Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática
(Mención de Ingeniería de Software)

Creación de un Juego para el Aprendizaje de Estructuras de Datos Estáticas

Autor:
D. Raúl Fernández Magdaleno

Tutores:
Dña. Alma María Pisabarro Marrón
Dr. Carlos Enrique Vivaracho Pascual

Agradecimientos

Desde aquí quiero agradecer a todas las personas que han hecho posible la realización de este proyecto.

En primer lugar a la profesora **Dña. Arancha Simon Hurtado** la cual me puso en contacto con los profesores que han supervisado este trabajo, sin ella este trabajo nunca se hubiera llevado a cabo.

Además a los compañeros y conocidos que han probado los distintos juegos, de esta manera logrando su mejora mediante la solución de diversos errores.

Resumen

Con este trabajo se pretende enseñar de una forma divertida, mediante juegos, los distintos conceptos relacionados con las estructuras de datos que se imparten en la asignatura de 1º, Fundamentos de Programación del grado en Ingeniería Informática y grado en Estadística.

Utilizando la herramienta de trabajo, Unity 3D se han realizado una serie de juegos para explicar los conceptos básicos de las estructuras de datos: *Strings*, *Arrays* unidimensionales, *Arrays* bidimensionales, Registros y Ficheros.

Índice

Índice de tablas

Índice de figuras

1. Introducción

1.1. Descripción

Este trabajo forma parte de un proyecto de gamificación de la asignatura de 1º, Fundamentos de Programación del grado en Ingeniería Informática y grado en Estadística, con el cual se busca explicar los distintos conceptos impartidos en esta mediante el desarrollo de juegos educativos, obteniendo de esta manera una mayor implicación de los estudiantes con la asignatura.

Este proyecto lleva desarrollándose varios años, y utilizándose durante el último curso, tanto en el grado en Ingeniería Informática, como en el grado en Estadística que se imparten en la UVa y ha tenido muy buen recibimiento por los alumnos, además de que, según su propia opinión, muchos de ellos han logrado mejorar y entender los conocimientos relacionados con la asignatura de Fundamentos de Programación. Es por ello, que en los últimos años, se han estado ofreciendo TFGs que amplíen el catálogo de juegos de este proyecto.

La gamificación [11] se basa en una metodología mediante la cual se traslada el mecanismo de los juegos a otros ámbitos no lúdicos. En particular, en el entorno educativo se puede utilizar el potencial de los juegos para intentar facilitar el proceso de aprendizaje de los alumnos.

En este caso la gamificación consiste en una serie de juegos, cada uno de los cuales intenta reforzar diferentes conceptos de la asignatura mencionada. En este trabajo en particular se desarrollarán varios juegos para la comprensión de las diferentes estructuras de datos estáticas: *Strings*, *Arrays* unidimensionales, *Arrays* bidimensionales, Registros y Ficheros.

Para facilitar la comprensión de las diferencias entre cada una de estas estructuras, se decidió que el juego constase de varios minijuegos, cada uno de ellos dedicado a una estructura de datos diferente. Además se añadió un juego final, con el que el alumno pueda verificar si ha comprendido las particularidades de cada una de las estructuras. Cada uno de ellos será explicado en capítulos distintos para diferenciarlos de una mejor manera.

1.2. Motivación

Es un hecho que los alumnos que estudian temas relacionados con la informática tienen una estrecha relación con el mundo de los videojuegos [10], algunos los consumirán en mayor o menor medida, pero la gran mayoría están relacionados con ellos.

Hay muchos conceptos que al explicarse con palabras pueden ser difíciles de entender, asimilar o incluso memorizar. Si pudieran explicarse mediante un juego, el alumno podría participar activamente y además, teniendo en cuenta la relación entre el tipo de alumnos que nos concierne y los videojuegos, estaría más motivado, facilitando su comprensión.

Esta es una de las ideas principales en las que se fundamenta la técnica de la gamificación, en la que se basa el desarrollo de este proyecto.

1.3. Listado de objetivos

Objetivo 1. Desarrollo del juego principal El desarrollo de los distintos juegos para explicar las estructuras de datos mencionadas anteriormente es el primer objetivo.

Objetivo 2. Mantenimiento de otros juegos Como segundo objetivo está el mantenimiento o solución de errores de juegos ya existentes en la plataforma. Este apartado será tratado más a fondo en su correspondiente sección.

2. Contexto

2.1. La Gamificación

Los videojuegos llevan muchos años con nosotros y tienen una gran cantidad de seguidores de todas las edades y géneros, no importa si eres mayor o pequeño, si eres hombre o mujer, no hace distinciones de ningún tipo. Por ello día a día son más las personas que ya sea para entretenerse o para desconectar un poco de la realidad se suman a este mundo.

Aunque la gran mayoría de las personas utilizan los videojuegos como una forma de diversión, en los últimos años ha ganado bastante influencia su uso en otros campos (gamificación). En la educación se utiliza la denominada *gamificación en el aula* para motivar a los estudiantes mediante el uso de juegos.

A continuación se mostrarán algunos de los beneficios [1] de este planteamiento:

- **La gamificación aumenta la motivación por el aprendizaje** El hecho de poder jugar a algo puede ser un gran incentivo para aprender.
- **La dificultad va en aumento** Cada juego se diseña atendiendo a ciertas dificultades, y poder desbloquear más niveles a medida que juegas es importante para la motivación.
- **Hace más divertidas las asignaturas** Dado que jugar es más divertido que leer o escuchar esto hace que la gamificación sea un método educativo más entretenido que los tradicionales.
- **Favorece la adquisición de conocimientos** Esto se debe a que la adquisición de conocimientos está directamente relacionada con el interés por algo y, como ya se ha dicho, la motivación que exista por ese tema.
- **Mejora el rendimiento académico** Como ya ha sido mencionado, un mayor interés por algo hace que el aprendizaje sea mayor y esto se verá reflejado en las calificaciones.
- **Estimula las relaciones sociales** El poder competiro trabajar en equipo con los compañeros mientras se juega, puede abrir muchas puertas

a nuevas relaciones sociales.

Esos son algunos de los beneficios de la gamificación aunque existen muchos más. Pero también hay inconvenientes [2], entre ellos se pueden mencionar:

- **Posibilidad de ser distraídos por el juego y la consiguiente pérdida de tiempo/productividad** Demasiado interés por algo puede dar lugar a que empleemos mucho tiempo en ello, desatendiendo otros deberes.
- **La posibilidad de crear una motivación pasajera** Esto se debe a que si una persona no es capaz de obtener ciertos logros puede perder el interés por algo, aunque también puede deberse a que la actividad no es suficientemente interesante.
- **Ganar se convierte en el objetivo principal** El objetivo de la gamificación en el aula es que el estudiante aprenda, si ganar se convierte en el objetivo principal el aprendizaje queda relegado, por lo que la técnica resulta inservible.

2.2. Conceptos Sobre el Desarrollo de Videojuegos, el GDD

Este es el *Documento de Diseño del Juego* [3] que contiene toda la información necesaria sobre el desarrollo de este.

Cada juego posee un GDD distinto, es decir, no está sujeto a una estructura estándar y a pesar de que hay muchas guías para realizar un buen GDD no todos son iguales.

En el caso de este juego los puntos a tratar son:

- **Resumen:** Exposición de forma resumida de lo que trata el juego.
- **Descripción:** Explicación detallada del funcionamiento del juego.
- **Audiencia:** Descripción del grupo de personas a las que va dirigido principalmente el juego.
- **Plataforma:** Plataforma sobre la que funcionará el juego.
- **Mecánicas:** Controles o características del juego (como se juega o que acciones especiales se toman).
- **Mundo:** Donde está ambientado el juego, generalmente se acompaña de un trasfondo o historia.
- **Recursos:** Los utilizados en la creación del juego. Dependiendo del juego pueden incluir entornos, personajes, texturas, animaciones, música, efectos etc...

3. Herramientas Utilizadas

3.1. Unity 3D

Unity es una herramienta para el desarrollo de vídeojuegos muy utilizada en el mundo, que permite diferenciar entre el desarrollo de juegos 2D, que suelen tener un estilo *Indie*, y juegos 3D, que es en lo que nos vamos a enfocar.

Dado que la plataforma en la que se va a alojar el juego está hecha explícitamente para juegos desarrollados con *Unity*, el uso de esta herramienta es un requisito imprescindible.

Unity además cuenta con la gran cualidad de ser un motor multiplataforma, lo cual permite desarrollar gran variedad de videojuegos distintos para diferentes plataformas, ya sea un teléfono móvil, una consola, un ordenador o realidad virtual (VR) y algunas otras.

Todas estas cualidades hacen que cada vez más gente se sume al uso de *Unity*, a pesar de que con los años, otros motores como *Unreal* están empezando a destacar. Esto hace que continuamente salgan actualizaciones tanto en la aplicación, como en su página web [4] que cuenta con numerosas herramientas muy útiles.

- **Asset Store:** Esta ha sido el complemento más utilizado en el desarrollo de este proyecto, ya que esta tienda proporciona de forma gratuita algunos *Assets*, es decir, complementos que mejoran el desarrollo de un videojuego, de entre todos ellos destacan los *prefabs*, objetos prefabricados listos para su uso, o los packs de texturas.

- **Unity Learn:** Esta sección contiene una gran cantidad de tutoriales e información muy útil para aprender sobre *Unity*.

- **Unity ProBuilde:r** Esta herramienta, aunque no ha sido utilizada en el desarrollo de este proyecto es muy interesante, ya que permite, de forma bastante intuitiva el desarrollo de mapeados u objetos para los juegos.

- **Unity Git:** Es un plugin el cual permite tener *Git Hub* listo desde la propia herramienta de *Unity*.

3.1.1. Estructura de un Juego en *Unity*

La base sobre la que se asienta un juego en *Unity*, son las *escenas*. Desde estas se desarrollan los videojuegos, es decir, una escena es el mapa donde se sitúa la acción.

Este mapa contiene los distintos *Game Objects*, estos son los componentes que forman la escena, un *Game Object* puede ser el propio terreno, una iluminación, la cámara o un efecto.

Los *Game Objects* a su vez contienen distintos componentes, el componente principal es el *Transform*, este componente contiene la información sobre la posición, escala y dirección del objeto en la escena. Un *Game Object* a su vez puede contener distintos componentes, de entre ellos destacan, el *Character Controller* el cual permite dotar de movimiento a un objeto. A su vez existe también el *Rigid Body* que permite dotar propiedades reales al objeto, como podrían ser la masa o la gravedad.

Otros componentes muy utilizados son los *Colliders* estos permiten el choque entre objetos, o activando la opción *Is Trigger*, el collider se convierte en un activador, es decir, si se entra en el rango del trigger se realiza cierta acción desde un *Script*.

Estos *Scripts* definen el comportamiento de los objetos, sin ellos no se podría realizar un juego dado que estos controlan todas las acciones realizadas por el jugador, el movimiento del ratón o el uso del teclado.

Otro componente muy útil es el *Animator* este, como su nombre indica, permite añadir animaciones.

3.2. Gestor de Proyectos Monday

Esta es una página web [5] de gestión de proyectos, siguiendo los principios de las metodologías ágiles esta página permite la creación de un *Espacio de Trabajo*, desde el cual se pueden ir creando *Sprints* y desde estos, añadir *Historias de Usuario* y gestionarlas desde un *Tablero Canvan*. Dado que este proyecto será realizado por una sola persona, su uso se verá limitado, así que no se comentarán todas las posibilidades.

Se tratará un poco más sobre el tema más adelante, en el apartado de metodologías.

3.3. Mixamo

Esta página web [6] permite la obtención de forma totalmente gratuita, de personajes listos para su uso, y de numerosas animaciones. Estos personajes o animaciones simplemente hay que descargarlos e importarlos al proyecto *Unity*.

3.4. Turbosquid

Probablemente la página web [7] más utilizada en el desarrollo del proyecto, ya que esta permite la descarga gratuita de numerosas animaciones, vídeos, texturas, plugins, música y *prefabs* (como ya se ha comentado, son objetos prefabricados, listos para su uso). Hay numerosos tipos de formato, desde modelos listos para importar en la herramienta de *Blender* o *prefabs* listos para su uso, que generalmente tienen el formato `.obj` o `.fbx`. Estos en numerosas ocasiones vienen incluso con packs de texturas personalizados. Es una herramienta muy útil que ha agilizado enormemente el desarrollo del juego.

3.5. Astah

Astah es una herramienta de modelado UML. La UVa posee una licencia para el uso gratuito de dicho software por los alumnos. Con ella se han realizado los distintos diagramas que se muestran en los GDDs [15].

3.6. OBS

OBS o *Open Broadcaster Studio* es un software gratuito y de código abierto para la grabación y transmisión de vídeo por internet. Con él se han grabado los vídeos que se muestran en los tutoriales de los juegos [17].

3.7. VideoPad

VideoPad es un software de edición de vídeo gratuito con el cual se han editado los vídeos que aparecen en los tutoriales [18].

3.8. Iconos8

Esta es una página web que provee de numerosos tipos de iconos los cuales puedes descargar de forma totalmente gratuita, si estos se encuentran disponibles dado que también tiene contenido de pago [16].

3.9. Overleaf & Latex

Overleaf [19] es editor de *Latex* online (*Latex* es un sistema de composición de textos, orientado a la creación de documentos escritos) desde el cual se ha realizado este documento TFG.

4. Metodología

4.1. Metodología de desarrollo

Para el desarrollo de este proyecto se ha seguido la metodología Ágil [9]. El motivo de esto han sido los conocimientos previos sobre esta metodología, impartidos en otras asignaturas de la carrera y además el hecho de que el uso de estas metodologías favorece a las empresas de tal manera, que consiguen gestionar sus proyectos de una manera flexible y eficaz reduciendo los costes e incrementando su productividad.

Esta metodología sigue los doce principios del manifiesto ágil [8]. En concreto se empleará la estrategia *Scrum* el cual consiste en un proceso iterativo, en el cual se efectúan *Sprints* que contienen las distintas *historias de usuario*. De esta manera se realizarán diferentes *Sprints* de unas 2 semanas de duración, y al pasar ese tiempo se presentará el trabajo.

Esta estrategia resulta muy útil con este proyecto debido a que este ha de ser revisado periódicamente por los tutores, ha de tener un trabajo continuo y además es realizado por pocas personas, en concreto una.

En la Figura 1 se puede apreciar el flujo de trabajo en un proyecto con metodología ágil:

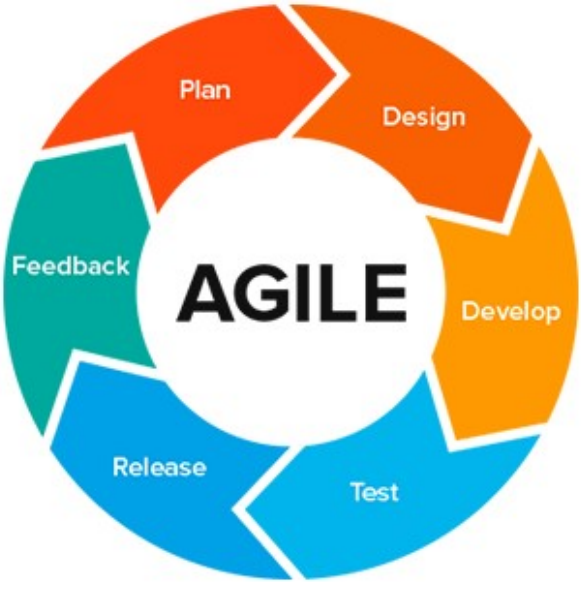

Figura 1: Representación de la metodología ágil.

Con esta metodología se empleó además el uso de una de las herramientas mencionadas anteriormente en la sección *Herramientas Utilizadas* llamada *Monday.com* esta página web es un gestor de proyectos online gratuito que sigue la metodología ágil, el cual permite realizar *Sprints* y gestionarlos desde un *Tablero Kanban*.

4.2. Organización del proyecto

Un proyecto *Scrum* cuenta de los siguiente usuarios:

- **Product Owner** Es el gestor de requisitos o un cliente que se encarga de gestionar el *Product Backlog*.
- **Scrum Master** Es el encargado de gestionar y asegurar que el proceso se lleva a cabo correctamente. Además se encarga de gestionar y eliminar posibles impedimentos.
- **Equipo de Desarrollo** Suele estar formado por entre 3 a 9 personas que se encargan de desarrollar el producto.

En este caso, el *Scrum Master* equivaldría a los tutores, encargados de gestionar el trabajo realizado e ir incrementándolo a medida que avanza el tiempo, y el equipo de desarrollo estaría formado por una sola persona que sería el alumno en cuestión.

Dado que no existe un posible cliente, en este caso el *Product Owner* serían los tutores que han encargado el trabaj, mientras que el alumno tomaría todas las actividades de *Analista, Diseñador y Desarrollador*.

5. Planificación Inicial

5.1. Distribución del Trabajo

Para la distribución del trabajo se tuvo en cuenta, tanto el tiempo disponible hasta la entrega del proyecto, como la complejidad en las distintas actividades que tenían que realizarse para el correcto cumplimiento de este.

El tiempo disponible para realizarlo comprendía desde mediados de febrero hasta finales de mayo. Teniendo esto en cuenta, y siguiendo la metodología explicada anteriormente, el proyecto se dividió en *Sprints* que habrían de entregarse progresivamente.

Para el Grado en Ingeniería Informática de la Universidad de Valladolid el TFG posee un valor de 12 créditos, lo que corresponde a 300 horas de trabajo. Debido al tiempo relativamente ajustado, los *Sprints* serían realizados en periodos de menor duración al principio y con mayor duración al final, agilizando de esta manera el trabajo inicial y manteniendo un ritmo constante durante las etapas media y final.

De esta forma el número estimado fue de 8 *Sprints*, con una duración de una semana para los cuatro primeros y de tres semanas para los cuatro últimos, derivando en un total de 16 semanas.

El total de horas trabajadas debe ajustarse, en la medida de lo posible a 300 horas como máximo.

5.1.1. Resumen de Tiempos

Planificación del Proyecto	
Fecha de Inicio	08/02/2022
Fecha de Fin	31/05/2022
Periodo de Sprints	1 y 3 semanas
Carga de Trabajo Diaria	Entre 1 y 4 Horas
Horas Totales Previstas	Aproximadamente 300 Horas

Cuadro 1: Tabla resumen de tiempos

5.1.2. Planificación de Sprints

Resumen de Sprints	
Sprint 1	08/02/22 a 15/02/22
Sprint 2	15/02/22 a 22/02/22
Sprint 3	22/02/22 a 01/03/22
Sprint 4	01/03/22 a 08/03/22
Sprint 5	08/03/22 a 29/03/22
Sprint 6	29/03/22 a 19/04/22
Sprint 7	19/04/22 a 10/05/22
Sprint 8	10/05/22 a 31/05/22

Cuadro 2: Tabla resumen de tiempos

6. Coste Inicial

6.1. Coste del Proyecto

En España un diseñador junior de videojuegos suele trabajar asistiendo a los diseñadores senior y su sueldo varía entre los 18.000 y los 26.000 [12] euros brutos anuales. El tiempo dedicado al proyecto ha sido de cuatro meses aproximadamente por lo que esto supone un coste total de 2100 euros, el equivalente a aproximadamente 7 euros/hora de trabajo, [13] para el total propuesto de 300 horas.

Los costes materiales suponen los del equipo informático utilizados para trabajar, estos equivalen aproximadamente a 1100 euros.

En cuanto al software utilizado, todos los programas son gratuitos o poseen licencias especificas para estudiantes. Los costes de recursos, como podrían ser compras en la *Asset Store*, han sido de 0 euros.

6.1.1. Resumen de costes del proyecto

Resumen de Costes	
Costes de personal	2100 €
Costes materiales	1100 €
Costes de software	0 €
Costes de recursos	0 €
TOTAL	3200 €

Cuadro 3: Tabla resumen de tiempos

7. Riesgos

7.1. Análisis de Riesgos

Los riesgos son situaciones, que sin una debida preparación, pueden resultar perjudiciales para el desarrollo de un proyecto [14].

El plan de riesgos se realiza generalmente de la misma manera:

1. **Identificar los Riesgos**
2. **Priorizar los Riesgos**
3. **Realizar un Plan de Riesgos**
4. **Monitorizar los Riesgos**

A continuación se describirán algunos de los posibles riesgos identificados, así como su probabilidad (probabilidad de que el riesgo ocurra), su impacto (nivel de daño que el riesgo podría causar si se manifiesta),reducción del riesgo (métodos usados para evitar que el riesgo se manifieste) y plan de contingencia (medida si el riesgo llega a manifestarse).

Mucho Tiempo de Aprendizaje	
Descripción	Se emplea demasiado tiempo en aprender a usar el software necesario
Probabilidad	Baja
Impacto	Medio
Reducción de Riesgo	Se conoce con anterioridad las bases del software usado
Plan de Contingencia	Se emplearían más horas para aprender lo necesario

Cuadro 4: Tabla riesgo tiempo de aprendizaje elevado

Enfermedad por Coronavirus	
Descripción	En la actual época de coronavirus sería posible contraer la enfermedad junto con otros riesgos de infección
Probabilidad	Baja
Impacto	Bajo
Reducción de Riesgo	El alumno está vacunado
Plan de Contingencia	Se acepta el riesgo

Cuadro 5: Tabla riesgo enfermedad por coronavirus

Fallo en el Medio de Trabajo	
Descripción	Un corte de luz, fallo en el software utilizado o una avería en el ordenador son posibles riesgos
Probabilidad	Baja
Impacto	Alto
Reducción de Riesgo	Se realizan numerosos backups
Plan de Contingencia	Se posee otro equipo desde el que trabajar y se procedería a reparar dicho error

Cuadro 6: Tabla riesgo fallo en el medio de trabajo

Atraso en el Cumplimiento de los Plazos	
Descripción	El atraso debido a cualquier imprevisto podría originar un atraso en la entrega de los Sprints
Probabilidad	Alta
Impacto	Medio
Reducción de Riesgo	La intensidad en los primeros Sprints hace posible que el margen de tiempo disponible sea más amplio
Plan de Contingencia	Se emplearían otros horarios para retomar el tiempo perdido

Cuadro 7: Tabla riesgo atraso en los plazos de entrega

Desmotivación a la Hora de Trabajar	
Descripción	El paso del tiempo puede originar cambios emocionales que pueden causar problemas en el desarrollo
Probabilidad	Media
Impacto	Alto
Reducción de Riesgo	Realizar el trabajo lo antes posible
Plan de Contingencia	Se realizarían descansos y pausas en el trabajo

Cuadro 8: Tabla riesgo desmotivación

Cambio en los Requisitos Expecificados	
Descripción	Es posible que se realicen cambios, añadiendo eliminando requisitos
Probabilidad	Alta
Impacto	Bajo
Reducción de Riesgo	La metodología ágil permite una gran flexibilidad frente al cambio de requisitos
Plan de Contingencia	Se realizarían dichos cambios con la mayor antelación posible junto con un cambio en la planificación

Cuadro 9: Tabla riesgo cambio de requisitos

8. Planificación Final

8.1. Seguimiento Actual del Proyecto

En esta sección se especificarán las horas invertidas aproximadamente en el desarrollo de cada *Sprint* así como si se cumplieron los plazos de entrega, y los desarrollos propuestos para la misma.

Para un mejor entendimiento se tratarán los siguientes temas:

- **Investigación:** Trata todo lo relacionado con el aprendizaje o la búsqueda de información/recursos necesaria para el desarrollo del proyecto.
- **Código:** Se tendrá en cuenta el tiempo empleado en la programación necesaria.
- **Estilo:** Tiempo empleado en el desarrollo de los mapeados y estructuras necesarias dentro de Unity 3D.
- **GDD:** Tiempo empleado en la redacción del GDD de cada juego.
- **Solución de Errores:** Tiempo empleado en la solución de posibles *BUGS* o contratiempos sufridos.
- **Memoria:** Tiempo empleado en la redacción del documento TFG.

8.1.1. Sprint 1

Durante este *Sprint*, se comenzó con el desarrollo del proyecto, esto incluía la instalación de todo el software necesario y la obtención de conocimiento sobre este. La duración del Sprint fue de una semana en la que se realizó el menú principal desde el cual se accede al juego y al menú de opciones y se comenzó con el desarrollo del mapeado del mundo abierto.

Se empezó con el desarrollo del GDD correspondiente al juego de *Strings* y se realizaron los primeros esquemas para el documento TFG.

Por último, se implementó el correspondiente código necesario para el movimiento del personaje en el mundo abierto y se realizó la investigación sobre las distintas páginas que serían utilizadas posteriormente para la obtención de los distintos recursos.

De esta manera el *Sprint* fue completado en las fechas previstas con un **total éxito**.

Resumen Sprint 1	
Fechas	08/02/22 a 15/02/22
Investigación	5 horas
Código	8 horas
Estilo	10 horas
GDD	6 horas
Solución de Errores	2 horas
Memoria	1 horas
TOTAL	32 horas

Cuadro 10: Sprint 1. Resumen de horas

8.1.2. Sprint 2

Durante este *Sprint* de una semana de duración, se terminó el desarrollo principal del mundo abierto, esto incluía todo el mapeado y los diferentes prefabs utilizados para dar ambientación. A su vez se terminó el menú de opciones totalmente funcional desde el menú de inicio.

Se finalizó el GDD del juego de *Strings*, aunque quedó pendiente una modificación para el añadido de algunas imágenes.

Se solucionaron algunos errores respecto al movimiento del personaje con el componente *Character Controller* y se añadieron texturas y animaciones obtenidas de *Mixamo*.

De esta manera el *Sprint* fue completado en las fechas previstas con un **éxito parcial** ya que no se invirtieron horas en el desarrollo del documento TFG.

Resumen Sprint 2	
Fechas	15/02/22 a 22/02/22
Investigación	2 horas
Código	3 horas
Estilo	9 horas
GDD	5 horas
Solución de Errores	5 horas
Memoria	0 horas
TOTAL	24 horas

Cuadro 11: Sprint 2. Resumen de horas

8.1.3. Sprint 3

Durante este *Sprint* de una semana de duración, se comenzó con el juego de *Strings*, esto incluía el diseño del mapeado y parte del código necesario para el funcionamiento del juego (en concreto el movimiento, arrastre y generación de objetos).

Se realizaron los avances necesarios en el TFG recuperando el tiempo no empleado del *Sprint* anterior.

En el mundo abierto se implemento la posibilidad de parar el juego, con la aparición de un menú de pausa que contenía el menú de opciones anteriormente desarrollado.

De esta manera el *Sprint* fue completado en las fechas previstas con un **total éxito** dado que se realizaron numerosos avances.

Resumen Sprint 3	
Fechas	22/02/22 a 01/03/22
Investigación	1 horas
Código	16 horas
Estilo	10 horas
GDD	0 horas
Solución de Errores	2 horas
Memoria	5 horas
TOTAL	34 horas

Cuadro 12: Sprint 3. Resumen de horas

8.1.4. Sprint 4

Según la planificación inicial este fue el último *Sprint* de una semana de duración. Durante este tiempo se completó el funcionamiento del juego de *Strings*, incluyendo la generación de cadenas de caracteres correctas o incorrectas según la dificultad seleccionada, la perdida de vidas y la suma o resta del temporizador en función del acierto o fallo.

Se avanzó ligeramente en el documento TFG y se comenzó con el GDD del juego de *Arrays*.

En la última reunión de este *Sprint* se añadió la posibilidad de que el usuario eligiese el juego desde un selector de niveles en el menú de inicio, o desde el propio mundo abierto, proporcionandole así una cierta funcionalidad.

De esta manera el *Sprint* fue completado en las fechas previstas con un **total éxito** dado que se realizaron numerosos avances.

Resumen Sprint 4	
Fechas	01/03/22 a 08/03/22
Investigación	3 horas
Código	15 horas
Estilo	4 horas
GDD	6 horas
Solución de Errores	1 horas
Memoria	3 horas
TOTAL	32 horas

Cuadro 13: Sprint 4. Resumen de horas

8.1.5. Sprint 5

Este fue el primer *Sprint* de 3 semanas de duración según la planificación. Durante este tiempo se incorporaron la posibilidad de seleccionar dificultades en el mundo abierto y en el selector de niveles. También se solucionaron algunos errores en el juego de *Strings* que hacían que en ocasiones las cadenas de texto desaparecieran.

Se terminó también el juego de los *Arrays* unidimensionales, el cual tendría una funcionalidad similar al juego de *Arrays* bidimensionales. Tras terminar ambos juegos se añadió la posibilidad de que se parar los juegos junto con un menú de pausa.

En los documentos se finalizó el GDD del juego de *Arrays* unidimensionales y se inició el del juego de *Arrays* bidimensionales, ambos juegos tendrían el mismo GDD, por lo que solo habría que añadir lo necesario.

El resultado del *Sprint* fue de un **total éxito** dado que se terminó un juego al completo, con una funcionalidad parecida a la del siguiente juego por lo que este sería más fácil de implementar.

Resumen Sprint 5	
Fechas	08/03/22 a 29/03/22
Investigación	3 horas
Código	9 horas
Estilo	5 horas
GDD	7 horas
Solución de Errores	3 horas
Memoria	6 horas
TOTAL	33 horas

Cuadro 14: Sprint 5. Resumen de horas

8.1.6. Sprint 6

Este fue el segundo *Sprint* de 3 semanas de duración según la planificación. Durante este *Sprint* se empezaron a notar los primeros indicios de fatiga mental a la hora de trabajar, que ya habían sido previstos según el riesgo 8. Además de esto durante este *Sprint* tuvo lugar el puente de *Semana Santa* por lo que no se dedicó demasiado tiempo al trabajo, esto dio lugar a que no se pudiera cumplir con la planificación prevista y que se tuviese que modificar, resultando en un total de 7 *Sprints*.

A pesar de todo lo comentado se completó con éxito el juego de *Arrays* bidimensionales, junto con su GDD. Además, se implementó también el juego final, que consiste en un juego de preguntas, de tal forma que solo quedaba realizar el juego del nivel anterior, que se corresponde con el juego de ficheros.

También se realizaron modificaciones en los tutoriales de los juegos, que ahora, pasarían a contener vídeos que reflejaran mejor el objetivo del juego.

Dejando de lado la falta de exactitud con la planificación inicial, el resultado del *Sprint* fue un **total éxito** dado que se completaron dos juegos junto con sus GDDs y tan solo quedaba realizar un juego que, tras acordarlo en la reunión final del *Sprint*, tendría un carácter especial.

Resumen Sprint 6	
Fechas	29/03/22 a 26/04/22
Investigación	4 horas
Código	16 horas
Estilo	11 horas
GDD	6 horas
Solución de Errores	3 horas
Memoria	2 horas
TOTAL	43 horas

Cuadro 15: Sprint 6. Resumen de horas

8.1.7. Sprint 7

Este último *Sprint* tuvo que ser añadido debido a la falta de correspondencia del *Sprint* anterior con la planificación inicial, debido a las razones ya comentadas. Durante este *Sprint* se completó el juego final, dado que faltaba añadir los tutoriales necesarios. También se finalizó el juego del nivel anterior que trata el tema de ficheros. Este juego sería en 2D y se buscaba que representase visualmente el funcionamiento de los ficheros secuenciales al realizar distintas acciones sobre ellos. A su vez se completó el GDD de dicho juego, y se realizaron pequeñas modificaciones en los anteriores GDDs para ajustarlos a los distintos modelos.

También, por petición de los tutores, se realizó un tutorial general en la pantalla de inicio el cual explicara como ganar en los juegos y obtener la puntuación máxima necesaria en la plataforma donde estos se alojan.

Se realizó la integración con la plataforma la cual llevó algo más de tiempo debido al gran tamaño del juego, y las diferencias en el *build de WebGL* en la versión actual de *Unity*.

Para terminar, se realizó el mantenimiento necesario en los juegos de la plataforma dados por los tutores. Estas modificaciones fueron muy cortas por lo que no llevaron más de una hora.

Este último *Sprint* fue un **completo éxito** dado que se consiguió terminar el juego al completo (a falta de posibles añadidos futuros) y la integración con la plataforma.

Resumen Sprint 7	
Fechas	26/03/22 a 03/06/22
Investigación	1 horas
Código	20 horas
Estilo	6 horas
GDD	7 horas
Solución de Errores	6 horas
Memoria	12 horas
TOTAL	52 horas

Cuadro 16: Sprint 7. Resumen de horas

9. GDD General

El juego consiste en un operario, que tiene como objetivo reparar varias máquinas de una fábrica. Para conseguirlo, el operario tendrá que resolver una serie de juegos relacionados con cada una de las estructuras de datos que se pretenden explicar. Con cada máquina reparada (juego superado) se obtendrá una puntuación hasta completar el juego.

9.1. Descripción

En este apartado se detallará el GDD del juego, este es el *Game Design Document* y ha de contener toda la información relacionada con dicho juego. En primer lugar se expondrán las características que abarcan a todos los distintos modos de juego y en cada modalidad se expondrán las características específicas.

Dentro de la pantalla de inicio se muestran una serie de botones desde los cuales se pueden acceder al *modo en primera persona*, *selector de nivel*, *menú de opciones* y *salir del juego*.

Desde el selector de nivel se permite directamente acceder a cualquier juego si se ha desbloqueado previamente y desde el panel de opciones se permite ajustar las distintas configuraciones tal y como el jugador desee.

El modo primera persona es un sistema incluido para dar algo único al juego, en comparación con los mismos de su clase. Desde este modo el jugador puede controlar a un personaje y moverlo por el mapeado disponible (la fábrica en cuestión con la distinta maquinaria). Dentro de este mapa se pueden encontrar las distintas máquinas desde las cuales se puede acceder a los distintos juegos, y al igual que en el selector de nivel, se permitirá acceder a un juego únicamente si el jugador lo ha desbloqueado con anterioridad. También desde este modo de juego el jugador podrá pausar el juego, accediendo así al menú de pausa desde el cual se podrá configurar las distintas opciones o salir al menú de inicio.

Es decir, en realidad se implementarán varios juegos diferentes (cada máquina se corresponde con uno) y cada uno de ellos requiere de su propio GDD, que se detallará en los siguientes capítulos.

Para evitar redundancias, los elementos comunes a todos los GDDs se detallarán únicamente en esta sección.

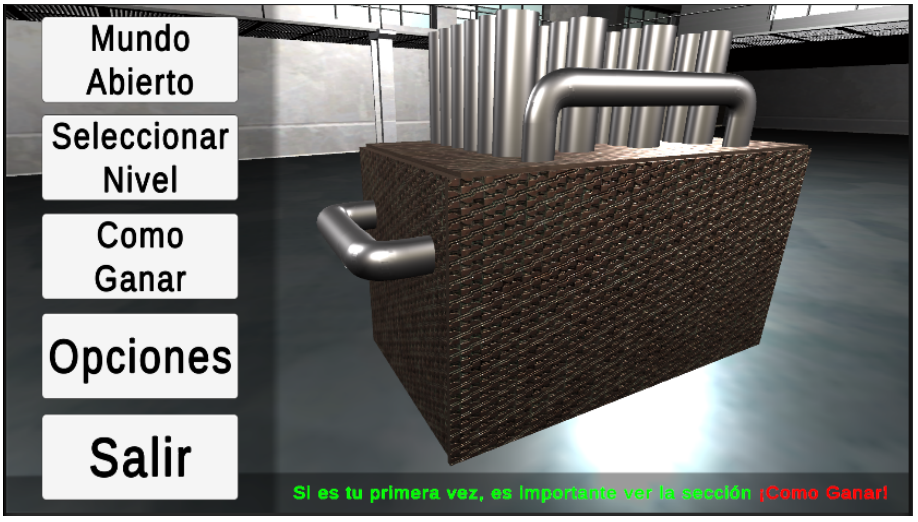

Figura 2: Figura pantalla del menú de inicio

Como se puede apreciar en la Figura 2 aparecen los distintos botones y a la derecha una de las máquinas, estas máquinas aparecen de forma aleatoria cada vez que se accede a esta escena.

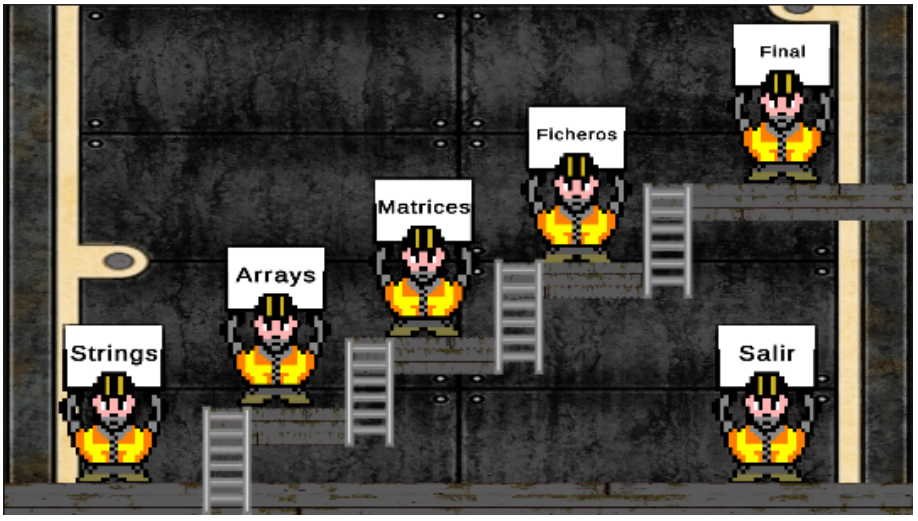

Figura 3: Figura pantalla de selección de nivel

En la Figura 7 muestra el selector de nivel con los distintos botones de acceso a cada juego que aparecen representados por el *sprite* del trabajador. Tras pulsar el botón se accederá al selector de dificultad, el cual aparece representado en la Figura 4.

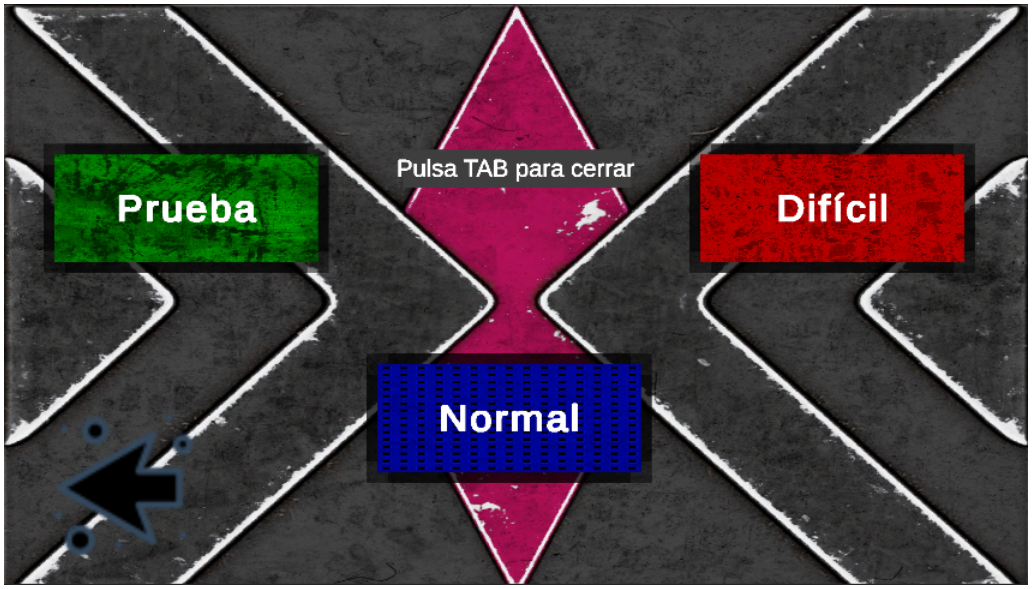

Figura 4: Figura pantalla de dificultades

Figura 5: Figura modo en primera persona

En la Figura 5 se muestra como se ve el mapa desde el modo de primera persona, en dicha figura se pueden apreciar que las distintas máquinas tienen un cartel que indica el nombre del juego, y el panel de acceso a cada juego.

En la Figura 6 se muestra el panel de pausa que se mostrará en cada juego:

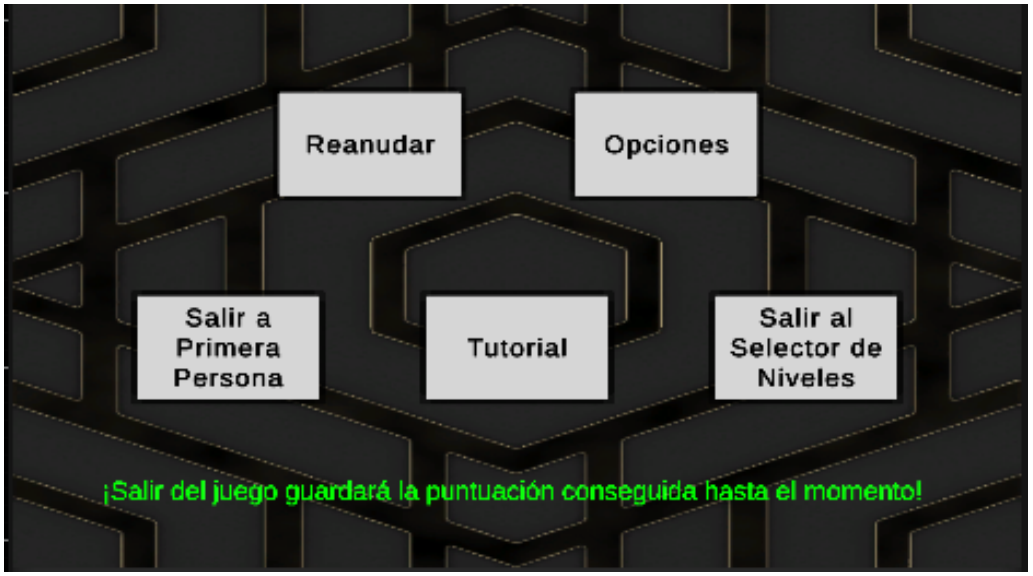

Figura 6: Figura menú de pausa de los juegos

Como se puede apreciar, desde él se podrá acceder a las opciones del juego: salir, (tanto al modo en primera persona, como al selector de nivel) ver el tutorial de cada juego y, por supuesto, reanudar la partida.

En la Figura 7 se puede ver la pantalla de fin del juego:

Figura 7: Figura pantalla de fin de los juegos

9.2. Audiencia

Dado que es un juego de carácter didáctico, la principal audiencia son los estudiantes de los grados de Ingeniería Informática y Estadística, en concreto de la asignatura de primero Fundamentos de Programación. Aunque no se dejará de lado a toda persona que tenga como objetivo aprender y entender los conceptos sobre estas estructuras de datos.

9.3. Plataforma

El juego está pensado para ser jugado en PC por lo tanto estará disponible en sistemas*Windows, Mac y Linux* que tengan acceso a la página web donde se aloja. No se contempla que esté disponible en otras plataformas.

9.4. Mundo

El entorno en el que gira el juego es el de una fábrica, en la cual el operario (el jugador) ha de arreglar ciertas máquinas (los distintos juegos) realizando una serie de actividades. Por ello tienden a predominar los colores grisáceos y apagados propios de una fábrica. Las texturas utilizadas en el desarrollo del juego han sido elegidas para satisfacer estos requisitos.

9.5. Mecánicas

En la Tabla 17 se muestran las distintas mecánicas de las que dispone este juego:

Mecánicas Modo Primera Persona	
Mecánica	**Descripción**
Movimiento del jugador	El jugador podrá mover al personaje utilizando las teclas: W,A,S,D,Espacio (Adelante, Izquierda, Atrás, Derecha y Salto respectivamente). Además se podrá correr pulsando la tecla Shift.
Movimiento de cámara	El jugador podrá mover la cámara usando el ratón
Interactuar con utilizable	El jugador podrá interactuar con los paneles y puertas pulsando la tecla E (aparecerá un mensaje en pantalla)
Pausar el juego	El jugador pausará el juego pulsando la tecla Escape

Cuadro 17: Tabla de mecánicas del modo primera persona

9.6. Recursos

Para este juego se han empleado una serie de *prefabs*, texturas, *sprites* y efectos. Todas las texturas utilizadas, así como los distintos efectos han sido obtenidos de la *Asset Store* de *Unity*. Por el contrario, los *sprites* han sido obtenidos de páginas que proporcionan iconos de forma gratuita [16].

La música utilizada ha sido obtenida de forma gratuita, dado que no contiene *CopyRight*, en la página web *Pixabay* [20].

El listado de *prefabs* es el siguiente:

- **Mapeado Fabrica:** Es el mapa por el cual se puede mover el jugador y ha sido obtenido de forma gratuita desde la página *TurboSquid* [7].
- **Objetos Mapeado 1:** Son algunos de los objetos que se pueden apreciar en el mapeado, no se puede interactuar con ellos y han sido obtenidas de forma gratuita en la *Asset Store* de *Unity* [4].
- **Objetos Mapeado 2:** Son algunos de los objetos que se pueden apreciar en el mapeado, no se puede interactuar con ellos y ha sido obtenido de forma gratuita desde la página *TurboSquid* [7].
- **Máquinas del Juego:** Corresponden a las máquinas desde las cuales se puede acceder a los distintos juegos, han sido creadas en su mayor parte utilizando los objetos que ofrece *Unity* 3D, aunque en algunos casos se han utilizado *prefabs* obtenidos de forma gratuita en *TurboSquid* [7]
- **Personaje:** Corresponde al *prefab* del personaje que mueve el jugador, ha sido obtenido de forma gratuita de la página *Mixamo* [6]. Las distintas animaciones del movimiento del personaje se han obtenido de la misma página.

9.7. Análisis del Juego

9.7.1. Requisitos del Sistema

En esta sección se comentarán los distintos requisitos funcionales y no funcionales que tiene el juego.

ID	Nombre	Descripción
RF-01	Mostrar Menú de Inicio	El sistema deberá mostrar el menú de inicio con todos los botones y recursos necesarios cuando se abra el juego
RF-02	Cambiar la Máquina en el Menú de Inicio	El sistema deberá de cambiar la máquina que se muestra en el menú de incio cada vez que se entre
RF-03	Iniciar Juego en Primera Persona	El sistema deberá iniciar el juego en primera persona cuando se pulse el botón necesario desde el menú de inicio
RF-04	Mostrar Selector de Nivel	El sistema deberá mostrar la pantalla de selección de nivel cuando se pulse el botón necesario desde el menú de inicio
RF-05	Iniciar Juego Desde Selector de Nivel	El sistema deberá iniciar el juego seleccionado desde el selector de nivel
RF-06	Mostrar Menú de Opciones	El sistema deberá mostrar el menú de opciones cuando se pulse el botón necesario desde el menú de inicio
RF-07	Mostrar el Tutorial General	El sistema deberá mostrar el tutorial en el que se explique el funcionamiento general del juego
RF-08	Salir del Juego	El sistema deberá cerrar el juego cuando se pulse el botón necesario desde el menú de inicio
RF-09	Avanzar, Retroceder y Salir del Tutorial	El sistema deberá permitir avanzar y retroceder en las pantallas del tutorial, además de que tendrá que permitir salir de este
RF-10	Mover Personaje	El sistema deberá permitir el movimiento del personaje cuando se pulsen las teclas: W,A,S,D o Espacio

RF-11	Utilizar Panel/Puerta	El sistema deberá permitir la utilización de los Paneles/Puertas cuando se pulse la tecla E
RF-12	Mostrar Panel de Dificultades	El sistema deberá mostrar el panel de dificultades tras seleccionar un juego, ya sea desde el selector de niveles o desde un panel en el modo primera persona
RF-13	Iniciar MiniJuego	El sistema deberá iniciar el juego elegido tras seleccionar la dificultad
RF-14	Puntuaciones Necesarias	El sistema deberá prohibir el acceso a los juegos siguientes hasta que no se haya obtenido cierta puntuación en el juego anterior
RF-15	Abrir Menú Pausa	El sistema deberá pausar el juego mostrando el menú de pausa cuando se pulse la tecla Esc
RF-16	Mensaje Utilizable	El sistema deberá mostrar un mensaje al acercarse a un objeto utilizable
RNF-01	Juego con Unity	El juego deberá realizarse con el motor de desarrollo Unity
RNF-02	Juego en la Web	El juego deberá tener formato WebGL para poder usarse en la plataforma web

Cuadro 18: Tabla de requisitos del menú de inicio y modo primera persona

ID	Nombre	Descripción
RF-01	Reanudar Juego	El sistema deberá permitir reanudar el juego pulsando el botón correspondiente
RF-02	Salir a Menú de Inicio	El sistema deberá permitir salir al menú de incio pulsando el botón correspondiente
RF-03	Menú de Opciones	El sistema deberá permitir mostrar el menú de opciones pulsando el botón correspondiente
RF-04	Salir de Menú de Opciones	El sistema deberá permitir salir del menú de opciones pulsando el botón correspondiente
RF-05	Modificar Opciones	El sistema deberá permitir modificar la resolución, la pantalla completa, el volumen y el brillo desde el menú de opciones

Cuadro 19: Tabla de requisitos del sistema del menúu de pausa del modo primera persona

ID	Nombre	Descripción
RF-01	Reanudar Juego	El sistema deberá permitir reanudar el juego pulsando el botón correspondiente
RF-02	Salir a Mundo Abierto	El sistema deberá permitir salir al modo de mundo abierto pulsando el botón correspondiente
RF-03	Salir a Selector de Nivel	El sistema deberá permitir salir al selector de nivel pulsando el botón correspondiente
RF-04	Menú de Opciones	El sistema deberá permitir mostrar el menú de opciones pulsando el botón correspondiente
RF-05	Mostrar Tutorial	El sistema deberá permitir mostrar el tutorial pulsando el botón correspondiente
RF-06	Salir del Tutorial y Opciones	El sistema deberá permitir salir del tutorial y el menú de opciones pulsando el botón correspondiente
RF-07	Modificar Opciones	El sistema deberá permitir modificar la resolución, la pantalla completa, el volumen y el brillo desde el menú de opciones
RF-08	Avanzar o Retroceder en el Tutorial	El sistema deberá permitir avanzar o retroceder las distintas pantallas del tutorial pulsando a los botonces correspondientes

Cuadro 20: Tabla requisitos del sistema del menú de pausa en el juego de *Strings*

9.7.2. Casos de Uso

En esta sección se tratarán los distintos casos de uso que tiene el juego.

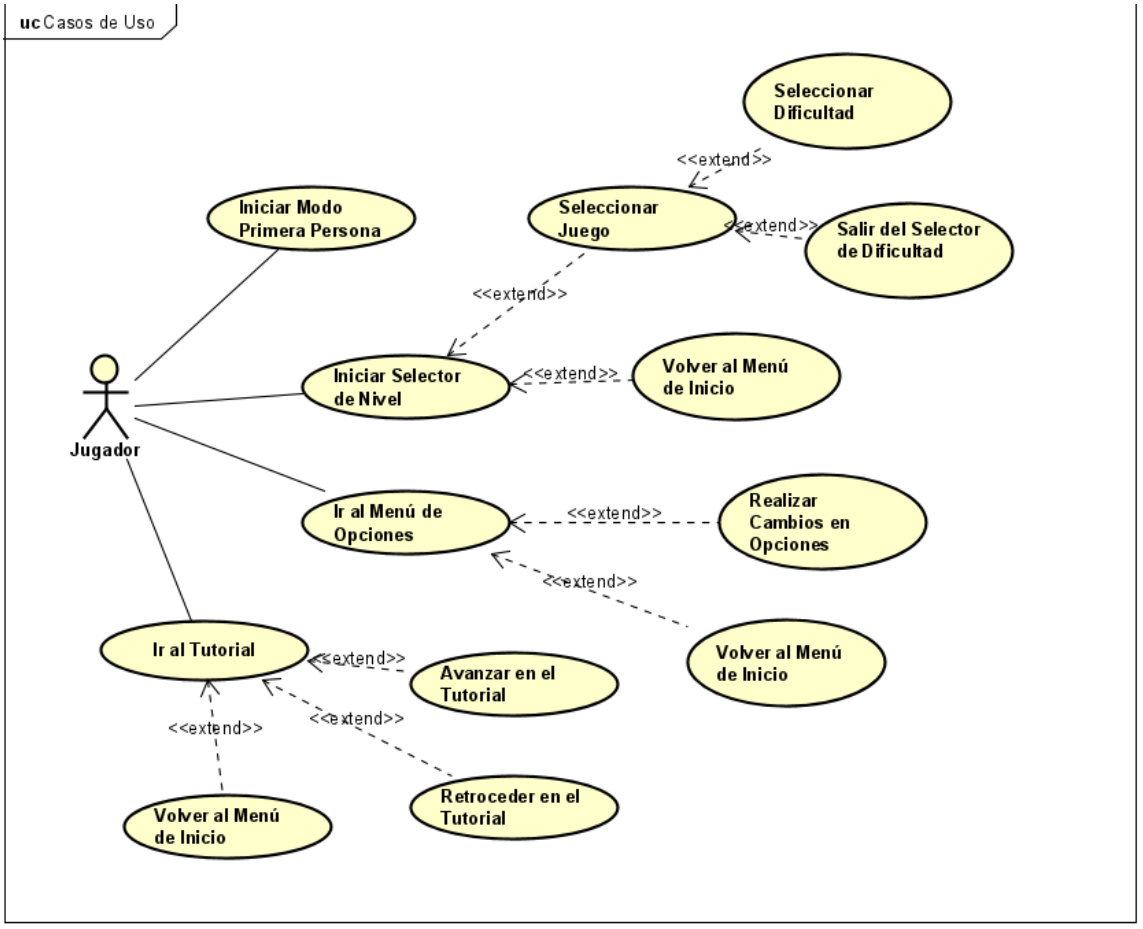

Figura 8: Figura casos de uso del menú de inicio

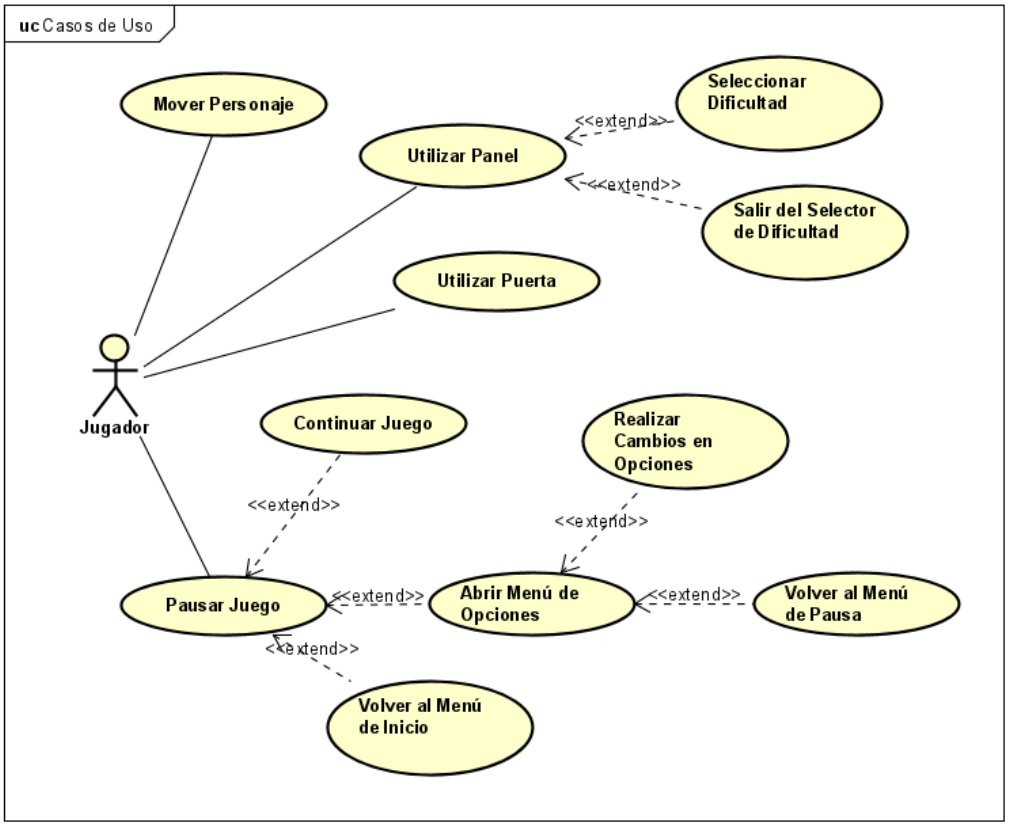

Figura 9: Figura casos de uso del modo primera persona

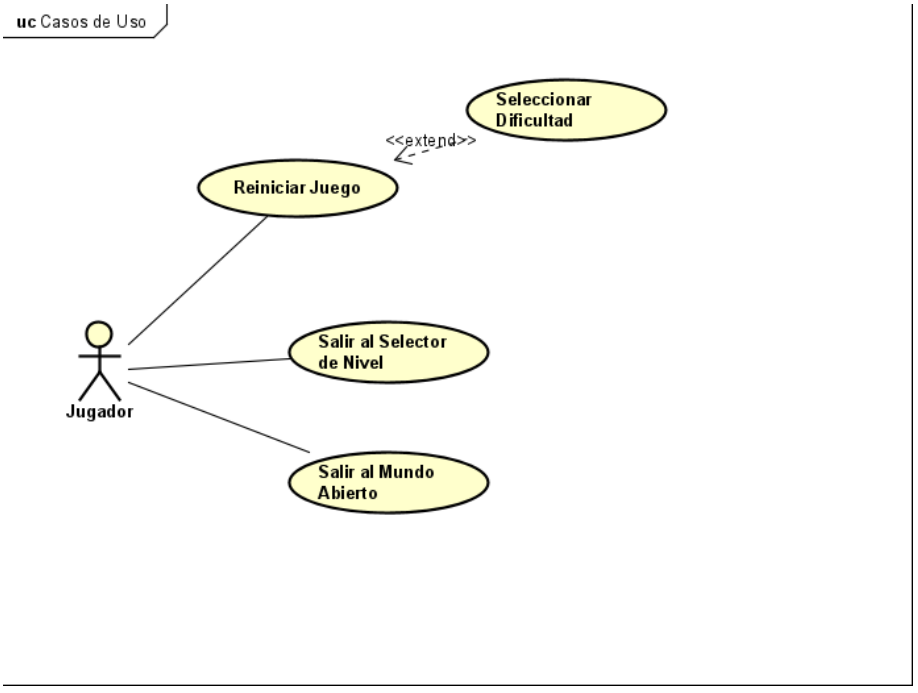

Figura 10: Figura casos de uso del fin del juego

9.7.3. Modelo de Dominio

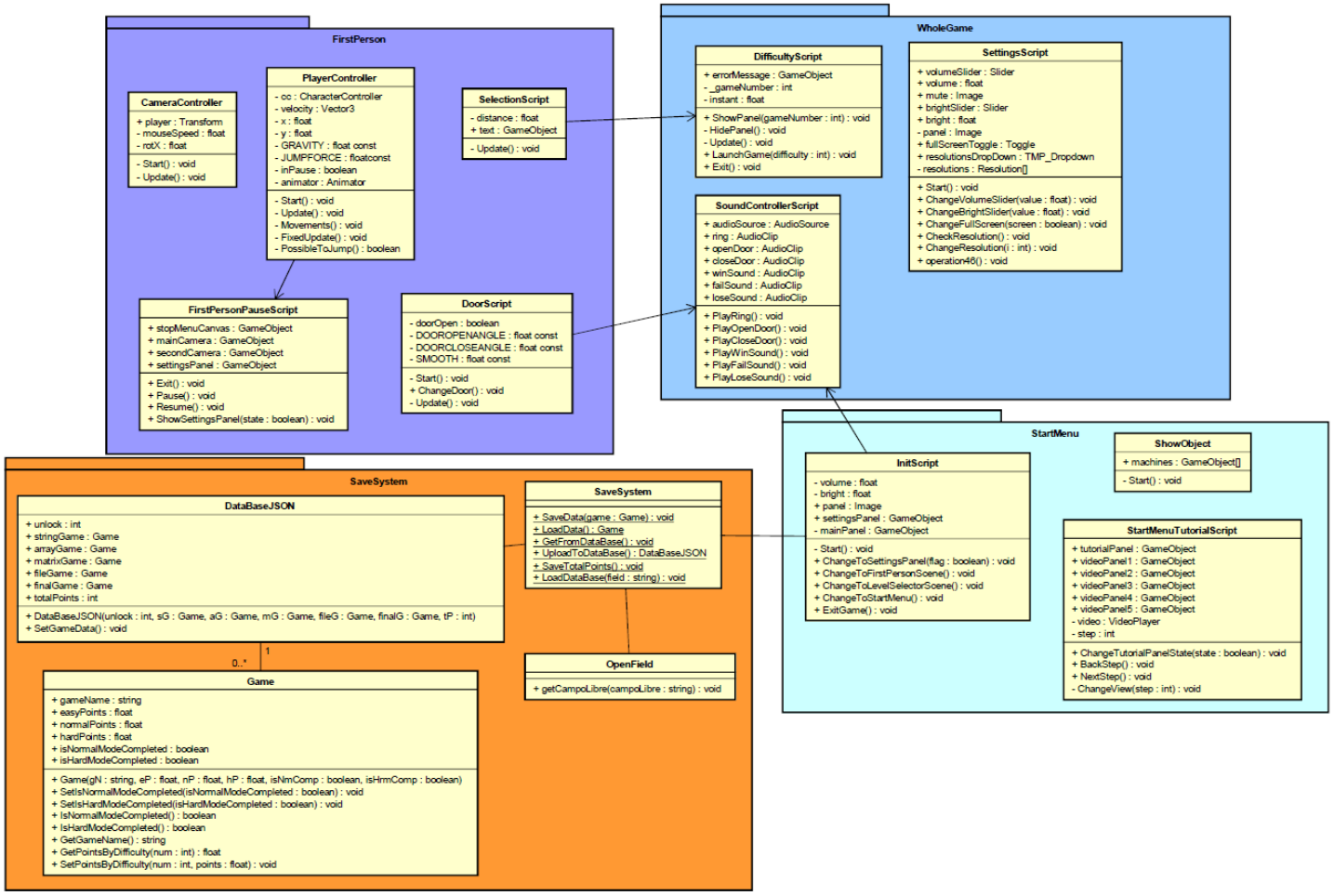

Figura 11: Figura modelo de dominio del menú de inicio y el modo primera persona

Como se indica en el nombre de las clases se ha empleado el patrón *Vista - Controlador*, similar al MVC solo que en este caso no existen clases de *Modelo*. Con el se gestionan las distintas puntuaciones o comportamientos de las clases que luego se verán reflejados en la pantalla.

Además, en ocasiones se ha empleado el patrón *Singleton* para ahorrar recursos mediante la creación de clases.

Como se puede ver en la Figura 11, en la carpeta *FirstPerson* se encuentran las clases utilizadas en el modo primera persona y en la carpeta *WholeGame* se encuentran las clases utilizadas en todo el programa, mientras que en la carpeta *StartMenu* se encuentran las clases esplícitas del menú de inicio.

Las clases *CameraController* y *PlayerController* se usan para el movimiento del personaje. Las clases *SelectionScript* y *DoorScript* se usan para interactuar con los distintos utilizables y para el moviento de las puertas. La case *FirstPersonPauseScript* gestiona el acceso al menú de pausa.

En la carpeta *StartMenu* la clase *InitScript* obtiene de la base de datos mediante el uso de las clases *SaveSystem* y *OpenField* un JSON, y lo transforma a un objeto *DataBaseJSON* instanciando todos los datos del juego. En esa misma carpeta, la clase *ShowObject* se encarga de gestionar como se muestran los objetos en el menú de inicio y la clase *StartMenuTutorialScript* se encarga de gestionar los tutoriales mostrados desde el menú de inicio.

En la carpeta *WholeGame* se encuentran la clase *DifficultyScript* que se encarga del acceso a los juegos según la dificultad seleccionada y la case *SettingsScript* que se encarga de la gestión del menú de opciones.

Por último, la clase *MusicScript* es la encargada de la reproducción de las distintas músicas o sonidos.

9.8. Pruebas en la Plataforma

En esta sección se enumerarán las distintas pruebas llevadas a cabo en la plataforma donde se alojarán los juegos.

Prueba - 01	
Nombre	Ir al panel de opciones
Descripción	Deberá aparecer el panel de opciones al pulsar en el respectivo botón
Resultado	Correcto

Cuadro 21: Descripción prueba 01

Prueba - 02	
Nombre	Modificar la resolución
Descripción	Se deberá poder modificar la resolución de la pantalla y guardar su información
Resultado	Incorrecto
Motivo	Debido a que el juego se aloja en una plataforma web, no se puede obtener las distintas resoluciones del PC.
Solución	No se ha encontrado solución

Cuadro 22: Descripción prueba 02

Prueba - 03	
Nombre	Pantalla Completa
Descripción	Se deberá pasar a pantalla completa al pulsar el correspondiente botón y guardar su información
Resultado	Correcto

Cuadro 23: Descripción prueba 03

Prueba - 04	
Nombre	Modificar el brillo
Descripción	Se deberá pasar aumentar o reducir el brillo de la pantalla mediante un *slider* y guardar su información
Resultado	Correcto

Cuadro 24: Descripción prueba 04

Prueba - 05	
Nombre	Modificar el volumen
Descripción	Se deberá pasar aumentar o reducir el volumen del juego mediante un *slider* y guardar su información
Resultado	Correcto

Cuadro 25: Descripción prueba 05

Prueba - 06	
Nombre	Volver al menú de inicio
Descripción	Deberá aparecer el menú de inicio tras pulsar el respectivo botón
Resultado	Correcto

Cuadro 26: Descripción prueba 06

Prueba - 07	
Nombre	Mostrar los tutoriales
Descripción	Deberá aparecer el panel de tutoriales tras pulsar el respectivo botón
Resultado	Correcto

Cuadro 27: Descripción prueba 07

Prueba - 08	
Nombre	Avanzar y retroceder tutorial
Descripción	Se deberán intercambiar los distintos vídeos al pulsar en los botones de avanzar y retroceder
Resultado	Correcto

Cuadro 28: Descripción prueba 08

Prueba - 09	
Nombre	Ir al selector de nivel
Descripción	Deberá aparecer la pantalla de seleccionar los distintos niveles al pulsar en el respectivo botón
Resultado	Correcto

Cuadro 29: Descripción prueba 09

Prueba - 10	
Nombre	Mostrar panel de dificultades
Descripción	Deberá aparecer el panel de dificultades al seleccionar cualquier juego
Resultado	Correcto

Cuadro 30: Descripción prueba 10

Prueba - 11	
Nombre	Texto de nivel no disponible
Descripción	Deberá aparecer un texto indicando que hay que haber obtenido cierta puntuación en el nivel anterior
Resultado	Correcto

Cuadro 31: Descripción prueba 11

Prueba - 12	
Nombre	Cerrar el juego
Descripción	Se deberá poder cerrar el juego devolviendo el control a la plataforma
Resultado	Correcto

Cuadro 32: Descripción prueba 12

10. GDD *Strings*

El juego consiste en recoger las cajas que contienen los *Strings* correctos y depositarlos en el contenedor, reparando de esta manera la máquina.

10.1. Descripción

Se trata de un juego de carácter didáctico que tiene como principal objetivo explicar de una manera divertida la formación de las cadenas de caracteres en los lenguajes de programación, conocidas como *Strings*.

El jugador tendrá que seleccionar haciendo click y arrastrar las cadenas de caracteres que forman correctamente un *String* (entendemos como *String* toda cadena compuesta por: *"texto"*, o la concatenación de cadenas de la forma *"texto"* + *"texto"* + *'c'* + *int*), al contenedor que aparece en la parte izquierda de la pantalla. Si una cadena incorrecta llega al final del camino, esta será destruida y el jugador perderá vidas y el tiempo disponible será decrementado.

En la Figura 12 se muestra de forma detallada el funcionamiento del juego:

Figura 12: Figura juego de *Strings*

En la Figura 12 se puede ver como el juego consta de una contenedor a la izquierda, donde se sitúan ciertos iconos que simbolizan el temporizador, número de vidas y la puntuación que el jugador ha conseguido hasta ese momento. Este contenedor a su vez es la zona donde el jugador tendrá que depositar los *Strings* que él considere correctos.

A la derecha se puede ver la zona de cintas transportadoras. Las cajas que contienen los *Strings* irán moviéndose por ellas hasta llegar al final del camino, que es la zona central donde se ve fuego.

Como ya se ha comentado, el jugador deberá agarrar y arrastrar las cadenas de caracteres que considere correctos al contenedor de la izquierda.

10.2. Recursos

Para este juego se han empleado una serie de *prefabs*, texturas, *sprites* y efectos. Todas las texturas utilizadas, así como los distintos efectos han sido obtenidos de la *Asset Store* de *Unity*. Por el contrario los *sprites* han sido obtenidos de páginas que proporcionan iconos de forma gratuita [16].

El listado de *prefabs* es el siguiente:

- **Cintas Transportadoras:** Son las cintas sobre las que se mueven las cajas y han sido creados utilizando las distintas formas que ofrece *Unity*.
- **Contenedores:** Son tanto el recipiente al que hay que mover las cajas como el del final del recorrido y han sido creados utilizando las distintas formas que ofrece *Unity* 3D.
- **Estructura del Juego:** Esta corresponde a la máquina que representa el mapeado del juego, esta ha sido creada utilizando las distintas formas que ofrece *Unity* 3D.
- **Cajas de *Strings*:** Corresponden a las cajas que se mueven sobre las cintas transportadoras y contienen los *Strings*. Han sido creadas utilizando las distintas formas que ofrece *Unity* 3D.

Los vídeos utilizados en los tutoriales han sido grabados con *OBS Studios* [17] y modificados con *VideoPadEditor* [18].

10.3. Mecánicas

En la Tabla 33 se muestran las distintas mecánicas de las que dispone este juego:

Mecánicas Juego de *Strings*	
Mecánica	**Descripción**
Dificultad seleccionada	La dificultad afecta al número de vidas del jugador la velocidad de las cadenas y su frecuencia de aparición
Jugador deposita una cadena correcta	Se suma 100 al contador de puntos y una cantidad de segundos al temporizador (la cantidad podría ser modificada en el futuro)
Jugador deposita una cadena erronea	Se disminuye una vida del jugador y un tiempo determinado
Cadena correcta llega al final del recorrido	Se disminuye una vida del jugador y un tiempo determinado
Cadena incorrecta llega al final del recorrido	Se aumenta una cantidad de tiempo el temporizador (la cantidad podría ser modificada en un futuro)
Caja es arrastrada hacia los bordes del mapa	La caja desaparece sin alterar los marcadores
Vidas llegan a 0	Fin del juego mostrando la puntuación obtenida, la máxima puntuación y el número de vidas.
Tiempo llega a 0	Fin del juego mostrando la puntuación obtenida, la máxima puntuación y el número de vidas.
Pausar la partida	Se parará el tiempo y se mostrará el menú de pausa con todas las opciones disponibles.
Salir del juego	Se mostrará el menú de fin de juego indicando todas las puntuaciones.

Cuadro 33: Tabla de mecánicas del juego de *Strings*

10.4. Análisis del Juego

10.4.1. Requisitos del Sistema

En esta sección se comentarán los distintos requisitos funcionales y no funcionales que tiene el juego.

ID	Nombre	Descripción
RF-01	Generar *Strings* Aleatorios	El sistema deberá generar los *Strings* que se mostrarán en las cajas según la dificultad seleccionada
RF-02	Mostrar Indicadores	El sistema deberá mostrar al iniciar el juego los distintos indicadores: puntuación, temporizador y vidas
RF-03	Mostrar Efectos	El sistema deberá mostrar los distintos efectos en el mapa
RF-04	Mover Caja	El sistema deberá mover las cajas por las cintas transportadoras
RF-05	Agarrar Caja	El sistema deberá permitir agarrar una caja
RF-06	Arrastrar Caja	El sistema deberá permitir arrastrar una caja agarrada
RF-07	Mostrar Caja	El sistema deberá mostrar las cajas que contendrán los *Strings*
RF-08	Sumar Puntos	El sistema deberá sumar cierta puntuación a su respectivo contador si se deposita una caja correcta en la zona correcta
RF-09	Eliminar Caja	El sistema deberá eliminar una caja si esta toca los bordes de la pantalla, cae al suelo,llega al final del camino, cae en la zona correcta o pasa un tiempo determinado
RF-10	Sumar Tiempo	El sistema deberá sumar cierto tiempo a su respectivo contador si se deposita una caja correcta en la zona correcta o una caja incorrecta llega al final del camino

RF-11	Restar Tiempo	El sistema deberá restar cierto tiempo a su respectivo contador si se deposita una caja incorrecta en la zona correcta, o una caja correcta llega al final
RF-12	Restar Vidas	El sistema deberá restar una vida al jugador cuando se deposite una caja incorrecta o una caja correcta llegue al final del camino
RF-13	Temporizador	El sistema deberá decrementar el número del tiempo como si de un temporizador se tratara
RF-14	Pausar Juego	El sistema deberá permitir al jugador pausar el juego mostrando así el menú de pausa

Cuadro 34: Tabla requisitos del sistema del juego de *Strings*

10.4.2. Casos de Uso

En esta sección se tratarán los distintos casos de uso que tiene el juego.

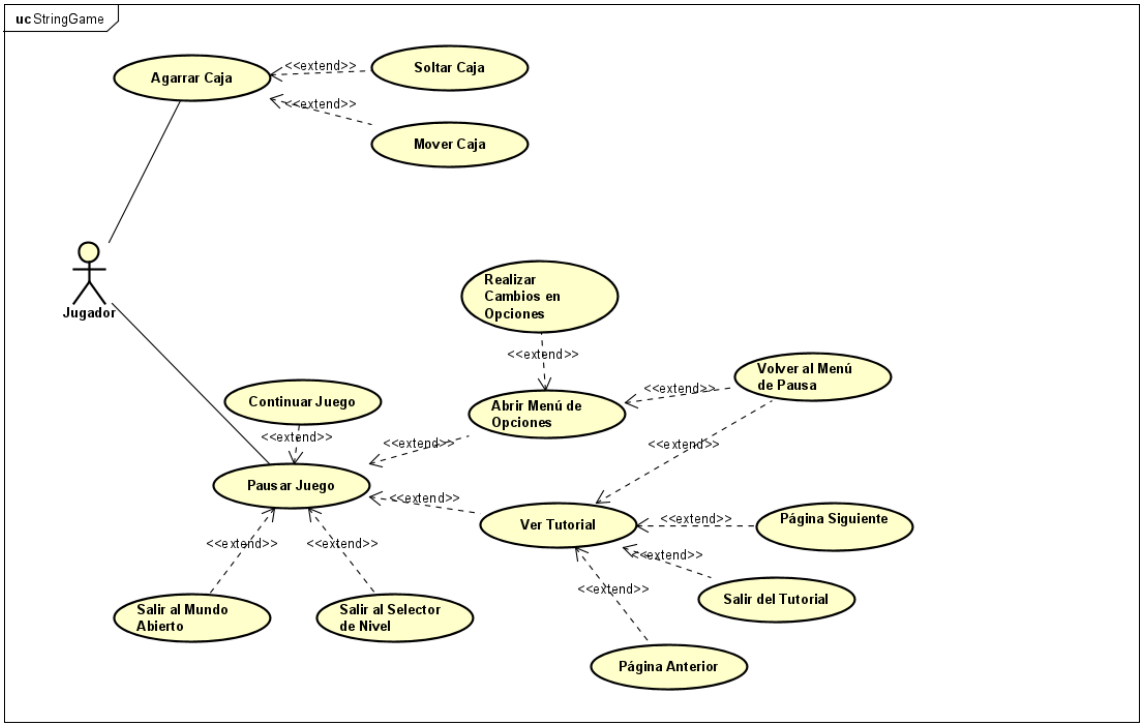

Figura 13: Figura casos de uso del juego de *Strings*

10.4.3. Modelo de Dominio

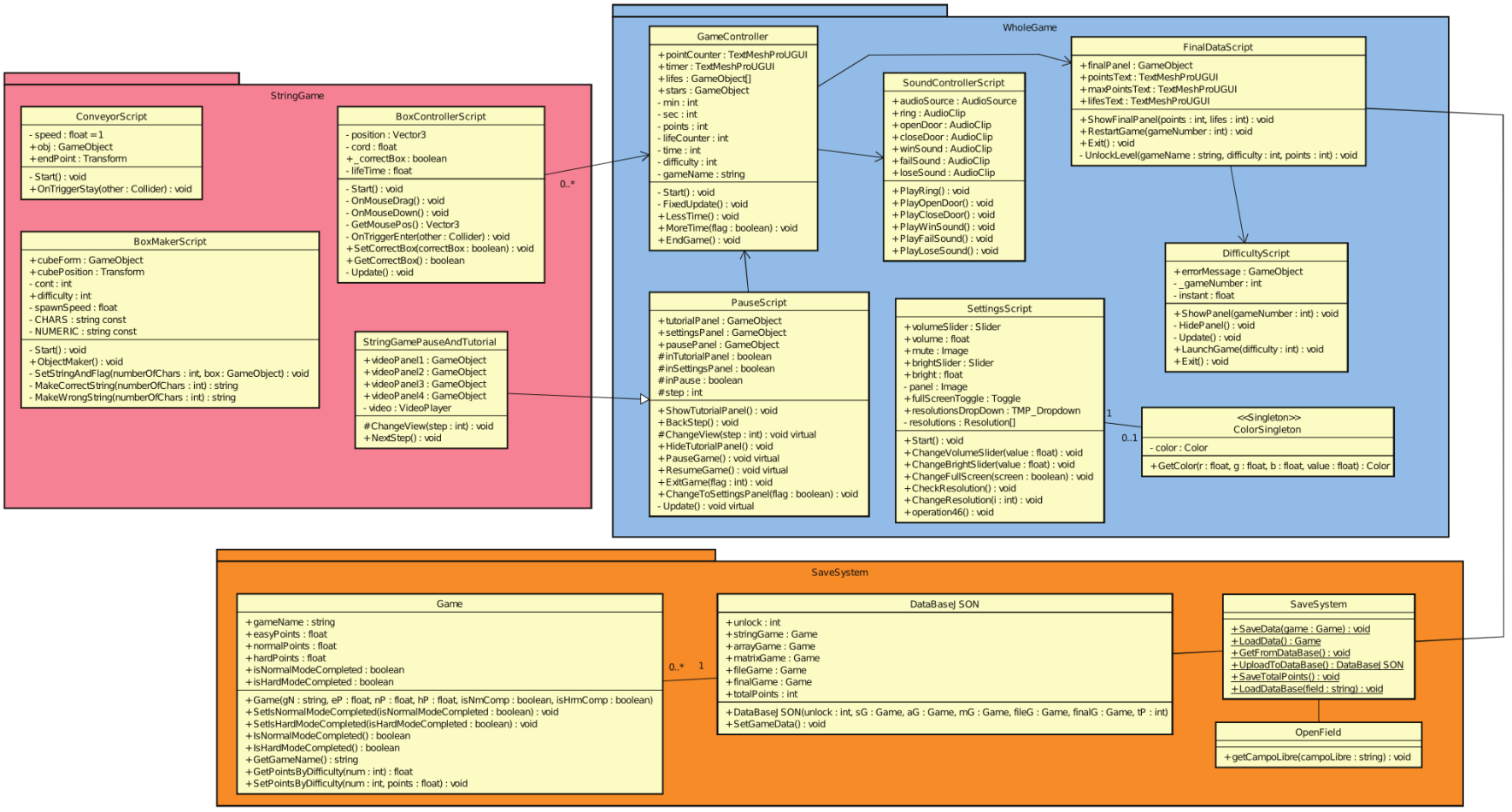

Figura 14: Figura modelo de dominio del juego de *Strings*

Tal y como se muestra en la Figura 17, la clase *GameController* se encarga de la gestión de puntuaciones y del estado del juego, mientras que la clase *BoxMakerScript* representa el controlador interno del juego, que se encarga de la creación de *Strings* según la dificultad seleccionada, estos *Strings* serán creados en función de ciertas probabilidades por lo que ninguna partida ni ningún *Strings* será igual al anterior, fomentando así la diversidad en el juego. La clase *BoxControllerScript* se encarga del control de las cajas, es decir su movimiento, aparición y desaparición. También se encarga de detectar si la cadena es correcta o incorrecta y avisar al *GameController* para que modifique la puntuación. La clase *CoveyorScript* se encarga del movimiento de las cajas.

Tras finalizar el juego la clase *FinalDataScript* avisa a la clase estática *SaveSystem* la cual creará un *Strings* en formato JSON, se modificaran las respectivas *PlayerPrefs* desde las cuales se obtienen los datos de cada partida y se dará el aviso a la clase *OpenField* la cual subirá dicho JSON a la base de datos, además de modificar la puntuación máxima conseguida si esta ha sido superada.

10.5. Pruebas en la Plataforma

En esta sección se enumerarán las distintas pruebas llevadas a cabo en la plataforma donde se alojarán los juegos.

Prueba - 01	
Nombre	Iniciar juego
Descripción	Se deberá iniciar el juego tras haber elegido la dificultad
Resultado	Correcto

Cuadro 35: Descripción prueba 01

Prueba - 02	
Nombre	Acceder al menú de pausa
Descripción	Deberá aparecer el menú de pausa tras pulsar TAB parando el tiempo
Resultado	Correcto

Cuadro 36: Descripción prueba 02

Prueba - 03	
Nombre	Reanudar juego
Descripción	Deberá ocultarse el menú de pausa, continuando de esta manera el juego al pulsar *Reanudar* ó TAB
Resultado	Correcto

Cuadro 37: Descripción prueba 03

Prueba - 04	
Nombre	Acceder al panel de opciones
Descripción	Deberá aparecer el panel de opciones al pulsar en *Opciones*
Resultado	Correcto

Cuadro 38: Descripción prueba 04

Prueba - 05	
Nombre	Acceder al tutorial
Descripción	Deberá aparecer el panel de tutorial tras pulsar en *Tutorial*
Resultado	Correcto

Cuadro 39: Descripción prueba 05

Prueba - 06	
Nombre	Finalizar juego
Descripción	Deberá aparecer el panel de fin de juego tras pulsar en *Salir al selector de nivel*, al perder todas las vidas o al llegar el tiempo a 0. Mostrando en este la puntuación conseguida, la puntuación máxima conseguida y las vidas
Resultado	Correcto

Cuadro 40: Descripción prueba 06

Prueba - 07	
Nombre	Volver al jugar
Descripción	Deberá aparecer el selector de dificultad tras pulsar a *Volver a jugar* desde el panel de fin de juego
Resultado	Correcto

Cuadro 41: Descripción prueba 07

Prueba - 08	
Nombre	Salir del juego
Descripción	Se deberá volver al selector de nivel tras pulsar en *Seleccionar Nivel* desde el panel de fin de juego
Resultado	Correcto

Cuadro 42: Descripción prueba 08

Prueba - 09	
Nombre	Agarrar y mover una caja
Descripción	Se deberá de poder agarrar y mover las cajas de *Strings* haciendo click y manteniendo en ellas
Resultado	Correcto

Cuadro 43: Descripción prueba 09

Prueba - 10	
Nombre	Caja correcta suma puntos
Descripción	Al depositar una caja correcta en el contenedor la puntuación y tiempo tendrán que aumentar, desapareciendo a su vez la caja
Resultado	Correcto

Cuadro 44: Descripción prueba 10

Prueba - 11	
Nombre	Caja correcta resta vidas
Descripción	Al llegar una caja correcta al fin del camino deberá disminuirse una vida y el temporizador, desapareciendo a su vez la caja
Resultado	Correcto

Cuadro 45: Descripción prueba 11

Prueba - 12	
Nombre	Caja incorrecta suma tiempo
Descripción	Al llegar una caja incorrecta al final del camino deberá sumarse tiempo al temporizador, desapareciendo a su vez la caja
Resultado	Correcto

Cuadro 46: Descripción prueba 12

Prueba - 13	
Nombre	Caja incorrecta resta vidas
Descripción	Al depositar una caja incorrecta en el contenedor deberá disminuirse una vida y el temporizador, desapareciendo as u vez la caja
Resultado	Correcto

Cuadro 47: Descripción prueba 13

Prueba - 14	
Nombre	Caja desaparece al tocar los bordes
Descripción	Deberán eliminarse las cajas al tocar los bordes del mapa
Resultado	Correcto

Cuadro 48: Descripción prueba 14

Prueba - 15	
Nombre	Caja desaparece al pasar un tiempo
Descripción	Deberán eliminarse las cajas si estas no llegan al final del camino o son depositadas en el contenedor pasado un tiempo
Resultado	Correcto

Cuadro 49: Descripción prueba 15

Prueba - 16	
Nombre	Generación de cadenas
Descripción	Se deberán generar diferentes *Strings* en función de la dificultad seleccionada
Resultado	Correcto

Cuadro 50: Descripción prueba 16

Prueba - 17	
Nombre	Movimiento de las cajas
Descripción	Se deberán desplazar las cajas por las cintas transportadoras hasta llegar al final, a mayor o menor velocidad en función de la dificultad
Resultado	Correcto

Cuadro 51: Descripción prueba 17

Prueba - 18	
Nombre	Puntuación cambia de color
Descripción	La puntuación deberá de cambiar a color rojo y aparecer las estrellas tras alcanzar la puntuación necesaria para completar el nivel
Resultado	Correcto

Cuadro 52: Descripción prueba 18

11. GDD *Arrays*

El juego consiste en introducir una serie de pelotas en distintas tuberías que representan un *Array* unidimensional o bidimensional, según el código que proporciona la máquina para repararla.

11.1. Descripción

Se trata de un juego de carácter didáctico que tiene como principal objetivo el explicar de una forma visual y motivadora el acceso a las distintas posiciones de un *Array* unidimensional y bidimensional.

Para el juego de *Arrays*, el jugador tendrá que seleccionar (haciendo click) y arrastrar una bola que aparecerá en pantalla hasta una serie de tuberías que se corresponden con un *Array* en el que hay una serie de números. También aparecerá a la derecha un código que corresponderá a una de las casillas (si llamamos al *Array* A, un código sería A[1 + A[0]]). El jugador tendrá que arrastrar dicha bola a la casilla correcta indicada por ese código (para el ejemplo anterior la casilla cuyo índice sea el resultado de sumar 1 al contenido de la casilla que ocupe la posición 0 en A).

En el caso del juego de *Arrays* bidimensionales, el sistema es igual, solo que ahora en vez de haber una sola fila (*Array* unidimensional) habrá varias filas (*Array* bidimensional), formando de esta manera una matriz. En la parte derecha aparecerá un código que seguirá el formato de una matriz (si llamamos a la matriz M, un código sería M[0][M[1][2] +1]) y el jugador tendrá que arrastrar la bola a la tubería correcta, indicada por ese código.

En las Figuras 15 y 16 se puede ver de forma detallada el funcionamiento de cada juego:

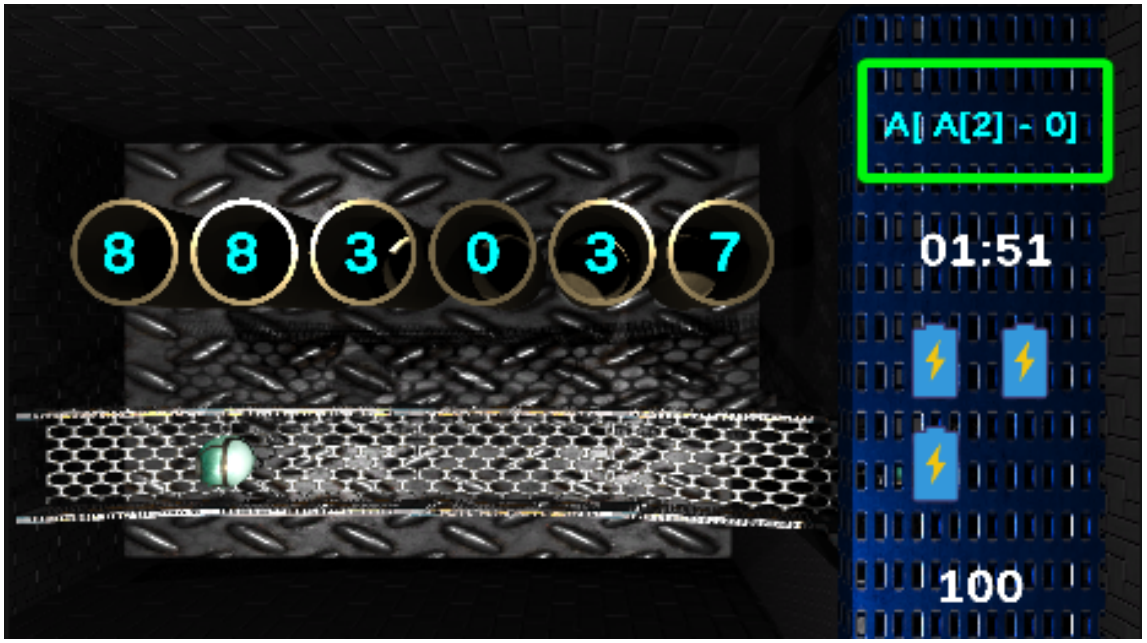

Figura 15: Figura juego de *Arrays* unidimensionales

Figura 16: Figura juego de *Arrays* bidimensionales (matrices)

En la figuras mencionadas, se puede ver como existen una series de tuberías, las cuales representan un *Array*, los números que aparecen sobre ellas representan los valores almacenados en cada posición del *Array*. A la derecha se puede ver el panel de puntuaciones, similar al del anterior juego, con la diferencia de que ahora existe un recuadro verde que contiene el código que indica la posición en el *Array*.

11.2. Recursos

Para este juego se han empleado una serie de *prefabs*, texturas, *sprites* y efectos. Todas las texturas utilizadas, así como los distintos efectos han sido obtenidos de la *Asset Store de Unity*. Por el contrario los *sprites* han sido obtenidos de páginas que proporcionan iconos de forma gratuita [16].

El listado de *prefabs* es el siguiente:

- **Pelota:** Es el principal recurso del juego. Ha sido creada utilizando las formas que ofrece *Unity* y se le ha aplicado una textura obtenida de la *Asset Store*.
- **Tuberías:** Representan la posición del *Array* y han sido obtenidas de *Turbosquid* [7] de forma gratuita. La textura aplicada ha sido obtenida de la *Asset Store*.
- **Estructura del Juego:** Esta corresponde a la máquina que representa el mapeado del juego. Ha sido creada utilizando las distintas formas que ofrece *Unity* 3D.

Los vídeos utilizados en los tutoriales han sido grabados con *OBS Studios* [17] y modificados con *VideoPadEditor* [18].

11.3. Mecánicas

En la Tabla 53 se muestran las distintas mecánicas de las que dispone este juego:

Mecánicas Juego de *Arrays/Matrices*	
Mecánica	**Descripción**
Dificultad seleccionada	La dificultad afecta al número de vidas del jugador y la complejidad del código generado
Jugador deposita la pelota en una posición correcta	Se suma 100 al contador de puntos y una cantidad de segundos al temporizador (la cantidad podría ser modificada en el futuro)
Jugador deposita la pelota en una posición incorrecta	Se disminuye una vida del jugador y un tiempo determinado
Pelota es arrastrada hacia los bordes del mapa	La pelota desaparece sin alterar los marcadores
Pelota cae al suelo	La pelota desaparece sin alterar los marcadores
Vidas llegan a 0	Fin del juego mostrando la puntuación obtenida, la máxima puntuación y el número de vidas.
Tiempo llega a 0	Fin del juego mostrando la puntuación obtenida, la máxima puntuación y el número de vidas.
Pausar la partida	Se parará el tiempo y se mostrará el menú de pausa con todas las opciones disponibles.
Salir del juego	Se mostrará el menú de fin de juego indicando todas las puntuaciones.

Cuadro 53: Tabla de mecánicas del juego de *Arrays/Matrices*

11.4. Análisis del Juego

11.4.1. Requisitos del Sistema

En esta sección se comentarán los distintos requisitos funcionales y no funcionales que tienen los juegos.

ID	Nombre	Descripción
RF-01	Generar Código Aleatorio	El sistema deberá generar el código que representará la posición del array según la dificultad seleccionada
RF-02	Mostrar Indicadores	El sistema deberá mostrar al iniciar el juego los distintos indicadores: puntuación, temporizador y vidas
RF-03	Mostrar Efectos	El sistema deberá mostrar los distintos efectos en el mapa
RF-04	Generar Pelota	El sistema deberá generar una pelota de forma automática cuando esta sea destruida
RF-05	Agarrar Pelota	El sistema deberá permitir agarrar la pelota
RF-06	Arrastrar Pelota	El sistema deberá permitir arrastrar la pelota cuando sea agarrada
RF-07	Eliminar Pelota	El sistema deberá eliminar la pelota si esta toca los bordes de la pantalla, cae al suelo, o es depositada en una de las tuberías
RF-08	Eliminar Pelota	El sistema deberá eliminar la pelota si esta toca los bordes de la pantalla, cae al suelo, o es depositada en una de las tuberías
RF-09	Sumar Tiempo	El sistema deberá sumar cierto tiempo a su respectivo contador si se deposita la pelota en una tubería incorrecta
RF-10	Restar Tiempo	El sistema deberá restar cierto tiempo a su respectivo contador si se deposita la pelota en una tubería incorrecta

RF-11	Temporizador	El sistema deberá decrementar el número del tiempo como si de un temporizador se tratara
RF-12	Pausar Juego	El sistema deberá permitir al jugador pausar el juego mostrando así el menú de pausa
RF-13	Restar Vidas	El sistema deberá restar una vida al jugador cuando la pelota sea introducida en una tubería incorrecta
RF-14	Cambiar Valores	El sistema deberá cambiar los valores que aparecen en las tuberías cada vez que se genere un nuevo código

Cuadro 54: Figura requisitos del sistema del juego de *Arrays/Matrices*

11.4.2. Casos de Uso

En esta sección se tratarán los distintos casos de uso que tiene los juegos.

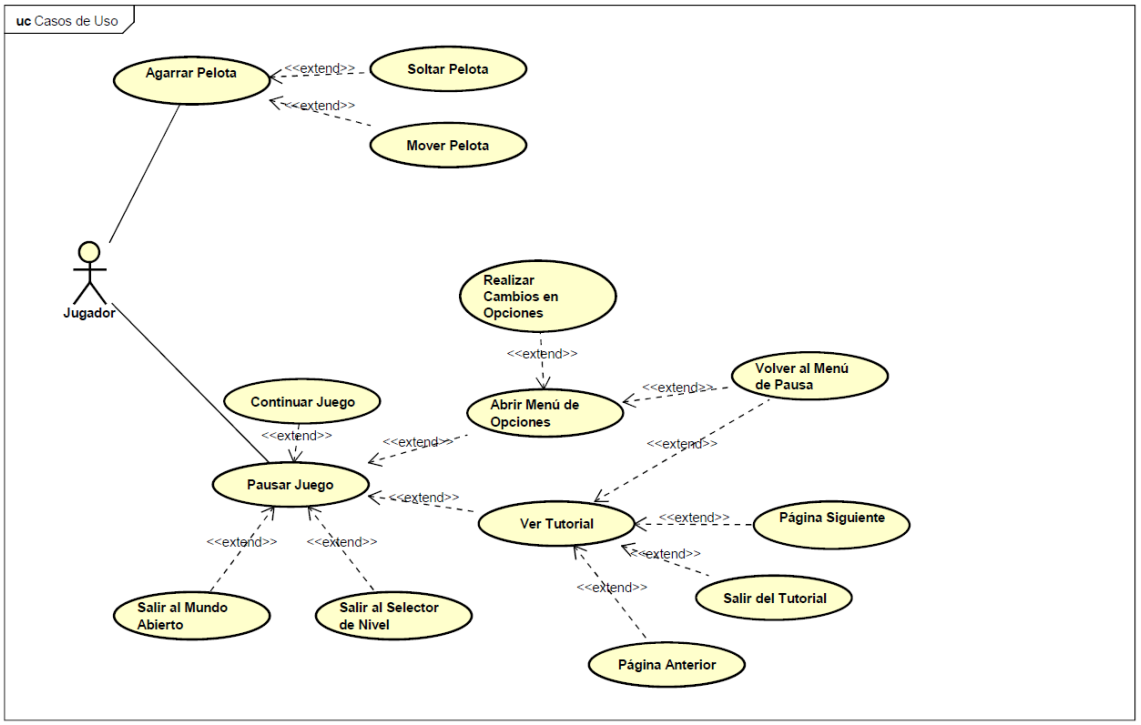

Figura 17: Figura casos de uso del juego de *Arrays/Matrices*

11.4.3. Modelo de Dominio

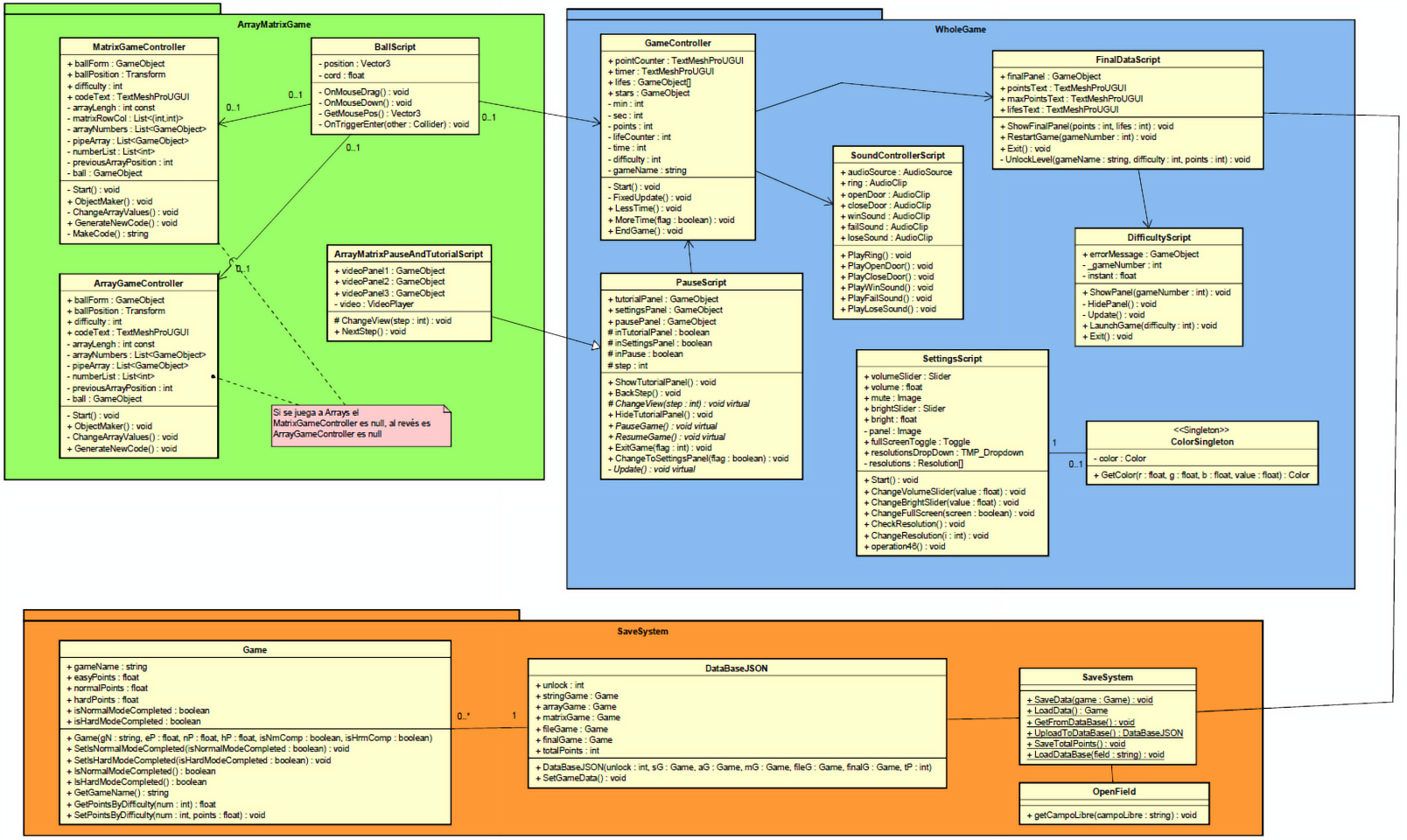

Figura 18: Figura modelo de dominio del juego de *Arrays/Matrices*

Tal y como se detalla en la Figura 18, la clase *GameController* se encarga de la gestión de puntuaciones y del estado del juego, mientras que las clases *ArrayGameController* y *MatrixGameController* representan el controlador interno del juego, que se encarga de la creación de los códigos. La clase *BallScript* se encarga del control de la pelota, es decir su movimiento, aparición y desaparición. También se encarga de detectar si la tubería donde se ha depositado la bola es correcta o incorrecta y avisar al *GameController* para que modifique la puntuación.

Tras finalizar el juego, la clase *FinalDataScript* dará el aviso a la clase estática *SaveSystem* la cual creará un string en formato JSON, se modificaran las respectivas *PlayerPrefs* desde las cuales se obtienen los datos de cada partida y se dará el aviso a la clase *OpenField* la cual subirá dicho JSON a la base de datos, además de modificar la puntuación máxima conseguida si esta ha sido superada.

11.5. Pruebas en la Plataforma

En esta sección se enumerarán las distintas pruebas llevadas a cabo en la plataforma donde se alojarán los juegos.

Prueba - 01	
Nombre	Iniciar juego
Descripción	Se deberá iniciar el juego tras haber elegido la dificultad
Resultado	Correcto

Cuadro 55: Descripción prueba 01

Prueba - 02	
Nombre	Acceder al menú de pausa
Descripción	Deberá aparecer el menú de pausa tras pulsar TAB parando el tiempo
Resultado	Correcto

Cuadro 56: Descripción prueba 02

Prueba - 03	
Nombre	Reanudar juego
Descripción	Deberá ocultarse el menú de pausa, continuando de esta manera el juego al pulsar *Reanudar* ó TAB
Resultado	Correcto

Cuadro 57: Descripción prueba 03

Prueba - 04	
Nombre	Acceder al panel de opciones
Descripción	Deberá aparecer el panel de opciones al pulsar en *Opciones*
Resultado	Correcto

Cuadro 58: Descripción prueba 04

Prueba - 05	
Nombre	Acceder al tutorial
Descripción	Deberá aparecer el panel de tutorial tras pulsar en *Tutorial*
Resultado	Correcto

Cuadro 59: Descripción prueba 05

Prueba - 06	
Nombre	Finalizar juego
Descripción	Deberá aparecer el panel de fin de juego tras pulsar en *Salir al selector de nivel*, al perder todas las vidas o al llegar el tiempo a 0. Mostrando en este la puntuación conseguida, la puntuación máxima conseguida y las vidas
Resultado	Correcto

Cuadro 60: Descripción prueba 06

Prueba - 07	
Nombre	Volver al jugar
Descripción	Deberá aparecer el selector de dificultad tras pulsar a *Volver a jugar* desde el panel de fin de juego
Resultado	Correcto

Cuadro 61: Descripción prueba 07

Prueba - 08	
Nombre	Salir del juego
Descripción	Se deberá volver al selector de nivel tras pulsar en *Seleccionar Nivel* desde el panel de fin de juego
Resultado	Correcto

Cuadro 62: Descripción prueba 08

Prueba - 09	
Nombre	Agarrar y mover una bola
Descripción	Se deberá de poder agarrar y mover la bola haciendo click y manteniendo
Resultado	Correcto

Cuadro 63: Descripción prueba 09

Prueba - 10	
Nombre	Tubería correcta suma puntos
Descripción	Al depositar la bola en la tubería correcta la puntuación y tiempo tendrán que aumentar, desapareciendo a su vez la bola
Resultado	Correcto

Cuadro 64: Descripción prueba 10

Prueba - 11	
Nombre	Tubería incorrecta resta vida
Descripción	Al depositar la bola en una tubería incorrecta se disminuirá una vida y el tiempo, desapareciendo a su vez la bola
Resultado	Correcto

Cuadro 65: Descripción prueba 11

Prueba - 12	
Nombre	Bola desaparece al tocar los bordes
Descripción	Deberán eliminarse la bola al tocar los bordes del mapa
Resultado	Correcto

Cuadro 66: Descripción prueba 12

Prueba - 13	
Nombre	Generación de código
Descripción	Se deberán generar diferentes códigos en función de la dificultad seleccionada cada vez que se acierta o se falla
Resultado	Correcto

Cuadro 67: Descripción prueba 13

Prueba - 14	
Nombre	Puntuación cambia de color
Descripción	La puntuación deberá de cambiar a color rojo y aparecer las estrellas tras alcanzar la puntuación necesaria para completar el nivel
Resultado	Correcto

Cuadro 68: Descripción prueba 14

Prueba - 15	
Nombre	Generar una sola bola
Descripción	Deberá aparecer una sola bola cada vez
Resultado	Incorrecto
Motivo	Se desconoce el por que aparecen en ocasiones más de una bola. Un posible motivo es el *prefab* dentro del juego
Solución	No se ha encontrado solución

Cuadro 69: Descripción prueba 15

12. GDD *Ficheros*

El juego consiste en pulsar una serie de botones en el orden adecuado para resolver un problema planteado. Estos botones representan distintas operaciones simples aplicables a ficheros de acceso secuencial.

12.1. Descripción

Se trata de un juego de carácter didáctico que pretende explicar el funcionamiento de los ficheros de acceso secuencial y los diferentes conceptos relacionados con la apertura en modo lectura o escritura de estos.

El objetivo del juego consiste en realizar una acción, como por ejemplo *"Se desea escribir en el fichero 1"*. Para alcanzar este objetivo, el jugador deberá ejecutar todos los pasos necesarios en el orden correcto pulsando los botones que representan posibles acciones simples.

En la Figura 19 se puede ver de forma detallada el funcionamiento del juego:

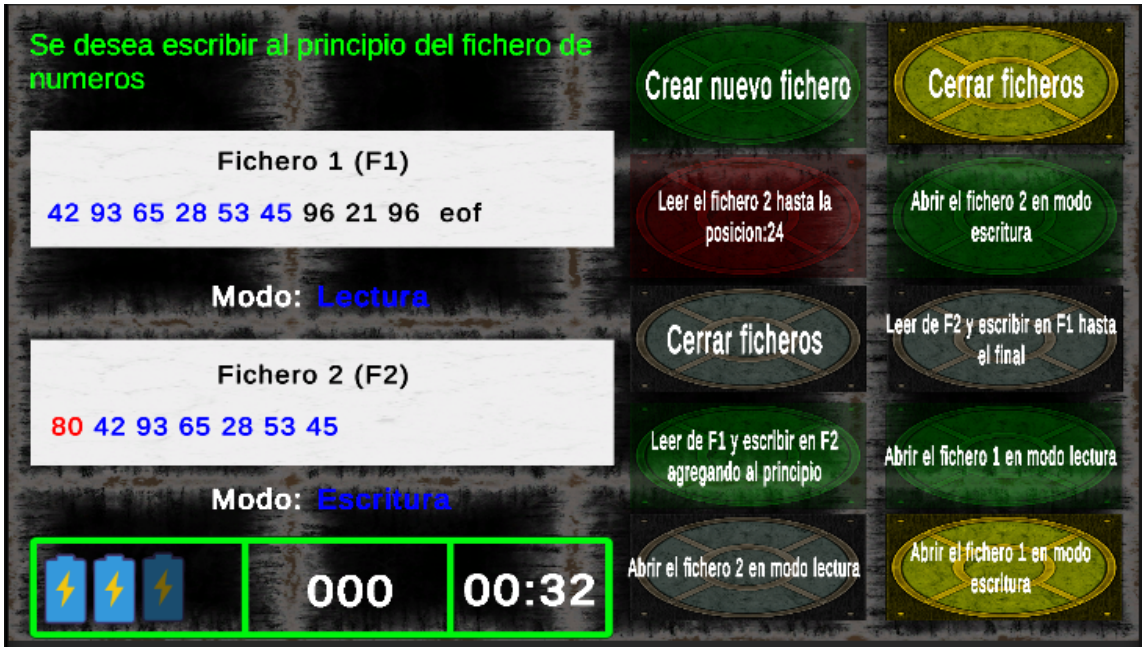

Figura 19: Figura juego de ficheros

Como se puede ver en la Figura 19, en la parte de arriba se sitúa la acción que se desea realizar y a la derecha aparecen los botones que simbolizan los distintos pasos. Cada acción puede necesitar más o menos pasos, y estos han de realizarse en un orden determinado, por ejemplo, no es posible *'Abrir el fichero 2 en modo escritura'* si previamente no se ha *'Creado un nuevo fichero'*. Los botones que aparecen en verde, son las opciones que se han marcado correctamente siguiendo el orden previsto. Los botones en amarillo son opciones que, aunque son correctas, no siguen el orden previsto. Los botones en rojo son opciones que no son correctas, por lo tanto el botón se deshabilita para no confundir al jugador.

Además, cada acción se verá reflejada en el fichero de la parte izquierda, para una mejor comprensión del jugador sobre lo que ocurre cuando se trabaja con ficheros.

12.2. Recursos

Para este juego se han empleado una serie de texturas y *sprites*. Todas las texturas utilizadas han sido obtenidas de la *Asset Store* de *Unity*. Por el contrario los *sprites* han sido obtenidos de páginas que proporcionan iconos de forma gratuita [16].

Los vídeos utilizados en los tutoriales han sido grabados con *OBS Studios* [17] y modificados con *VideoPadEditor* [18].

12.3. Mecánicas

En la Tabla 70se muestran las distintas mecánicas de las que dispone este juego:

Mecánicas Juego de Ficheros	
Mecánica	**Descripción**
Dificultad seleccionada	La dificultad afecta al número de vidas del jugador y el temporizador.
Jugador selecciona una acción correcta	El botón de dicha acción se pone de color verde, se deshabilita y se representa dicha acción en el fichero.
Jugador selecciona una acción parcialmente correcta	El botón de dicha acción se pone de color amarillo, y no se deshabilita ni se muestra nada.
Jugador selecciona una acción incorrecta	El botón de dicha acción se pone de color rojo, se deshabilita y se reduce una vida, así como una cantidad al temporizador.
Jugador acierta todas las acciones	Se resetearán los botones y se mostrará otra acción a realizar junto con sus respuestas
Vidas llegan a 0	Fin del juego mostrando la puntuación obtenida, la máxima puntuación y el número de vidas.
Pausar la partida	Se pausará el juego mostrando el menú de pausa con todas las opciones disponibles.
Salir del juego	Se mostrará el menú de fin de juego indicando todas las puntuaciones.

Cuadro 70: Tabla de mecánicas del juego de ficheros

12.4. Análisis del Juego

12.4.1. Requisitos del Sistema

En esta sección se comentarán los distintos requisitos funcionales y no funcionales que tiene el juego.

ID	Nombre	Descripción
RF-01	Generar Pregunta	El sistema deberá generar preguntas aleatorias obtenidas de un fichero
RF-02	Mostrar Indicadores	El sistema deberá mostrar al iniciar el juego los distintos indicadores: puntuación, vidas y tiempo
RF-03	Generar Respuestas	El sistema deberá generar las tres respuestas que pertenecen a la pregunta mostrada, obteniéndolas de un fichero
RF-04	Presionar Botones	El sistema deberá permitir al jugador presionar los distintos botones que contienen las respuestas, cambiando el color a verde si esta respuesta es correcta en el orden determinado, en rojo si es incorrecta o en amarillo si es correcta pero no sigue el orden determinado
RF-05	Sumar Puntos	El sistema deberá sumar cierta puntuación a su respectivo contador si se dispara a la diana con la respuesta correcta
RF-06	Cambiar Preguntas y Respuestas	El sistema deberá cambiar la pregunta y respuestas cuando se resuelva la pregunta anterior
RF-07	Efectos en el Juego	El sistema deberá mostrar en pantalla un nuevo fichero cuando se presione el botón *crear un nuevo fichero* o mostrar efectos de escritura cuando se pulsen los botones de leer y escribir
RF-08	Restar Vidas	El sistema deberá restar una vida al jugador si se presiona un botón incorrecto
RF-09	Pausar Juego	El sistema deberá permitir al jugador pausar el juego mostrando así el menú de pausa

Cuadro 71: Tabla requisitos del sistema del juego de ficheros

12.4.2. Casos de Uso

En esta sección se tratarán los distintos casos de uso que tiene el juego.

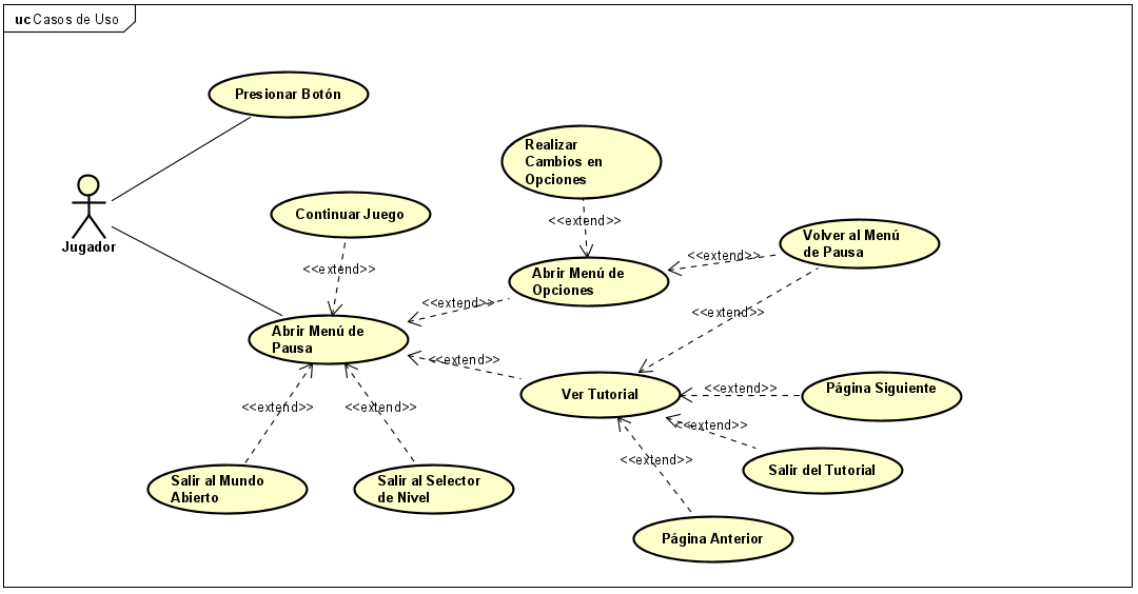

Figura 20: Figura casos de uso del juego de ficheros

12.4.3. Modelo de Dominio

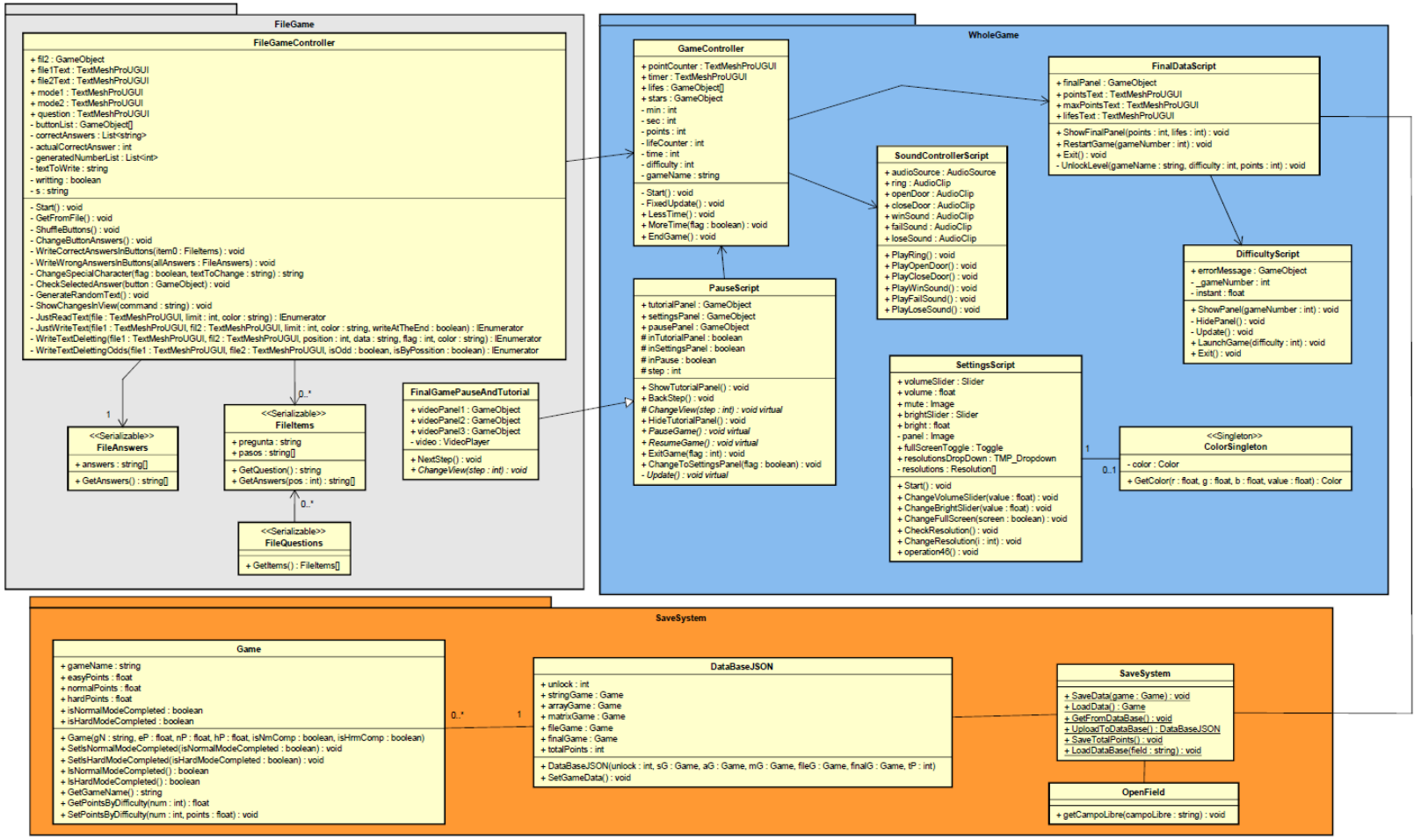

Figura 21: Figura modelo de dominio del juego de ficheros.

Como se detalla en la Figura 21, la clase *GameController* se encargará de la gestión de puntuaciones y del estado del juego. El controlador principal del juego es la clase *FileGameController* que se encarga de la generación de la cuestión a resolver, generar las respuestas posibles a dicha cuestión y representar en los ficheros las distintas acciones según el botón pulsado. Por ejemplo, Si se pulsa el botón *Leer del fichero 1 y escribir en el 2* se verá como poco a poco se van leyendo y escribiendo los datos en dicho fichero, con esto se espera conseguir que el alumno obtenga una mejor comprensión del funcionamiento de los ficheros mediante una representación visual.

Las cuestiones son obtenidas de un fichero que guarda un JSON de preguntas, y estas son transformadas a objeto empleando la librería *JSONUtility* que posee *Unity*, para ello es necesario crear dos clases. La que tiene mayor utilidad es la clase *FileItems*, que contiene las preguntas y respuestas, pero también es necesaria una clase ”puente”, en este caso la clase *FileQuestions* para lograrlo, es por ello que no existe relación entre la clase *FileGameController* y la clase *FileQuestions*, además las respuestas erroneas son obtenidas de otro fichero general que solo contiene respuestas posibles, para ello, se hace uso de la clase *FileAnswers*. Es importante añadir que estas clases han de ser *Serializables* para que puedan ser utilizadas.

La clase *FileGamePauseAndTutorial* como en el resto de casos se encarga de la gestión del tutorial del juego.

Tras finalizar el juego la clase *FinalDataScript* dará el aviso a la clase estática *SaveSystem* que creará un *String* en formato JSON, se modificaran las respectivas *PlayerPrefs* desde las cuales se obtienen los datos de cada partida y se dará el aviso a la clase *OpenField* la cual subirá dicho JSON a la base de datos, además de modificar la puntuación máxima conseguida si esta ha sido superada.

Cabe recalcar que, como se puede apreciar en la Figura 21, en el paquete *FileGame* el número de clases es bastante menor en comparación a otros juegos, esto es debido a que este juego es en 2D, por lo que se emplean menos clases para hacerlo funcionar.

12.5. Pruebas en la Plataforma

En esta sección se enumerarán las distintas pruebas llevadas a cabo en la plataforma donde se alojarán los juegos.

Prueba - 01	
Nombre	Iniciar juego
Descripción	Se deberá iniciar el juego tras haber elegido la dificultad
Resultado	Correcto

Cuadro 72: Descripción prueba 01

Prueba - 02	
Nombre	Acceder al menú de pausa
Descripción	Deberá aparecer el menú de pausa tras pulsar TAB parando el tiempo
Resultado	Correcto

Cuadro 73: Descripción prueba 02

Prueba - 03	
Nombre	Reanudar juego
Descripción	Deberá ocultarse el menú de pausa, continuando de esta manera el juego al pulsar *Reanudar* ó TAB
Resultado	Correcto

Cuadro 74: Descripción prueba 03

Prueba - 04	
Nombre	Acceder al panel de opciones
Descripción	Deberá aparecer el panel de opciones al pulsar en *Opciones*
Resultado	Correcto

Cuadro 75: Descripción prueba 04

Prueba - 05	
Nombre	Acceder al tutorial
Descripción	Deberá aparecer el panel de tutorial tras pulsar en *Tutorial*
Resultado	Correcto

Cuadro 76: Descripción prueba 05

Prueba - 06	
Nombre	Finalizar juego
Descripción	Deberá aparecer el panel de fin de juego tras pulsar en *Salir al selector de nivel*, al perder todas las vidas o al llegar el tiempo a 0. Mostrando en este la puntuación conseguida, la puntuación máxima conseguida y las vidas
Resultado	Correcto

Cuadro 77: Descripción prueba 06

Prueba - 07	
Nombre	Volver al jugar
Descripción	Deberá aparecer el selector de dificultad tras pulsar a *Volver a jugar* desde el panel de fin de juego
Resultado	Correcto

Cuadro 78: Descripción prueba 07

Prueba - 08	
Nombre	Salir del juego
Descripción	Se deberá volver al selector de nivel tras pulsar en *Seleccionar Nivel* desde el panel de fin de juego
Resultado	Correcto

Cuadro 79: Descripción prueba 08

Prueba - 09	
Nombre	Presionar respuesta correcta
Descripción	El botón presionado deberá ponerse en color verde e inhabilitarse, aumentando de esta manera la puntuación y el tiempo
Resultado	Correcto

Cuadro 80: Descripción prueba 09

Prueba - 10	
Nombre	Presionar respuesta parcialmente correcta
Descripción	El botón presionado deberá ponerse en color amarillo y no inhabilitarse
Resultado	Correcto

Cuadro 81: Descripción prueba 10

Prueba - 11	
Nombre	Presionar respuesta incorrecta
Descripción	El botón presionado deberá ponerse en color rojo e inhabilitarse, reduciendo una vida y el tiempo
Resultado	Correcto

Cuadro 82: Descripción prueba 11

Prueba - 12	
Nombre	Generación de preguntas y respuestas
Descripción	Se deberán obtener las preguntas y respuestas de un fichero según la dificultad y mostrarse estas de forma aleatoria
Resultado	Incorrecto
Motivo	En *Unity WebGL* no se puede realizar la lectura de ficheros en Straming Assets
Solución	Se ha creado una clase estática dentro del propio juego que contiene los JSON de preguntas

Cuadro 83: Descripción prueba 12

Prueba - 13	
Nombre	Efectos en los ficheros al presionar botones
Descripción	Se deberán poner en distinto color los caracteres o escribirse en el nuevo fichero al pulsar ciertas respuestas
Resultado	Correcto

Cuadro 84: Descripción prueba 13

Prueba - 14	
Nombre	Puntuación cambia de color
Descripción	La puntuación deberá de cambiar a color rojo y aparecer las estrellas tras alcanzar la puntuación necesaria para completar el nivel
Resultado	Correcto

Cuadro 85: Descripción prueba 14

13. GDD Juego Final

El juego consiste en disparar a una serie de dianas las cuales representan las respuestas a una pregunta. El objetivo es conseguir el mayor número de disparos certeros posibles.

13.1. Descripción

Se trata de un juego de carácter didáctico de tipo *quiz*, en el que se realizan preguntas relacionadas con las distintas estructuras de datos vistas en el juego.

El jugador verá en pantalla tres dianas las cuales contendrán las respuestas, y tendrá que disparar a la respuesta que el crea correcta. En esta modalidad el tiempo siempre avanza y las dianas irán cambiando de pregunta si el jugador no responde.

En la Figura 22 se puede ver de forma detallada el funcionamiento del juego:

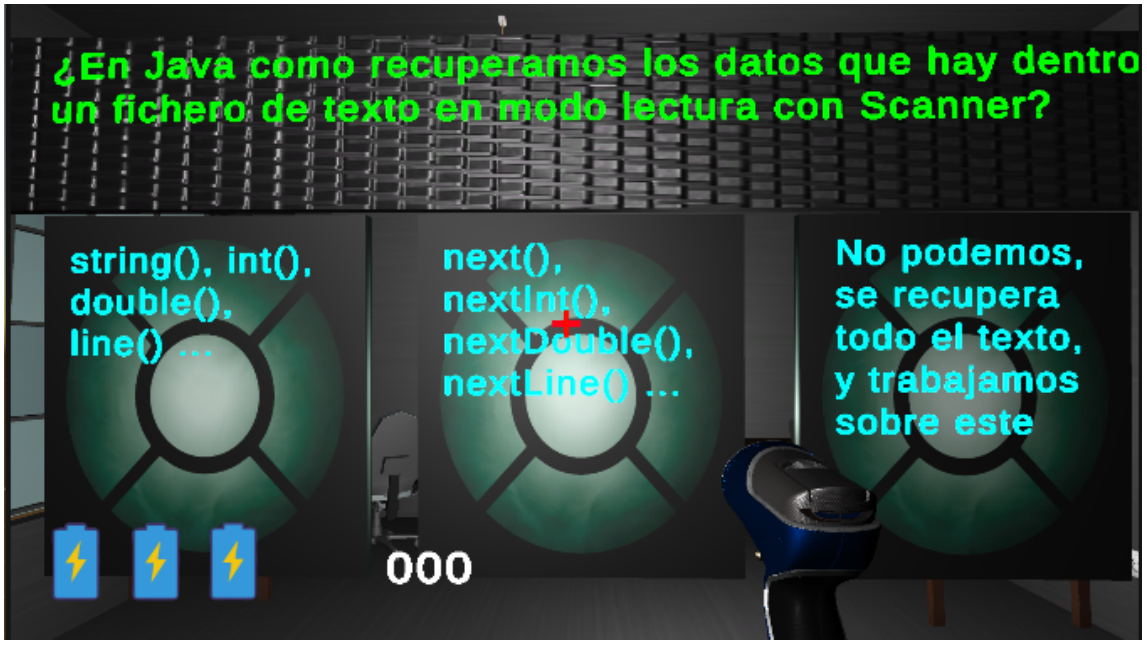

Figura 22: Figura juego final

Como se puede ver en la Figura 22, en la parte de arriba se sitúa la pregunta que hay que responder, mientras que en las tres dianas de abajo se encuentran las posibles respuestas, de las cuales solo una es correcta. Acertar o fallar hará aumentar la puntuación o reducir las vidas que hay en la parte inferior izquierda. El taladro que se ve en la imagen es la herramienta de disparo del jugador con la cual tendrá que apuntar a la diana que crea correcta y disparar.

Como se mencionó con anterioridad, este juego no posee temporizador como las preguntas van cambiando continuamente el jugador ya tiene limitado el tiempo para responder por lo que poner un temporizador a mayores sería contraproducente.

13.2. Recursos

Para este juego se han empleado una serie de *prefabs*, texturas, *sprites* y efectos. Todas las texturas utilizadas, así como los distintos efectos han sido obtenidos de la *Asset Store* de *Unity*. Por el contrario los *sprites* han sido obtenidos de páginas que proporcionan iconos de forma gratuita [16].

El listado de *prefabs* es el siguiente:

- **Dianas:** Son los paneles sobre los que aparecen las distintas preguntas y a los que hay que disparar. Han sido creadas con las distintas formas que ofrece *Unity* 3D.
- **Taladro:** Es la herramienta de disparo. El *prefab* utilizado para representarla ha sido obtenido de *TurboSquid* [7] de forma gratuita, mientras que las texturas utilizadas para darle color han sido obtenidas de la *Asset Store*.
- **Tornillos:** Son las balas que utiliza el taladro para disparar. El *prefab* utilizado ha sido obtenido de *TurboSquid* [7] de forma gratuita, mientras que las texturas utilizadas para darle color han sido obtenidas de la *Asset Store*.
- **Efectos:** Son los efectos que se producen al disparar (chispas) y al impactar la bala contra la diana (explosión) ambos efectos han sido obtenidos de forma gratuita de la *AssetStore*.

Los vídeos utilizados en los tutoriales han sido grabados con *OBS Studios* [17] y modificados con *VideoPadEditor* [18].

13.3. Mecánicas

En la Tabla 86 se muestran las distintas mecánicas de las que dispone este juego:

Mecánicas Juego Final	
Mecánica	**Descripción**
Dificultad seleccionada	La dificultad afecta al número de vidas del jugador y la velocidad de cambio de pregunta.
Disparo	Se dispara con click izquierdo del ratón
Movimiento de cámara	La cámara se moverá mediante el movimiento del ratón, este movimiento a su vez tendrá un rango limitado.
Jugador dispara a una diana correcta	Se suma 100 al contador de puntos y se cambia de pregunta.
Jugador dispara a una diana incorrecta	Se disminuye una vida del jugador y se cambia de pregunta.
Jugador no dispara y pasa el tiempo	Se cambia de pregunta sin alterar los marcadores.
Vidas llegan a 0	Fin del juego mostrando la puntuación obtenida, la máxima puntuación y el número de vidas.
Pausar la partida	No se parará el tiempo por lo tanto las preguntas seguirán cambiando. Se mostrará el menú de pausa con todas las opciones disponibles.
Salir del juego	Se mostrará el menú de fin de juego indicando todas las puntuaciones.

Cuadro 86: Tabla de mecánicas del juego final

13.4. Análisis del Juego

13.4.1. Requisitos del Sistema

En esta sección se comentarán los distintos requisitos funcionales y no funcionales que tiene el juego.

ID	Nombre	Descripción
RF-01	Generar Pregunta	El sistema deberá generar preguntas aleatorias obtenidas de un fichero
RF-02	Mostrar Indicadores	El sistema deberá mostrar al iniciar el juego los distintos indicadores: puntuación y vidas
RF-03	Mostrar Efectos	El sistema deberá mostrar los distintos efectos en el mapa
RF-04	Generar Respuestas	El sistema deberá generar las tres respuestas que pertenecen a la pregunta mostrada, obteniéndolas de un fichero
RF-05	Disparar	El sistema deberá permitir al jugador disparar
RF-06	Mover las Dianas	El sistema deberá mover las dianas tras un tiempo determinado en función de la dificultad, o cuando la bala impacte en la diana
RF-07	Sumar Puntos	El sistema deberá sumar cierta puntuación a su respectivo contador si se dispara a la diana con la respuesta correcta
RF-08	Cambiar de Preguntas	El sistema deberá cambiar la pregunta mostrada cuando la diana termine su movimiento
RF-09	Cambiar de Preguntas	El sistema deberá cambiar la pregunta mostrada cuando la diana termine su movimiento
RF-10	Restar Vidas	El sistema deberá restar una vida al jugador si se dispara a una diana incorrecta
RF-11	Pausar Juego	El sistema deberá mostrar el menú de pausa sin pausar el juego

Cuadro 87: Tabla requisitos del sistema del juego final

13.4.2. Casos de Uso

En esta sección se tratarán los distintos casos de uso que tiene el juego.

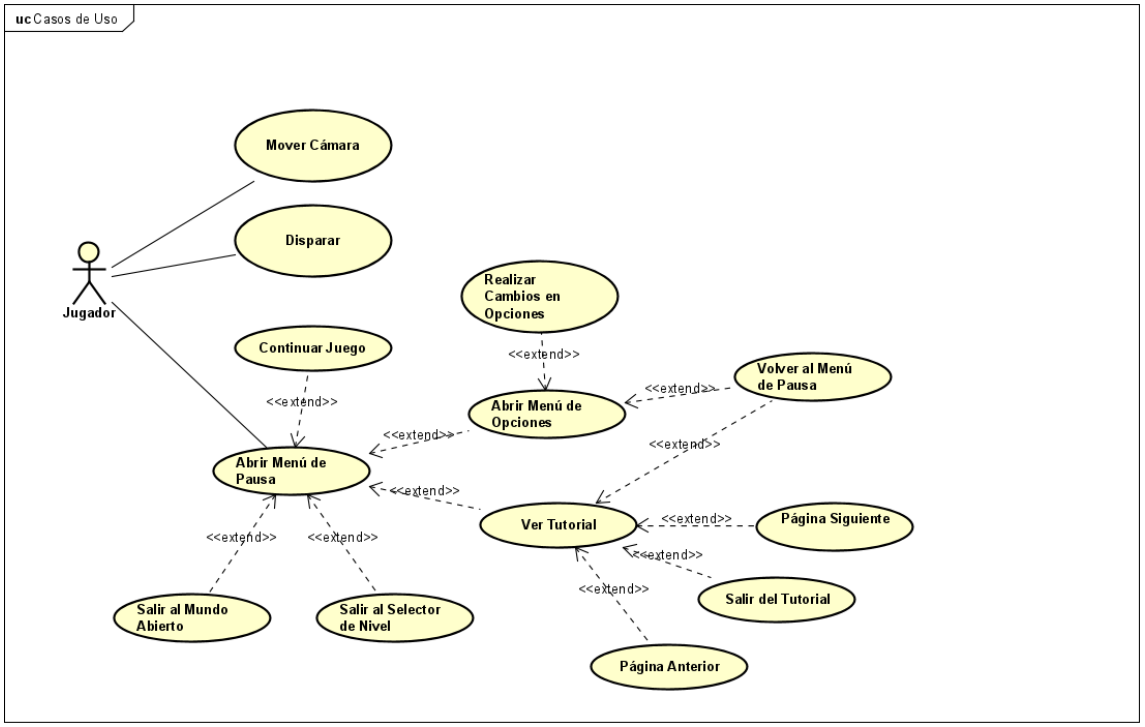

Figura 23: Figura casos de uso del juego final

13.4.3. Modelo de Dominio

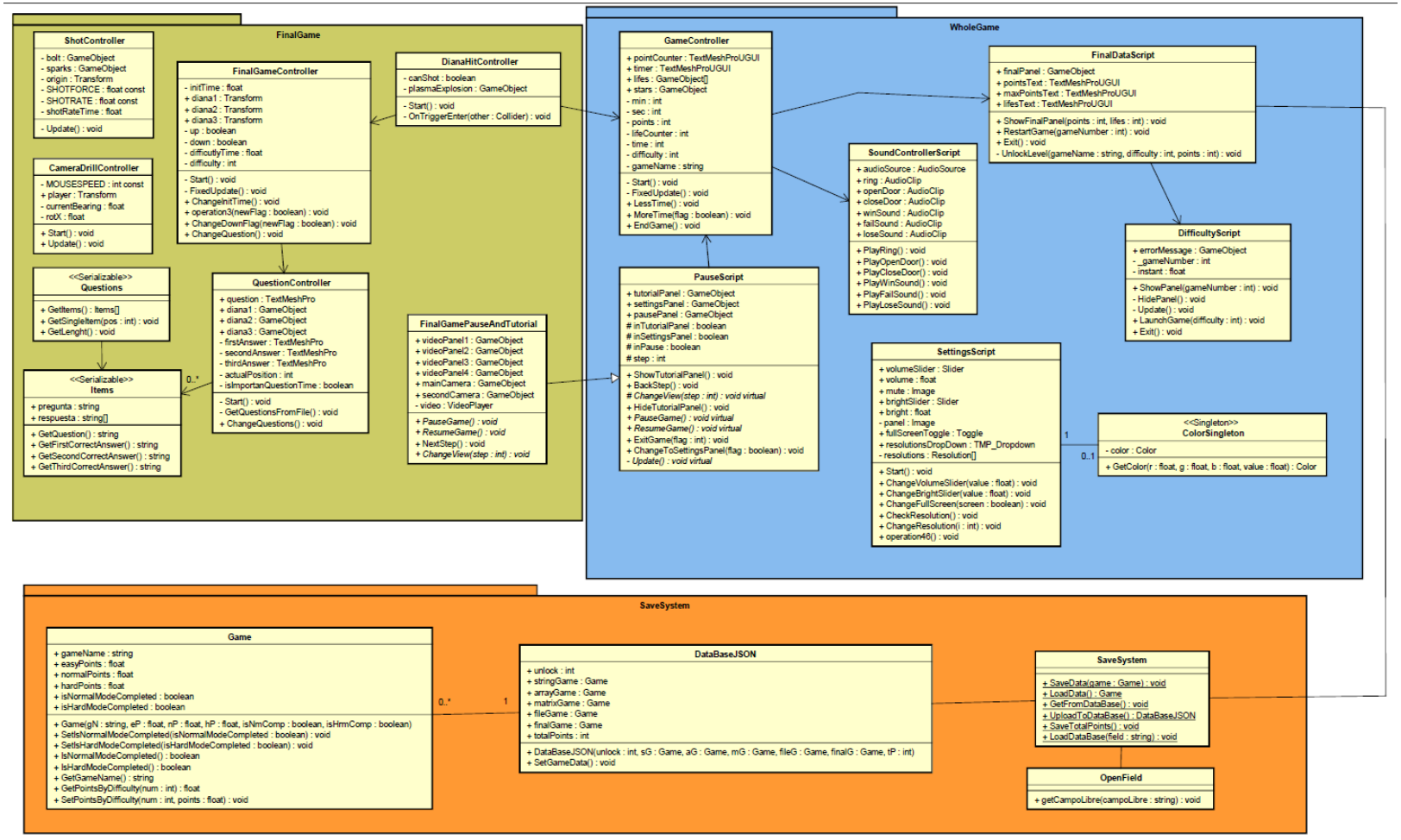

Figura 24: Figura modelo de dominio del juego final.

Como se muestra en la Figura 24, la clase *GameController* se encarga de la gestión de puntuaciones y del estado del juego. La clase *DianaHitController* se encarga de detectar a que diana se ha disparado e informar al *GameController* de si es un acierto o fallo. Esta clase también avisa al *FinalGameController* que se encarga de gestionar el movimiento de las dianas y de las preguntas y respuestas que aparecen.

La clase *QuestionController* se encarga de obtener las preguntas de un fichero que guarda un JSON de preguntas, estas son transformadas a objeto empleando la librería *JSONUtility* que posee *Unity*, para ello es necesario crear dos clases. La de mayor utilidad es la clase *Items*, que contiene las preguntas y respuestas, pero, para conseguirlo es necesario una clase ”puente”, en este caso, se emplea la clase *Questions*. Esta es la razón por la que no existe relación entre la clase *QuestionController* y la clase *Questions*. Es importante añadir que estas clases han de ser *Serializables* para que puedan ser utilizadas.

Las clases *CameraDrillController* y *ShotController* se encargan del movimiento de cámara y del disparo. La clase *FinalGamePauseAndTutorial*, tal y como ya se ha comentado, se encarga de la gestión del menú de pausa.

Tras finalizar el juego, la clase *FinalDataScript* dará el aviso a la clase estática *SaveSystem* la cual creará un *String* en formato JSON, se modificaran las respectivas *PlayerPrefs* desde las cuales se obtienen los datos de cada partida y se dará el aviso a la clase *OpenField* la cual subirá dicho JSON a la base de datos, además de modificar la puntuación máxima conseguida si esta ha sido superada.

13.5. Pruebas en la Plataforma

En esta sección se enumerarán las distintas pruebas llevadas a cabo en la plataforma donde se alojarán los juegos.

Prueba - 01	
Nombre	Iniciar juego
Descripción	Se deberá iniciar el juego tras haber elegido la dificultad
Resultado	Correcto

Cuadro 88: Descripción prueba 01

Prueba - 02	
Nombre	Acceder al menú de pausa
Descripción	Sin parar tiempo del juego deberá aparecer el menú de pausa tras pulsar TAB
Resultado	Correcto

Cuadro 89: Descripción prueba 02

Prueba - 03	
Nombre	Reanudar juego
Descripción	Deberá ocultarse el menú de pausa, continuando de esta manera el juego al pulsar *Reanudar* ó TAB
Resultado	Correcto

Cuadro 90: Descripción prueba 03

Prueba - 04	
Nombre	Acceder al panel de opciones
Descripción	Deberá aparecer el panel de opciones al pulsar en *Opciones*
Resultado	Correcto

Cuadro 91: Descripción prueba 04

Prueba - 05	
Nombre	Acceder al tutorial
Descripción	Deberá aparecer el panel de tutorial tras pulsar en *Tutorial*
Resultado	Correcto

Cuadro 92: Descripción prueba 05

Prueba - 06	
Nombre	Finalizar juego
Descripción	Deberá aparecer el panel de fin de juego tras pulsar en *Salir al selector de nivel*, al perder todas las vidas Mostrando en este la puntuación conseguida, la puntuación máxima conseguida y las vidas
Resultado	Correcto

Cuadro 93: Descripción prueba 06

Prueba - 07	
Nombre	Volver al jugar
Descripción	Deberá aparecer el selector de dificultad tras pulsar a *Volver a jugar* desde el panel de fin de juego
Resultado	Correcto

Cuadro 94: Descripción prueba 07

Prueba - 08	
Nombre	Salir del juego
Descripción	Se deberá volver al selector de nivel tras pulsar en *Seleccionar Nivel* desde el panel de fin de juego
Resultado	Correcto

Cuadro 95: Descripción prueba 08

Prueba - 09	
Nombre	Disparar
Descripción	Se deberá poder disparar pulsando *Click Derecho*, tendrá que haber cierto delay entre disparos
Resultado	Correcto

Cuadro 96: Descripción prueba 09

Prueba - 10	
Nombre	Mover cámara
Descripción	La cámara deberá moverse empleando el movimiento del ratón
Resultado	Correcto

Cuadro 97: Descripción prueba 10

Prueba - 11	
Nombre	Movimiento de dianas
Descripción	Las dianas deberán moverse cada cierto tiempo de esta manera cambiando las preguntas
Resultado	Correcto

Cuadro 98: Descripción prueba 11

Prueba - 12	
Nombre	Generación de preguntas y respuestas
Descripción	Se deberán obtener las preguntas y respuestas de un fichero mostrarse estas de forma aleatoria
Resultado	Incorrecto
Motivo	En *Unity WebGL* no se puede realizar la lectura de ficheros en Straming Assets
Solución	Se ha creado una clase estática dentro del propio juego que contiene los JSON de preguntas

Cuadro 99: Descripción prueba 12

Prueba - 13	
Nombre	Efectos al disparar
Descripción	Se deberán generar ciertos efectos visuales al disparar e impactar una bala en una diana
Resultado	Correcto

Cuadro 100: Descripción prueba 13

Prueba - 14	
Nombre	Disparo en diana correcta
Descripción	Al disparar en una diana correcta se tendrá que sumar cierta puntuación al marcador, haciendo de esta manera que se muevan las dianas y se cambie de pregunta y respuestas
Resultado	Correcto

Cuadro 101: Descripción prueba 14

Prueba - 15	
Nombre	Disparo en diana incorrecta
Descripción	Al disparar en una diana incorrecta se tendrá que disminuir una vida haciendo de esta manera que se muevan las dianas y se cambie de pregunta y respuestas
Resultado	Correcto

Cuadro 102: Descripción prueba 15

Prueba - 16	
Nombre	Puntuación cambia de color
Descripción	La puntuación deberá de cambiar a color rojo y aparecer las estrellas tras alcanzar la puntuación necesaria para completar el nivel
Resultado	Correcto

Cuadro 103: Descripción prueba 16

14. Mantenimiento de Juegos en la Plataforma

14.1. Introducción

En esta sección se explicará el mantenimiento realizado en dos juegos ya existentes en la plataforma creados por otros alumnos del grado en Ingeniería Informática y grado en Estadística.

Las modificaciones solicitadas vienen derivadas de los comentarios realizados por los alumnos durante las pruebas en explotación. Ambos juegos se han utilizado durante este curso en condiciones reales y con el público para el cual estaban diseñados.

En ambos casos las modificaciones a realizar fueron muy sencillas y no llevaron demasiado tiempo.

14.2. Juego 1: Apilas

En este juego, en primer lugar, se debían de traducir todos los métodos utilizados en las distintas clases al inglés, de esta forma logrando una mayor estandarización de la herramienta.

En segundo lugar había que generar las distintas bolas que aparecen en el juego de forma aleatoria, dado que anteriormente los colores de estas estaban prefijados y no había aleatoriedad, por lo que un alumno que hubiera jugado al juego varias veces lograría una puntuación muy alta sin apenas esfuerzo.

En tercer lugar fue necesario modificar el sistema de guardado de puntos para que los *logs* que se muestran en la base de datos fueran correctos. Esto se debe a que a pesar de realizar nuevas partidas el juego siempre aparecía como completado en dichos *logs*.

14.3. Juego 1: Apuntados

En este juego había que traducir los distintos métodos de las clases al inglés y modificar el sistema de guardado, como se había realizado en el juego anterior.

15. Conclusiones

Tras finalizar el proyecto se puede afirmar de forma satisfactoria que se han cumplido prácticamente todos los objetivos propuestos.

Se han completado los 5 juegos que tenían pensado realizarse para el apartado de gamificación de la asignatura de *Fundamentos de Programación* sobre *Estructuras de Datos* y a su vez estos juegos han sido adaptados a la plataforma.

El uso de las metodologías ágiles ha resultado útil para planificar el proyecto y seguir en la medida de lo posible el plan acordado, además el realizar *Sprints* con más carga de trabajo al principio ha servido para reducir los posibles riesgos finales.

Tras un total de 7 *Sprints* la duración del proyecto fue aproximadamente de unas 250 horas, esto quiere decir que el trabajo fue completado con una duración menor a la planificada inicialmente.

Dado que la duración del proyecto es menor, el coste de personal pasaría a ser de 1750 euros (el equivalente a aproximadamente 7 euros/hora), y dado que no se tuvo ningún gasto adicional el coste final del proyecto sería de 2850 euros.

15.1. Logros Obtenidos

- La división de trabajo en *Sprints* hizo posible una mejor gestión de este, logrando con ello los objetivos propuestos.
- El listado de riesgos inicial hizo posible que se tuvieran en cuenta diferentes técnicas de trabajo, lo cual se vio reflejado en la mayor carga de trabajo inicial y una reducción del trabajo al final.
- La gran amplitud del juego deja lugar a numerosas mejoras futuras, además de que la implementación de estos ha sido realizada teniendo en cuenta esta posibilidad.
- La buena relación con los profesores dio lugar a que incluso la carga de trabajo por parte del alumno se viera reducida enormemente.

15.2. Inconvenientes en el Proyecto

- Se trabajó mucho en la creación de un modo primera persona, que a pesar de que en un futuro puede utilizarse y recibir cierto protagonismo, actualmente no puede utilizarse en la plataforma web dado que su única función es seleccionar el nivel, pero moviéndose por el escenario, lo cual puede dar lugar a confusiones y distracciones, por lo que se ha decidido no añadirlo en la versión final de la plataforma. Esto ha sido debido a un planteamiento erróneo inicial del juego, ya que se pensó que podría tener cierto impacto en el juego, o que se podría añadir algún *Easter Egg*, pero no se encontró una aplicación útil para él. A pesar de su poca utilidad práctica actual, hay que decir que gracias al desarrollo de este modo, se aprendió enormemente sobre el funcionamiento de *Unity* 3D.
- La opción de ajustar la resolución en el menú de opciones no funciona en la plataforma debido a que esta opción obtiene las resoluciones de la máquina donde se está ejecutando, por lo que en la plataforma web este menú de resoluciones aparece vacío. Aunque no representa un problema, supuso un trabajo adicional por no saberlo a priori.
- Debido a las distintas actualizaciones que tienen los programas constantemente, la interacción con la plataforma se demoró más de lo esperado ya que la generación de los archivos había sido modificada en la última versión de *Unity*.
- No se consiguió seguir con el plan inicial, lo cual hizo que los *Sprints* finales fueran más largos, o se añadiesen *Sprints* de más, aunque ya había sido tenido en cuenta en la lista de riesgos.
- El cansancio mental en la etapa final del trabajo hizo que este se realentizase, aunque al haberse tenido en cuenta en los riesgos se pudo solventar mediante una mayor carga de trabajo inicial.
- Debido a la fecha de realización del juego no se pudieron realizar pruebas con los alumnos que cursan la asignatura para la que está pensado el juego, que hubiera sido lo ideal. Aun así, se realizaron pruebas externas con alumnos ya experimentados y su opinión fue muy útil a la hora de concluir el proyecto.

15.3. Discusión

Con todo lo mencionado anteriormente, si pudiera empezar el proyecto de nuevo, la planificación hubiera sido la misma, dado que ha resultado ser efectiva, pero se hubiera dado otro planteamiento del juego, evitando de esta manera la creación del modo en primera persona, o dándole un buen uso a este, no siendo simplemente un escenario donde puedas moverte. También se hubiera dedicado algo más de tiempo a plantear un juego específico para registros, los cuales actualmente están incluidos en el juego final.

Respecto al resto del proyecto ha sido muy satisfactorio trabajar en ello y no podría haber salido de mejor manera.

15.4. Trabajos futuros

En esta sección se enumeraran distintas implementaciones y mejoras que podrían realizarse en el futuro:

- Añadido de *items* adicionales, en algunos juegos como el de *Strings*, estos tendrían cierto impacto en el juego, como recuperar una vida, ralentizar el tiempo, puntos dobles, etc...
- Darle un buen uso al juego el primera persona, poco hay más que añadir que no se haya mencionado con anterioridad.
- Añadir más estructuras de datos, es decir, crear nuevos juegos para explicar otras estructuras de datos.
- Añadido de música y sonidos adicionales los cuales podrían dar más ambiente al juego y crear una mejor experiencia para el jugador.
- Crear un sistema mediante el cual las preguntas de los juegos final y de ficheros estén alojadas en la base de datos o en un servidor, de esta manera obteniéndolas como si de una carpeta se tratase, mejorando así enormemente el sistema de añadido de preguntas y su mantenimiento.

16. Anexo

16.1. Repositorio *GitLab*

El siguiente enlace pertenece al repositorio *GitLab* donde se aloja el código del juego: https://gitlab.inf.uva.es/minijuegos_gamificacion/estructurados.git

16.2. Manual de Generación de Juego y Subida a la Plataforma

En esta sección se explicará como hacer *build* del proyecto y subirlo a la plataforma para versiones posteriores de *Unity* 2020, dado que el método explicado en versiones anteriores ha quedado obsoleto.

16.2.1. Configuración del Build

Una vez terminado nuestro juego el último paso es generar el *build* de este, para ello será necesario ir a **File** → **Build Settings** en la parte de arriba a la izquierda del editor de *Unity*.

Tras eso se abrirá un panel como el de la Figura 25:

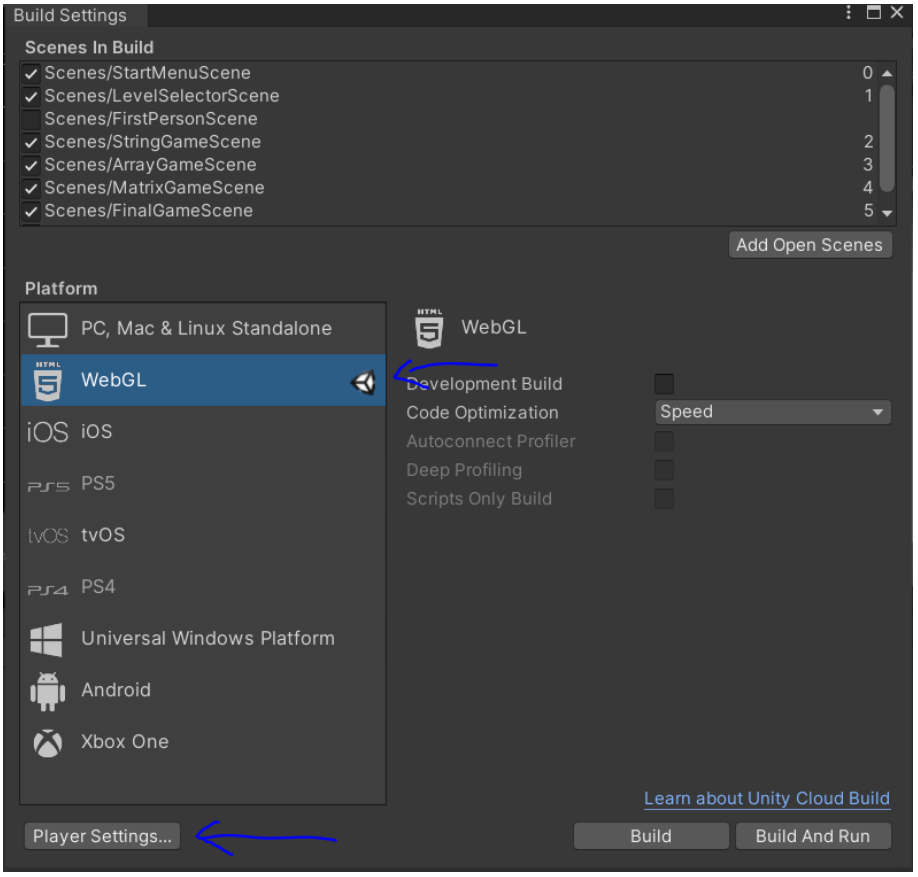

Figura 25: Figura de *Build Settings*

Como se puede ver en la Figura 25 aparecerán una serie de plataformas para hacer *build* del proyecto, la que interesa es la de `WebGL`, una vez seleccionado muy probablemente donde pone **Build**, aparecerá **Switch Platform**, habrá que cambiar la plataforma seleccionada.

Una vez modificada la plataforma presionaremos en **Player Settings**, con lo que se abrirá el siguiente panel:

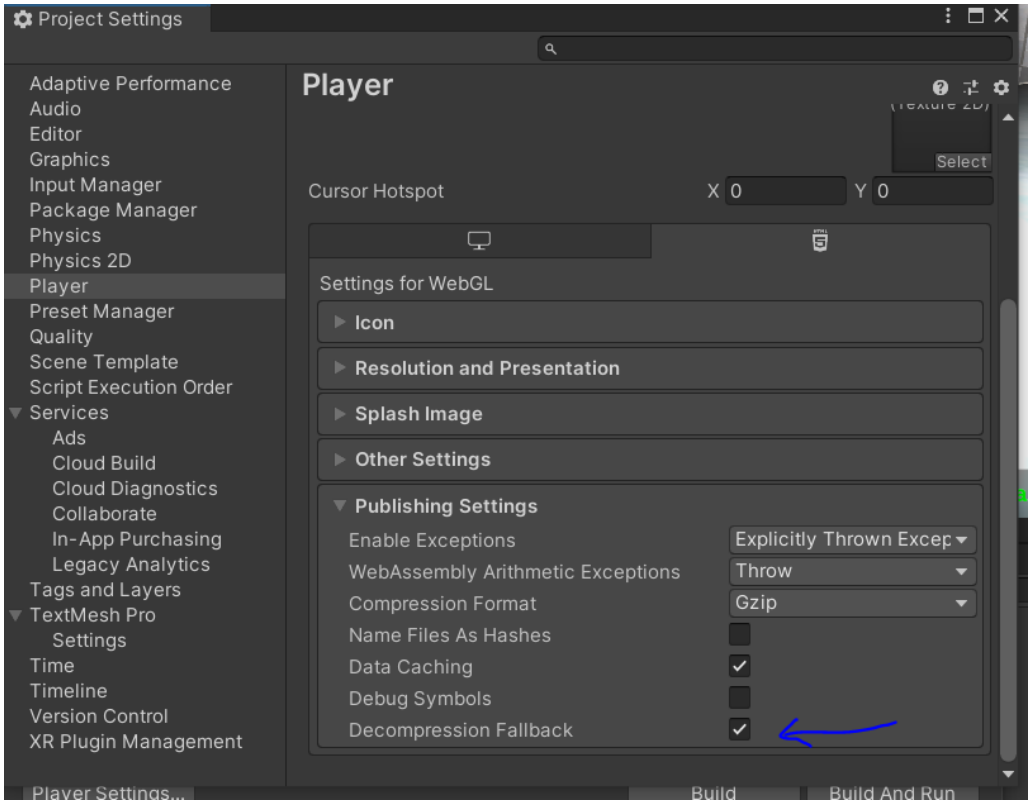

Figura 26: Figura de *Player Settings*

Al igual que en la Figura 26 iremos a **Publishing Settings** y marcaremos la opción ***Decompression Fallback*** **¡ESTA PARTE ES MUY IMPORTANTE**.

16.2.2. Añadir Juego

Antes de hacer el *build* es muy importante ir a la plataforma, tras iniciar sesión habrá que ir al panel de desarrollador y seleccionar **Añadir Juego**, en la Figura 27 se pueden ver todos los campos a rellenar:

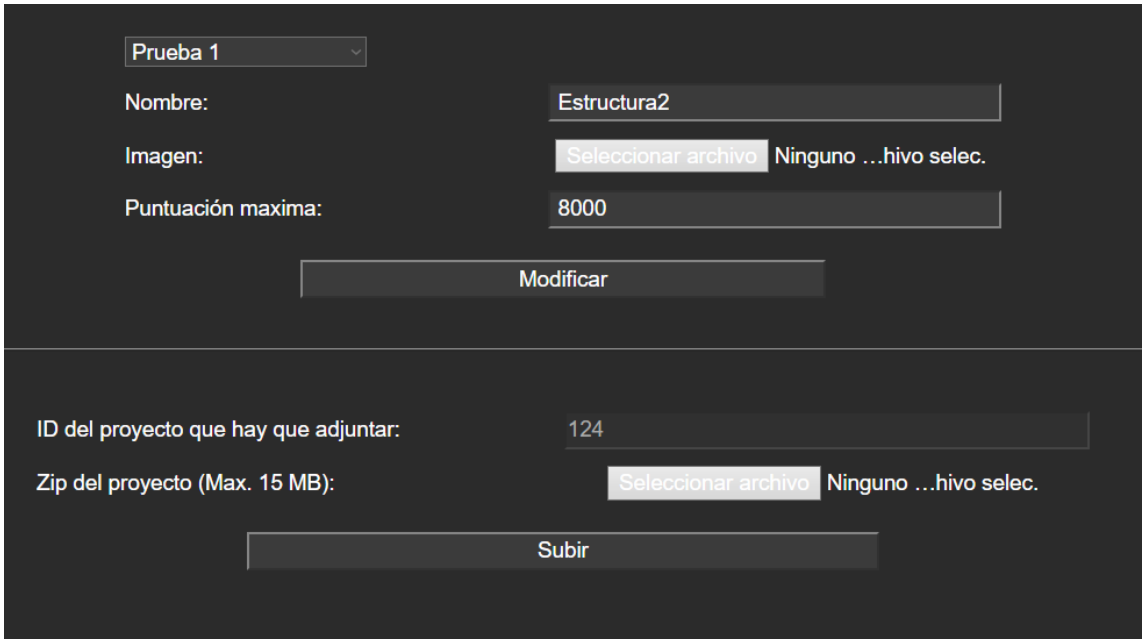

Figura 27: Figura de Añadir Juego

Todos los campos que se ven son obligatorios, pero la página dará *feedback* por si se olvida alguno, lo que interesa es el **ID del proyecto**.

Se procederá a crear la carpeta donde se hará *build* del juego, esta tendrá que llamarse igual que el ID del proyecto, en este caso es 124.

En la Figura 28 se puede ver como se ha creado dicha carpeta, también será necesario **crear un ZIP que se llame igual que el ID del proyecto**:

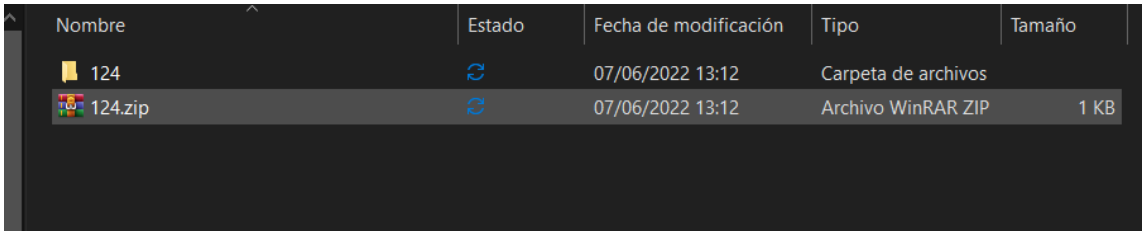

Figura 28: Figura de la carpeta y ZIP del proyecto

Una vez hecho el *build* del proyecto en la carpeta, su interior debería ser como este:

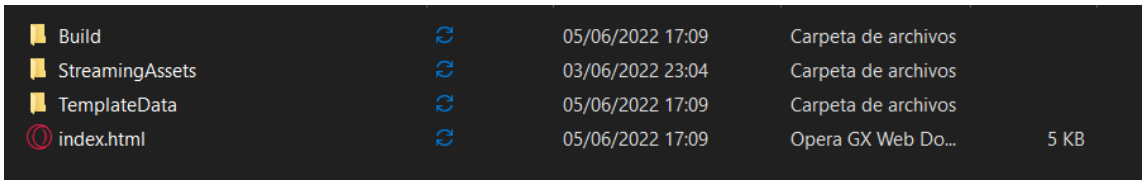

Figura 29: Figura de la carpeta donde se aloja el proyecto

De esta carpeta interesa la carpeta *Build* y el `index.html`, comprobaremos que en la carpeta *Build* todo está correcto. Su interior debería ser como el siguiente:

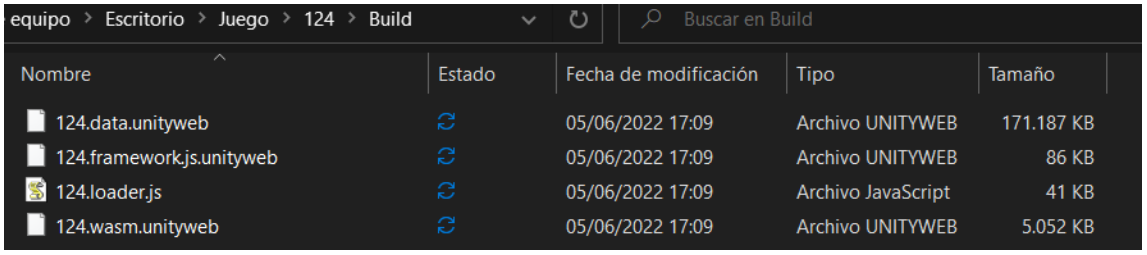

Figura 30: Figura de la carpeta y ZIP del proyecto

Como se puede ver en la Figura 30, hay varios archivos `.unityweb`, esto solo aparecerá si se ha seleccionado el **Decompression Fallback** en **Player Settings** por eso era tan importante esa parte, si no se hubiera marcado, dentro de esta carpeta habría una serie de `.ZIPs`.

16.2.3. Modificación del Index.HTML

Ahora por último se procederá a modificar el `index.html` que en las versiones de *Unity* 2020 y posteriores es distinto, el resultado final debería ser como el siguiente:

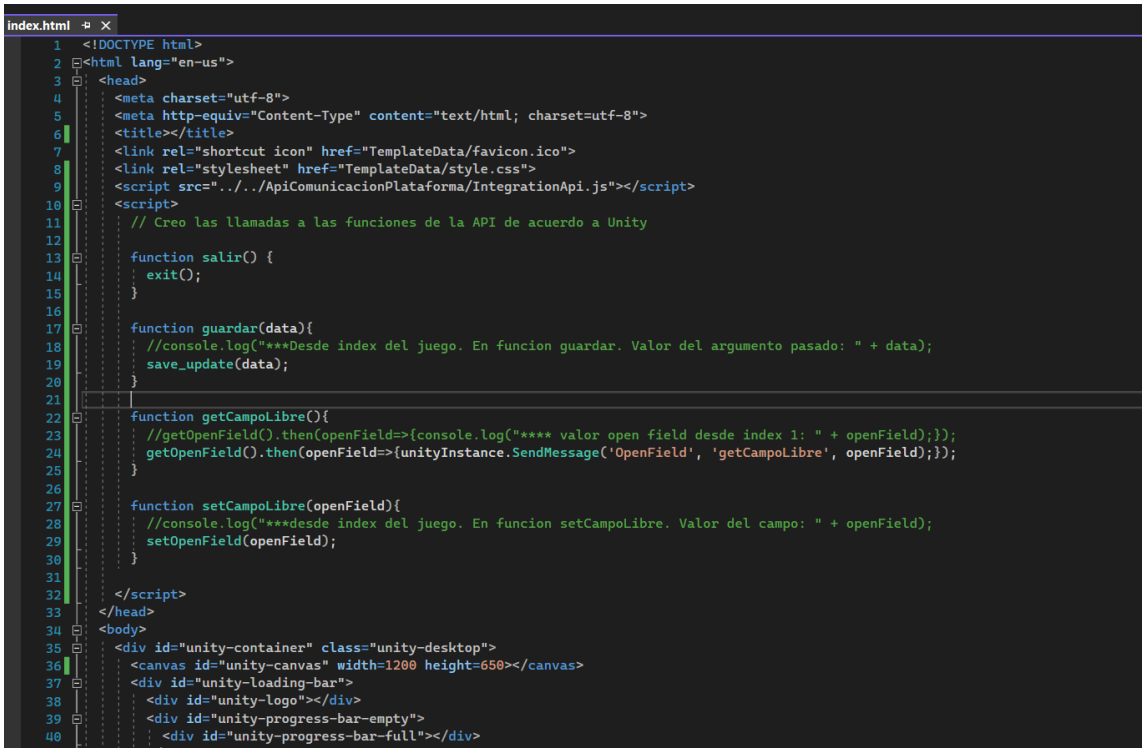

Figura 31: Figura del Index.html

Al igual que en la Figura 31, hay que copiar justo al final de la etiqueta *Head* los *Scripts* que llaman a la plataforma.

Como último paso tan solo hay que subir el juego, **¡Importante esperar hasta que la página nos devuelva el mensaje de confirmación!**. Generalmente si el juego no supera los 50MB el tiempo de espera es muy corto, pero si supera los 200MB puede llevar hasta un minuto (Actualmente el tamaño máximo es de 250MB).

Una recomendación es subir el juego desde el navegador de *Mozilla Firefox* debido a que otros navegadores al poseer bloqueadores de anuncios o configuraciones en el *proxy* pueden producir conflictos con las llamadas a la base de datos.

17. Bibiliografía

Referencias

[1] Ebot, 10 Benefícios de la Gamificación en el Aula https://ebot.es/beneficios-gamificacion-aula/ (Último acceso 18/05/2022).

[2] Smartmind, Gamificación en el aula, ventajas y desventajas [4 Octubre 2018] https://www.smartmind.net/blog/gamificacion-en-el-aula-ventajas-y-desventajas/ (Último acceso 18/04/2022).

[3] El Documentalista Audiovisual, Documentación en Videojuegos: Documento de diseño (GDD) [6 Febrero 2015] https://eldocumentalistaudiovisual.com/2015/02/06/documentacion-en-videojuegos-documento-de-diseno-gdd/ (Último acceso 18/04/2022).

[4] Unity https://unity.com/es(Último acceso 24/05/2022)

[5] Monday https://monday.com/lang/es/(Último acceso 02/06/2022)

[6] Mixamo, Get Animatedhttps://www.mixamo.com/ (Último acceso 30/04/2022)

[7] ShutterStock, Turbosquid 3D Models for Professionals https://www.turbosquid.com(Último acceso 10/05/2022)

[8] Manifiesto por el Desarrollo Ágil de Software https://agilemanifesto.org/iso/es/manifesto.html (Último acceso 19/04/2022)

[9] Desarrollo Ágil de Software https://es.wikipedia.org/wiki/Desarrollo_agil_de_software (Último acceso 19/04/2022)

[10] Videojuegos, arte en el mundo de la informática https://www.nacion.com/viva/cultura/los-videojuegos-arte-en-el-mundo-de-la-informatica/ZEPTZVEOQJFTJDJINOERBLI6JA/story/(Último acceso 20/04/2022).(Último acceso 22/04/2022)

[11] Que es la gamificación y cuales son sus objetivos https://www.educaciontrespuntocero.com/noticias/gamificacion-que-es-objetivos/(Último acceso 20/04/2022).

[12] Sueldo de un diseñador de videojuegos: ¿cuánto cobra? https://www.tokioschool.com/noticias/sueldo-disenador-videojuegos-cuanto-cobra/#:~:text=Dise~nador%20de%20videojuegos%3A%20sueldo%20junior,los%2026.000%20euros%20brutos%20anuales. (Último acceso 20/04/2022)

[13] Sueldo del Programador de Videojuegos en España https://www.jobted.es/salario/programador-videojuegos (Último acceso 20/04/2022).

[14] *Bob Hughes & Mike Cotterell* (2009), Software Project Managment Fifth Edition. McGraw-Hill Education.

[15] Astah, The power of software modeling https://astah.net

[16] Icons8, Iconos, ilustraciones, fotos, música y herramientas de diseño https://iconos8.es/icons/s

[17] OBS, Open Broadcaster Studio https://obsproject.com/es

[18] VideoPad, editor de vídeo https://www.nchsoftware.com/videopad/es/index.html?ns=true&kw=videopad&gclid=CjwKCAjwx46TBhBhEiwArA_DjOZQvAWPOyrjUn-G_VUAv9bm27xD3x0ZjS4kO6lmXHk5bETJXJHmJxoCmUsQAvD_BwE

[19] Overleaf, Editor de Latex online https://es.overleaf.com

[20] Pixabay, Música de videojuegos gratis para descargar https://pixabay.com/es/music/search/genre/video%20juegos/ (Último acceso 15/05/2022)