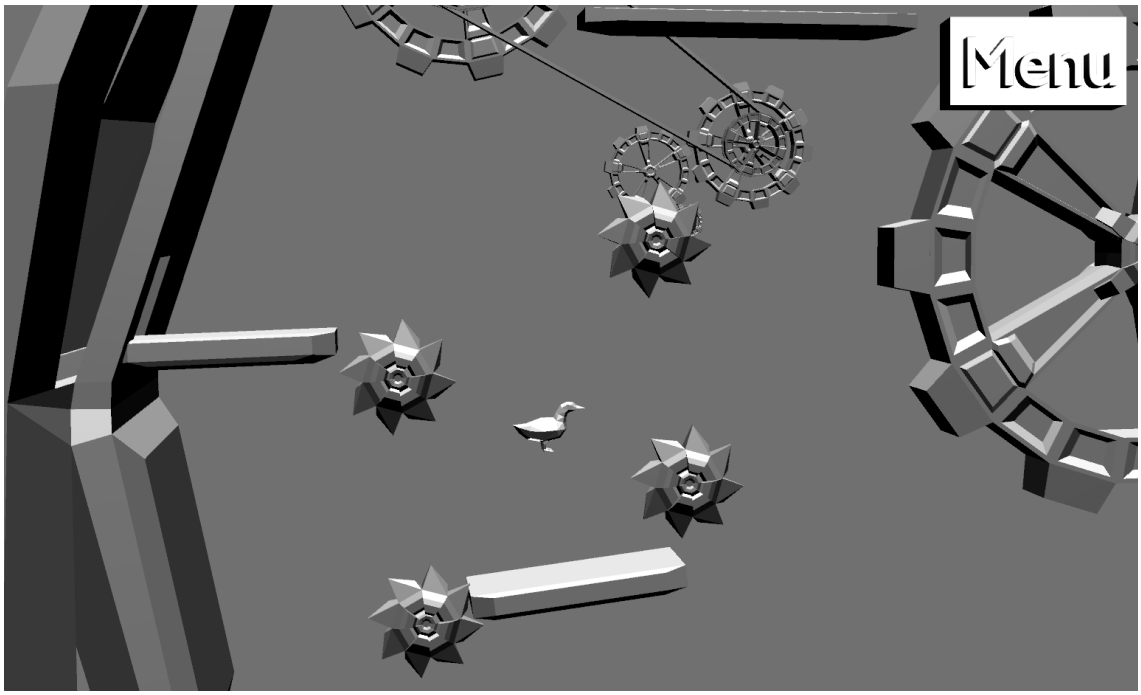
Institut Puig Castellar
Santa Coloma de Gramenet



Blender: Diseño de un videojuego

CFGM Sistemas Microinformáticos y redes

Pablo Jiménez Sanz

SMX 2B
31 / 5 / 2017

Resumen

El mundo de los videojuegos se encuentra en avance constante y, hoy en día, cualquier persona puede disfrutar de esta forma de entretenimiento, independientemente de sus características y preferencias.

Los desarrolladores cuentan cada vez con más herramientas y tecnologías que facilitan su trabajo y permiten realizar proyectos mas ambiciosos y avanzados.

Nos hemos propuesto realizar el proceso de desarrollo de un videojuego mediante el uso de un programa que agrupa gran parte de las herramientas y funciones que necesitaremos para ello, Blender, un programa libre principalmente utilizado para el modelado y animación 3D, pero que incluye grandes herramientas ideales para todo tipo de proyectos multimedia: Modelado, animación, renderizado, edición de vídeo e imagen, desarrollo de simulaciones y videojuegos, etc.

Utilizaremos las facilidades que nos ofrece Blender a la hora de realizar el desarrollo, como el apartado de físicas, los bloques lógicos para facilitar la programación y su cómoda interfaz visual.

Para realizar la programación que necesitaremos en el proyecto, prescindiremos todo lo posible de los bloques lógicos y utilizaremos el lenguaje Python, debido a su integración con Blender.

Siguiendo con el apartado visual, realizaremos uno a uno todos los modelados que necesitemos, utilizando las herramientas 3D de Blender.

Finalmente, exportaremos el proyecto de forma que pueda ser ejecutado en las plataformas mas populares, en este caso, sobre los sistemas operativos Linux y Windows.

Abstract

The videogame industry finds itself on constant evolution. Nowadays, anyone can enjoy this way of entertainment independently of its characteristics and preferences.

Game developers have access to more and more tools and technologies that ease their work and allow to develop more ambitious and advanced projects.

We have decided to follow the development process of a game. We will do so using a piece of software that includes the majority of tools and functionalities that we will need, Blender. Blender is a free software application mainly used in 3D modeling and animation, although it includes great tools ideal for every kind of multimedia project: Modeling, animation, render, video and image editing, simulation and game development, etc.

Blender offers us several elements that we will use in order to ease the progress of the project, as its physics, the logic bricks, that will ease the programming tasks and its convenient GUI.

In order to fulfill the programming tasks that we will need on our project, we will use the logic bricks for as little as possible, and use Python as a programming language, due to its integration with Blender.

On the visual section, we will create all the models we need, using the 3D tools Blender offers.

Finally, the project will be exported so it can be executed on both Linux and Windows operating systems.

Sumario

1. Introducción

1.1 Contexto y justificación del Trabajo

Desde principios de los años 50, con el uso dispositivos analógicos conectados a un osciloscopio con el objetivo de crear un medio interactivo, hasta los títulos más novedosos de la actualidad, que nos ofrecen un mundo lleno de detalles, pasando por la crisis de los setenta, y el posterior resurgimiento de la industria, los videojuegos han supuesto una gran parte del entretenimiento de la población en el último medio siglo.

Además de simple entretenimiento, el desarrollo de proyectos cada vez mas complejos y exigentes han contribuido con el avance de ciertas tecnologías, que resultan ciertamente útiles fuera de la propia industria del videojuego (necesidad de una mayor potencia de computación, creación de inteligencias artificiales mas complejas, desarrollo de proyectos de realidad virtual…)

Los títulos actuales poseen una gran cantidad de elementos unidos y combinados para obtener el resultado que los desarrolladores desean, desde la lógica del juego, que se encarga de mover los engranajes y crear una experiencia jugable, hasta el apartado artístico, que proporcionan al proyecto su aspecto visual, sonoro y, en ciertos casos, un argumento que nos une a los personajes y nos hace vivir los sucesos.

En nuestro proyecto, planeamos realizar el desarrollo de un videojuego en tres dimensiones, aunque con un apartado artístico no excesivamente complejo, y con una lógica completamente escrita por nosotros.

De igual manera que sucede con la gran parte de títulos actuales, utilizaremos un motor ya existente. Esto es un software creado específicamente para el desarrollo de videojuegos, que ayuda al desarrollador a agrupar y unir los elementos de su proyecto.
En nuestro específico caso, utilizaremos Blender.

Blender es un programa originalmente creado para la realización de animaciones en 3D inicialmente desarrollado por el estudio de animación neerlandés ‘Neo Geo’, que pasó a ser Software Libre tras la compra del estudio por otra compañía y la entrada en bancarrota de la que mantenía el proyecto de Blender. Hoy en día tanto los propios desarrolladores como la comunidad mantienen el programa y lo hacen crecer. Una de las funciones que el programa adquirió tras su liberación fue la de motor para videojuegos, con una gran integración con python. Esto, junto a su gran capacidad para la creación de modelados y animaciones, y su “facilidad” de uso, resultan en una gran herramienta para la realización de proyectos como el nuestro, pese a que no es la herramienta mas potente del mercado.

1.2 Motivaciones:

El mundo del videojuego se encuentra en expansión constante, que no ha dejado de acelerar en las últimas décadas. Podemos decir que una enorme parte de la población es capaz de disfrutar de las experiencias que estos nos ofrecen, sin importar edad, sexo o preferencias.

La variedad de títulos y géneros es inmensa, así como la intención de los autores, desde las grandes compañías que buscan satisfacer al público, en busca del máximo beneficio, hasta los desarrolladores que ponen una parte de ellos mismos en su trabajo, y crean verdaderas obras de arte interactivas.

Igual de diversas son las motivaciones que provocan a nuevos desarrolladores a entrar en este mundo. En nuestro caso, lo que mas nos seduce de este mundo es la unión de la expresión artística con el trabajo técnico.

1.3 Objetivos del Trabajo

Nuestro objetivo principal es el de completar el desarrollo de un videojuego simple, utilizando técnicas similares a las utilizadas comúnmente en el desarrollo de los títulos actuales, pero mediante el uso de herramientas y procesos dentro de nuestro alcance, con el objetivo de conocer y comprender el proceso seguido en la realización de estos proyectos:

- Comprensión de la utilización de la herramienta principal (motor)
  - Conocimiento de las funciones y herramientas que nos ofrece el software utilizado.

- Diseño e implementación de la lógica necesaria para las funciones propias del proyecto
  - Aprendizaje de un nuevo lenguaje de programación (Python).
  - Conocimiento de las librerías encargadas de la interacción entre el código y el motor.

- Creación de elementos visuales propios
  - Conocimiento del proceso de creación de modelados
  - Realización de los modelados necesarios
  - Conocimiento del proceso de creación y implementación de texturas y materiales
  - Realización de las texturas y materiales necesarios

- Creación de los elementos sonoros.

- Diseño y realización de los escenarios.

- Unión y integración de los elementos para la obtención del resultado final.

- Exportación del proyecto.

1.4 Enfoque y método seguido

Si bien, como ya hemos especificado anteriormente, la utilización de un motor desarrollado previamente es la practica mayoritaria, es posible realizar el desarrollo completo desde un fichero en blanco.

Existen diversas metodologías comunes a la hora de realizar un proyecto de este tipo, entre las cuales existían dos opciones que llegamos a considerar:

- **Desarrollo completo mediante el uso de librerías:**

No es poco usual que, sobretodo al tratarse de desarrolladores independientes, se prefiera escribir todo el código que compondrá el proyecto, ahora bien, delegando las funciones mas complejas, como la interacción necesaria con el dispositivo gráfico o con los controladores, en el uso de librerías ya existentes. Esta opción permite al desarrollador escoger su lenguaje de preferencia, y las librerías que desea utilizar.

Buenos ejemplos de librerías libres son Allegro5 o SDL.

El inconveniente que encontramos en este método es la complejidad del proyecto y la necesidad de conocimientos previos, incrementada todavía mas si se desean utilizar modelados en tres dimensiones.

- **Desarrollo mediante el uso de un motor:**

El uso de estas herramientas simplifican el proceso, tanto por la posibilidad de utilizar una interfaz gráfica, como por el hecho de que cuenten con ciertos elementos utilizables en el proyecto, como un sistema de físicas.

Existen herramientas diseñadas específicamente para 2D o 3D, y gran cantidad de alternativas dentro de cada categoría.

Dentro de estas, nos decantamos por el uso de Blender por encima de otras herramientas como Unity o UDK por el hecho de tratarse de Software Libre, a diferencia de las otras alternativas, pero sobre todo, por la gran cantidad de herramientas que nos ofrece Blender: Un sistema de creación de modelados y animaciones, el propio motor, e incluso herramientas de edición de texto, vídeo, sonido e imagen.

1.5 Planificación del proyecto

- **Recursos:**

Sistema en el que se ha realizado el proyecto:

La mayor parte del proyecto se ha realizado en un sistema con las siguientes especificaciones:

- Intel Core i5-4460 @ 3.20GHz
- 8GB Ram
- Ubuntu 16.04 - 64Bits

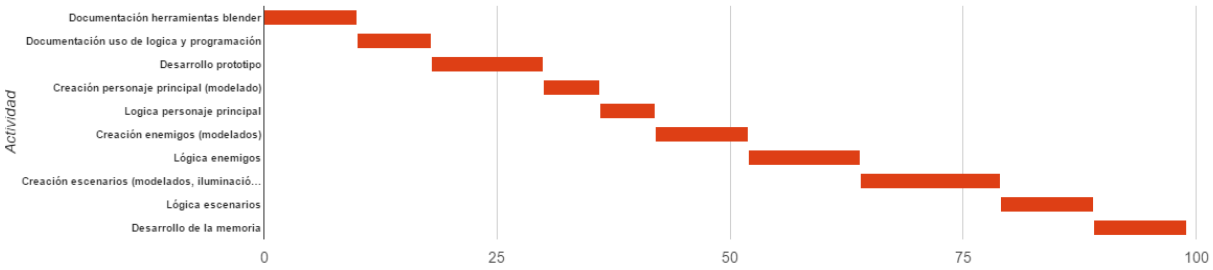
- **Software:**

En cuanto al programario utilizado, el software principal y en el que se ha basado el proyecto es Blender, utilizado como motor del videojuego, herramienta de modelado y editor de código.

- **Lenguaje de programación:**

El lenguaje de programación escogido es Python, debido a su integración nativa con Blender.

- **Tiempo:**



La planificación realizada al principio del proyecto ha acabado suponiendo uno de los mayores problemas con el que nos hemos encontrado.

En esta dedicábamos muy poco tiempo a tareas que han acabado resultando mas complejas de lo pensado en un principio, e incluso ciertas ideas se ha tenido que abandonar completamente, por lo que esta planificación no ha sido utilizada.

2. Software empleado: Blender

Como se ha especificado anteriormente, Blender es un programa centrado en el modelado, animación y renderización de elementos tridimensionales, pero que ha ido incorporando una multitud de nuevas funciones a lo largo del tiempo.

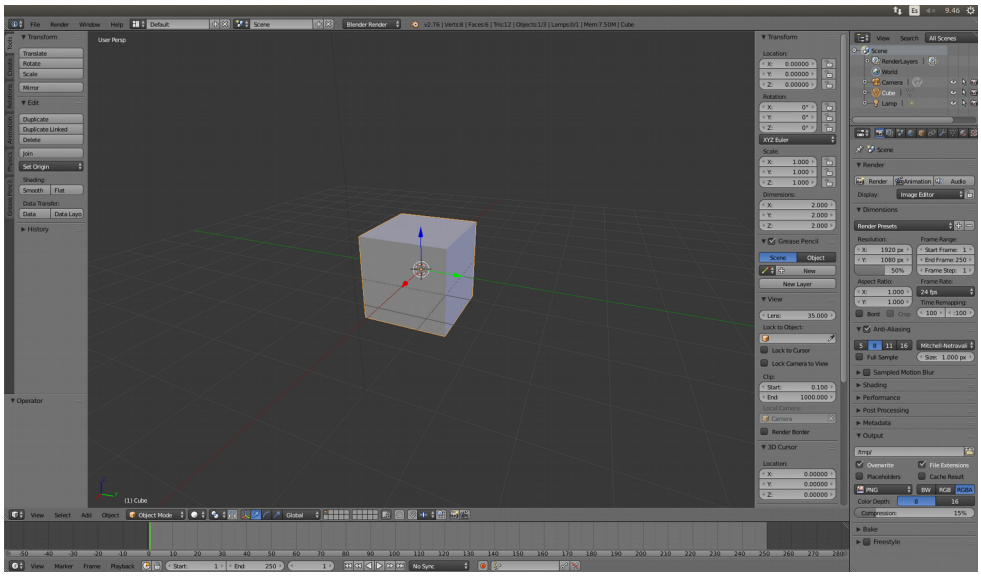
Actualmente, posee una gran variedad de herramientas que lo convierten en un programa perfectamente competente para casi cualquier clase de producción multimedia, incluyendo funciones de modelado, renderización, animación, edición de vídeo, creación de efectos especiales y de postprodución, composición, texturizado, creación de esqueletos, diversos tipos de simulación y desarrollo de videojuegos.

Otras cualidades que lo convierten en una gran herramienta son las siguientes:

- Es Distribuido bajo la licencia GPL.
- Es multiplataforma, con una GUI basada en OpenGL, uniforme en todas las plataformas mayores y personalizable mediante scripts de Python.
- Posee una arquitectura 3D de gran calidad que permite un flujo de trabajo rápido y eficiente
- Dispone de un gran soporte de la comunidad, vía foros o IRCs
- Tamaño ligero, posibilidad de ser ejecutado como portable.

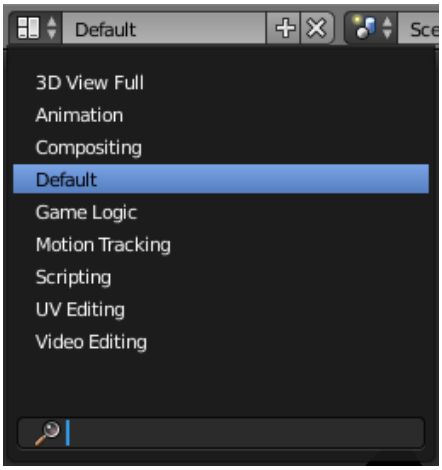
Interfaz del programa:

Nada mas abrir el programa por primera vez, podremos ver una ventana muy similar a la siguiente:



Esta es la interfaz por defecto de Blender. En ella podemos encontrar las herramientas principales del programa, y esenciales para el modelado.

Una de las grandes comodidades de este software, es la posibilidad de cambiar entre las diferentes interfaces predefinidas que nos ofrece, cada una de las cuales se especializa en una función, y pone a fácil disposición las herramientas mas importantes para cada tipo de trabajo.



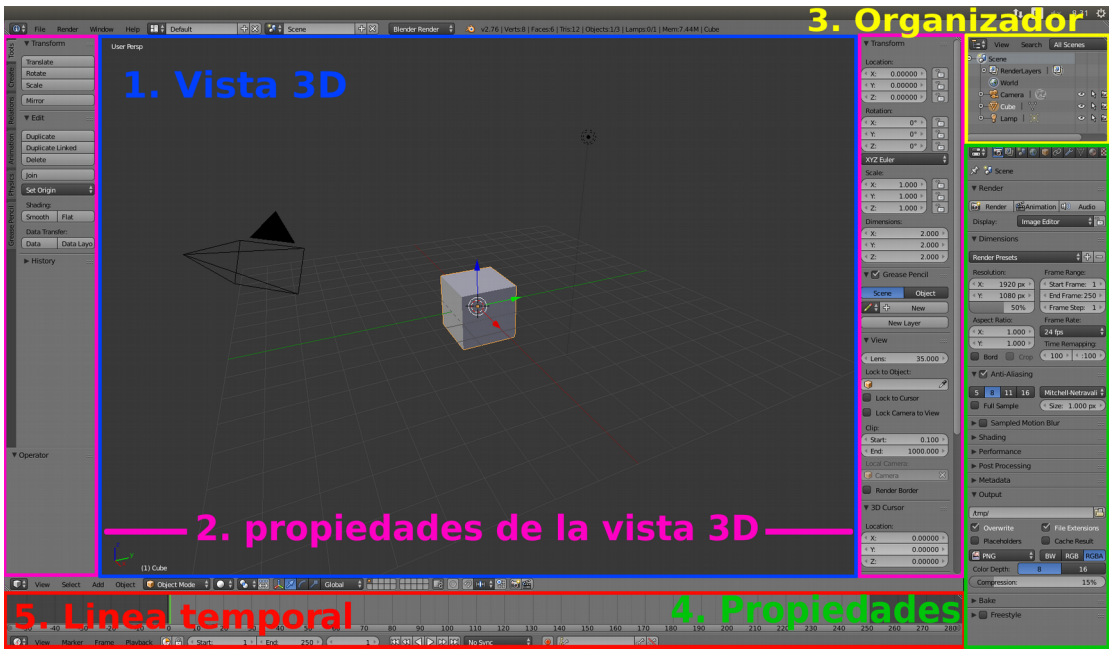
Además de esto, disponemos de la posibilidad de personalizar la interfaz a nuestro gusto, intercambiando las ventanas y redimensionandolas. Adicionalmente, en caso de que lo creamos conveniente, podemos guardar estas nuevas disposiciones de la interfaz, para poder utilizarla de nuevo mas tarde.

En nuestro caso, hemos utilizado predominantemente dos de las interfaces que nos ofrece el programa, la disposición por defecto, y la especializada en el desarrollo del juego.

A continuación, realizaremos una pequeña revisión de los elementos que nos ofrecen y sus funciones.

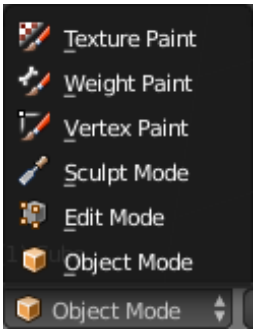
Interfaz: Default

La interfaz por defecto de Blender se centra en proporcionarnos las herramientas más importantes del programa, principalmente orientadas al modelado y al proceso básico de animación.



1. Vista 3D

Esta ventana es utilizada para interactuar con la escena 3D. Nos permite visualizar los elementos, así como moverlos y editarlos.



Menú de modos de edición

Los objetos pueden ser modificados de varias formas. Una vez seleccionados, podemos acceder a los distintos modos de edición utilizando el menú desplegable situado en la parte inferior de la ventana de vista 3D.

Object Mode:

El modo por defecto nos permite mover, rotar o escalar los objetos deseados. Es el modo principal para modificar la disposición de los elementos en la escena.

Edit Mode:

Este modo nos permite editar los vértices, lados y caras de los objetos, asi como crear nuevos. Por esto es el modo principal para editar la forma de los objetos.

Sculpt Mode:

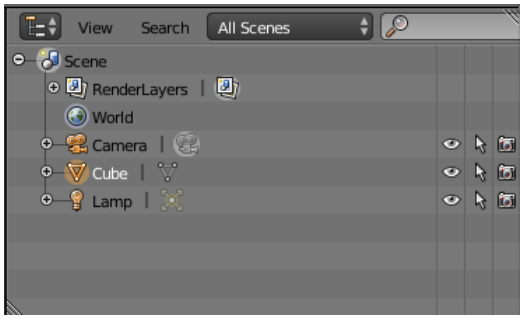
El modo escultura nos proporciona un entorno en el cual podremos tratar la malla de los objetos como si de una escultura se tratara, modificándola mediante el uso de pinceles.

2. Propiedades de la vista 3D

Estos menús forman parte de la vista 3D, y pueden ser ocultados y mostrados a necesidad del usuario. Contienen propiedades de los objetos activos y los seleccionados, así como propiedades del propio editor, que varían en función del modo que este siendo utilizado.

3. Organizador

Contiene una lista con todos los datos incluidos dentro del archivo, como los elementos de la escena, la información de esta, o las preferencias del usuario

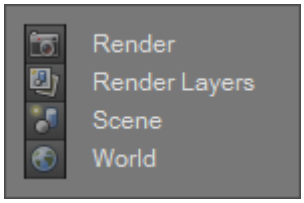


4. Propiedades

El editor de propiedades es utilizado para editar datos y propiedades de la escena y el objeto seleccionado.

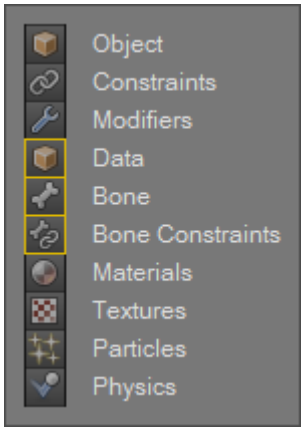
El editor dispone de varias pestañas, que según las funciones y propiedades que agrupan, podemos dividir en dos tipos, las de escena y renderizado, y las de objetos y su información.

Las pestañas de escena y renderización agrupan las opciones y funciones referentes a como actuará el motor de renderizado, las propiedades de la escena actual, y ciertas características sobre el fondo o la iluminación del mundo.



Pestañas de escena y renderización

Las pestañas de objetos y sus datos contienen las opciones referentes a las propiedades individuales del objeto seleccionado actualmente, desde las más básicas, como su localización, rotación o escala, hasta las físicas que debe seguir el objeto a la hora de interactuar con el resto de elementos, pasando por modificadores, que editarán la malla del objeto siguiendo ordenes preestablecidas, pero configurables, los materiales o texturas que le aportarán color o, en caso de que así lo deseemos, las partículas que puede emitir.



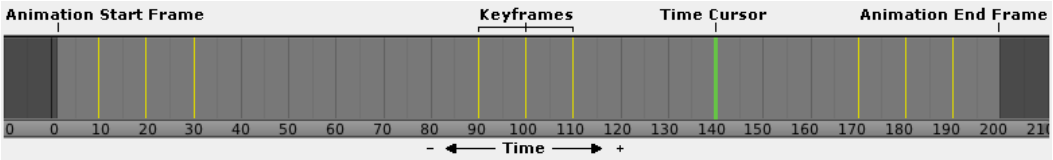
Pestaña de objetos y datos de objetos

5. Linea temporal

Esta ventana, mas que un editor, nos aporta información y control.

Esta ventana nos proporciona una visión del apartado de animación de la escena:
el cuadro, o frame, de tiempo en el que nos encontramos, donde se encuentran los frames claves del objeto activo, los frames de inicio y final de la animación, marcadores, etc.

Dispone además de controles para reproducir y pausar la animación, y saltar a través de la escena.

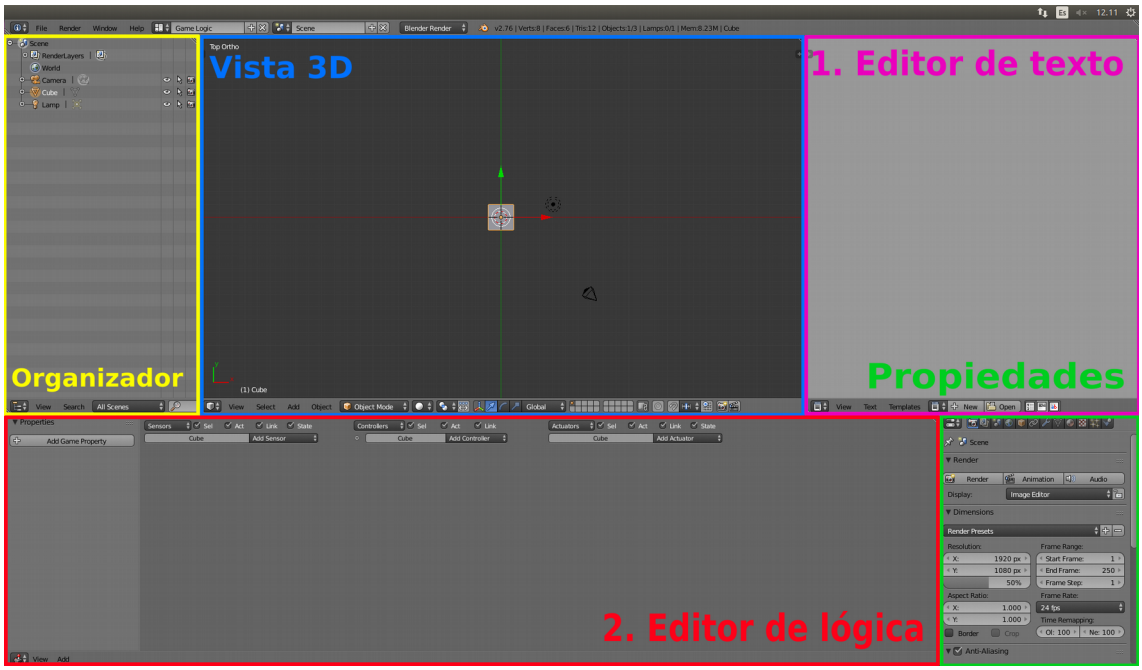


Linea temporal

Gracias a todas las herramientas mencionadas, la vista por defecto nos provee de los recursos que necesitamos para la creación y edición de modelados. El fácil acceso a estas y sus posibilidades de personalización nos proporciona, una vez comprendemos las herramientas y como son utilizadas, un buen flujo de trabajo.

Interfaz: Game Logic

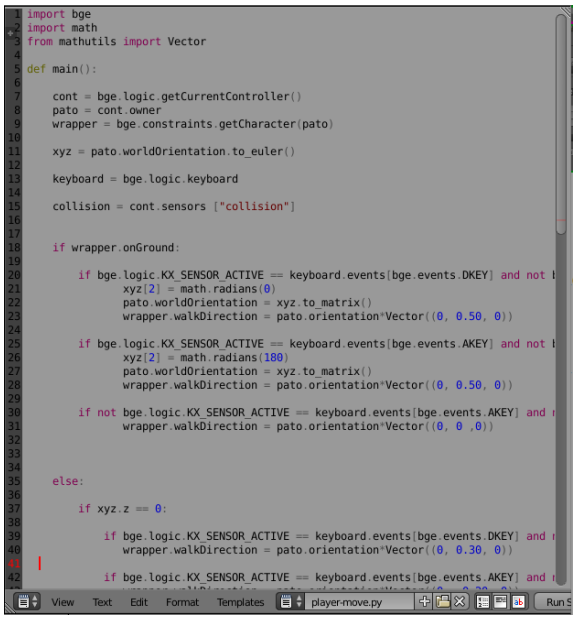
Esta interfaz, especializada en el desarrollo de la lógica del juego, nos vuelve a proporcionar algunas de las ventanas que ya nos proporcionaba la interfaz estándar, pero nos ofrece dos nuevas, el editor de texto y el editor de lógica, esenciales para dicha función.



1. Editor de texto

El editor de texto nos permite crear ficheros de documentación dentro del propio fichero .blend, ideales si se desea compartir el proyecto con otros usuarios.

Además de esto, y la principal función por lo que nos es de gran ayuda, nos ofrece la posibilidad de crear y editar scripts, lo que unido al interprete de Python integrado en Blender, nos facilita enormemente la tarea de programación.

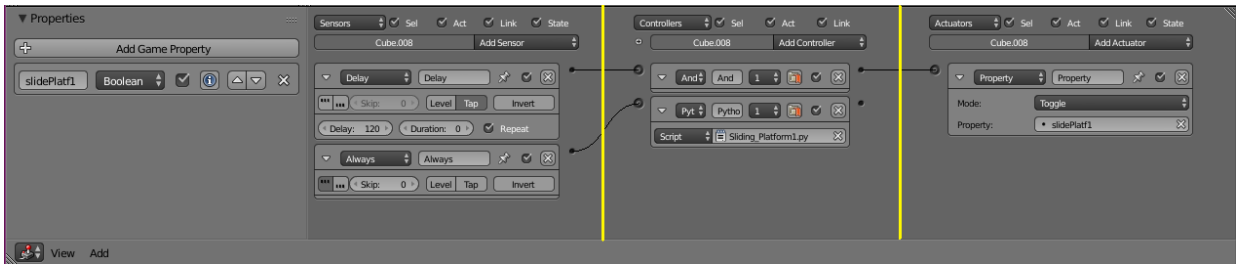


2. Editor de lógica

Esta ventana nos proporciona las herramientas necesarias para el desarrollo de la lógica del juego.

La funcionalidad del editor se basa en la creación y configuración de bloques lógicos y las relaciones que se asignen entre ellos. Esta es una forma muy simple e intuitiva de crear los procesos lógicos que se necesiten en el juego, sin necesidad de conocimientos de programación.

Podemos considerar que el menú se encuentra dividido en tres secciones, según el tipo de bloques que ofrecen: Sensores, Controladores y Actuadores.



Sensores:

Los sensores son los bloques lógicos que causan el inicio del proceso. Estos generan una señal cuando se cumple la condición indicada.

Los Sensores mas importantes que hemos utilizado son los siguientes:

Collision:

Este Sensor genera una señal al detectar que el objeto se encuentra en contacto con otro.

Delay:

Genera una señal cuando pasa el tiempo indicado. Se puede especificar si se desea que se reinicie el temporizador una vez activado, de forma que se active cada vez que se cumpla el tiempo de forma infinita.

Always:

Dependiendo de lo que se indique, este bloque generará una señal al ejecutar el juego, o una señal por cada ciclo del juego (generalmente, 60 veces por segundo).

Controladores:

Estos bloques recogen la señal enviada por los sensores y aplican unas condiciones lógicas a estas. Tras esto, envían la señal a los actuadores.

Los Controladores mas importantes que hemos utilizado son los siguientes:

AND:

Este bloque genera una señal cuando todos los sensores conectados a este se encuentren activos.

Python:

A diferencia de los demás controladores, este bloque no genera ninguna señal, sino que ejecuta el script de Python que se le especifique

Actuadores:

Cuando reciben una señal, ejecutan la acción que se les ha asignado.

Los Actuadores mas importantes que hemos utilizado son los siguientes:

Property:

Este bloque permite modificar de distintas formas las propiedades del objeto en el que se encuentra, desde aumentar o reducir el valor si esta se trata de un integer, cambiar su valor si se trata de un booleano, etc

Scene:

Este bloque nos proporciona la posibilidad de cambiar la escena que se está visualizando, o incluso añadir una escena por encima de la actual.

3. Tareas previstas inicialmente:

- Diseño y creación de los elementos visuales
  - Diseño y creación de los modelados
  - Diseño y creación de las texturas y materiales

- Diseño y creación de los escenarios
  - Organización y posicionamiento de los elementos visuales
  - Posicionamiento de la cámara
  - Posicionamiento de la iluminación

- Diseño y creación de la interfaz
  - Creación del menú
  - Creación del overlay

- Creación de los elementos lógicos
  - Escritura de los scripts necesarios
  - Enlazado de los bloques lógicos.

- Creación de los elementos sonoros
  - Creación de los efectos de sonido
  - Creación de la música

- Creación de esqueletos y animación de los personajes

Aunque obtener un resultado profesional no era una de nuestras expectativas, el proyecto que nos propusimos en un primer momento se encontraba lejos de nuestras posibilidades. A medida que hemos avanzado, por desgracia, nos hemos visto en la necesidad de prescindir de algunos de los elementos que deseábamos incluir, con el objetivo de poder dedicar el tiempo necesario a las partes mas esenciales y importantes del proyecto.

Algunos de estos elementos han sido los siguientes:

Creación del esqueleto y animación de los personajes: Debido a su complejidad y a la enorme cantidad de tiempo necesario para la obtención de un resultado que no acabaría resaltando en exceso.

Creación de los elementos sonoros: Si bien este apartado no habría supuesto una dificultad tan grande como la que representa la animación, todo y que habría implicado la necesidad del aprendizaje de nuevas herramientas, no hemos dispuesto de tiempo suficiente para implementarlo en nuestro proyecto.

Creación de texturas y materiales: Pese a que este podría considerarse el segundo elemento mas importante referente al aspecto visual de un proyecto de estas características, nos habría obligado a aprender el funcionamiento de nuevo programario y la realización de un proceso creativo adicional, por lo que, debido a la falta de tiempo, nos hemos visto obligados a prescindir de ello.

Creación de enemigos: Debido a la decisión de que el juego se basase mas en la interacción entre el jugador y obstáculos presentes en el terreno, la presencia de enemigos pasó a no ser necesaria.

Conocimientos previos:

Para la realización de estas tareas necesitaríamos un seguido de conocimientos previos de la herramienta y los procesos a seguir.

Al comenzar el proyecto, disponíamos de ciertas nociones sobre el funcionamiento de Blender y la utilización de su interfaz. Ya habíamos creado algunos modelados y habíamos interactuado con las herramientas más básicas y esenciales que nos ofrece el programa. Estos conocimientos nos han permitido dar los primeros pasos en el proyecto, aunque a medida que avanzamos, como ya teníamos previsto, nos vimos en la necesidad de extenderlos y documentarnos acerca de las nuevas herramientas y las funciones que nos ofrecen las herramientas que ya habíamos utilizado y desconocíamos o no habíamos explorado, con el objetivo de continuar el progreso del proyecto.

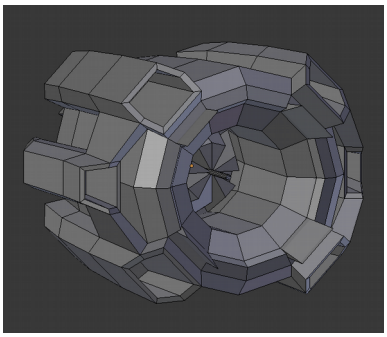
4. Desarrollo

1. Desarrollo Visual

Dentro del apartado visual del proyecto, podemos dividir el trabajo realizado en dos grandes elementos, los modelados individuales, y los escenarios, resultado de la agrupación de estos en una escena.

Modelados:

El estilo por el que hemos apostado a la hora de diseñar el apartado visual, centrándonos mayoritariamente en los modelados, es un estilo simple que hace un gran uso de formas geométricas. Este estilo, bastante popular en proyectos en los que no se dispone de personal especializado en este proceso creativo, permite a los desarrolladores obtener un resultado atractivo, sin necesidad de invertir grandes esfuerzos y tiempo. Además, el hecho de que los modelados finalizados no cuenten con un elevado número de polígonos, reduce la potencia necesaria para ejecutar el juego.

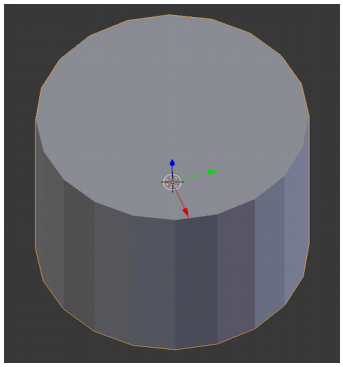


Modelado final de los motores que aparecen en el segundo escenario

En el momento de la creación de los modelados, hemos seguido una serie de pasos:

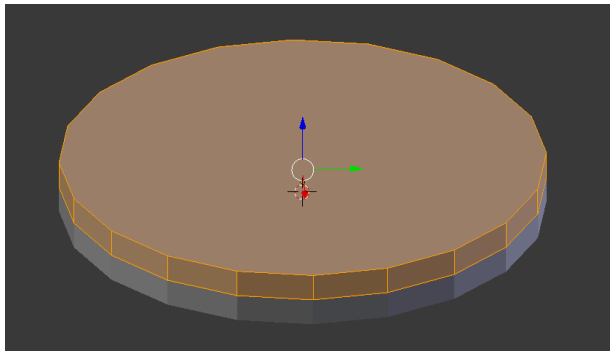
1. Forma base del objeto:

Una vez decidido el objeto que se desea modelar, se busca una forma básica desde la que modelarlo. En este caso, modelaremos uno de los engranajes que podemos encontrar en el segundo nivel. Para este ejemplo comenzaremos con un cilindro.



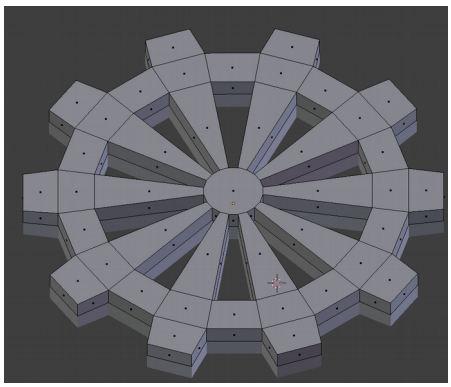
2. Escalado y forma general:

Una vez decidida la forma básica que mas nos ayudará, reescalaremos los lados que necesitemos para acercarnos mas a la forma final.



A la hora de modelar, sobre todo elementos simétricos, existe un modificador que nos ahorrará una buena parte de trabajo, este es el modificador espejo, que imitará los cambios que realicemos en el eje que le indiquemos.

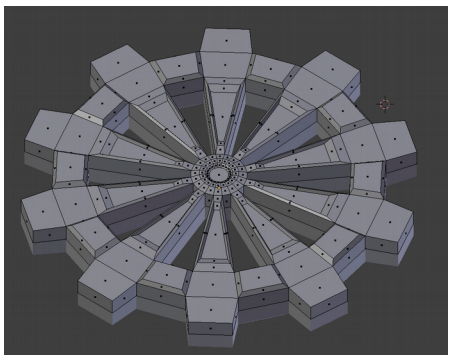
3. Extrusión



Mediante la herramienta de extrusión, podemos generar nuevas caras en el modelado, a la vez que podemos eliminar otras, dando mas forma al diseño.

Una vez en este punto, ya poseemos un elemento con una forma similar a la que habíamos ideado en un principio.

4. Detallado



Finalmente, de nuevo mediante el uso de las herramientas de extrusión y escalado, o editando los vértices manualmente, podemos crear detalles el modelo.

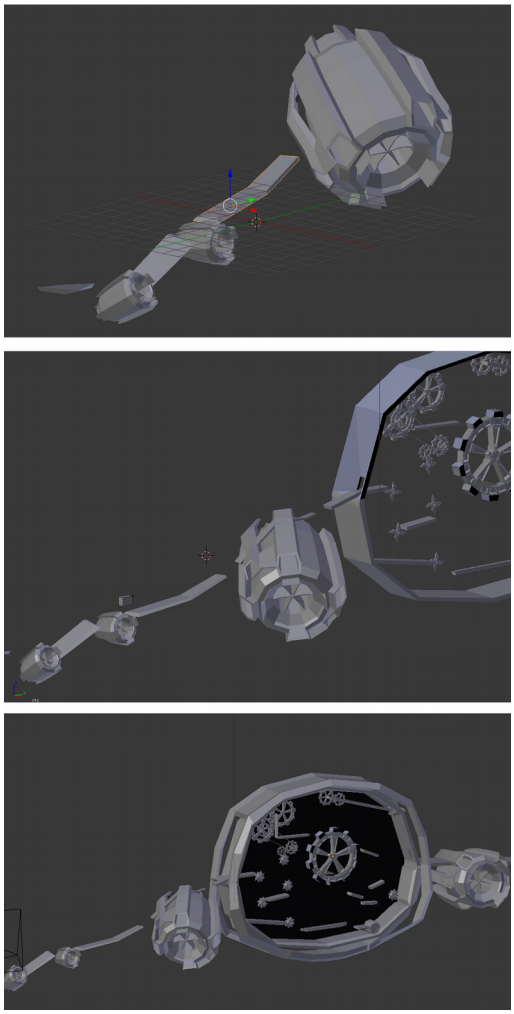
Una vez consideremos que hemos obtenido el resultado deseado, habremos finalizado el proceso de creación.

Escenarios:

Una vez disponemos de todos los modelados que formarán el escenario, estos deben ser colocados en la escena uno a uno, creando la estructura del nivel.

Tras esto, puede ser necesario ajustar los bordes de colisiones de forma que el terreno permita al personaje avanzar como es deseado.

Finalmente, podemos realizar los retoques que creamos necesarios en los modelados. Añadimos la iluminación y cámaras necesarias, y finalmente, incluimos la lógica necesaria para el nivel.



Proceso de montaje del escenario

2. Desarrollo Lógico:

El desarrollo de la lógica del juego ha sido la actividad que que mas tiempo ha consumido, además de ser el objetivo principal del proyecto.

A continuación se enumera una lista de las acciones mas importantes que hemos implementado.

- **Personaje principal:**

  - **Movimiento terrestre del personaje:**

```
    if wrapper.onGround:

        if bge.logic.KX_SENSOR_ACTIVE == keyboard.events[bge.events.DKEY] and not bge.logic.KX_SENSOR_ACTIVE == keyboard.events[bge.events.AKEY]:
               xyz[2] = math.radians(0)
               pato.worldOrientation = xyz.to_matrix()
               wrapper.walkDirection = pato.orientation*Vector((0, 0.50, 0))

        if bge.logic.KX_SENSOR_ACTIVE == keyboard.events[bge.events.AKEY] and not bge.logic.KX_SENSOR_ACTIVE == keyboard.events[bge.events.DKEY]:
               xyz[2] = math.radians(180)
               pato.worldOrientation = xyz.to_matrix()
               wrapper.walkDirection = pato.orientation*Vector((0, 0.50, 0))

        if not bge.logic.KX_SENSOR_ACTIVE == keyboard.events[bge.events.AKEY] and not bge.logic.KX_SENSOR_ACTIVE == keyboard.events[bge.events.DKE
               wrapper.walkDirection = pato.orientation*Vector((0, 0 ,0))
```

Mientras el personaje se encuentre en contacto con el terreno, las teclas A y D del teclado le permiten girarse y moverse hacia la derecha y la izquierda de la pantalla.

  - **Movimiento aéreo del personaje:**

```
else:

    if xyz.z == 0:

        if bge.logic.KX_SENSOR_ACTIVE == keyboard.events[bge.events.DKEY] and not bge.logic.KX_SENSOR_ACTIVE == keyboard.events[bge.events.AKEY]:
           wrapper.walkDirection = pato.orientation*Vector((0, 0.30, 0))

        if bge.logic.KX_SENSOR_ACTIVE == keyboard.events[bge.events.AKEY] and not bge.logic.KX_SENSOR_ACTIVE == keyboard.events[bge.events.DKEY]:
           wrapper.walkDirection = pato.orientation*Vector((0, -0.20, 0))


        if not bge.logic.KX_SENSOR_ACTIVE == keyboard.events[bge.events.AKEY] and not bge.logic.KX_SENSOR_ACTIVE == keyboard.events[bge.events.DKE
           wrapper.walkDirection = pato.orientation*Vector((0, 0 ,0))

    else:

        if bge.logic.KX_SENSOR_ACTIVE == keyboard.events[bge.events.DKEY] and not bge.logic.KX_SENSOR_ACTIVE == keyboard.events[bge.events.AKEY]:
           wrapper.walkDirection = pato.orientation*Vector((0, -0.20, 0))

        if bge.logic.KX_SENSOR_ACTIVE == keyboard.events[bge.events.AKEY] and not bge.logic.KX_SENSOR_ACTIVE == keyboard.events[bge.events.DKEY]:
           wrapper.walkDirection = pato.orientation*Vector((0, 0.30, 0))


        if not bge.logic.KX_SENSOR_ACTIVE == keyboard.events[bge.events.AKEY] and not bge.logic.KX_SENSOR_ACTIVE == keyboard.events[bge.events.DKE
           wrapper.walkDirection = pato.orientation*Vector((0, 0 ,0))
```

Cuando el personaje no se encuentre en contacto con el suelo, no podrá girarse y se desplazará a una mayor velocidad en la dirección hacia la que se encuentre mirando.

- **Salto del personaje:**

```
if bge.logic.KX_SENSOR_ACTIVE == keyboard.events[bge.events.SPACEKEY]:
          wrapper.jump()
```

Éste realizará un salto cuando se pulse el botón espacio del teclado.

- **Muerte del personaje:**

```
def main():

    cont = bge.logic.getCurrentController()
    pato = cont.owner
    collision = cont.sensors ["collision"]

    if collision.positive:
            target = collision.hitObject
            list = collision.hitObject.getPropertyNames()
            for x in list:
                if x == "muerte":
                    for scene in bge.logic.getSceneList():
                        scene.end()
                    bge.logic.addScene('Menu')


main()
```

Cuando entre en contacto con uno de los elementos que contienen la propiedad llamada “muerte”, finalizará la escena actual y devolverá al jugador al menú principal.

- **Primer escenario:**

  - **Cámara**

```
def main():

    cont = bge.logic.getCurrentController()
    own = cont.owner
    scn = bge.logic.getCurrentScene()

    pato = scn.objects["1patobox"]

    patopos = pato.position
    x = patopos[0]
    y = patopos[1]
    z = patopos[2]

    cam = own.position
    camx = cam[0]
    camy = cam[1]
    camz = cam[2]


    own.worldPosition = [ camx , y, camz]


main()
```

Comprueba constantemente las coordenadas del personaje principal y mantiene la cámara unida a éste.

- **Plataformas:**

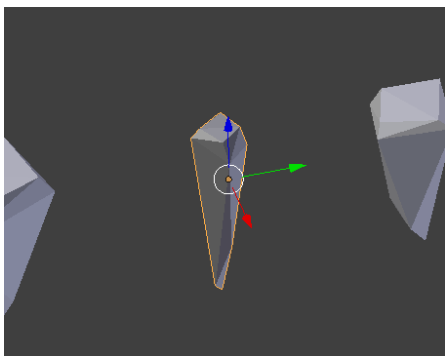
```
def main():

    cont = bge.logic.getCurrentController()
    own = cont.owner
    collision = cont.sensors ["collision"]

    if collision.positive:
            target = collision.hitObject
            list = collision.hitObject.getPropertyNames()
            for x in list:
                if x == "vida":
                    own ["fallPlatf1"] = True

    if own ["fallPlatf1"] == True:
        own.applyMovement([0, 0, -0.3], False)

main()
```



Cuando la plataforma entra en contacto con algún objeto que contenga la propiedad “vida”, el personaje principal, esta cae.

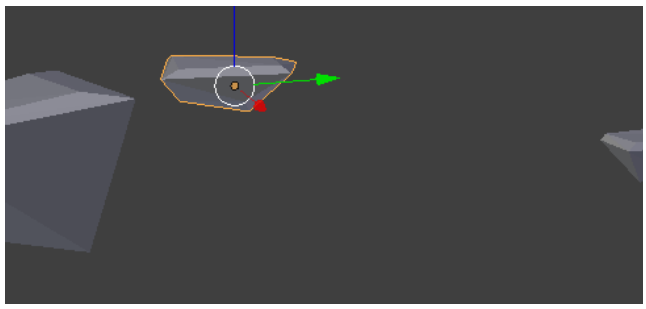
```
def main():

    cont = bge.logic.getCurrentController()
    own = cont.owner


    if own ["movPlatf1"] == True:
        own.applyMovement([0, 0, -0.2], False)

    else:
        own.applyMovement([0, 0, 0.2], False)

main()
```



Comprueba constantemente la propiedad “movPlatf1”, cuyo valor varía periódicamente entre cierto y falso, haciendo que la plataforma se mueva de un lado al otro constantemente.

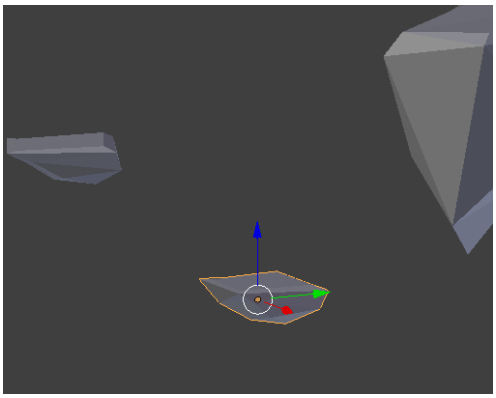
```
def main():

    cont = bge.logic.getCurrentController()
    own = cont.owner
    collision = cont.sensors ["collision"]

    if collision.positive:
            target = collision.hitObject
            list = collision.hitObject.getPropertyNames()
            for x in list:
                if x == "vida":
                    own ["upPlatf1"] = True

    if own ["upPlatf1"] == True:
        own.applyMovement([0, 0, 0.7], False)

main()
```

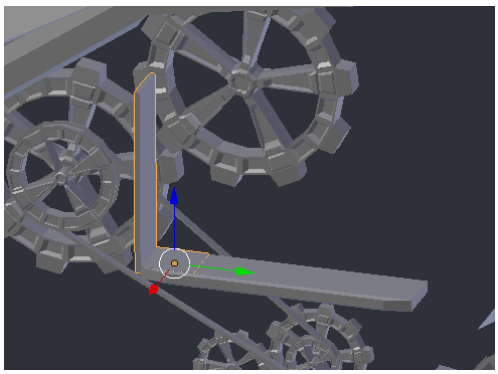


Cuando esta plataforma entra en contacto con un elemento con la propiedad “vida”, se eleva.

- **Segundo Escenario**

  - **Final del nivel**

```
if collision.positive:
        target = collision.hitObject
        list = collision.hitObject.getPropertyNames()
        for x in list:
            if x == "vida":
                own ["Final"] = True
```



Cuando el personaje entra en contacto con esta plataforma, varía la propiedad llamada “final”, lo que provoca ciertos cambios en el escenario.

- **Final = false**

```
if own ["Final"] == False:

    saw1.applyRotation([3, 0, 0], False)
    saw2.applyRotation([3, 0, 0], False)
    saw3.applyRotation([3, 0, 0], False)
    saw4.applyRotation([3, 0, 0], False)
    saw5.applyRotation([3, 0, 0], False)
    saw6.applyRotation([3, 0, 0], False)

    gear1.applyRotation([-0.012, 0, 0], False)
    gear2.applyRotation([0.012, 0, 0], False)
    gear3.applyRotation([0.028, 0, 0], False)
    gear4.applyRotation([-0.04, 0, 0], False)
    gear5.applyRotation([0.1, 0, 0], False)
    gear6.applyRotation([0.018, 0, 0], False)
    gear7.applyRotation([-0.018, 0, 0], False)
    gear8.applyRotation([0.007, 0, 0], False)
    gearplatfs.applyRotation([0.007, 0, 0], False)

    pulley1.applyRotation([0.012, 0, 0], False)
    pulley2.applyRotation([0.028, 0, 0], False)

    finalplatf.setLinearVelocity([0, 0, 0], False)
    finalplatf.applyRotation([0, 0, 0], False)
```

Antes de que eso se produzca, los engranajes, las sierras y las otras decoraciones cuentan con una velocidad de rotación determinada.

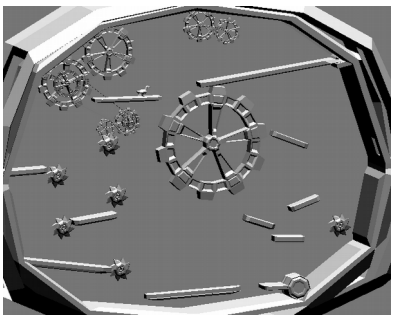
- **Final = true**

```
if own ["Final"] == True:
    support.worldPosition = [ 0.0, 0.0, 300.0]

    saw1.applyRotation([-0.03, 0, 0], False)
    saw2.applyRotation([-0.07, 0, 0], False)
    saw3.applyRotation([-0.05, 0, 0], False)
    saw4.applyRotation([-0.09, 0, 0], False)
    saw5.applyRotation([-0.01, 0, 0], False)
    saw6.applyRotation([-0.02, 0, 0], False)

    gear1.applyRotation([-0.001, 0, 0], False)
    gear2.applyRotation([0.0054, 0, 0], False)
    gear3.applyRotation([0.009, 0, 0], False)
    gear4.applyRotation([-0.008, 0, 0], False)
    gear5.applyRotation([0.03, 0, 0], False)
    gear6.applyRotation([0.005, 0, 0], False)
    gear7.applyRotation([-0.007, 0, 0], False)

    pulley1.applyRotation([0.0054, 0, 0], False)
    pulley2.applyRotation([0.009, 0, 0], False)
```



Una vez se ha activado la plataforma, la rotación de los elementos cambia, y se abre la salida del nivel.

5. conclusiones

El desarrollo de este proyecto nos ha permitido observar el proceso de diseño y creación de un videojuego, pudiendo apreciar el trabajo y los elementos necesarios para este.

Si bien no se han cumplido todos los objetivos que se habían propuesto en un principio, no nos encontramos disgustados por el resultado final.

El principal motivo por el que no ha sido posible cumplir todo lo que se había propuesto ha sido una mala valoración inicial del trabajo y tiempo que sería necesario para realizarlos, intentando incluir elementos complejos y avanzados sin disponer de experiencia ninguna.

Debido a esto, seguir la planificación realizada en un comienzo no era viable, por lo que el orden en el que se han producido las tareas necesarias para el proyecto y el tiempo dedicado a estas ha variado dependiendo de la necesidad en el momento.

De cara a futuros proyectos, hemos comprendido la necesidad de analizar mas exhaustivamente las tareas y sus requisitos, con el objetivo de evitar situaciones similares a la sucedida.

Bibliografía:

- **Página web oficial de la documentación de Blender:**

  https://docs.blender.org/manual/en/dev/index.html

- **Wiki oficial de Blender:**

  https://wiki.blender.org/

- **Página web oficial de la documentación de las librerías de Blender para Python**

  https://docs.blender.org/api/blender_python_api_2_60_1/bge.logic.html

- **Página no oficial de manuales y ayudas referente a las librerías de Blender para Python.**

  http://bgepython.tutorialsforblender3d.com/