Rediseño de un juego 2D hecho en Flash a Unity

Autor: David Gata Balboa

Director: Lluis Solano Albajes

Ingeniería en Informática Plan 2003

1 Septiembre 2016

Índice

Capítulo 1 – Introducción

En este capítulo se va a exponer la motivación, descripción detallada del proyecto y sus objetivos. Además de una pequeña guía sobre la memoria para ayudar al lector con el documento.

1.1 Motivación

Desde hace tiempo la Universidad Politécnica de Catalunya (UPC) y la Secretaria d'Universitats i Recerca de la Generalitat de Catalunya colaboran en un proyecto que trata de acercar el conocimiento sobre personajes de la historia de la ciencia y la tecnología de Catalunya mediante una serie de juegos pensados para estudiantes de primaria.

El proyecto llamado "Personatges en joc" es una colección de juegos sobre distintos personajes históricos catalanes: Narcís Monturiol inventor del primer submarino, Ferran Alsina que jugó un papel importante en la industrialización de Catalunya, Dolors Aleu Riera la primera médica licenciada en el estado Español, Francesc de Castellví i Obado historiador que participó en la defensa de Barcelona en 1714, Joan de Peratallada maestro en alquimia de la época medieval y Miquel Crusafont un reconocido paleontólogo. Estos juegos se desarrollaron inicialmente en la plataforma Adobe Flash y pueden ser jugados desde un navegador web mediante la página http://www.personatgesenjoc.cat/jugar/

Flash es una tecnología que tuvo una gran aceptación en el desarrollo web cuando se lanzó debido a sus capacidades gráficas, fue una gran novedad, convirtiéndose en un estándar, **no abierto,** muy extendido. Actualmente esta tecnología se encuentra en desuso ya que está siendo sustituida por HTML5, una nueva tecnología que está siendo muy impulsada, ya que aparte de tener las mismas capacidades que Flash, es más ligero y además abierto.

Debido a que flash está sentenciado a desaparecer, ya hay incluso navegadores que no los soportan, se pretende hacer una actualización de estos juegos a una tecnología más reciente y que ofrezca más posibilidades, como por ejemplo el desarrollo multiplataforma. De esta manera los juegos pueden seguir funcionando para el objetivo que se crearon, llevar a los más pequeños conocimientos sobre personajes que influenciaron en la historia catalana.

1.2 Descripción del Proyecto

El proyecto consiste en realizar un estudio, análisis e implementación parcial del juego llamado **Narcís Monturiol,** para realizar su rediseño o "port" al motor gráfico Unity3D.

Este análisis además de tener como objetivo estudiar el problema y encontrar las soluciones para su implementación, también pretende obtener los costes necesarios para la realización de dicha tarea y así tener una medida de la viabilidad del proyecto.

Primero se realizará un estudio "empírico del juego" para saber sus mecánicas, los controles, los modos de juego, etc. Se descompondrá el juego en módulos que sean más fáciles de abordar. Para cada módulo resultante se estudiará la forma de implementarlo con el motor Unity3D de una manera que sea fácil de extender para ganar tiempo y costes.

Una vez se tienen claras las soluciones, se hará una implementación parcial del juego. Esto significa que se implementará la base del juego como por ejemplo la dinámica del personaje principal, el sistema de puntuaciones y menús, pero no se realizará la implementación de todos los niveles y contenidos, ya que el objetivo es saber el coste de implementación, no la implementación en si.

Finalmente, a partir de las etapas anteriores se obtendrán los costes que supone realizar la implementación completa del videojuego.

1.3 Objetivos del proyecto

- Estudio del juego Narcís Monturiol realizado en Flash y descomponerlo en módulos.

- Analizar los módulos resultantes y encontrar una implementación para el motor Unity3D

- Realizar la implementación de los módulos básicos del juego y la parcial de los niveles y contenidos del juego.

- Realizar un estudio de los costes que conllevaría la implementación total del juego a partir de las soluciones encontradas.

1.4 Estructura de la Memoria

Primeramente, encontraremos el capítulo 2 donde se describe un subconjunto de las posibilidades que ofrece Unity como motor gráfico, sobre todo las funcionalidades que se pueden utilizar para la implementación de este juego. De esta forma se estará más familiarizado de cara a la comprensión de los siguientes apartados.

En el capítulo 3, se procede a hacer el análisis del juego, del cual se obtendrán los requisitos para la implementación. Este capítulo hace un recorrido por todas las partes en las que se ha dividido el juego. Seguidamente se encuentra el capítulo 4 que es un resumen de los requisitos obtenidos del análisis, en él se presenta de forma esquemática

los componentes del videojuego a implementar y la solución Unity que se usará.

El capítulo 5 explica detalladamente las soluciones implementadas en Unity para cada parte del videojuego. El capítulo 6 contiene la planificación y coste de las etapas del desarrollo de este proyecto, en función de las horas dedicadas.

Finalmente, los capítulos 7, que contiene las conclusiones obtenidas, y el capítulo 8 donde se explican ampliaciones futuras que podría tener el videojuego.

Capítulo 2 – Descripción de Unity

Unity es un motor de desarrollo de videojuegos creado por Unity Technologies. Está disponible como plataforma de desarrollo para Windows, Os X y Linux y, como gran baza, tiene soporte de compilación para muchas plataformas, como por ejemplo: Web, PC, dispositivos móviles, Smart TV, Consolas y dispositivos de realidad virtual.

La primera versión se lanzó en la Conferencia Mundial de Desarrolladores de Apple en el 2005, teniendo la capacidad de generar proyectos sólo para plataforma Mac. Con la idea de llamar la atención de desarrolladores más grandes y estudios independientes se lanzó en 2010 Unity 3, que disponía de herramientas de desarrollo hasta el momento sólo accesibles a los grandes estudios. Actualmente va por la versión 5 que fue lanzada en 2015.

El motor gráfico utiliza Direct3D para Windows y Xbox, OpenGl en Mac y Linux, OpenGl ES para Android e iOS e interfaces propietarias para consolas, como Wii.

El scripting está construido en Mono, que es la implementación open-source de .Net. Los Programadores puede utilizar C#, Java Script o Boo para la realización de sus proyectos.

Inicialmente Unity era un entorno de desarrollo de juegos 3D, se podían desarrollar entornos y juegos 2D adaptando los parámetros del juego para simular 2D, como por ejemplo: cámaras ortográficas, texturas planas, etc. Con las últimas actualizaciones el desarrollo de juegos 2D ya se soporta de forma nativa disponiendo de objetos específicos para este modo: cámaras 2D, física 2D, sprites, etc.

Unity tiene una gran acogida entre los desarrolladores independientes y por lo tanto dispone de una gran comunidad. Los recursos: tutoriales y foros, contienen una gran cantidad de datos para aprender a desarrollar en esta plataforma.

También dispone de una tienda de recursos en la que cualquier programador puede publicar sus desarrollos llamada **Asset Store**. Los Assets tienen distintas categorías, por ejemplo: modelos 3D, texturas, materiales, scripts, extensiones para el editor, etc. Existen Assets de pago y gratuitos y pueden ser utilizados en cualquier proyecto de Unity, lo que hace ahorrar tiempo de implementación.

1.2 Licencias

Hay dos licencias principales para desarrolladores: Unity personal y Unity Professional.

La versión gratuita es un poco más limitada y muestra una pantalla de bienvenida en los juegos independientes y una marca de agua en los juegos web que no se puede desactivar. Sólo puede usarse si la facturación de la empresa no supera los 100.000$ anuales

La licencia Pro cuesta 1.500$ por persona. Permite utilizar todas las funciones de Unity hasta en 2 ordenadores de la misma persona. La versión Pro contiene sobre todo mejoras para el desarrollo 3D, efectos, texturas y rendimiento. Aquí están incluidas las licencias para Windows Phone y BlackBerry, pero no las de IOS y Android que tienen un precio extra de 1.500$. Estas licencias son necesarias para el uso de Sockets .Net, que sirven para desarrollar juegos multijugador en juego real. La suma de costes es alta y es la principal barrera de esta tecnología, pero compensa por el ahorro de tiempo en desarrollo y la calidad final del producto.

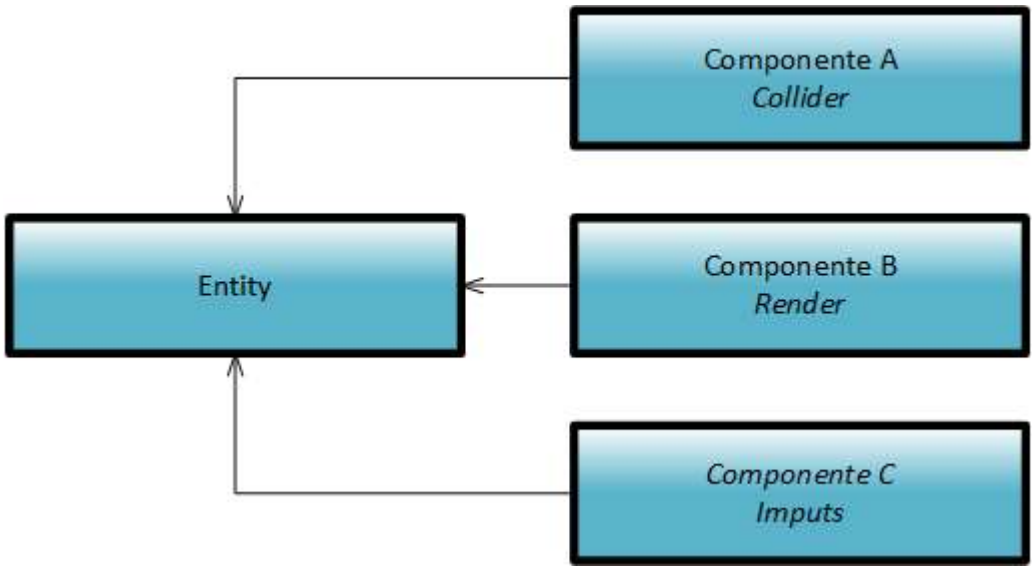
Además de la licencia de Unity, si se quiere publicar para consolas habrá que pagar algo más en función de la plataforma para que se quiera desarrollar.

Existen licencias educativas con la estipulación de que es para la compra y uso de las escuelas, exclusivamente para la enseñaza.

1.3 Arquitectura del software

Unity está basado en el concepto Entity Framework, aportando beneficios como: la utilización de componentes por encima de la herencia, mantiene clases pequeñas centradas en una sola responsabilidad, promoviendo por tanto la modularidad.

El concepto Entity Framework está inspirado en el patrón de diseño Entity-Component. Una entidad contiene múltiples dominios. Para mantener los dominios aislados, el código de cada uno se coloca en su clase correspondiente. Al final la entidad se reduce a un contenedor de componentes.



Patrón Entity-Component

Las entidades de Unity se llaman ***GameObjects*** y los componentes están basados en componentes propios de Unity o en implementaciones de scripts ***Script Component***.

Los scripts son la principal herramienta del programador, a través de ellos se puede acceder y modificar otras entidades y componentes, es decir se define el comportamiento de cada componente. La lógica del videojuego se realiza a través de scripts, cada script estará encargado de una tarea concreta, como por ejemplo, el movimiento de un personaje, el control de la cámara.

Todas las entidades se instancian en una ***Escena***. La escena representa el mundo virtual del videojuego, es lo que ve el jugador cuando se ejecuta el programa, para ello cada escena dispone de una entidad con el componente cámara que es la encargada de qué y cómo se ve la escena.

Algunos de los componentes más importantes de Unity y que se han utilizado para la realización del videojuego:

- **RigidBody:** Controla la posición de un objeto a través de la simulación de físicas. Si se quiere aplicar cualquier tipo fuerza a un objeto del juego debe contener este componente.

- **Collider:** este componente sirve para la detección de colisiones. Las colisiones se producen a partir de objetos que tengan este componente añadido.

- **Sprite Renderer:** Renderiza imágenes 2D llamadas sprites.

- **Script:** los scripts se encargan de la lógica del videojuego, son para programar el comportamiento de las entidades del juego. Unity ofrece como lenguaje: c#, java script y Boo.

- **Cámara:** es el componente que se encarga de renderizar la vista de la escena.

- **Componentes de interfaz de usuario:** hay unos componentes especializados para crear interfaces de usuario: botones, cavas, paneles, imágenes, etc.

1.3.1 Funciones de evento

Para ejecutar los scripts, Unity se encarga de llamar, para cada script, a una serie de funciones predefinidas cada cierto tiempo, alguna de ellas solamente en un momento determinado, una vez ejecutadas estas funciones el control vuelve a Unity.

Unas de las funciones más importantes son por ejemplo:

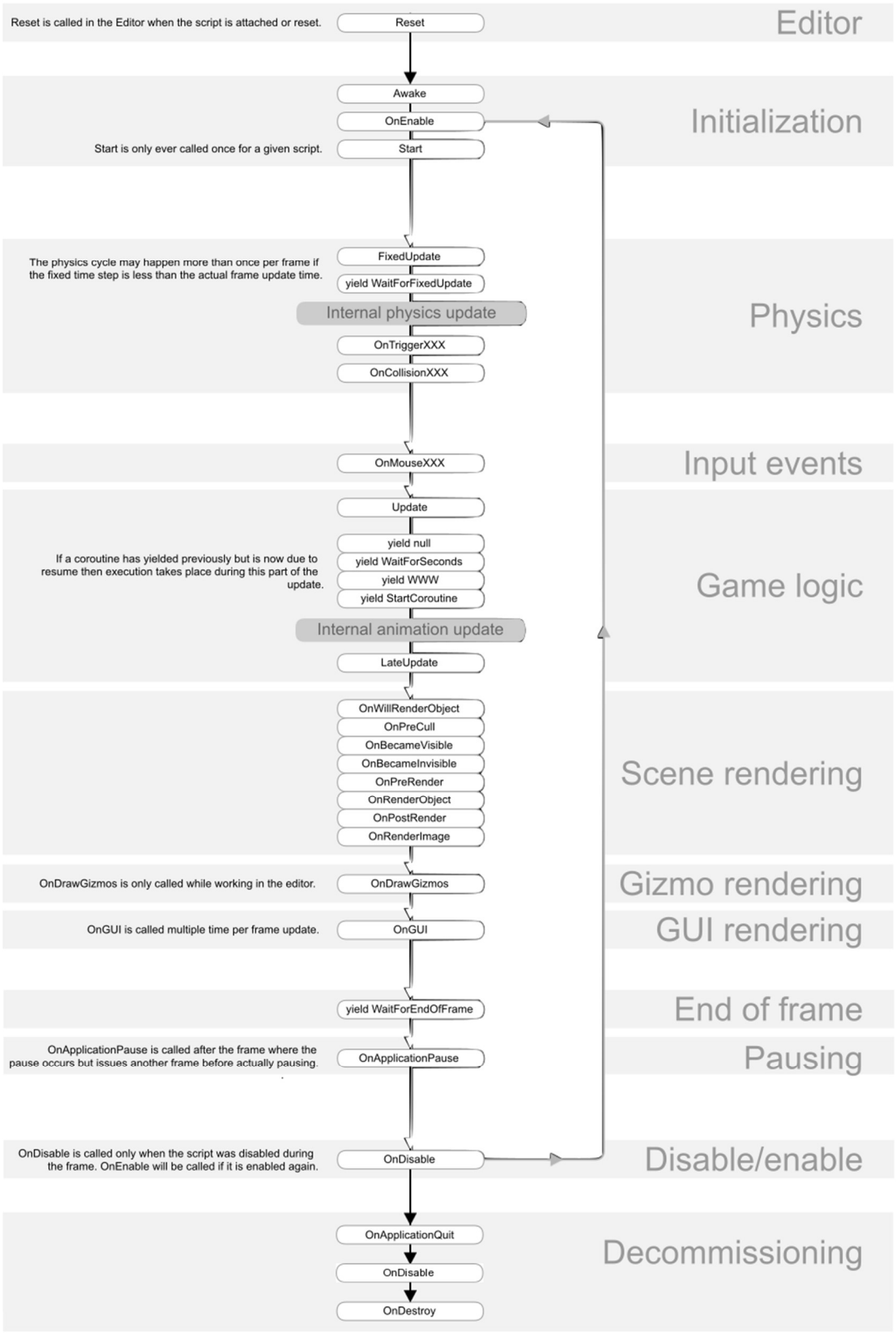
- **Update:** esta función se llama antes de la actualización de cada frame, siendo un frame una vuelta completa a la lógica de un videojuego. En esta función se suele poner la lógica general del juego, por ejemplo, la lectura de los inputs de usuario. La llamada a esta función depende del frame rate a la que se ejecuta el juego.

- **FixedUpdate:** esta función se llama más veces que Update, es independiente del frame rate del juego. Se utiliza para trabajar con las físicas del juego.

- **Start:** esta función se llama al iniciar el juego, utilizada generalmente para la inicialización de propiedades.

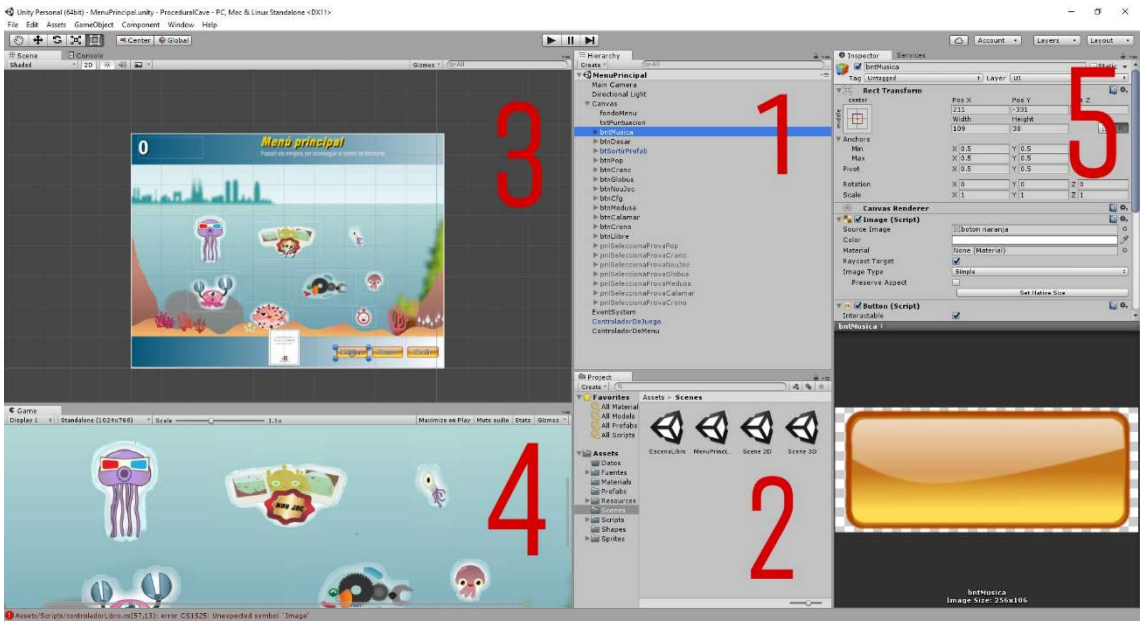
- **Awake:** esta función se llama la primera vez que se ejecuta un script, a diferencia del start, esta se puede llamar más de una vez, por ejemplo, cada vez que se reactiva un objeto del juego.

A continuación, se expone el diagrama con el ciclo de vida de un script:



1.3.2 Interfaz de usuario del editor

Unity cuenta con una interfaz de usuario muy intuitiva y personalizable, a continuación, se explican las partes más importantes.



1. Ventana **Jerarquía,** Lista jerárquica de los elementos que contiene la escena activa.

2. Ventana **Project,** lista todos los elementos del proyecto. Permite ordenar de forma sencilla la aplicación. En esta vista se encuentran las imágenes, scripts, audios, texturas, prefabs y todos los elementos que se podrá utilizar en el juego.

3. Ventana **Escena,** sirve para diseñar las escenas del juego. En esta ventana se pueden añadir los objetos que formarán parte de las escenas: modelos 3D, sprites, animaciones, cámaras, luces, etc. La interfaz es muy sencilla, se basa en un sistema de arrastrar y soltar, de esta manera se añaden los elementes y se cambian sus posiciones y atributos.

4. Ventana **Juego,** esta ventana visualiza como se verá el juego. Es posible cambiar a distintas resoluciones o relaciones de aspecto.

5. Ventana **Inspector,** muestra y define las propiedades de los elementos de la escena. Permite añadir distintos componentes a los objetos que forman parte el juego, como por ejemplo: scripts, colliders, texturas, etc.

1.4 Sistema de interfaz de usuario

Unity tiene un sistema para crear interfaces de usuario (UI) de forma rápida e intuitiva, seguidamente se explican los componentes que se han utilizado para la creación de los menús del videojuego.

1.4.1 Canvas

El canvas es un área que contiene todos los elementos de la UI, todos los componentes de la UI tienen que depender directamente de un objeto canvas.

A la hora de renderizar los objetos de la interfaz Unity se basa en el orden que ocupan dentro del canvas, un elemento que se dibuje antes que otro quedará tapado por el segundo.

1.4.1.1 Modos de renderizado del canvas

El canvas se puede dibujar en la pantalla de distintas maneras:

- **Screen Space – Overlay:** este modo el canvas se dibuja por encima de todos los elementos de la escena.

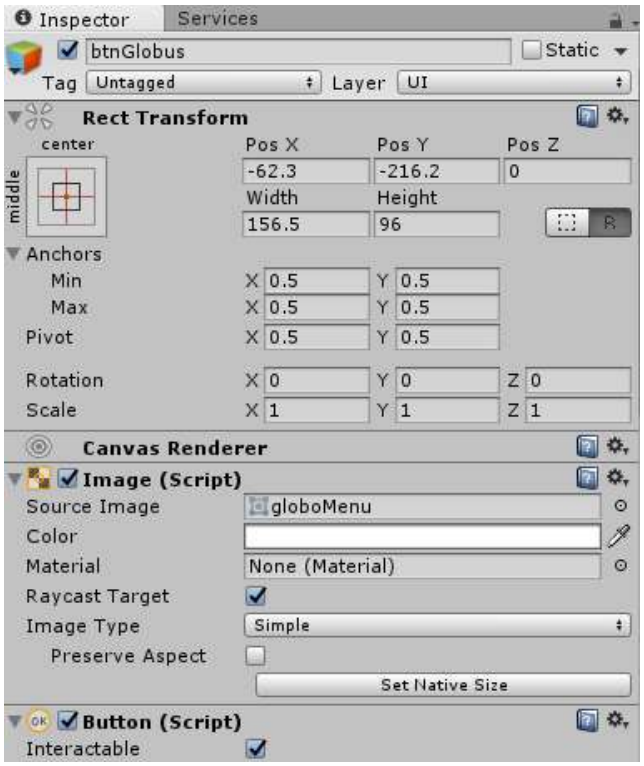
- **Screen Space – Camera:** este modo hace que el canvas se vea afectado por las características de la cámara, lo que hace que también se vea afectada la UI.

- **World Space:** con este modo el canvas se renderiza como un elemento mas de la escena, es decir, en lugar de estar posicionado en el primer plano de la pantalla, el canvas tiene una posición dentro del mundo del videojuego, profundad, altura y anchura.

1.4.2 Button

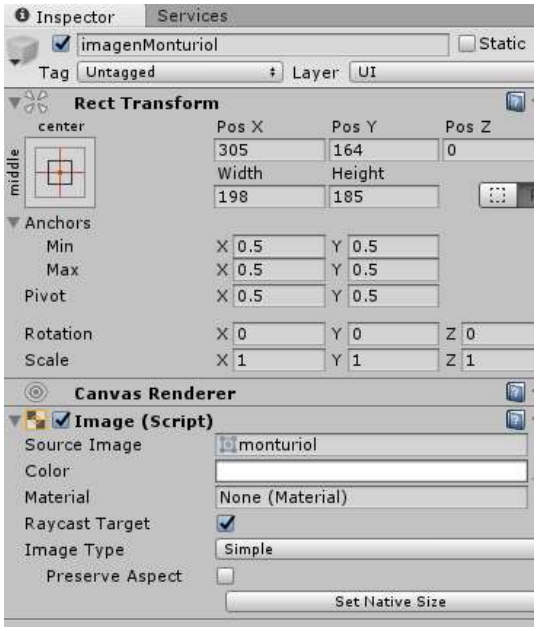
Es el botón típico de una interfaz de usuario, responde a los eventos de pulsado al cual se le puede asociar la llamada a una función de un script o a una propiedad de un elemento. El botón tiene varios estados: Normal, seleccionado, pulsado y desactivado.

Es bastante personalizable disponiendo de varias propiedades, como por ejemplo, el tamaño, color, imagen, etc.



1.4.3 Image

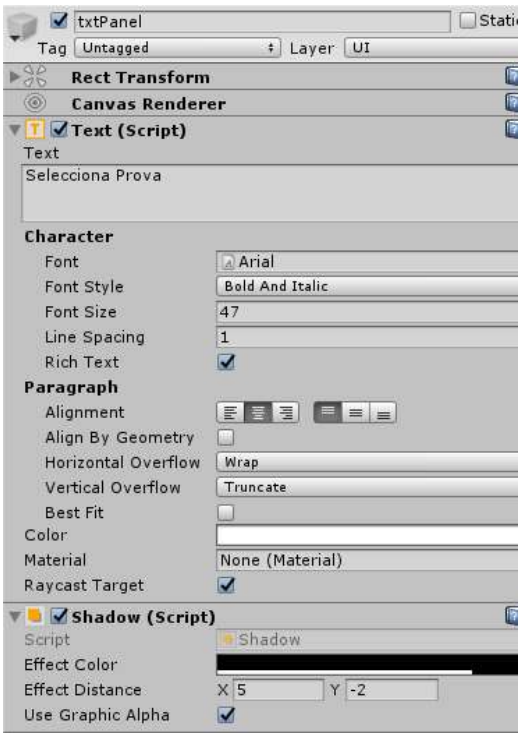
Es un contenedor para imágenes, se le pueden modificar distintas propiedades.



1.4.4 Text

Es un contenedor para mostrar el texto de la UI. A parte de la medida, se le puede asignar una tipografía concreta, un color y añadir componentes como sobras y otros efectos.

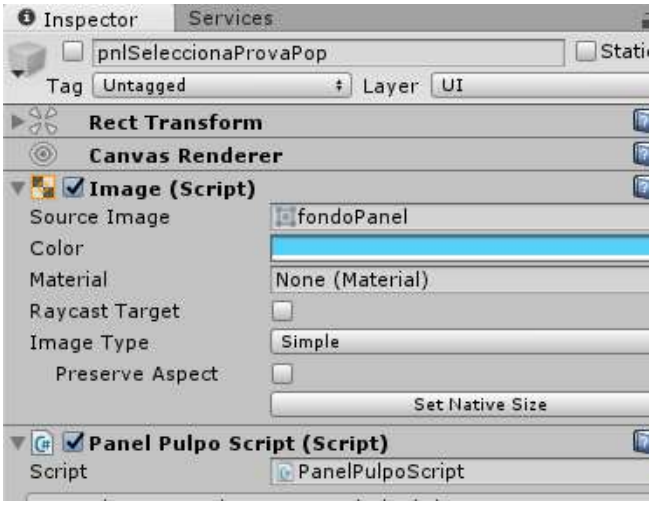
A continuación, se muestra la captura de las propiedades de un texto con un componente sobra añadido.



1.4.5 Panel

Es un contenedor para incluir elementos de UI, un canvas puede tener distintos paneles. Es útil para agrupar distintos componentes de la interfaz de usuario, de esta manera es sencillo, por ejemplo, activar o desactivarlos todos, o crear un script a nivel de panel que modifique el comportamiento de los componentes incluidos.

Otra propiedad utilizada del panel es la posibilidad de asignarle una imagen de fondo.

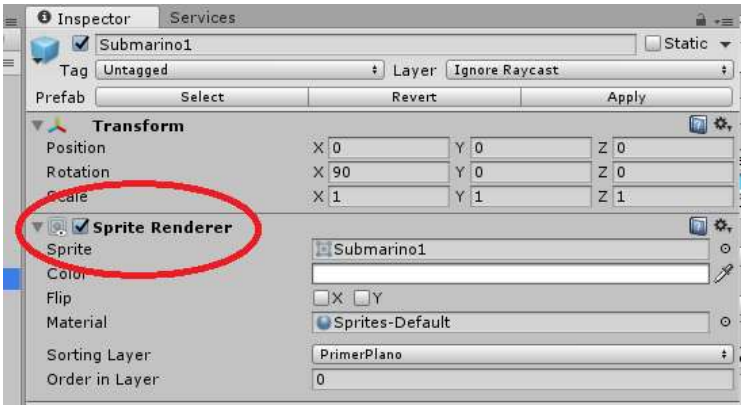


1.4.6 Slider

Es una barra de desplazamiento que representa un conjunto de valores acotados, a medida que la barra se desplaza su valor cambia dentro del dominio que tiene definido.

1.5 Orden de renderizado

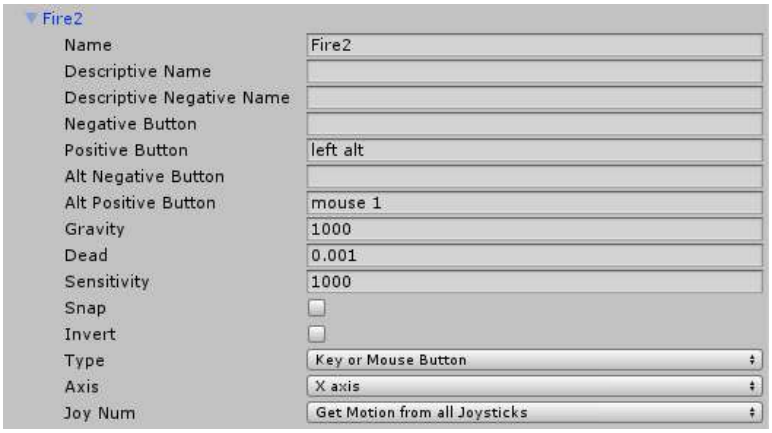
En un juego 2D, para que Unity sepa el orden en el que se tienen que dibujar los diferentes componentes de una escena existe un sistema de capas llamado ***Sorting Layer***. Cada capa tiene un orden de renderizado, para que un objeto se dibuje en un orden deseado sólo hay que asignarlo a una capa a través de su componente ***Sprite Renderer***.



1.6 Inputs

El sistema para gestionar los inputs del juego se llama ***Input Manager***. Es capaz de obtener la entrada de teclados, joysticks, pantallas táctiles y sensores de dispositivos móviles, como por ejemplo, acelerómetros.

Funciona a través de alias, estos alias tienen diferentes propiedades para asignarles, por ejemplo, una tecla, un eje concreto del joystick, etc. Para obtener el estado de un input concreto a través de un script se utilizará el alias asignado y el objeto ***Input***.



1.7 Sistema de física

Unity contiene un sistema de física integrado que proporciona a los componentes un comportamiento dinámico muy realista cuando se le aplican fuerzas, como por ejemplo, la gravedad, colisiones con otros objetos, caídas, etc. El sistema está separado en componentes para juegos 2D y 3D, pero comparten muchas características.

Para que un componente del juego sea vea modificado por el sistema de física bastará con añadirle el componente **RigidBody**, Automáticamente se verá afectado por la gravedad, que generalmente siempre está activada en el juego.

Estos son los diferentes tipos de fuerza que se puede aplicar a un RigidBody:

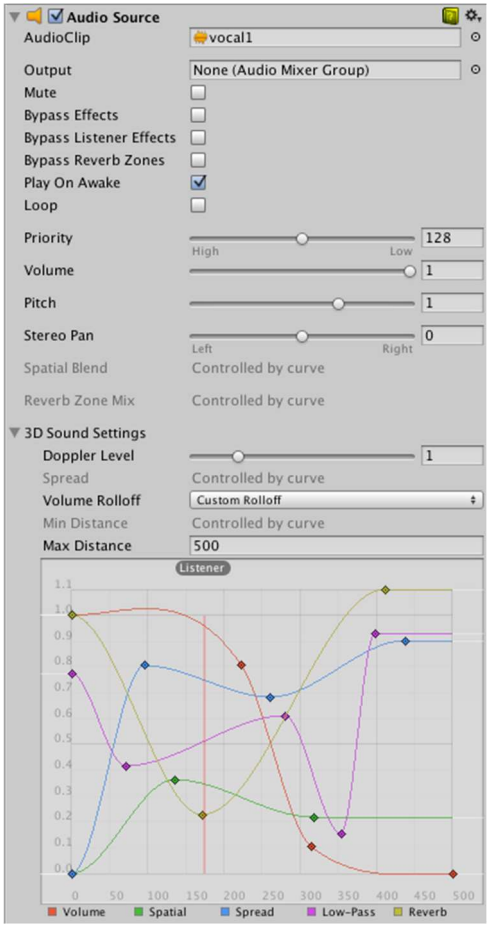
- **Force:** añade una fuerza continua al objeto utilizando su masa.
- **Acceleration:** añade una aceleración continua al objeto ignorando la masa de este.
- **Impulse:** añade una fuerza instantánea al objeto teniendo en cuenta su masa.

- **Velocity:** añade una velocidad instantánea ignorando la masa del objeto.

1.8 Sistema de sonido

Unity contiene un sistema de sonido completo aportando mezclas y masterización en tiempo real, sonido 3D y efectos predefinidos. Es capaz de cargar distintos formatos: AIDD, WAV, MP3, OGG, xm, mod, it y s3m, estos cuatro últimos para música y los primeros para efectos, aunque también estos es posible utilizarlos para música.

Hay varios componentes de sonido: **Audio Clip, Audio Listener, Audio Mixer, Audio Filters, etc**, pero para el juego sólo se utilizará el **Audio Source** para los efectos de sonido y música:



1.9 Prefabs

Los Prefabs son objetos reutilizables creados por el usuario desde la vista de diseño a partir de objetos y componentes básicos de Unity, es como si se tratase de una plantilla o una clase de POO.

Estos Prefabs tiene características definidas, comportamientos, componentes añadidos, etc. Una vez diseñados se pueden instanciar en el juego y ser reutilizados las veces que sean necesarias. Las características de todas las instancias de un Prefab pueden ser modificadas a la vez, por ejemplo, para cambiar el tamaño o el color de un sprite.

Por poner un ejemplo, se puede diseñar un enemigo de un juego que contenga su sprite, su collider, un comportamiento programado a través de script, etc. Después durante la ejecución del juego se puede utilizar este prefab para instanciar tantos enemigos como sea necesarios.

Capítulo 3 – Estudio y análisis

En este capítulo se describirá en detalle el estudio que se ha realizado del videojuego que se tiene que pasar a Unity: Narcís Monturiol. Este estudio se ha realizado directamente sobre el juego, probando la jugabilidad, viendo los contenidos y revisando todas sus mecánicas y funcionalidades que ofrece.

En el estudio se ha tratado de definir claramente las partes del videojuego para trabajar con ellas de forma modular y así poder acotar el esfuerzo que se tiene que dedicar a cada una de ellas.

3.1 Descripción del videojuego

Narcís Monturiol es un juego de plataformas concebido para el aprendizaje de chavales de primaria, el juego ofrece un contenido educativo basado en el personaje histórico Narcís Monturiol, inventor catalán conocido por haber construido uno de los primeros submarinos de la histórica **l'Ictineu**.

A medida que el juego transcurre y el jugador va superando las distintas fases, se van desbloqueando contenidos en forma de libros virtuales. Al terminar una fase del juego se muestran preguntas al jugador para poner en práctica los conocimientos adquiridos a través de estos libros, si el jugador acierta conseguirá puntuaciones extra. Hay incluso un tipo de subjuego que se trata de avanzar contestando preguntas sobre los libros en un tiempo limitado de tiempo.

A continuación, se describirán todas las partes en las que se ha descompuesto el videojuego.

3.1.1 Menú principal

El menú principal permite al jugador acceder a los niveles de los distintos mini juegos, la configuración y acceder a los libros de contenidos. A continuación, se listan los diferentes elementos que lo componen:

- Fondo de pantalla.

- Botones animados que permiten acceder a los mini juegos (1, 2, 3, 4, 6, 7, 8).

    - **1.Pulpo:** da acceso al juego “Explorar”.
    - **2.Cangrejo:** acceso al juego “Pescador”.
    - **3.Pez globo:** acceso al juego “Peix Globus”.
    - **4.Nuevo Juego:** acceso al juego “Gos d’atura”.
    - **6.Medusa:** acceso al juego “Passejar”.
    - **7.Calamar:** acceso al juego “Preguntes”.
    - **8.Crono:** acceso al juego “Cursa”.

- Botón animado para acceder a la configuración del personaje (submarino) (5).

- Botones normales para salir del juego, desactivar la música y guardar el contenido (10, 11, 12).

- Botón para acceder a los libros de contenidos (9).

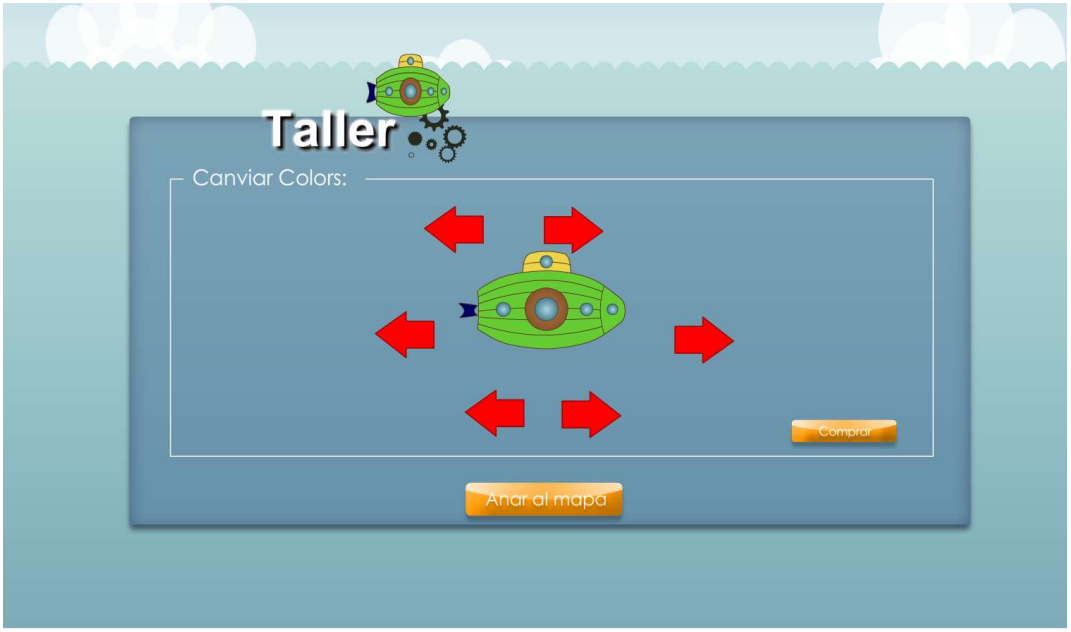
- Puntuación acumulada del jugador (13).
  Cuando un botón de mini juego se clica, aparece una pantalla para seleccionar el nivel deseado:

Al clicar un botón de este panel el juego se pondrá inmediatamente en marcha en el nivel seleccionado. El botón “Sortir” simplemente cierra el panel.

En el caso de clicar el botón de configuración, aparece una nueva ventana donde se puede personalizar el submarino:

A nivel de comportamiento, vida, velocidad, aceleración, oxígeno y carga:

A nivel estético, cambiando los colores:



El botón del libro permite acceder a los libros que se han ido desbloqueando durante el juego, al clicarlo aparece una primera pantalla con los libros disponibles:

Si se clica sobre un libro, se abre una nueva pantalla donde se puede consultar su contenido de forma interactiva, pasando hacia adelante y hacia detrás las páginas, para ello existen dos botones “Endarrera” y “Endavant”. También existe la posibilidad de pasar página con el ratón como si de un libro real se tratara, para ello usa un efecto 3D muy conseguido que tiene pinta de ser un control desarrollado específicamente para estos menesteres.

Estos libros incluyen su portada, contra portada y diversas páginas. En el contenido de las páginas podemos encontrar:

- Imágenes.
- Textos.
- Imágenes con texto.
- Animaciones.
- Fondos.

Finalmente tenemos los botones de función:

- “Sortir”, para salir del juego.
- “Desar”, para guardar el avance del juego.
- “Música”, que desactiva la música del juego.

Al principio del juego no todos los elementos del menú están desbloqueados, para desbloquear los juegos y los libros el jugador tiene que ir superando pantallas.

Diagrama de transiciones del menú principal

Diagrama del menú principal:

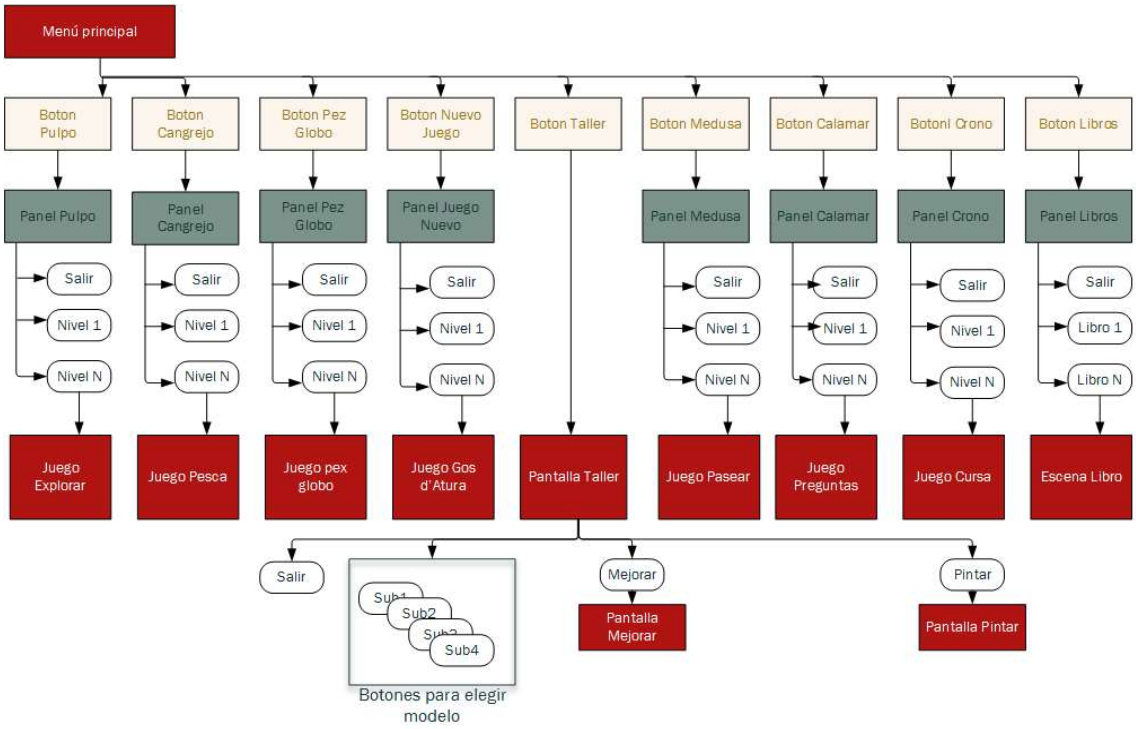


Diagrama de la pantalla de pintar el submarino:

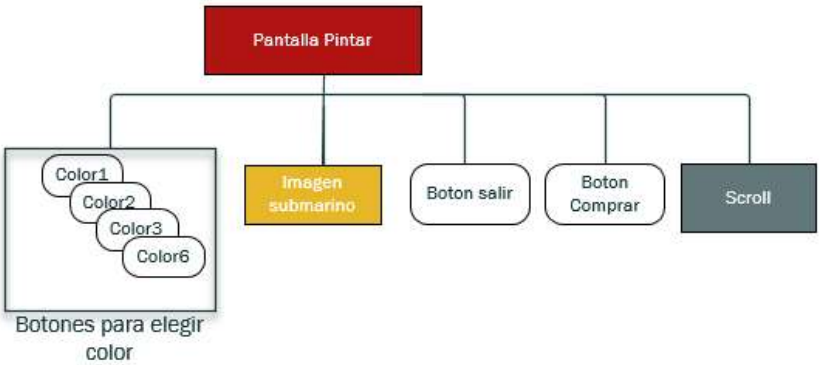
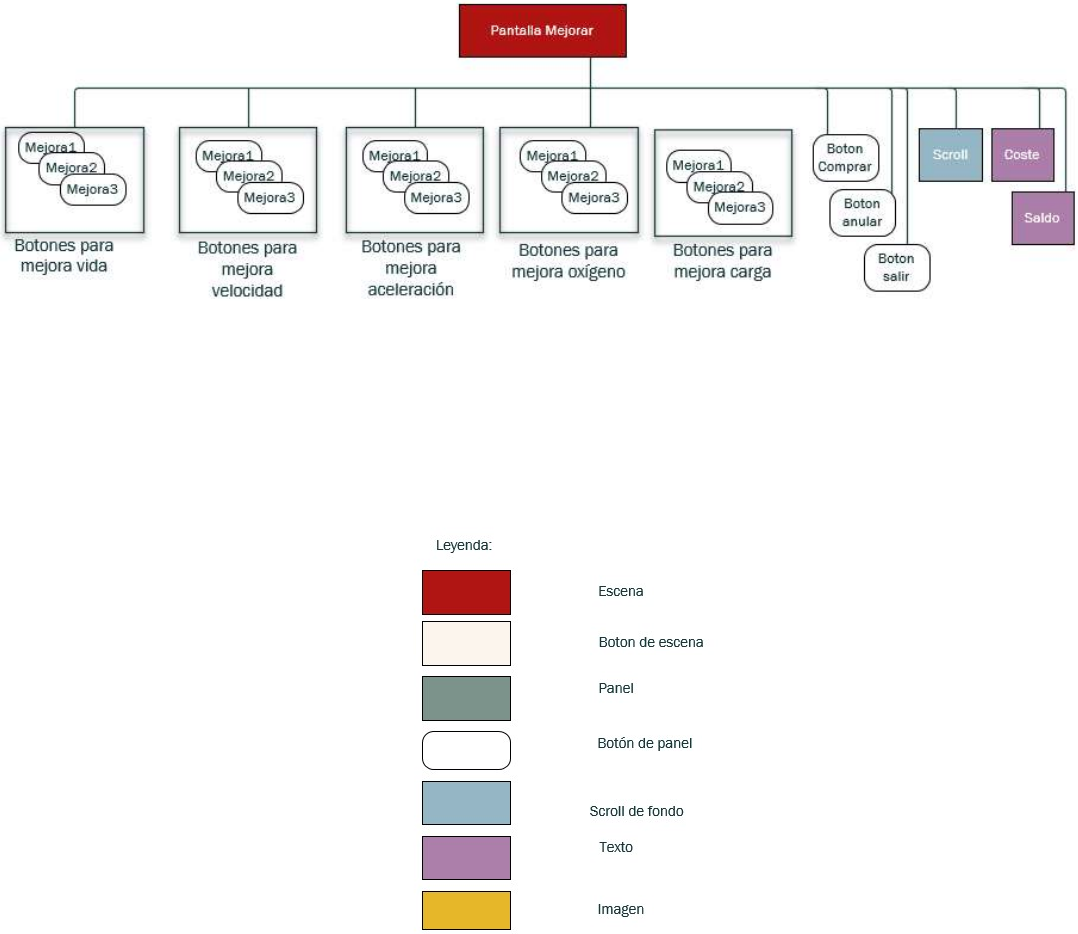


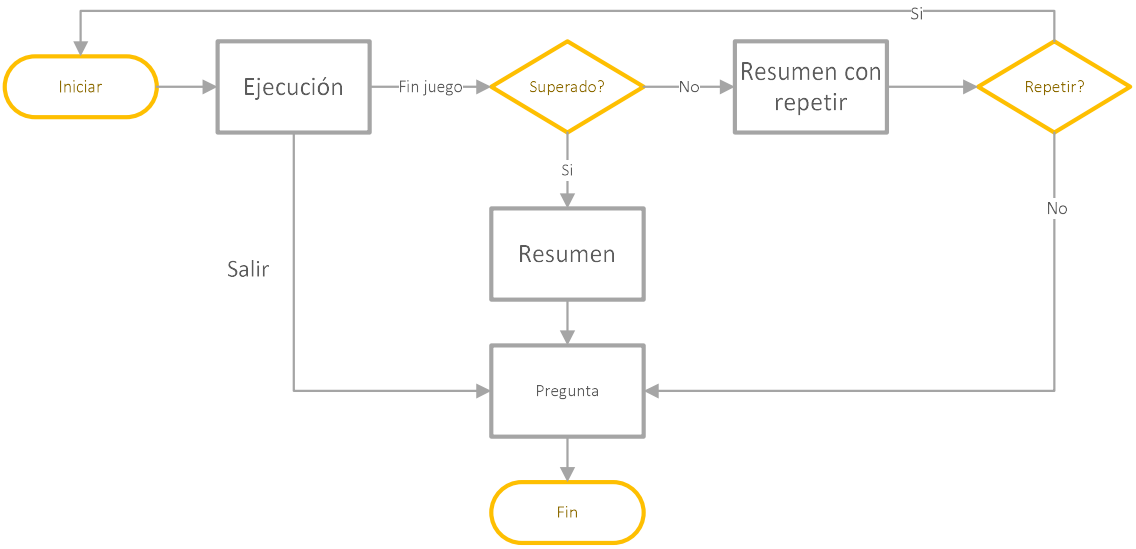
Diagrama de la pantalla mejorar el submarino:



3.1.2 Modos de juego

A continuación, se describen los modos de juego disponibles. El estudio se hizo a partir de observar el comportamiento del submarino y la mecánica del juego.

Todos los juegos responden a este diagrama de estados:



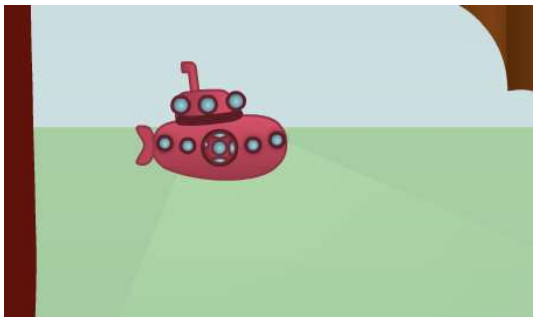
3.1.2.1 Explorar

Este modo de juego trata de manejar el submarino por un mapa que simula cuevas en forma de laberinto parcialmente sumergidas en el mar. El objetivo es encontrar la salida del mapa y conseguir el máximo número de puntos. Para ello el jugador dispone de tiempo limitado, un máximo de 120 segundos.

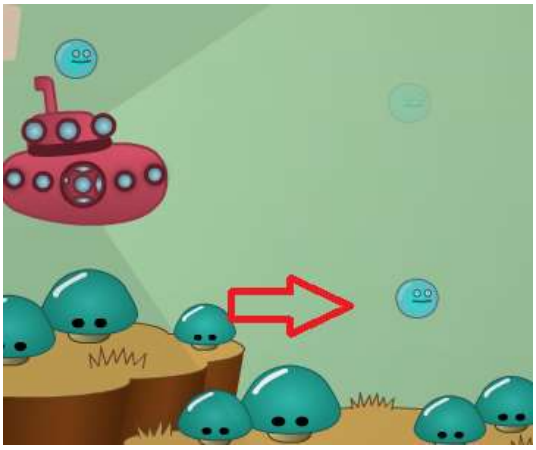
Además del tiempo el submarino dispone de una cantidad de oxígeno limitada que irá perdiendo mientras permanezca sumergido en el agua, esta cantidad está representada por una barra en el hud. A medida que se acaba el oxígeno la barra va disminuyendo hasta que se acaba. Si el oxígeno llega a cero se acaba la partida.

Hay dos formas de reponer el oxígeno perdido:

- subiendo a la superficie.



- O cogiendo burbujas que se generan en sitios concretos del mapa.



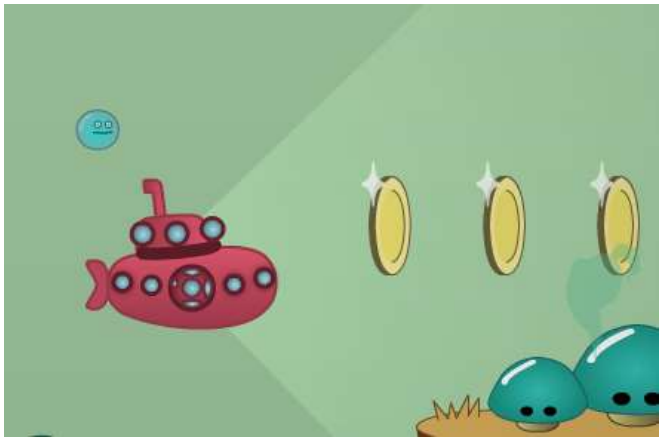
El submarino tiene una cantidad de energía representada también por una barra, si el submarino choca contra las paredes de las cuevas irá perdiendo energía y la barra disminuirá. Si la energía se agota, el juego acaba. No hay forma de reponer la energía del submarino.

Este modo de juego también tiene enemigos que quitan vida al submarino si chocan con él. Los enemigos se explican en su propio apartado ya que aparecen en más de un mini juego.

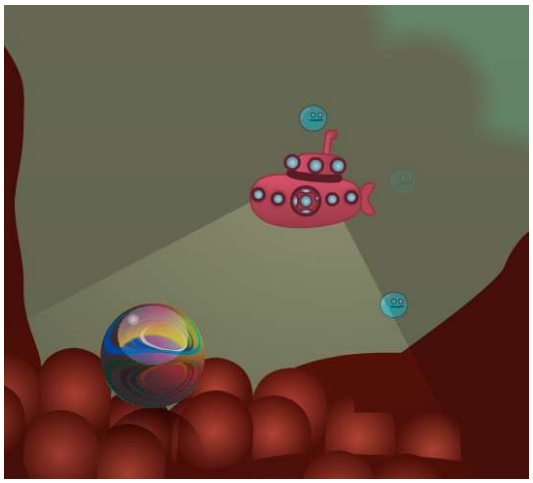
Existe también, a parte de los enemigos, un personaje que es un pulpo, si el submarino se acerca a él mostrará una tarjeta con información que es útil para responder las preguntas del juego y ganar así más puntos.

Para obtener puntos el jugador puede hacer lo siguiente:

- Recoger monedas que dan una puntuación de 10.



- Encontrar una perla que está en algún lugar del mapa, al tocarla hay que contestar una pregunta que, en el caso de contestarla bien, da 50 puntos al jugador.



Al final del juego se muestra un panel con la cantidad de puntos obtenidos y avisando de si se ha conseguido pasar o no la prueba. Este panel permite repetir la prueba o salir al menú principal.

Elementos del hud

- Puntos conseguidos.
- Título con el número de nivel que se está jugando.
- Tiempo restante del nivel

- Oxígeno y vida.

Elementos del juego

- Mapa. En este caso el mapa es un conjunto de cuevas en forma de laberinto más o menos extenso, el submarino recorre el mapa y se va haciendo scrolling a medida que se mueve.

- Submarino.

- Enemigos. Explicados en su apartado.

- Corriente. Es un elemento situado en el mapa que cada X tiempo se activa viéndose como una animación de corriente. Si el submarino pasa cerca sale disparado en dirección a la corriente.

- Monedas. Es un objeto con la imagen de una moneda rotando. Al pasar por encima el submarino el objeto desaparece y aumente la puntuación del jugador.

- Perla. Es un objeto con forma de esfera, tienes dos estados: activado y desactivado. Cuando el submarino se acerca muestra una pregunta y luego se desactiva. Si el jugador acierta la pregunta se le suman 50 puntos, en caso contrario cero.

- Pulpo. Es un objeto con una imagen de un pulpo. Permanece quieto en una posición del mapa. Si el submarino se acerca tiene que mostrar una tarjeta con contenido educativo del juego.

Controles

En este mini juego el submarino se mueve con las flechas del teclado. El submarino al avanzar tiene una aceleración y una velocidad máxima. Esto se tendrá en cuenta de cara a la implementación del comportamiento.

3.1.2.2 “Pescador”

Este juego trata de transportar a unos personajes con forma de cangrejo hasta un portal mientras estrellas de mar persiguen el submarino para tratar de impedirlo. Los cangrejos se desplazan por el fondo de la pantalla horizontalmente de izquierda a derecha sin parar para que sea un poco más difícil de atrapar. Las estrellas de mar hacen que sueltes el cangrejo si te tocan, haciendo que se tenga que volver a buscar.

Para que el submarino pueda transportar los cangrejos dispone de un rayo tractor que al ponerse encima del cangrejo lo atrae y lo puede llevar volando hasta el portal.

El objetivo de cada nivel consiste en recoger un número específico de cangrejos y llevarlos al portal, para ello el jugador dispone de un tiempo máximo de 40 segundos. A medida que se avanzan niveles, el número de cangrejos para pasar de pantalla y el de enemigos aumenta.

Elementos del hud

- Numero de cangrejos recogidos
- Oxígeno y vida. En este caso la vida y el oxígeno no disminuyen
- Número objetivo de cangrejos a recoger.
- Título con el número de nivel que se está jugando.

Elementos del juego

- Cangrejos. Se mueven horizontalmente en el fondo de la pantalla. A medida que se van recogiendo otros nuevos van apareciendo. Existe dos tipos de cangrejos se diferencia por el Sprite y por la puntuación que aportan al recogerlos.
- Estrellas de mar. Explicada en el apartado de enemigos.
- Submarino.
- Portal estelar. Es una especie de agujero negro que al dejar los cangrejos los atrae hasta desaparecer. Cada vez que se deja un cangrejo ahí aumenta la puntuación del jugador.

Controles

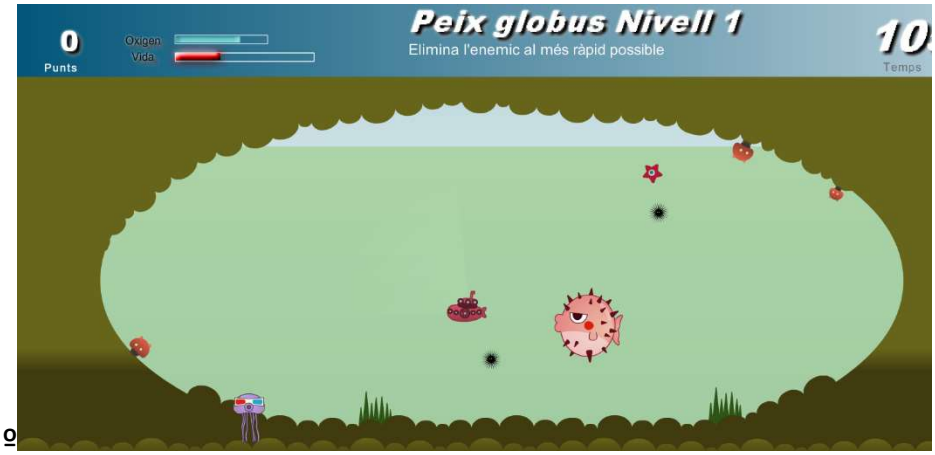
En este juego el submarino sigue el puntero del ratón.

3.1.2.2 “Peix Globus”

Este juego trata de eliminar un pez globo gigante en el menor tiempo posible. Para ello el jugador tiene que dirigir el pez hacia los enemigos

repartidos por el mapa para que se los coma. Si consigue que se coma todos, el pez explotará y habrá ganado la partida. Para ello el jugador dispone de un máximo de 120 segundos.

En este juego el jugador perderá oxígeno mientras el submarino permanezca sumergido. Si choca contra el escenario o contra un enemigo la energía le disminuirá.



o

Elementos del hud

- Puntuación. En este caso no se utiliza.
- Título con la descripción del nivel.
- Tiempo restante de la pantalla.
- Oxígeno y energía.

Elementos del juego

- Mapa. En este caso se trata de un mapa estático, una cueva sin salida parcialmente sumergida para que el submarino pueda salir de agua y recuperar oxígeno.

- Submarino.

- Enemigos: erizos, estrellas de mar y pez globo.

- Pulpo. Al acercarse a él da información de cómo hay que eliminar al pez globo.

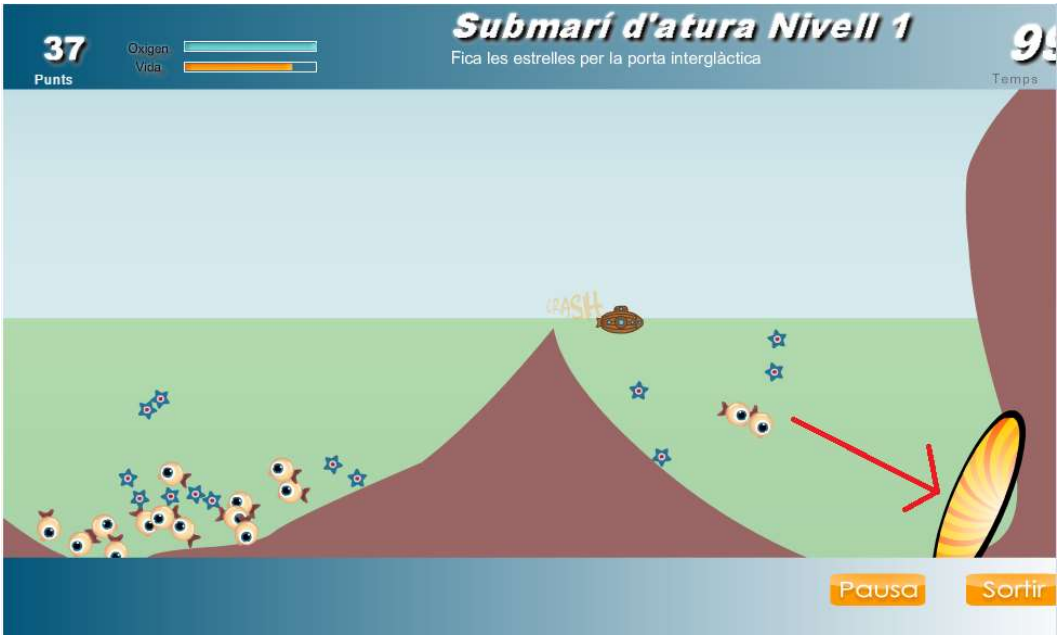
Controles

En este juego el submarino se controla con las flechas del teclado.

3.1.2.3 Gos d'atura

En este juego el jugador tiene que llevar a una serie de personajes, peces y estrellas de mar, que están desperdigados por el mapa hacia una posición en concreto del mapa donde se encuentra una "puerta estelar" que los atraerá hacia ella. Al entrar en la puerta los peces desaparecen y aumenta la puntuación del jugador.

Para poder llevar los peces hacia la puerta, el jugador los que tiene que empujar chocando con el submarino. El objetivo es meter todos los peces y estrellas de mar en la puerta antes de que se consuma el tiempo, 120 segundos.



Elementos del hud

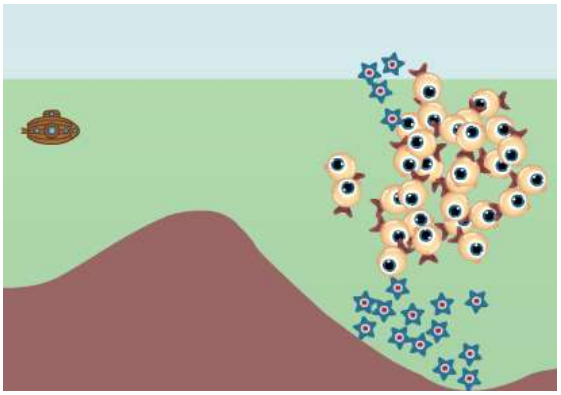
- Puntuación del jugador.
- Título con la descripción del juego.
- Tiempo restante de la pantalla.
- Oxígeno y vida del submarino.

Elementos del juego

- Mapa. Los mapas de este juego son variados, hay cuevas totalmente cerradas con forma de círculo, cuevas laberínticas y mapas muy abiertos. Todos tienen situado en algún lugar la puerta estelar para llevar a los peces.
- Submarino.

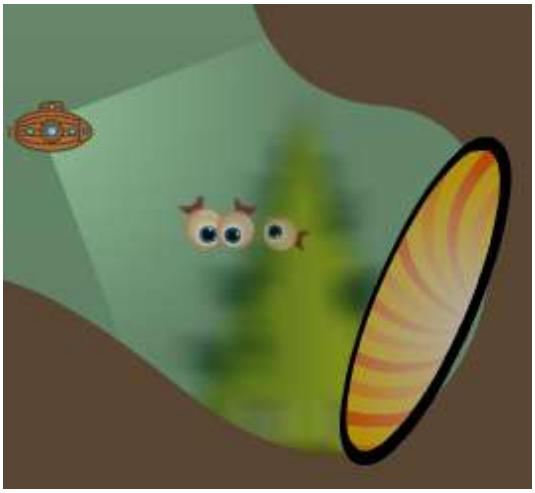
- Peces y estrellas de mar. Los peces están desperdigados por el mapa y reaccionan cuando son chocados por el submarino, estos salen disparados en la misma dirección con la que les choca el submarino. El choque no afecta a la vida del submarino.

  Los peces del mismo tipo se atraen y forman grupos, como si fuese una especia de banco de peces.



  Existen peces que dan una puntuación distinta cuando son llevados a la puerta.

- Puerta interestelar. Es un objeto situado en el mapa que atrae a los peces y al submarino cuando entran en su radio de acción. Los peces cuando entran en la puerta desaparecen y aumentan la puntuación, el submarino es atraído, pero no desaparece.



Controles

En este juego el submarino se controla con las flechas del teclado.

3.1.2.4 “Passejar”

Este modo se trata de un juego “infinite run”. Este tipo de juego se basa en una pantalla en scroll horizontal que no tiene final, se va generando todo el rato, en donde el submarino va navegando mientras le quede tiempo. Durante el avance del submarino aparecen enemigos que le quitan energía, ítems, que aumentan el tiempo o que hacen recuperar vida, y monedas que aumentan la puntación al recogerlas.

El objetivo del juego es durar el suficiente tiempo navegando para obtener una cantidad igual o mayor de puntos especificada en cada nivel. Hay un contador de tiempo que no para de descender, cuando llega a cero el juego acaba. Si el jugador consigue los puntos pasa de pantalla, en caso contrario no.

Elementos del hud

- Puntuación acumulada.
- Título descriptivo del nivel.
- Tiempo restante.
- Oxígeno y vida del submarino.
- Mensajes de información. Cuando el submarino coge algún ítem hay un mensaje encima de la barra oxígeno que indica el tipo de ítem que ha cogido.

Elementos del juego

- Mapa. Para este juego el mapa es un scroll horizontal infinito, no hay cuevas ni paredes con las que se pueda chocar. El mapa está avanzando sin parar horizontalmente a una cierta velocidad, que puede ser más rápida o más lento según la velocidad del submarino.
- Scroll. En el fondo hay un scroll que simula el movimiento del agua.
- Submarino.
- Enemigos. Medusa y estrellas, ambos le quitan vida al submarino, pero la medusa al chocar desaparece, la estrella no desaparece y además empuja el submarino. Ambos tienen un movimiento vertical.

- Corrientes de agua. Al pasar cerca el submarino es empujado en la misma dirección de la corriente.

- Ítems:

    - Monedas. Al recogerlas dan una puntuación de 10 al jugador.
    - Tiempo. Al coger el ítem añade tiempo extra al jugador.
    - Velocidad. Al coger el ítem el submarino aumenta su velocidad máxima.
    - Energía. Al coger el ítem repone energía al submarino.

3.1.2.5 “Preguntes”

Este juego trata de llevar el submarino desde el fondo del mar hasta la superficie antes de que se acabe el oxígeno, para ello el jugador tendrá que responder preguntas cuyas respuestas son caminos por donde navegará el submarino. La pantalla tiene N niveles hasta la superficie, la

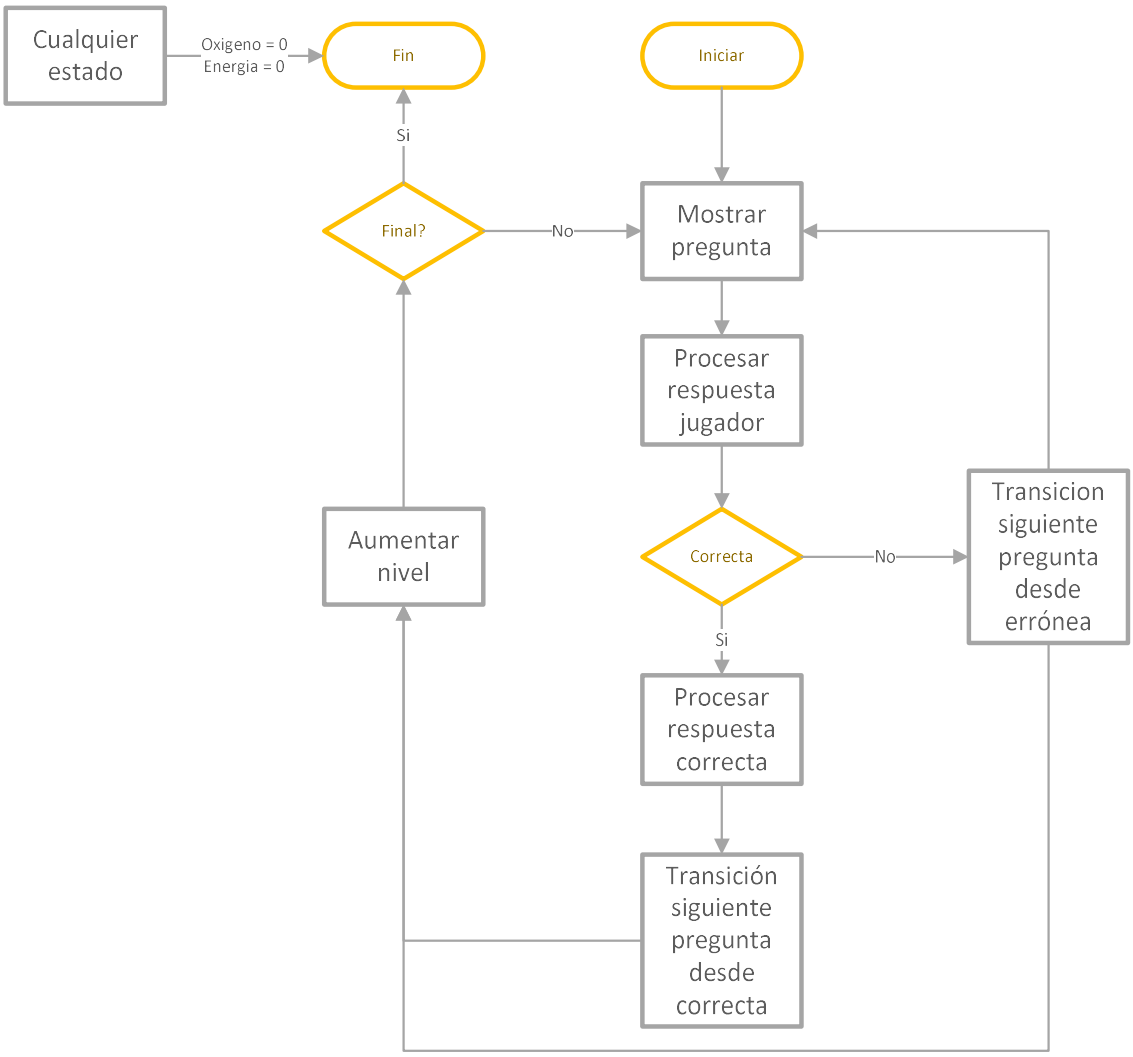
cantidad de niveles son la cantidad mínima de preguntas que tiene que responder el jugador.

Si la respuesta elegida es correcta el jugador ganará puntos y el submarino pasará por un camino hasta la siguiente pregunta sin problemas, además estará un nivel más cerca de la superficie.

Si la respuesta es errónea no se sumarán puntos y en el camino hasta la siguiente pregunta el submarino perderá energía. Es posible que el jugador no suba nivel, ya que hay transiciones de respuestas erróneas que no permiten que el submarino suba al siguiente nivel.

La pantalla se supera cuando se alcanza el último nivel. Si el submarino pierde todo el oxígeno o toda la energía el juego acaba sin ser superada la pantalla.

Diagrama de estados del juego “Preguntes”



Elementos del hud

- Puntuación acumulada.

- Título del nivel del juego, con la información sobre los niveles que faltan por superar.

- Oxígeno y vida del submarino.

Elementos del juego

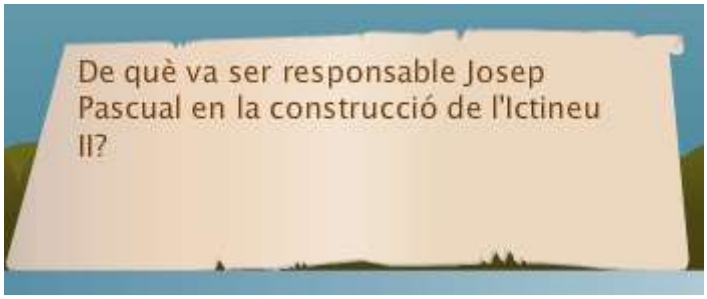
- Mapa. Este juego dispone de dos tipos de pantalla:

    - Las preguntas, que es un mapa cerrado con 4 salidas, cada salida representa una respuesta de la pregunta.

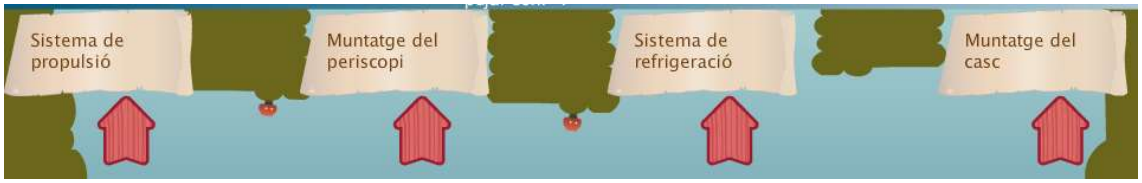
    - Las pantallas de transiciones, es un mapa en el que no puede interactuar el usuario y muestra como el submarino asciende al siguiente nivel.

- Submarino.

- Pregunta. Un panel donde se muestra la pregunta que el jugador tiene que responder.



- Respuesta. Son caminos con un panel en la entrada, el jugador clicará el panel que cree que es la respuesta correcta y el submarino se dirigirá a ese camino.



Controles

En este juego el submarino no se controla directamente. El jugador cuando clica a una respuesta el submarino se dirige automáticamente al camino escogido. En las transiciones al siguiente nivel el submarino se mueve solo.

3.1.2.6 Cursa

Este es un juego de velocidad, se trata de conducir al submarino por un recorrido en el menor tiempo posible. El jugador para pasar el nivel tiene que cruzar todos los puntos de control del recorrido. Hay dos tipos de cursas: un recorrido con puntos de control que se navega desde el primer control hasta el último o un recorrido cerrado en el que hay que dar vueltas, en este último sólo hay un punto de control que es la meta.

El juego guarda la repetición de la mejor vuelta del jugador para que la siguiente vez que juegue el mapa pueda competir contra su “sombra”.

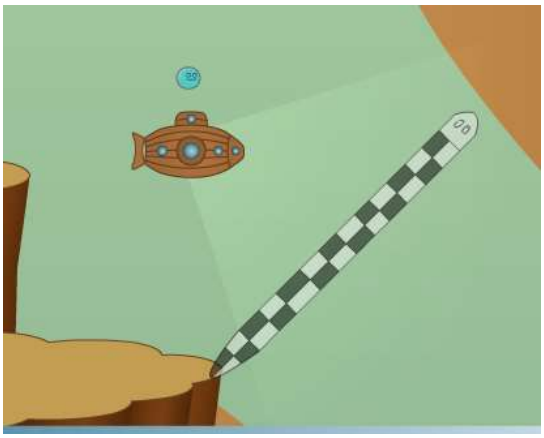
Elementos del hud

- Vueltas o puntos de control que faltan por pasar.
- Cronómetro que cuenta el tiempo.
- Oxígeno y energía del submarino.
- Título explicativo del juego.

Elementos del juego

- Mapa. El mapa es un recorrido, abierto o cerrado por el que tiene que navegar el submarino lo más rápido posible.

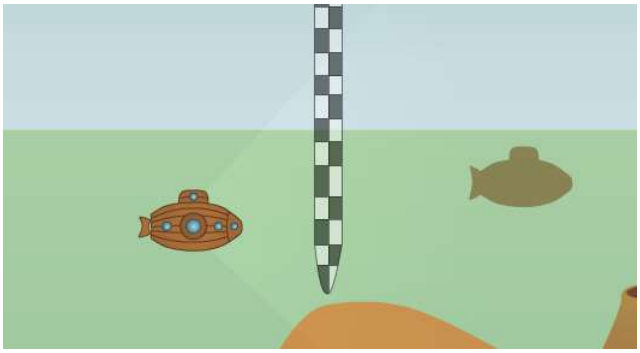
- Puntos de control, son objetos que están en el mapa y tienen que ser atravesados en orden por el submarino para poder acabar la cursa.



- Corrientes. Son corrientes de agua que empujan al submarino si pasa cerca, es un objeto que hay que evitar.

- Cuenta atrás inicial, al iniciar la cursa hay una cuenta atrás de 3 segundos.

- “Fantasma del jugador”. El juego guarda la mejor carrera del jugador. La siguiente vez que el jugador juega esa pantalla se puede ver el fantasma corriendo a la vez.



3.1.3 El submarino

El personaje principal del juego es el submarino Ictineu. El jugador lo manejará durante todos los juegos.

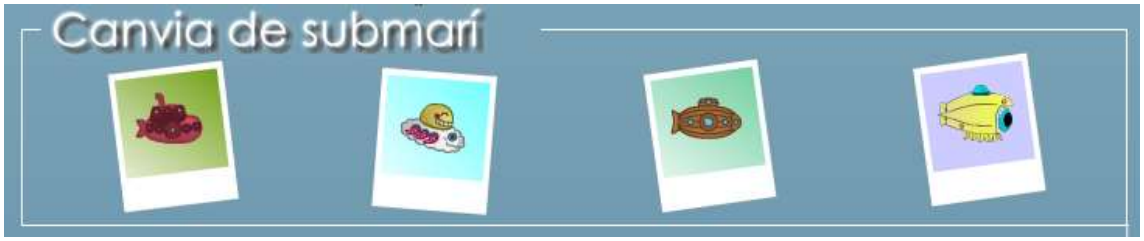
3.1.3.1 Movimiento

Se puede mover mediante las flechas del teclado o el ratón. El movimiento no es lineal, tiene una aceleración y una inercia para simular el movimiento debajo del agua.

Cuando se mueve mediante el ratón el submarino simplemente sigue el puntero de este.

3.1.3.2 Sprite

El submarino tiene cuatro modelos diferentes a elegir.



3.1.3.3 Rayo tractor

El submarino dispone de un rayo tractor para utilizar en algún mini juego específico, el resto de juegos está desactivado. El usuario no puede activar o desactivar el rayo tractor. El rayo tractor tiene una capacidad de carga que se puede mejorar en el taller.

3.1.3.4 Características

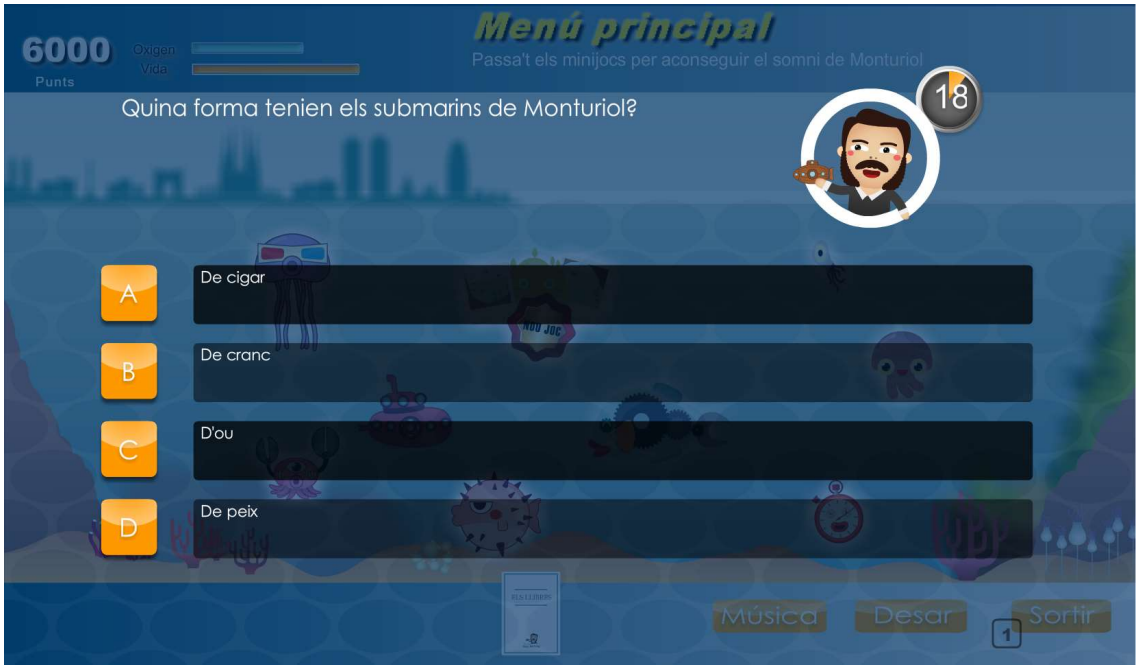
El submarino tiene las siguientes características:

- Energía.
- Velocidad máxima.
- Aceleración.
- Cantidad de Oxígeno.
- Capacidad de carga.

3.1.4 Preguntas

A lo largo del juego se van sucediendo una serie de preguntas, sobre el contenido del personaje Narcís Monturiol, que el usuario tiene que responder correctamente para ganar puntuación extra.

Estas preguntas se realizan después de cada pantalla y durante la ejecución del juego “Explorar” al acercarse a un objeto con forma de perla.



Las preguntas son de tipo test y tienen 4 posibles respuestas. El jugador tiene 20 segundos para responder y si lo hace correctamente ganará 10 puntos si es una pregunta hecha después de u n juego, o 50 puntos si es una pregunta del juego “Explora”.

3.1.5 Libros

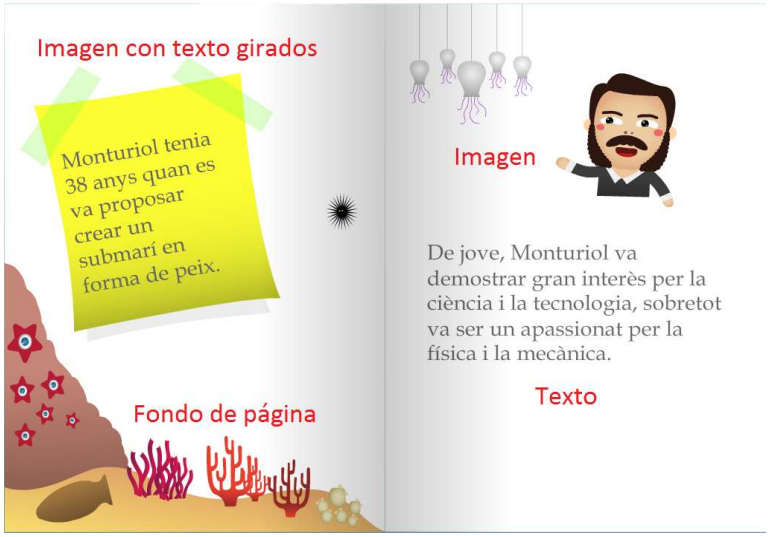
El juego presenta contenidos históricos en forma de libros virtuales, estos libros constan de las siguientes partes:

- Portada, que a su vez contiene:
  - Fondo.
  - Título.
  - Título principal.
  - Número de volumen.
  - Imagen de pie de página.

- Contraportada, que contiene:
  - Fondo.
  - Imagen de pie de página.

- Páginas, que contienen:
  - Imágenes.
  - Textos.
  - Animaciones.

Los elementos de la portada y contraportada están siempre en la misma posición. Los elementos de las páginas se pueden cambiar de posición y rotación.



3.1.6 Enemigos

En este apartado se explican los diferentes enemigos que aparecen en el juego y los diferentes comportamientos y características que pueden tener.

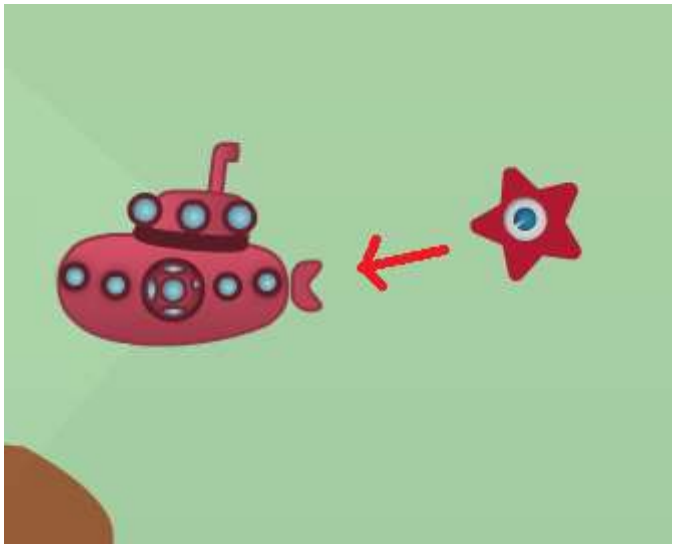
3.1.6.1 Estrella de Mar

En cuanto el submarino entra en su radio de acción empieza a perseguirlo. Hay pantallas en que el radio de acción de la estrella es mayor y no hace falta que el submarino se acerque para que lo persiga.

Este enemigo tiene dos efectos diferentes contra el submarino dependiendo del mini juego:

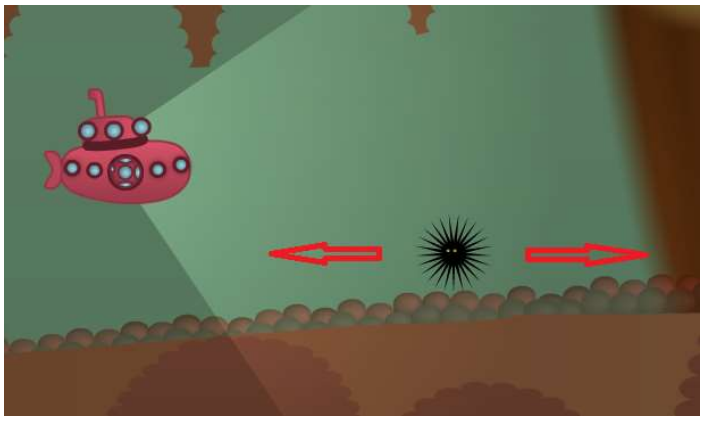
- Al chocar con el submarino resta vida al jugador y ejerce una fuerza que afecta al control del submarino.

- Al chocar con el submarino cuando tiene el rayo tractor activado, hace que pierda la carga. Con este comportamiento no quita vida al jugador y no afecta al control del submarino.



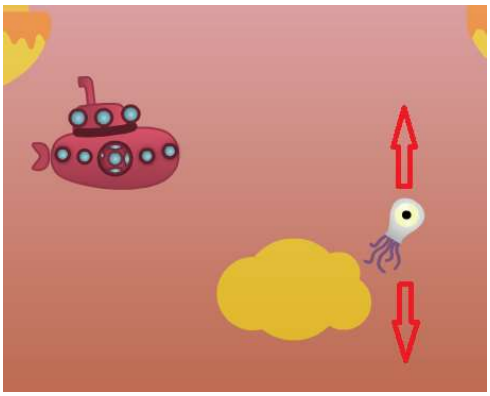
3.1.6.2 Erizo de mar

Tiene un movimiento lineal fijo horizontal entre dos puntos. Cuando choca con el submarino le hace perder vida y lo empuja afectando el control del submarino.



3.1.6.3 Medusa

Tiene un movimiento lineal fijo vertical entre dos puntos. Cuando choca con el submarino le hace perder vida y desaparece del mapa, no afecta el control del submarino.

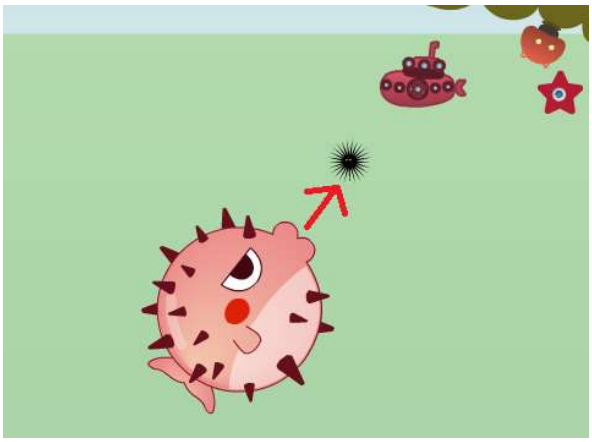


3.1.6.4 Pez globo

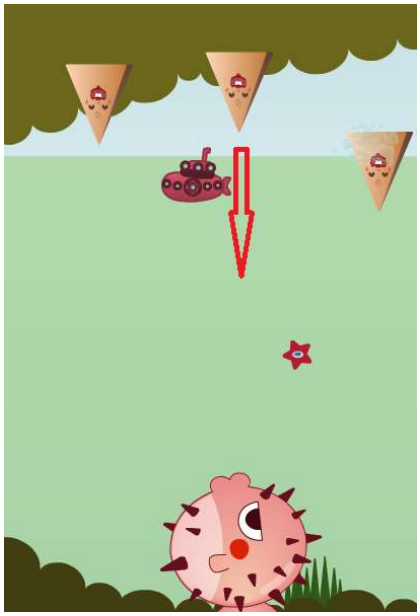
Este es algo parecido a un enemigo final de pantalla, se trata de un pez globo muy grande que persigue lentamente el submarino. Si el pez globo choca contra el submarino le provoca pérdida de energía.

Hay dos formas de eliminar este enemigo:

- Haciendo que coma otros enemigos hasta que llegue al máximo que puede comer.



- Haciendo que objetos del mapa caigan sobre él.



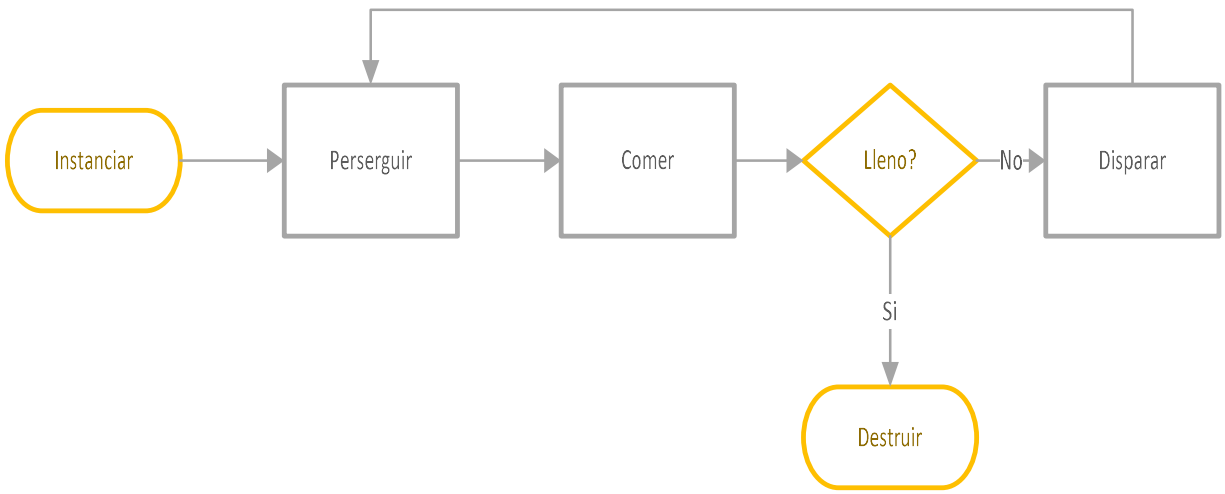
El pez globo tiene varios comportamientos distintos:

- Perseguir el submarino por toda la pantalla.

- Comer un enemigo erizo con lo que aumenta de tamaño y acumula un enemigo comido.

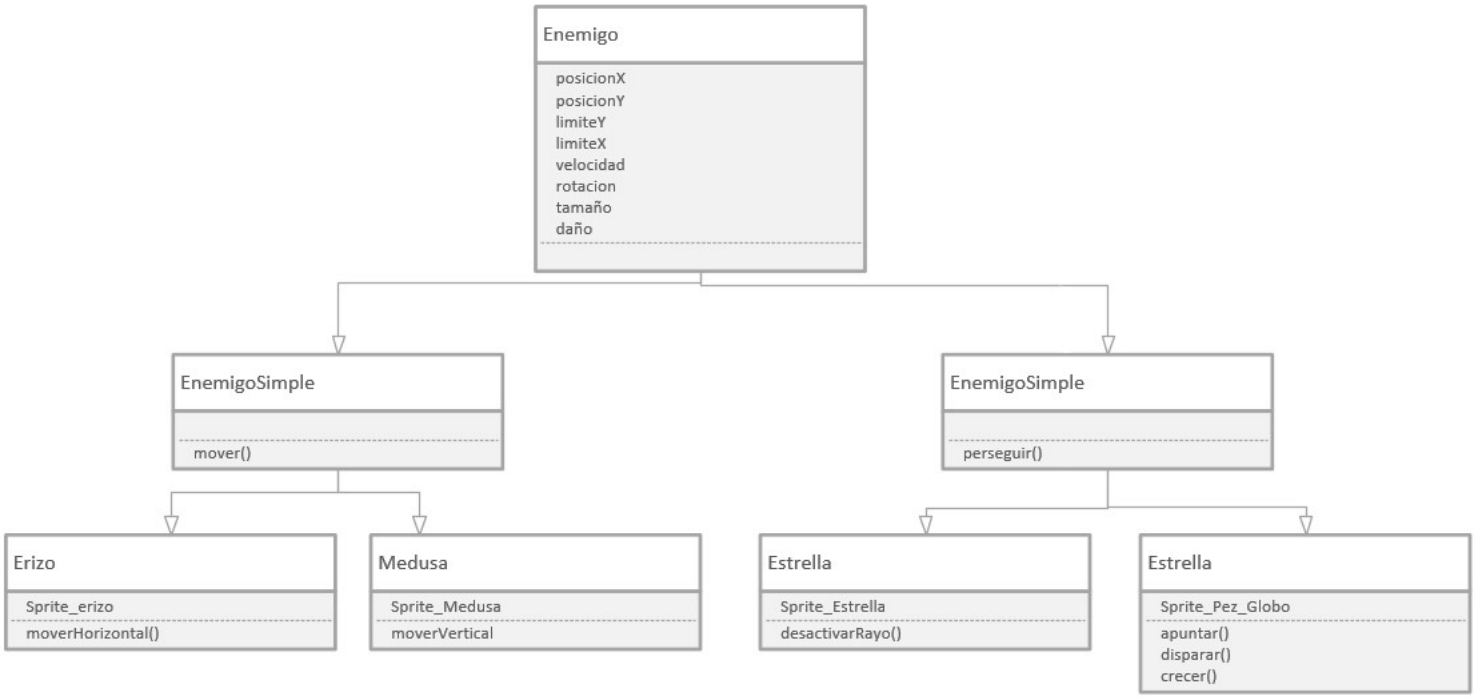
- Comer una estrella de mar, al principio engorda y se queda quieto, a partir de ahí apunta al submarino allá donde vaya para escupirle la estrella de mar a modo de cañón. Una vez escupida la estrella vuelve a su estado normal y retoma la persecución del submarino. La estrella de mar es el último enemigo que se tienen que comer el pez globo.

Diagrama de estados



3.1.6.5 Diagramas de enemigos

Diagrama básico para los enemigos del juego:



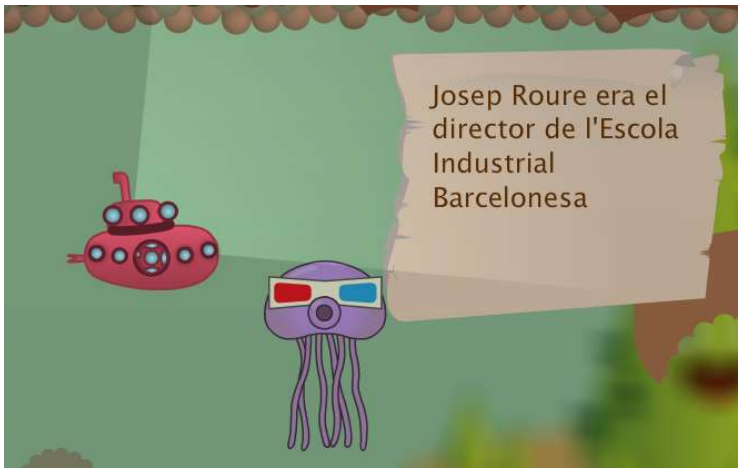
3.1.7 Objetos los mapas

Características de los objetos que se encuentran en el mapa y con los que el jugador puede interactuar.

3.1.7.1 Pulpo

El pulpo es un personaje que aparece en los mapas y al acercarse a él muestra un cartel con pistas sobre como conseguir el objetivo del mapa o sobre el contenido histórico del juego que luego ayudará a contestar bien las preguntas.

Cada personaje de pulpo tendrá asignado un contenido en el mapa.



3.1.7.2 Corrientes

Son objetos posicionados en el mapa que cada cierto tiempo se activan y generan una corriente de agua que empuja al submarino.

3.1.7.3 Perla

Es un objeto estático del mapa, con dos estados: activado y desactivado. Cuando el submarino se acerca lanzará una pregunta que tiene que contestar el jugador, si la acierta ganará puntos, sino nada. La perla después de hacer la pregunta se desactiva y no se puede volver a utilizar durante la partida

3.1.7.4 Ítems o “power ups”

Estos ítems aparecen durante el juego de “Passeig”, el “infinite run”, sirven para conseguir más tiempo, más velocidad o más vida, de este modo el jugador puede estar más tiempo jugando la pantalla y conseguir más puntos.

3.1.7.5 Monedas

Son objetos en forma de monera que al cogerlas dan diez puntos al jugador. Tiene una posición fija en la pantalla y una animación de como si estuviesen girando.

Capítulo 4 – Resumen del análisis

A partir del análisis se reconocen todos los componentes del videojuego que hay que resolver en Unity. A continuación, se muestra una tabla con todos ellos y la manera que se implementará:

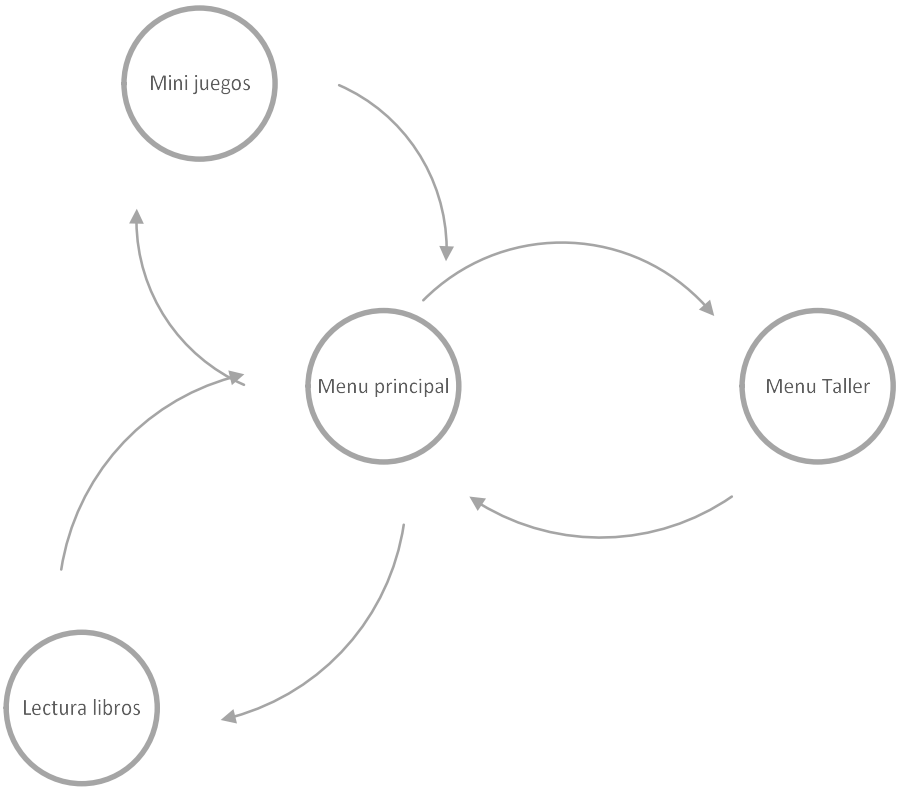
Componente	**Implementación en Unity**
Menú principal	Se utilizará el sistema de creación de interfaces de usuario. Ofrece componentes como botones, imágenes, barras de desplazamiento, etc.
imágenes personajes, enemigos e ítems	Se utilizará el componente **Sprite Renderer** de Unity, que permite el manejo de gráficos y sus propiedades.
Colisiones	Unity será el encargado de la detección de colisiones, para ello tiene unos objetos especializados que se llaman **Colliders.**
Input del usuario	Unity ofrece la clase **Input** desde la que se pueden obtener los inputs del teclado y del ratón sin tener que programar nada.
Lógica del juego y comportamiento de objetos	La lógica del juego se programa en **Scripts,** estos scripts se añaden como un componente más a los objetos del juego. El script le dará el comportamiento necesario a ese componente. Otro tipo de scripts son los controladores que no están asociados a un tipo de objeto, estos se encargarán de la gestión del juego: empezar, acabar, cargar, etc.
Contenido persistente del juego	Todos los datos necesarios para el juego se guardarán en archivos XML. Para ello se utiliza la capacidad de serializar objetos de **C#.**
Mensajes en el juego	Para los mensajes que van apareciendo durante el juego se utilizará el **3D Text** que renderiza textos en el espacio.
Cronómetro	Muchos mini juegos hacen uso de un crono, se implementará un **prefab** de Unity encargado de contabilizar el tiempo transcurrido.

Componente	Implementación en Unity
Puntuaciones y datos volátiles del juego	Se implementará un controlador global para llevar el recuento de puntos, el avance del jugador y las características que se le van añadiendo al submarino
Submarino	Será un prefab con los siguientes componentes de Unity: · Sprite Renderer, para la imagen · Collider, para la detección de colisiones. · Spot light para simular el foco. · Script, para el comportamiento y la gestión de los inputs de usuario.
Libros	Se creará una clase serializable que es una abstracción de los libros del juego. La clase será capaz de cargarse y guardarse en un archivo XML.
Renderizado de libros	Para hacer la parte de lectura de libros se creará una escena que utiliza el sistema de UI de Unity. La escena cargará el libro desde un archivo XML y lo presentará en pantalla en forma de paneles. Para simular el efecto de pasar páginas se utilizará la capacidad 3D de Unity.
Preguntas	Al igual que los libros las preguntas se guardarán en archivos XML. También se implementará una clase que representará a las preguntas.
Renderizado de preguntas	También se utiliza el sistema de UI de Unity, las preguntas se presentarán en pantalla completa por encima del juego.
Mapas	Para generar los mapas se ha pensado utilizar el motor gráfico 3D de Unity para darle un aspecto más atractivo. Utilizar un algoritmo de generación aleatoria, apoyado por un editor manual, que simplificará la creación de los mapas. Evitar que ocupen mucho espacio en disco, para ello se guardarán en formato XML con la información necesaria para volverlos al cargar.
Enemigos	Como el submarino, serán prefabs con su comportamiento programado y los componentes necesarios para detectar colisiones, tener su imagen y moverse.

Componente	Implementación en Unity
Objetos del mapa	Al igual que los enemigos serán **prefabs** que se pueden instanciar en los niveles.
Sonido	Se utilizarán los componentes que ofrece Unity para la reproducción de sonidos y músicas, **Audio Clip.**
Mini juegos	Los mini juegos se implementarán como una **escena** de Unity y constarán de los siguientes elementos: - Mapa - Ítems - Enemigos - Objetos - Submarino - Lógica de control
Pantallas mini juegos	Para generar las pantallas solamente es necesario replicar la escena del mini juego correspondiente, cambiar la forma del mapa y quitar/ añadir enemigos e ítems al gusto, la lógica del juego se encargará luego de moverlo todo como toca. Todo esto se hace desde el mismo editor de Unity mediante drag&drop de una manera muy sencilla. Después sólo quedará asocias esa nueva escena a un botón de la interfaz del menú principal para poder ser jugada.

Capítulo 5 – Implementación en Unity

Todas las pantallas del juego han sido implementadas como escenas de Unity: los diferentes menús, los distintos niveles de los mini juegos y la lectura de los libros.



5.1 Controlador de Juego

Este objeto es el que controlará la parte de información persistente del juego:

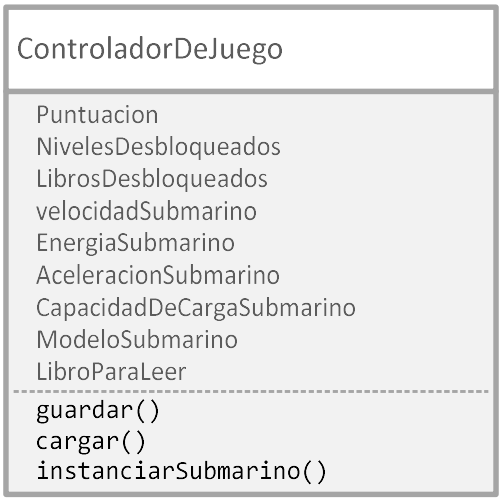
- Puntuación

- mejoras del submarino.
- Niveles desbloqueados.
- Libros desbloqueados.
- Tiempos de las cursas.

Está implementado siguiendo el patrón de diseño **Singleton**, ya que es necesario que solo exista una instancia de él. Además, este objeto tiene que ser accesible desde cualquier escena, debido a que registra la información del jugador, y esta puede ser modificada desde cualquier escena o juego.

Esta misma clase sirve para guardar el avance del juego una vez se sale de él. Para ello tiene funciones implementadas para guardar su estado en un archivo XML que luego se puede volver a cargar.

Por último, también tiene la responsabilidad de instanciar el submarino con las mejoras que tiene compradas en ese momento.



5.1 Menú principal

En este apartado se explican los detalles de implementación del menú principal. Se utiliza el sistema de creación de interfaces de usuario, que provee al programador una serie de elementos básicos muy personalizables.

El menú principal es una escena de Unity, las escenas tienen sus propios componentes, cámara e iluminación. En general cada vez que se acceda a una parte del menú diferente, es decir que haya un cambio en la pantalla, se estará accediendo a otra escena diferente. Las escenas que se ha realizado se pueden ver en el diagrama de transiciones del apartado 3.1.1.

5.1.1 Controlador de menú principal

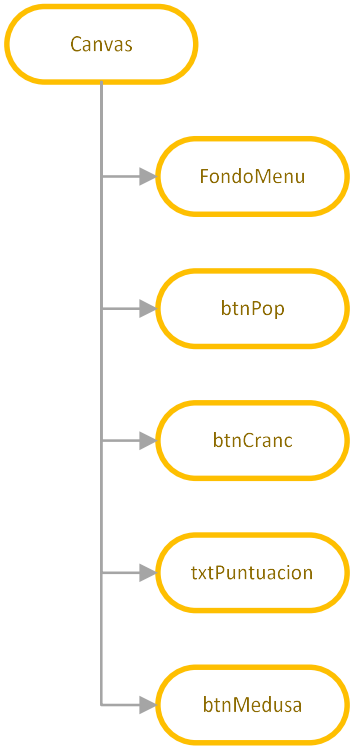
Este componente gestiona toda la lógica de los botones del menú principal.

5.1.2 Controlador de Panel

Este controlador gestiona la lógica de los botones que tiene un panel, generalmente todos los botones de un panel que son clicados ponen en marcha un mini juego, por lo que el controlador se encargará de cargar la escena que toque.

5.1.3 Fondo del menú

Para el fondo del menú se utiliza un objeto Canvas, todos los objetos de la interface tienen que depender de un objeto de este tipo, es una especie de contenedor. La Imagen de fondo de menú también está incluida en este objeto. (En el gráfico no se muestran todos los componentes)



5.1.4 Botones

Hay varios tipos de botones en el juego original: **animados y simples**. Los botones animados, como dice el nombre, están compuestos por animaciones y no ocupan una posición fija en la escena, tienen un pequeño movimiento en vertical y/u horizontal. Los botones simples tienen una posición fija y una imagen estática.

El tema de las animaciones se ha obviado por que no se disponen de los gráficos necesarios, sólo tenemos imágenes estáticas. Sobre el movimiento, se ha programado un script para que cada botón tenga un comportamiento personalizable. Este script permite que el botón se mueva horizontal y verticalmente en unos rangos especificados por parámetro, lo mismo pasa con el tamaño.

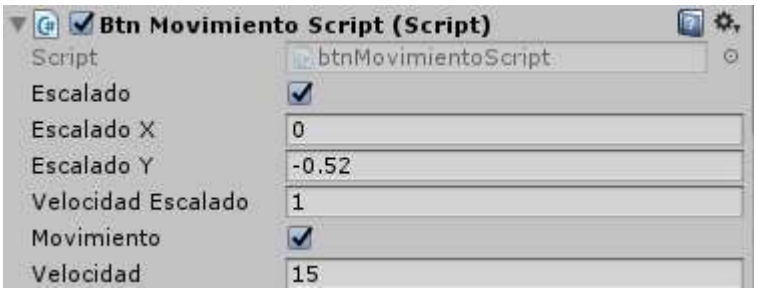
A su vez los botones pueden clasificarse en los que dependen del Canvas principal y los que están asignados a un panel.

Los botones tienen una acción asignada. Los que dependen directamente del canvas principal llaman a funciones del controlador del menú donde están ubicados: controlador menú principal, controlador menú de mejoras del submarino, etc. Los botones que se encuentran dentro de paneles llaman funciones que están dentro del controlador a nivel de panel.

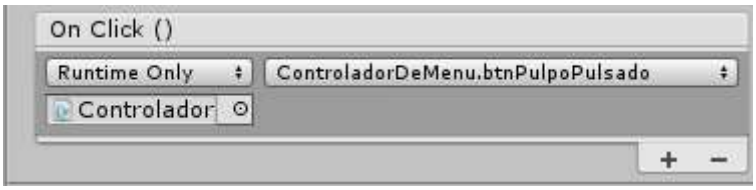
Este botón se ha guardado como prefab, de esta manera siempre que se necesite un botón sólo hay que crear una instancia de él y personalizar la imagen que tendrá. Este prefab se utiliza para los dos tipos de botones, ya que los botones simples se pueden tomar como un tipo de botón animado sin movimiento.

Componentes del botón

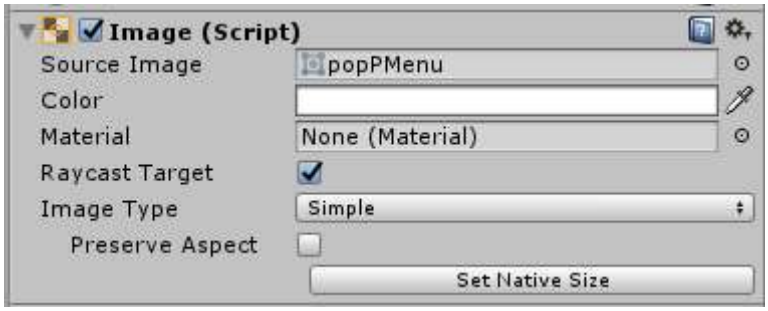
Script con sus parámetros para la animación:



Botón asociado al controlador de menú:



Componente imagen asociada al botón:

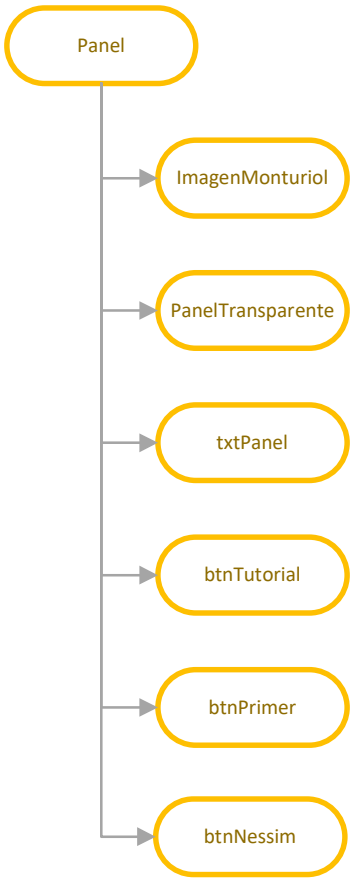


5.1.5 Paneles

En los menús otro elemento importante son los paneles, contenedores que agrupan botones e imágenes. Los paneles tienen un fondo asignado y un script que controla las acciones de los botones que contiene.

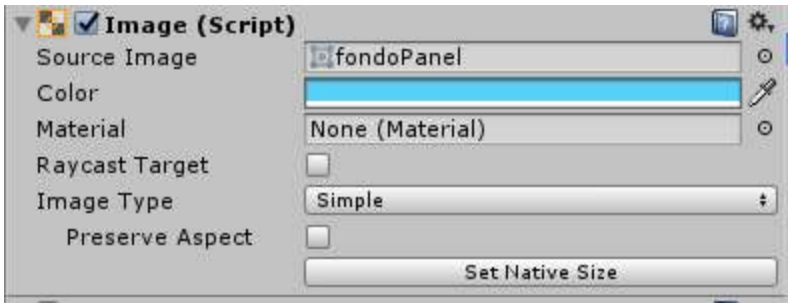
Componentes del panel

El panel tiene asignados una serie de componentes de forma jerárquica, botones, imágenes y textos:



Cada panel tiene asignado un script que controla la lógica de los componentes que contiene capturando los eventos que estos generan:

Un panel se puede personalizar con un fondo de pantalla.



5.2 Menú Taller

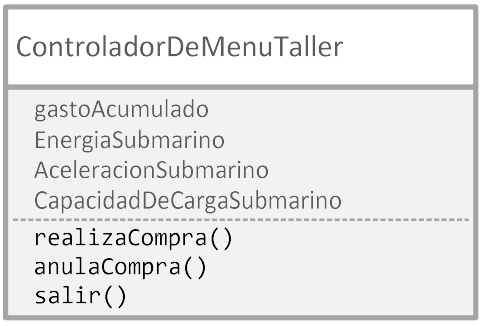
Este menú se trata de otra escena, se carga con la siguiente información que contiene el controlador del juego:

- La puntuación que hace de saldo para comprar cosas.

- Las mejoras que tiene el submarino hasta el momento. Las mejoras que ya tiene activas el submarino no se podrán comprar otra vez. Las mejoras que no tiene se podrán comprar si el jugador tiene suficiente saldo

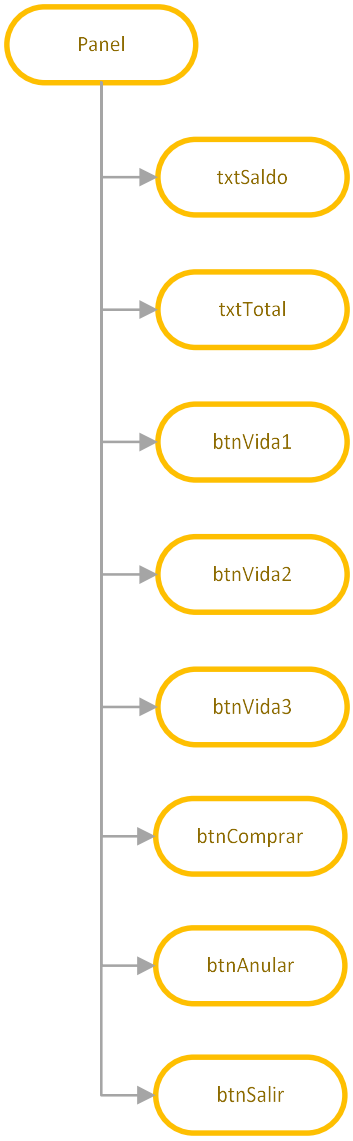
Controlador

Esta lógica la lleva el controlador del menú taller:



Panel

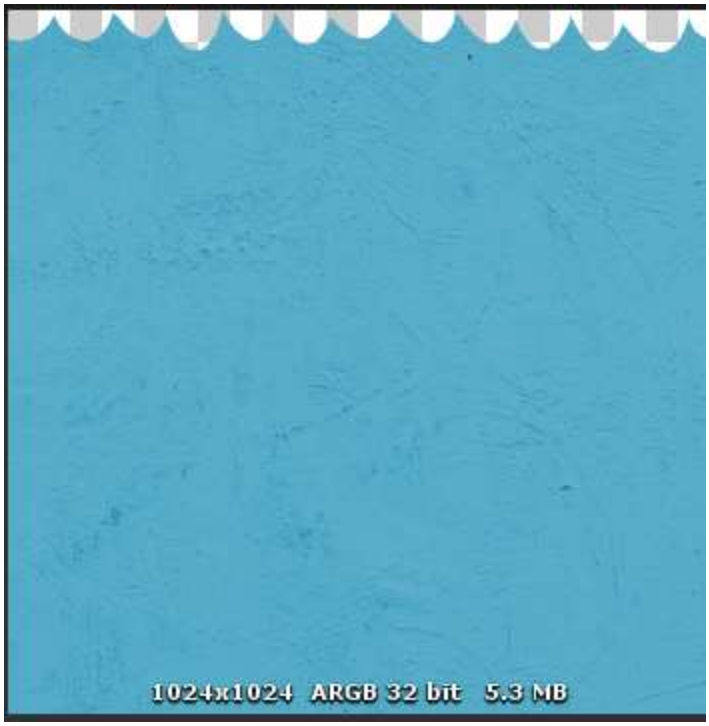
El menú taller se compone de un canvas como el del menú principal, pero con un único panel que contiene los botones.



Scroll

También contiene un scroll que simula el agua del mar moviéndose. Para implementar el scroll se utiliza un objeto Quad de Unity y se le aplica una textura que hará de mar, luego mediante un script se irá moviendo el offset de textura, de esta forma se simula el movimiento.

Esta es la textura que se usa para simular el mar, se trata de un fondo azul con transparencias arriba que simulan las olas.



Y pequeño script que se añade al Quad encargado de mover la textura. Es posible modificar la velocidad del scroll con el parámetro velocidad.

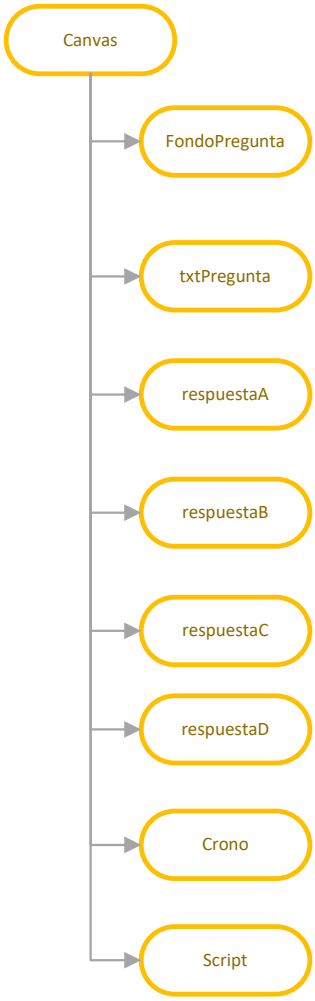
```
public class scroll : MonoBehaviour
{
    public float velocidad = .2f;

       // Use this for initialization
       void Start ()
    {

       }

       // Update is called once per frame
       void Update ()
    {
        Vector2 offset = new Vector2(Time.time * velocidad, 0);

        GetComponent<Renderer>().material.mainTextureOffset = offset;


       }
}
```

5.3 Menú pregunta

Aunque se le llame menú en realidad es un canvas que contiene la información para pintar la pregunta en pantalla. Este canvas se hará visible a pantalla completa cuando haya que realizar la pregunta.

El script de control asociado es el encargado de pintar los textos y los botones, también gestionará si la respuesta elegida por el usuario es correcta o no y en función de eso actuará sobre la puntuación acumulada.

Componentes del canvas

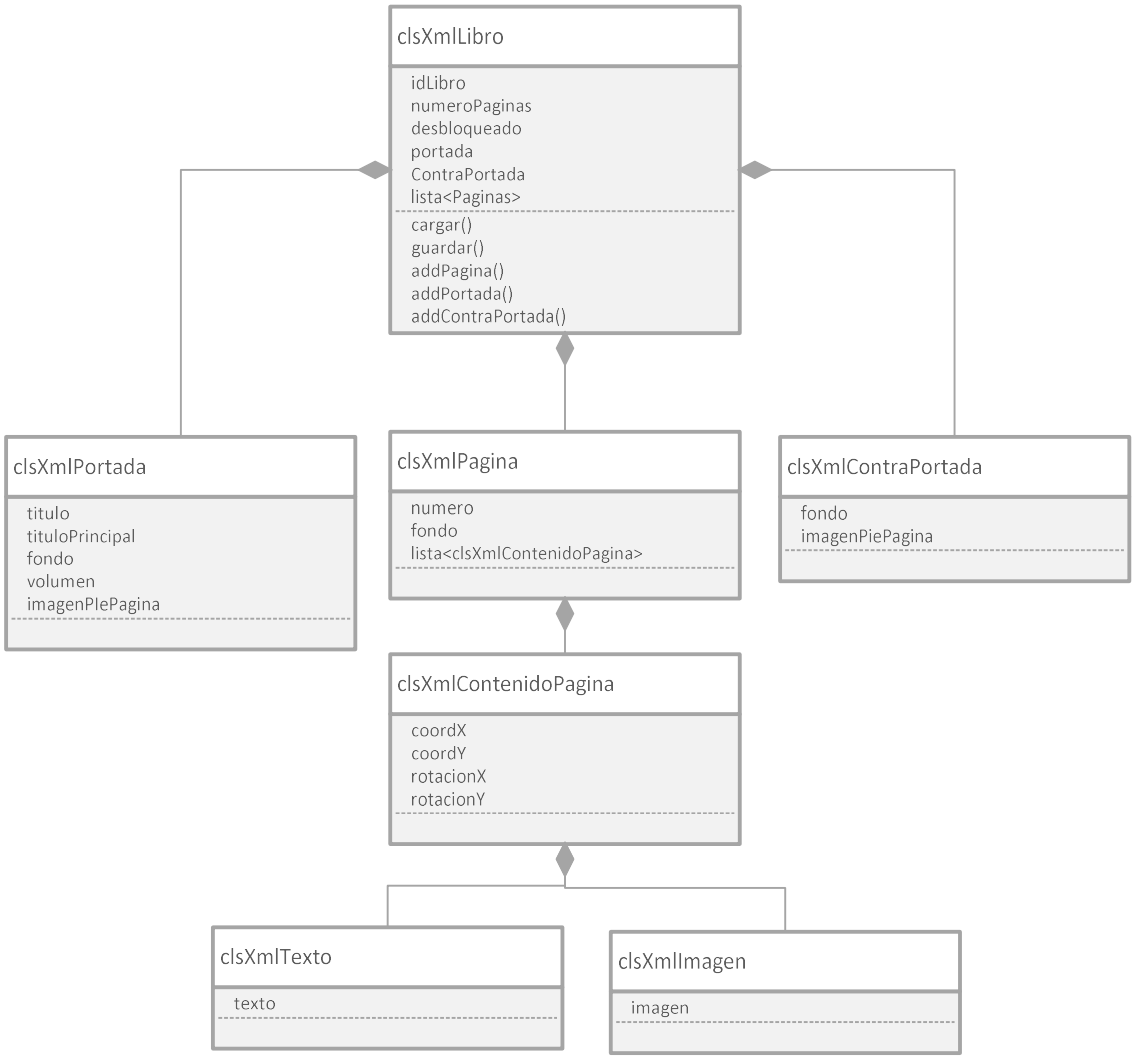


5.4 Menú libros

Este menú sirve para consultar el contenido educativo que tiene el juego sobre el personaje Monturiol en forma de libros virtuales.

5.4.1 Libros

Antes de seguir con el menú para leer los libros, hay que explicar cómo se guardarán y manejarán los contenidos de ellos. Del análisis del juego se ha hecho la siguiente abstracción:



La clase **clsXmlLibro** representa todo el contenido que puede tener un libro, está compuesta de una portada (**clsXmlPortada)**, una contra portada (**clsXmlContraPortada)** y N páginas (**clsXmlPagina)**.

Las paginas están compuestas por contenido de texto y/o imágenes, este contenido está implementado en las clases **clsXmlTexto**, **clsXmlImagen** que derivan de **clsXmlContenidoPagina**. Estas clases guardan la posición y rotación dentro de la página a la que pertenecen.

Para hacer persistente el contenido de los libros se hace a través del formato XML, la clase **clsXmllibro** implementa una función que serializa la clase en un archivo XML, de esta forma es muy fácil modificar el contenido del libro. La misma clase tiene una función para cargar un libro desde un archivo XML.

Ejemplo de archivo XML que representa un libro:

```
<?xml version="1.0" encoding="Windows-1252"?>
<libro xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" numeroPaginas="2" desbloqueado="false">
  <paginas>
    <pagina numero="1">
      <contenidos>
        <contenido xsi:type="contenidoTexto">
          <coordX>10</coordX>
          <coordY>10</coordY>
          <rotacionX>0</rotacionX>
          <rotacionY>0</rotacionY>
          <texto>hola1</texto>
        </contenido>
        <contenido xsi:type="contenidoTexto">
          <coordX>10</coordX>
          <coordY>10</coordY>
          <rotacionX>0</rotacionX>
          <rotacionY>0</rotacionY>
          <texto>hola2</texto>
        </contenido>
      </contenidos>
    </pagina>
    <pagina numero="2">
      <contenidos>
        <contenido xsi:type="contenidoTexto">
          <coordX>10</coordX>
          <coordY>10</coordY>
          <rotacionX>0</rotacionX>
          <rotacionY>0</rotacionY>
          <texto>hola1</texto>
        </contenido>
        <contenido xsi:type="contenidoTexto">
          <coordX>10</coordX>
          <coordY>10</coordY>
          <rotacionX>0</rotacionX>
          <rotacionY>0</rotacionY>
          <texto>hola2</texto>
        </contenido>
      </contenidos>
    </pagina>
  </paginas>
  <portada>
    <fondo>fondoNuves</fondo>
    <titulo>Primer título</titulo>
    <tituloPrincipal>Título principal</tituloPrincipal>
    <volumen>Volumen 1</volumen>
    <imagenPie>pie1</imagenPie>
  </portada>
  <contraportada>
    <fondo>fondoNuves</fondo>
    <imagenPie>pie1</imagenPie>
  </contraportada>
</libro>
```

5.4.2 Escena libro

Una vez sabemos cómo se guarda un libro vamos a ver como se representa en la pantalla. Para ello la escena contiene un canvas en donde están los siguientes botones:

- “Endarrere”, retrocede una página.
- “Endavant”, avanza una página.
- “Sortir”, sale al menú principal.

Las lógicas de esta escena está programa en el **controladorLibro:**

- Control de los botones.
- Control del avance y retroceso gráfico del libro.
- Renderizado del libro.

Renderizado

Para renderizar un libro el **controladorLibro** carga el archivo XML en memoria con la clase **clsXmlLibro**. Entonces va creando Paneles en el canvas de la escena que simulan las hojas. El recorrido lo empieza por la contraportada ya que los últimos elementos tienen que estar tapados por los primeros.

Para crear los paneles el controlador puede instanciar unos prefabs que ya han sido implementados y representan los componentes del libro:

- **portadaPrefab**: es un panel con los textos colocados en unas posiciones fijas que representan los títulos y la imagen de pie de página. Estos elementos se personalizan en el momento de renderizado del libro.

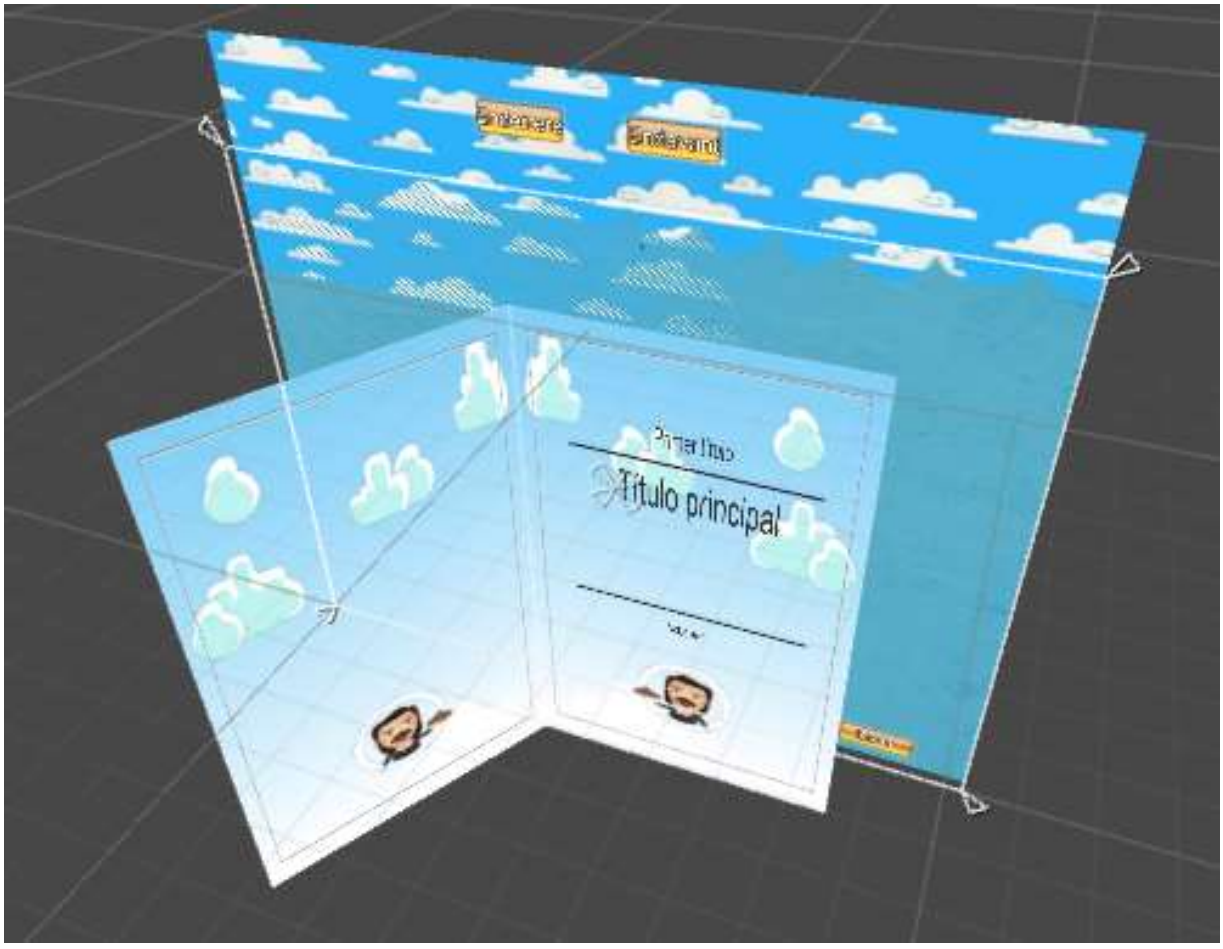
- **contraPortadaPrefab**: es un panel con una imagen en una posición predefinida como pie de página y un fondo.

- **paginaImarPrefab**: es un panel en blanco, en él se crearán de forma dinámica los contenidos definidos en la clase **clsXmlPagina**. Esta página se pinta volteada porque se supone que está boca abajo hasta que se pase de página, tiene el punto de pivote para rotar en su derecha.

- **paginaParPrefab**: igual que la **paginaImparPrefab**, diferenciando que se pinta de forma normal y el punto de pivote para rotar está definido en su izquierda.

Para simular el paso de las páginas se aprovecha la capacidad 3D de Unity, cada vez que el usuario aprieta el botón “Endavant” y “Endarrera”, el controlador, que sabe la página actual, rota una página par y una impar para simular el paso de hoja.

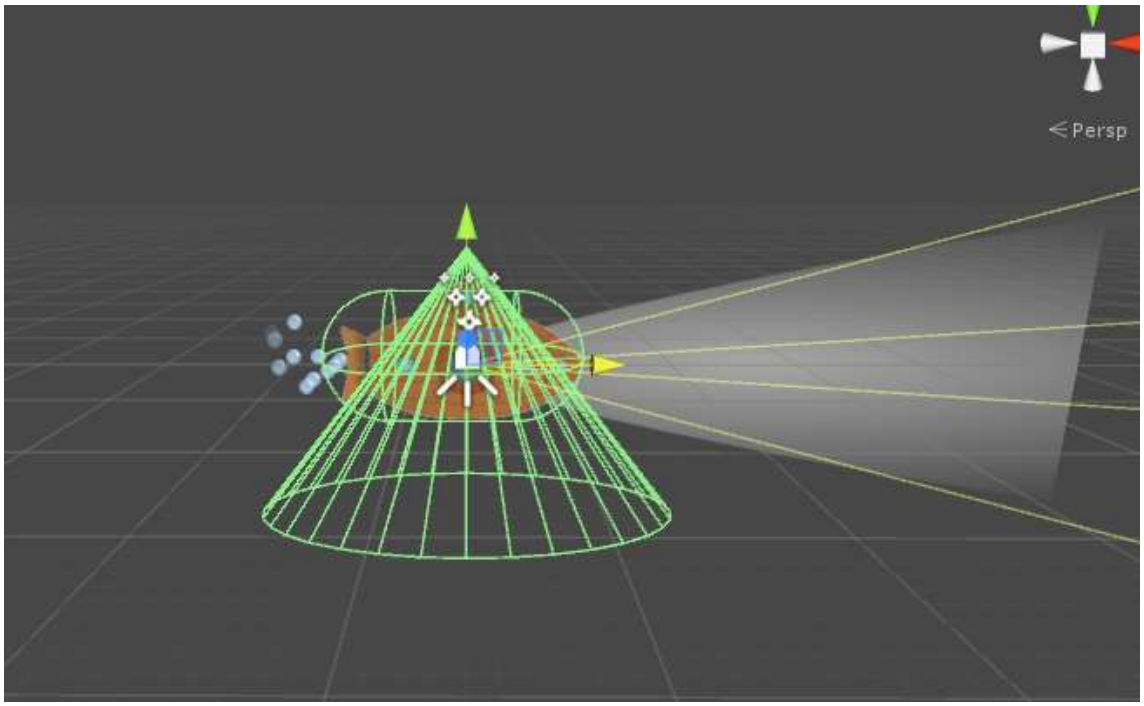
En la siguiente imagen se puede observar el funcionamiento:



En la siguiente se ve el efecto en 2D:

5.5 Submarino

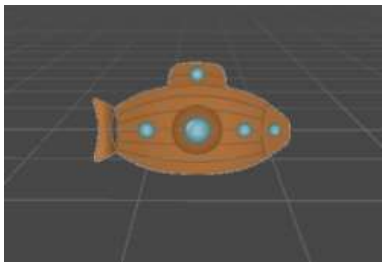
Para implementar el submarino se ha creado un prefab usando componentes 3D que ofrece Unity: colliders, meshes y luces. También componentes 2D: Sprite Renderer.



5.5.1 Componentes del submarino

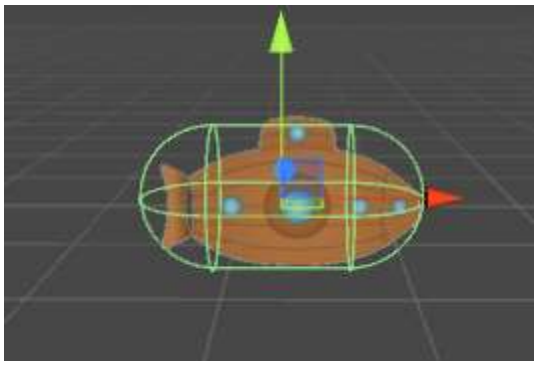
Sprite

Es lo más sencillo, una imagen que representa al submarino, esta imagen se puede personalizar por las cuatro que hay en el juego.



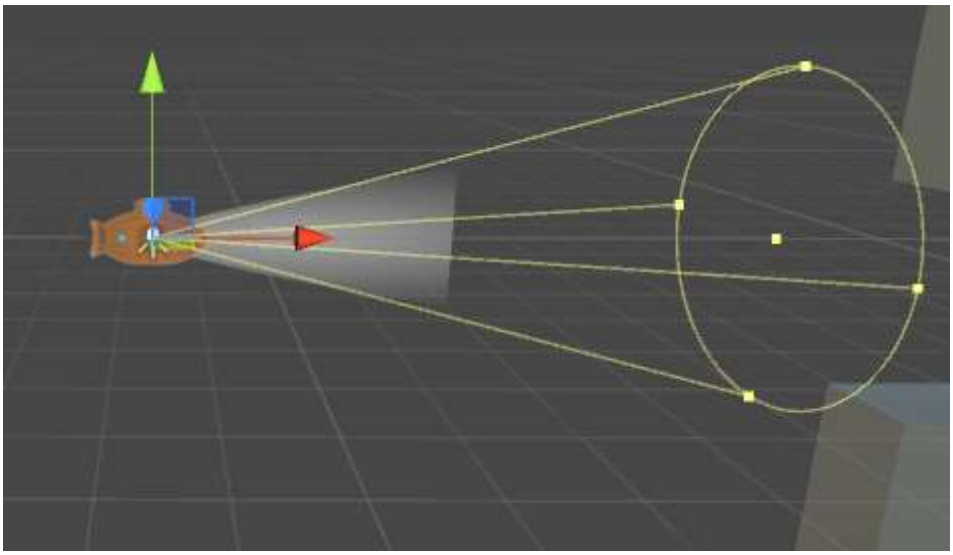
Capsule Collider

Este componente se encarga de detectar las colisiones con otros objetos, como por ejemplo, el mapa, enemigos e ítems.



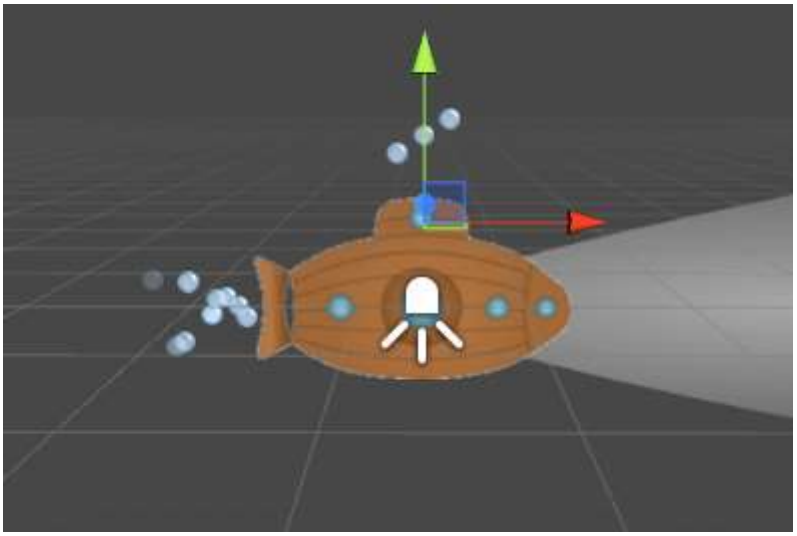
Luz

Aprovechando las capacidades de Unity se ha decidido darle un toque más espectacular a la luz que tiene el submarino, para ello se ha añadido un componente **spot light**, es una luz direccional que iluminará la escena en tiempo real. Como el cono de luz es invisible se ha añadido un Sprite plano que hace de cono del foco.



Burbujas de oxígeno y hélice

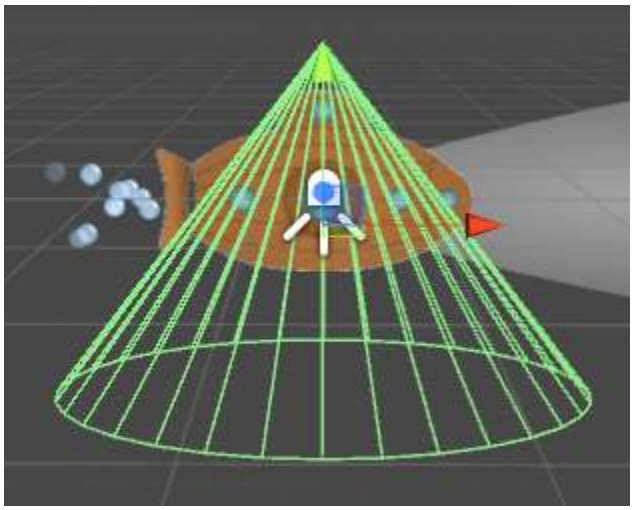
Para simular la pérdida de oxígeno del submarino se ha añadido un sistema de partículas que expulsa burbujas por arriba cada cierto tiempo. También se ha añadido un sistema de partículas en la hélice del submarino que se activa cuando se mueve, de esta manera se obtiene algo más de realismo en el movimiento.



Rayo tractor

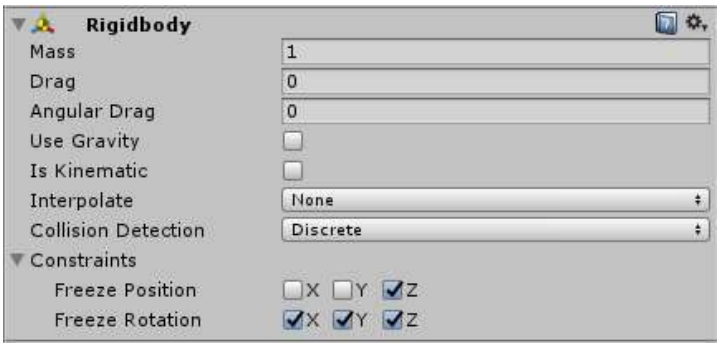
Para el rayo tractor se ha generado un mesh 3D en forma de cono. Los objetos que colisionen con el cono y tengan añadida el script de "Atracción" se verán atraídos hacia el submarino. El script utiliza las funciones **OntriggerEnter** y **OntriggerExit** que tienen como parámetro el Collider con el que choca el objeto.

Para que el cono no se vea se ha desactivado el componente **Mesh Renderer.**



RigidBody

Este componente es necesario para mover el submarino, al contener un **RigidBody** se le pueden aplicar fuerzas y de esta manera impulsarlo por el mapa y recibir golpes de los enemigos. Al ser un juego en 2D tiene la componente Z desactivada. Por defecto la gravedad no le afecta, esto es útil para simular la flotabilidad debajo del agua.



Script: controlJugador

Se ha implementado un script que maneja el submarino a base de aplicarle fuerzas en función de la dirección que el usuario genera con las flechas del teclado o la posición del puntero del ratón. El script también controla la rotación del foco de luz que siempre se pondrá alumbrando en la dirección de movimiento.

```
private void moverJugador( Vector3 direccion )
	{

		if (direccion != Vector3.zero)
		{

			direccion = direccion.normalized * m_aceleracion;

			m_rbJugador.AddForce (direccion);

			if (m_rbJugador.velocity.magnitude > m_velocidadMaxima)
			{
				m_rbJugador.velocity = m_rbJugador.velocity.normalized
* m_velocidadMaxima;
			}


		}

		if (m_rbJugador.velocity.magnitude > 0.01f)
		{
			m_rbJugador.drag = 0.5f;
		}
		else
		{
			m_rbJugador.drag = 0;
			m_rbJugador.velocity = Vector3.zero;
		}
	}
```

En función de la velocidad del submarino ajustará la cantidad de burbujas que salen de la hélice y si el submarino está parado las detendrá.

También está encargado de detectar las colisiones y aplicar una fuerza de choque en caso de que se produzcan.

```
m_rbJugador.AddForce (-transform.forward.normalized * m_fuerzaColision,
ForceMode.VelocityChange);
```

La función **Update** del script se encarga del comportamiento del submarino, se llama cada frame y realiza las siguientes funciones:

- Obtener el input del usuario
- Mover el foco.
- Dibujar el Sprite del submarino en función de la dirección.
- Ajustar las burbujas de la hélice.
- Ajustar la cantidad de oxígeno.

La función **FixedUpdate** se encarga de mover el submarino, se utiliza esta función porque se usa el sistema de física de Unity y esta función asegura que se llama independientemente del frame rate del juego, de esta manera se evita que el submarino se mueva a trompicones en ordenadores con menos potencia.

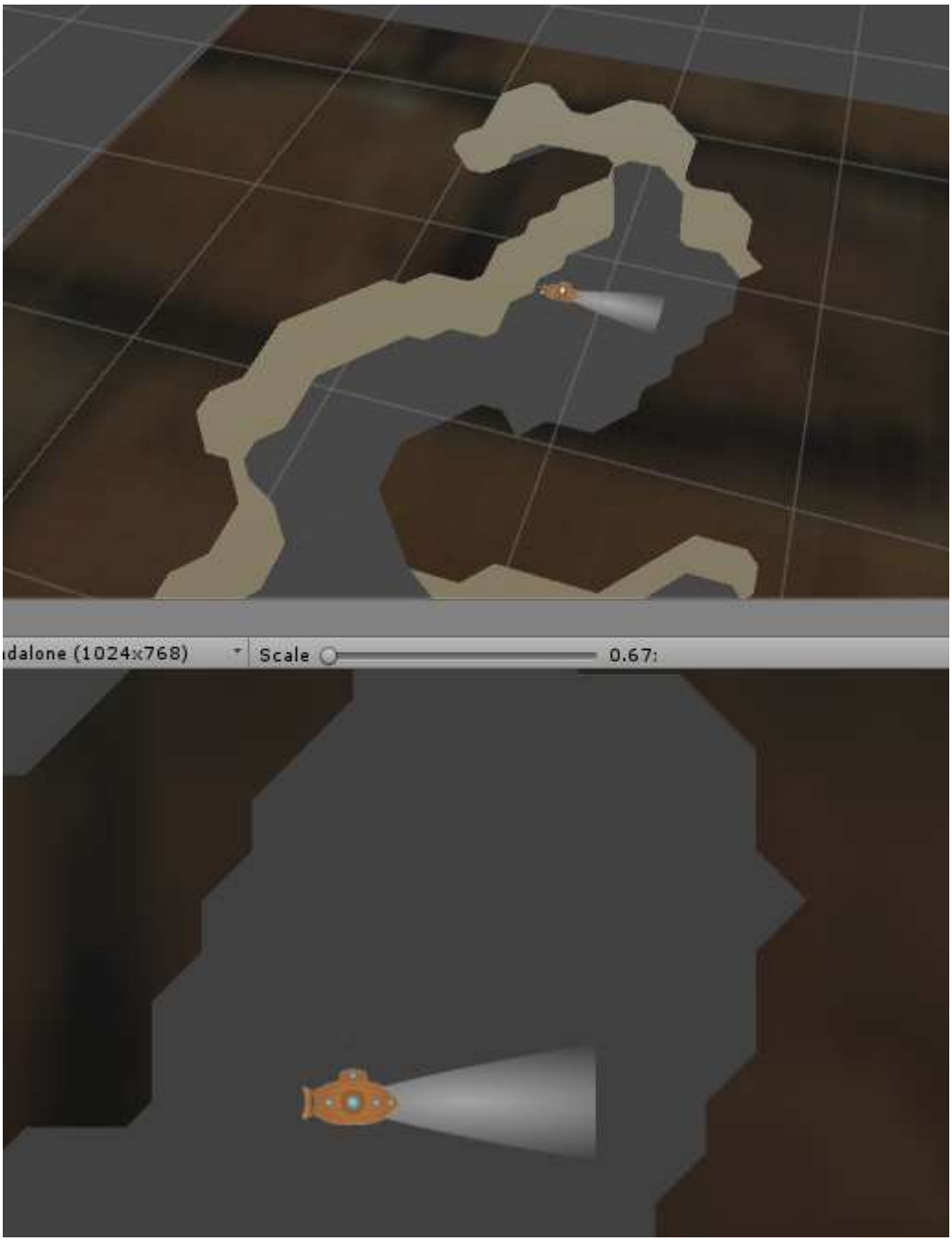
Para gestionar las colisiones el script se basa en los eventos que generan los componentes Collider cuando chocan, es en la función **OnCollisionEnter(),** cuando se mira

5.6 Mapas del juego

Para tratar de hacer de la generación de mapas una tarea sencilla se ha optado por hacer una función que cree los mapas de forma aleatoria en

base de unos parámetros. Esto sólo sirve para crear una base de mapa, ya que después será el diseñador quien genere la versión final del mapa.

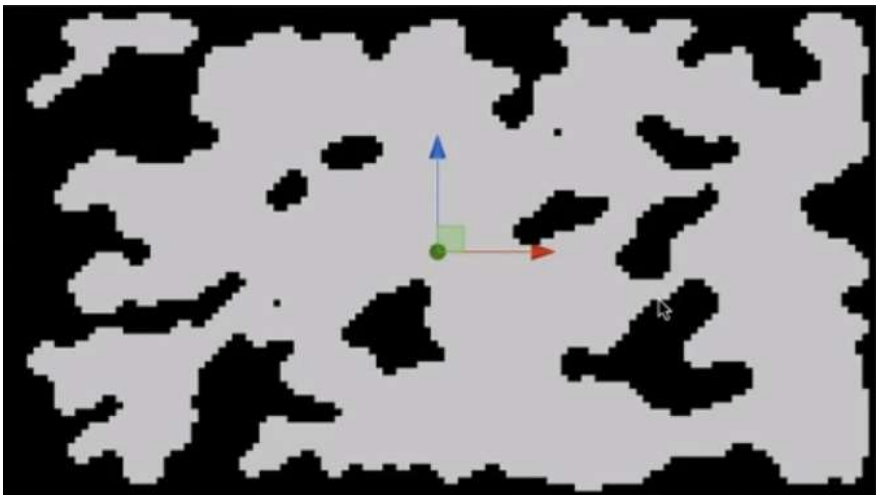
Los mapas se crean en 3D a utilizando la función básica de Unity para la generación de formas en 3D, para ello se le pasa un mapa de puntos que representan los vértices de la forma del mapa.



Mapa 3D y visión en 2D

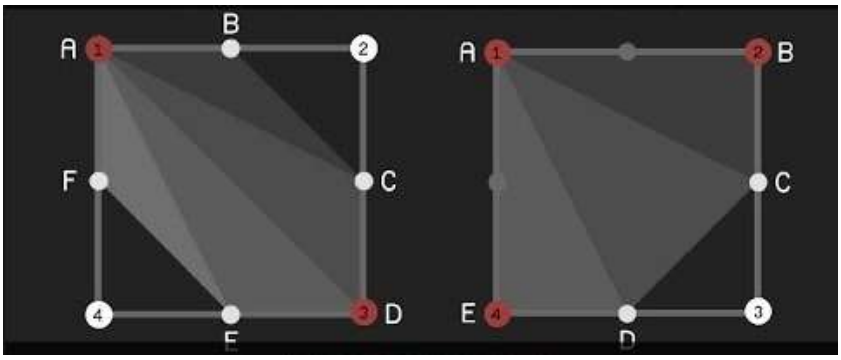
5.6.1 Algoritmo de generación aleatoria

El mapa consiste una matriz que representa los vértices de una malla de cuadrados, los vértices pueden estar activados o no de forma aleatoria. Para la generación del mapa se utiliza una semilla, la altura y anchura del mapa, el porcentaje de llenado del mapa y el tamaño del cuadrado.



Para pintar la forma del mapa se recorren los cuadrados y se miran que vértices tienen activados. Los vértices activados se unirán para formar triángulos que representarán el dibujo del mapa. Entre los vértices principales del cuadrado se crean unos vértices auxiliares que sirven para dibujar más triángulos y la forma del mapa sea menos angulada.

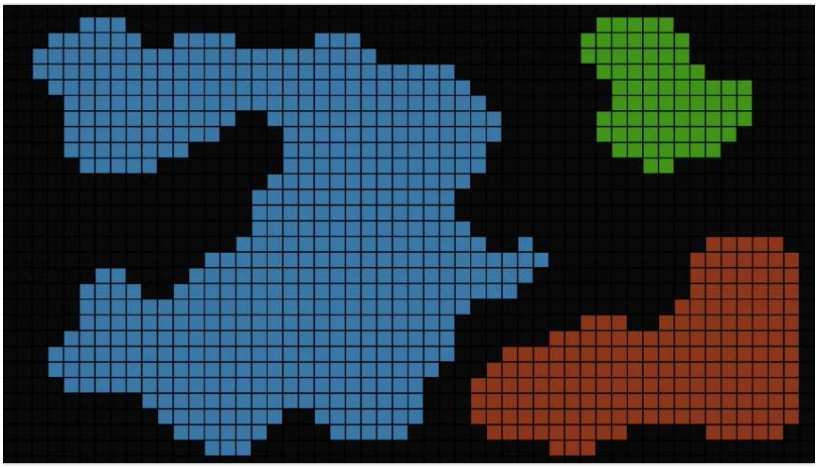
Ejemplos:



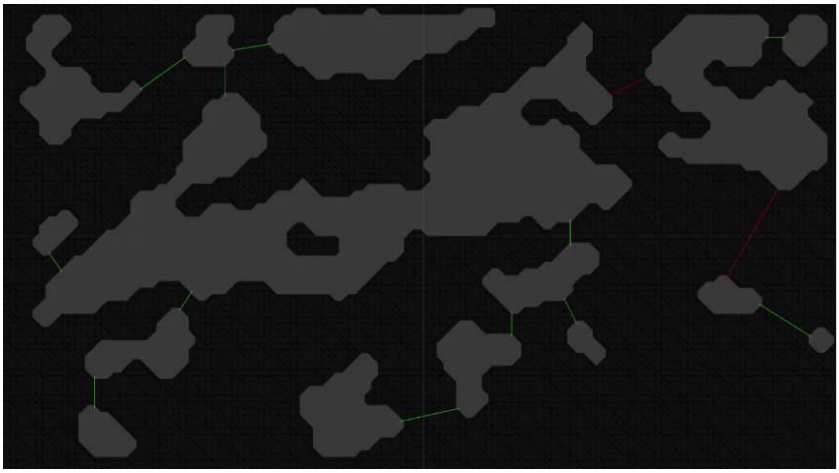
En estos ejemplos están activados los vértices A y D del primer cuadrado y los vértices A, B y E del segundo.

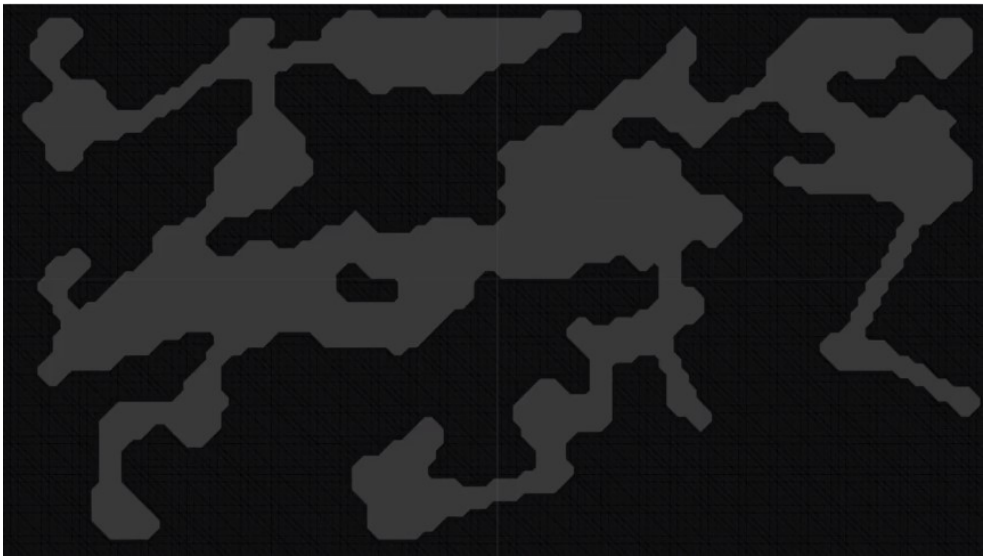
Además de los cuadrados también se pinta una pared hacia el eje Z para dar profundidad al conjunto.

En el siguiente paso se identificarán las regiones vacías que se han creado.



Una vez identificadas se crearán uniones entre ellas para que todas sean accesibles a través de un camino:

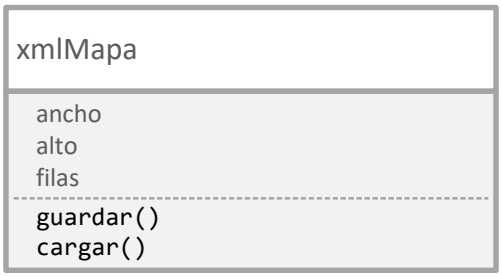




Una vez está el mapa generado se puede guardar tal cual, o se puede modificar con el ratón activando o desactivando vértices de los cuadrados.

5.6.2 Guardar mapas

Para guardar y cargar los mapas se ha implementado la clase **xmlMapa** que contiene el array de vértices del mapa y su estado. A simple vista se puede ver la forma del mapa que contiene, e incluso se puede modificar a mano.



Fichero XML que representa un mapa:

5.6.3 Generación de los niveles

Los niveles de juego se crean a partir del editor de Unity. Primero se tiene que utilizar el generador de mapas para obtener una base del mapa. Después con el editor se tiene modificar para obtener el resultado deseado en función de para que mini juego se está haciendo. El mapa generado hay que guardarlo en un archivo XML.

Habrá que añadir al mapa los prefabs necesarios en función del juego: enemigos, ítems, cronómetro, etc.

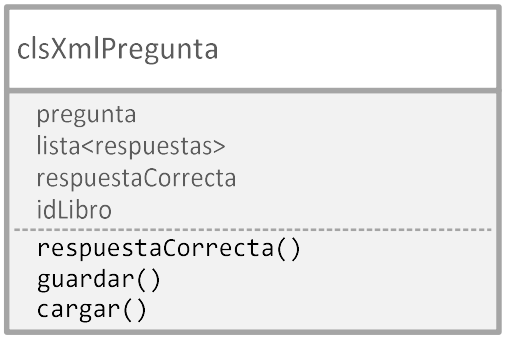
También hay que añadir el controlador de juego para que tenga la lógica necesaria para funcionar, cada mini juego tiene su propio controlador, este debe tener una referencia al mapa del nivel creado para que pueda cargarlo.

Por lo general todas las escenas tienen una cámara, en caso de no haberla hay que añadirla. A la cámara hay que añadirle el script **Control Camara,** que se encarga de seguir el submarino y de evitar que la cámara se salga de la escena.

El script de la cámara trabaja junto un componente **Box Collider 2D** que hay que añadir a la escena. Este Collider contendrá los límites máximos y mínimos en los que se podrá mover la cámara, este límite también servirá para que el submarino no se salga de la escena. Estos límites deben coincidir con la anchura y altura del mapa que se ha generado para el nivel.

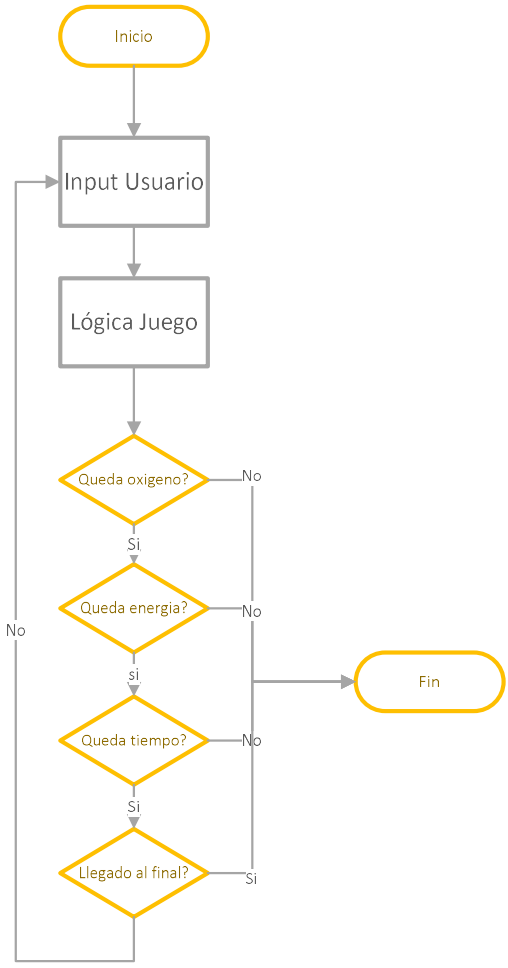
5.7 Preguntas

Para las preguntas se ha implementado una clase parecida a la del libro, pero más sencilla. También puede cargar su contenido desde un archivo XML



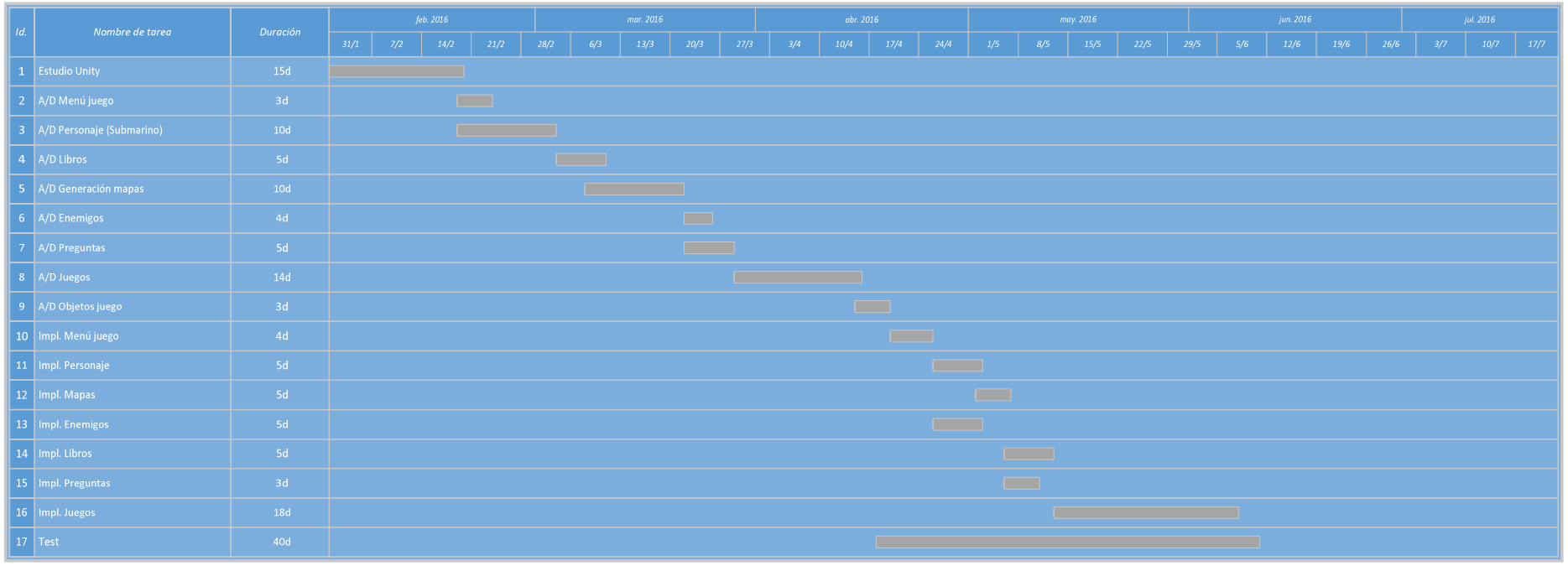
5.8 Juegos

Todos los juegos utilizan una lógica muy parecida programada en su script de control, a continuación, se muestra el esquema base que siguen estos scripts.



Capítulo 6 – Planificación y coste económico

A continuación, se muestra el diagrama con las tareas del proyecto y sus costes en días:



Id.	Nombre de tarea	Duración
1	Estudio Unity	15d
2	A/D Menú juego	3d
3	A/D Personaje (Submarino)	10d
4	A/D Libros	5d
5	A/D Generación mapas	10d
6	A/D Enemigos	4d
7	A/D Preguntas	5d
8	A/D Juegos	14d
9	A/D Objetos juego	3d
10	Impl. Menú juego	4d
11	Impl. Personaje	5d
12	Impl. Mapas	5d
13	Impl. Enemigos	5d
14	Impl. Libros	5d
15	Impl. Preguntas	3d
16	Impl. Juegos	18d
17	Test	40d

El proyecto ha sido realizado en aproximadamente 690 horas, contando aprendizaje, diseño, implementación y test. Todas las horas se toman de un Analista/programador, el precio hora rondará los 35€.

Como los efectos y los gráficos del juego se reaprovechan del juego original no hace falta invertir en diseño gráfico. El coste de obtener los gráficos no se cuenta.

En total tenemos 690h x 35€ = 24150€ en recursos humanos.

El equipo utilizado es un PC con procesador i5-5200U con 8GB de RAM, valorado en aproximadamente 500€.

Al final el proyecto, sumando las cantidades, tiene un coste total de 24650€.

Con la base implementada del proyecto, que tiene sólo una fase implementada, se puede estimar la cantidad de tiempo necesaria para acabarlo entero. En total hay 35 fases, contando que diseñar, implementar e integrar una fase puede costar entre 30 y 40 minutos, nos sale un total de 20h para implementar las todas las fases del juego.

Las licencias del software son gratuitas ya que se ha utilizado la licencia Personal de Unity y una licencia de Windows para estudiantes. Para manejar las imágenes del juego se ha utilizado la herramienta gratuita Gimp.

Capítulo 7 – Conclusiones del proyecto

La conclusión principal que se puede obtener es que Unity es una herramienta increíble para desarrollar juegos, ofrece un framework que implementa todas las funcionalidades necesarias para programar cualquier tipo de juego, ya sea 2D o 3D. Además, su interfaz de usuario permite una forma de trabajar con un flujo ordenado y bien estructurado.

La filosofía de trabajar mediante componentes hace que la programación de estos sea reutilizable y extensible, ya que se pueden exportar fácilmente.

Es muy fácil de utilizar, cualquier persona sin conocimientos sobre gráficos o 3D puede desarrollar en Unity, ya que este ofrece unas funciones de alto nivel que se encargan de hacer el trabajo duro, como: pintar modelos 3D, rotaciones, traslaciones, shaders, etc. Además, la herramienta tiene una gran cantidad de contenidos y tutoriales en internet, ya sean en formato texto o en videos. También dispone de una comunidad grande y activa que participa en los foros de Unity proponiendo soluciones a problemas que plantean otros usuarios y reportando posibles bugs que pueda tener Unity.

Otra de las grandes ventajas, es que con un mismo proyecto de Unity puedes publicar para casi cualquier plataforma que existe hora mismo: Android, IOS, PC, etc. Esto hace que el producto pueda llegar al mayor número de personas sin tener que dedicar ningún tiempo extra de desarrollo.

En definitiva, esto hace que Unity sea una plataforma estupenda para el desarrollo de videojuegos, haciendo que el coste en tiempo y dinero se abarate considerablemente.

Como contra partida, tener un motor tan genérico hace que Unity sea más o menos bueno en todo, pero no el mejor en nada. Esta “generidad” hace que el motor no esté optimizado para ninguna tarea y no dará el mismo rendimiento que un motor implementado para un tipo de juego concreto.

Dejando a un lado el tema del rendimiento, que es algo más de cara a usuarios o empresas profesionales, el principal hándicap de Unity para el programador que se está iniciando es el precio de las licencias, ya que alguien que quiera empezar a publicar profesionalmente tendrá que realizar un desembolso considerable de dinero, que aumenta a medida que se quiere publicar para plataformas extras.

Capítulo 8 – Ampliaciones posibles

Para el futuro se han dejado los siguientes puntos como características interesantes a implementar:

- Ampliación de los niveles. Ahora mismo se dispone de una cantidad de niveles limitada.

- Multijugador online/offline: competir en carreras contra otro usuario en tiempo real.

- Persistencia online de las puntuaciones de los usuarios. Implementar un servidor que mantenga las puntuaciones de los jugadores y poder consultarlas a través del juego o de la web.

- Pasar el juego completamente a 3D. Con Unity se podría hacer el juego mucho más atractivo visualmente, cambiando los modelos que ahora son sprites por modelos 3D y añadiendo efectos de luces mucho más espectaculares.

- Publicación del contenido en alguna tienda de aplicaciones, como por ejemplo, PlayStore de Android o AppStore de iOS.

Bibliografía

Joseph Hocking. *Unity in action.*

https://www.manning.com/books/unity-in-action

Unity Technology. *Manual*

https://docs.unity3d.com/Manual/index.html

Unity Technology. *Scripting API*

https://docs.unity3d.com/ScriptReference/index.html

Unity Technology. Foros.

http://forum.unity3d.com/

Codigofacilito [Youtube]. *Curso de Unity*

https://www.youtube.com/playlist?list=PLpOqH6AE0tNiPHU2BCm_ei3C2BmjN9IGt

Brackeys [Youtube] *Unity Developer.*

https://www.youtube.com/channel/UCYbK_tjZ2OrIZFBvU6CCMiA

Personatges en joc [Web]

http://www.personatgesenjoc.cat/jugar/