Videojuego basado en la generación procedimental de mazmorras

Game Design Document

ÍNDICE DE CONTENIDOS

1.

2. Descripción y objetivos

Con gran inspiración en el juego de 2008, Spelunky, este proyecto se presenta como una implementación 3D de este, pero en un entorno más similar a Rogue (1980) o The Legend Of Zelda (en sus versiones clásicas 2D), donde encontramos habitaciones y pasillos más diferenciados, de construcción humana, en contraste con los entornos del estilo de cuevas naturales de Spelunky. Esto permite implementar trampas específicas para pasillos/habitaciones, que resultarían más difícil de colocar en una disposición irregular de tipo cueva.

El personaje principal se moverá por las distintas habitaciones, evitando trampas, recogiendo tesoros y eliminando enemigos. El objetivo es llegar al final de la mazmorra en el menor tiempo posible y con la mayor cantidad de tesoro.

Para las mazmorras del juego se intentarás cumplir los siguientes objetivos:

- Habitaciones conectadas entre sí por pasillos.
- Se generan procedimentalmente, siendo una mazmorra diferente en cada partida.
- Objetos para mejorar o recuperar la vida.
- Objetos especiales de mazmorra como puedan ser mapas, brújulas, llaves para abrir puertas o cofres...
- Enemigos.
- Trampas como paneles de pinchos, fosos, bolas rodantes...
- Opcionalmente se consideran los siguientes objetivos:
  - Varios tipos de armas, como arrojadizas.
  - Jefes de mazmorra en la habitación final o en alguna habitación protegiendo un tesoro.

Importante notar que tanto la distribución de las habitaciones como de todo lo demás se generará de manera procedimental. Los elementos varían según la dificultad que se le quiera imprimir, que aumenta según se van superando mazmorras.

La vista del juego será 3D, pero con un aspecto pixelado estilo Minecraft, y se considera el uso de sprites para los personajes del juego.

3. Herramientas

3.1. Motor de juego

Debido a que el objetivo principal del juego es demostrar la generación procedimental de las mazmorras se va a utilizar **Unity** en su versión free como **motor de juego** para facilitar el desarrollo. Unity proporciona diversas herramientas integradas con un editor, estas incluyen un motor gráfico, motor de físicas, sistema de audio y sistema de GUI entre otros, suficiente para cubrir las necesidades de desarrollo.

3.2. Diseño de assets

Para los elementos 2D del juego como los botones de menú, títulos y resto de elementos del HUD, así como otros elementos 2D como texturas se utiliza The **Gimp** o **Paint.net**, ambos editores gratuitos. Para el diseño de los assets **3D** se hace uso de **Blender**. Se utilizan recursos externos con licencia libre para todo lo que se considere adecuado.

4. Requisitos hardware y software

Estos son los requisitos que debe cumplir el sistema cliente para ejecutar el juego:

- Windows 7 o superior.
- Teclado y ratón o mando de juego con soporte para XInput.
- Sistemas basados en Linux.
- Tarjetas gráficas HD Intel 4000 o superior, NVIDIA o ATI.

5. Argumento

¡Aventuras, oro, gemas preciosas! Esa es la vida de Tass, el mejor saqueador habido y por haber de las tierras Niilunga.

Niilunga es un reino por el que han pasado numerosos reyes y reinas, culturas y civilizaciones de todos los tipos, y durante su período han dejado abandonados cantidad de tesoros, ocultos en las profundidades de los complejos templos y otros edificios.

Tu eres Tass, perdiste a tus padres cuando eras pequeño y desde entonces has tenido que buscarte la vida por tu cuenta como has podido, ahora dejas la sucia y … ciudad de Corinto y te alejas en las tierras de Niilunga buscando aventura. Pasas los días de ruina en ruina, recogiendo tesoros mientras esquivas trampas y te enfrentas a terribles criaturas. Pero algún día serás lo suficientemente rico como para retirarte para siempre y dedicar el resto de los días a vivir la vida de la mejor de las maneras.

¡Corre a la aventura, recorre las peligrosas mazmorras y recoge todos los tesoros que encuentres!

6.

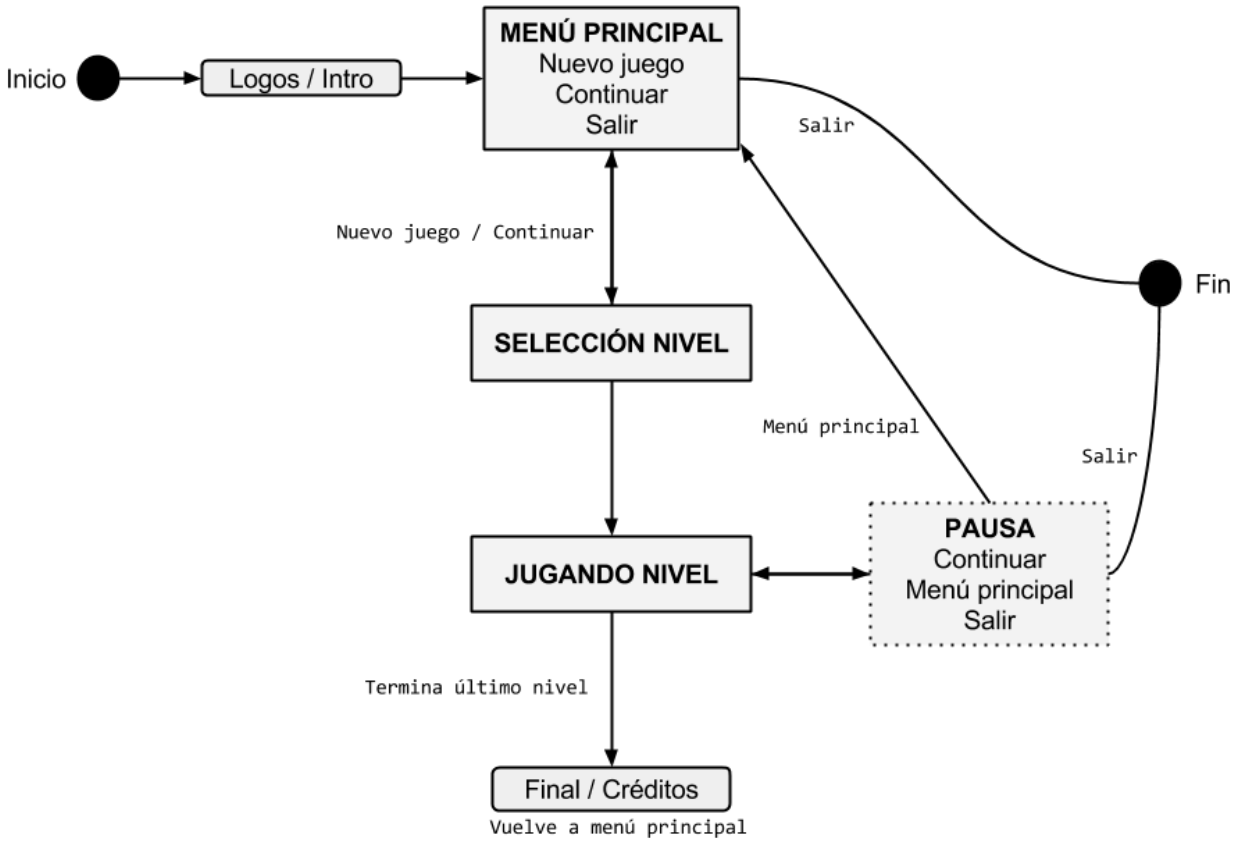
7. Jugabilidad y mecánicas

7.1. Objetivos

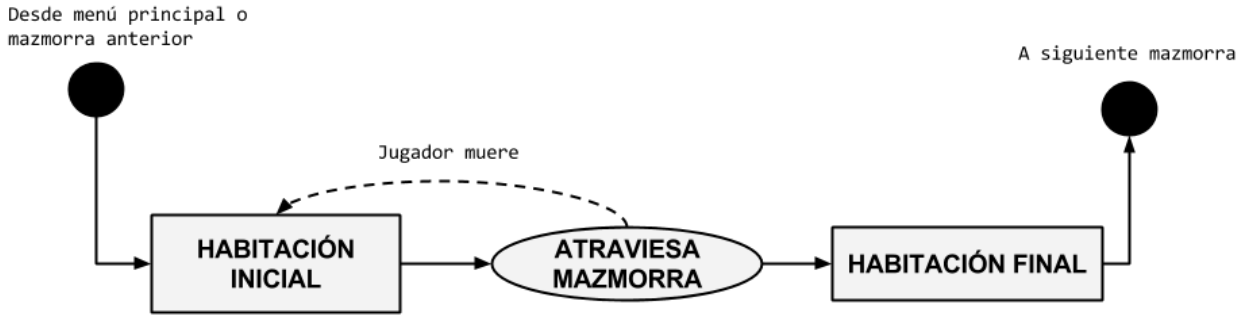
El juego consiste en atravesar un nivel de tipo mazmorra, compuesto por habitaciones y pasillos, desde una entrada hasta una puerta de salida. Por el camino se pueden recoger tesoros, que contribuyen a la puntuación del jugador, junto al tiempo en que se completa el nivel. Durante el transcurso del nivel se plantean trampas que sortear y enemigos contra los que luchar o evitar. El juego es de niveles infinitos y solo termina cuando el jugador pierde todas sus vidas o reinicia un nuevo juego.

7.2. Flujo de juego

Al tratarse de un juego de partidas rápidas, el flujo a través de este debe ser también rápido y muy modular. Queremos que el usuario pueda iniciar o continuar una partida y terminar un nivel en menos de 10-15 minutos o abandonarlo en cualquier momento, volviendo al mismo estado en el que estaba antes de comenzar a jugar ese nivel.



Dentro del nivel/mazmorra el objetivo es llegar a la salida consiguiendo por el camino la mayor cantidad de oro y joyas. No hay puntos de guardado o de control, cuando el jugador muere vuelve al inicio de la mazmorra.



7.3. Controles de jugador

El jugador maneja a un personaje en un entorno tridimensional y podrá realizar distintas acciones:

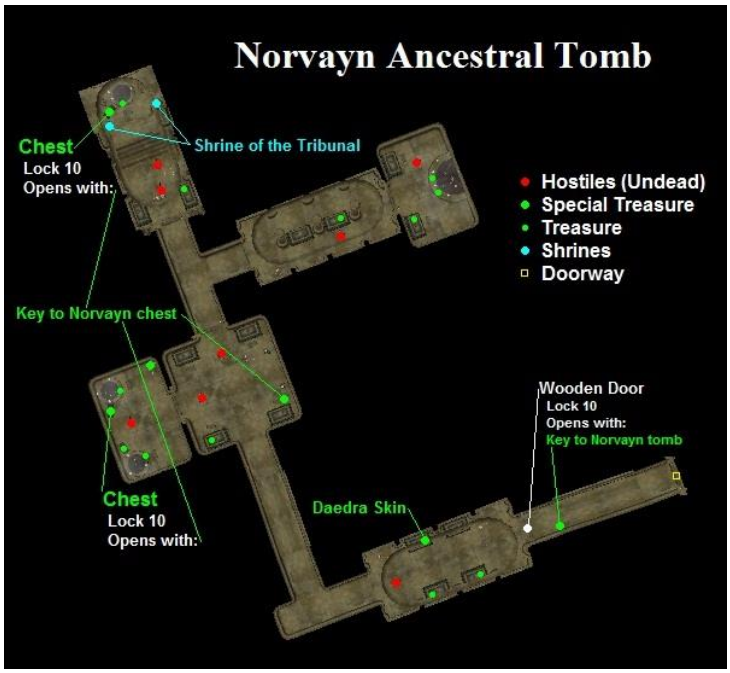
- **Movimiento**: El jugador puede mover en el plano x,z que representa el suelo en todas las direcciones disponibles dentro de este. Se utilizarán las **flechas de teclado** para este movimiento.
- **Correr**: Añade una velocidad extra al **movimiento** normal. Mientras el jugador se mueve se puede correr manteniendo pulsado la tecla **Shift**.
- **Saltar**: Con la tecla **espacio** el jugador puede realizar un salto para sortear trampas, su longitud dependerá de la velocidad de movimiento.

Para el resto de detalles sobre las mismas mazmorras, objetos y enemigos que se podrán encontrar vamos a extendernos un poco más en las siguientes secciones.

8. Mazmorras

Al tratarse de un juego de tipo dungeon crawler de tipo aventuras las referencias más representativas de lo que deseamos las podemos encontrar en la saga de videojuegos The Legend Of Zelda (los juegos clásicos en 2D) o la saga Diablo, juegos completamente diferentes en temática, pero que comparten esa jugabilidad sobre mazmorras de diversos estilos.

Pero no buscamos algo tan homogéneo como podríamos encontrar en el primer juego de la saga Zelda, donde las habitaciones son prácticamente del mismo tamaño y simplemente se sitúan una al lado de la otra. En este caso buscamos algo más desordenado, con pasillos que unan las habitaciones, con formas irregulares, pero sin llegar al aspecto de cueva natural. Algo similar a algunas de las mazmorras que podemos encontrar en Diablo.



Mapa de Diablo donde podemos distinguir claramente un inicio y fin de mazmorra, habitaciones y pasillos que las unen.

Este tipo de distribución nos permite colocar ciertas trampas en pasillos, donde hacen mejor efecto que en habitaciones amplias, y dejar las habitaciones para otro tipo de trampas, enemigos y objetos.

Dejando esto claro, podemos definir el contenido que luego agregaremos a la mazmorra, podremos encontrar:

- **Entrada**. Habitación donde comienza el jugador.
- **Salida**. Habitación donde termina la mazmorra, se debe colocar alejada de la entrada. No tiene por qué ser la última habitación a visitar, sino que el jugador puede escoger terminar la mazmorra o seguir explorando un poco más.
- **Habitación normal**. Casi siempre contendrá algún enemigo y posiblemente trampas, pero en menor medida.
- **Habitación especial**. Son habitaciones que contienen algún objeto especial, como una llave que abre la puerta de otra habitación, o un cofre con tesoro extra o alguna poción.
- **Pasillo**. Une dos habitaciones u otro pasillo. No contiene ningún enemigo pero sí trampas, algunas de estas solo aparecen en pasillos.

8.1. Generación de contenido

El aspecto más importante del juego será la generación procedimental de contenido, y más en concreto de la distribución y aspecto de las habitaciones de las mazmorras, así como de los objetos, trampas y enemigos que vamos a situar en estas.

8.1.1. Habitaciones - Estructura de la mazmorra

Primero debemos generar la estructura de la mazmorra, es decir, decidir las dimensiones de esta y crear las habitaciones y pasillos respetando ciertos parámetros. Estos parámetros dependen del algoritmo que finalmente se seleccione para la generación, se baraja el uso de árboles binarios de partición, el método del autómata celular o simplemente la generación aleatoria de habitaciones sin solapamiento.

Independientemente del algoritmo utilizado, y teniendo como base la rejilla del editor de unity, donde un objeto cubo ocupa una unidad de esta rejilla, establecemos unas medidas estimadas de la mazmorra y habitaciones según ascendemos en niveles.

Las dimensiones no son absolutas, sino que se escoge al azar entre el valor del nivel actual y el valor del siguiente. Por ejemplo, en el nivel 1 sería ancho entre 50 y 80 y alto entre 50 y 80.

1 unidad = 1 unidad/celda de la rejilla de Unity

Las dimensiones no son exactas para cada nivel, sino que se escoge al azar entre el valor del nivel actual y el valor del siguiente. Por ejemplo, en el nivel 1 sería ancho entre 50 y 80 y alto entre 50 y 80.

Nivel	Min. Dimensiones mazmorra	Dimensiones Habitación
1	50x50	15
2	80x80	15
3	80x80	12
4	100x100	15

```
5       105x105                     14
6       110x110                     14
7       115x115                     12
8       120x120                     12
...
```

Estos valores se han decidido después de realizar varias pruebas en Unity con mazmorras creadas manualmente. A partir del nivel 8 los valores serán prácticamente los mismos y la dificultad variará con el contenido de la mazmorra.

Se selecciona una habitación de entrada y una de salida antes de continuar generando el contenido dentro de las habitaciones.

8.1.2. Colocación de objetos

Lo primero después de la creación de las habitaciones es colocar los objetos sobre los que después nos basaremos para las trampas y enemigos. Tenemos distintos tipos de objetos que se explican en su correspondiente sección, pero ya podemos establecer ciertas reglas:

- Nunca aparecen en pasillos.
- Hay objetos en todas las habitaciones, incluidas la entrada y la salida.
- Hay objetos comunes, como monedas de oro, y otros menos comunes, como cofres o llaves.
- Algunos objetos solo aparecen si son requeridos en la mazmorra, como las llaves.
- La generación de cierto tipos de objetos afectará a la generación de trampas y enemigos en esa habitación y posiblemente las habitaciones y pasillos adyacentes. Queremos que el acceso a ciertos objetos especiales sea más difícil.

8.1.3. Trampas

Las trampas son objetos animados distribuidos por toda la mazmorra con cierta aleatoriedad basándonos en los siguientes principios:

- Se pueden generar tanto en habitaciones como en pasillos.
- Dependiendo del tipo de trampa puede que algunas solo sean exclusivas para habitación o para pasillo.
- Si en una habitación podemos encontrar un cofre o un objeto especial, como puede ser una llave, es muy probable que alrededor de este se genera algún tipo de trampa, como los suelos de pinchos.
- No se deberían incluir más de 3 tipos de trampas en una misma habitación o pasillo.
- La cantidad varía dependiendo de la cercanía con la salida y habitaciones con objetos especiales.
- Las trampas no interactúan o dañan a enemigos.

Tales trampas aparecen en pasillos en tal medida, junto a cofres y en habitaciones especiales. Interactúan con otras trampas o con enemigos...

8.1.4. Enemigos

Finalmente se generan enemigos de varios tipos, como se describe más adelante, la colocación de estos es como sigue:

- Siempre aparecen en habitaciones, nunca en pasillos.
- Entre 1 y 5 como máximo por habitación.
- Toman precedencia sobre las trampas, pero pueden aparecer junto a alguna.
- Aparecen en más cantidad cuanto más cerca están de una habitación con un objeto especial o hacia el final de la mazmorra.
- No aparecen en la habitación de entrada pero pueden aparecer en la de salida.

9. Objetos

En general los objetos que podemos encontrar que no sean trampas o decoración de escenario se clasificará en pociones de vida o dinero, pero pueden estar contenidos en

cofres u otros objetos. A continuación listo una posible serie de objetos que se incluyen en el juego:

- **Pociones**. Se encuentran esparcidas por la mazmorra, aunque no son muy comunes, existen pociones de varios efectos. El efecto de las pociones se activan por contacto pero se estudiará el uso de un inventario para almacenarlas y que sea el mismo jugador el que las active manualmente.
  - **Vida**. Recuperan la vida del jugador. Pueden recuperar la vida por completo o solo un corazón.
  - **Invencibilidad**. Hacen que el jugador sea invencible durante cierto período de tiempo. Son poco comunes de encontrar y aparecen en zonas o mazmorras de nivel difícil, donde pueda haber pasillos con varias trampas.
- **Tesoro**. Esto representa los puntos en el juego, y el objetivo es obtener el máximo posible antes de terminar la mazmorra, ya que esto, junto con el tiempo, indicará la puntuación del usuario. Podemos encontrar distintos tipos de tesoro.
  - **Oro (10 pts)**. Se representa mediante una moneda de oro, es el ítem más común del juego pero el que menor valor monetario tiene. Puede que algunos dejen una moneda de oro al morir.
  - **Gemas (50 pts)**. Es más difícil de encontrar y en menor cantidades. Aparecen en ciertas habitaciones apartadas, normalmente junto a otro ítem de tipo poción.
  - **Diamantes (100 pts)**. Los más valiosos del juego y difíciles de encontrar. Pueden aparecer en algún cofre y habitaciones apartadas, pero esto es menos común.
- **Cofre**. Los cofres pueden incluir ítems de tipo pociones o tesoro. Cuando incluyen tesoro es más probable de encontrar gemas que oro o diamantes. Puede que mezcle varios de estos.
- **Llave**. Algunas habitaciones están cerradas con llave. Para ello se debe encontrar una de estas llaves. Solo tienen un uso y habrá tantas llaves como puertas en la mazmorra. Suelen estar protegidas por enemigos o trampas.

10. Trampas

Uno de los aspectos más importantes del juego es el uso de trampas, posicionadas por toda la mazmorra en habitaciones o pasillos en distintas combinaciones como parte de las técnicas de generación procedimental de contenido.

Estas trampas deberán ser sorteadas por el jugador atendiendo a las características de estas, como su tipo, posición o combinación.

10.1. Características

Cada trampa presenta una construcción y comportamiento diferente basándose en diversos aspectos. Además la combinación de distintos tipos de trampas en una habitación o pasillo hace que el jugador deba replantear la estrategia en cada situación a pesar de ya conocer el funcionamiento de estas trampas individualmente.

Pero debemos establecer una serie de premisas y parámetros para colocar las trampas en lugares adecuados y combinarlas, siempre permitiendo que puedan ser superadas sin recibir daño. Algunos de estos requisitos podrían ser:

- **Dificultad de mazmorra.** La dificultad o nivel de la mazmorra
- **Tipo de habitación**. Las habitaciones especiales con llaves u otros ítems especiales contendrán menos carga de trampas, pero los pasillos y habitaciones adyacentes o directamente conectados a estas serán, en contraste, las que más tengan.
- **Dimensiones de la habitación.** Los pasillos y las habitaciones tendrán trampas y combinaciones distintas como veremos en la descripción de las trampas.
- **Cercanía con otras trampas**. Las trampas del mismo tipo normalmente se pueden encadenar o combinar, como colocar varios suelos de pinchos seguidos en un pasillo. En estas combinaciones se deberán ajustar los tiempos de activación de cada una para que puedan ser superables sin que el jugador reciba daño.

10.2. Listado de trampas

Para entender mejor lo expuesto aquí se listan las trampas posibles y sus características y modos de funcionamiento.

10.2.1. Pared de pinchos

Son placas que se posicionan en las paredes y de las que surgen un...

10.2.2. Suelo de pinchos

Similares a las de pared, estas se sitúan en el suelo y contienen unos pinchos más pequeños que igual de dañinos.
A diferencia de las anteriores estas pueden ser automáticas, activándose cada cierto tiempo, o de presión, activándose cuando el jugador pasa por encima. Tanto para uno como para otro modo tiene que haber un tiempo de “recarga”.

10.2.3. Dispensador de dardos/flechas

Se sitúan en la pared y se activan cuando el jugador pasa por delante o automáticamente cada cierto tiempo. En el primer caso también tienen un tiempo de recarga, por lo que si el jugador está delante de estas no se produce un disparo constante, sino que se hará cada 2 segundos.

Estas trampas deben pasarse corriendo o lo más separado posible de estas, ya que los dardos es disparan a gran velocidad.

En los pasillos se pueden combinar junto con paredes de pinchos para mantener al jugador centrado en el corredor de manera que no puede evitarlas simplemente pegándose a la pared contraria.

10.2.4. Bola rodante

Las podemos encontrar exclusivamente en pasillos largos. Se trata de una roca gigante que rueda hacia el jugador normalmente matándolo directamente al contacto. Por ello no son difíciles de evitar y se puede escapar corriendo en dirección contraria y dejando que golpeen alguna pared. Pero pueden combinarse en pasillos que ya tienen otras trampas, obligando al jugador a volver a pasar sobre estas.

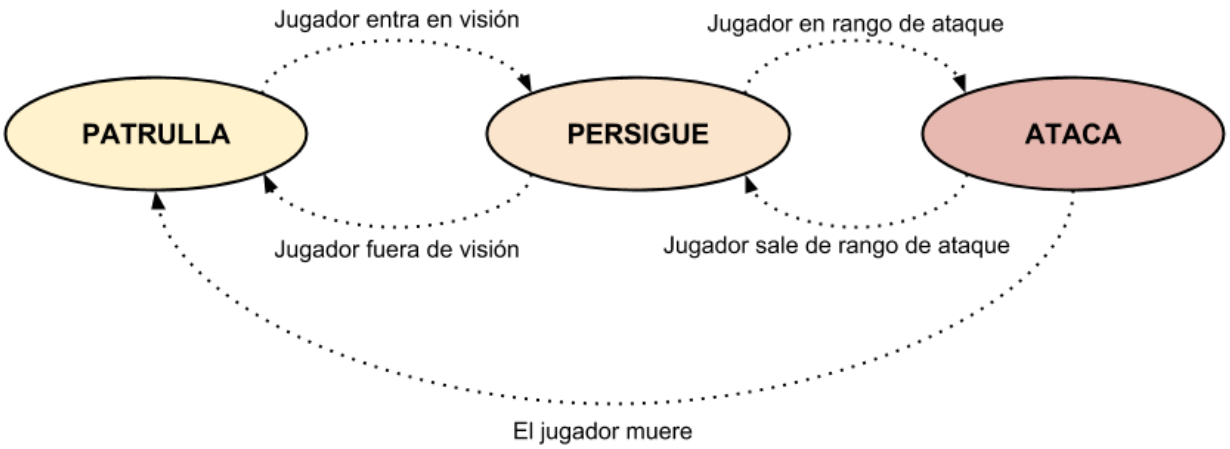
Se activan al pasar por cierta zona del pasillo donde el jugador activa un trigger invisible que hace que la roca se mueva hacia este.

11. Enemigos

11.1. Inteligencia artificial

Los enemigos implementan una inteligencia artificial simple, basadas en estados y círculos de visión o cajas de colisión, según sea el caso, los comportamientos pueden ser los siguientes:

- **Patrullar**: El personaje patrulla una zona, moviéndose en una línea o en un rectángulo.
- **Perseguir**: El personaje persigue al jugador cuando este entra en su círculo de visión. Cuando el jugador se aleja demasiado entonces el enemigo vuelve a su zona de patrulla.
- **Atacar**: Cuando el jugador está suficientemente cerca del personaje, este ataca en la dirección del jugador. Tendrá un tiempo de espera entre ataque y ataque.



En principio estos tres estados se complementarán. Por ejemplo, un enemigo de tipo **cangrejo** patrullará de un lado a otro y cuando jugador entre en su rango de ataque entonces pasará al estado de atacar. Los **goblins**, además de patrullar, podrán perseguir al jugador cuando este entre en su rango de visión y atacará cuando entre en su rango de ataque. Si el jugador se aleja demasiado entonces este vuelve a su zona de patrulla.

12. Aspecto Visual

12.1. General

A pesar de que a lo largo de este documento he nombrado a Zelda y Spelunky como referencia para la jugabilidad y temática de este, el aspecto que vamos a escoger se desarrollará sobre un entorno 3D pero con ciertas peculiaridades. El entorno, como ya he dicho, será 3D, pero los personajes como el jugador y los enemigos serán sprites 2D. A esto se lo conoce como billboards y se suele utilizar para representar vegetación en videojuegos con gran cantidad de estos elementos, ya que en vez de un modelo 3D estamos trabajando sobre un quad mucho más sencillo.

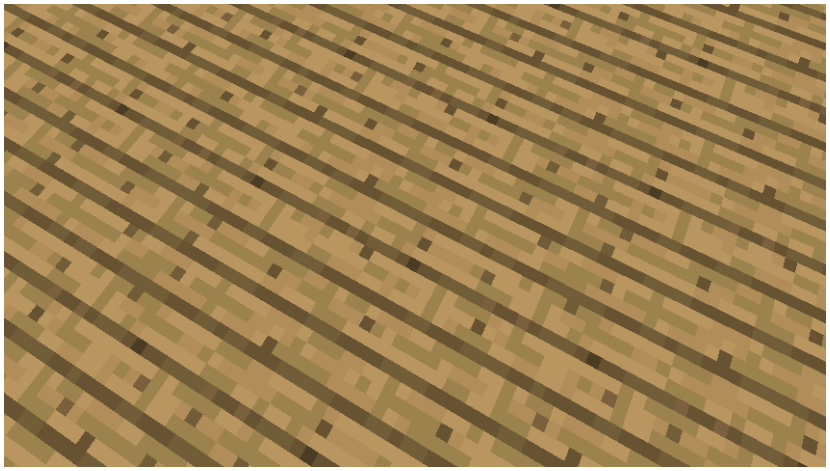
El uso de **billboards** sobre personajes surge en juegos como **Doom**, donde los enemigos y los propios jugadores eran sprites que siempre miraban a la cámara y en algunas ocasiones, para dar la sensación de que se giraban simplemente se cambiaba el sprite, pero siempre eran objetos planos. No solo en Doom, sino que este estilo se ha utilizado en juegos de rol japonés como la saga **Ys**.

Ys Chronicles presenta sprites 2D en un entorno 3D

La elección de este estilo es primordialmente debido a que es mucho más fácil crear o incluso encontrar hojas de sprites que pueda usar en el juego, de buena calidad y que tengan un estilo similar entre sí para la creación de los personajes en comparación con

tener que realizar modelos 3D y animarlos correctamente, lo cual no es el objetivo de este trabajo. Pero por otro lado también está mi interés en utilizar este estilo para darle un toque más de dibujo animado, que sea un juego 3D pero manteniendo el estilo alegre de un juego 2D como pueda ser Spelunky.

Para el resto de elementos 3D vamos a escoger un estilo sencillo, de **baja carga poligonal** para que encaje con el estilo general simple pero alegre. Por esto las **texturas** también tendrán un estilo **Minecraft**, basadas en arte de píxel.

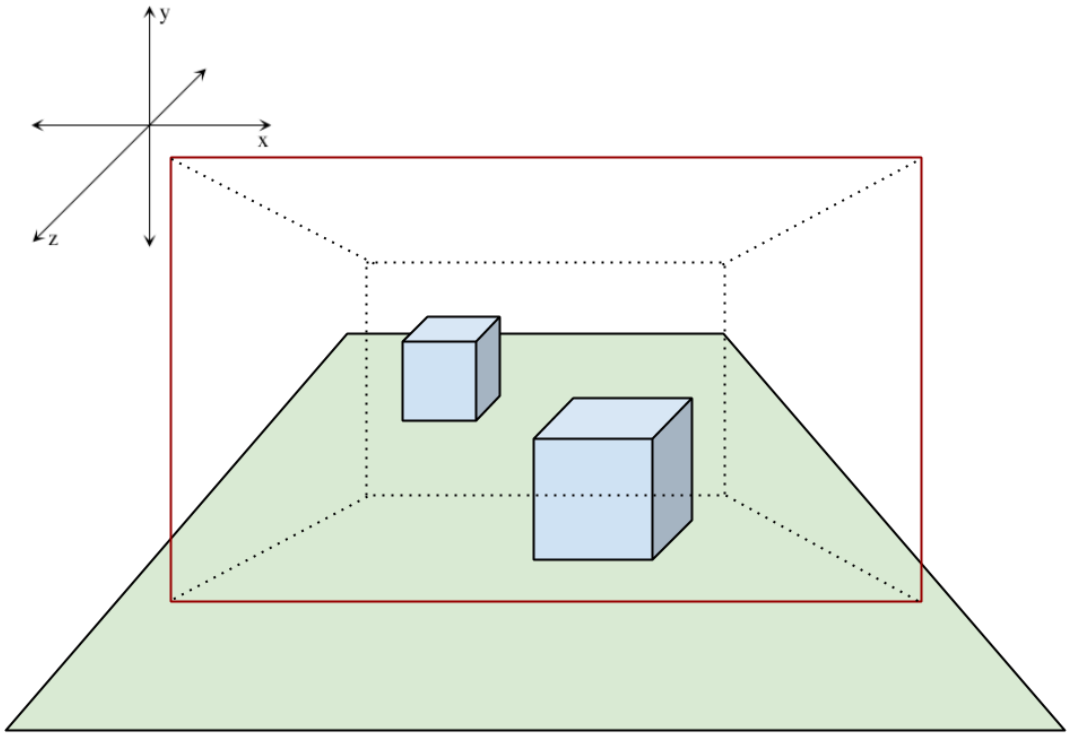


Textura de madera en minecraft, simple pero efectiva

En cuanto al estilo de colores y texturas en sí vamos a hacer referencia, de nuevo, a Minecraft, que tiene un aspecto pixelado, simple pero efectivo. También debemos tener en cuenta que estamos realizando un juego estilo Spelunky, por lo que la temática deberá estar reflejada en estas texturas, con el uso de roca, madera, cuero, moho y otros elementos que representen aspecto natural.

12.2. Perspectiva y cámaras

Escogemos una vista 3D con una cámara con proyección en perspectiva, dando efecto de profundidad ya que es importante poder medir correctamente las distancias para sortear trampas y golpear a enemigos.



La cámara principal seguirá al jugador con un efecto de muelle, con lo que no se desplazará inmediatamente a la posición centrada en el jugador sino que se realizará un movimiento retrasado suavizado mediante un interpolado lineal.

La cámara se situará frente al jugador, o detrás de este, a cierta distancia de separación. Esta distancia se podrá modificar dentro de ciertos límites. También se podrá subir o bajar la vista dentro de ciertos límites pero en ningún caso se podrá girar en torno al eje vertical de manera manual.

Otras cámaras que se puedan incluir serán automáticas para resaltar ciertas situaciones, pero en ningún caso el jugador tiene control sobre estas. Estas situaciones se refieren a pequeñas escenas cinemáticas cuando el nivel comienza y se hace un sobrevuelo sobre el mapa o cuando el nivel termina y se hace un zoom al jugador. En la sección de animaciones de cámara se describen estas situaciones.

Se incluirá una cámara libre a propósitos de testeo, con un control similar al que implementa el mismo editor de Unity.

12.3. Interfaz

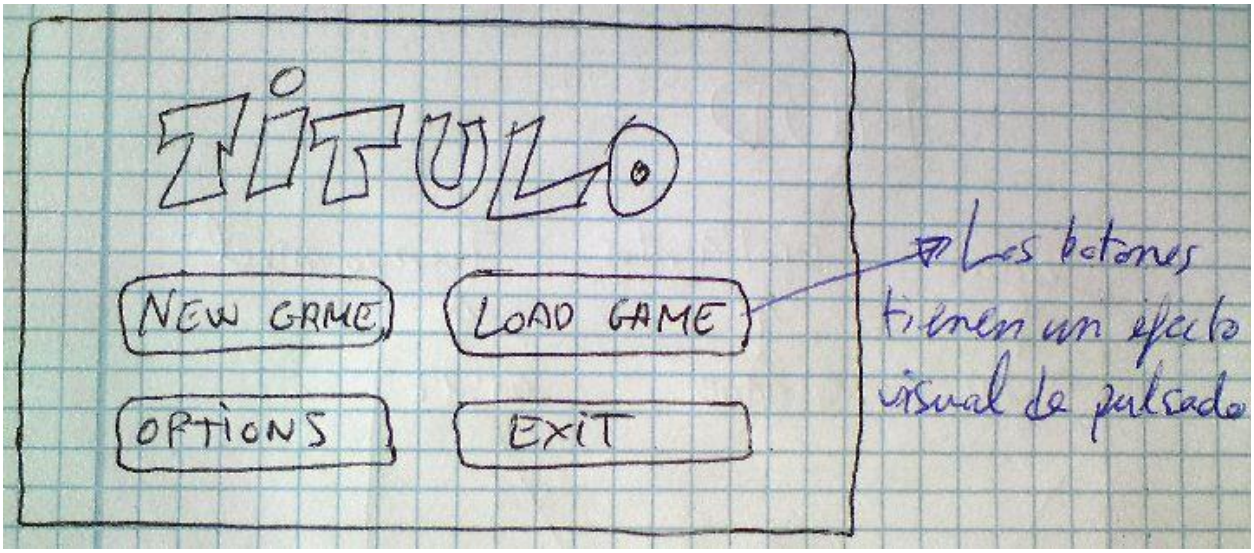
Cuando hablamos de interfaz vamos a describir, por un lado, el sistema de menús para acceder al juego, durante la pausa y en escenas intermedias, como las estadísticas de final de fase, y por otro lado, el sistema de HUD, es decir, la información que se muestra durante la partida indicando principalmente el estado actual del jugador.

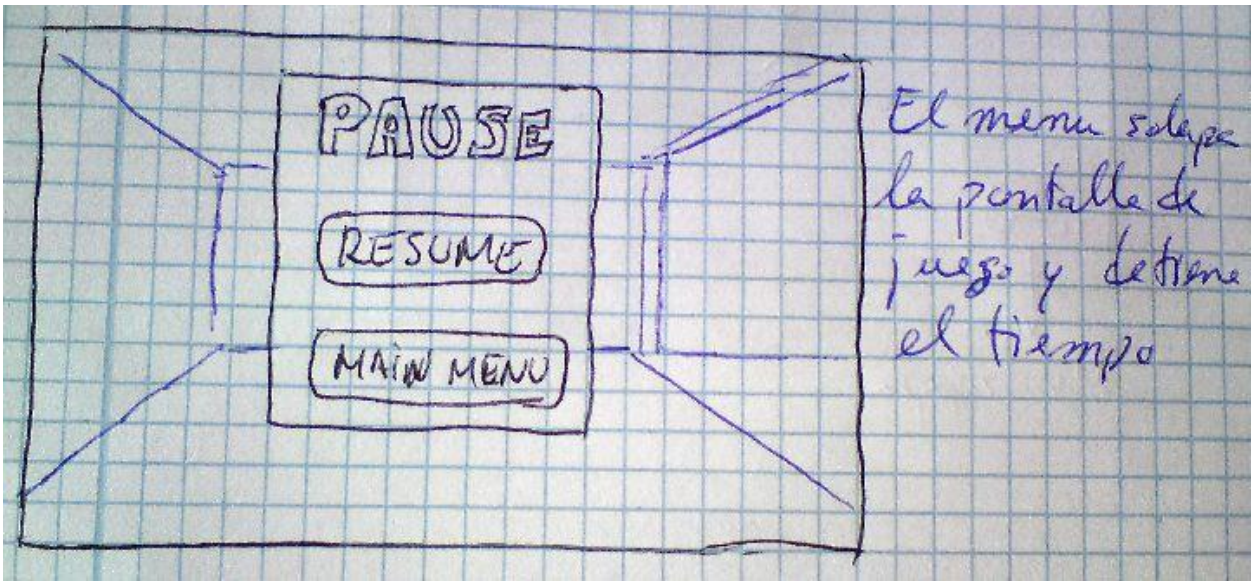
12.3.1. Resoluciones

El juego se puede adaptar a distintas resoluciones y aspectos, pero se desarrollará teniendo en cuenta un **aspecto de 16:9** y una **resolución mínima de 960x540**, por lo que no se asegura que los elementos de la interfaz se adapten correctamente en resoluciones más bajas.

12.3.2. Menús

Como ya hemos visto en la sección donde se explica el flujo de juego, vamos a tener un menú principal para escoger una nueva partida, opciones y tests, además de un menú de pausa. También se mostrará una pantalla al final de cada fase indicando las estadísticas de esta. Estas pantallas compartirán un estilo homogéneo en los botones, tipografías y comportamiento. Los menús podrán ser navegados y activables tanto por ratón como por teclado.

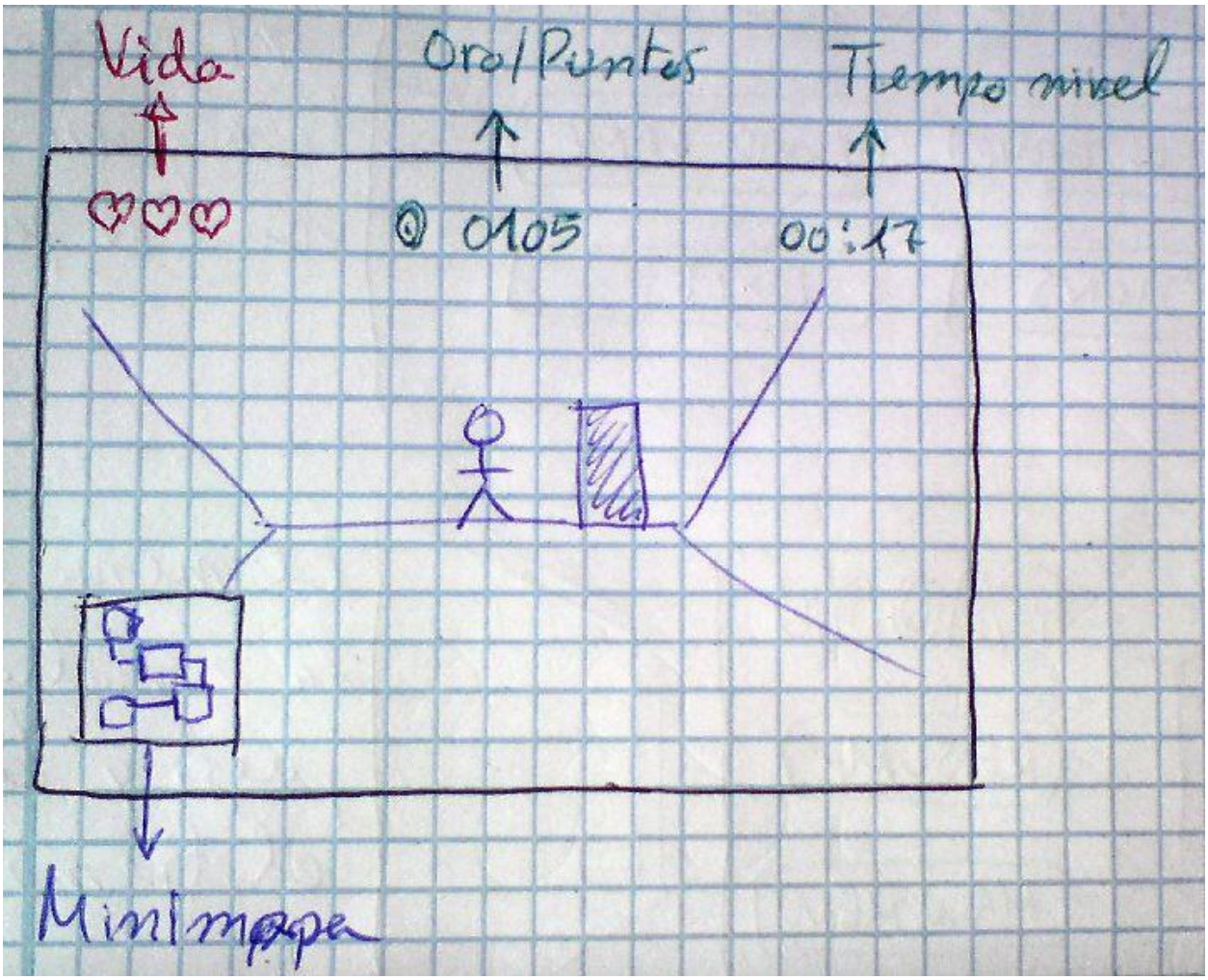




12.3.3. Heads-up display

Durante la partida se mostrará, bordeando la pantalla, cierta información sobre el estado del jugador. Serán elementos no interactivos, pero se actualizarán automáticamente durante la partida. Este HUD contendrá los siguientes elementos:

- **Vida del jugador**. Representada mediante corazones. Indica la salud del jugador, la cantidad de golpes que puede recibir. Se actualizará cada vez que este reciba daño o se cure mediante algún objeto.
- **Oro/Puntos**. Cantidad de oro, gemas y otras joyas recogidas durante el nivel. Se mostrará el valor total de la suma de estas.
- **Tiempo de juego**. Tiempo que se lleva jugando ese nivel. En principio simplemente se utilizará para propósitos de speedruns. Pero es considerará aplicar recompensas (más puntos/fortunas) para los mejores tiempos.
- **Minimapa**. Muestra, a vista de pájaro, la porción de la mazmorra descubierta.



12.4. Animaciones

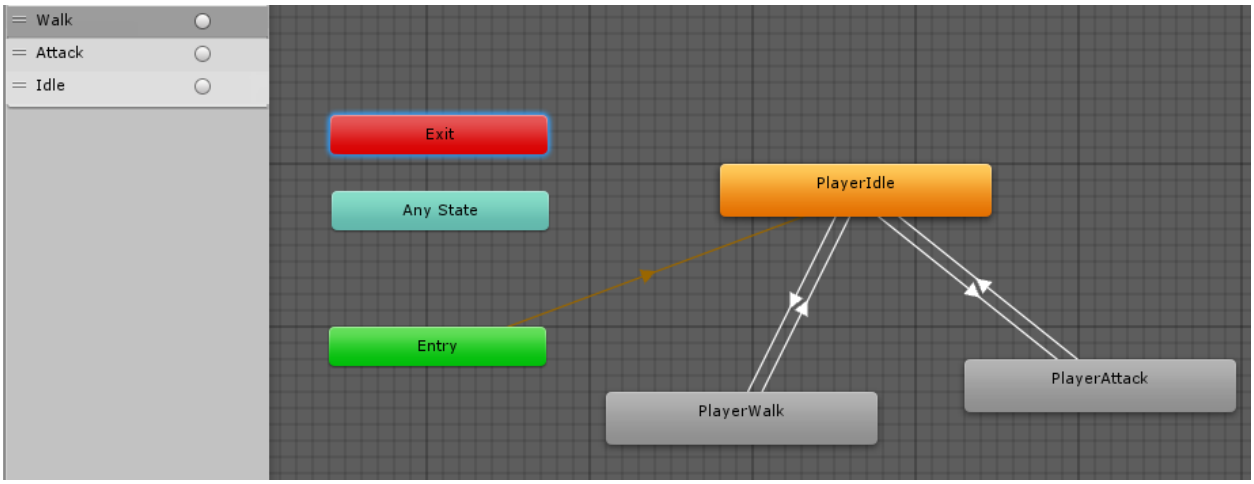
12.4.1. Objetos 3D

Los objetos interactivos 3D como puedan ser cofres o trampas serán animados, cuando sea posible, desde el mismo editor Unity 3D, ya que permite más flexibilidad y rapidez a la hora de ajustar tiempos. Para las trampas esto puede ser sencillo ya que, por ejemplo, las trampas de pinchos simplemente se basan en animar la posición en un eje. Por otro lado trampas como el disparador de dardos o la roca rodante tendrán movimientos basados en físicas, es decir, no serán animaciones en sí sino que se les aplicará una fuerza física para definir su comportamiento.

Otros objetos como cofres en principio serán animados en Blender ya que se trata de objetos compuestos donde se anima la rotación utilizando un punto de anclado concreto. Esto hace que sea más conveniente realizar la animación previamente en Blender e importarla a Unity.

12.4.2. Personajes 2D

Para los personajes ya hemos definido que tendrán un aspecto 2D y se utilizarán sprites para representarlos en pantalla. Por ello se utilizar sprite sheets conteniendo todas estas animaciones, pero se tendrán que editar desde Unity, que proporciona un módulo de animación con el que podemos crear las distintas acciones por separado y luego utilizar un gestor basado en nodos para activar cada una de estas.



El animator de Unity permite gestionar los estados mediante triggers

12.4.3. Cámara / Escenas automáticas

Aunque en principio esto es algo extra, que se agregará si hay tiempo, ya que se trata de algo más estético, se agregarán diversas animaciones a la cámara en ciertas ocasiones:

- El jugador recibe daño, la cámara tiembla.
- Se activa la trampa de roca, la cámara tiembla. Este ya más complicado, ya que se debe hacer patente que en realidad es el mismo escenario el que tiembla.

- El jugador abre un cofre, la cámara hace un acercamiento para ver mejor que se ha encontrado en este. Durante esta escena se debería detener el tiempo para el resto de elementos del juego, o al menos los enemigos, ya que no queremos que el jugador reciba daño durante esta animación.
- El jugador alcanza el final del nivel, la cámara se acerca a este. En esta misma escena se muestran las estadísticas del nivel y las opciones de continuar al siguiente o salir.

13. Audio

13.1. Sonido

Durante el juego podemos escuchar los sonidos básicos del personaje que maneja el **jugador** como:

- **Salto**. Hace un pequeño gruñido o se escucha un sonido al estilo Super Mario Bros que representa el sonido del personaje separándose del suelo. Si al caer se utiliza un sonido deberá ser algo muy sutil, un sonido seco.
- **Golpea**. Se escucha el sonido del arma moviéndose. Además puede que se agregue un sonido de esfuerzo del mismo personaje. Cuando se produce contacto del arma contra un enemigo se escucha un sonido seco de golpe. En principio no se usará sonido al golpear otros objetos, pero se puede hacer una excepción si se necesita acentuar el sonido al abrir un cofre de un golpe.
- **Moverse y correr**. En principio se probará con sonido solo al correr, se puede probar con un sonido constante de pasos, muy sutil, o simplemente al arrancar el sprint.
- **Daño**. El personaje lanza un quejido al ser golpeado. Si el golpe produce la muerte entonces el sonido es más largo y dramático.

Los **enemigos** también pueden realizar sonidos al golpear, ya sea emitiendo un gruñido como con algún sonido del arma que utilicen. Algunos enemigos como los cangrejos o las arañas puede realizar un pequeño sonido de pasos al moverse, al descolgarse o realizar otras animaciones.

Otros sonidos provendrán de **objetos**, ya sean trampas como ítems para el usuario. Los sonidos deberán ser simples, en lo posible, y representar claramente el objeto y acción que se está produciendo.

Los botones de **menú** pueden incluir sonido al ser pulsados o cuando el ratón pasa por encima. Se tratará de sonidos de roca o madera moviéndose, como si se tratara de un mecanismo manual.

Durante la pausa durante el juego se reducirá el sonido del nivel, como la música, y se podrá escuchar claramente el sonido de los elementos del menú.

La activación/desactivación de menús puede incluir algún sonido simple indicando que se ha producido una acción.

Acceder a un nivel o comenzar un juego nuevo también podrá incluir un sonido de transición o pequeño clip de música.

13.2. Música

Se trata de un juego de espíritu arcade, por lo que la música deberá tener cierta marcha, incitando al jugador a estar atento, pero sin resultar frenética, un sonido **funky**. Tanto las canciones de los menús como las de mazmorras tendrán un estilo similar, intentando lograr una homogeneidad entre el cambio de escenas.

Los instrumentos son de estilo electrónico, recordando en cierta manera a la música chip, pero con un estilo más moderno.

Para establecer unas referencias de manera más clara podemos dirigirnos a los siguientes enlaces:

Remix de la canción original de Spelunker

Banda sonora de Spelunky

14. Planificación

Finalmente expongo la planificación para el desarrollo del mismo videojuego. Mi intención aquí no es entrar en detalle con respecto a las fechas concretas y tampoco incluir la planificación de otros aspectos del estudio del trabajo de fin de grado, sino simplemente del orden de implementación de las características del videojuego expuestas en este documento de diseño.

Puntos del videojuego y orden de implementación.

14.1. Etapa 1

- Implementación del algoritmo de creación de habitaciones y pasillos. Pruebas con distintos parámetros y ajustes según el nivel de la mazmorra.
- Implementación del jugador, movimientos sobre un escenario creado manualmente para realizar pruebas los controles.
- Creación de assets, como el sprite del jugador y objetos.

14.2. Etapa 2

- Creación de sprites de enemigos.
- Creación de los modelos para las trampas y animación de estas.
- Implementación de la IA de los enemigos. Pruebas sobre un escenario creado manualmente.
- Implementación de la lógica de los objetos. El jugador interactúa con estos.
- Creación del flujo de pantallas, intersección entre estas y menús.

14.3. Etapa 3

- Implementación de la lógica de las trampas e interacción con el usuario.

- Estudio y desarrollo de la generación de objetos, trampas y enemigos sobre el escenario generado procedimentalmente.
- Integración final, pruebas, corrección de bugs y ajustes finales.