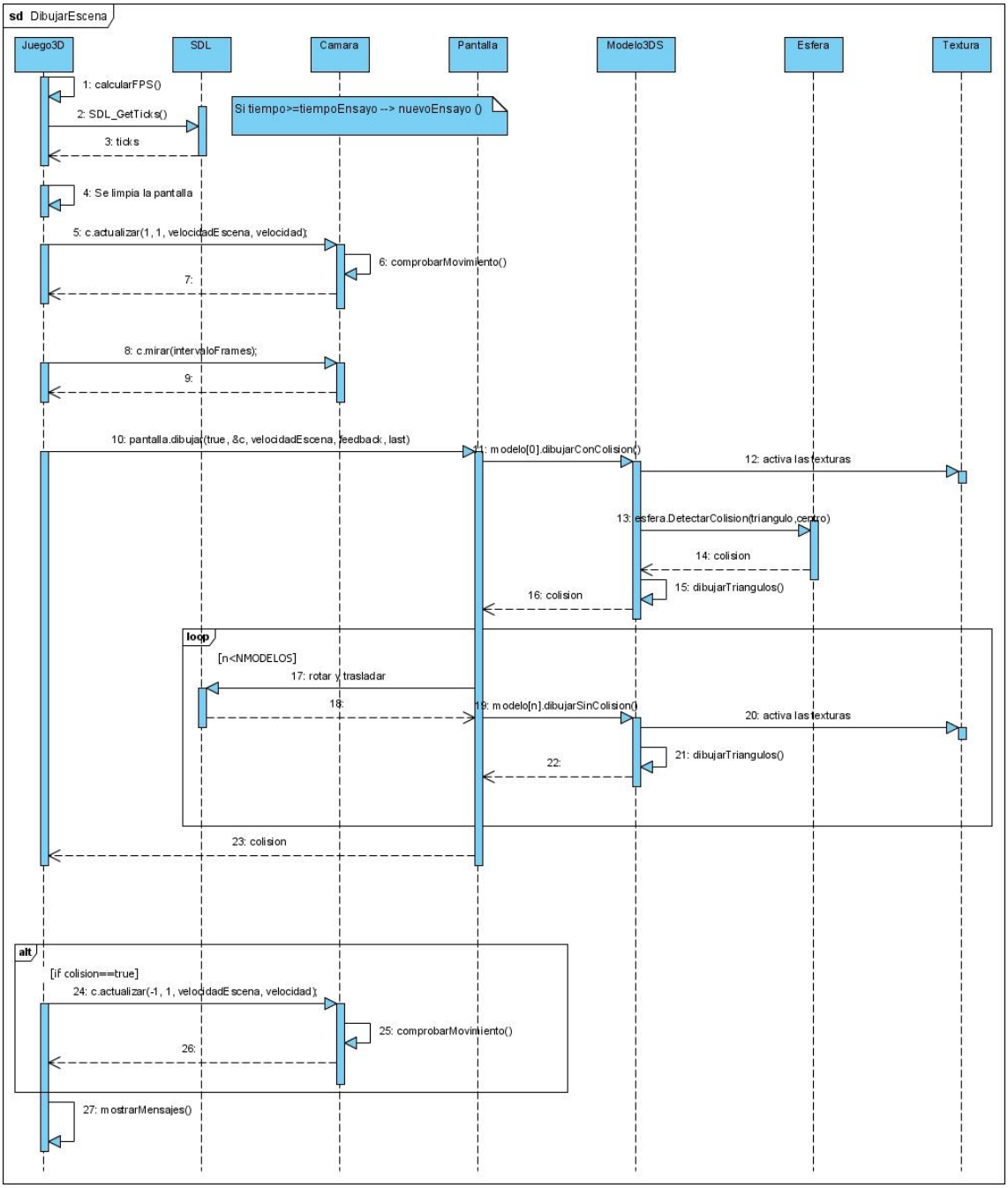


Figura 2.23. Diagrama de secuencia de dibujar escena.

Revés de la pagina

Ver las trayectorias.

La clase **principal** contiene el main(). En ella se reciben los parámetros introducidos al lanzar Trayectoria1 o Trayectoria2. Esta clase crea un objeto de tipo **Trayectoria1** o **Trayectoria2** y llama al método ejecutar, pasándole los parámetros recibidos al lanzar el ejecutable. El método ejecutar llama a su vez al método iniciarSDL(archivoEstimulos) también de **Trayectoria1 o Trayectoria2.** Ya se ha comentado anteriormente cómo funciona iniciarSDL por lo que omito esa parte.

El bucle principal se repite mientras no se pulse la tecla escape. Se llama a SDL_PollEvent() para capturar eventos. Una vez capturado el evento, se comprueba que tipo de evento es.

Si es pulsación de una tecla, se comprueba si es la tecla Enter, y si lo es, se aumenta el número de líneas a pintar y se llama al método teclaAbajo($event,aumentarTiempoG) de la clase **Cámara**. Si no era la tecla Enter, solo se llama a teclaAbajo($event,aumentarTiempoG).

Si es soltar una tecla, se llama a teclaArriba(&event.key.keysym) de la clase **Cámara**.

Una vez capturado el evento, se llama al método dibujarEscena() de la misma clase **Trayectoria1**. Este método funciona igual que anteriormente. El método que si ha cambiado es mostrarMensajes() de dibujarEscena(), que ahora se llama mostrarLineas() se ocupa de pintar las trayectorias leídas. Pintará tantas trayectorias como número de líneas a pintar haya.

Si se pulsa la tecla escape, se finaliza SDL, los sonidos, el subsistema de joystick y todo lo que se hubiera abierto antes.

La Figura 2.24 muestra de forma gráfica el proceso antes descrito.

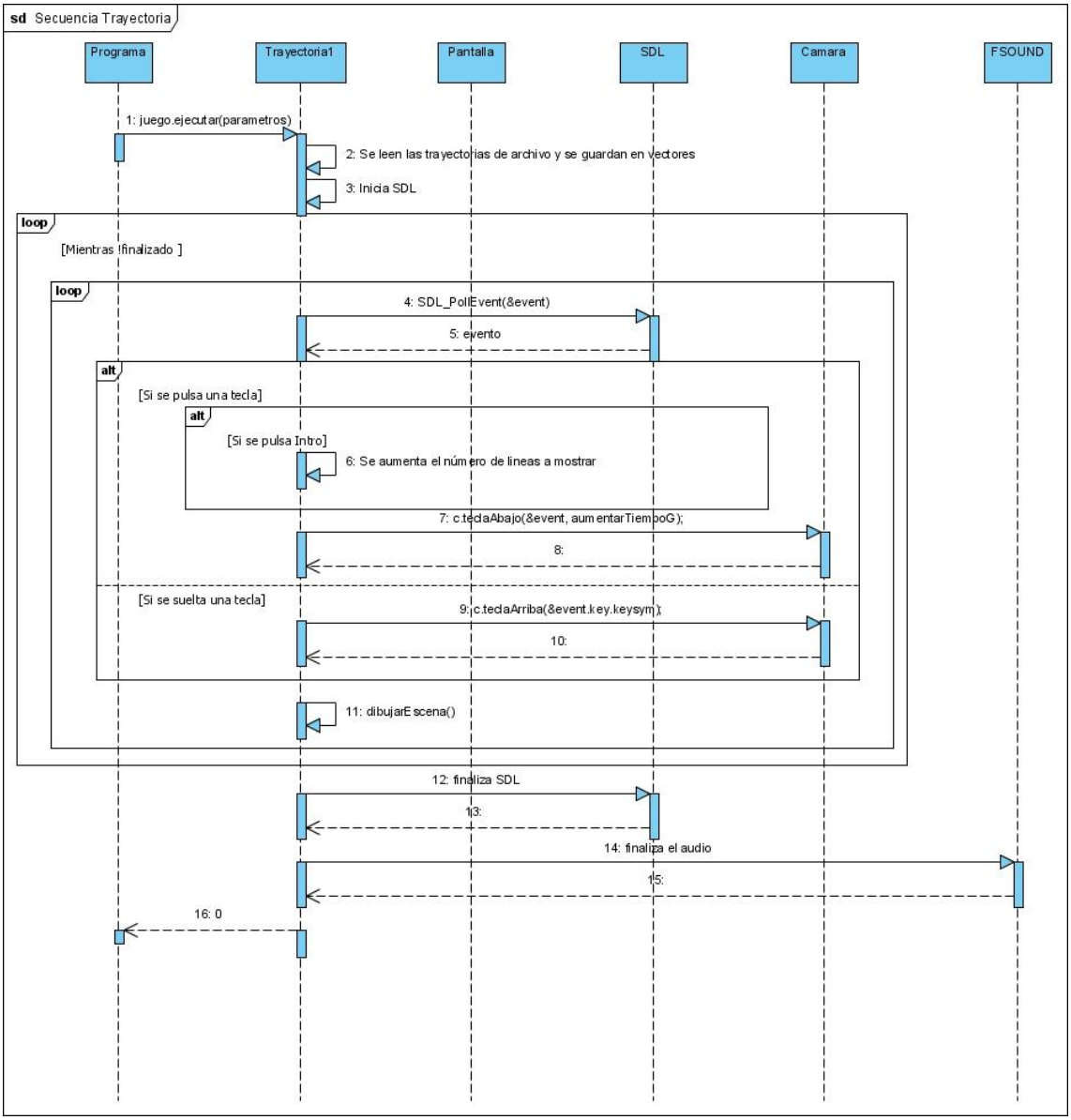


Figura 2.24 Diagrama de secuencia de ver trayectoria.

Revés de la pagina

2.3.4. Diagrama de actividades

Los diagramas de actividad describen la secuencia de las actividades en un sistema. Los diagramas de actividad son una forma especial de los diagramas de estado, que únicamente (o mayormente) contienen actividades.

La Figura 2.25 describe la secuencia de actividades a seguir durante la ejecución del programa.

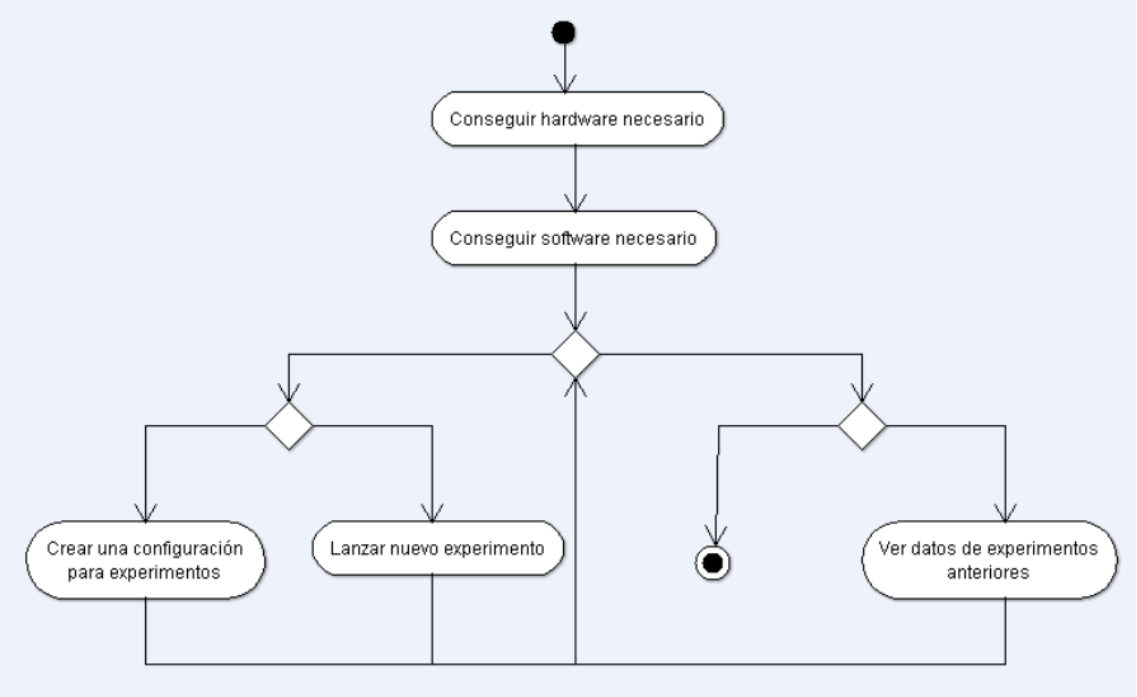


Figura 2.25. Diagrama de actividades de la ejecución del programa

En el diagrama anterior, se observa que los primeros pasos es conseguir el hardware necesario para poder ejecutar el programa (un PC y/o un joystick) y conseguir el software necesario (instalador de la aplicación y la clave). Una vez que ya se tienen ambas cosas, se puede ejecutar la aplicación. Desde ella se puede crear configuraciones para los experimentos, ejecutar nuevos experimentos o ver datos de los experimentos anteriores.

En el siguiente diagrama, Figura 2.26, se observan las acciones necesarias para crear una configuración para los experimentos, ya sean de memoria espacial o de memoria de trabajo.

En el caso de memoria espacial, se ha de introducir el nombre de la configuración, seleccionar los estímulos que aparecerán en la habitación y las zonas premiadas.

En el caso de memoria de trabajo, se ha de introducir el nombre de la configuración y el número de bloques de que constará el experimento, los estímulos que aparecerán en la habitación y entonces se pasará a configurar los bloques individualmente. Este último paso se explica en el próximo diagrama.

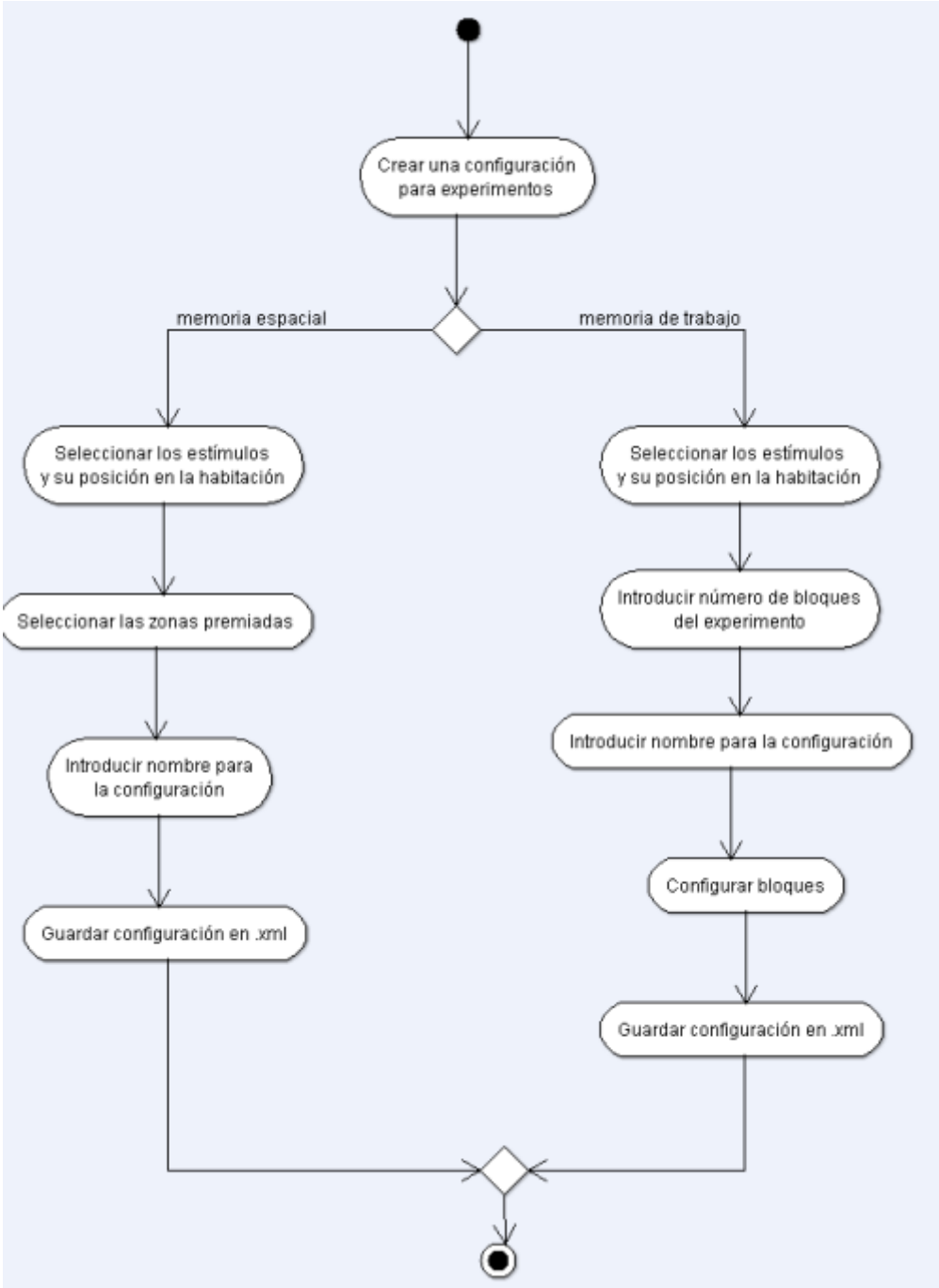


Figura 2.26. Diagrama de actividades para configurar un experimento.

El siguiente diagrama, Figura 2.27, muestra las acciones a llevar a cabo para configurar un bloque de un experimento de memoria de trabajo. Se han de introducir las posiciones de salida de

los 2 ensayos de que consta cada bloque, así como la duración de cada ensayo y el tiempo entre cada ensayo. También se elegirán las zonas premiadas.

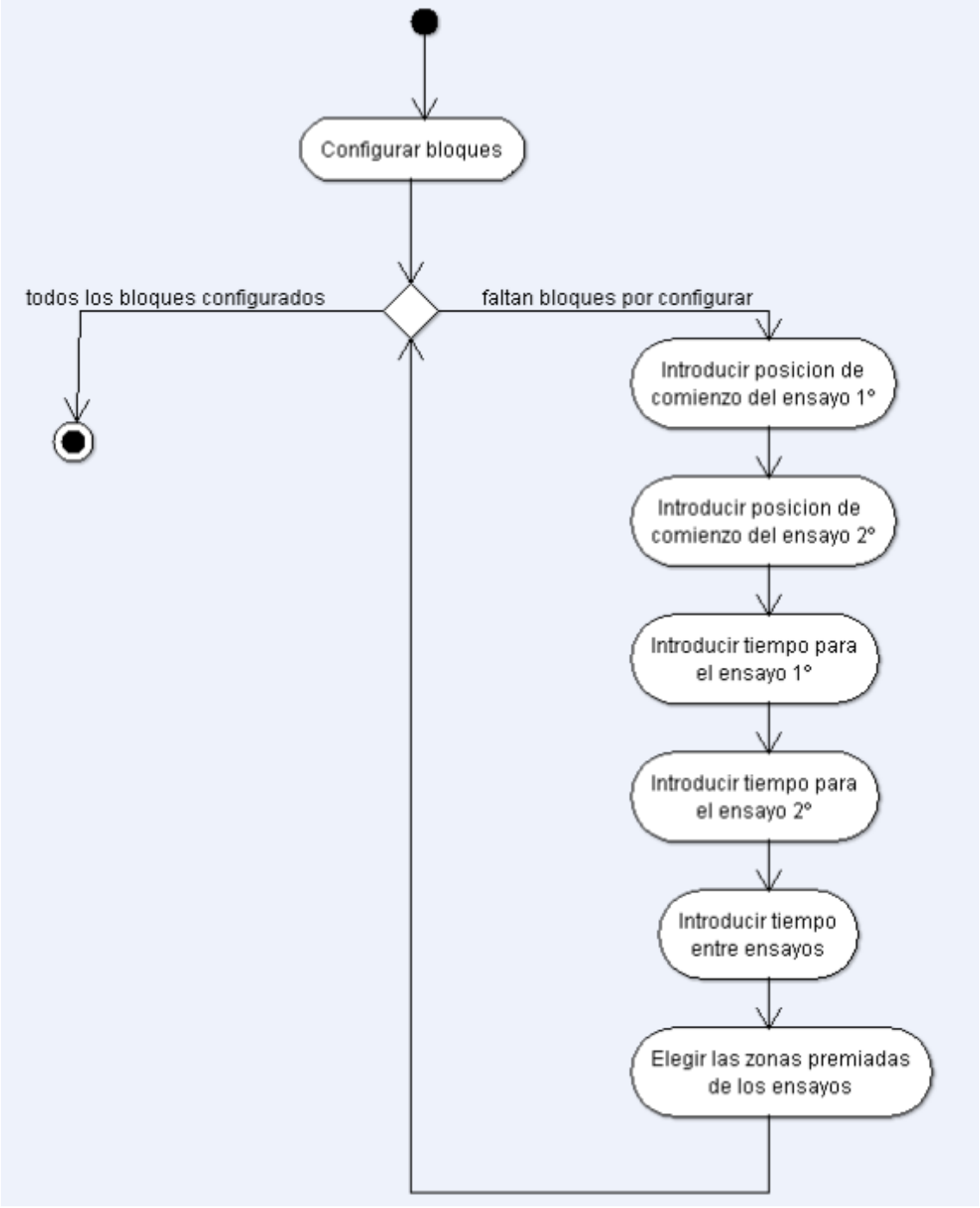


Figura 2.27. Diagrama de actividades para configurar un bloque.

Para lanzar un nuevo experimento de memoria espacial o de memoria de trabajo, se han de realizar las siguientes actividades, recogidas en la Figura 2.28.

Si es un experimento de memoria espacial, se introducirá el nombre del sujeto, el número de ensayos de que constará el experimento, la duración de los mismos, se elegirá el tamaño de las zonas premiadas, la velocidad de desplazamiento, el modo de video, el archivo de configuración de memoria espacial y se lanzará la aplicación EVEMEHME.

Si es un experimento de memoria de trabajo, se introducirá el nombre del sujeto, el tiempo entre los bloques del experimento, se elegirá el tamaño de las zonas premiadas, la velocidad de desplazamiento, el modo de video, el archivo de configuración de memoria de trabajo y se lanzará la aplicación EVEMEHMT.

Figura 2.28. Diagrama de actividades para lanzar un nuevo experimento.

Si se opta por lanzar un experimento de memoria espacial, EVEMEHME, las actividades a realizar son las que muestra el diagrama de la Figura 2.29. Se observa que solo se puede desplazar

hacia adelante o rotar la cámara. Los ensayos acabarán cuando se alcance el tiempo que se introdujo o cuando se encuentren todas las zonas premiadas. El programa acabará cuando se ejecuten todos los ensayos.

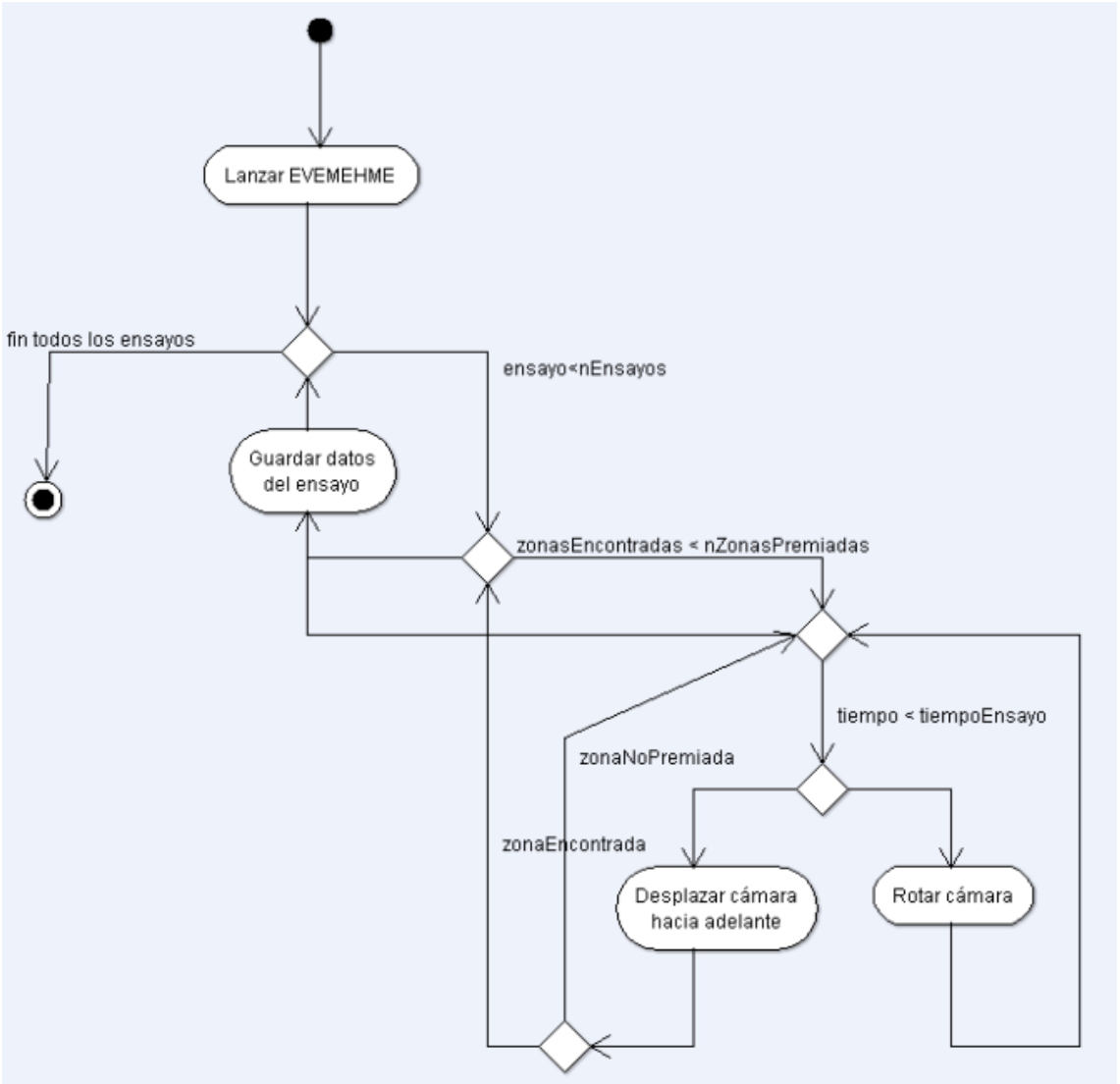


Figura 2.29. Diagrama de actividades de la aplicación EVEMEHME

Si, en cambio, se opta por lanzar un experimento de memoria de trabajo, EVEMEHMT, las actividades a realizar son las que muestra el diagrama de la Figura 2.30. Se observa que solo se puede desplazar hacia adelante o rotar la cámara. Los ensayos acabarán cuando se alcance el tiempo que se introdujo o cuando se encuentren todas las zonas premiadas. Los bloques acaban cuando se ejecutan los 2 ensayos de que consta cada bloque. El programa acabará cuando se ejecuten todos los bloques.

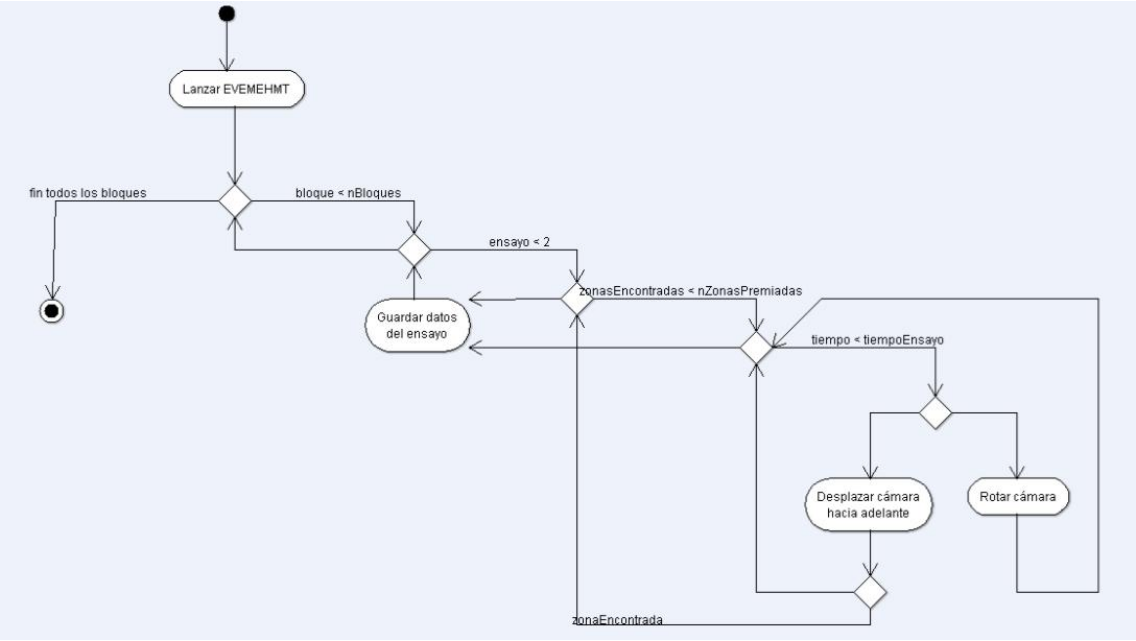


Figura 2.30. Diagrama de actividades de la aplicación EVEMEHMT

Si se desea ver los datos de experimentos anteriores, ya sean de memoria espacial o de memoria de trabajo, se ha de elegir el experimento a ver. Mientras se ven los datos del experimento, se puede salir o se puede ver la trayectoria del ensayo. La trayectoria muestra la vista en planta de la habitación con las zonas premiadas que haya encontrado el sujeto durante el ensayo. Esta vista se puede desplazar o rotar.

Se puede ver esto de forma gráfica en el diagrama de la Figura 2.31.

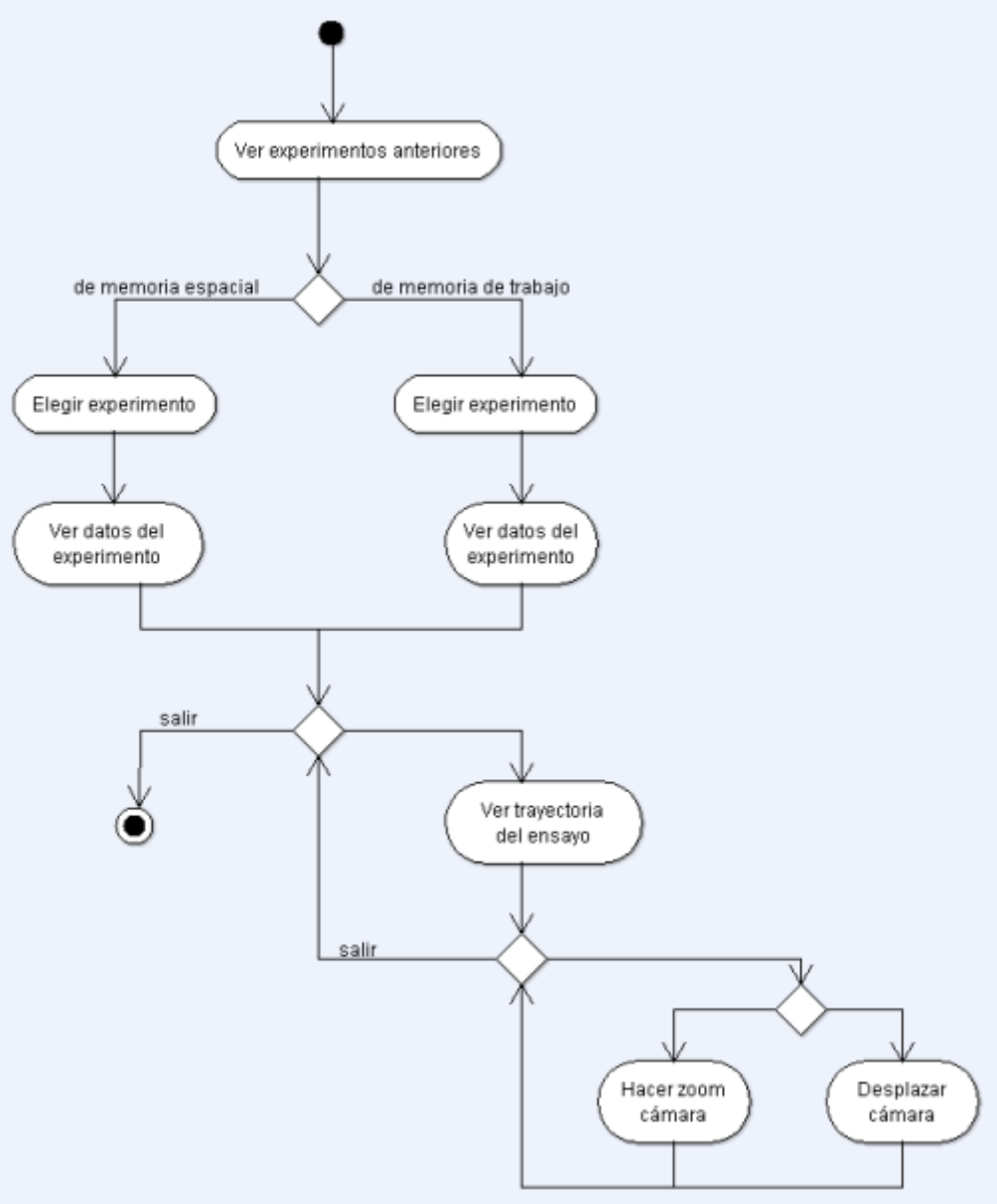
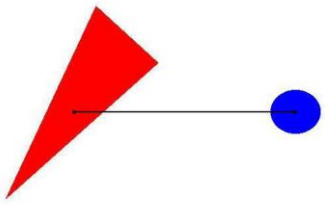


Figura 2.31. Diagrama de actividades para ver un experimento anterior.

2.4 Descripción de algoritmos específicos

2.4.1 Colisión esfera-triángulo



Aunque existen muchos algoritmos de detección de colisiones, se ha elegido un algoritmo que detecta la colisión entre una esfera 'imaginaria' y un polígono triangular. Las razones para la elección de este algoritmo son:

a) Es ideal para comprobar si un personaje (vista en tercera persona) o una cámara (vista en primera persona, como es mi caso) colisiona con los elementos de un mundo.

b) Es uno de los algoritmos de detección de colisiones más rápidos que existen, ya que existen otros que usan por ejemplo un cilindro (como el algoritmo MDK2), elipses o cubos, y se obtiene un menor rendimiento a lo largo de la animación.

Por lo tanto este algoritmo, aunque sencillo, es muy eficiente. Los pasos que sigue son los siguientes:

1) Colisión esfera-plano infinito. En primer lugar se comprueba si la esfera colisiona con el plano del triángulo. Hay que tener en cuenta un detalle: aunque el polígono triangular es finito, el plano del mismo es infinito. Si la esfera no colisiona con dicho plano infinito, no lo hará con el polígono (que es una pequeña parte de dicho plano), luego el algoritmo finalizará ofreciendo el resultado de 'no colisión'. Si la esfera sí colisiona con dicho plano, entonces tendremos que ir al siguiente paso.

2) Punto de proyección esfera-plano infinito. Una vez se conoce que la esfera colisiona con el plano falta saber si colisiona con el triángulo. Para ello el siguiente paso consiste en obtener el punto de intersección de la esfera con el plano infinito. Para ello se necesita el vector normal al triángulo y la distancia que hay desde el centro de la esfera al plano infinito. Multiplicando el vector por la distancia se obtiene un desplazamiento. Restando este desplazamiento al centro de la esfera se obtiene la posición del centro de la esfera dentro del plano infinito en la dirección de su normal.

3) Colisión esfera-triángulo. En el último paso, se calculan 3 ángulos, los formados por los 3 vectores que parten del punto de proyección, obtenido en el paso anterior, hacia los vértices del triangulo. Finalmente se suman todos los ángulos.

Si la suma de los 3 ángulos es 360º el punto de proyección calculado en el paso 2 realmente pertenece al triángulo, y si es menor, dicho punto estará en el plano infinito del triángulo pero no en el polígono en sí.

2.4.2. Decisiones a la hora de tratar las colisiones

En las primeras versiones del programa, a la hora de dibujar la escena, se tenían en cuenta las colisiones para todos los objetos de los modelos. Esto suponía una carga de trabajo que, si bien no influía significativamente en el rendimiento del programa, si presentaba a veces cierta ralentización cuando la habitación estaba completa de estímulos.

Además, por la forma de los modelos que conforman los estímulos, a veces el sujeto podía quedar atrapado. Uno de los requerimientos impuestos por José Manuel, era este mismo, que no hubiera posibilidad de quedar atrapado dentro de un objeto. Por lo que había que buscar una solución.

Se optó por crear un cilindro alrededor de la pared de la habitación, que sería con el que se colisionaría y así no habría posibilidad de salirse de la habitación. Además, el cilindro permite desplazarse por la habitación y encontrar las zonas sin posibilidad de quedarse atrapado dentro de algún objeto, puesto que los modelos de los estímulos quedan situados entre este cilindro y la pared y no hay forma de colisionar con ellos. Esta técnica se utiliza sobre todo en videojuegos, donde los modelos muy complejos se rodean con una malla más simple, de forma que se simplifique la colisión con ellos.

Una vez tomada esa decisión, ya no era necesario comprobar las colisiones a la hora de dibujar los modelos que forman los estímulos. Esto provocó un aumento en el rendimiento y fluidez de la aplicación.

Otro punto a tener en cuenta en las primeras versiones eran las colisiones con las zonas premiadas. En principio, se tenían 3 cilindros concéntricos con distinto radio de la base, para los tamaños pequeño, mediano y grande. Cuando la esfera que rodea a la cámara colisionaba con estos cilindros se comprobaba si estaban premiados o no.

El problema era que de esta forma se podía colisionar con los cilindros sin llegar a estar dentro de ellos, ya que el radio de la esfera de la cámara, colisiona antes de que la cámara esté dentro del cilindro.

Para solucionar este problema, se optó por colocar un cilindro dentro de las zonas premiadas. En función del tamaño de las zonas, se varía el radio de la esfera, de forma que cuando la esfera colisione con este cilindro, la distancia ente la esfera y el cilindro sea menor que la del radio de la base de los cilindros de las zonas premiadas.

Capítulo 3

Pruebas y experimentación

Resumen

En este tercer capítulo, se incluyen algunas pruebas realizadas tras la implementación de la aplicación.

1) Se mostrará un ejemplo práctico de un experimento de memoria espacial. Primero se configurará el experimento, luego se ejecutará y, por último, se verán los resultados obtenidos.

2) Se mostrará un ejemplo práctico de un experimento de memoria de trabajo. Primero se configurará el experimento, luego se ejecutará y, por último, se verán los resultados obtenidos.

3.1. Ejemplo 1: Configuración, ejecución y vista de resultados de un experimento de memoria espacial

Al ejecutar el programa, tras la ventana de bienvenida, Figura 3.1, aparecerá la ventana principal de la aplicación, Figura 3.2 y Anexo 1.6.

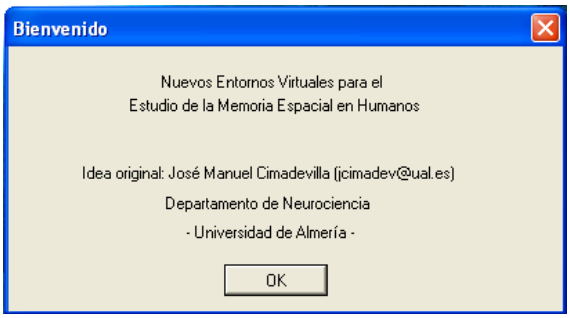


Figura 3.1. Captura de la ventana de Bienvenida.

Figura 3.2. Captura de la ventana principal.

La ventana principal consta de un menú situado en la parte superior izquierda de la pantalla, como muestra la Figura 3.3.

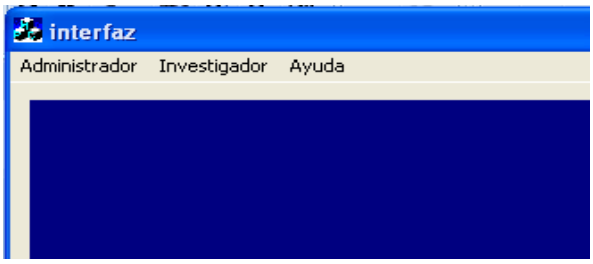


Figura 3.3. Detalle del menú principal.

Para configurar los experimentos para evaluar la memoria espacial, se ha de seleccionar en el menú *"Administrador"*, la opción *"Configurar Experimento. Memoria Espacial"*, como ilustra la Figura 3.4.

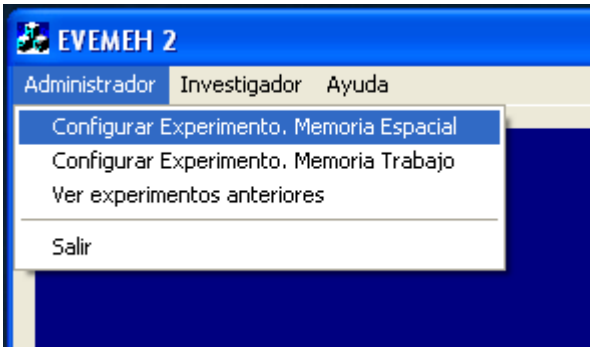


Figura 3.4. Detalle del menú de administrador.

Aparecerá entonces una ventana como la de la Figura 3.5 o Anexo 1.1:

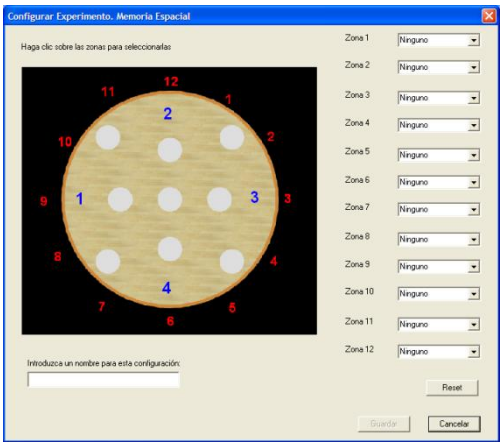


Figura 3.5. Ventana de configuración de experimentos para memoria espacial.

Tomando la numeración de las zonas, se recoge en la Figura 3.6:

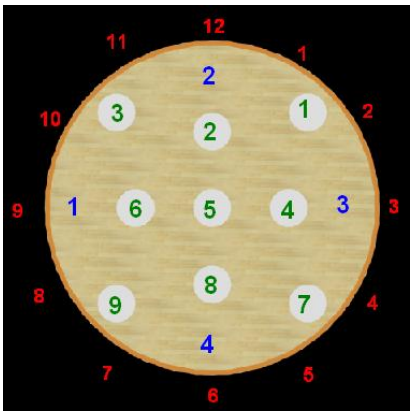


Figura 3.6. Numeración de las zonas.

Seleccionamos las zonas 1, 2 y 6. Como estímulos, se seleccionarán una chimenea a las 3, un cuadro a las 6, una losa a las 9 y una lámpara a las 12. En el cuadro de texto del nombre se introduce "prueba". Si pulsamos el botón *"Guardar"*, se guardará la configuración.

Las configuraciones creadas se almacenan en la carpeta *"C:\EVEMEH2_9\Datos\Configuración\Espacial\"*. En esta carpeta se almacenan las configuraciones como un archivo en XML.

El archivo XML correspondiente a la configuración que se ha introducido es el siguiente, relfejado en la Figura 3.7:

```
- <experimentoME>
  - <estimulos>
    - <estimulo>
        <zonaH>3</zonaH>
        <objeto>chimenea</objeto>
      </estimulo>
    - <estimulo>
        <zonaH>6</zonaH>
        <objeto>cuadro1</objeto>
      </estimulo>
    - <estimulo>
        <zonaH>9</zonaH>
        <objeto>losa</objeto>
      </estimulo>
    - <estimulo>
        <zonaH>12</zonaH>
        <objeto>lampara</objeto>
      </estimulo>
    </estimulos>
  - <zonasPremiadas>
      <zonaPremiada>1</zonaPremiada>
      <zonaPremiada>2</zonaPremiada>
      <zonaPremiada>6</zonaPremiada>
    </zonasPremiadas>
  </experimentoME>
```

Figura 3.7. Archivo de configuración de memoria espacial.

Para ejecutar los experimentos para evaluar la memoria espacial, se ha de seleccionar dentro del menú *"Investigador"*, la opción *"Nuevo Experimento. Memoria Espacial"*. La Figura 3.8 muestra el detalle de este menú.

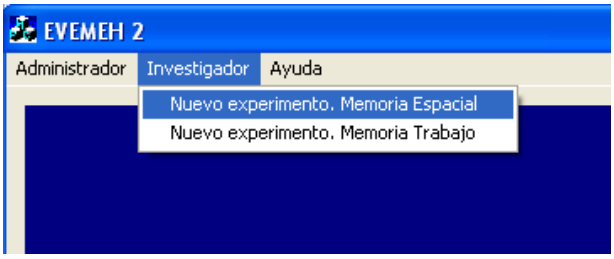


Figura 3.8. Detalle del menú investigador.

Al seleccionarlo, aparecerá una ventana como la de la Figura 3.9 o Anexo 1.7:

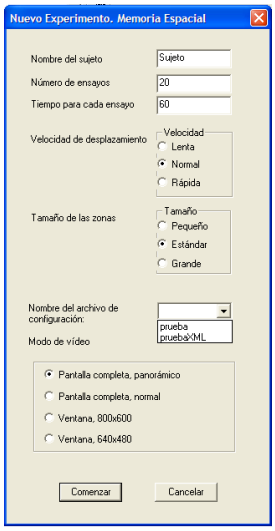


Figura 3.9. Captura de la ventana para lanzar nuevos experimentos de memoria espacial.

Se introduce entonces la siguiente información:

Nombre del sujeto: Armando

Numero de ensayos: 5

Tiempo para cada ensayo: 60

Velocidad de desplazamiento: Normal

Tamaño de las zonas: Estándar

Nombre del archivo de configuración: prueba

Modo de video: pantalla completa, panorámico.

Con el botón "*Comenzar*" se lanza la aplicación 3D. En las siguientes figuras, se muestran en detalle los estímulos seleccionados durante la configuración del experimento. Para más detalle, observar los anexos 1.9 en adelante.

Figura 3.10. Detalle de la chimenea.

Figura 3.11. Detalle de la losa.

Figura 3.12. Detalle de la lámpara.

Figura 3.13. Detalle del cuadro.

Una vez lanzado el experimento, se ha de recorrer la habitación en busca de las zonas premiadas. Si hayamos una zona premiada, aparecerá un mensaje indicando que hemos encontrado la zona y las zonas premiadas que aún faltan por encontrar. Un ejemplo de mensaje se ve en la Figura 3.14, o Anexo 1.13:

Figura 3.14. Detalle de un mensaje de zona encontrada.

Además, se indicará al sujeto del experimento la posición de la zona premiada, mostrando un círculo de color verde en el suelo situado sobre la zona premiada y según el tamaño de la zona. La Figura 3.15 muestra ese círculo. Para más detalle, ver Anexo 1.14.

Figura 3.15. Captura de la ejecución del programa.

Cuando se encuentren todas las zonas premiadas de la habitación, se avisará al usuario mostrando un mensaje y una alerta sonora, y se pasará al siguiente ensayo. La Figura 3.16 muestra el mensaje. Para más detalle, ver Anexo 1.15.

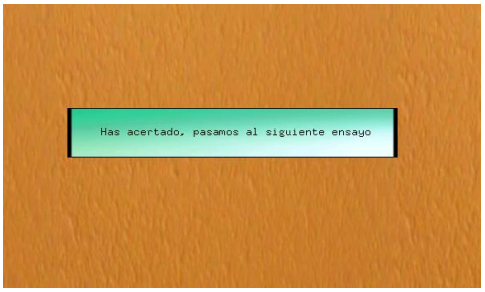


Figura 3.16. Detalle de un mensaje de acierto y fin de ensayo.

Cuando terminen todos los ensayos, se podrán ver los resultados de los experimentos. Para ello, dentro del menú Administrador, se selecciona la opción Ver experimentos anteriores. La Figura 3.17 muestra el detalle del menú.

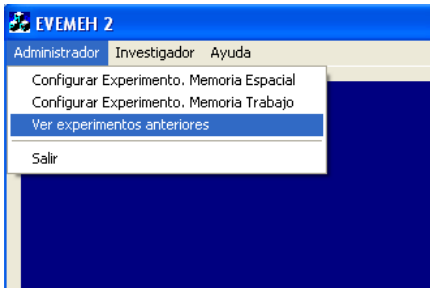


Figura 3.17. Captura del menú administrador

Aparecerá entonces una ventana como la siguiente, en la que se puede elegir el experimento que queremos ver seleccionándolo del menú desplegable, y haciendo clic en Ver. La Figura 3.18 y el Anexo 1.5 muestran el detalle de la ventana.

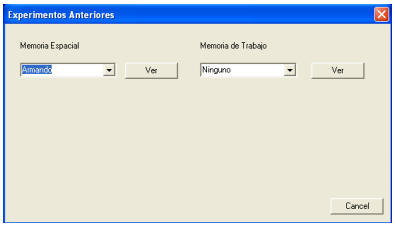


Figura 3.18. Captura de la ventana de ver experimentos anteriores.

Se abrirá entonces una página HTML, Figura 3.19, en la que se muestra la información referente a la configuración del ensayo en la parte superior y el centro el detalle de los ensayos:

"NUEVOS ENTORNOS VIRTUALES PARA LA EVALUACION DE MEMORIA ESPACIAL EN HUMANOS"

- Nombre del sujeto: Armando
- Número de ensayos: 7
- Tiempo para cada ensayo: 60
- Velocidad de desplazamiento: 2
- Número de zonas premiadas: 3
- Zonas premiadas: 1 2 6

Nº ensayo	*Tiempo total*	*Distancia recorrida*	*Velocidad media*	*Zonas Encontradas*	*Nº aciertos*	*Trayectoria*
1	22 s	17.190601 m	0.781391 m/s	2 5 1 6	3	Ver
2	11 s	15.051401 m	1.368309 m/s	6 5 2 1	3	Ver
3	14 s	17.197601 m	1.228400 m/s	4 5 6 2 1	3	Ver
4	19 s	19.577601 m	1.030400 m/s	8 5 2 1 6	3	Ver
5	16 s	17.827601 m	1.114225 m/s	4 5 6 2 1	3	Ver
6	11 s	7.781200 m	0.707382 m/s	6 5 2 1	3	Ver

Figura 3.19. Captura de la página html con los datos de los experimentos.

Para ver las trayectorias de cada uno de los ensayos, basta con hacer clic en "*Ver*" que ejecutará el programa *Trayectoria1.exe* para visualizar los pasos seguidos en ese ensayo. Aparecerá una ventana preguntándonos si queremos ejecutar o guardar el enlace. En este caso, pulsar sobre la opción 'Ejecutar', como en la Figura 3.20.

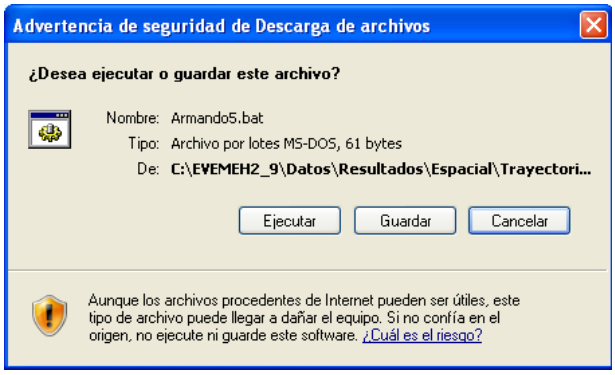


Figura 3.20. Ventana de confirmación de ejecución de trayectoria.

A continuación aparece una vista en planta de la habitación, con las zonas premiadas de color verde, y la trayectoria del sujeto, como en la Figura 3.21 o Anexo 16. Además se permite mucha interacción a través de teclado:

- Teclas de dirección (arriba, abajo, derecha e izquierda): podemos desplazar la cámara.
- Teclas 'q' y 'w': permiten hacer zoom (acercar y alejar).
- ENTER: permite ver la trayectoria del usuario 'paso a paso'.

Para finalizar la ejecución, pulsar ESC.

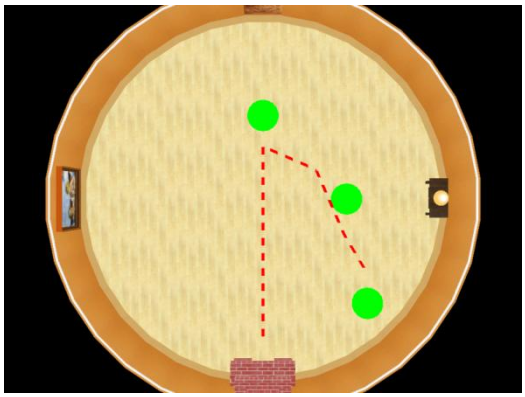


Figura 3.21. Captura de la vista de una trayectoria

3.2 Ejemplo 2: Configuración, ejecución y Configuración, ejecución y vista de resultados de un experimento de memoria de trabajo

Al ejecutar el programa, tras la ventana de bienvenida, Figura 3.22, aparecerá la ventana principal de la aplicación, Figura 3.23 o Anexo 1.6.

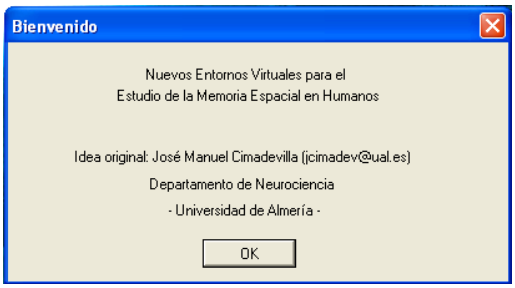


Figura 3.22. Captura de la ventana de Bienvenida.

Figura 3.23. Captura de la ventana principal.

Para configurar los experimentos para evaluar la memoria de trabajo, hemos de seleccionar en el menú *"Administrador"*, la opción *"Configurar Experimento. Memoria Trabajo"*. Se puede ver en la Figura 3.24.

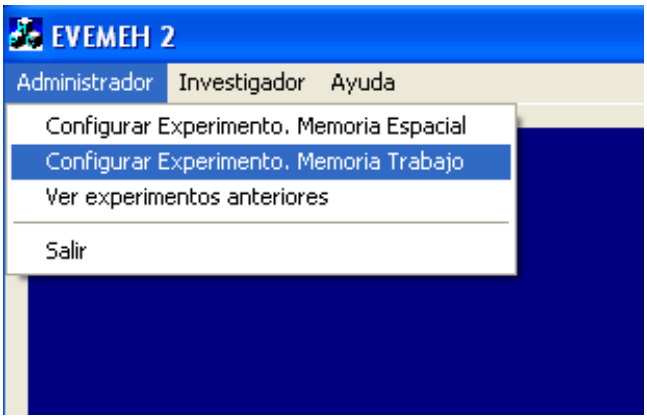


Figura 3.24. Detalle del menú administrador.

Aparecerá entonces una ventana como la de la Figura 3.25 o Anexo 1.8:

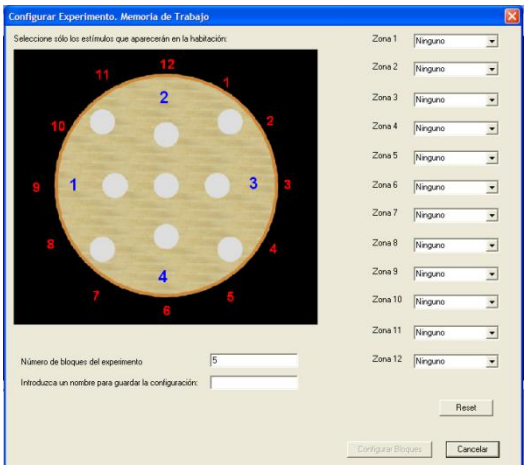


Figura 3.25. Ventana de configuración de experimentos para memoria de trabajo

Se seleccionarán como estímulos una chimenea a las 3 y a las 9, y una lámpara a las 6 y a las 12. En número de bloques se introducen "2" y en nombre del archivo se pondrá "pruebaTrabajo".

Si pulsamos el botón *"Configurar Bloques"*, pasaremos a la ventana de configuración de los bloques, Figura 3.26 o Anexo 1.2. Se mostrará 2 veces, para configurar los 2 bloques que se han introducido anteriormente.

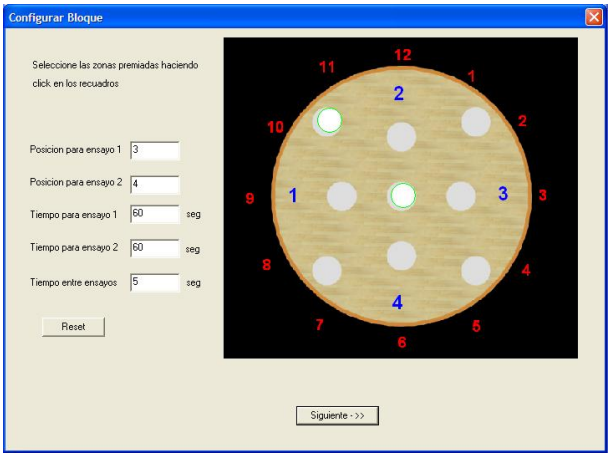


Figura 3.26. Ventana de configuración de bloques.

Se introducirá la siguiente información en las ventanas:

Bloque 1:	Bloque 2:
Posicion para ensayo 1: 1	Posicion para ensayo 1: 3
Posicion para ensayo 2: 2	Posicion para ensayo 2: 4
Tiempo para ensayo 1: 60	Tiempo para ensayo 1: 60
Tiempo para ensayo 2: 60	Tiempo para ensayo 2: 60
Tiempo entre ensayos: 5	Tiempo entre ensayos: 5
Zonas premiadas: 2, 6	Zonas premiadas: 3, 5

Las configuraciones creadas se almacenan en la carpeta *"C:\EVEMEH2_9.\Datos\Configuración\Trabajo\"*. En esta carpeta se almacenan las configuraciones como un archivo en xml.

El archivo correspondiente a la configuración creada es *pruebaTrabajo.xml*, mostrado en la Figura 3.27 o en el Anexo 1.3:

```
- <experimentoMT>
  - <estimulos>
    - <estimulo>
        <zonaH>3</zonaH>
        <objeto>chimenea</objeto>
      </estimulo>
    - <estimulo>
        <zonaH>6</zonaH>
        <objeto>lampara</objeto>
      </estimulo>
    - <estimulo>
        <zonaH>9</zonaH>
        <objeto>chimenea</objeto>
      </estimulo>
    - <estimulo>
        <zonaH>12</zonaH>
        <objeto>lampara</objeto>
      </estimulo>
    </estimulos>
  - <bloques>
    - <bloque>
        <posicion1>1</posicion1>
        <posicion2>2</posicion2>
        <tiempo1>60</tiempo1>
        <tiempo2>60</tiempo2>
      - <zonasPremiadas>
          <zonaPremiada>2</zonaPremiada>
          <zonaPremiada>6</zonaPremiada>
        </zonasPremiadas>
        <tiempoEntreEnsayos>5</tiempoEntreEnsayos>
      </bloque>
    - <bloque>
        <posicion1>3</posicion1>
        <posicion2>4</posicion2>
        <tiempo1>60</tiempo1>
        <tiempo2>60</tiempo2>
      - <zonasPremiadas>
          <zonaPremiada>3</zonaPremiada>
          <zonaPremiada>5</zonaPremiada>
        </zonasPremiadas>
        <tiempoEntreEnsayos>5</tiempoEntreEnsayos>
      </bloque>
    </bloques>
</experimentoMT>
```

Figura 3.27. Archivo de configuración de experimentos de memoria de trabajo.

Para ejecutar los experimentos para evaluar la memoria de trabajo, se ha de seleccionar dentro del menú *"Investigador"*, la opción *"Nuevo Experimento. Memoria Trabajo"*.

Al seleccionarlo, aparecerá una ventana como la de la Figura 3.28 o Anexo 1.8:

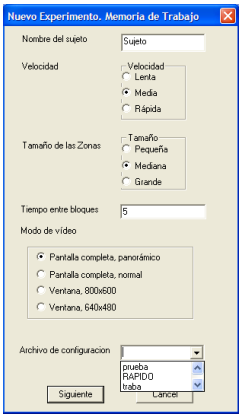


Figura 3.28. Ventana para lanzar experimentos de memoria de trabajo.

Se introduce entonces la siguiente información:

Nombre del sujeto: Armando

Numero de ensayos: 5

Tiempo para cada ensayo: 60

Velocidad de desplazamiento: Normal

Tamaño de las zonas: Estándar

Nombre del archivo de configuración: prueba

Modo de video: pantalla completa, panorámico.

Si se hace clic en *"Comenzar"*, empezará la ejecución del experimento. Si se ha producido algún error a la hora de configurar el experimento, aparecerá una ventana de notificación y no empezará la ejecución.

La Figura 3.29 o Anexo 1.17 muestra una captura de la ejecución del juego:

Figura 3.29. Captura de la ejecución del juego.

Al comienzo de cada bloque, aparece un mensaje informativo indicando el número de bloque correspondiente. Puede observarse en la Figura 3.30 o Anexo 1.18

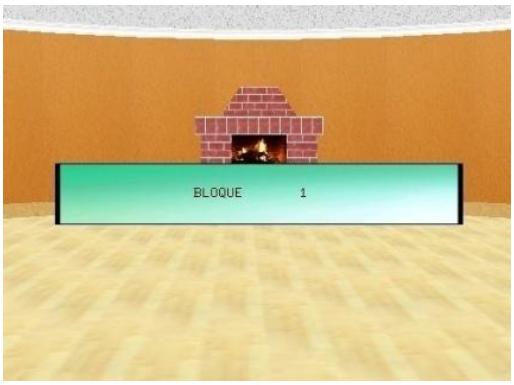


Figura 3.30. Mensaje informativo de comienzo de bloque

Cuando se encuentran todas las zonas premiadas, se mostrará un mensaje indicando que el primer ensayo ha terminado, Figura 3.31 o Anexo 1.19, y que se pasa al siguiente ensayo del bloque. Si no se encontraran las zonas, no se muestra ningún mensaje, y se pasa al siguiente ensayo directamente.

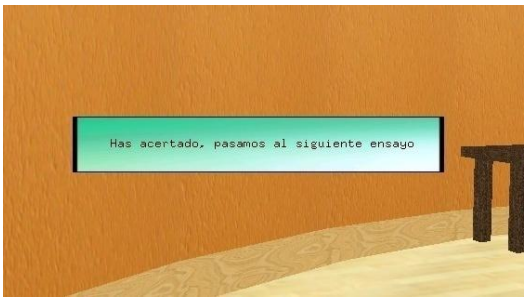


Figura 3.31. Mensaje informativo de ensayo acabado.

Si se encuentran todas las zonas del segundo ensayo del bloque, se mostrará un mensaje indicando que se ha terminado el ensayo, Figura 3.32 o Anexo 1.20, y que se pasa al siguiente bloque.

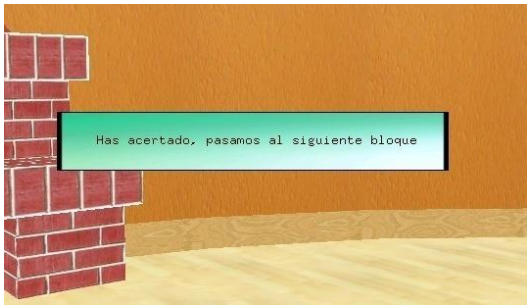


Figura 3.32. Mensaje informativo paso al siguiente bloque.

Cuando terminen todos los ensayos, se podrán ver los resultados de los experimentos. Para ello, dentro del menú Administrador, se selecciona la opción Ver experimentos anteriores, como muestra la Figura 3.33.

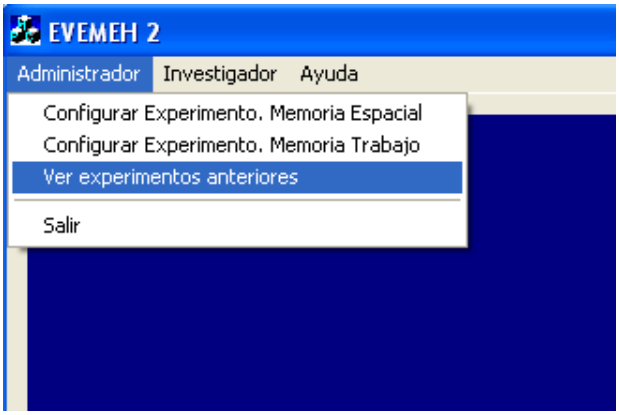
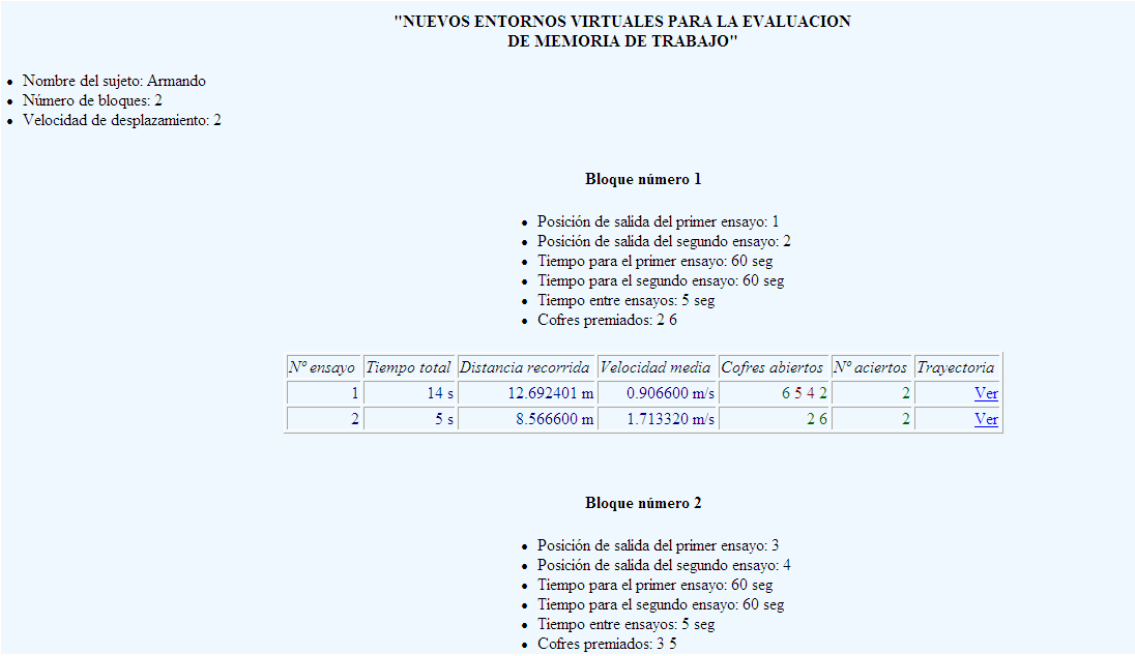


Figura 3.33. Captura del menú de administrador.

Aparecerá entonces una ventana como la siguiente, Figura 3.34 o Anexo 1.5, en la que podremos elegir el experimento que queremos ver seleccionándolo del menú desplegable, y haciendo clic en Ver.

Figura 3.34. Captura de la ventana ver experimentos anteriores.

Se abrirá entonces una página HTML, en la que se muestra la información referente a la configuración del ensayo en la parte superior y una tabla en el centro, con el detalle de los ensayos, como ilustra la Figura 3.35:



"NUEVOS ENTORNOS VIRTUALES PARA LA EVALUACION DE MEMORIA DE TRABAJO"

- Nombre del sujeto: Armando
- Número de bloques: 2
- Velocidad de desplazamiento: 2

Bloque número 1

- Posición de salida del primer ensayo: 1
- Posición de salida del segundo ensayo: 2
- Tiempo para el primer ensayo: 60 seg
- Tiempo para el segundo ensayo: 60 seg
- Tiempo entre ensayos: 5 seg
- Cofres premiados: 2 6

Nº ensayo	*Tiempo total*	*Distancia recorrida*	*Velocidad media*	*Cofres abiertos*	*Nº aciertos*	*Trayectoria*
1	14 s	12.692401 m	0.906600 m/s	6 5 4 2	2	Ver
2	5 s	8.566600 m	1.713320 m/s	2 6	2	Ver

Bloque número 2

- Posición de salida del primer ensayo: 3
- Posición de salida del segundo ensayo: 4
- Tiempo para el primer ensayo: 60 seg
- Tiempo para el segundo ensayo: 60 seg
- Tiempo entre ensayos: 5 seg
- Cofres premiados: 3 5

Figura 3.35. Captura de la página html con los datos de los ensayos

Para ver las trayectorias de cada uno de los ensayos, basta con hacer clic en Ver, que ejecutará el programa *Trayectoria2.exe* para visualizar los pasos seguidos en ese ensayo, dependiendo de si se trata de experimentos de memoria espacial o de trabajo. Aparecerá una ventana preguntándonos si queremos ejecutar o guardar el enlace. En este caso, pulsar sobre la opción '*Ejecutar*'. La siguiente figura, Figura 3.36, muestra la ventana que aparecerá:

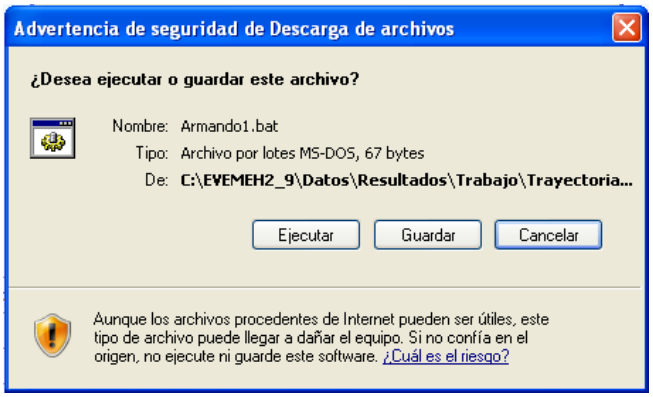


Figura 3.36. Captura de la ventana de petición de ejecución.

A continuación aparece una vista en planta de la habitación, Figura 3.37 o Anexo 1.21, con las zonas premiadas de color verde, y la trayectoria del sujeto. Además se permite mucha interacción a través de teclado:

- Teclas de dirección (arriba, abajo, derecha e izquierda): podemos desplazar la cámara.

- Teclas ‘q’ y ‘w’: permiten hacer zoom (acercar y alejar).
- ENTER: permite ver la trayectoria del usuario ‘paso a paso’.

Para finalizar la ejecución, pulsar ESC.

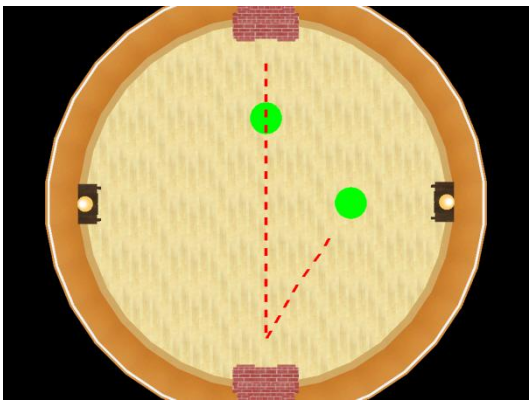


Figura 3.37. Captura de la vista de una trayectoria

Capítulo 4

Conclusiones y líneas de trabajo abiertas

Resumen

En este cuarto capítulo, reflexiono sobre las conclusiones y valoraciones a las que se llega al acabar el proyecto. Explicaré las ventajas y desventajas del trabajo realizado y la utilidad de éste, así como discutiré las tareas que han quedado sin resolver y posibles extensiones. Además, se dan posibles soluciones para estas líneas abiertas planteadas.

El objetivo de mi proyecto era crear una aplicación que permitiera estudiar la memoria espacial y de trabajo en humanos haciendo uso de entornos virtuales creados por ordenador usando 3DS Max.

Como los usuarios del sistema no van a ser expertos en informática, he intentado crear un entorno sencillo de utilizar y que fuera eficiente. A pesar de la complejidad a la hora de representar los modelos en 3D y de toda la complejidad del control del entorno 3D, considero que el programa es suficientemente sencillo como para que lo pueda usar cualquier persona.

Se facilita la creación de la configuración para los experimentos y su posterior uso en los mismos mediante menús desplegables. También se facilita añadir nuevos estímulos que después puedan ser mostrados. Para ello basta con diseñarlos en base a unas restricciones sencillas. Y se intenta mostrar la información relativa a los experimentos en forma de tabla que los resuma y muestre de una forma clara.

En la elaboración de la documentación, uno se da cuenta de la necesidad de explicar correctamente y detalladamente lo que realiza antes, durante y después de la programación de una aplicación. Gracias al uso de técnicas de estimación, puede saberse de antemano cuanto se tardará en tener completo una aplicación, así como gracias a los diagramas UML, se puede entender el funcionamiento del programa.

La elección de OpenGL vienen dada porque es una API que ofrece una completa funcionalidad y en concreto, la necesaria para la realización de este proyecto. Además, esta API se usa en la asignatura Ampliación de Informática Gráfica, la cual cursé, de forma que me era más familiar que Direct3D. Además, al haberla usado en la realización de las prácticas, pude comprobar que se ajustaba a mis necesidades y a las del proyecto, de forma que me decanté por ella antes que por otra.

Relacionado con el motor gráfico, el tratamiento de las texturas y luces podría haber sido más cuidado. Se podrían haber añadido distintas fuentes de luz, sistemas de partículas, efectos HDR, etc. Ello daría a la aplicación un aspecto mucho más real. Sin embargo, supondría una reducción en el rendimiento que habría que tratar para poder optimizarla lo máximo posible. Recordemos que lo

que prima en el programa es la funcionalidad y no la apariencia gráfica. Además, el programa cumple con las exigencias marcadas y este mejor tratamiento de luces y texturas no consta como una de las prioridades.

Las posibles mejoras o modificaciones sobre este programa vendrán impuestas por los usuarios del mismo, ya que serán ellos los que decidan si algún aspecto no les gusta o si necesitan agregar nueva funcionalidad al programa.

Capítulo 5

Bibliografía

5.1 Referencias bibliográficas

[Gutiérrez, 2002] José Gutiérrez Maldonado (2002). Aplicaciones de la realidad virtual en psicología clínica. Disponible en http://www.ub.es/personal/jgutierrez/realidadvirtual.pdf.

[Espínola *et al.*, 2006] Moisés Espínola, Fernando Parra, Luis Iribarne (2006). Iniciación al diseño y programación de videojuegos 3D, ISBN: 84-96270-69-6.

[Espínola *et al.*,] Moisés Espínola, Antonio López Márquez, José Luis Berenguel Gómez, Luis Iribarne. Programación de animaciones en 3D - Un motor gráfico en OpenGL. ISBN: 978-84-96270-89-3.

[Botella *et al.*, 2000] Cristina Botella, Rosa María Baños, Helena Villa, Conxa Perpiñá, C, Azucena García-Palacios (2000). Virtual Reality in the treatment of claustrophobic fear: a controlled multiple baseline design.

[Etchepareborda *et al.*, 2005] M.C. Etchepareborda, L. Abad-Mas (2005). Memoria de trabajo en los procesos básicos del aprendizaje.

[Pressman *et al.*, 1997] Roger S. Pressman, Rafael Ojeda, Joaquín Sánchez, Luis Joyanes, etc. (1997). Ingeniería del software. Un enfoque práctico. Mc-Graw-Hill, páginas 24-25. ISBN:8-4481-3214-9.

[Sommerville, 2005] Ian Sommerville (2005). Ingenieria del software. Pearson Addison Wesley, páginas 92-95, ISBN 84-7829-074-5

[Luque *et al*., 1999] Irene Luque Ruiz, Miguel Ángel Gómez-Nieto (1999). Ingeniería del software: fundamentos para el desarrollo de sistemas. Servicio de publicaciones de la universidad de Córdoba, páginas 69-75, 79-81. ISBN: 84-7801-486-1

[North *et al.*, 1998] North M, North M, Coble J (1998). Virtual Reality Therapy: An Effective Treatment for Phobias. En Riva, G., Wiederhold, B.K. y Molinari, E (eds.): Virtual Environments in Clinical Psychology and Neuroscience. Amsterdam. IOS Press.

[Wald *et al*., 2000] Wald J, Taylor S (2000). Efficacy of Virtual Reality Exposure Therapy to Treat Driving Phobia. A case report. Journal of Behavior Therapy and Experimental Psychiatry.

[Carling *et al.*, 1997] Carling A, Hoffman HG y Weghorst S (1997). Virtual reality and tactile augmentation in the treatment of spider phobia: a case report. Behavior Research and Therapy.

[Goetestam *et al*., 1996] Goetestam KG, Hollup S y Graave RW (1996). Virtual reality in the treatment of spider phobia. Comunicación presentada en el Congreso Mundial de Psiquiatría de Barcelona.

[Hollup SA, 1996] Hollup SA, (1996). Virtual reality therapy in treatment of spider phobia. Tesis doctoral. Departamento de Psicología. Norwegian University of Science and Technology, Trondeheim, Norway.

[Cimadevilla *et al.*, 2008] José Manuel Cimadevilla, Rosa Cánovas, Moisés Espínola, Luís Iribarne (2008). A new virtual task to evaluate human place learning.

[Dawson et al, 2002] Dawson C.W., Martín G (2002). El proyecto fin de carrera en Ingeniería Informática: una guía para el estudiante. Pearson Educación S.A. ISBN: 84-205-3560-5

5.2 Enlaces

[LightWave] http://www.newtek.com/lightwave/

[Blender] http://www.blender.org/

[AutoCAD] http://www.autodesk.es/adsk/servlet/index?siteID=455755&id=12382581

[3DSMax] http://www.autodesk.es/adsk/servlet/index?siteID=455755&id=12341473

[OpenGL] http://www.opengl.org/

[DirectX] http://msdn.microsoft.com/es-es/directx/default(en-us).aspx

[Renderman] https://renderman.pixar.com/

[XNA] http://msdn.microsoft.com/en-us/xna/default.aspx

[VisualStudio] http://msdn.microsoft.com/en-us/library/ms950417.aspx

[COSMOS] http://csciwww.etsu.edu/cosmos/

[ArgoUML] http://argouml.tigris.org/

[MSProject] http://office.microsoft.com/es-es/project/FX100487773082.aspx?ofcresset=1

[WBSChart] http://www.criticaltools.com/wbsmain.htm

[Eclipse] http://www.eclipse.org/

[LevelHead] http://selectparks.net/~julian/levelhead/

Capítulo 6

Glosario de términos

0-9

3D Studio Max: programa de creación de gráficos y animación 3D desarrollado por Autodesk Media & Entertainment. Utilizado en mayor medida por los desarrolladores de videojuegos, aunque también en el desarrollo de proyectos de animación como películas o anuncios de televisión, efectos especiales y en arquitectura.

.3DS: Formato de exportación de 3D Studio Max. Es el formato usado en este proyecto para almacenar los modelos 3D, los cuales pueden ser leídos y representados gráficamente por el motor gráfico.

A

Alpha Blending: es una técnica que permite crear objetos transparentes de forma que cualquier objeto situado detrás de él sea visible respetando su opacidad. Consiste en combinar dos objetos en pantalla teniendo en cuenta los valores alfa que designan su grado de transferencia, posibilitando que el color del píxel del objeto mostrado en segundo plano se vea influenciado por el grado de transparencia del píxel correspondiente situado en primer plano.

API: Una interfaz de programación de aplicaciones o API es el conjunto de funciones y métodos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

ArgoUML: aplicación para diseñar diagramas en UML (Unified Modeling Language) escrita en Java y publicada bajo la Licencia BSD Open Source.

B

Blender: software para modelado, texturizado, animación, render y videojuegos. Blender soporta curvas, mallas poligonales, texto, NURBS y metaballs.

Bump mapping: es una técnica de gráficos 3D consistente en dar un aspecto rugoso a las superficies de los objetos, empleada para dar un efecto de relieve en las superficies de los objetos planos.

C

COSMOS: herramienta de estimación y análisis de proyectos software, que proporciona a los desarrolladores una idea del tamaño, esfuerzo y planificación de su proyecto de software. Combina los puntos de función bien conocidos y los modelos COCOMO, así como un modelo Rayleigh de acumulación de personal propuesto por Lawrence Putnam.

D

.dwg: Extensión de archivo electrónico de dibujo computarizado, utilizado principalmente por el programa AutoCAD de Autodesk.

.dxf: tipo de archivo nativo de AutoCAD.

Direct3D: Direct3D es parte de DirectX, una API propiedad de Microsoft disponible tanto en los sistemas Windows de 32 y 64 bits, como para sus consolas Xbox y Xbox 360 para la programación de gráficos 3D. Es la clara competidora de OpenGL. El objetivo de esta API es facilitar el manejo y trazado de entidades gráficas elementales, como líneas, polígonos y texturas, en cualquier aplicación que muestre gráficos en 3D, así como efectuar de forma transparente transformaciones geométricas sobre dichas entidades.

E

Eclipse: entorno de desarrollo integrado de código abierto multiplataforma. Mediante un plugin, es usado en este proyecto para crear los diagramas de secuencia.

Engine 3D: Un motor 3D es una colección de estructuras, funciones y algoritmos utilizados para visualizar, después de realizar cálculos y transformaciones, objetos tridimensionales en una pantalla bidimensional.

Estímulo: cualquier cosa que influya efectivamente sobre los aparatos sensitivos de un organismo viviente, incluyendo fenómenos físicos internos y externos del cuerpo.

EVEMEH: acrónimo de Entornos Virtuales para la Evaluación de la Memoria Espacial en Humanos, es una aplicación que permite implementar en humanos las tareas efectuadas por los modelos animales en el laboratorio. Consta de un entorno virtual 3D que permite conocer cómo se comporta el ser humano en tareas que demandan memoria espacial.

F

Frame: cuadro o imagen individual de las que componen una animación. La sucesión de frames consecutivos con pequeñas diferencias producen la ilusión de movimiento. La velocidad de la animación se mide en Frames por segundo (FPS).

Frames por segundo: medida de rendimiento que equivale a la velocidad de la animación, es decir, el número de frames o imágenes que se muestran en un segundo por pantalla.

G

Geometry Shader: Es capaz de generar nuevas primitivas dinámicamente.

GLU: es el acrónimo de OpenGL Utility. Esta biblioteca está compuesta por una serie de funciones de dibujo de alto nivel que, a su vez, se basan en las rutinas primitivas de OpenGL y se suele distribuir normalmente junto a él. Entre sus características podemos encontrar el mapeado entre pantalla y coordenadas, generación de texturas mipmap, dibujado de superficies cuádricas, NURBS, texelación de primitivas poligonales, etc.

GLUI: Interfaz de usuario basada en GLUT. Proporciona elementos de control tales como botones, cajas de selección y spinners. Es independiente del sistema operativo, sustentándose en GLUT para manejar los elementos dependientes del sistema.

GLUT: OpenGL Utility Toolkit, es una librería de utilidades para programas OpenGL que principalmente proporciona diversas funciones de entrada/salida con el sistema operativo. Entre las funciones que ofrece se incluyen declaración y manejo de ventanas y la interacción por medio de teclado y ratón.

Gráficos 3D: trabajos de arte gráfico que fueron creados con ayuda de ordenadores y programas especiales 3D. En general, el término puede referirse también al proceso de crear dichos gráficos, o el campo de estudio de técnicas y tecnología relacionadas con los gráficos 3D.

L

Layout: subprograma perteneciente a LightWave 3D en el que se **r**ealiza el *Rigging* o configuración del esqueleto de una malla, Dinámicas, FX, Render, configuración de cámaras y luces.

LightWave 3D: sistema de modelado, renderizado y animación. Usado extensamente en producciones televisivas, efectos especiales en películas, desarrollo de videojuegos, impresión de gráficos y visualización. LightWave 3D se divide en dos subprogramas: Modeler y Layout.

M

Mapeado de texturas: El mapeo de texturas es el método de adición de detalles, superficies o colores a un modelo 3D generado por ordenador.

Memoria Espacial: memoria íntimamente relacionada con la orientación espacial. Permite recordar la posición de determinados objetos o estímulos que podrán ser usados como orientación a la hora de ubicarse en un contexto dado.

Memoria de Trabajo: es la memoria que guarda y procesa durante breve tiempo la información que viene de los registros sensoriales y actúa sobre ellos y sobre otros. Esta memoria nos capacita para recordar la información pero, es limitada y susceptible de interferencias.

Microsoft Project: herramienta software que permite realizar planificaciones para proyectos. Entre otras funciones importantes puede realizar red de tareas PERT y diagramas Gantt.

Microsoft Visual Studio: entorno de desarrollo integrado para sistemas Windows. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET.

Mipmap: son colecciones de imágenes de mapas de bits que acompañan a una textura principal para aumentar la velocidad de renderizado y reducir sus artefactos. Son ampliamente usados en los videojuegos en 3D, simuladores de vuelo, y otras aplicaciones con imágenes tridimensionales.

Modeler: subprograma perteneciente a LightWave 3D, en el que se realiza el modelado del objeto con una filosofía orientada a capas.

N

NURBS: Acrónimo inglés de la expresión *Non Uniform Rational B-Splines*. Modelo matemático muy utilizado en los gráficos por ordenador para generar y representar curvas y superficies.

O

OpenGL: OpenGL (Open Graphics Library) es una especificación estándar que define una API multilenguaje y multiplataforma para escribir aplicaciones que produzcan gráficos 2D y 3D. La interfaz consiste en más de 250 funciones diferentes que pueden usarse para dibujar escenas tridimensionales complejas a partir de primitivas geométricas simples, tales como puntos, líneas y triángulos

P

Phyton: lenguaje de script de Blender que permite programar algunas características del game engine que incorpora.

Pixel Shader: Un pixel shader sirve para manipular un píxel, o lo que es lo mismo, aplicar un efecto sobre la imagen (realismo, bump maping, sombras, explosiones y efectos). Se trata de una función gráfica que calcula los efectos sobre una base per-pixel. Dependiendo de la resolución, una cantidad de 2 millones de píxeles puede ser necesaria para ser renderizado, iluminado, sombreado, y coloreado para cada marco.

R

Rasterizar: operación de convertir una imagen vectorial en otra de mapa de bits.

Realidad Virtual: Aquella tecnología informática que genera entornos tridimensionales con los que el sujeto interactúa en tiempo real, produciéndose de esa manera una sensación de inmersión semejante a la de presencia en el mundo real.

Render: término que se refiere a la generación de un frame o imagen individual. El render consume potencia de cómputo debido a los cálculos necesarios según el número de objetos, luces y efectos en la escena y según el punto de vista de la cámara.

Renderman: es una API desarrollada por Pixar para transformar las escenas tridimensionales de sus producciones en imágenes digitales de alta calidad.

Rigging: configuración del esqueleto de una malla para prepararla para una animación.

S

SDL: librería auxiliar de OpenGL que ayuda a gestionar aspectos como los eventos producidos por el teclado o ratón, y añade mucha más funcionalidad a OpenGL. Ofrece mayor rendimiento que otras librerías en principio equivalentes.

Shader: es un conjunto de instrucciones gráficas destinadas para el acelerador gráfico, estas instrucciones dan el aspecto final de un objeto. Los Shaders determinan materiales, efectos, color, luz, sombra y etc. Pueden ser pixel shader, vertex shader o geometry shader.

Softbody: objetos de cuerpo blando, o sea, objetos para simular ropa o banderas por ejemplo.

T

Textura: imagen proyectada sobre parte o la totalidad de la superficie de un objeto para darle apariencia foto realística. Existen texturas procedimentales (calculadas) o texturas de mapa de imagen.

V

Vertex Shader: es una herramienta capaz de trabajar con la estructura de vértices de los modelos tridimensionales y con ello realizar operaciones matemáticas modificando estas variables y así definiendo colores, texturas e incidencia de la luz.

W

WBS: diagrama de descomposición donde se visualizan las diferentes tareas o procesos que se tienen que realizar.

X

XBOX360: consola de la actual generación de consolas perteneciente a Microsoft. Se pueden programar juegos en XNA para ella.

XNA: plataforma de desarrollo de videojuegos para PC y XBOX360 de Microsoft. Sigue la filosofía de ofrecer la máxima funcionalidad intentando simplificar lo máximo posible el desarrollo del juego.

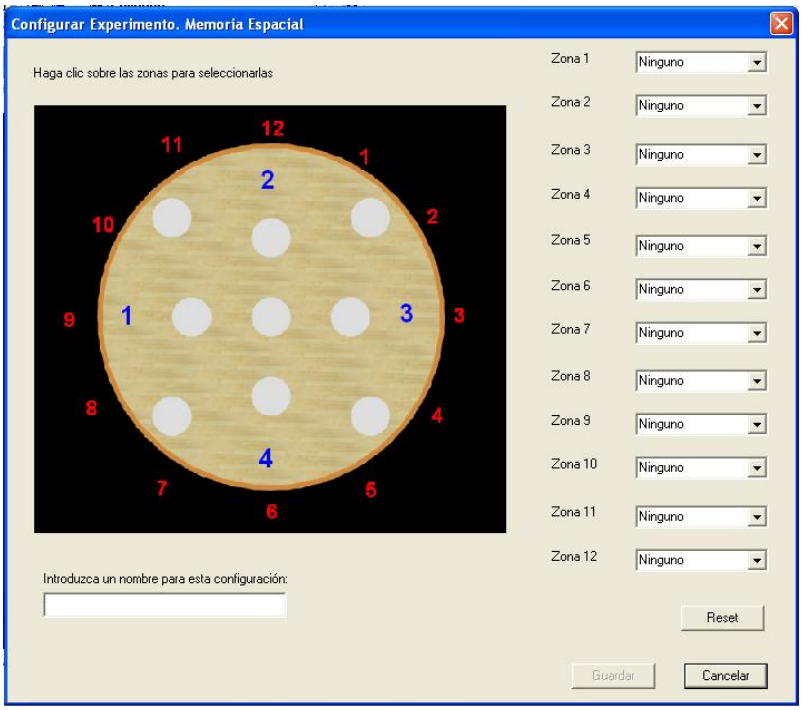
Z

Z-buffering: es la parte de la memoria de un adaptador de video encargada de gestionar las coordenadas de profundidad de las imágenes en los gráficos en tres dimensiones (3-D), normalmente calculados por hardware y algunas veces por software.

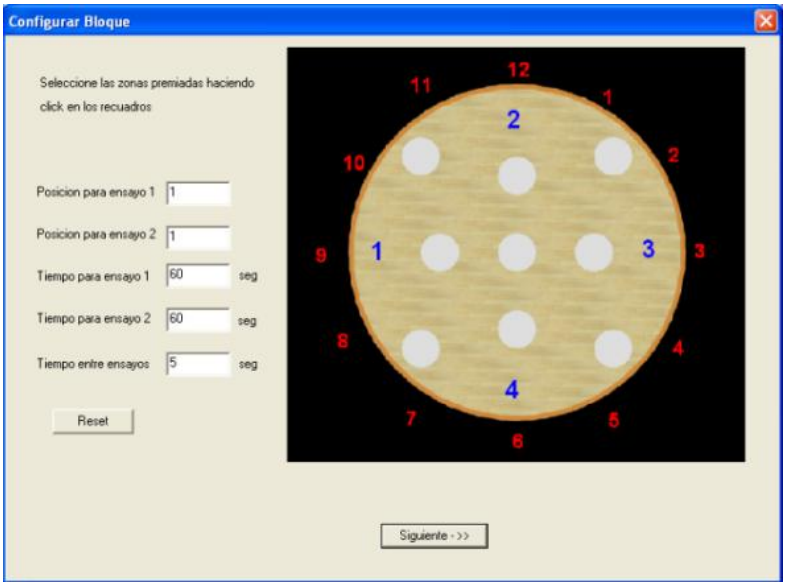
Capítulo 7

Anexos

Anexo 1.1



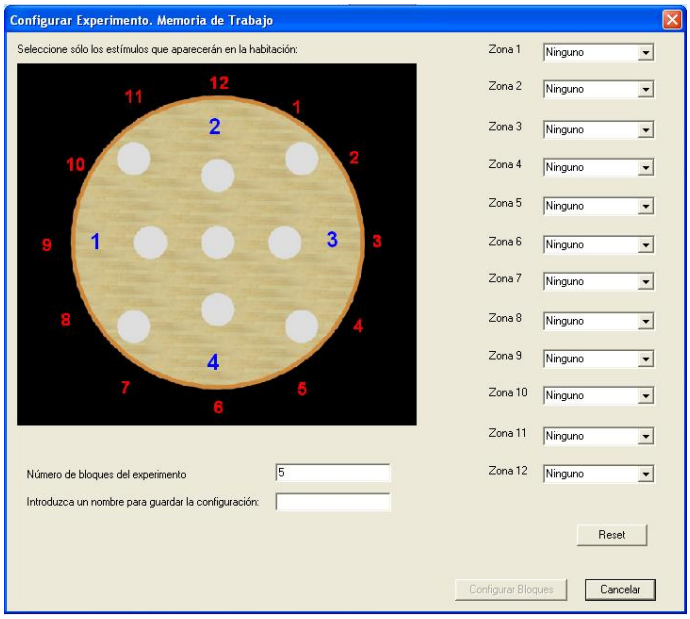
Anexo 1.2



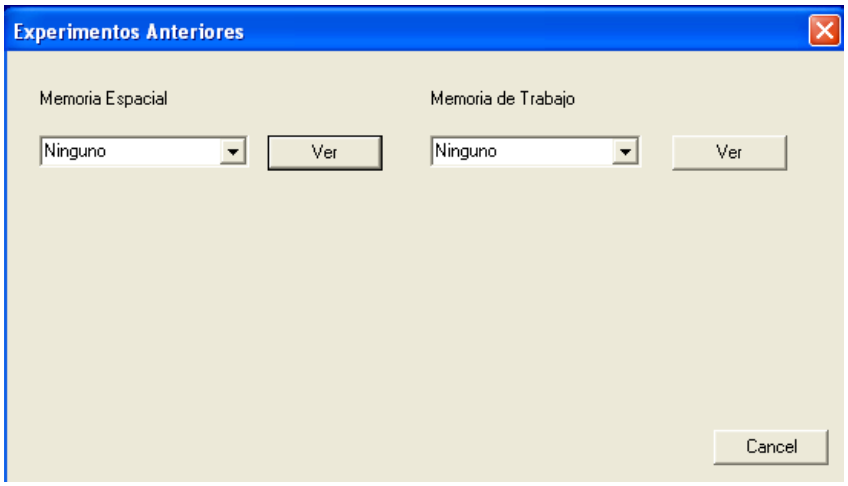
Anexo 1.3

```
- <experimentoMT>
  - <estimulos>
    - <estimulo>
        <zonaH>3</zonaH>
        <objeto>chimenea</objeto>
      </estimulo>
    - <estimulo>
        <zonaH>6</zonaH>
        <objeto>lampara</objeto>
      </estimulo>
    - <estimulo>
        <zonaH>9</zonaH>
        <objeto>chimenea</objeto>
      </estimulo>
    - <estimulo>
        <zonaH>12</zonaH>
        <objeto>lampara</objeto>
      </estimulo>
    </estimulos>
  - <bloques>
    - <bloque>
        <posicion1>1</posicion1>
        <posicion2>2</posicion2>
        <tiempo1>60</tiempo1>
        <tiempo2>60</tiempo2>
      - <zonasPremiadas>
          <zonaPremiada>2</zonaPremiada>
          <zonaPremiada>6</zonaPremiada>
        </zonasPremiadas>
        <tiempoEntreEnsayos>5</tiempoEntreEnsayos>
      </bloque>
    - <bloque>
        <posicion1>3</posicion1>
        <posicion2>4</posicion2>
        <tiempo1>60</tiempo1>
        <tiempo2>60</tiempo2>
      - <zonasPremiadas>
          <zonaPremiada>3</zonaPremiada>
          <zonaPremiada>5</zonaPremiada>
        </zonasPremiadas>
        <tiempoEntreEnsayos>5</tiempoEntreEnsayos>
      </bloque>
    </bloques>
</experimentoMT>
```

Anexo 1.4

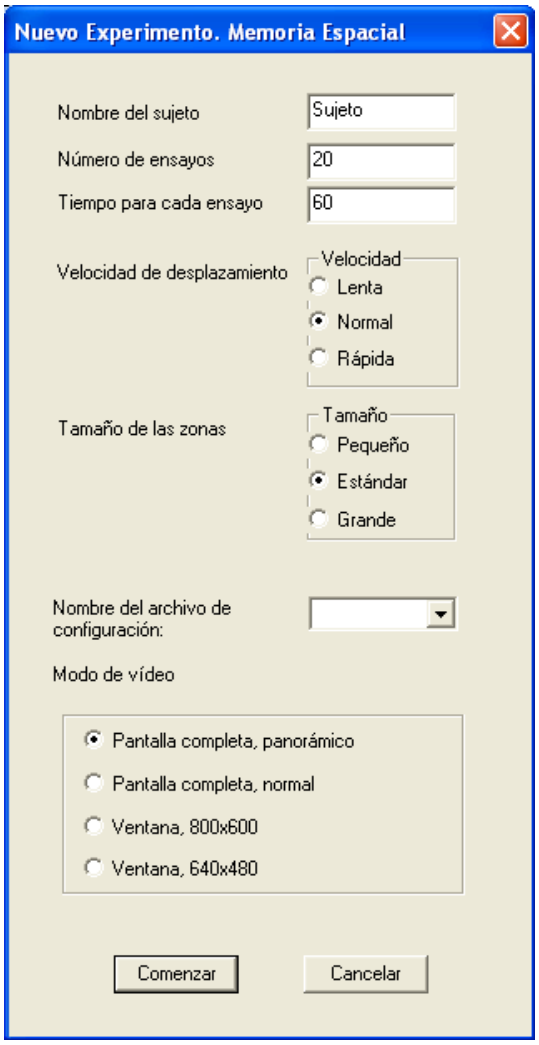


Anexo 1.5

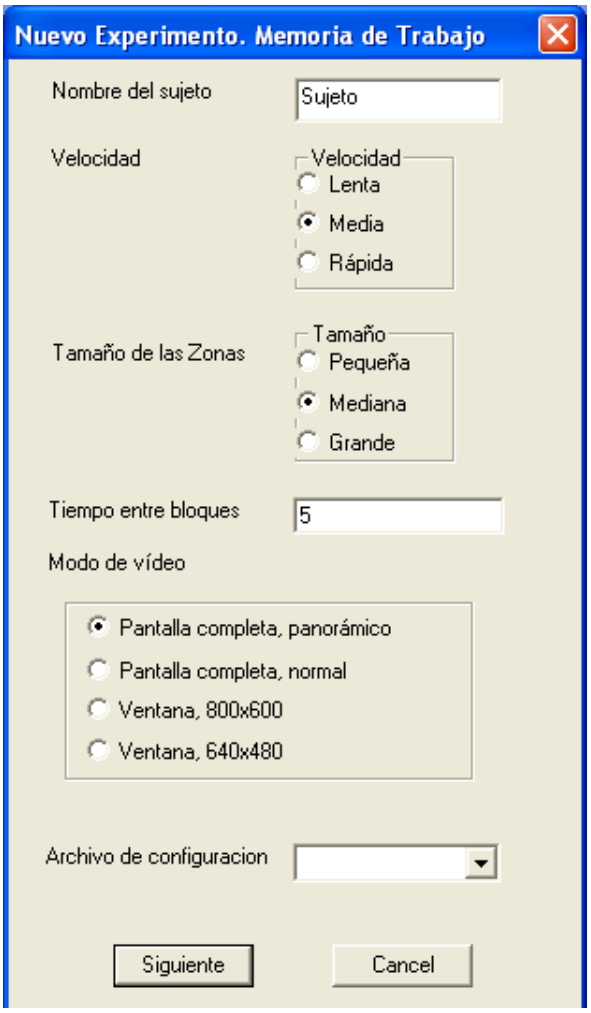


Anexo 1.6

Anexo 1.7


Nuevo Experimento. Memoria Espacial
Nombre del sujeto
Sujeto
Número de ensayos
20
Tiempo para cada ensayo
60
Velocidad de desplazamiento
Velocidad
Lenta
Normal
Rápida
Tamaño de las zonas
Tamaño
Pequeño
Estándar
Grande
Nombre del archivo de configuración:
Modo de vídeo
Pantalla completa, panorámico
Pantalla completa, normal
Ventana, 800x600
Ventana, 640x480
Comenzar
Cancelar

Anexo 1.8


Nuevo Experimento. Memoria de Trabajo
Nombre del sujeto
Sujeto
Velocidad
Velocidad
Lenta
Media
Rápida
Tamaño de las Zonas
Tamaño
Pequeña
Mediana
Grande
Tiempo entre bloques
5
Modo de vídeo
Pantalla completa, panorámico
Pantalla completa, normal
Ventana, 800x600
Ventana, 640x480
Archivo de configuracion
Siguiente
Cancel

Anexo 1.9

Anexo 1.10

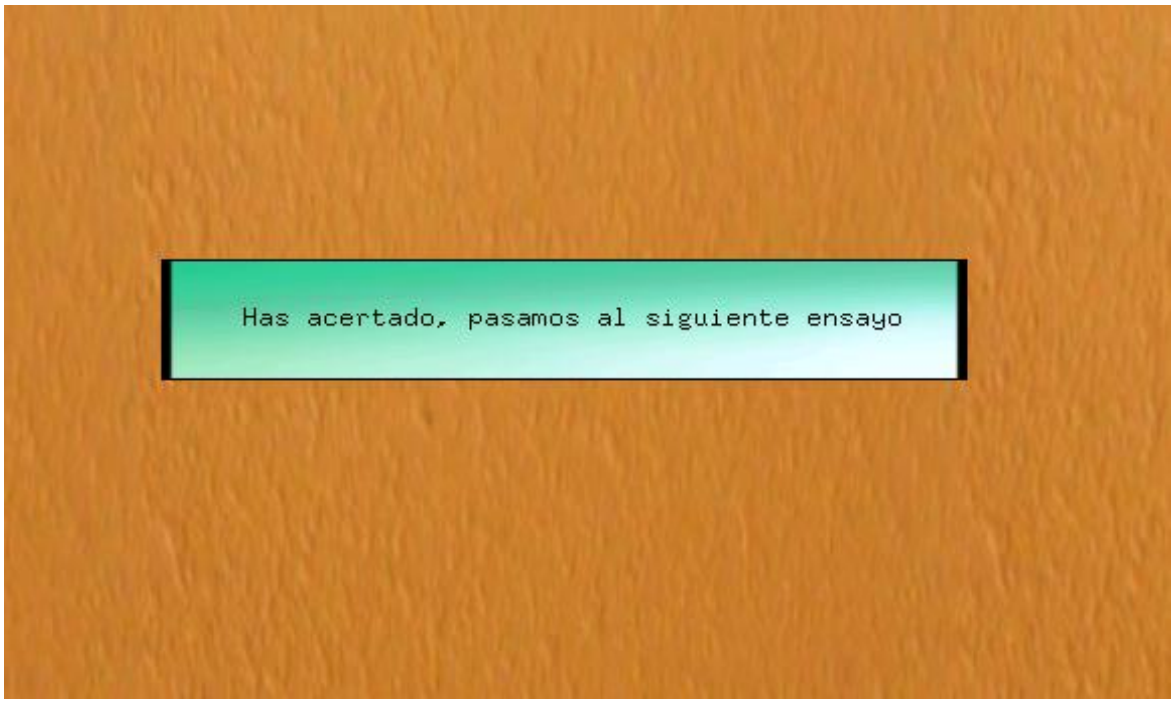
Anexo 1.11

Anexo 1.12

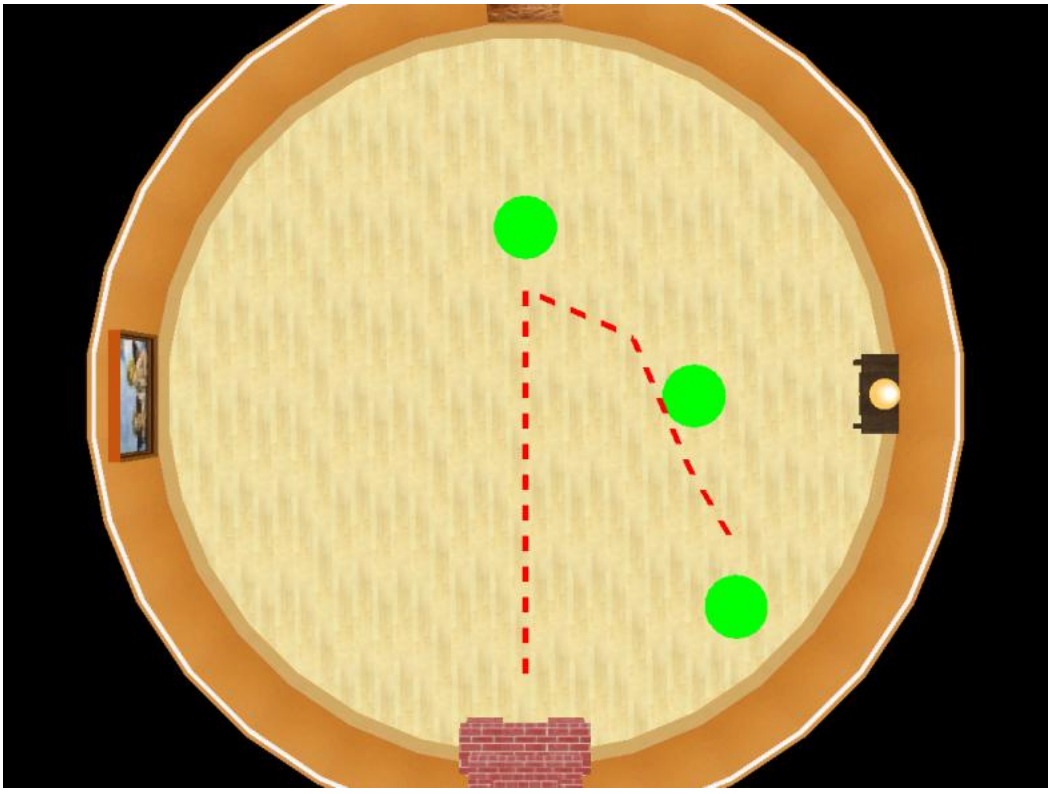
Anexo 1.13

Anexo 1.14

Anexo 1.15

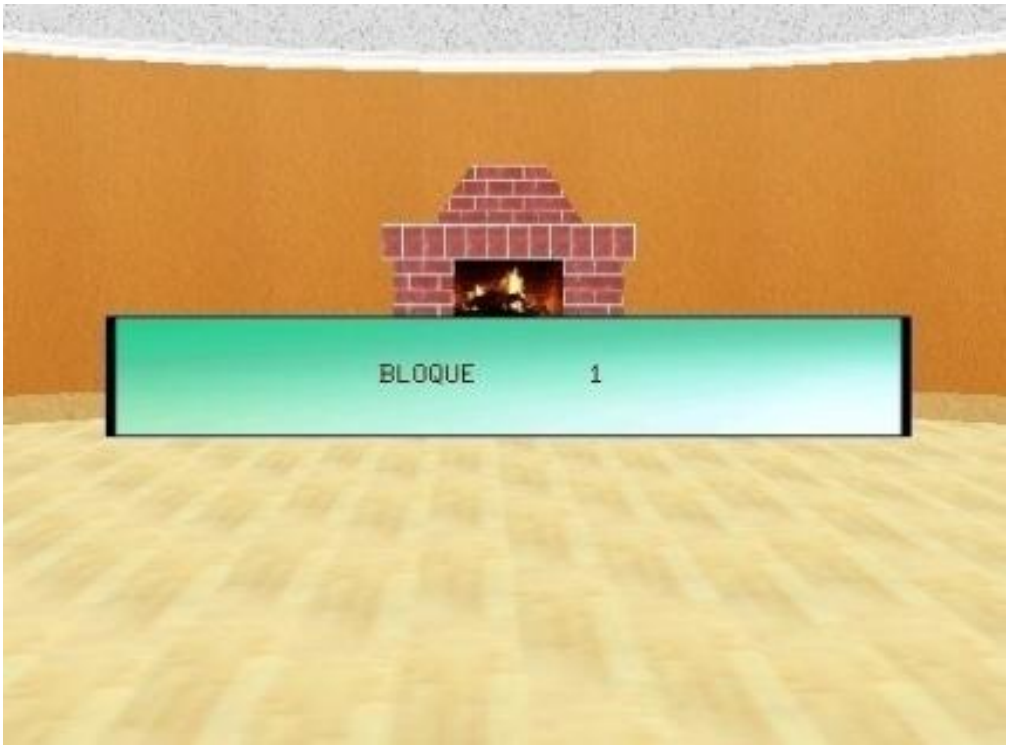


Anexo 1.16

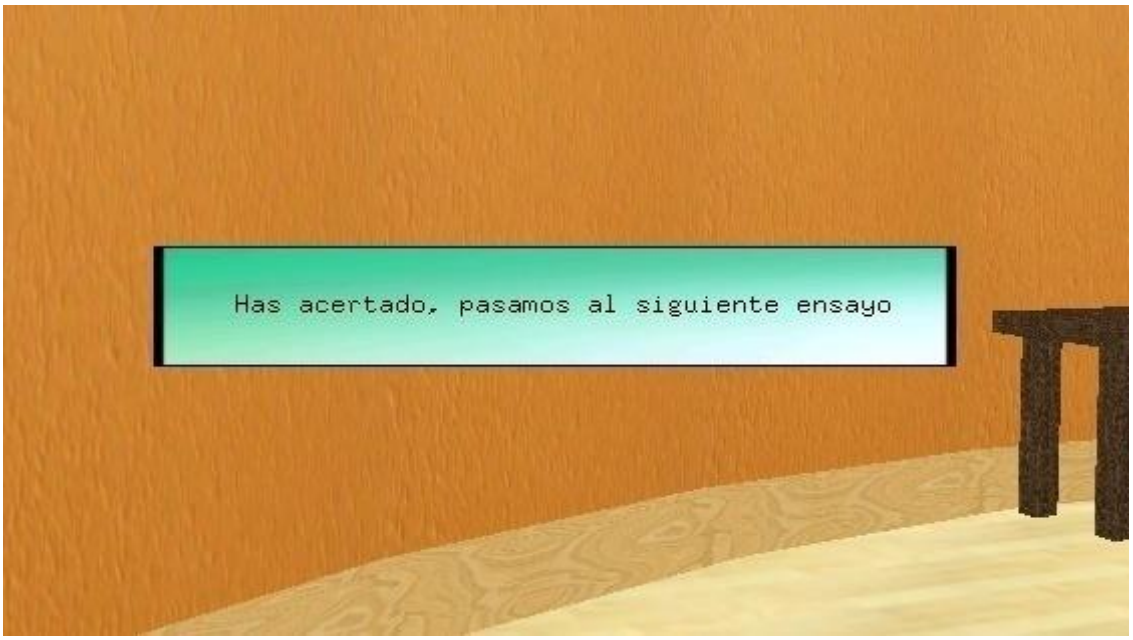


Anexo 1.17

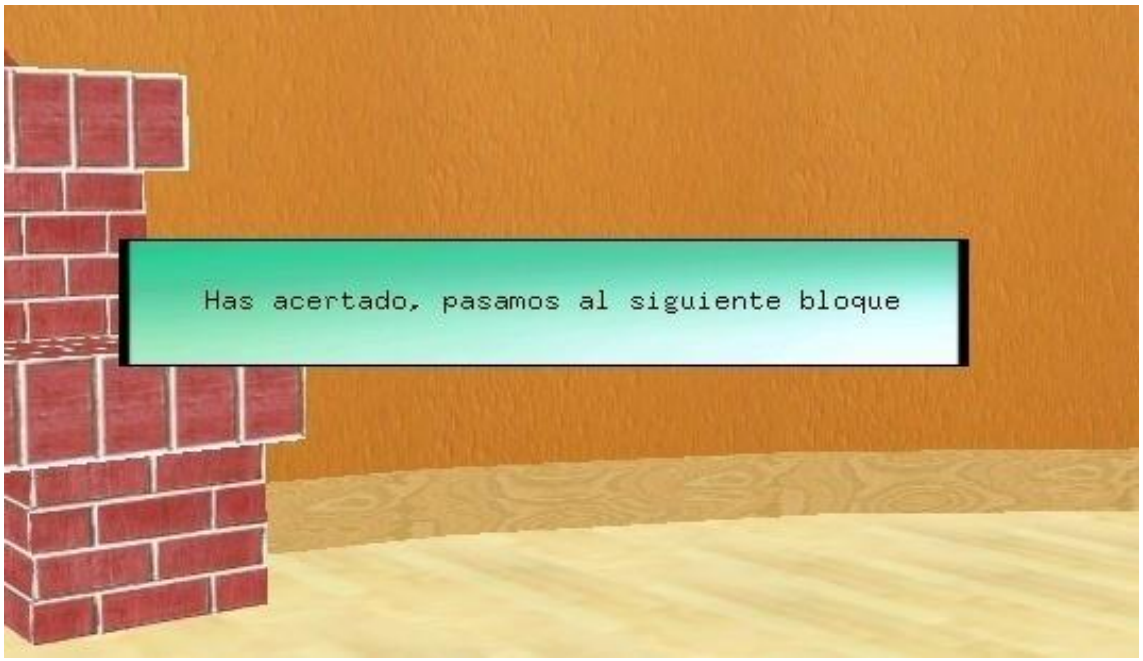
Anexo 1.18



Anexo 1.19



Anexo 1.20



Anexo 21