Institut Puig Castellar
Santa Coloma de Gramenet

The Chronicles of Xiao

(Proyecto de desarrollo)

CFGS Desarrollo de Aplicaciones Multiplataforma

ESP

Resumen del proyecto

Este proyecto trata sobre un juego, el cual será un RPG Aventura, este podrá almacenar partidas de los usuarios con el progreso que estos usuarios tengan.

El equipo quiere crear una primera versión de un juego RPG con movimientos, 2 tipos diferentes de enemigos. Este juego se prevé que tendrá la calificación Pegi 3.

Se usará una metodología en parte de investigación y otra parte de implementación, ya que el equipo tratará de investigar cómo funciona Godot como framework y Godot Scripts como herramienta de desarrollo, y en la parte de implementación, el equipo construirá el videojuego.

El videojuego contará con gestión de partidas en la cual los usuarios podrán seleccionar o borrar una partida y para esto usaremos dos tipos de metodología como son la investigación y la implementación.

Palabras clave

➢ Videojuego
➢ Godot
➢ Godot Script
➢ Partidas
➢ Progreso
➢ PEGI 3

EN

Project summary

This project is about a game, which will be an Adventure RPG, this will be able to store user games with the progress that these users have.

The team wants to create a first version of an RPG game with movements, 2 different types of enemies. This game is anticipated to have a Pegi 3 rating.

A methodology that is part research and part implementation will be used, since the team will try to investigate how Godot works as a framework and Godot Scripts as a development tool, and in the implementation part, the team will build the video game.

The video game will have game management in which users can select or delete a game and for this we will use two types of methodology such as research and implementation.

Keywords

➢ Game
➢ Godot
➢ Godot Script
➢ Game Management
➢ Game Progress
➢ PEGI 3

Índice

1. Introducción

1.1 Contexto y justificación

El equipo va ha realizar un videojuego, el cual será un RPG de aventura inspirado en The Legend of Zelda de la Gameboy. El videojuego, en primer lugar será para la plataforma de PC. Se desarrollará con un entorno gráfico utilizando Godot Engine 3.0.

La creación de un videojuego es un tema relevante para el equipo. Desarrollar una aplicación de la que tienes una idea clara junto con la actitud positiva, provocará un desarrollo satisfactorio para todos. Para realizar este proyecto hemos acordado aprender un nuevo lenguaje como lo es GDScript que será utilizado más adelante en el entorno gráfico Godot.

Al completar el trabajo, queremos un videojuego confortable para cualquier tipo de usuario, y al mismo tiempo poder presenciar como damos por finalizados los objetivos marcados al principio del proyecto de desarrollo en equipo.

El videojuego está catalogado como un RPG de aventura, donde se tratará de crear una gestión de partidas, guardando la partida, borrándola o seleccionando la partida que quieras continuar. Animaciones del jugador, dónde cada movimiento que haga el jugador realizará una animación en pantalla. Gestor de colisiones, donde el jugador chocará con árboles, paredes, etc. También contará con 2 enemigos clásicos, como son los murciélagos y los slimes. Contaremos también con un inventario, donde el jugador almacenará todos los objetos que encuentre por el mundo. Interacciones con NPC’s, donde recibirá pistas sobre las misiones que deba realizar y por último tendrá un cambio de escena, donde el jugador cambiará de el mapa principal a una segunda escena donde continuar con las misiones.

1.2 Objetivos

A la hora de cumplir con éxito el trabajo, el equipo quiere conseguir los siguientes apartados. Son los objetivos para finalizar el proyecto.

- Conseguir que el juego funcione de forma fluida.
- Implementar diferentes escenas
- Gestionar las partidas
- El jugador pueda moverse libremente por el mapa
- Interactuar con NPC's
- Destruir enemigos
- Gestionar inventario

Objetivos específicos

Los objetivos indispensables para nuestro juego son los siguientes:

- **Gestión de partidas**: Que el usuario que esté jugando al videojuego pueda crear, borrar o seleccionar una partida.

- **Diseño de mundo**: Crear las colisiones de todos los objetos que puedas encontrar por el mapa.

- **Enemigos:** Estos personajes son con los que el personaje principal peleará a lo largo de la historia, con la posibilidad de que al morir le den al personaje principal algún objeto aleatorio.

- **Movimientos del personaje principal**: Que el personaje que controla el jugador pueda moverse en todas las direcciones.

- **Animaciones de sprites**: Que por cada movimiento o acción que quiera hacer el jugador con el personaje principal visualicemos su movimiento mediante estos sprites.

- **NPC**: Personajes que están en el mapa pero que no son controlados por el jugador. El jugador podrá interactuar con ellos para conversar o conseguir algún objeto nuevo.

- **Cambio de escenas**: Cambio entre los diferentes escenarios del videojuego.

- **Gestión de inventario**: El jugador dispondrá de un inventario del personaje principal donde almacenará objetos que consiga durante la partida.

- **Minimapa:** Crear un minimapa en el que nos aparecerá la posición de los enemigos y los cofres que haya en el mapa, respecto a la posición del jugador.

1.3 Estrategia y planificación del trabajo

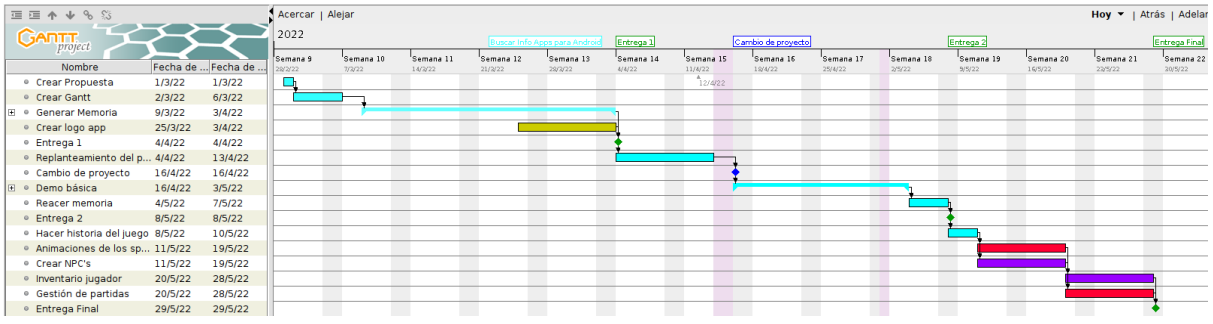
El proyecto lo realizamos un equipo de dos personas, donde la gestión será primordial para cumplir los objetivos y fecha de la entrega. En primer lugar el equipo hará una serie de pruebas para familiarizarnos con Godot.

Una vez el equipo haya realizado pruebas de aprendizaje, y con la idea de proyecto ya confirmada, el equipo realizará una serie de diagramas para tratar de organizar, orientar y preparar la organización de tareas.

En un comienzo uno de nosotros trabajará en el diseño, donde incluye todos los sprites del personaje principal, los enemigos, NPC's, menú principal y pantalla de game over mientras que el otro integrante del equipo inicia el desarrollo del videojuego. Una vez terminada la parte de diseño, se incorporará al desarrollo del videojuego junto al otro integrante del equipo.

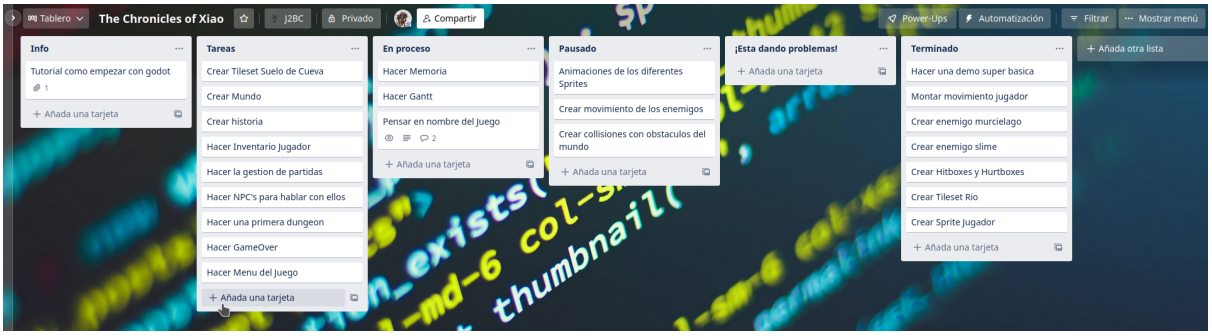
1.4 Metodología de trabajo

Primeramente se ha decidido utilizar Gantt Project, para hacernos una idea por encima de las tareas a realizar.



(Aquí se muestra cerrado para ver más, dejamos aquí el archivo gant)

Más adelante trabajaremos con la plataforma Trello, ya que, permite hacer modificaciones al instante y más rápidas, además de marcar Sprints y quitarnos mucho trabajo de edición.



1.5 Estudio económico y presupuesto

Puestos	Sueldo €/hora	Nº Trabajadores	Horas Aproximadas
Programador Jr	10,05	2	110
Diseñador Jr	9,90	1	20
Tester	11,43	2	30
Redactor	10,19	2	40
Total: 3910,00€			

- **Programador Jr:** El equipo precisa de dos programadores Jr para el desarrollo del videojuego. Según Indeed, un programador Jr en España, tiene un salario base promedio de 10,05€ la hora. El desarrollo de nuestro videojuego ocupa un total de 110 horas, por lo que tendríamos un total de 2211€ que tendríamos que añadir al total del presupuesto de gastos.

- **Diseñador Jr:** El grupo va necesita contar con un diseñador Jr, para crear todos los sprites necesarios para nuestro videojuego. Según Indeed, un diseñador Jr en España tiene un salario base promedio de 9,90€ la hora. El diseño de todos los sprites que necesitamos en nuestro videojuego ocupa un total de 20 horas, por lo que tendríamos un total de 198€ en gastos para el diseño del videojuego.

- **Tester:** El equipo, una vez finalizado el desarrollo del videojuego, requerirá de dos testers para comprobar que todos los aspectos del videojuego funcionen correctamente y esté libre de bugs. El testeo de nuestro videojuego ocupará un total de 30 horas por cada tester a lo largo del proyecto, por lo que esta parte del proyecto, conociendo según Indeed que el salario por hora promedio en España es de 11,43€, tendríamos un total de 685,8€.

- **Redactor:** El equipo, requerirá de dos redactores para narrar la historia de la que tratará nuestro videojuego RPG. Esta redacción ocupará un total de 40 horas a lo largo del proyecto, dado que puede haber correcciones o cambios en ella. El salario por hora promedio de un redactor en España según Indeed es de 10,19€ la hora, por lo que el total será de 815,2€.

1.6 Software utilizado

Estas son las aplicaciones y el software utilizados para el proyecto:

- **Ubuntu 22.04 LTS (José Mª Becerra Marin):** Ubuntu és un sistema operativo de código abierto. És una distribución de Linux basada en la arquitectura de Debian. El desarrollo de Ubuntu está liderado por Canonical, una empresa fundada en el Reino Unido. El proyecto Ubuntu está comprometido públicamente, recomienda a las persona que utilizan programas de software libre, estudiar cómo funciona y mejorarlo.

- **Kubuntu 22.04 LTS (Juan Carlos Carretero Roldán):** Kubuntu es un sistema operativo de código abierto construido en base al núcleo Linux. Kubuntu es una distribución Linux que utiliza KDE Plasma como entorno de escritorio. Es desarrollado por Blue Systems y sus colaboradores.

- **Windows 10/11:** Windows es el sistema operativo más utilizado en el mundo. Este es desarrollado por Microsoft como parte de la familia de sistemas operativos Windows NT.

- **Google Drive:** Es un lugar en el que se puede almacenar y acceder a todos tus archivos, donde el usuario dispone de cierta capacidad de almacenamiento de forma gratuita.
  El equipo ha decidido utilizar Drive, ya que es un entorno con el que está familiarizado, y gracias a tener cuenta del Instituto ha podido disfrutar de almacenamiento ilimitado y de poder compartir entre todos los integrantes todo el contenido del proyecto.

➢ **Discord:** Es una plataforma digital que se utiliza con acceso a internet y permite realizar reuniones mediante servidores o salas. De forma totalmente gratuita.
El equipo ha decidido utilizar esta plataforma, para que a la hora de comunicarse entre los desarrolladores sea de forma más rápida y clara, gracias a las funciones que ofrece la plataforma.

➢ **Gantt Project:** Es una aplicación de escritorio multiplataforma para la programación y gestión de proyectos. Se ejecuta en Windows, Linux y MacOs, es libre y su código es opensource.
El equipo ha decidido utilizar esta aplicación, para hacerse una primera idea de cómo organizar las diferentes tareas en el tiempo.

➢ **Trello:** Es una herramienta flexible para la gestión del trabajo, con la que los equipos pueden diseñar planes, colaborar en proyectos, organizar flujos de trabajo y hacer un seguimiento del progreso de una manera visual y productiva.
Una vez hecha la primera organización de las diferentes tareas, el equipo ha decidido utilizar esta herramienta para detallar más profundamente las tareas a realizar, ya que es totalmente gratuita y el equipo ya tenía conocimientos de su uso.

➢ **Godot Engine:** Es un motor gráfico de desarrollo de videojuegos, multiplataforma y de código abierto. Permite desarrollar videojuegos en 2D y 3D mediante un sistema jerárquico de nodos y escenas. Incluye herramientas necesarias para el desarrollo de manera centralizada y visual. El programa está disponible para Windows, MacOs y Linux y puede exportar videojuegos para dispositivos móviles. Su principal lenguaje de programación es el GDScript, un lenguaje desarrollado específicamente para este motor.
El equipo ha decidido utilizar este motor gráfico y su lenguaje de programación, con el fin de aprender un nueva metodología de programación, desconocida hasta el momento para el equipo.

➢ **Pixelorama:** Es un editor de imágenes que se enfoca en el pixel art. Ofrece herramientas para el pincelado y coloreado de las cuadrículas. También duplica sectores, mide ángulos y gestiona capas. El equipo de diseño ha decidido utilizar este editor de imágenes, ya que es muy intuitivo a la hora de elegir la herramienta o función necesaria para el caso que necesitara.

- **Github:** GitHub es una plataforma de desarrollo colaborativo ideal para alojar proyectos utilizando el sistema de control de versiones Git nombrado anteriormente. Se podría definir de forma más casual como un sitio web y un servicio en la nube que ayuda a los desarrolladores a almacenar y administrar su código, al igual que llevar un registro y control de cualquier cambio sobre este código gracias a Git.
  El equipo de desarrolladores ha podido trabajar de forma segura a través de las opciones que permite este sistema, como son las ramificaciones, que permiten la división de trabajo de forma óptima para evitar borrar o modificar el trabajo realizado por otro desarrollador.

- **Modelio:** Modelio es una herramienta Open-Source desarrollada por Modeliosoft utilizada para la creación de diagramas UML.
  Los desarrolladores se han decantado por esta herramienta, ya que están familiarizados con la misma, al trabajar con ella estos últimos meses y conocer las opciones y elementos que aporta.

2. Descripción del proyecto

2.1 Análisis de requisitos

- **Que funcione correctamente:** El equipo quiere conseguir que el videojuego funcione de forma correcta y fluida.

- **Poder gestionar las partidas del jugador:** El usuario que esté jugando al videojuego pueda crear, borrar o seleccionar una partida.

- **Poder Interactuar con NPC's:** El equipo quiere crear varios personajes que aparecen en ciertos puntos del mapa, pero que no son controlados por el jugador. El jugador podrá interactuar con ellos para conversar o conseguir algún objeto nuevo.

- **Poder moverte por el mapa:** El equipo quiere que el personaje que controla el jugador pueda moverse en todas las direcciones.

- **Poder atacar y hacer volteretas:** Por cada movimiento o acción que quiera hacer el jugador con el personaje principal visualicemos su movimiento mediante estos sprites.

Requisitos funcionales

- **Menú Inicio del Juego:** El equipo quiere desarrollar un menú principal, donde el usuario del videojuego pueda administrar sus partidas, como podría ser "Crear una nueva partida", "Cargar una partida" o "Borrar una partida".

- **Combate y estrategia:** El usuario, mediante algunas teclas del teclado, podrá golpear a los enemigos que vaya encontrando a medida que avance en el videojuego.

- **Interacciones entre NPC y Jugador:** El usuario podrá interactuar mediante diálogo, con algunos personajes dentro del videojuego. Estos personajes no son jugables por el usuario, ya que son controlados por el mismo juego.

- **Función de Pausa:** El jugador podrá pausar el juego en todo momento mediante una tecla del teclado. Pausando todos los estados dentro del videojuego, como pueden ser los enemigos.

- **Función de Inventario:** Mediante una tecla del teclado, el jugador podrá abrir el inventario del personaje principal, pudiendo acceder de esa forma a todos los items almacenados en el transcurso de su aventura. Accediendo también a las opciones de guardado y salir de la partida.

- **Minimapa:** Mientras el usuario esté disfrutando de su partida en el videojuego, en la parte inferior derecha de la pantalla, podrá ver un minimapa que le mostrará la posición exacta de los enemigos o los cofres que haya en el mapa respecto a él. Además podrá hacer zoom en este minimapa, para poder acercar o alejar la vista.

Requisitos no funcionales

- **Jugabilidad intuitiva:** Poder ofrecer un sistema de juego fácil de aprender para el usuario.

- **Diseño atractivo:** Conseguir un diseño que guste al usuario, un estilo pixelado, con el que los usuarios potenciales de la aplicación se sientan a gusto.

2.2 Tecnologías

Comparativa de las tecnologías valoradas

El equipo ha estado mirando varias tecnologías como lo son Angular, Android Studio y React Native para la primera opción de proyecto de la aplicación. Al final, el equipo ha decidido cambiar de proyecto y para esto ha decidido usar Godot Engine como tecnología para el desarrollo de un videojuego RPG. También, como no, como vamos a trabajar en equipo se ha valorado varias metodologías de control de versiones como lo son Git o Subversion. En este caso el equipo ha decidido usar Github Desktop.

Tecnologías escogidas

El equipo ha escogido como tecnología para el desarrollo del videojuego, utilizar Godot Engine, ya que el grupo ha pensado en crear un videojuego en 2D, y Godot tiene buenas prestaciones para dicho desarrollo. Además de guardar todo el trabajo en GitHub(con Git) ya que es a lo que estamos acostumbrados.

2.3 Estructura del proyecto

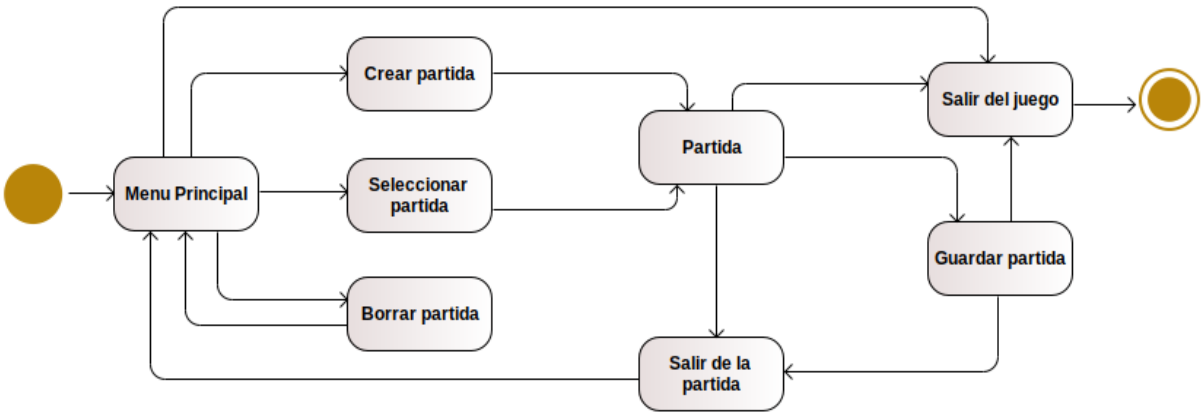
The Chronicles of Xiao tiene una arquitectura de cliente, en la que todas las interacciones que tenga el usuario dentro del juego, serán almacenadas y controladas por el mismo cliente.

2.4 Descripción de los componentes

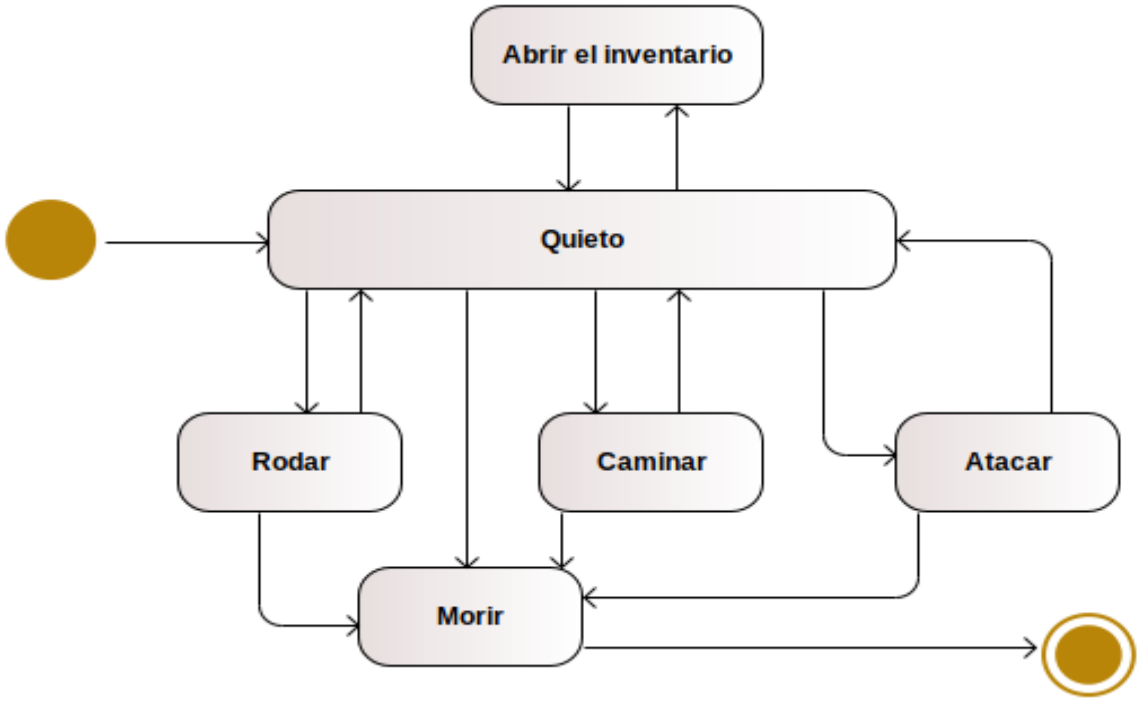
Diagramas

Para representar el funcionamiento de algunos aspectos de nuestro videojuego, hemos hecho los siguientes diagramas de estados para detallar de una forma más clara, los estados por los que pueden pasar.

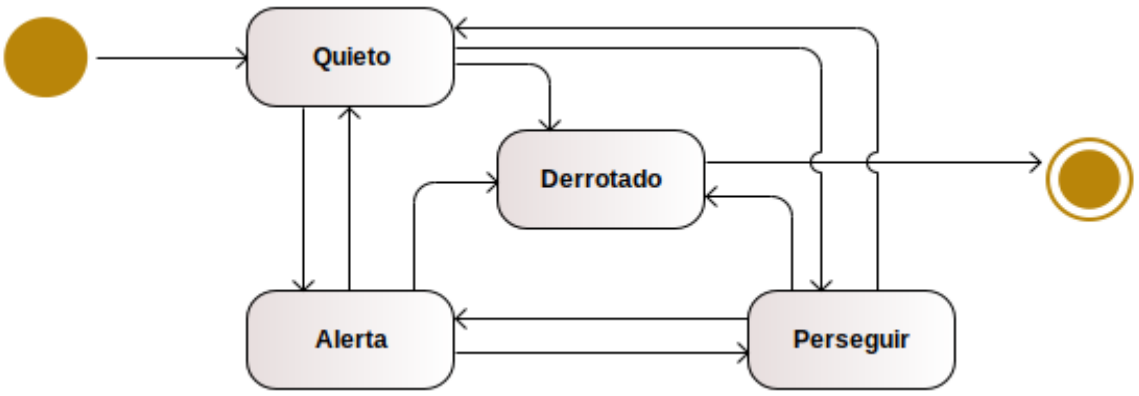
- **Partida:** El siguiente diagrama nos muestra las diferentes funciones que dispone el juego. En primer lugar encontramos el “Menú principal”, una vez allí, nos encontraremos con 4 opciones. La primera, “Crear partida”, en la cual comenzaremos una nueva partida desde 0. La segunda, “Seleccionar partida”, si tenemos una o varias partidas empezadas, podemos elegir qué partida queremos continuar desde el punto que lo habíamos dejado. La tercera, en la que podemos eliminar la partida que ya no queramos continuar. Como cuarta opción tenemos la de salir del juego, en la que cerraremos el juego. Una vez hayamos creado o seleccionado una partida y estemos dentro de ella, podemos “salir de la partida”, que nos mandará al menú principal, “Guardar partida”, que nos guardará todo el progreso realizado durante la partida o “Salir del juego”, que como hemos dicho antes, cerraremos el juego.



- **Player:** En este diagrama definimos los estados en los que el personaje principal puede estar o las funciones que puede realizar. El primer estado del personaje es el de “Quieto”, en el que si no pulsamos ninguna tecla el personaje permanecerá quieto en la escena. A partir de aquí el jugador puede entrar en diferentes estados, según le indiquemos mediante el teclado. Mediante la tecla “I”, el jugador podrá abrir el inventario del personaje, donde encontraremos diferentes objetos. Con la tecla “K” el personaje podrá rodar por el mapa. Con las teclas “A, S, D o W”, el personaje podrá moverse en la dirección que le indiquemos. Con la tecla “J”, podrá atacar a los enemigos. El último estado del personaje sería el de “Morir”, que una vez la vida de este se agote se habrá terminado la partida.



- **Enemigo:** Este diagrama define los estados implementados en los enemigos. Tenemos el estado “Quieto”, en la que el enemigo está en la misma posición, el estado “Alerta”, en la que el enemigo va realizando una serie de movimientos aleatorios, el estado “Perseguir”, en el cual, cuando el personaje entre en el área de acción del enemigo, éste le perseguirá hasta que el personaje salga de este área, entonces volverá a su posición del mapa anterior. Por último, tenemos el estado de “Derrotado”, en el que cuando el personaje aseste cierto número de golpes sobre el enemigo, este será eliminado.



2.5 Definición de las funcionalidades

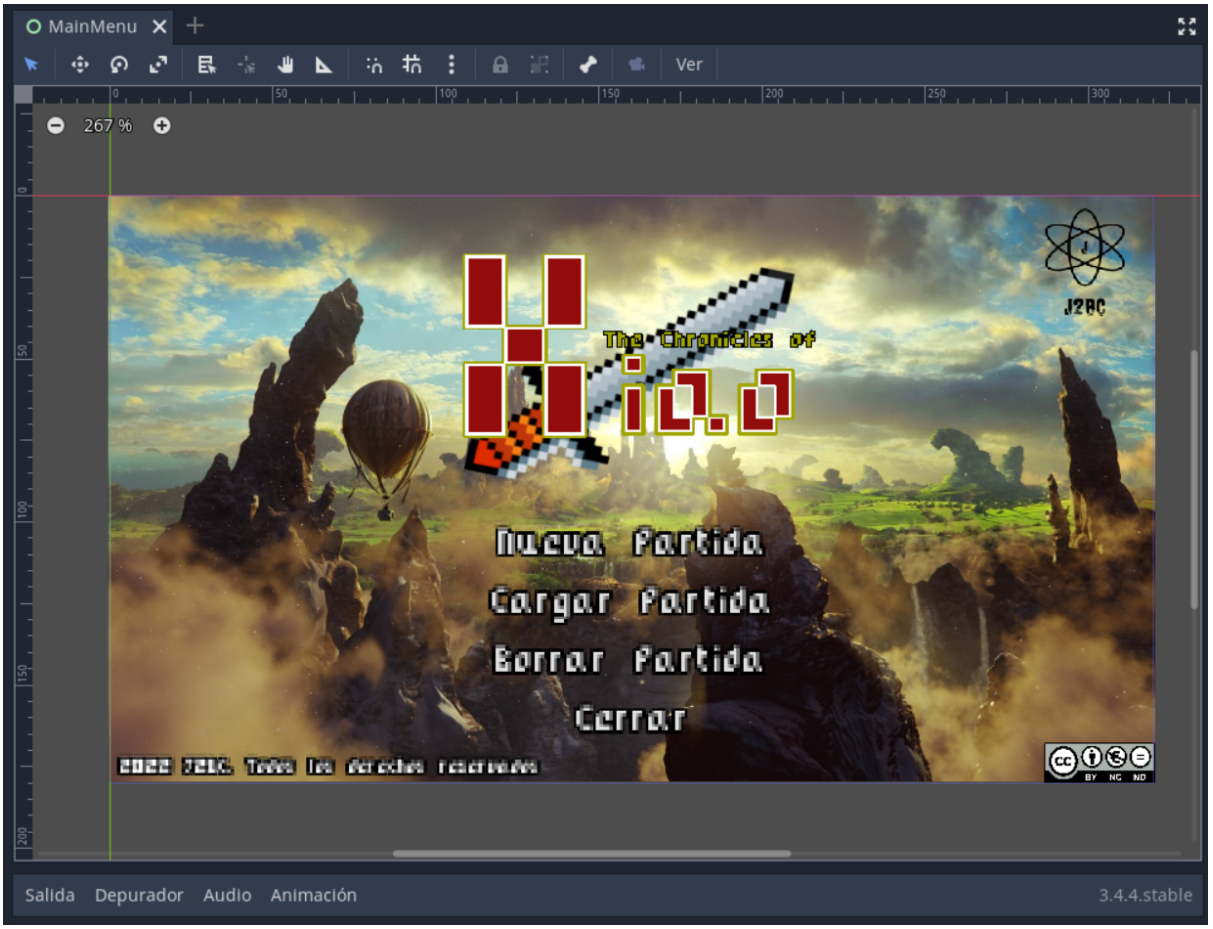
- Administración de las partidas mediante las opciones de guardar, crear y borrar partidas.
- Movilidad del jugador por el mundo.
- Administración del inventario, donde cada usuario podrá gestionar su inventario, alternar objetos utilizables etc. Desde este menú el juego se pausara.
- Interacción con NPC, el jugador podrá hablar con diferentes “Non Playable Character” y estos le contaran un poco la historia de lo que está ocurriendo.

3. Desarrollo

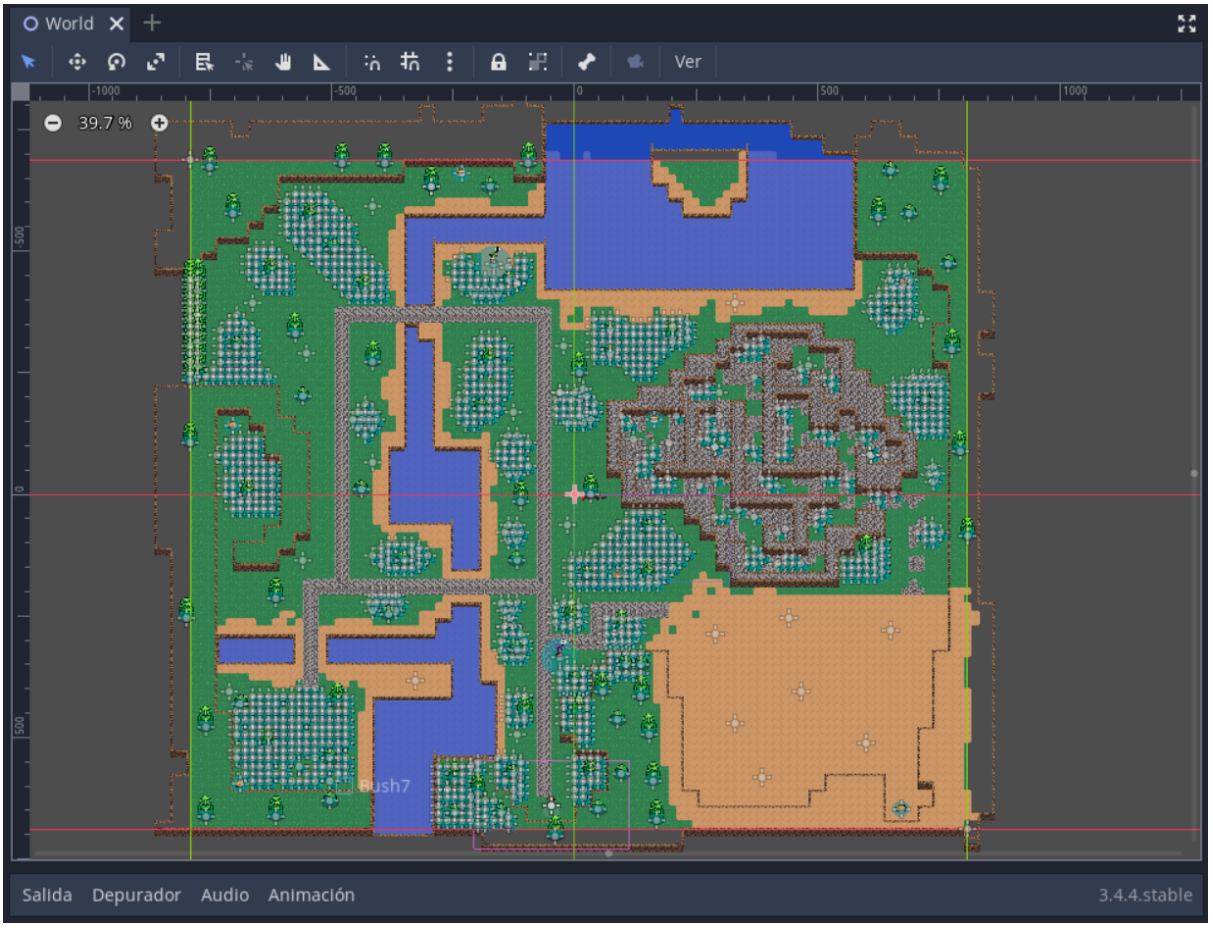
Escenas: Las escenas es una característica muy importante en Godot Engine, ya que contendrá todo el contenido de sus nodos y scripts correspondientes. Esto quiere decir que utilizar una escena podrá tener acceso a todos los nodos hijos.

Nodo: El nodo es una clase base para todos los objetos dentro de una escena. En este caso el equipo ha utilizado los llamados 'Node2D' los cuales son específicamente los que se usan para el desarrollo de objetos en 2D.

Menú principal: La primera escena que nos encontramos sería la del menú principal. Dónde nos encontraremos cuatro opciones: 'Nueva Partida', donde iniciaremos una nueva partida, 'Cargar partida', dónde cargaremos nuestra última partida guardada, 'Borrar partida', dónde podremos borrar nuestro último progreso y por último 'Cerrar', con el que saldríamos del juego.

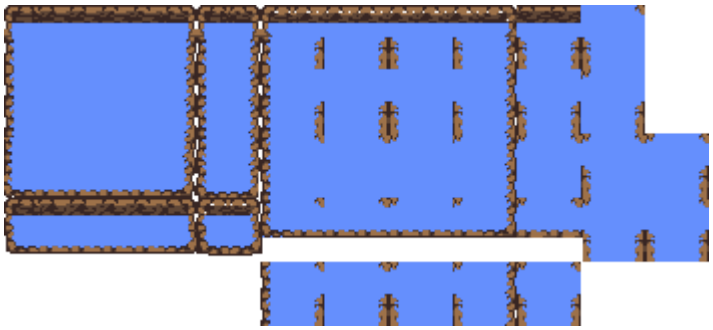


Mapa: La siguiente escena es el mapa de nuestro mundo, hemos utilizado un nodo llamado 'Tilemap', este nodo sirve para la construcción del mundo. En el cual hemos añadido diferentes texturas, para crear el entorno en el que se desarrollará la partida.

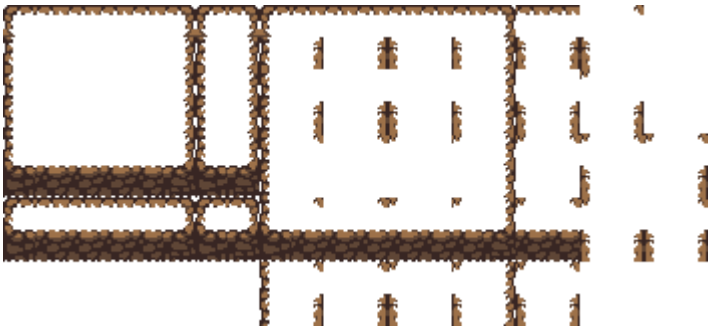


A continuación mostraremos los Tilemap y también los sprites que el equipo ha utilizado:

- **Rio:** Este es el Tilemap que se ha utilizado para crear el agua. Este Tilemap es para crear superficies elevadas. Este tiene un tamaño de 32*32

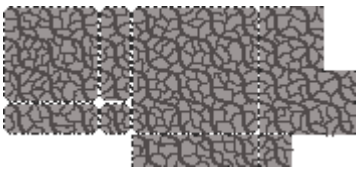


- **Montaña:** Este Tilemap es para crear superficies elevadas. Este tiene un tamaño de 32*32



- **Arena:** Este Tilemap se ha utilizado para crear zonas arenosas. Este tiene un tamaño de 16*16

- **Grava/Piedras:** Este último se ha utilizado para crear suelos de grava o montañas de piedra. Este tiene un tamaño de 16*16



- **Suelo:** Este sprite se ha utilizado para crear el suelo base del juego. En este caso se ha duplicado mediante una metodología que tiene Godot de crear en mosaico.

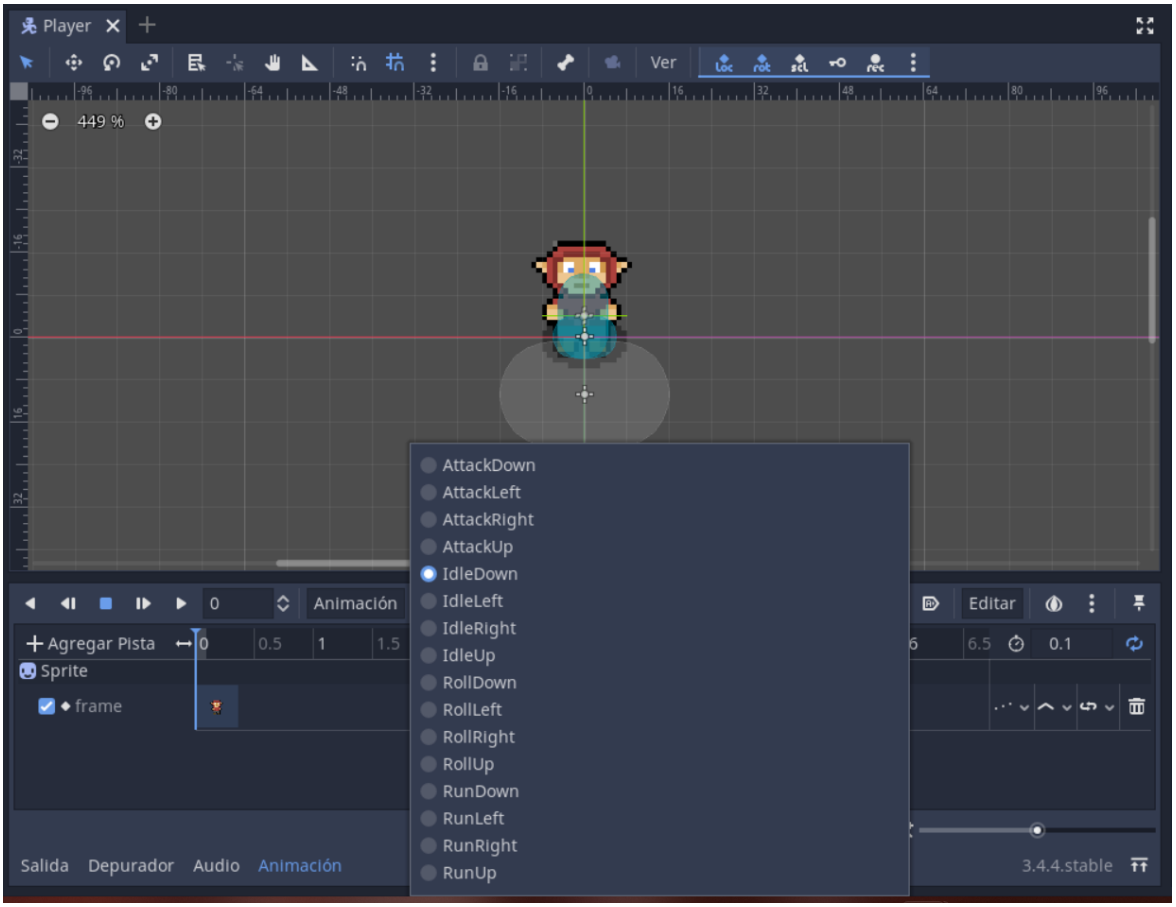
- **Árbol:** Este sprite se ha utilizado para crear un poco de vida durante la partida.

- **Arbusto:** Este sprite se ha utilizado para crear un poco de vida durante la partida.

➢ **Planta:** Este sprite se ha utilizado para crear un poco de vida durante la partida. En un futuro se ha pensado implementar que suelte objetos al azar.

➢ **Cofre:** Este sprite se ha utilizado para incitar la exploración durante la partida, ya que en un futuro estos darán recompensas.

Player: Para el player, el equipo ha diseñado un sprite con 60 fotogramas, para las diferentes animaciones que puede realizar.



(Aqui los 60 sheets)

Muestra de algunas acciones que el player puede realizar:

- Con esta función el player podrá moverse en todas direcciones.

```
func move_state(delta):
    var input_vector = Vector2.ZERO
    input_vector.x = Input.get_action_strength("ui_right") -
    Input.get_action_strength("ui_left")
    input_vector.y = Input.get_action_strength("ui_down") -
    Input.get_action_strength("ui_up")

    input_vector = input_vector.normalized()

    velocidad = move_and_slide(velocidad)
```

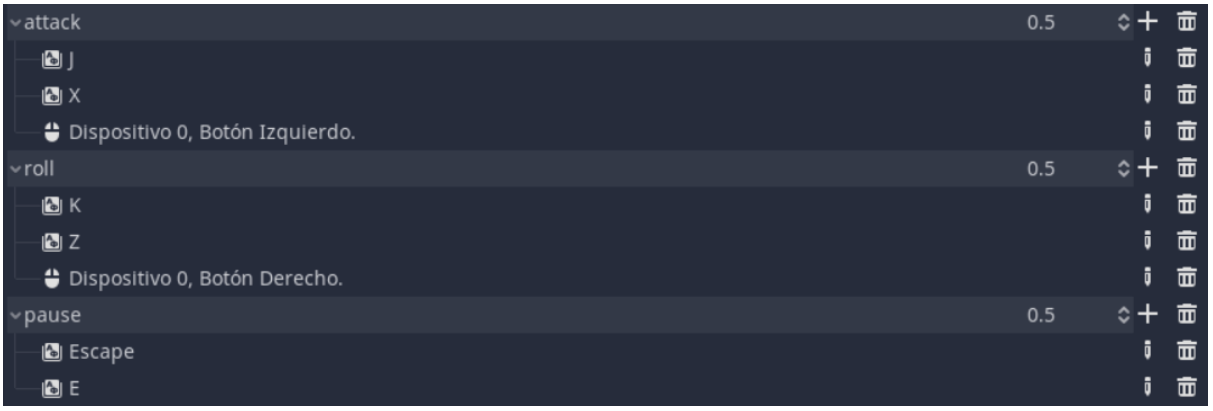
- Con esta función el player podrá rodar en la dirección a la que esté mirando.

```
func roll_state():
    velocidad = roll_vector * ROLL_SPEED
    animacionState.travel("Roll")
    velocidad = move_and_slide(velocidad)
```

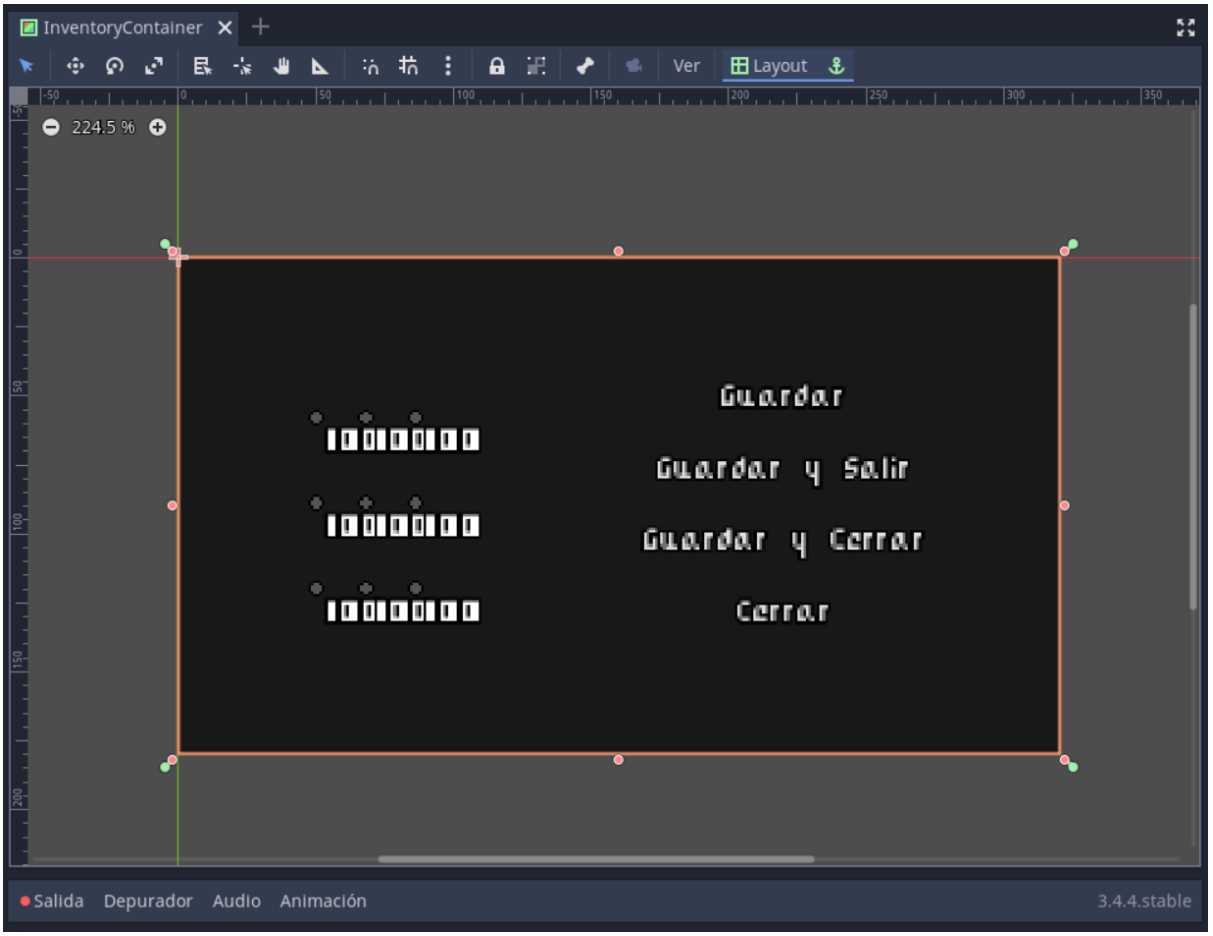
- Con la siguiente función el jugador podrá atacar a los enemigos.

```
func attack_state():
    velocidad = Vector2.ZERO
    animacionState.travel("Attack")
```

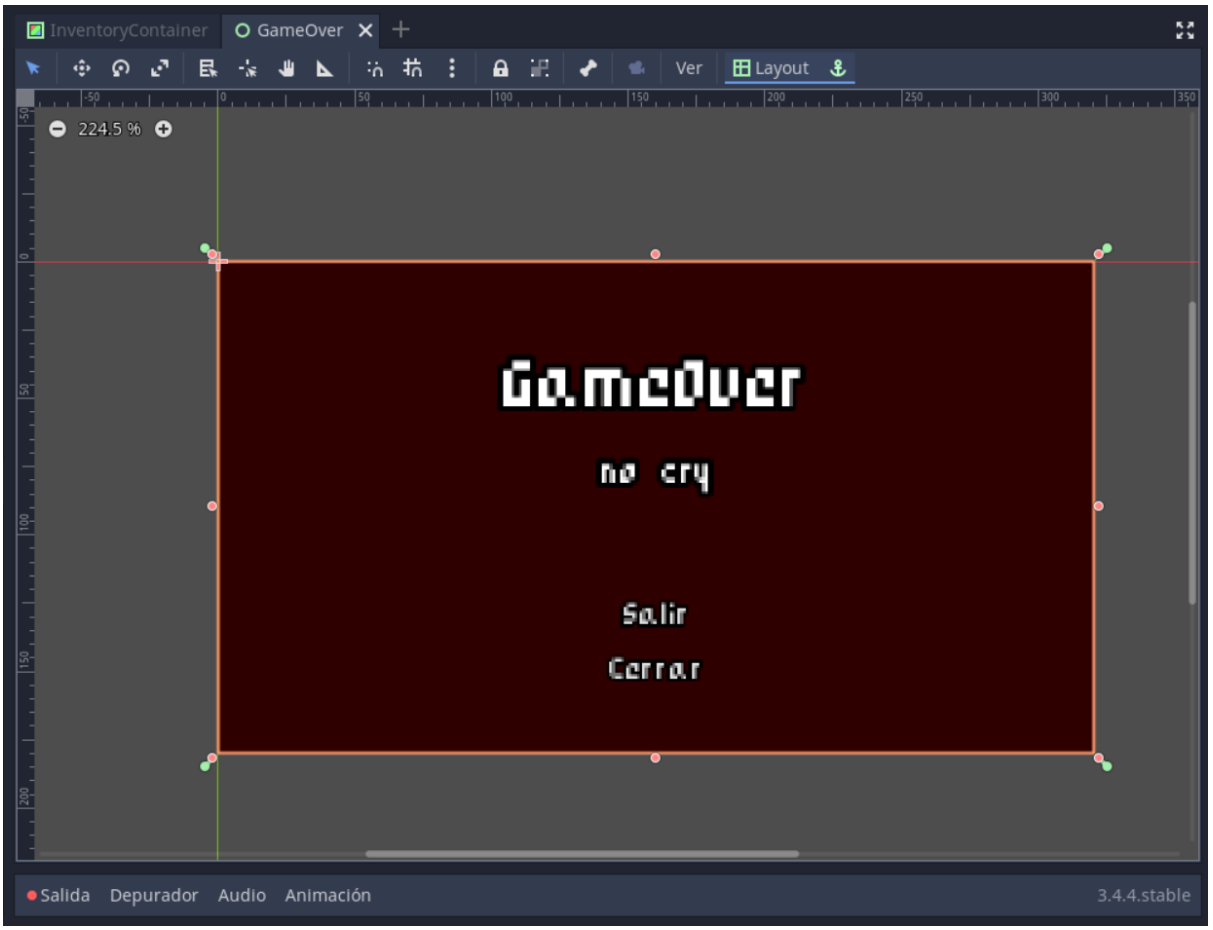
En la configuración del 'Mapa de entrada' se pueden configurar las teclas que queramos asignar a nuestros movimientos o acciones.



Inventario: Esta es la escena de inventario, donde el player podrá ver todos los objetos que tiene. Además de cuatro opciones donde podemos ‘Guardar’ la partida, ‘Guardar y salir’, donde guardaremos la partidas y nos mandará al menú principal, ‘Guardar y salir’, donde guardaremos la partida y nos cierra el juego o ‘Cerrar’, en la que cerraremos el juego directamente



Game Over: Esta escena nos aparecerá cuando el player haya perdido todas las vidas y tendremos dos opciones, en las que podemos salir al menú principal o cerrar el juego.



Enemigos: El equipo ha creado dos tipos de enemigos, los cuales tienen animación para los movimientos. Estos tienen una IA básica, para realizar los movimientos, los ataques y la persecución del Player. Además pueden volver a su posición inicial.

- **Murciélago**

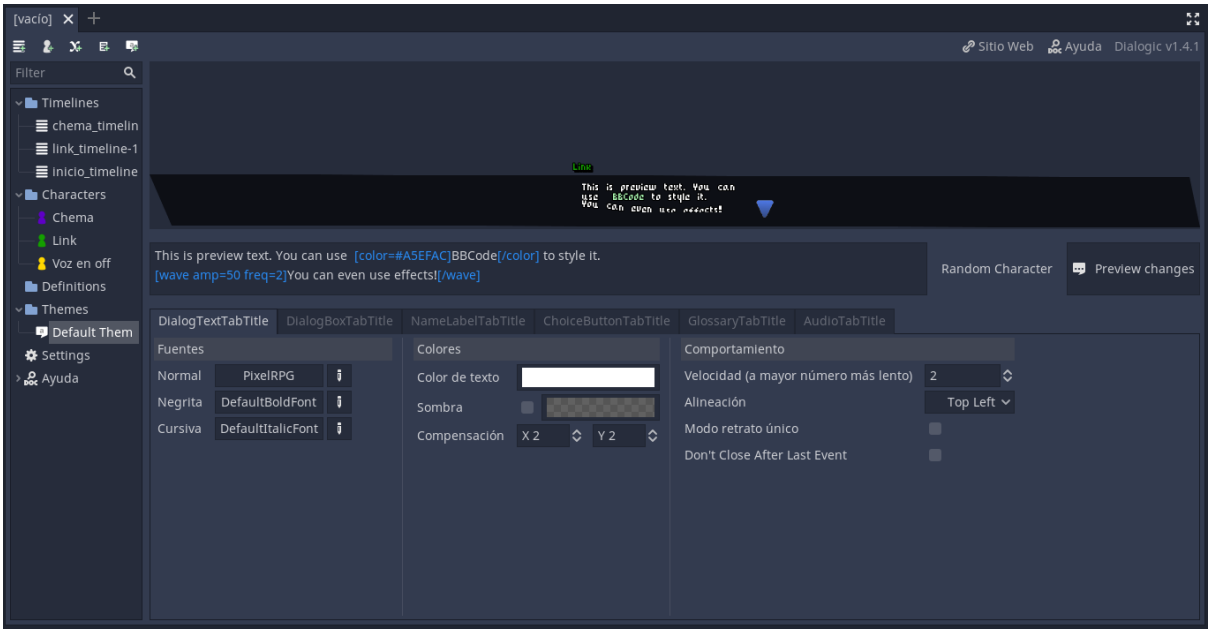
- **Limo**

A continuación mostramos la función de la IA del enemigo, con la cual le indicamos diferentes estados, 'IDLE'(quieto), 'WANDER'(vigilando) y 'CHASE'(cazando/siguiendo):

```
func _physics_process(delta):
	knockback = knockback.move_toward(Vector2.ZERO, FRICCION * delta)
	knockback = move_and_slide(knockback)

	match state:
		IDLE:
			velocidad = velocidad.move_toward(Vector2.ZERO, FRICCION*delta)
			seek_player()
			if wanderController.get_time_left() == 0:
				update_wander()

		WANDER:
			seek_player()
			if wanderController.get_time_left() == 0:
				update_wander()
			accelerate_towards_point(wanderController.target_position, delta)

			if global_position.distance_to(wanderController.target_position) <=
WANDER_TARGET_RANGE:
				update_wander()

		CHASE:
			var player = playerDetectionZone.player
			if player != null:
				accelerate_towards_point(player.global_position, delta)
			else:
				state = IDLE
```

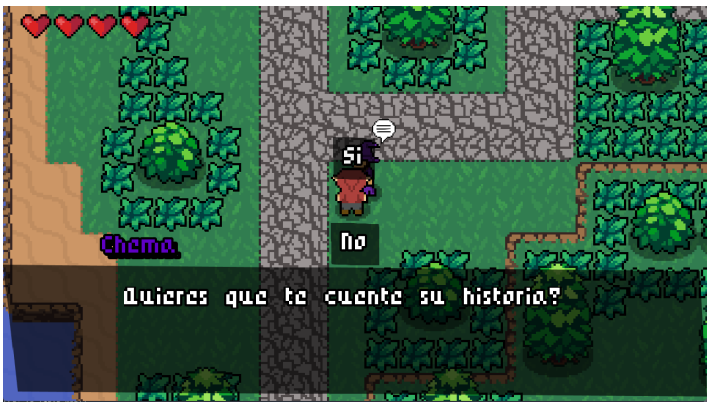
NPC: Los NPC o non playable jugable, son personajes que no son controlados por el jugador. El equipo los ha utilizado para interactuar con el player, ofreciendo información sobre lo que ha de hacer.

Addons: En este caso el equipo ha encontrado un addon perfecto que permite crear y editar diálogos con facilidad este addon se llama Dialogic por desgracia la versión del addon en Godot - Linux no funciona como debería entonces el equipo ha decidido continuar el proyecto en Windows.



Interacción: Para la interacción del Player con los NPC el equipo ha utilizado la siguiente función que nos aporta el addon:

```
func _input(event):
        if get_node_or_null('DialogNode') == null:
                if event.is_action_pressed("ui_accept")and activo:
                        get_tree().paused = true
                        var dialog = Dialogic.start('chema_timeline-1')
                        dialog.pause_mode =
Node.PAUSE_MODE_PROCESS
                        dialog.connect('timeline_end',self,'unpause')
                        add_child(dialog)
```



4. Conclusiones

Antes de terminar, cabe decir, que el desarrollo del videojuego ha tenido sus dificultades, pero a la vez ha sido reconfortante ver cómo poco a poco hemos ido viendo como iba creciendo el proyecto. Y aunque no hemos llegado a todos nuestros objetivos marcados en un inicio, nos sentimos orgullosos del trabajo realizado.

Partíamos con un conocimiento nulo sobre Godot Engine y es cierto que nos costó arrancar, además añadir, que veníamos de intentar crear un proyecto totalmente diferente. Pero gracias a que Godot tiene una gran comunidad, ayudó mucho a resolver ciertos problemas que hemos tenido a lo largo del desarrollo.

El resultado final del videojuego ha sido muy satisfactorio, el reto de hacerlo en un lenguaje nuevo para nosotros, ha ayudado al crecimiento personal de los integrantes, que ya sentían curiosidad por utilizar un motor gráfico para la creación de videojuegos.

Todo y que han quedado cosas en el tintero, se espera mejorar el videojuego en un futuro.

Somos conscientes de que con un poco más de tiempo, todas esas partes que hemos tenido que dejar por falta de tiempo, se podrían haber completado. Es complicado destinar el tiempo necesario en el proyecto y a la vez administrarlo con prácticas, exámenes, etc.., la acumulación de tareas nos lo ha hecho pasar un poco mal, pero, hemos visto que con esfuerzo y dedicación se puede conseguir.

Para finalizar comentar que no todo ha sido un camino tortuoso, hemos pasados momentos divertidos diseñando los personajes, el mapa, el menú, etc.., es muy motivador el hecho de empezar de cero un proyecto y poder dirigirlo hacía donde queramos, darle la estética y la temática que queramos, etc… Hemos aprendido mucho a realizar un proyecto en grupo, tanto como solucionar los problemas que iban apareciendo y estamos muy contentos con el esfuerzo que hemos realizado.

5. Webgrafia

Repositorio del Juego

Godot Docs

Base del videojuego

Inventario

Addon Dialogic

Tutorial Dialogic

Tutorial MiniMapa

Tutorial Guardado/Cargado

Información sobre sueldos

Spawn enemigos

6. Glosario

RPG: Género de videojuego cuyas siglas quieren significar “Role Playing Game”.

Videojuego: Juego electrónico en el que una o más personas interactúan por medio de un controlador.

Animación: Proceso para dar la sensación de movimiento a imágenes.

Interacción: Acción, relación o influencia entre el usuario y el NPC.

Sprite: una serie de imágenes unidas en un mismo archivo una al lado de otra y que representan al mismo personaje

Node2D: Un objeto de juego 2D, con una posición, rotación, escala y le da control al orden de renderizado del nodo. Todos los nodos de física 2D y los sprites heredan de Node2D.

Escena: Ejecutar un juego significa ejecutar una escena. Un proyecto puede contener varias escenas, cada nivel del juego es una escena

GDScript: Es el lenguaje nativo de programación del godot, que se encarga de controlar todas las acciones del juego.

Slots: Estos son ranuras, en este caso son las ranuras del inventario.

7. Anexo

Estando en el menú principal, nos encontraremos estas 4 opciones:

- **Nueva partida**: Si clicamos esta opción crearemos una nueva partida.

- **Cargar partida:** Si clicamos esta opción nos cargará la última partida guardada.

- **Borrar partida:** Si clicamos esta opción borraremos la última partida guardada.

Una vez entramos a nuestra nueva partida, nos encontraremos la siguiente escena:

A continuación ya podemos empezar a jugar, donde nos darán algunas instrucciones sobre los movimientos o acciones que puede realizar el jugador. Los detallamos a continuación.

Controles del jugador:

- **Dirección**: Podemos movernos con la tecla ‘W’ / ’Flecha Arriba’ hacía arriba, con la tecla ‘S’ / ‘Flecha abajo’ hacía abajo, la tecla ‘A’ / ‘Flecha Izquierda’ para ir hacía la izquierda y con la tecla ‘D’ / ‘Flecha Derecha’ iremos hacía la derecha.

➢ **Atacar:** Con la tecla ‘J’, ‘X’ o con el click izquierdo del ratón, Xiao sacará su espada y atacará a los enemigos.

➢ **Rodar:** Con la tecla ‘K’, ‘Z’ o click derecho del ratón, Xiao rodará hacía la dirección que le indiquemos.

➢ **Inventario:** Pulsando las teclas ‘E’ o ‘ESC’ entraremos en el menú del inventario, que además de poder gestionar nuestros items, nos encontraremos 4 opciones.

➢ **Guardar:** Podemos guardar nuestros avances en el juego.

➢ **Guardar y salir:** Además de guardar la partida sales al menú principal del juego.

➢ **Guardar y cerrar:** Además de guardar la partida sales del juego.

➢ **Cerrar:** Sales del juego.

Items: En la parte del inventario, podemos stackear los ítems arrastrando el item encima del que queremos stackear. Además podemos cambiar las posiciones de estos si el item arrastrado es diferente al que se arrastra.
En un futuro el equipo integrará la función de usar los objetos con el click derecho del raton.

Pantalla GameOver: Una vez que hayamos perdida todos los puntos de vida, nos aparecerá la pantalla de game over, en la cual encontraremos 2 opciones:

- **Salir:** Si clicamos sobre 'Salir' saldremos al menú principal del juego

- **Cerrar:** Si clicamos sobre 'Cerrar' cerraremos el juego por completo.