Universitat Oberta de Catalunya

Enfrente de los pisos

Autor: Esteban Herreros Suárez
Tutor: Joan Arnedo Moreno
Profesor: Helio Tejedor Navarro

Máster Universitario en Diseño y Programación de Videojuegos
Diseño de experiencias de juego

05/06/2022

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Enfrente de los pisos*
Nombre del autor:	*Esteban Herreros Suárez*
Nombre del colaborador/a docente:	*Joan Arnedo Moreno*
Nombre del PRA:	*Helio Tejedor Navarro*
Fecha de entrega (mm/aaaa):	*06/2022*
Titulación o programa:	*Máster universitario de Diseño y Programación de videojuegos*
Área del Trabajo Final:	*Trabajo final de Máster*
Idioma del trabajo:	*Español*
Palabras clave	*Unity, 2D, Aventura gráfica*

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

La finalidad de este proyecto es desarrollar un videojuego completo, partiendo de cero y creando todos sus elementos. En él, controlaremos un preso de una cárcel de mínima seguridad, cuyo objetivo será escapar. Para ello deberá urdir un plan de fuga y llevarlo a cabo utilizando lo que encuentre a su alrededor, además de hablar con el resto compañero y guardias para conseguir información.

El resultado será una aventura gráfica en 2D, con vista *top down* y estilo *píxel art*, donde es necesario averiguar cuándo, dónde y qué objetos hay que utilizar o combinar para alcanzar la libertad. Como en cualquier prisión, hay un horario muy estricto y, cada segundo en la vida real corresponde a un minuto en la partida, por lo que cada poco rato habrá un cambio de escenario, pasando por ejemplo de la celda al comedor a la hora del desayuno. Esto añade un comportamiento cíclico y aporta dinamismo, pues precisa del jugador una cierta destreza para realizar las tareas en el tiempo estipulado. Además, en algunos momentos será necesario esquivar a los guardias para no ser detectado mientras se consigue alguna meta, con lo que hay también algunos toques de sigilo.

Para el desarrollo se utilizará una metodología ágil, dividiendo el trabajo en pequeñas secciones y realizando sobre cada una de ellas las pruebas necesarias.

Abstract (in English, 250 words or less):

The purpose of this project is to develop a complete video game, starting from scratch and creating all its elements. In it, we will control a prisoner in a minimum security prison, whose objective will be to escape. To do this, he must devise an escape plan and carry it out using what he finds around him, as well as talking to the rest of his companion and guards to get information.

The result will be a 2D graphic adventure, with a top down view and pixel art style, where it is necessary to find out when, where and what objects must be used or combined to achieve freedom. As in any prison, there is a very strict timetable and, every second in real life corresponds to a minute in the game, so every so often there will be a change of scenery, passing for example from the cell to the dining room at lunch time. This adds a cyclical behavior and provides dynamism, since the player requires a certain dexterity to carry out the tasks in the stipulated time. Also, in some moments it will be necessary to avoid the guards to avoid being detected while achieving some goal, so there are also some stealth touches.

An agile methodology will be used for the development, dividing the work into small sections and carrying out the necessary tests on each of them.

Índice

Figuras y tablas

Índice de figuras

1. Introducción

1.1. Introducción

A lo largo del máster hemos visto la mayoría de las partes involucradas en la creación de un videojuego, desde la programación de la lógica hasta el diseño de escenarios y otros recursos. Esto nos ha dado un conocimiento más profundo sobre la dimensión que tiene realizar cualquier tipo de producción de este arte.

Personalmente, ser consciente de esta realidad hace que pueda valorar como se merece el trabajo de personas como Lucas Pope o Toby Fox. Ellos han sido capaces de crear grandes experiencias de juego, como Papers Please o Undertale, sin la colaboración de nadie en ningún área del proceso [1]. Echando un vistazo a las obras de estos desarrolladores, creo que hay un denominador común: la originalidad. Esto demuestra que el arte puede compensar las carencias técnicas.

Mi futuro en el mundo del desarrollo de videojuegos me gustaría que fuera, salvando las diferencias, como el de los ejemplos anteriormente mencionados. Poder crear, de forma independiente, historias y formas de jugar que realmente sean interesantes y entretenidas, buscando la satisfacción personal por encima de la obtención de beneficios económicos.

Por estos motivos he optado por crear una aventura gráfica, un género adecuado para cumplir con los objetivos marcados. El juego transcurre en una cárcel y el objetivo, como suele ser habitual, es conseguir escapar. Esta decisión viene motivada por la intención de introducir ciertos elementos. El primero de ellos es el sigilo. Con la idea de aportar algo de dinamismo a la jugabilidad, habrá secciones donde el jugador tenga que ir de un punto a otro sin ser detectado. El segundo es el uso de pocos escenarios que se repitan cíclicamente, algo muy apropiado para simplificar la creación de recursos y que encaja perfectamente con la vida en prisión. Por último, para aportar un elemento diferenciador dentro del género y reforzar el punto anterior, se incluye el tiempo. Hay un horario dentro de la cárcel que marca el tiempo que se está en cada sección.

1.2. Descripción

Como se ha introducido previamente, el juego se desarrollará en una cárcel. A lo largo del día el jugador recorrerá cuatro escenarios que irán cambiando en función del horario: la celda, el comedor, el patio y el taller de trabajo. En cada uno de ellos habrá diferentes objetos y personajes no jugables. Como en toda aventura gráfica, se deberán recolectar los ítems necesarios, usándolos para conseguir nuevos objetivos hasta llegar al final. Sin embargo, al haber cacheos cada vez que se va a la celda, habrá que pensar cómo conservar esos objetos y cómo hacerlos llegar de un sitio a otro, haciendo que el trabajo tenga que hacerse de una vez sin poder acumularlo. Además, la introducción de la mecánica del tiempo hará que sea necesario realizar las tareas de forma rápida.

La idea con todo esto es que el jugador vaya aprendiendo cada día que pasa en prisión y desarrolle un plan para que, cuando lo vaya a ejecutar, tenga la presión de no fallar, pues no puede guardar su progreso. Por ejemplo, cada día en la cena se utilizan cuchillos de plástico, el cual es necesario llevar al taller como parte del plan. Sin embargo, al taller se va por la mañana y, si se intenta llevar a la celda en el cacheo se le detecta y se le quita al jugador. Con lo cual, la solución sería esconderlo en la basura, que recogen siempre a medio día, y en el desayuno recuperarlo para llevarlo al taller. Es posible que sean necesarios varios intentos hasta dar con la tecla, con lo cual el usuario va adquiriendo conocimiento y cada vez desarrolla el plan más rápidamente.

Respecto al apartado gráfico, en aras de la sencillez y dado el tiempo del que se dispone para realizar el proyecto, el juego estará hecho en dos dimensiones y con una vista casi cenital, también conocida como *top down*. La estética será *píxel art*. Un ejemplo muy similar de todo esto sería cualquier versión de Pokémon para Game Boy. En pantalla se verá constantemente el inventario abierto, el juego en sí mismo y un reloj con el tiempo.

En cuanto a la jugabilidad, el movimiento del jugador se realizará mediante los controles direccionales. También, al posicionarse el jugador delante de un objeto con el que puede interactuar, aparecerá la opción que acerca de ellos (ver, usar, hablar, etc). Para utilizar los ítems del inventario sobre ellos simplemente habrá que tenerlos seleccionados previamente.

1.3. Objetivos generales

1.3.1. Objetivos principales

Objetivos del juego:

- Ofrecer una experiencia de juego completa.
- Utilizar recursos propios en todos los apartados del proyecto.

Objetivos para el cliente:

- Proporcionar un producto entretenido.
- Causar la necesidad de finalizar la partida, con una aventura equilibrada en su propuesta.

Objetivos personales del autor del TF:

- Aplicar los conocimientos adquiridos en el máster.
- Aprender a dibujar mediante la técnica *píxel art*.
- Adquirir experiencia en la creación de videojuegos.

1.4. Metodología y proceso de trabajo

El enfoque elegido para este proyecto es el de desarrollar un producto nuevo, comenzando desde cero todos sus elementos. Esto debe ser así por dos motivos, el principal es cumplir el objetivo de utilizar recursos propios en todo el proceso, el segundo es que intentar aprovechar assets ya existentes perjudicaría la armonía del juego al no estar toda la estética o la lógica conectadas.

Las metodologías de trabajo suelen estar pensadas para desarrollos en equipos de varios integrantes, pero se pueden adaptar a este caso obviando las partes de comunicación con el resto del grupo. Los métodos más conocidos hoy en día son los llamados ágiles, los cuales se caracterizan, como su nombre indica, por tener una gran capacidad para realizar cambios sobre la planificación. Dentro de estos existe una variante llamada Personal Extreme Programming (PXP) [2], que es la seleccionada para este proyecto.

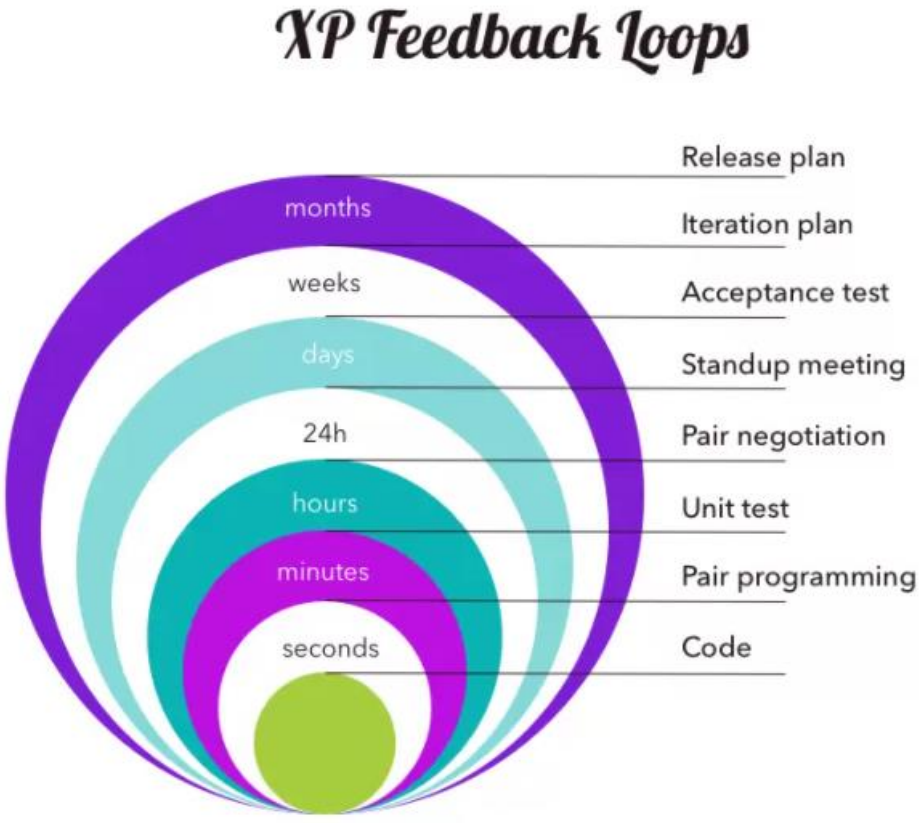


Figura 1: Metodología PXP

Una de las principales características de este tipo de métodos es la existencia de ciclos de tiempo donde se van revisando diferentes aspectos, los cuales permiten revelar rápidamente las necesidades de replanificación del proyecto.

Como luego se verá en la planificación, el trabajo está dividido en tareas pequeñas, cada una de las cuales coincidirá con un plan de entrega siguiendo con esta metodología.

En cuanto a los recursos necesarios para sacar adelante el proyecto, se pueden ver en la siguiente lista:

- Hardware:
    - Ordenador portátil.

- Software:
    - Unity3D: Framework para el desarrollo de videojuegos.
    - Photopea: Web de diseño gráfico.
    - Visual Studio Code: Editor de código.
    - Reaper: DAW para la realización de la música y sonidos.
    - GitLab: Repositorio con control de versiones del trabajo.
    - Office: Editor de texto para la redacción de la memoria.

1.5. Planificación

El tiempo total para la realización del proyecto son aproximadamente 15 semanas, las cuales se dividen en cuatro partes, cada una con su correspondiente entrega.

- **PEC 1 – Plan de proyecto:** 2 semanas. Esta primera parte consiste en desarrollar la idea, pero sin entrar en la implementación. El objetivo es definir claramente cómo va a ser el proyecto en todas sus áreas, desde la jugabilidad hasta los gráficos pasando por la historia. Además de definir los objetivos y la planificación.

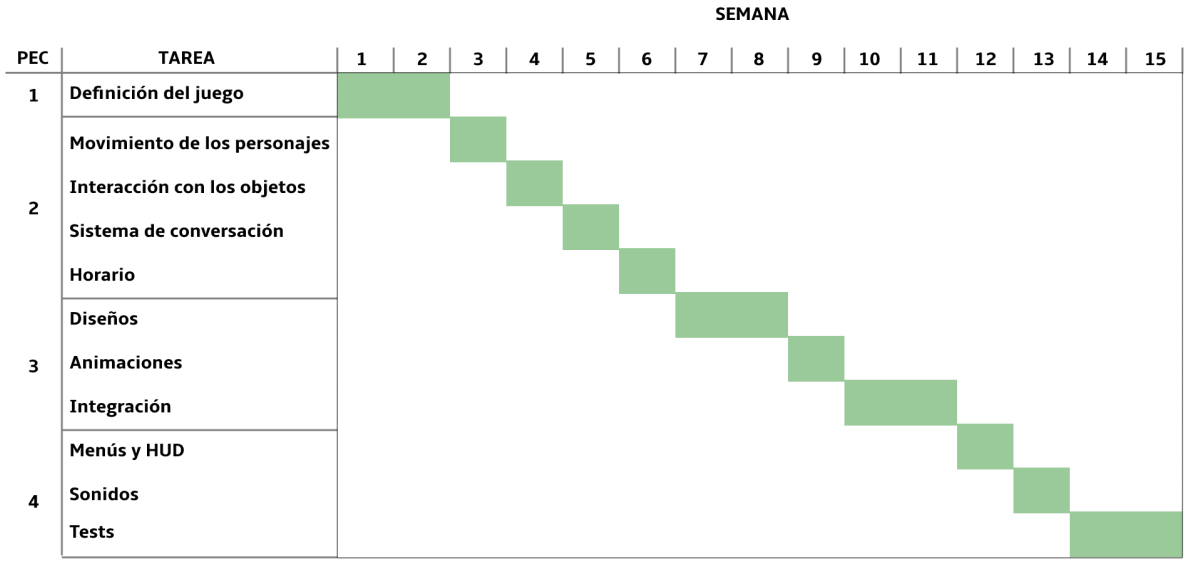
- **PEC 2 – Primera versión:** 4 semanas. Para esta entrega el objetivo es tener programadas las diferentes mecánicas que hay dentro del juego, en concreto el movimiento del protagonista, la detección por parte de los guardias, la interacción con los objetos y el escenario, un sistema de conversaciones y la implementación del horario. Como los esfuerzos se van a centrar en la programación, en el apartado gráfico se utilizarán formas primitivas.

- **PEC 3 – Versión jugable:** 5 semanas. Aquí ya hay que poner todos los elementos en conjunto, creando una versión preliminar pero jugable. Se trabajan temas estéticos, como las animaciones de los personajes o el diseño de los objetos e ítems. Además, se realizará todo el trabajo que tiene que ver con poner en común todos los desarrollos y recursos creados.

- **PEC 4 – Producto Final:** 4 semanas. Como su nombre indica, hay que tener el juego completo, con lo cual es un tiempo para pulir detalles y trabajar en elementos que no tienen tanto que ver con el núcleo del juego pero que también son importantes, como los menús, el HUD o el sonido.

En el siguiente diagrama pueda verse la planificación del proyecto. En general se ha procurado dividir las tareas por semanas, para no tener objetivos demasiado largos. Evidentemente esto es una estimación y su parecido con la realidad será una mera coincidencia.

Hay algunas tareas que requieren más tiempo que las otras. La primera de ellas es la definición del juego, ya que una buena planificación hace que luego se pueda ir directo al grano y haya menos cambios. El segundo es el diseño gráfico, al ser la disciplina más alejada a los conocimientos disponibles. Seguidamente sería la integración, que consistiría en unir todas las piezas creadas e hilar las secuencias. Por último, estaría la fase de test que, aunque cada tarea incluye sus propios test unitarios, es necesario realizar muchas pruebas de forma global para pulir los detalles que sean necesarios.



1.6. Estructura del resto del documento

A lo largo de la memoria se detallarán los siguientes aspectos:

- **Estado del arte**

  Estudio del género de las aventuras gráficas. También una descripción y comparación de las principales herramientas utilizadas para llevar a cabo el proyecto.

- **Definición del juego**

  Descripción detallada de los principales elementos que conformarán el juego. Desde la historia, pasando por el diseño general, hasta las interacciones.

- **Diseño**

  Especificación de las herramientas utilizadas para el desarrollo. Incluyendo la arquitectura del software utilizada, los sprites creados para los personajes y objetos, las animaciones, etc.

- **Implementación**

  Diseño de todos los escenarios disponibles a lo largo de la partida, mostrando los objetos incluidos en estos y sus personajes no jugables.

- **Demostración**

  Incluye las instrucciones del juego, el análisis de costes del proyecto y los requisitos técnicos mínimos.

- **Conclusiones**

  Apartado donde se presentan las conclusiones a las que se han llegado tras finalizar el proyecto, además de las posibles líneas de futuro del juego.

2. Estado del arte

2.1. Género

La mayoría de los videojuegos comparte mecánicas y funciones de diferentes géneros. *Enfrente de los pisos* no es una excepción, sin embargo, la categoría predominante y sobre la que gira toda la experiencia es la aventura gráfica.

Las aventuras gráficas se originaron en la década a los 90 y provienen de un género más antiguo, la aventura conversacional [3]. Debido a la falta de capacidad gráfica en los orígenes de los videojuegos, surgieron un tipo de experiencias en las que lo principal era el texto, donde se presentaba la situación actual de la historia y el jugador, escribiendo por teclado, tenía que decidir qué hacer y la partida tomaba un rumbo u otro en consecuencia. Con los años este género evolucionó añadiendo imágenes y convirtiéndose en lo que se conoce como aventura gráfica, aunque la premisa de juego es la misma. Se presenta una situación en la que el jugador tiene que interactuar con el entorno, tanto con los objetos como con el resto de los personajes. Estas interacciones, si son las programadas por los desarrolladores, producen una serie de resultados que hacen que la historia avance y se logren nuevos objetos o lugares, que a su vez producen nuevos eventos.

Por lo tanto, la experiencia de juego de este género se basa en la resolución de rompecabezas, a diferencia de otros donde podría ser la habilidad o rapidez ejecutando comandos o acciones por parte del usuario. Un ejemplo básico de reto que propone este tipo de juegos puede ser el siguiente: hay puerta cerrada con llave y un personaje que la tiene, pero no la quiere entregar, por lo que es necesario pensar cómo obtenerla. Las posibilidades son infinitas y dependen de cómo lo hayan planteado los desarrolladores, con lo cual el jugador tiene que utilizar su imaginación y la investigación del entorno para hallar la respuesta. Una vez conseguido, se pasaría a otro escenario donde se plantearían otros retos y así sucesivamente.

Los juegos más conocidos del género suelen tener soluciones no demasiado obvias a los rompecabezas para que realmente sea un reto, además de conversaciones con los personajes bien trabajadas, pues es un elemento muy importante en la experiencia.

2.2. Antecedentes

2.2.1. Maniac Mansion

Uno de los estudios de desarrollo más importantes de la historia, en el género de las aventuras gráficas, es sin duda Lucasfilm Games (convertido años más tarde en LucasArts). Uno de los primeros títulos que realizaron es Maniac Mansion en el año 1987 [4]. Este juego sentó las bases del género, añadiendo un menú de acciones, algo que después aparecería en la mayoría de los juegos del estilo.

Figura 2: Maniac Mansion (1987)

Otra de las características que tiene, tanto este como el resto de las producciones de LucasArts, es que los rompecabezas son encadenados, es decir, que para avanzar en la partida hay que solucionar de la forma correcta cada reto. Otros juegos del género, sin embargo, tienen un enfoque distinto. Pudiendo el jugador dejarse algo que sea necesario en una fase más avanzada de la partida y, por lo tanto, no poder superar la aventura.

Otro rasgo distintivo es el uso del humor, tanto en los diálogos como en las situaciones y los métodos para resolver los acertijos. Para este proyecto, la intención es inspirarse en todos los rasgos comentados de este título, su interacción con el entorno, sus rompecabezas encadenados y su humor.

2.2.2. Pokémon

Sin duda, uno de los videojuegos más famosos de todos los tiempos, desarrollado por Game Freak en 1996 y lanzado para la consola portátil Game Boy [5]. No se trata de una aventura gráfica, sino un juego de rol, pero tiene diversos elementos que sirven de referencia para el título a desarrollar.

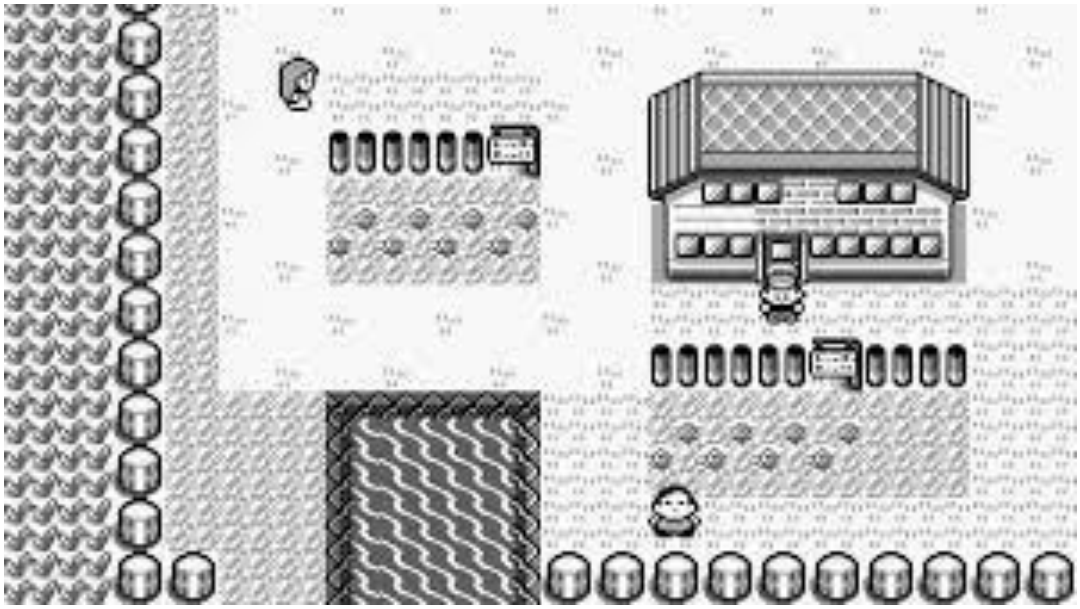


Figura 3: Pokémon Rojo (1996)

La primera característica es su apartado gráfico, en concreto su perspectiva *top down*, la cual se identifica por ser un plano cenital, pero sin estar completamente perpendicular al suelo, con lo cual se ve todo con una ligera perspectiva. No deja de ser un videojuego en dos dimensiones, como por ejemplo Super Mario Bros, pero aquí el eje vertical no representa la distancia respecto al suelo sino un movimiento en dirección norte o sur.

Otro elemento importante es su sistema de movimiento y configuración de los elementos. Todo el mapa está dividido internamente en una cuadrícula, cada personaje por ejemplo ocupa una casilla, mientras que las casas pueden ocupar varias. El movimiento del protagonista solo puede realizarse de coordenada a coordenada, algo que se usará en este proyecto. También el sistema de interacciones, siendo necesario estar delante del elemento en cuestión, con la orientación correcta y pulsar el botón de acción.

2.3. Tecnología

Los videojuegos están cada vez más presentes en la vida de las personas, han pasado de ser un producto de nicho a algo que puede consumir cualquier tipo de usuario. Esto aumento de popularidad ha hecho que su desarrollo haya crecido en los últimos años, a consecuencia de lo cual se pueden encontrar cada vez más herramientas para llevarlos a cabo. Seleccionar la adecuada es algo crucial para llevar a buen puerto un proyecto como este.

El primero a valorar es Godot. Este motor de videojuegos es gratuito y orientado tanto a 2D como a 3D, además de ser multiplataforma. Su lenguaje de programación es una variación de Python e incluye físicas, editor de animación y prácticamente todas las herramientas necesarias para este proyecto. Sin embargo, su uso no está tan extendido como otros, por lo que su comunidad y el acceso a recursos y documentación es más limitado.

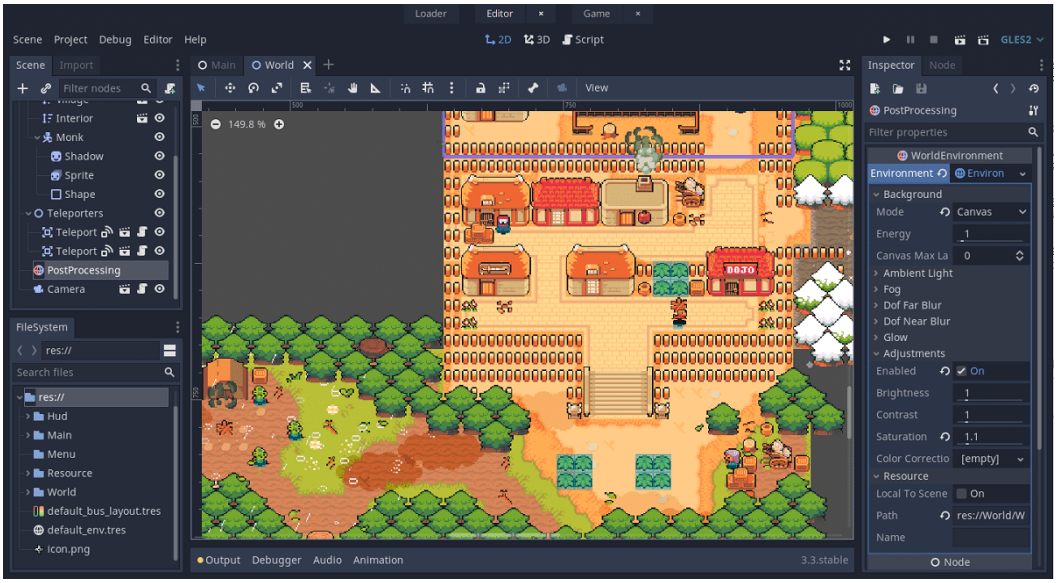


Figura 4: Godot Engine

Siguiendo con la valoración de motores gráficos, uno de los más conocidos y utilizados es Unreal Engine. Desarrollado por Epic Games, ha ido ganando mucha popularidad con los años y, en cada iteración, ha presentado muchas novedades y unos resultados realmente sorprendentes. Sin embargo, sus puntos fuertes salen a la luz en grandes producciones, especialmente en tres dimensiones y con equipos formados por varios profesionales. Por este motivo se considera que no se adecúa con los requerimientos de este proyecto.

Figura 5: Unreal Engine 5

Por último, está Unity. Este motor de juego es multiplataforma y cuenta con una versión gratuita. Su lenguaje de programación es C# y dispone de una amplia gama de herramientas, entre las que está la detección de físicas, control de animaciones, etc. Una de sus grandes virtudes es la gran comunidad de desarrolladores que tiene detrás, además de una curva de aprendizaje asequible y un enfoque más orientado hacia el desarrollo indie. Todo esto hace que se considere como la herramienta ideal para este proyecto.

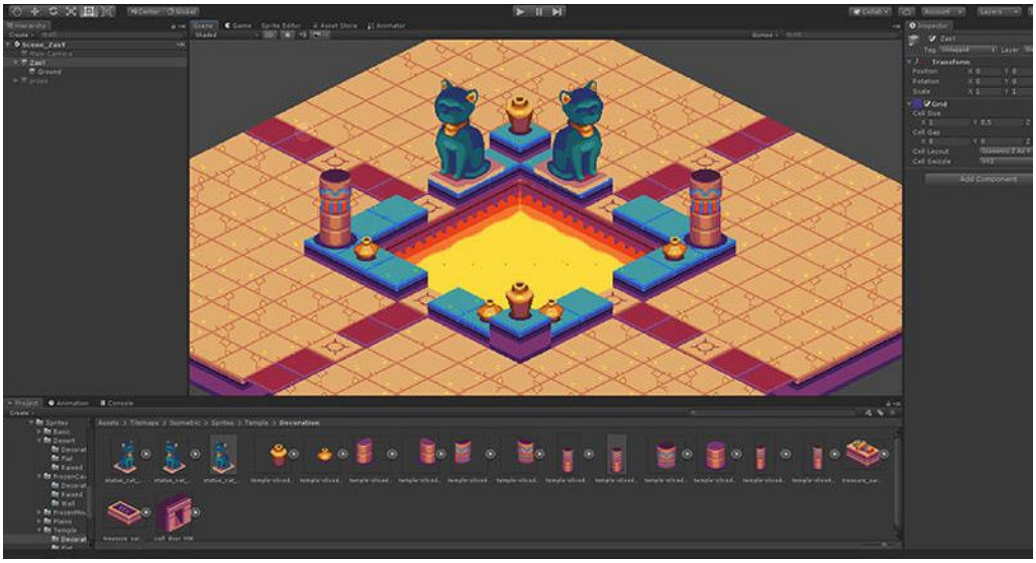


Figura 6: Unity

3. Definición del juego

3.1. Descripción del juego

Enfrente de los pisos es una aventura gráfica en la que hay que ayudar al protagonista a huir de la cárcel en la que se encuentra retenido. Al igual que en prisión, hay un horario muy estricto y, cada pocos minutos, se cambia de escenario. En cada uno de ellos se encontrará con diferentes personajes y objetos, los cuales tiene que utilizar para alcanzar su objetivo final.

La partida se desarrolla de forma cíclica y sin la posibilidad de guardar, ya que, una vez que se conoce la solución, ejecutarla se puede hacer de forma rápida. Se pueden jugar partidas cortas con la intención de investigar ciertas acciones e ir aumentando el conocimiento sobre lo que funciona y lo que no.

Una vez conseguidos todos los objetivos se produce la fuga. Esta se desarrolla como un nivel independiente y, si se supera, termina la partida.

3.2. Historia y ambientación

El juego está ambientado en algún momento indeterminado del siglo pasado, donde no existía la tecnología de hoy en día y en las cárceles no había grabación de imágenes, aunque a lo largo de la partida no se concreta. Del mismo modo, no se especifica dónde está ubicada esta prisión. De esta forma el usuario puede hacerla suya e imaginar que lo ocurrido se desarrolla en una que conozca.

Tampoco se especifica una historia sobre el personaje, ya que no es el elemento central de la experiencia, sino que es lo que pasa dentro de la cárcel. Es como si, al entrar, se pasara a formar parte de un mundo completamente distinto a lo que había fuera y, cualquier pasado que puedan tener los reclusos, no importa dentro.

3.3. Interacciones

Los controles son más bien escasos, buscando la simplicidad característica de este tipo de juegos y de esta plataforma. Entre las acciones que se pueden realizar se encuentran las siguientes:

- Movimiento del jugador: Se utilizan los botones direccionales habituales para dirigir al protagonista.
- Ítems: En la parte inferior de la pantalla están los ítems recolectados, el inventario se encuentra completo en la pantalla, no dentro de un menú o apartado, así que para seleccionarlos y poder utilizarlos simplemente hay que pulsar sobre ellos.
- Menú: Un botón que permite interrumpir la partida para acceder a las opciones.
- Acción: Este botón se utiliza para interactuar con el entorno, aunque su nombre cambia en función del contexto para dar feedback. Por ejemplo, si se selecciona un ítem este botón mostrará la palabra “usar”, si por el contrario no hay objeto seleccionado y el protagonista se encuentra delante de un personaje secundario, aparecerá la palabra “hablar”. Cuando no hay interacción posible el botón desaparece, dando así indicaciones al jugador de cuándo puede y cuándo no interactuar con el entorno.
- Siguiente escena: También en la parte inferior, hay un botón que permite saltar a la siguiente escena. Esto permite que la partida no se haga tediosa si ya se conoce parte de la solución.

En cuanto a los NPC, siguen unos caminos ya establecidos, en base a unos checkpoints que recorren de manera cíclica.

3.4. HUD

El juego está pensado para jugar en pantallas con formato 16:9 en posición vertical, tomando como ejemplo una resolución de 1920x1080. La imagen está dividida en tres secciones, una de ellas cuadrada donde se mostrará el contenido del juego. Por lo tanto, están los siguientes apartados:

- Sección superior (420x1080): Al ser la parte más alejada de los dedos se procura que haya pocas interacciones en esta zona. En ella están los diálogos y el botón de menú.
- Sección central (1080x1080): Contenido del juego, donde se muestra el personaje, el escenario, etc. Además, es la zona en que se realizan los gestos para el movimiento del personaje.
- Sección inferior (420x1080): Es la zona más accesible y por tanto donde hay más interacciones. Aquí se ubica el botón de acción y el inventario.

Figura 7: Boceto del HUD

4. Diseño

4.1. Entorno de desarrollo

4.1.1. Unity

Como se había adelantado en el apartado de tecnología, el entorno de desarrollo de videojuegos seleccionado es Unity. Tanto por estar familiarizado con él, dado su uso a lo largo del máster, como por ser adecuado para la naturaleza del producto llevar a cabo, un videojuego en dos dimensiones y de bajo presupuesto.

La versión utilizada ha sido la 2020.3.25f1, ya que era, en el momento de comenzar el desarrollo, una de las versiones LTS más recientes.

4.1.2. Visual Studio Code

El editor de código gratuito de Microsoft ofrece todas las herramientas necesarias para implementar los scripts de este proyecto. Es compatible con la gran mayoría de lenguajes de programación, en este caso con C#. Además, es gratuito, de código abierto y con una comunidad muy activa, la cual proporciona extensiones que facilitan el trabajo, a través del autocompletado y características similares.

4.1.3. Photopea

Photopea es un editor gráfico totalmente online, con lo cual no es necesario descargar nada y es gratuito, siempre y cuando se permita la emisión de anuncios en la web. Para este proyecto los diseños son sencillos, solamente hace falta una cuadrícula representando los píxeles, una herramienta de lápiz que permita pintar cada casilla individualmente y una paleta de colores.

Todo esto lo proporciona Photopea y, como las animaciones del videojuego son muy sencillas, se ha utilizado también para crear cada frame y luego montarlo en Unity. Por lo tanto, toda la parte visual de este proyecto ha sido creado utilizando esta herramienta, teniendo en cuenta además que no hay uso de recursos externos.

4.1.4. Reaper

Para la creación de todos los elementos de sonido se utiliza Reaper. Este DAW permite la edición multipista, el uso de efectos, VST y cualquier cosa que se la pueda pedir a la producción de audio. Incluye una versión gratuita y puede ser utilizado en múltiples sistemas operativos.

La experiencia previa con esta herramienta y sus cualidades la hacen ideal para crear la banda sonora y los efectos necesarios.

4.1.5. Microsoft Office

El conjunto de programas de ofimática por excelencia. En este proyecto se empela tanto para crear la documentación necesaria como, en las hojas de cálculo, hacer una composición de todas las interacciones que hay en el juego.

4.1.6. GitLab

Este repositorio de código permite tener un control sobre todas las versiones de los scripts implementados y del estado general del desarrollo, pudiendo así volver a fases anteriores en caso de error. GitLab se ha considerado como la mejor opción, dado que es gratuito y ha sido la herramienta utilizada a lo largo del máster.

4.1.7. OBS

OBS permite, entre otras cosas, grabar el contenido de la pantalla del ordenador. Esta herramienta es muy útil para crear los vídeos del contenido in-game, necesarios para este proyecto. También es gratuito y de código abierto, algo que ayuda a controlar los costes del desarrollo.

4.2. Requisitos técnicos

A continuación, se detallan los requisitos mínimos necesarios para ejecutar Unity, el cual es el programa más exigente de los utilizados.

Sistema Operativo:
Windows 7 o superior, Mac OS 10.13 o superior, Ubuntu 18.04 o superior

CPU:
X64 arquitectura con SSE2

GPU:
Compatible con DX10 o superior, con Metal o con OpenGL 3.2 o superior.

Almacenamiento:
100MB de espacio.

RAM:
1GB como mínimo.

4.3. Assets

En este aparatado se incluyen todos los assets utilizados para desarrollar el proyecto. Por la naturaleza de éste, todo el material gráfico está elaborado en dos dimensiones utilizando la técnica píxel art, tomando como base un lienzo de 16x16.

Una de las premisas era no utilizar nada ya existente, por lo que todo lo mostrado a continuación se ha elaborado específicamente para este videojuego.

4.3.1. Personajes

En aras de la eficiencia, todos los personajes del juego están creados a partir del mismo patrón, aunque con algunas modificaciones para dejar patente que no son iguales.

A continuación, podemos ver al protagonista, que resalta por tener una melena rizada de color negro. Al ser un preso sus ropas son de color naranja. Para crear al resto de reclusos se ha optado por modificar la cara, pero dejar el cuerpo igual, esto es una ayuda también para crear las animaciones, ya que lo que presenta movimiento es el cuerpo, mientras que la cabeza se mantiene quieta.

Figura 8: Diseño protagonista

En la siguiente imagen puede verse un ejemplo de cómo, cambiando el pelo y las sombras de la cara, se crea a otro personaje que se percibe como distinto.

Figura 10: Diseño NPC 1

Figura 9: Diseño NPC 2

En el caso de los guardas ocurre algo parecido, aunque en este caso no es necesario diferenciarlos pues no aportan valor narrativo, todos son iguales para el jugador ya que lo único que hacen es atraparlo, no hay conversaciones con ellos ni algún otro tipo de interacción. Hay dos modelos de vigilante, el que dispone de gorra y se pasea por el patio y la de la cocina, la cual al sí poder interactuar con ella tiene una apariencia distinta.

Figura 12: Diseño guardia común

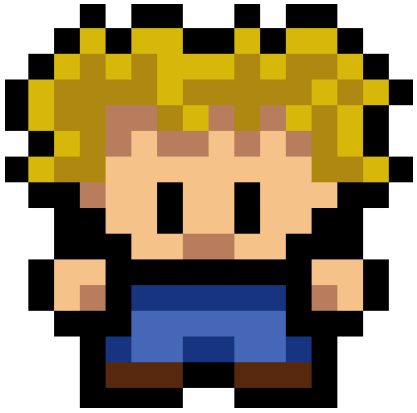


Figura 11: Diseño guardia comedor

En cuanto a las animaciones, todos comparten el mismo esquema, que podemos ver a continuación. Hay 2 apartados principales: Idle y Movement. Este último, que representa el caminar de los personajes, tiene dos variantes sobre las que va alternando, una por cada pierna. De esta forma cuando se avanza, en la primera pulsación lo hace con la pierna izquierda y, en la segunda, con la derecha.

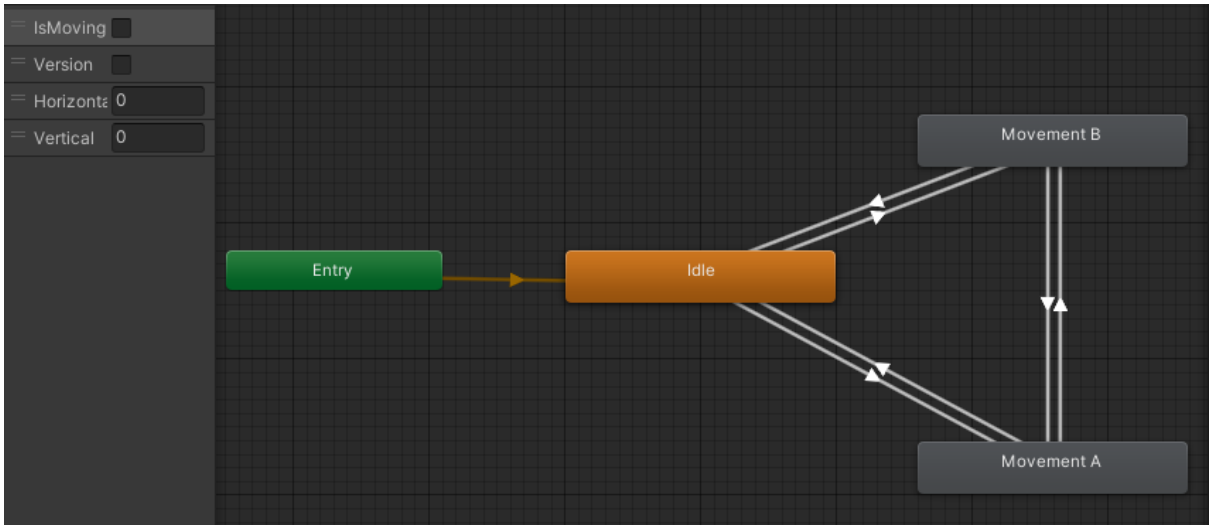


Figura 13: Diagrama principal de animaciones

Ya dentro de esas categorías, se selecciona una animación u otra en función de la dirección que tenga el personaje en ese momento, según los ejes X e Y, alojados en variables de tipo float.

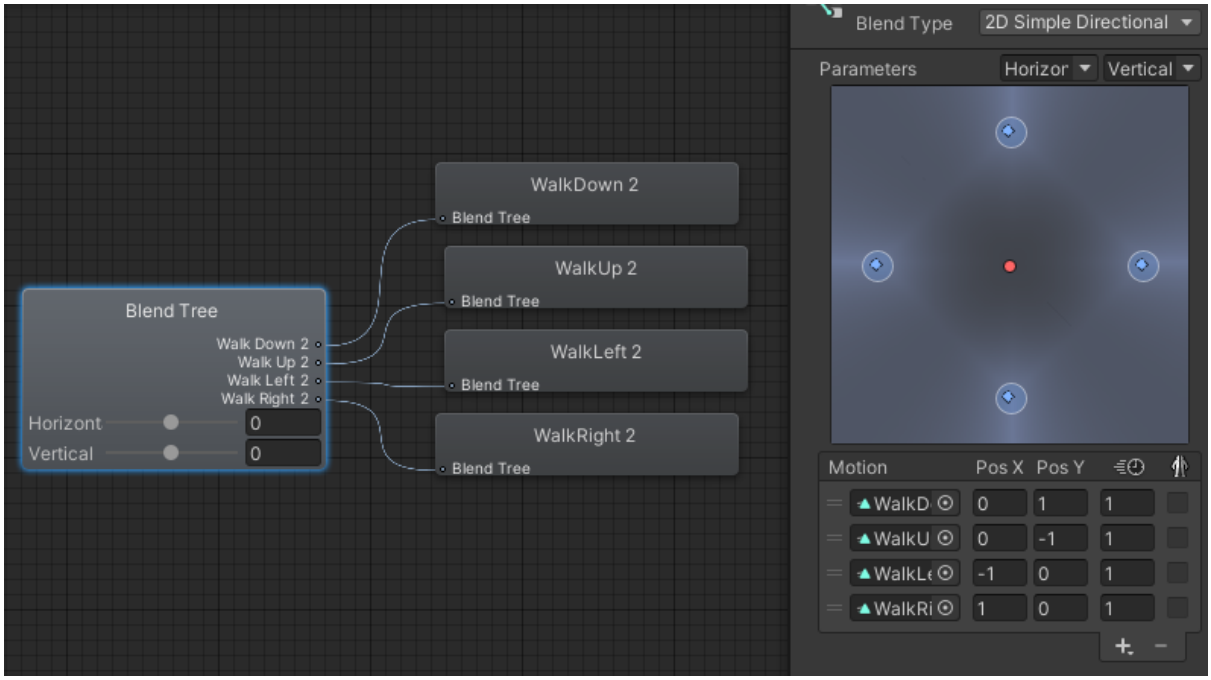


Figura 14: Diagrama de animaciones de movimiento

En total, las animaciones de los personajes cuentan con 12 fotogramas, 3 para cada dirección. A continucación puede verse el ejemplo del protagonista. En el resto de personajes el funcionamiento es exactamente el mismo.

Figura 15: Sprites de movimiento del jugador

4.3.2. Objetos

Los objetos siguen la misma lógica de píxeles que los personajes, aunque algunos, por su gran tamaño, están creados a partir de más de una baldosa. Un ejemplo es la cama, que se muestra a continuación, la cual mide 16x32, formado por 2 unidades. También la mesa, de 32x32, con 4 unidades. En este último caso, hay 2 piezas intermedias que permiten crear una mesa del ancho que se quiera.

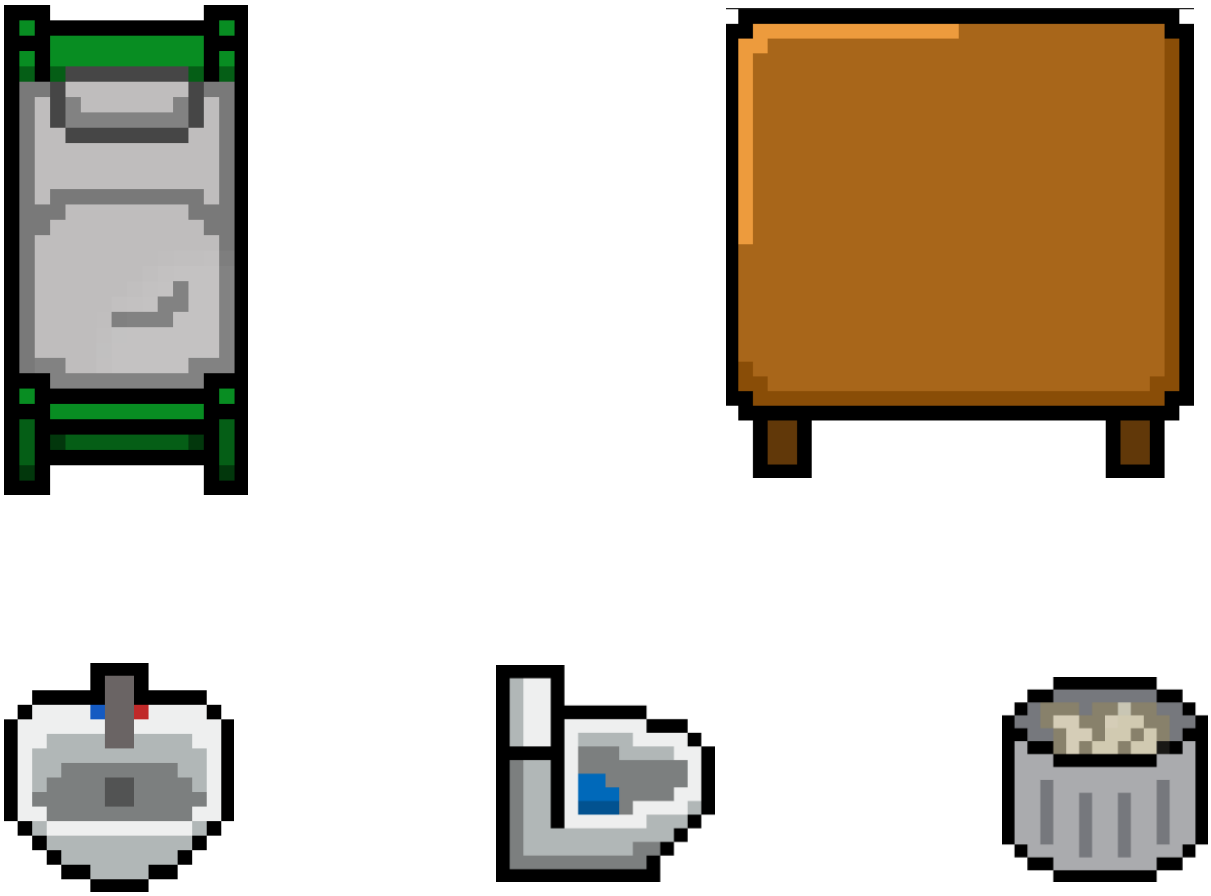


Figura 16: Diseño de objetos

4.3.3. Ítems

Los ítems, son los elementos que el jugador puede añadir a su inventario para, posteriormente, utilizar sobre los personajes o sobre objetos del entorno. Su formato sigue las mismas directrices que el resto.

Figura 17: Diseño de ítems

4.3.4. Entorno

En el caso de los gráficos de los escenarios, como son los suelos o las paredes, se ha procurado realizar unos diseños que puedan ponerse juntos para formar una unidad. Por ejemplo, para los ladrillos de las paredes se han creado dos variantes que, al ponerlas juntas, no se percibe que son elementos individuales. Algo similar ocurra con las cajas que sirven para ocultarse en el patio, su diseño permite unir varias.

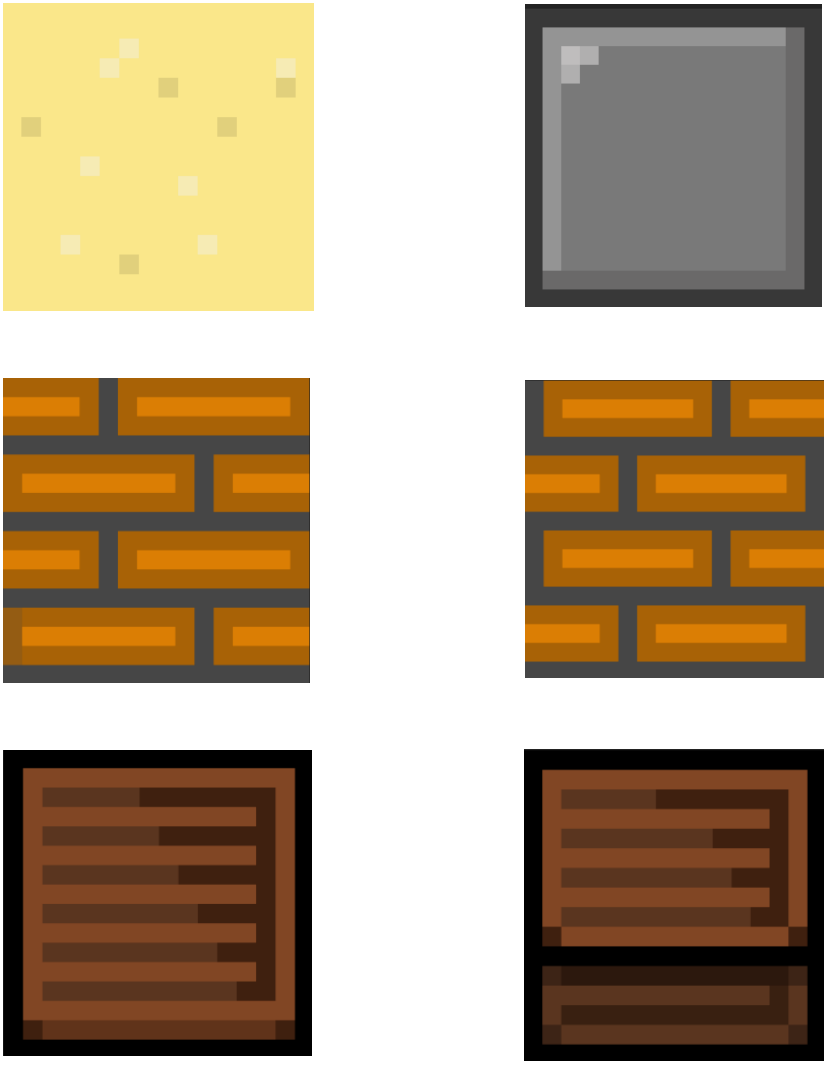


Figura 18: Diseño del entorno

4.3.5. Minijuegos

Para los minijuegos también ha sido necesario realizar material gráfico. Para el que se realiza en el comedor, se ha diseñado una mano, la cual representa la de los compañeros presos intentando robar al protagonista su postre. Para el minijuego del taller, se han dibujado un martillo y unos clavos. En el caso del de la celda, no ha sido necesario realizar material adicional, ya que se han utilizado los ladrillos hechos para las paredes.

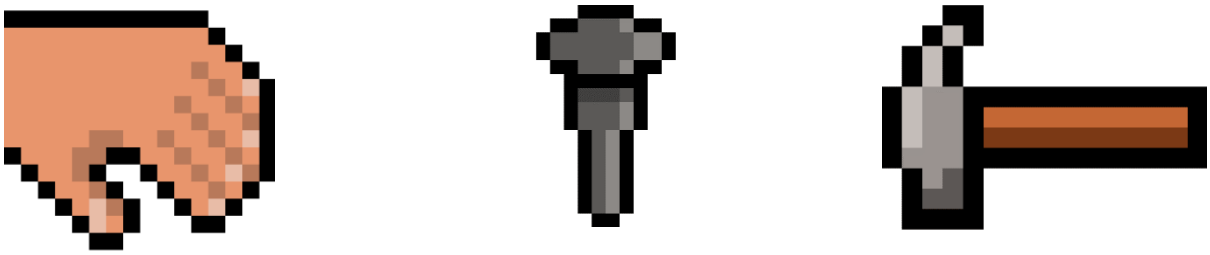


Figura 19: Diseño de minijuegos

4.3.6. UI

Para conseguir una estética unificada también se han diseñado los botones de la interfaz. Entre ellos están el de pausar la partida, el de pasar la escena actual y uno genérico para alojar dentro el texto que corresponda.

Figura 20: Botones de la UI

4.4. Arquitectura

Este proyecto está formado por una serie de escenas, las cuales se ejecutan de forma cíclica salvo cuando se cumplen ciertas condiciones. El siguiente esquema muestra cómo un día del juego comienza a las 08:00 y termina a las 23:00, dividiendo esas horas entre los distintos entornos. Esto se repite hasta que el jugador haya conseguido todos los objetivos, momento en el cual al llegar la noche se pasa a jugar la pantalla final, que consiste en huir procurando no ser visto por los guardias.

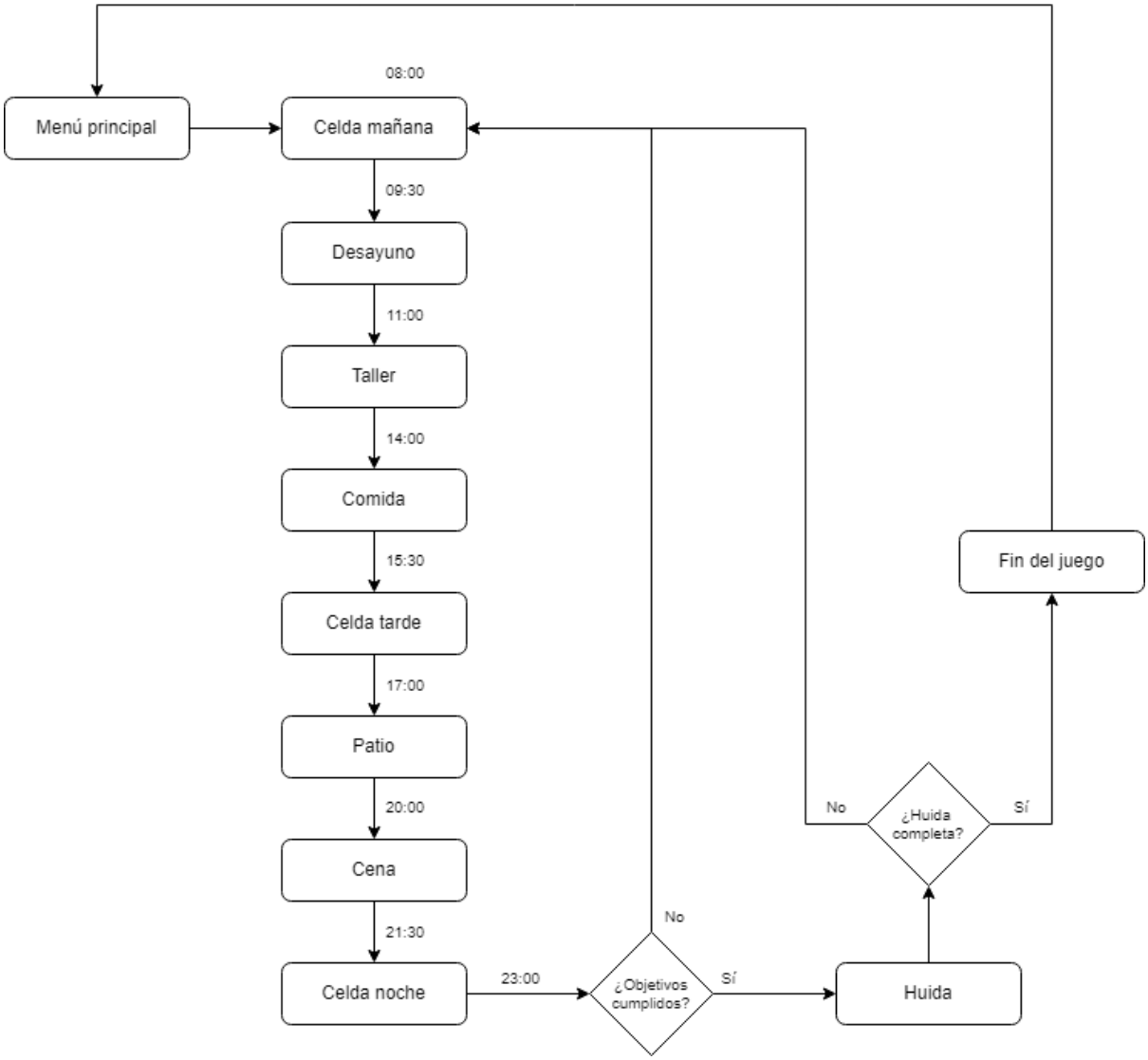


Figura 21: Flujo de las escenas del juego

La arquitectura del software de este proyecto se caracteriza por disponer de una serie de scripts, llamados managers, que controlan distintas áreas de la lógica. La comunicación entre ellas se procura que sea mediante eventos, siempre que sea factible, para conseguir así eliminar las dependencias o la necesidad de que se hagan referencia entre sí. Un elemento que también contribuye al aislamiento de la lógica es el uso de un scriptable object que aloja información importante de la partida, como el inventario o el estado de ciertos eventos propios del juego.

La implementación de este sistema de eventos se sustenta en dos elementos. El primero de ellos es un scriptable object llamado GameEvent que contiene la definición del propio evento. Incluye una lista donde se alojan todos los listeners que están pendientes de él y una función, Raise, que notifica a todos ellos de que el evento ha ocurrido.

La otra entidad necesaria es la clase GameEventListener, que en este caso hereda de MonoBehavior para poder añadirla en el inspector del GameObject.. Esta clase incluye una referencia el GameEvent y una a un UnityEvent.

Su funcionamiento es el siguiente, en los objetos donde se quiera ejecutar una función en base a un evento, se añade un GameEventListener y se referencia el scriptableobject del evento escuchado y la función interna a ejecutar cuando dicho evento se lance. A continuación, podemos ver un ejemplo con Input Manager, el cual hace de emisor de distintos eventos en función de la tecla que se pulse.

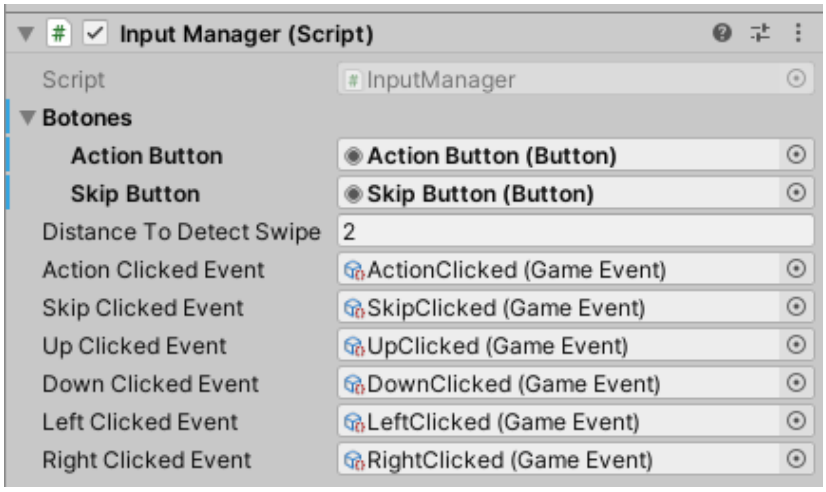


FIgura 22: Inspector de la clase Input Manager

Por otra parte, tenemos al jugador, que está escuchando esos eventos para ejecutar una función en consecuencia. En la siguiente imagen se ve cómo, si Input Manager activa el evento “ActionClicked”, esto acciona la función homónima dentro de su clase PlayerManager o, si el evento es “RightClicked”, se activa un método en PlayerMovement.

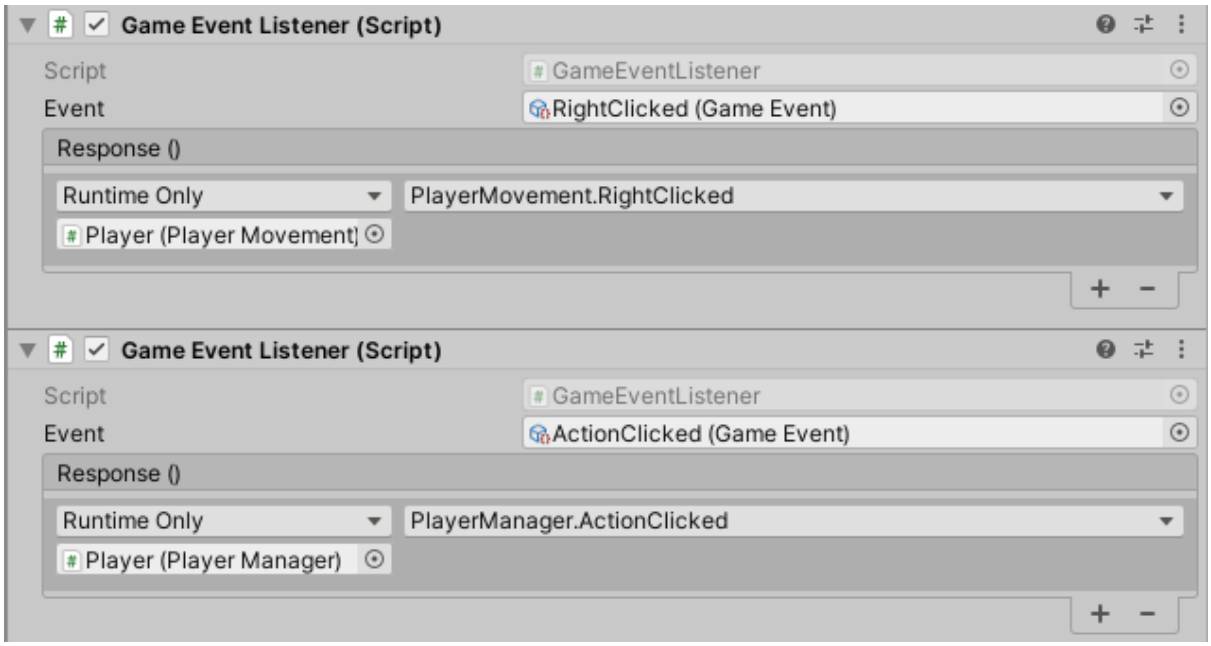


FIgura 23: Inspector del GameObject del jugador

También se incluye una máquina de estados, aunque no con una implementación para cada estado, sino que estos generan eventos que cambian el comportamiento de los controladores. Por ejemplo, mientras el estado sea Menú, el tiempo del juego no pasa y pulsar hacia arriba no hace que el jugador se mueva.

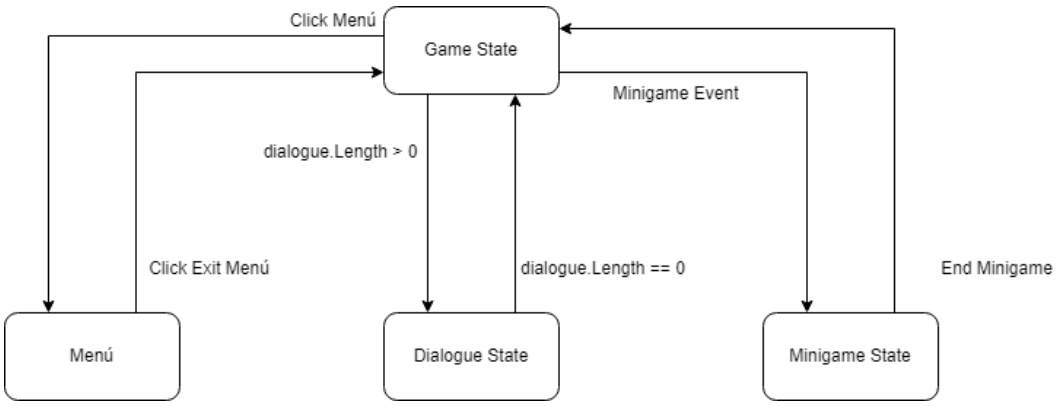


Figura 24: Diagrama de estados del juego

Hay una multitud de clases que conforman toda la lógica, pero a continuación se muestran las más relevantes.

4.4.1. Input Manager

Esta clase se encarga de controlar todo lo referente a la interacción del usuario con el juego. Captura tanto pulsaciones de botones de la interfaz como de los controles direccionales, así como el movimiento del ratón o si se hace click sobre algo. Para ello dispone de una serie de atributos editables, como por ejemplo el rango necesario para diferenciar entre una pulsación y arrastrar con el cursor. Cuando se produce alguna de estas acciones lanza el evento correspondiente.

4.4.2. Canvas Manager

Canvas Manager se encarga de mostrar y actualizar todo lo referente a la interfaz de usuario. En general no dispone de ninguna lógica propia, sino que son otras clases o mediante eventos los que le introducen los datos a mostrar.

En primer lugar, tiene una función pública para actualizar el tiempo, cuyo valor lee de un scriptable object que rellena el Game Manager.

En segundo lugar, se encarga de mostrar el inventario. Esta clase está suscripta a un evento que lanza el script encargado del inventario y, cuando hay alguna modificación, se ejecuta una actualización de los ítems mostrados. Esta comunicación es bidireccional ya que, cuando el usuario pulsa sobre un ítem esta información se le envía al scriptable object para guardar cuál es el elemento seleccionado.

También, de forma similar al tiempo, se encarga de mostrar y modificar el contenido del botón acción. El Player es quien evalúa el contenido y lo notifica en el scriptable object, pero es el Canvas Manager quien lo muestra.

Por último, está la representación de los diálogos. Esta clase dispone de un atributo string el cual se va llenando a petición de otros componentes. Mediante una corutina, cada vez que este string se llena se lanza un evento para notificar al resto que se cambia de estado a Dialogue State y se muestra sobre la pantalla el texto contenido en dicho string, separándolo por frases, hasta que se vacía y se notifica el fin de dicho estado.

4.4.3. Player Movement

Este script se responsabiliza de mover al jugador por la pantalla y de controlar sus animaciones, además de leer el objecto que tiene delante para que se pueda modificar el botón acción. Dispone de unos atributos que permiten editar su velocidad de desplazamiento y qué tilemap se considera válido para pisar. Cada vez que un evento del Input Manager es lanzado, se ejecuta la función del movimiento correspondiente. Antes de hacerlo, se comprueba que la posición de destino es válida, mirando si hay algún obstáculo mediante un Raycast y así evitar que se pueda traspasar nada.

Si se considera como un movimiento posible, se activa la animación de caminar y se amplía el collider del propio jugador para que ocupe, además de su casilla, la de destino. La finalidad de esto es que ningún NPC pueda intentar moverse a esa posición mientras se está realizando un desplazamiento. Esta misma lógica también la tienen los NPC para que tampoco el jugador pueda hacer un movimiento a una posición done otro ya ha decidido posicionarse. Mediante un bucle y MoveTowards se transporta al jugador a su destino a la vez que se va moldeando su collider para dejarlo como estaba, ocupando solo su posición.

4.4.4. NPC Movement

Su funcionamiento es muy similar al Player Movement, con la diferencia que la posición de destino de un movimiento se calcula a través de unos Wander Points, los cuales el personaje va siguiendo, casilla a casilla, hasta que los alcanza. En ese momento su destino se marca como el siguiente de la lista.

Dispone de algunos parámetros, como por ejemplo si el movimiento se hará en bucle, es decir, si una vez llegado a la última posición de la lista se vuelve a la primera.

También controla la animación y está suscrito al evento de cambio de estado del Manager principal, para que, si no se está en el estado apropiado, los personajes no se muevan.

Como algunos NPC son guardias, incluye un FieldOfView, que básicamente es una clase que dibuja un triángulo en la escena, representando lo que puede ver ese personaje y así el jugador sepa cuándo será detectado. Este campo de visión tiene en cuenta los colliders del entorno y se adapta a ellos. Si el jugador entra en contacto con su collider se lanza un evento para notificar que ha sido descubierto.

4.4.5. Camera Manager

Se responsabiliza de la ubicación de la cámara. Por defecto sigue al jugador, por lo que en todo momento este se mostrará en el centro de la pantalla. Para hacer esto la clase incluye una referencia al Player Movment de la escena y, en el Update, actualiza su posición en base a la del protagonista.

Como las escenas pueden incluir minijuegos, la cámara tiene la ubicación de estos, ya que están en la misma escena, pero alejados para que no puedan verse mientras se controla al jugador. Cuando el Manager principal cambia el estado de la partida a minijuego, la cámara, que está suscrita a este evento, cambia de posición para mostrarlo, Una vez se cambia de estado, la cámara vuelve a su posición anterior.

4.4.6. Object, Item y NPC Manager

Estas tres clases son muy similares, aunque cada una con ciertos matices, pero con la misma finalidad, devolver un resultado a las interacciones del usuario con los mismos.

Tienen asociado un Scriptable Object donde se definen todas las interacciones que acepta ese elemento, teniendo en cuenta múltiples parámetros como la hora a la que ocurre, los ítems de entrada, los eventos que han ocurrido, etc. Su principal tarea es encontrar la interacción de la lista que coincida con la entrada y entregarla al Manager principal.

Internamente está compuesto por dos clases, Input Interaction y Output Interaction, las cuales moldean los datos de entrada y salida que tiene una interacción. Las variables principales son los ítems utilizados, la hora y el estado del juego, definido por scriptable objects.

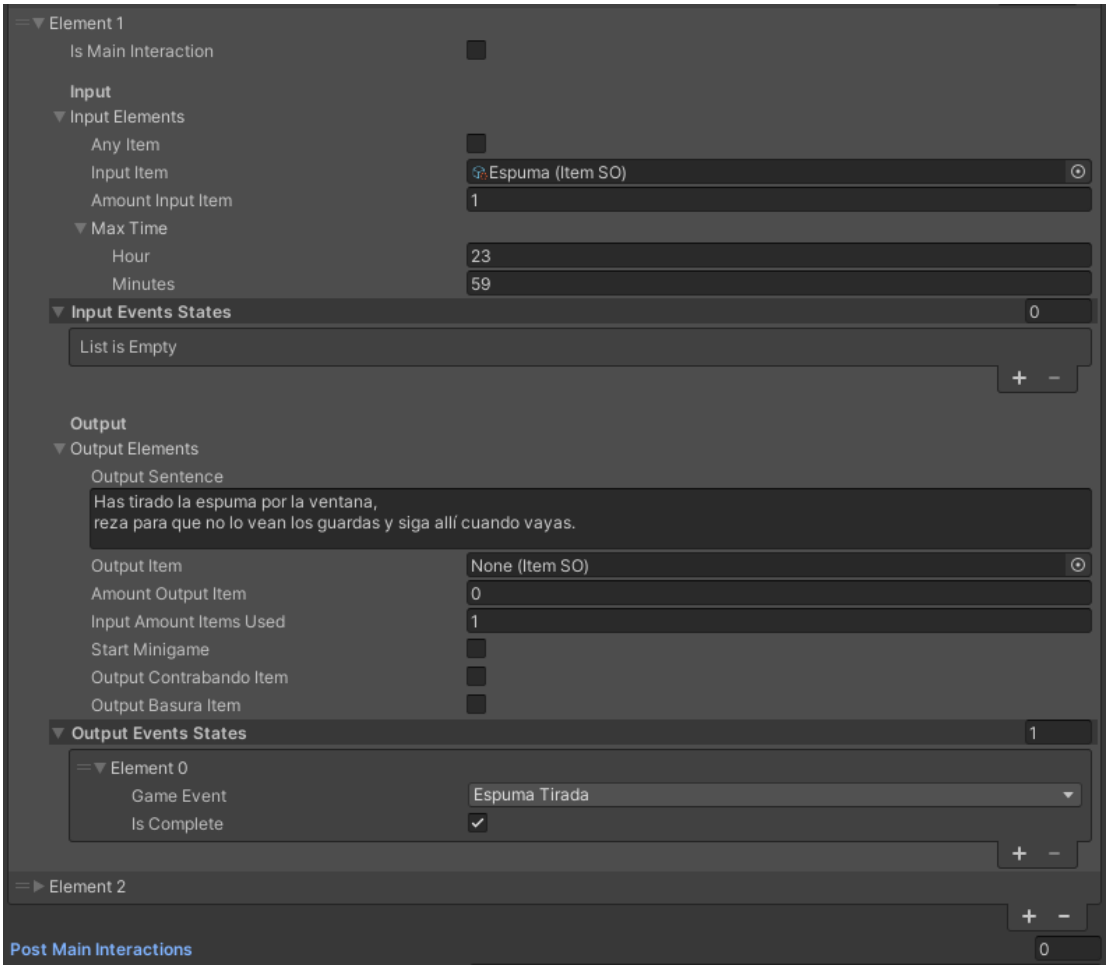


Figura 25: Sistema de configuración de interacciones

En la anterior imagen podemos ver un ejemplo de interacción definida en un Scriptable Object, en este caso de la ventana de la celda. Las condiciones de entrada, dentro del grupo Input, son: que el ítem seleccionado por el usuario en el momento de la interacción sea la espuma, la hora es indiferente (por esa razón la máxima hora es 23:59) y no es necesario comprobar ningún evento del juego. Las condiciones de salida, marcadas como Output, son una frase, que el Canvas Manager mostrará, ningún ítem de salida, que se gaste una unidad del ítem de entrada y que se marque como completado el evento de juego “espuma tirada”.

4.4.7. Manager

Esta última clase se encarga de varias cosas. En primer lugar, es la encargada de cambiar de estado y lanzar un evento para informar a los componentes que estén suscritos.

También controla las escenas, las cuales pueden cambiar por varias razones, entre ellas porque ha llegado el tiempo indicado, o porque el usuario pulsa el botón de “saltar escena”. Ese primer parámetro se define desde el inspector.

Otra de las posibilidades por las que puede haber un cambio de escena es cuando el jugador es atrapado por las guardias. En ese momento estos generan un evento al cual está suscrito el Manager.

Otra de las funciones que tiene es borrar todo el inventario cuando la escena corresponde con un cacheo, en concreto cuando el protagonista entra en la celda.

5. Implementación

En total, este proyecto cuenta con 4 escenarios principales, algunos de los cuales pueden estar repetidos con pequeñas variaciones.

5.1. Celda

Esta estancia se visita, a lo largo de una jornada dentro del juego, en tres ocasiones. En todas ellas el diseño del nivel es el mismo, existiendo algunos cambios a nivel de personajes.

Al tratarse de la primera escena jugable se ha procurado que sirva también como tutorial para la jugabilidad y las mecánicas. Es por ello por lo que toda la escena puede verse de un vistazo, sin necesidad de recorrer distancias. También el hecho de incluir un subapartado, en este caso el baño, enseña al jugador cómo están delimitados los espacios, en concreto, que si no hay suelo es que no se puede pisar.

Figura 26: Diseño de la celda

Se han añadido elementos para mostrar cómo funciona la mecánica de las interacciones y qué tipos hay. Esta es una de las razones por las que, además del protagonista, hay un compañero en la celda. Dicho compañero muestra distintas frases en función de la hora y sus diálogos sirven de guía. También podemos encontrar distintos objetos con los que se puede interactuar, como la ventana, la cama y el baño.

Este nivel, además, incluye un minijuego oculto que se activa interactuando con el trozo de pared que hay entre el lavabo y el váter. Al ser indispensable superarlo para el desarrollo del juego, se ha ubicado en una zona donde puede percatarse de su existencia sin buscarlo, aunque las conversaciones con el resto de los reclusos dan pistas para encontrarlo.

5.2. Comedor

Al igual que ocurre con la celda, el comedor también se visita tres veces al día: desayuno, comida y cena. Su diseño tampoco tiene variaciones, salvo alguna a nivel de personajes.

La escena empieza con el jugador cerca de la cocinera, empujando así a interactuar con ella y que su presencia no pase desapercibida. Al hacerlo se obtiene un ítem de comida, añadiendo un nuevo elemento a la jugabilidad.

Incluye también un minijuego que se activa al utilizar la comida recientemente recibida sobre el sitio adecuado. Para realzar dicha ubicación se han colocado a todos los personajes no jugables sobre sillas y se ha dejado solo una libre, procurando así generar una relación de conceptos en el jugador.

Entre los elementos con los que se puede interactuar hay una basura, cuya utilización es necesaria para el progreso del juego. Por esta razón se ha colocado cerca del asiento asignado al jugador, de forma que cuando se dirija hacia allí vea este objeto.

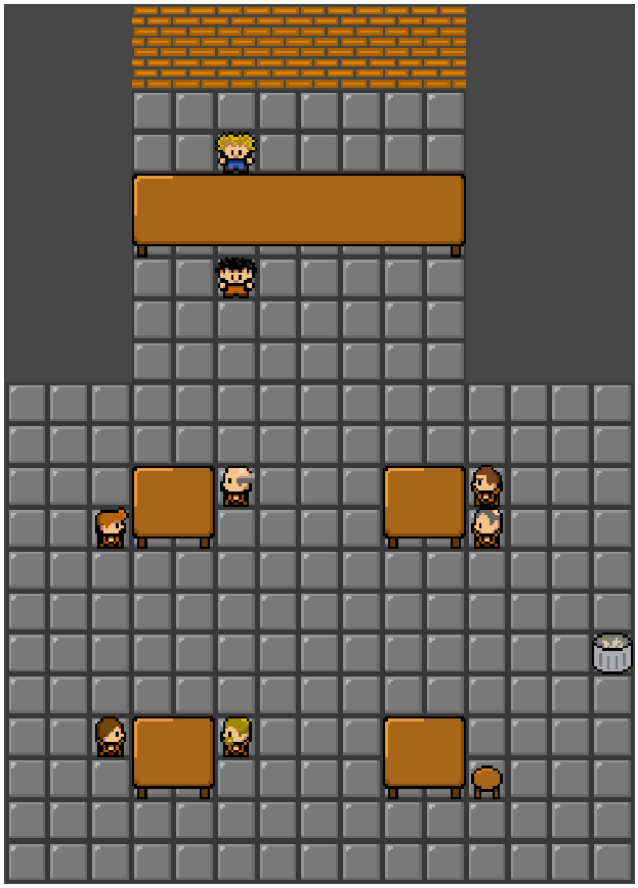


Figura 27: Diseño del comedor

5.3. Taller

De forma similar a lo que ocurre en el comedor, aquí también hay un minijuego, aunque este se activa interactuando sin ningún ítem. Se reutiliza la idea de la silla vacía, con la que el jugador ya está familiarizado.

Aquí, además de la zona de trabajo, hay una sección al fondo cuyo paso está bloqueado por un guardia. Para verlo el jugador debe investigar un poco ya que no se ve si sigue simplemente el trayecto hasta su mesa o solo habla con sus compañeros. A partir de cierta hora el guardia cambia de posición, permitiendo pasar.

Hay una verja cerrada, para cuya apertura es necesario un código. El teclado para que el jugador introduzca la clave se ha ubicado junto a la puerta, para que sea fácil ver la relación que tienen. Una vez la puerta está abierta se puede interactuar con la mesa, de la forma a como se hace con la de trabajo, aunque con la particularidad de que aquí sí es necesario utilizar un objeto.

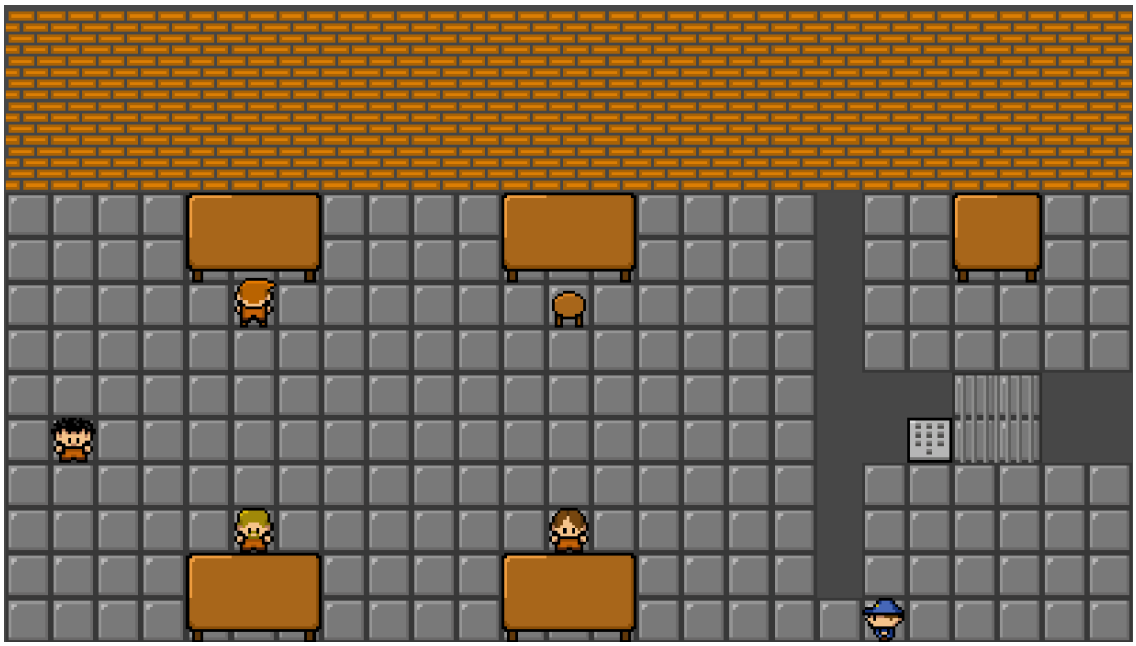


Figura 28: Diseño del taller

5.4. Patio

El patio es, sin duda, la estancia más grande del juego. Incluye una variación, en el momento de la fuga, con algunos guardias más, pero manteniendo el mismo diseño.

Está dividido en dos secciones. La primera de ellas, a la izquierda de la imagen, es la zona segura, donde están el resto de los personajes no jugables. Cada uno de ellos tiene un movimiento predeterminado, pero siempre dentro de los límites.

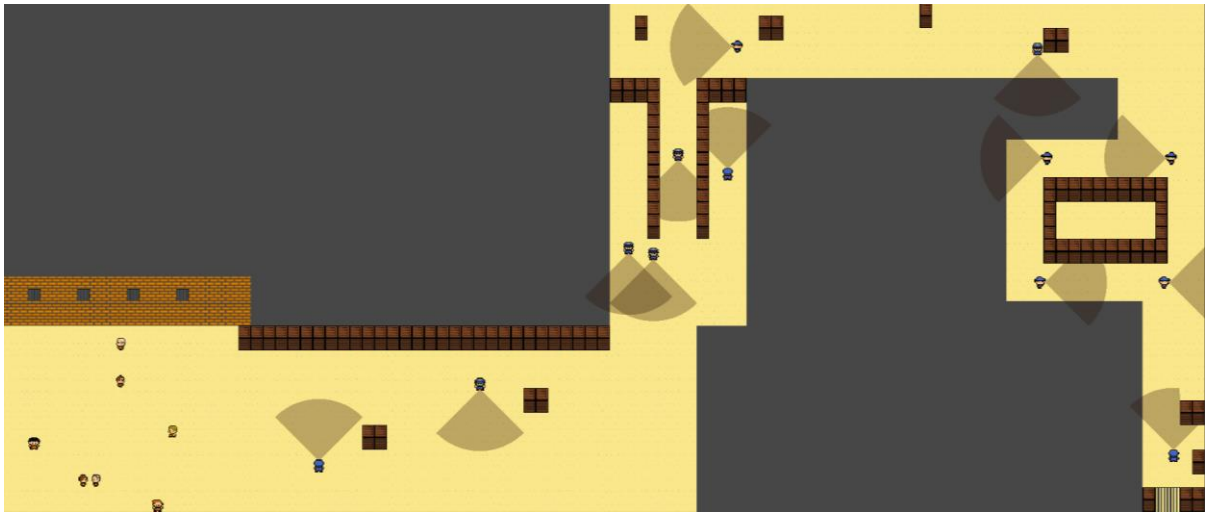


Figura 29: Diseño del patio

La otra zona está llena de guardias, cada uno de ellos con un campo de visión que, si el jugador entra en él, es capturado. Internamente hay 5 subsecciones. En la primera hay un par de guardias con algunos obstáculos para que el jugador aprenda cómo funciona el campo de visión. En la segunda hay tres guardias paseando arriba y abajo, aunque solo uno de ellos está en un camino que tenga salida. Por el tamaño del pasillo es necesario recorrer algunos pasos dentro de él, para que si el jugador toma la decisión equivocada sea atrapado. La tercera parte empieza con un guardia que se mueve de izquierda a derecha, ocupando todo el espacio de visión, siendo necesario colocarse tras la caja para no ser visto. Seguidamente, hay una situación muy similar pero este guardia sí modifica su trayectoria para que, si el jugador no se mueve, sea atrapado. A continuación, hay cuatro guardias caminando en círculos, están a cierta distancia para que quede un pequeño hueco entre ellos, obligando al jugador a caminar su ritmo y por el mismo camino para conseguir llegar al final. En este último tramo hay un solo guardia junto a la puerta de salida. En ese momento sirve para conseguir un ítem, interactuando por la espalda, y para que el jugador no pueda acceder a la puerta de salida. En el momento de la fuga este guardia no está, dejando así vía libre hacia la libertad.

6. Demostración

6.1. Instrucciones de juego

A continuación, se describen los controles y acciones disponibles a lo largo del juego, en concreto para teclado y ratón. Al tratarse de una aventura gráfica sencilla, solo hay de dos tipos, desplazamiento e interacción:

- Movimiento del jugador: Botones direccionales.
- Acción: Espacio / Click izquierdo sobre el botón

La mejor estrategia para pasarse el juego es, sin duda, la investigación. Hablar con todos los personajes no jugables a diferentes horas, procurar recolectar todos los ítems posibles y probarlos donde sea posible. En cuanto a la sección de sigilo, requiere de recorrer la zona y estudiar los movimientos de los guardias para poder llegar al final sin ser detectado.

6.2. Análisis de cotes

En este apartado se analiza el coste estimado de llevar a cabo este proyecto. Entre las partidas se incluyen tanto los gastos de personal como los de las herramientas utilizadas, así como otros costes derivados si los hubiera.

Para calcular el coste de los sueldos primero es necesario calcular las horas necesarias para llevar a cabo el proyecto. Un crédito ECTS equivale a 25 horas de trabajo [6]. Como el TFM computa por 12 créditos, se puede decir que son necesarias unas 300 horas en total. Teniendo en cuenta el sueldo por hora que gana un programador medio en España, 10.04€ [7], y que solo ha sido necesario uno, el monto asciende a 3012€.

En cuanto al equipamiento, el único utilizado ha sido un ordenador portátil valorado en 950€ [9] en el momento de la compra. Como todos los programas utilizados son gratuitos, no hay más gastos asociados.

Teniendo en cuenta todo lo anteriormente comentado, el coste total del proyecto son 3962€.

7. Conclusiones y líneas de futuro

7.1. Conclusiones

La principal lección al desarrollar este proyecto es que, crear un videojuego, es una tarea tan gratificante como compleja. Como ya se presuponía, es necesario una gran dedicación y abarcar muchas disciplinas, pero es un arte donde no hay límites. Se pueden hacer grandes guiones, dibujos, músicas, algoritmos y, en definitiva, experiencias.

La planificación, en líneas generales, se ha cumplido. Seguramente la principal causa de esto ha sido plantear un producto sencillo, siendo muy consciente de las capacidades disponibles, tanto temporales como técnicas. En cuanto a la metodología, al tratarse de un desarrollo en solitario, es difícil ser estricto, ya que no hay una cadena de mando ni unas consecuencias directas por variar el rumbo.

Haciendo un repaso a los objetivos propuestos al iniciar el proyecto, la sensación general es positiva.

Los objetivos referentes al juego eran ofrecer una experiencia de juego completa y utilizar recursos propios en todos los apartados. Este último sin duda se ha cumplido, ya que no hay ningún elemento que provenga de otro proveedor. La música, los dibujos y los scripts se han creado específicamente. Respecto a ofrecer una experiencia completa, desde el principio se planteó que el juego iba tener una duración mucho menor a otro tipo de producciones y, aunque el resultado es ciertamente humilde, no se muestra como un producto a medias, por lo que también puede considerarse como un objetivo alcanzado.

Respecto a los que tienen que ver con el cliente, es difícil ser objetivo al tener la experiencia de haber sido el desarrollador. El planteamiento es relativamente original, pero, para ofrecer un producto entretenido y que el usuario quiera finalizar la partida, es necesario hacer modificaciones. Lo principal sería añadir muchas más interacciones, para dar la sensación de que hay multitud de opciones y que, para verlo todo, haya que jugar una gran cantidad de días. Con más tiempo se podría haber obtenido un mejor resultado en este apartado, pero se ha priorizado presentar algo acabado y simple.

Por último, están los objetivos personales. Estos consistían en aplicar los conocimientos adquiridos, adquirir experiencia en la creación de videojuegos y la introducción en la técnica de dibujo píxel art. Los dos primeros son intrínsecos de la naturaleza del proyecto, pero no por ello menos valiosos. Realizar un proyecto al completo es la prueba hay unas competencias alcanzadas, a la vez que asegura una primera experiencia en la creación de un videojuego. El objetivo de aprender a dibujar ha sido el más optimista pero el más entretenido. El resultado es el propio de alguien que acaba de empezar, pero el hecho de adentrarse en una nueva disciplina ha sido muy estimulante y supone el primer paso de algo que, presumiblemente, solo acaba de empezar

7.2. Líneas de futuro

El elemento con más peso en el desarrollo ha sido el sistema de interacciones. Ha motivado la investigación del uso de eventos, la utilización intensiva de scriptable objects o la customización del inspector de Unity. Todo ello con el objetivo de crear una herramienta que permita de forma ágil crear este elemento tan importante de una aventura gráfica. El resultado, por el momento, está lejos de ese objetivo, pero es una primera aproximación. La principal meta a futuro es continuar ese camino para que, finalmente, crear un juego de estas características sea más rápido y solo haya que preocuparse de crear los diálogos, los dibujos o los ítems, pero no de la lógica.

Bibliografía

[1] The top videogames created by a single developer [en línea] [consulta: 20 de marzo de 2022]. Disponible en https://careerkarma.com/blog/games-made-by-one-person/

[2] Personal Extreme Programming [en línea] [consulta: 22 de marzo de 2022]. Disponible en https://www.alpha-epsilon.de/programming/2017/12/06/personal-extreme-programming/

[3] En el principio fue la aventura conversacional [en línea] [consulta 01 de abril de 2022]. Disponible en https://www.xataka.com/literatura-comics-y-juegos/en-el-principio-fue-la-aventura-conversacional

[4] Maniac Mansion [en línea] [consulta: 02 de abril de 2022]. Disponible en: https://es.wikipedia.org/wiki/Maniac_Mansion

[5] Pokémon Rojo y Azul [en línea] [consulta: 02 de abril de 2022]. Disponible en: https://es.wikipedia.org/wiki/Pok%C3%A9mon_rojo_y_Pok%C3%A9mon_azul

[6] ¿A cuántas horas equivalen los créditos universitarios? [en línea] [consulta: 20 de mayo de 2022]. Disponible en: https://www.barcelonaresidencias.com/a-cuantas-horas-equivalen-los-creditos-universitarios-n-155-es

[7] Sueldo del programador de videojuegos en España [en línea] [consulta 20 de mayo de 2022] Disponible en: https://www.jobted.es/salario/programador-videojuegos