Universitat Oberta de Catalunya

uoc.edu

Corredor de Laberintos

Jonathan Naranjo Vargas
Ingeniería Informática
Videojuegos

Profesora / Consultora: Ester Arroyo Garriguez
Profesor: Javier Luis Cánovas Izquierdo

06/2019

DEDICATORIA

A mi padre Milton, que siempre ha creído en mí y me apoyado incondicionalmente.

A mi hermano Darío, que siempre ha estado ahí alentándome para seguir adelante.

A mi cuñada Maribel, que a pesar de ser un total desconocido siempre se ha portado genial conmigo.

A mi primo Héctor, quien hace mucho tiempo atrás me dejó jugar en su PC por primera vez, él no sabe que en ese instante plantó una semilla de algo más grande.

A mi gato Sunny, que siempre me acompañado con su presencia en las largas horas de estudio.

AGRADECIMIENTOS

A todos los que me habéis apoyado y dado aliento en esta larga travesía.

Muchas gracias.

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Corredor de Laberintos*
Nombre del autor:	*Jonathan Naranjo Vargas*
Nombre del consultor/a:	*Nombre y dos apellidos*
Nombre del PRA:	*Nombre y dos apellidos*
Fecha de entrega (mm/aaaa):	06/2019
Titulación:	*Ingeniería Informática*
Área del Trabajo Final:	*Videojuegos*
Idioma del trabajo:	*Español*
Palabras clave	*Plataforma, aventura, habilidad*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

El objetivo de este proyecto es el desarrollo de un videojuego como parte del trabajo de fin de grado de los estudios de informática. Para ello, utilizaré todos los conocimientos adquiridos durante mi etapa académica y los enfocaré a una serie de herramientas que me ayudarán a alcanzar este objetivo.

El título del videojuego es “Corredor de Laberintos”, es un juego de aventura/acción con esencia de los clásicos de plataformas. Sus mecánicas y controles son bastante simples, la diversión radica en sortear las diversas dificultades que existen en los mapas. De modo que, sólo los jugadores más habilidosos podrán completar cada nivel.

Inicialmente, el mayor reto que me propuse era desarrollarlo sin que me supiera ningún costo económico, es decir, utilizando herramientas y recursos gratuitos de Internet. Esta idea implica utilizar software con funcionalidades más modestas y que generalmente son mantenidos por comunidades afines a nuestro objetivo, como lo es MonoGame y Nez Engine que utilizaremos principalmente. Por otro lado, la metodología aplicada ha sido: idear, planificar, ejecutar y corregir.

Para concluir, siento una gran satisfacción personal por este trabajo realizado y por los conocimientos y experiencia que me aportado. En este proyecto he podido plasmar muchas de las ideas que siempre he querido realizar, pero el tiempo es limitado y algunas de ellas han quedado en el tintero, estoy convencido que este videojuego se puede mejorar.

Abstract (in English, 250 words or less):

The objective of this project is the development of a video game as part of the end-of-degree work of computer studies. To do this, I will use all the knowledge acquired during my academic stage and focus on a series of tools that will help me achieve this goal.

The title of the videogame is "Labyrinth Corridor", it is an adventure/action game with essence of the classic platforms. Its mechanics and controls are quite simple, the fun lies in overcoming the various difficulties that exist in the maps. So, only the most skillful players will be able to complete each level.

Initially, the biggest challenge I set myself was to develop it at no economic cost, that is, using free Internet tools and resources. This idea implies using software with more modest functionalities and that are generally maintained by communities related to our objective, such as MonoGame and Nez Engine that we will use mainly. On the other hand, the methodology applied has been: devising, planning, executing and correcting.

To conclude, I feel a great personal satisfaction for this work done and for the knowledge and experience that I brought. In this project I have been able to capture many of the ideas that I have always wanted to do, but time is limited and some of them have been left in the inkwell, I am convinced that this video game can be improved.

Índice

Lista de figuras

1. Introducción

1.1 Contexto y justificación del Trabajo

Hoy en día el desarrollo de videojuegos se ha convertido en un arte multidisciplinar que implica dominar diversos campos de la ingeniería, gráficas, sonido, etc. Asimismo, los profesionales involucrados necesitan aplicar un enfoque creativo y sincronizarse para confeccionar todas las pizas necesarias que darán vida al juego.

Por otro lado, esta industria del entretenimiento ha ido creciendo imparablemente con el paso de los años, y se ha convertido en un gigante que incluso ha superado al cine en ingresos. Así pues, para suplir la demanda del mercado se han creado muchos estudios y empresas alrededor todo el mundo, cada día se ofertan nuevos puestos de trabajo a profesionales cualificados.

En definitiva, el propósito con este proyecto es incursionar y experimentar de primera mano todo lo que con lleva realizar un videojuego. Además, dejar documentado todo este proceso y que sea de código libre y disponible en Internet. Esto ayudará a otros académicos a su desarrollo profesional en este ámbito o simplemente contribuir a esta noble afición.

1.2 Objetivos del Trabajo

Al iniciar este proyecto necesitamos establecer unos objetivos, los cuales voy a dividirlos en primarios y secundarios.

- **Objetivo principal**

  Desarrollar un videojuego 2D desde la idea inicial hasta su versión jugable en la plataforma PC Windows.

- **Objetivos secundarios**

  - Guion y estructura del juego.
  - Gestión del proyecto.
  - Diseño artístico de personajes, objetos y escenarios.
  - Diseño de sonido.
  - Programación del juego.
  - Aprendizaje de MonoGame y Nez Engine.
  - Documentar las fases de desarrollo.

1.3 Enfoque y método seguido

Por lo general, la creación de un videojuego suele ser el trabajo de un equipo, donde cada uno de sus miembros tiene un perfil definido, ya sea técnico, artístico

o burocrático. Con esto quiero dar a entender que este proyecto ha sido desarrollado por una sola persona, donde he tenido que desempeñar diversos roles para lograr que lograr que el juego final tenga cierta calidad y sea un producto estable.

Por otro lado, en lo personal siempre he sido un apasionado por los videojuegos, me divierto más desarrollándolos y me satisface enormemente que otros jueguen mis creaciones. Sin embargo, hasta hace un par de años atrás mis limitados conocimientos me lo impedían. En aquel entonces esto fue un punto de inflexión para mí, ya que gracias a esta limitación me animé a estudiar ingeniería, y aquí estoy.

Ahora bien, la idea del videojuego se concibe gracias a mis gustos por las películas de Indiana Jones y por los juegos de plataformas. Es más, he tomado características de cada uno de ellos y lo he transformado en lo que ahora conocemos como el **Corredor de Laberintos**.

También, he querido que el proceso de desarrollo sea lo más parecido a como se hacía antaño, es decir, que tenga ese toque artesanal. Para lograr esto he tenido que evitar el uso de motores consolidados en el mercado, enfocándome en librerías más modestas, pero sin tener que programar cosas a muy bajo nivel, ya que repercutiría en el tiempo de planificación.

1.4 Planificación del Trabajo

La planificación de objetivos de este proyecto se ha realizado en un diagrama de Gantt. En él se ha especificado los diferentes hitos y fechas junto con la planificación para cada uno de estos. Además, en él se muestra los objetivos alcanzados y los que no se han podido ejecutar por problemas o replanteamientos surgidos a lo largo del desarrollo.

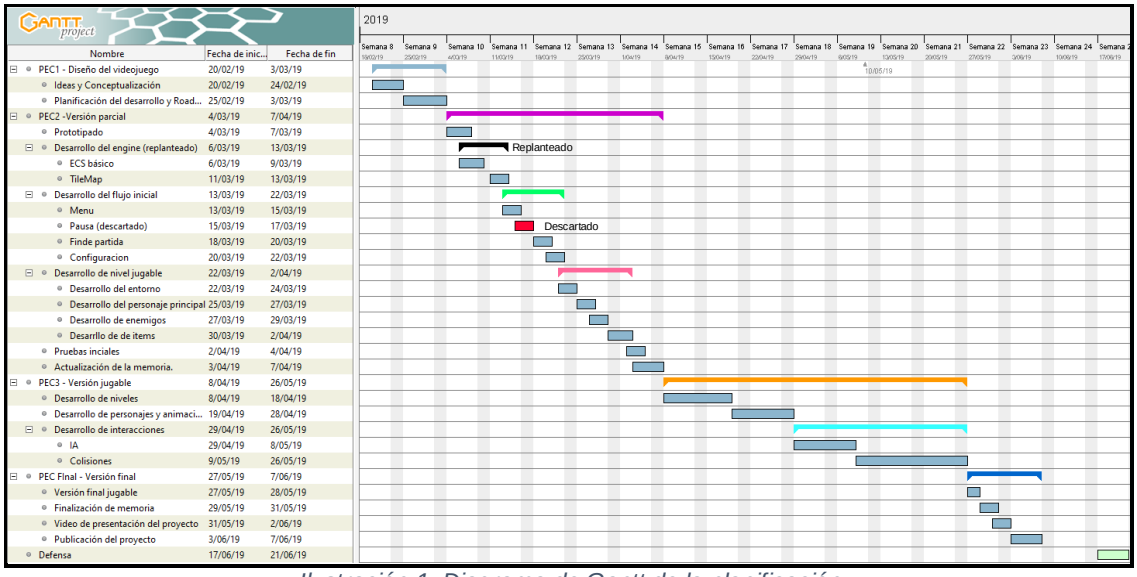

Ilustración 1. Diagrama de Gantt de la planificación

1.5 Breve sumario de productos obtenidos

A lo largo de este cuatrimestre he trabajado intensamente por alcanzar los objetivos marcados en la planificación. Finalmente, llegados a este punto se desglosan los productos más importantes obtenidos:

- Versión jugable (beta) para sistemas operativos Windows PC.
- Memoria final de trabajo
- Informe de autoevaluación
- Video explicativo
- Código fuente del proyecto.

1.6 Breve descripción de los otros capítulos de la memoria

- **Capítulo 2**: Se establece el tipo de género y evolución, prueba de tecnologías del mercado para el desarrollar, valoración económica y plataforma de distribución.
- **Capítulo 3**: Descripción de la historia y ambientación, referencia a otros títulos similares, objetivos del jugador y el arte conceptual de los actores.
- **Capítulo 4:** Explicación del juego desde un enfoque técnico, así como su justificación del uso de determinadas herramientas y sus requisitos. También, se describe los recursos gráficos y sonoros empleados.
- **Capítulo 5:** Guía de creación del nivel jugable desde el boceto hasta materializarlo con azulejos.
- **Capítulo 6:** Manual de usuario con instrucciones.
- **Capítulo 7:** Conclusiones generales del proyecto.
- **Capítulo 8:** Glosario con términos utilizados.
- **Capítulo 9:** Bibliografía, Webgrafía y otras referencias.
- **Capítulo 10:** Anexos con los requerimientos, instrucciones y repositorio del código fuente del videojuego.

2. Estado del arte

2.1 Género del juego

> Los **videojuegos de plataformas**, o simplemente **plataformas**, son un género de videojuegos que se caracterizan por tener que caminar, correr, saltar o escalar sobre una serie de plataformas y acantilados, con enemigos, mientras se recogen objetos para poder completar el juego. Este tipo de videojuegos suelen usar vistas de desplazamiento horizontal hacia la izquierda o hacia la derecha. Es un género muy popular de videojuegos, surgido a comienzo de la década de 1980 y que sigue manteniendo bastante popularidad en la actualidad.
>
> *Wikipedia*

El género de este videojuego es de tipo plataformas en dos dimensiones, con sus típicos tiles para los mapas y una cámara frontal que sigue al jugador. Actualmente, este estilo de juego es bastante común en el mercado y entre los desarrolladores INDIE (*Independent video game development*), supongo que su popularidad es porque es más sencillo de implementar, lo cual ayuda enormemente a reducir el tiempo de desarrollo. Con esto no quiero dar a entender que sea malo, al contrario, en mi opinión, es un estilo consolidado en este sector, y si se ejecuta correctamente una idea original puede llegar a ser muy divertido.

Por otro lado, en este proyecto no se ha inventado nuevas mecánicas, se ha tomado como referencia algunos videojuegos clásicos y actuales, eso sí, dándole un toque de originalidad y que pueda destacarse.

2.1.1 Evolución del Referente

- **Super Mario Bros Classic**

Ilustración 2. Mario Bross Classic

Desarrollado de por la empresa Nintento, es un juego donde Mario y Luigi empiezan una aventura en búsqueda de la princesa Peach secuestrada por Bowser. Hay poco que decir de este juego, es un gran conocido en el mundo de los videojuegos, sentó las bases de los juegos de plataformas.

- **Spelunky**

Ilustración 3. Spelunky

Es un videojuego independiente de acción y aventura creado por Derek Yu. El héroe es un buscador de tesoros que explora una serie de cavernas coleccionando tesoros, salvando damiselas y evadiendo trampas. Este juego tiene la peculiaridad de que cada vez que entras en la mazmorra esta se genera aleatoriamente.

- **Celeste**

Ilustración 4. Celeste

En este juego encarnas a Madeleine la cual llega a la falda de una montaña y tiene que llegar a su cima, es difícil, a medida que vas avanzando y muriendo cada vez más, lo cual no importa porque vas cogiendo la destreza suficiente hasta saber manejar tus fallos. Mecánicamente es sencillo con una curva de dificultad clara y precisa, buena banda sonora y un trasfondo precioso que trata enfermedades mentales y los ataques de ansiedad que sufre la protagonista.

2.2 Plataformas utilizadas para el desarrollo

Hace diez años atrás cuando empezaba a interesarme el desarrollo de videojuegos no existía apenas motores dedicados, todo el proceso se desarrollaba de forma muy artesanal y casi siempre había que primero programar el motor base y luego el videojuego. Por lo que aumentaba enormemente el tiempo de desarrollo y la inversión económica.

Hoy en día, esto ya no es así, en el mercado existe una gran variedad de herramientas que facilitan todo el proceso, los costos son más bajos y el tiempo mucho menor. Podemos encontrar *game engines* que se han convertido en estándar de facto entre las compañías desarrolladoras y estudios indie. A continuación, se menciona algunos de los más importantes que encajarían para este proyecto.

2.2.1 GameMaker Studio 2

Ilustración 5. GameMaker

Es un motor comercial enfocado especialmente en el desarrollo de videojuegos en 2D. Tiene su propio lenguaje de script llamado GML (GameMaker Language), su tiene una sintaxis bastante sencilla. También, tiene un editor muy versátil, este usa un sistema de arrastrar y soltar (drag and drop). En general, para tareas básicas se puede usar los eventos prefabricados que ofrece el motor, pero si quiere cosas más complejas y a medida, se puede programar scripts personalizados.

2.2.2 Unity3D

Ilustración 6. Unity 3D Engine

Es un motor gráfico comercial orientado para el desarrollo de videojuegos tanto 2D como 3D. Los desarrollos 2D se renderizan en un entorno 3D, esto da mucho juego porque permite usar todas las posibilidades 3d y plancharlo para que tenga una profundidad de dos dimensiones. Además, Unity permite crear videojuegos con su versión personal gratuita e incluso monetizarlos. Asimismo, utiliza un lenguaje de programación de alto nivel como es C#.

En general, es uno de los motores más completos del mercado, y es muy fácil de usar gracias a su editor, su curva de aprendizaje no es tan empinada, y lo mejor de todo, permite publicar el videojuego para casi cualquier plataforma de entretenimiento del mercado.

2.2.3 MonoGame

Ilustración 7. MonoGame

Es un C# framework usado para el desarrollo de videojuegos para múltiples plataformas, tales como: iOS, Android, MacOS, Linux, PlayStation 4, PlayStation Vita, Xbox One and Nintendo Switch. Está basado en el desaparecido Microsoft XNA, se podría decir que es una implementación libre y de código abierto. Sin embargo, aunque éste framework tiene clases de alto nivel para el manejo de pipeline, a nivel de desarrollo de videojuegos es bastante básico, no tiene editor, ni patrones de alto nivel como puede ser ECS (Entity Componente System) muy necesario hoy en día para el desarrollo moderno de videojuegos.

No obstante, en mi opinión, para cuestiones de aprendizaje considero que es muy versátil, te ofrece una estructura de programación básica y con ella puedes construir grandes cosas. Podemos ver los resultados en juegos como Bastión, Fez o Celeste.

2.2.4 Nez Engine

Ilustración 8. Nez Engine

Nez Engine es una capa de abstracción de alto nivel que se asienta sobre el framework de MonoGame. Además, este motor 2D tiene características modernas que facilitan el desarrollo de videojuegos en menor tiempo. Algunas de estas implementaciones son:

- Patrón Entity Componente System (ECS).
- TileMap.
- Detector de colisiones.
- Componentes prefabricados.
- Manejo de sprites.
- Soporte de sistema de partículas.
- Manejo de camaras.
- SceneManager
- etc.

2.3 Valoración económica del trabajo

Si bien es cierto que el desarrollo de videojuegos es multidisciplinar, esto supone tener conocimientos de: programación, diseño gráfico, diseño de sonido, etc. Por ello, llevo mucho tiempo utilizando herramientas habituales de estas disciplinas y he adquirido ciertas destrezas que me permitirán llevar a cabo el desarrollo en solitario.

2.3.1 Software

Para este proyecto me he propuesto utilizar herramientas gratuitas de Internet. Por tanto, en lo que respecta a software no nos va a suponer gastos económicos, ni regalías a largo plazo.

Coste de Software: 0,00 €

2.3.2 Hardware

El equipo utilizado en este desarrollo es mi actual portátil, lo he usado en todos estos años de carrera. Además, las características de su hardware cumplen de sobra con los requerimientos de las herramientas que vamos a usar. Su precio actualmente ronda los 900 €.

Coste de Hardware: 900 €

2.3.3 Recursos Humanos

- **Diseño de Sprites**

  En principio, no se va a necesitar crear los gráficos desde cero, voy a reutilizare recursos 2D gratuitos de páginas de Internet:

Itch.io	https://itch.io/game-assets/free
OpenGameArt	https://opengameart.org/

Gastos de diseño sprites: 0,00 €

- **Diseño de Sonidos**

  Igualmente, reutilizare recursos de sonidos gratuitos de estas estas páginas:

FreeSound	https://freesound.org/
OpenGameArt	https://opengameart.org/

Gastos de diseño de sonidos: 0,00 €

- **Programación**

  Toda la codificación del videojuego correrá a cargo de mi persona. Por ello, en este campo si me siento en capacidad de valorar mi trabajo como programador. Así pues, calculo que el desarrollo me ha tomado más o menos unas 160 horas y el costo por hora sería unos 20€.

  Gastos de programación: 160 x 20 € = 3200 €

2.3.4 Gastos totales

Con las cifras antes calculadas podemos obtener el coste total que llevaría hacer este videojuego con una sola persona:

Coste de Software	0,00 €
Coste de Hardware	900 €
Coste de Recursos Humanos	3200 €
Total	4100 €

2.4 Plataforma de destino

En principio, la plataforma de destino estaba pensada para sistemas Windows y Android. Sin embargo, mi desconocimiento sobre desarrollo en sistemas móviles podría no encajar con las mecánicas y experiencia que se intenta llevar al usuario final. Además, programar el videojuego para varias plataformas incrementa tiempo y la complejidad del proyecto.

Finalmente, la distribución de los binarios sólo se realizará para PC Windows, por ser la más versátil para nuestro objetivo y que se adapta mejor a los recursos que disponemos. De hecho, este sistema tiene el mayor porcentaje de usuario en el mundo.

3. Definición del juego

3.1 Historia y ambientación

La historia del juego nos lleva a encarnar a Tony, un audaz aventurero que se dedica a recorrer laberintos y mazmorras, al más puro estilo de Indiana Jones.
Él deberá atravesar peligrosos mundos llenos de trampas y poblados por diversos tipos de enemigos, los cuales impedirán que avancemos.

Para ayudarnos en nuestro viaje tendremos un amplio surtido de habilidades que iremos adquiriendo en el transcurso del videojuego. Asimismo, los niveles están repletos de tesoros que se pueden ir recogiendo por el camino y ayudar a la economía del héroe.

3.2 Objetivos planteados al jugador

El jugador al entrar en cada nivel tiene como objetivo principal conquistar el laberinto, para ello deberá ir sorteando obstáculo y enemigos hasta encontrar la salida. Además, en el mapa existen objetos valiosos como las monedas y diamantes, si se recogen todos, entonces obtendrá una puntuación máxima.

3.3 Definición de los personajes y elementos principales

- **Jugador principal**

  El jugador no posee ningún tipo de poder o arma, se le ha dotado de habilidades físicas que le permite desplazarse por lugares de difícil acceso en el mapa y eliminar enemigos.

- **Enemigos**

  Actualmente existe tres tipos de enemigos:

  - Bug1: Insecto con caparazón.
  - Bug2: Insecto con caparazón espinado.
  - Mine: Mina de contacto, explota al ser tocada.

  Ciertamente, cada tipo de enemigo tiene una calidad distintiva, pero todos tiene el mismo objetivo, evitar que el jugador avance.

- **Objetos recogibles**

Lo objetos recogibles o ítems están colocados en varias partes del mapa. Actualmente, hay dos tipos: monedas y diamantes. Asimismo, recoger estos objetos incrementará la puntuación del jugador.

- **Objetos no recogibles**

  Estas entidades son básicamente objetos estáticos que están ubicados en el mapa Actualmente tenemos la puerta de salida o exit, cuya utilidad es la de cambiar nivel o finalizar el juego.

3.4 Arte conceptual

A continuación, mostramos los artes conceptuales del nivel y los actores que participarán el juego.

3.4.1 Primer nivel

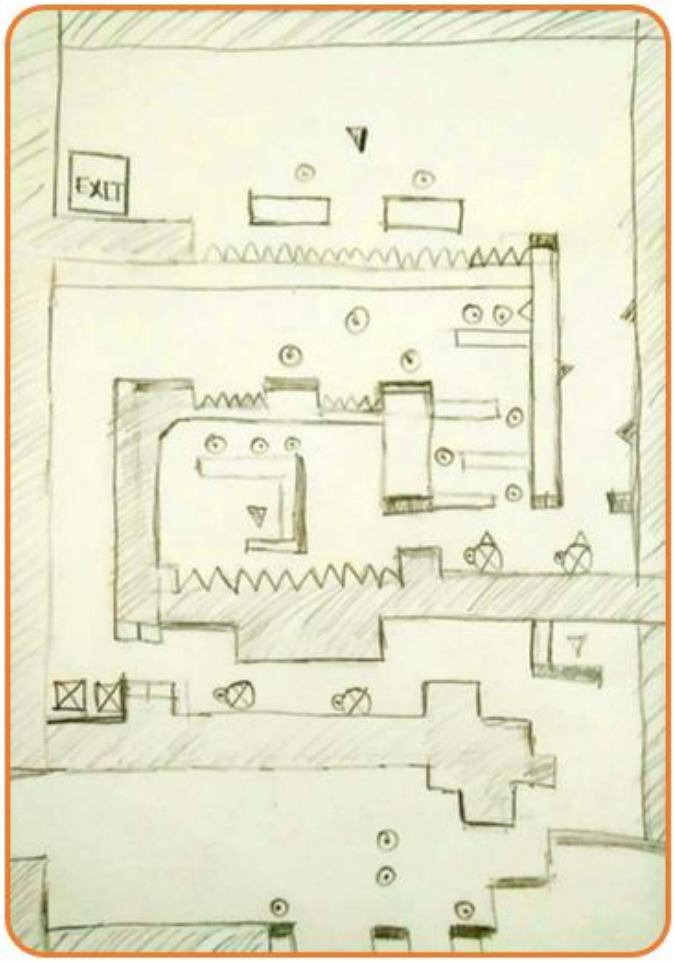

Ilustración 9. Boceto del primer nivel

3.4.2 Jugador principal

Ilustración 10. Miniatura del jugador principal

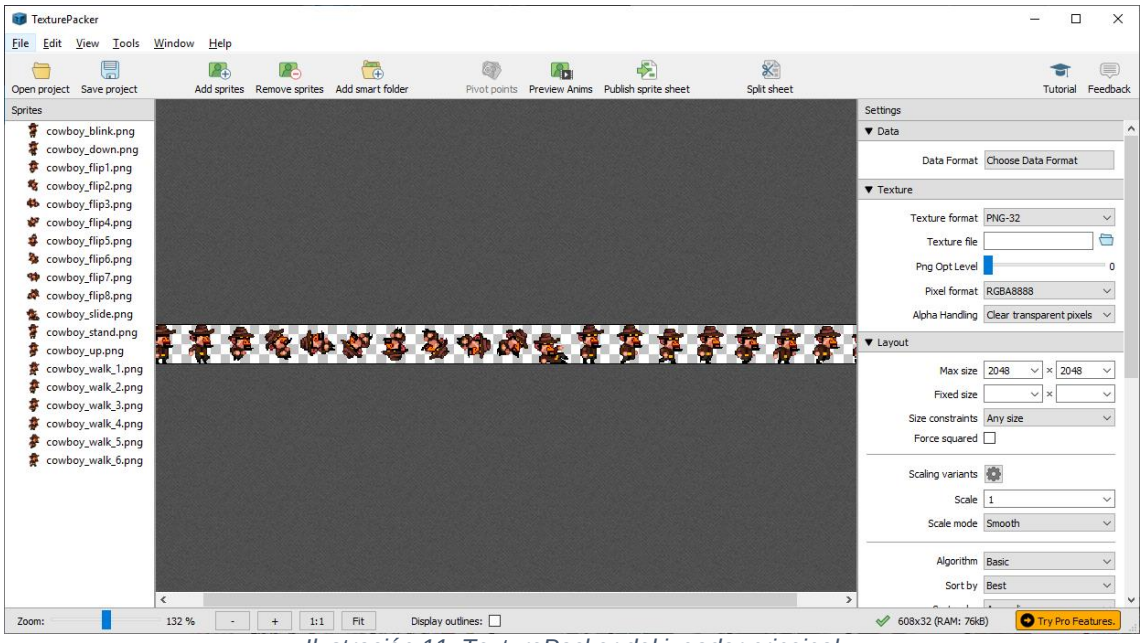

Ilustración 11. TexturePacker del jugador principal

3.4.3 Enemigos

Ilustración 12. Miniatura del enemigo Bug1

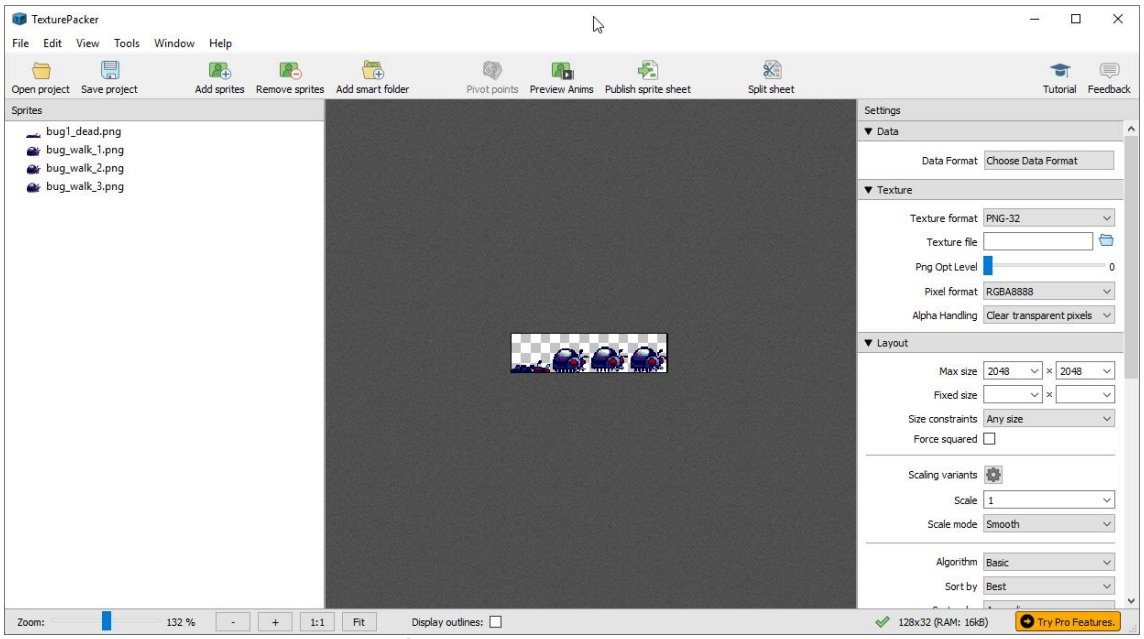

Ilustración 13. TexturePacker del enemigo Bug1

Ilustración 14. Miniatura del enemigo Bug2

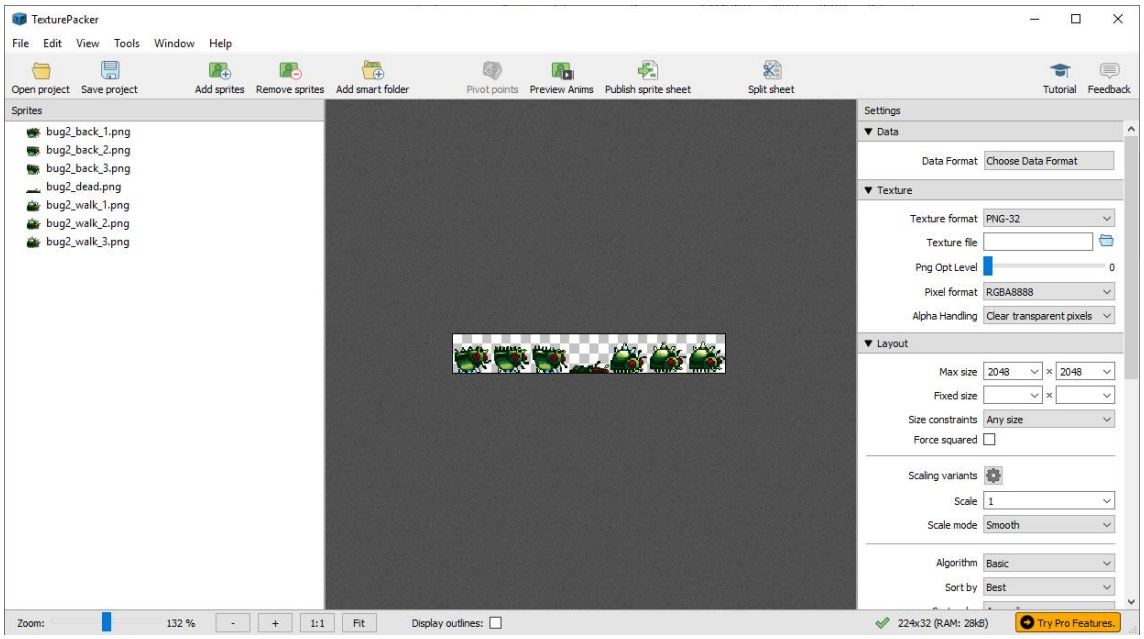

Ilustración 15. TexturePacker del enemigo Bug2

Ilustración 16. Miniatura del enemigo Mine

3.4.4 Objetos recogibles

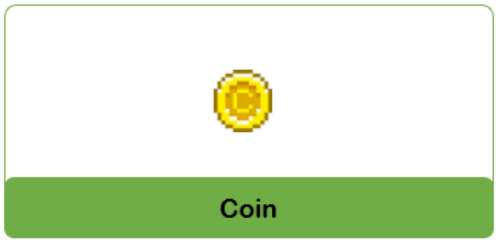

Ilustración 17. Miniatura del ítem Coin

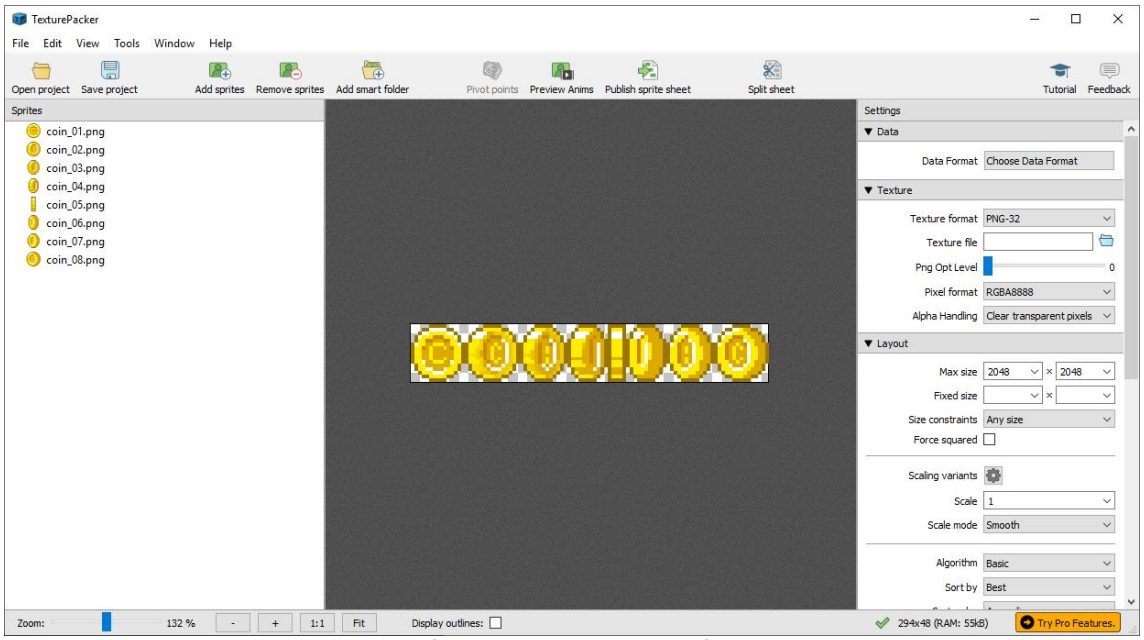

Ilustración 18. TexturePacker del ítem Coin

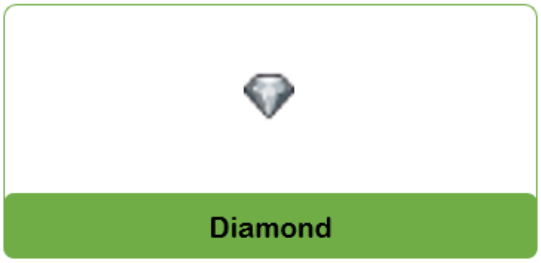

Ilustración 19. Miniatura del ítem Diamond

3.4.5 Objetos no recogibles

Ilustración 20. Miniatura del objeto no recogible Exit

4. Diseño técnico

4.1 Entorno escogido y requerimientos

Ilustración 21. Monogame + Nez Engine

En este desarrollo se utilizará **MonoGame** como framework base que se encargará de inicializar la aceleración gráfica intercambiando entre las APIs (Aplication Programing Interface) Directx o OpenGL dependiendo de la plataforma de destino. Además, se integrará Nez Engine como una capa de alto nivel que se asentará encima de MonoGame, esta proporciona características modernas para el desarrollo de videojuegos.

Por otro lado, la razón principal del por qué elegido MonoGame y Nez Engine y no otros motores consolidados del mercado, es porque me ofrece un conjunto de implementaciones básicas sin llegar a ser un producto completo. Esto me da margen a usarlo con otras herramientas y me da mayor control del código del juego. Asimismo, estos motores al ser gratuitos y de código abierto me permite leer sus implementaciones internas y aprender de ello.

4.2 Requerimientos del desarrollo

Mínimos:

- **Sistema operativo:** Windows 7 o posterior
- **Procesador:** Intel Core i3
- **Memoria:** 4 GB de RAM
- **Gráficos:** Intel HD 4000
- **OpenGL:** Versión 3.1 o posterior
- **Net Framework:** 4.7.1
- **Almacenamiento:** 3 GB de espacio disponible

Recomendados:

- **Sistema operativo:** Windows 7 o posterior
- **Procesador:** Intel Core i5
- **Memoria:** 8 GB de RAM
- **Gráficos:** Intel HD 5100
- **OpenGL:** Versión 3.1 o posterior
- **Net Framework:** 4.7.1
- **Almacenamiento:** 3 GB de espacio disponible

4.3 Herramientas utilizadas

Visual Studio 2019

Es un entorno de programación integrado de Microsoft, soporta muchos lenguajes (VB, C/C++, C#). Para este proyecto usaré su versión gratuita llamada Community. Asimismo, la codificación del videojuego será en lenguaje C#.

Ilustración 22. Visual Studio 2019

Audacity

Es un programa para la edición de sonidos. Nos permitirá modificar y adaptar nuestras pistas de audio. Es gratuito.

Ilustración 23. Audacity

Ilustración 24. Paint.Net

Paint.Net

Programa de edición gráfica 2D que me ayudará a editar los sprites y tiles. Es decir, con él editaremos todos los recursos gráficos de nuestro juego. Es gratuito.

TexturePacker

Programa que nos facilita la creación de hoja de sprites (*Sprite sheets*). Dichos ficheros sprites son colocados en una hoja formateada por una rejilla y será exportado a un solo fichero imagen. Esta imagen será cargada en el videojuego y recortada en ejecución. Este método reduce considerablemente el consumo de memoria. El programa tiene una versión gratuita no comercial.

Ilustración 25. TexturePacker

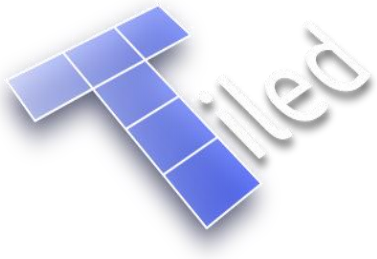

Ilustración 26. Tiled

TileD

Es un editor de niveles 2D especializado en tilemaps. Asimismo, este programa nos permitirá diseñar nuestros niveles y exportarlo a un formato XML fácilmente cargable desde nuestro videojuego. Es gratuito.

Bitmap Font Generator

Este programa te permite generar fuentes de mapa de bits a partir de fuentes TrueType. La aplicación genera archivos de imagen y descripciones de caracteres que un juego puede leer para una fácil representación de las fuentes. Es gratuito.

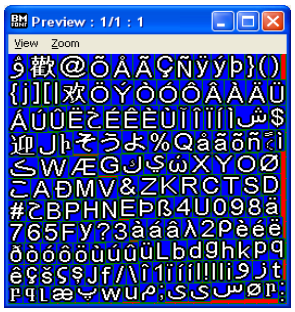

Ilustración 27. Bitmap Font Generator

Gravit Designer

Es una aplicación de diseño de gráficos vectoriales. Además, permite insertar gráficos rasterizados, esto es muy útil para poder crear las pantallas del juego. Tiene una versión gratuita.

Ilustración 28. Gravit Designer

4.3 Recursos del juego

Todos los recursos tanto gráficos como de sonidos usados en “Corredor de Laberintos” han sido obtenidos de páginas de distribución recursos gratuitos mantenidos por la comunidad, estas páginas son:

- OpenGameArt.org
- itch.io
- freesound.org

4.3.1 Sprites

Se ha utilizado más de 40 sprites diferentes para este proyecto. Aquí se incluyen personajes, ítems, objetos del entorno y efectos:

Ilustración 29. Hojas de sprites de los actores

También se incluye pantallas personalizadas para las opciones del menú:

configurar.png credits.png end_game.png menu_background.png

Ilustración 30. Pantallas del juego

4.3.2 Sonidos

Se ha utilizado tanto efectos de sonido como piezas musicales para la banda sonora de cada nivel. Estas pistas son:

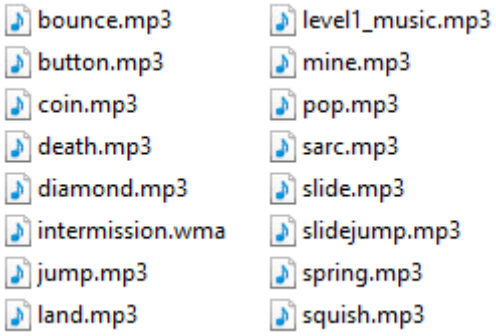

Ilustración 31. Sonidos del juego

4.3.3 Fuentes

Para representar texto en pantalla es necesario crear una imagen con los caracteres representados. La herramienta Bitmap Font generator nos facilita este proceso, convierte una fuente tradicional en una imagen de caracteres:

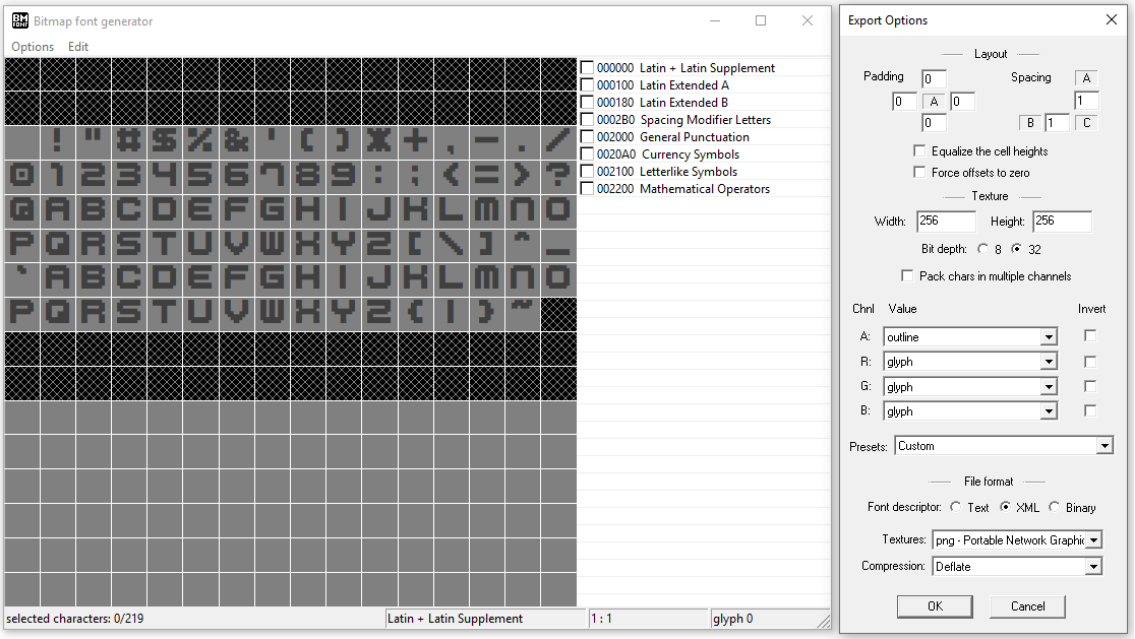

Ilustración 32. Creación de bitmap

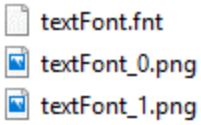

Ilustración 33. Bitmap del juego

4.4 Arquitectura del juego y diseño

A lo largo de este capítulo se detallará los procesos de diseño de la arquitectura del juego, empezando por las animaciones de los personajes, hasta su implementación. De la misma forma, el diseño de mapas es crucial en este proyecto por lo que tiene reservado un capítulo propio y, en él, se detallarán exclusivamente las decisiones y tipos de recursos usados.

Por otra parte, tomando en cuenta que el objetivo de esta memoria no es la de ofrecer un código comentado del proyecto, si no, la de ofrecer el punto de vista del desarrollador respecto a las decisiones ejecutadas.

4.4.1 Animación y diseño de personajes

En el primer nivel del videojuego existe siete entidades, algunos tienen animaciones y otros dadas sus características son estáticos. Para las animaciones he utilizado colección de sprites en una hoja situados en un solo carrete:

Ilustración 34. Animaciones del Tony

Ilustración 35. Animaciones de Bug1

Ilustración 36. Animaciones de Bug2

Ilustración 37. Animaciones de Coin

Ilustración 38. Animaciones de explosión de mina

El proceso de animación consiste en cargar la hoja de animaciones y recortar cada sprite. Estos recortes o frames podrán ser agrupados en una secuencia y presentados en pantalla uno a uno en un intervalo de tiempo, al ojo humano creará la sensación de una animación.

4.4.2 Lógica del personaje

Para poder realizar los cambios entre las diversas animaciones del jugador se ha diseñado una máquina de estados cuyas aristas sincroniza los movimientos en cada instante:

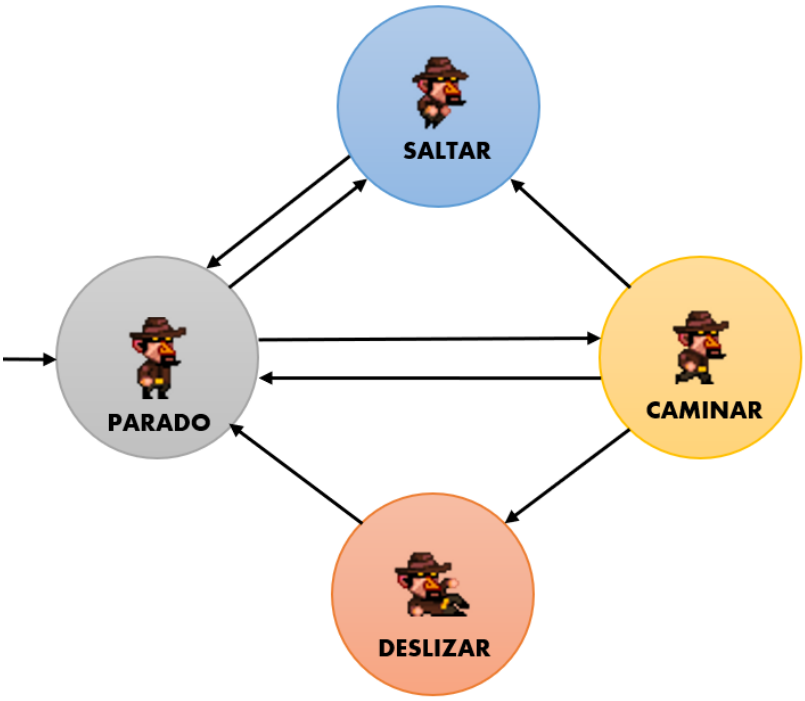

Ilustración 39. Máquina de estados del jugador

Las aristas responden a eventos que se disparan al presionar determinadas teclas de entrada. Estas acciones provocan los cambios de estados y su animación específica.

4.4.3 Objetos interactivos y colisiones

Ciertamente, la mayoría de los eventos se producen por señales de los periféricos de entrada. Sin embargo, hay otros tipos de acciones que sólo se disparan mediante la interacción con otros actores, es decir, sólo se activan mediante colisiones entre ellos, por ejemplo: cuando el jugador recoge una moneda, esta desaparece y aumenta su puntuación. Así pues, en este videojuego dichas colisiones se activan por intersección de rectángulos, si presionamos F12 podemos entrar en modo depuración y visualizar estos rectángulos:

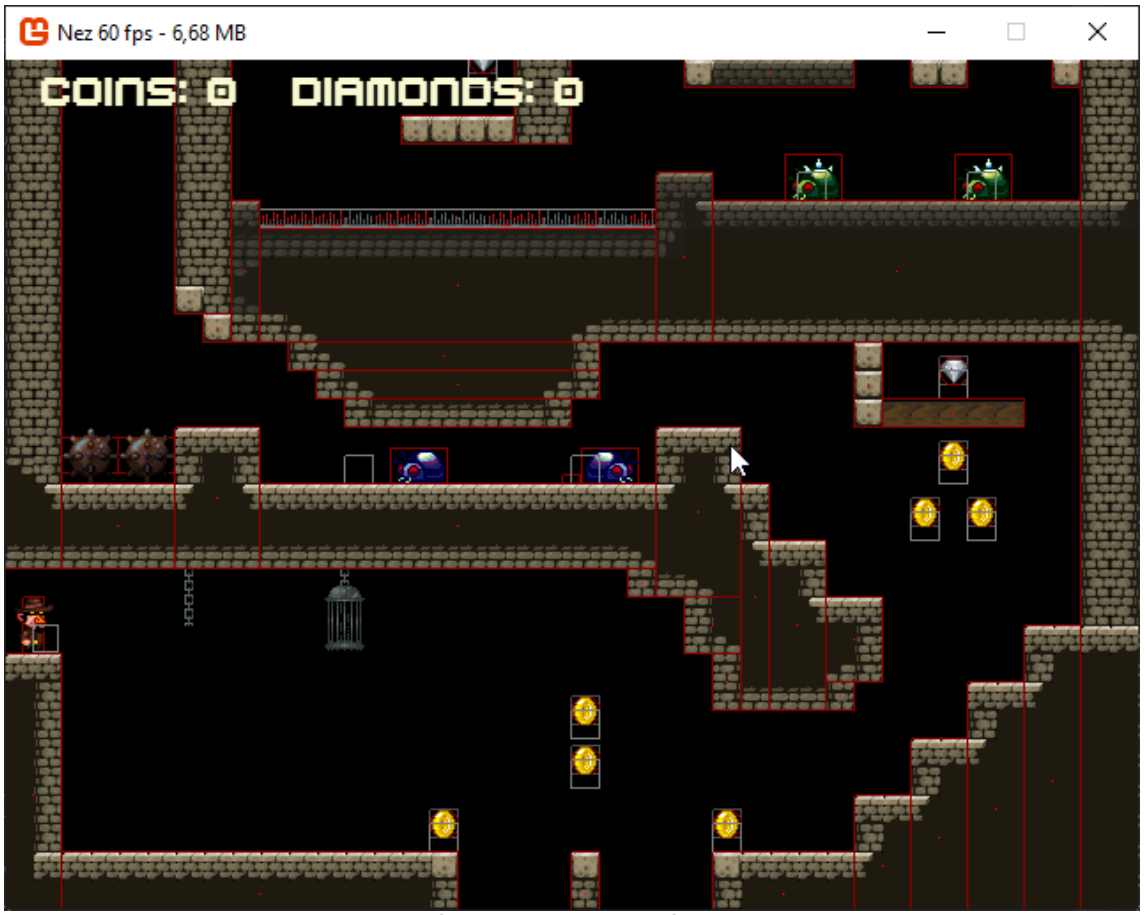

Ilustración 40. Modo depuración del juego

4.4.4 Enemigos IA

La mayoría de enemigo móviles tienen ciertas características diferenciables, pero, en general todos ellos están equipados con una “IA básica”. Esta consiste en el enemigo caminará hasta tocar un muro, seguidamente cambiará de sentido y repetirá el ciclo.

4.4.7 HUD

El HUD (head-up display) se ha intentado se lo más simple y descriptiva posible. En ella se informa al jugador sobre el número de monedas y diamantes recolectados. Se encuentra situada en la parte superior izquierda:

Ilustración 41. Hud del juego

5. Diseño de niveles

5.1 Proceso de diseño

Para la creación de niveles se ha utilizado programa **Tiled**, el cual nos proporciona un lienzo y herramientas que facilita el diseño de mapas con azulejos (tiles). Así pues, para empezar a crear el primer necesitamos definir sus dimensiones (posteriormente se puede volver ajustar) y el tamaño de las baldosas o patrones:

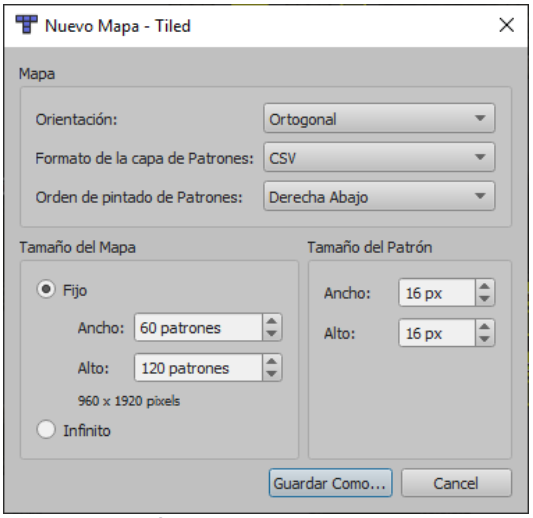

Ilustración 42. Nuevo mapa con Tiled

Asimismo, se ha creado tres capas bien nombradas (entidades, map, background) que nos ayudará a separar los diversos elementos que conforman del nivel. De hecho, considero necesario hacerlo, porque a la hora de cargar el mapa dinámicamente permite maniobrar entre una y otra capa, también, separar la parte artística o embellecedora de las partes colisionables:

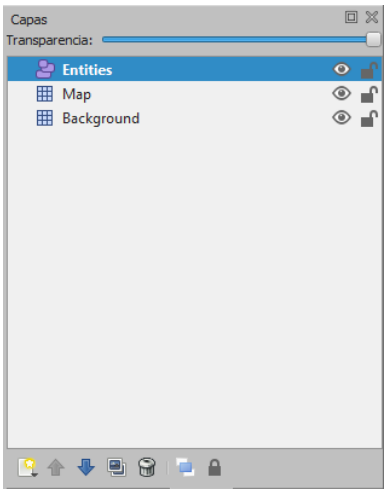

Ilustración 43. Definición de capas

5.1.2 Conjunto de azulejos

Necesitamos definir los patrones de las baldosas con los que se va a trabajar. Para ello, cargaremos tileset (hoja con sprites) y definimos las dimensiones de la rejilla, en este caso es de 16 x 16:

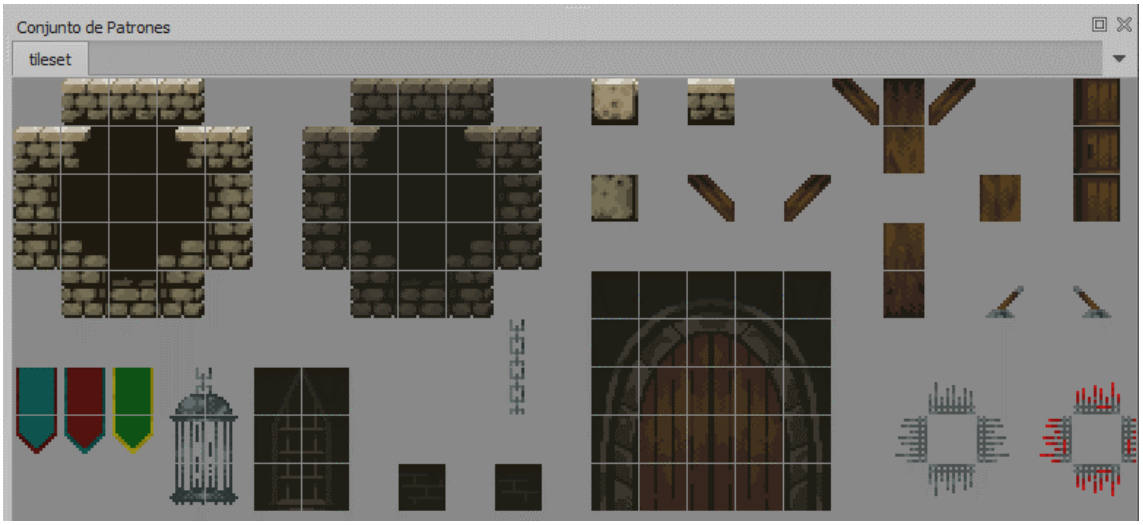

Ilustración 44. Conjunto de azulejos del juego

5.1.3 Diseño de mapas

Una vez todo configurado procedemos a diseñar el nivel en base al boceto que se realizó anteriormente (*ilustración 9*). Ahora bien, el diseño final puede cambiar un poco respecto al prototipo que se tenía inicialmente, y esto se debe a que el bosquejo del dibujo no tomó como referencia medidas reales. Así pues, el mapa ha quedado de la siguiente forma:

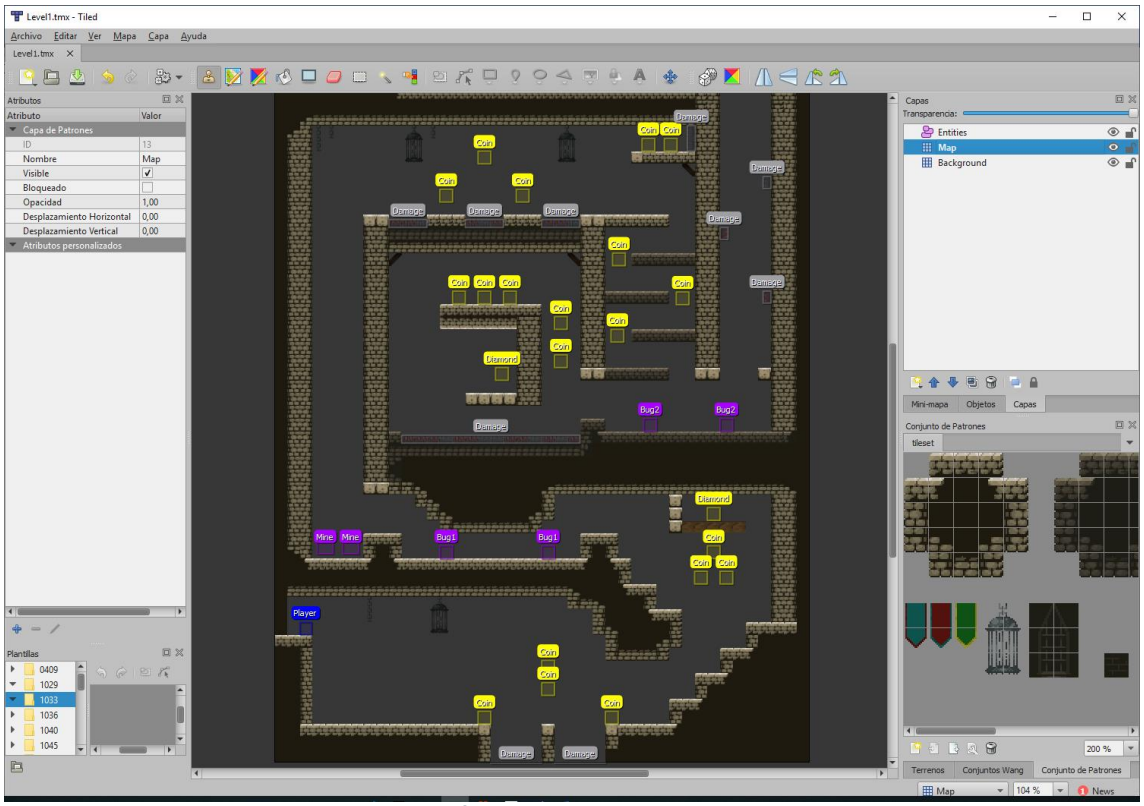

Ilustración 45. Diseño del primer nivel

6. Manual de usuario

6.1 Requisitos del juego

Requisitos mínimos:

- **Sistema operativo:** Windows 7 o posterior
- **Procesador:** Intel Core i3 M380
- **Memoria:** 2 GB de RAM
- **Gráficos:** Intel HD 4000
- **OpenGL:** Versión 3.1 o posterior
- **Net Framework:** 4.7.1
- **Almacenamiento:** 30 MB de espacio disponible

6.2 Instrucciones

La configuración de los controles podrá ser visualizada desde el menú principal:

Ilustración 46. Pantalla de configuración de controles

7. Conclusiones

7.1 Conclusiones sobre el trabajo final

Llegados a la parte final del proyecto considero que he a prendido muchas cosas durante todo el proceso de desarrollo. Me he enfrentado a situaciones de riesgo, donde el tiempo jugaba en mi contra, asimismo, he tenido que tomar decisiones complicadas para poder sortear las dificultades que ido encontrando por el camino.

Por otra parte, trabajar con motores no tradicionales me ha llevado a invertir más horas de investigación para saber el funcionamiento su API que en desarrollar el propio videojuego. Pero, a pesar de todo, he disfrutado este cuatrimestre como ningún otro, lo único que lamento es no haber tomado en cuenta el peso ciertos factores que han influido en los objetivos que me propuse inicialmente.

En conclusión, la experiencia que me aportado este proyecto me ha enriquecido tanto a nivel intelectual como profesional en este sector. Además, estoy muy satisfecho con el resultado final del videojuego, es tal y como lo imaginé en un principio. En definitiva, considero que el juego cumple con su cometido, ser vistoso y divertido.

7.2 Reflexión sobre objetivos planteados

Un par de objetivos que me propuse inicialmente fue programar mi propio patrón ECS (Entity Componente System) y una librería para la carga de Tilemaps. Sin embargo, fracasé estrepitosamente, no porque no lo pudiera hacer, sino porque necesitaba más tiempo para su implementación y realizar las pruebas necesarias que me aseguren su funcionamiento correcto. Así pues, tuve que decidir entre continuar implementándolos y reestructurar la planificación o simplemente desechar el código. Al final, opté por esto último (fue la mejor decisión), además, para solventar esa implementación perdida empecé a usar el motor Nez, cuyas características ya suplen esta necesidad.

Por otro lado, una vez superado el anterior problema nos encontramos con este otro. Quizá uno de los mayores inconvenientes del proyecto ha sido trabajar con Nez Engine. La carencia de una buena documentación y la ausencia de una comunidad dispuesta a resolver dudas ha hecho que determinados objetivos consumieran más tiempo de lo que la planificación me lo permitía. También, el motor actualmente está en temprano desarrollo y hay algunas características que están incompletas o simplemente no han sido implementadas. Ahora bien, gracias a mi experiencia con otros motores de similar arquitectura he podido ir solventando cada problema que me ha surgido.

7.3 Seguimiento de la planificación y metodologías

En principio, el 90% de los objetivos propuestos de la planificación inicial se han cumplido. Por el contrario, el otro 10% no se ha llegado a ejecutar por las siguientes causas:

- **Desarrollo del engine:** La propuesta consistía en crear librerías ECS y Tilemap sobre el framework MonoGame. Sin embargo, como ya he comentado antes, resultó inviable implementarlo desde cero por cuestiones de tiempo. La solución que se buscó fue utilizar características ya existentes del motor Nez. En otras palabras, se podría decir que se aplicó una solución rápida y efectiva para que no afectara a la planificación. (replanteado)

- **Pause:** La razón del por qué no se llegó a implementar esta característica en el videojuego, es por Nez. Este utiliza una máquina de estados donde cada nivel empieza por el estado inicial, luego pasa al estado de carga y seguidamente permanece en un bucle alternando entre los estados update y draw. Si se intenta intercambiar con otro nivel y luego volver al anterior el motor reinicializa el nivel empezando nuevamente desde estado inicial. En otros términos, la solución implicaría cambios en el propio motor, lo cual no me resultaba viable con el tiempo que disponía en la planificación. (descartado)

Por otro parte, la metodología de desarrollo se ha basado en cuatro pasos: idear, planificar, ejecutar y corregir. En este sentido, teniendo en cuenta la poca

experiencia con el motor elegido y fijándonos en el resultado final del videojuego, considero que el método seguido ha sido exitoso.

Para terminar, opino que la planificación ha seguido su curso adecuado, se adaptado a las circunstancias, se han tomado decisiones importantes que de alguna manera ha llevado a buen puerto el proyecto.

7.4 Líneas de trabajo futuro

En el **Corredor de Laberintos** se han implementado la gran mayoría de ideas que tenía en mente, pero, hay muchas otras que han quedado en el tintero, algunas de las más interesantes son:

- Una versión para móvil.
- Editor de niveles propio, para soporte de mods.
- Nuevas modalidades de juego:
    - Modo corredor.
    - Modo tiempo.
- Para hacer más dinámicos los mapas sería interesante agregar plataformas móviles.
- Nuevos niveles más variados.

En mi opinión, creo que este videojuego podría adaptarse perfectamente a sistemas móviles, la jugabilidad es muy compatible; incluso podría ser un gran éxito en la PlayStore de Google. En definitiva, al Corredor de Laberintos le queda mucho camino por delante, espero en este verano poder sacar una nueva versión con todas estas ideas.

8. Glosario

Indie (videojuegos indipendientes)

Es un movimiento de creación de videojuegos independiente de la industria tradicional y dominante. Se caracteriza fundamentalmente en la búsqueda de una expresión artística más que en un impacto en las ventas, es decir, es una oposición a la industria dominante.

ECS (Entity Component System)

Es un patrón arquitectónico que se utiliza principalmente en el desarrollo de juegos. ECS sigue el principio de composición sobre herencia que permite una mayor flexibilidad en la definición de entidades donde cada objeto en la escena de un juego es una entidad (por ejemplo, enemigos, balas, vehículos, etc.). Cada entidad consta de uno o más componentes que agregan comportamiento o funcionalidad. Por lo tanto, el comportamiento de una entidad se puede cambiar en tiempo de ejecución agregando o eliminando componentes. Esto elimina los problemas de ambigüedad de las jerarquías de herencia profundas y amplias que son difíciles de entender, mantener y extender.

Sprite

Imagen de mapa de bits gestionada por un hardware especializado, generalmente es usado para representaciones gráficas en videojuegos 2d.

IDE (Integrated Developement Enviroment)

Entorno de desarrollo integrado empleado para realizar programación.

Frame

Cada una de las imágenes que se generan en la unidad de tiempo del procesador. En otros ámbitos audiovisuales se emplea como sinónimo la palabra fotograma.

Tileset

Conjunto de imágenes agrupadas para ser usadas mediante una referencia única en el juego. Una referencia contiene las coordenadas para buscar el tile o baldosa dentro de un mismo conjunto.

HUD (Head-up display)

Información que en todo momento se muestra en pantalla durante la partida, generalmente en forma de íconos y números.

9. Bibliografía

MonoGame – Documentación
http://www.monogame.net/documentation

Nez – Wiki
https://github.com/prime31/Nez/wiki

Tonypa's tile-based tutorials
http://www.gotoandplay.it/_articles/2004/02/tonypa.php

RB Whitaker's Wiki
http://rbwhitaker.wikidot.com/xna-tutorials

GameFromScratch
https://www.gamefromscratch.com/page/MonoGame-Tutorial-Series.aspx

Game Programming Patterns
https://gameprogrammingpatterns.com/

Ejercicios de Física - Cinemática
https://ejerciciosdefisica.com/

10. Anexos

Repositorio del videojuego:

Ilustración 47. Pantalla principal

- GitHub [Código + Ejecutables]
  https://github.com/JonathanNaranjo/CorredorDeLaberintos