Una pantalla de televisión encendida

Descripción generada automáticamente

Logotipo

Descripción generada automáticamente

PROYECTO FINAL

Videojuego 2D

Andrés Pérez, Jaime Sansón y Francisco Pagadizabal

Logotipo

Descripción generada automáticamente

ÍNDICE

[Introducción 2](#_Toc166495125)

[Justificación 2](#_Toc166495126)

[Objetivos del Proyecto 2](#_Toc166495127)

[Metodología 3](#_Toc166495128)

[Herramientas de Trabajo 4](#_Toc166495129)

[Diseño del Videojuego 7](#_Toc166495130)

[Desarrollo Técnico 10](#_Toc166495131)

[Control de Calidad 11](#_Toc166495132)

[Estado 12](#_Toc166495133)

[Referencias 12](#_Toc166495134)

[Anexos 13](#_Toc166495135)

[Conclusión 14](#_Toc166495136)

# Introducción

Antes de llegar a la idea final, exploramos diferentes opciones en el mundo de los motores de juego. Inicialmente, consideramos trabajar con Unreal Engine 5. Sin embargo, debido a la complejidad técnica de esta idea, nos vimos obligados a adaptarnos y trasladarnos a Unity. Allí, nos enfocamos en un género que sigue siendo muy popular: los roguelike. Además, consideramos la posibilidad de incorporar una vista en 2D, similar a la de un Metroidvania.

Finalmente, convergimos en la idea de crear un juego de plataformas con elementos de roguelike. Esta decisión nos permitió fusionar la esencia de los videojuegos clásicos con la frescura y la creatividad característica de los juegos indie contemporáneos.

Nuestro objetivo con este proyecto es la creación de un videojuego en el motor Unity. Con esto, no buscamos innovar sino ponernos a prueba en un entorno en el cual nos sentimos cómodos.

Este proyecto es un reto para nosotros ya que no podemos utilizar nada de lo aprendido estos años atrás, situándonos frente a un muro de un nuevo programa y lenguaje, aunque este sigue la estructura de C#.

# Justificación

Nuestro interés radica en aprender un nuevo lenguaje de programación y adentrarnos en el mundo del desarrollo de videojuegos. Todos nosotros hemos disfrutado de juegos creados por otros, y ahora queremos experimentar el proceso desde el otro lado, convirtiéndonos en desarrolladores.

Partiendo desde cero, nos embarcamos en este proyecto con el objetivo de explorar hasta dónde podemos llegar. Nos motiva especialmente la combinación de uno de los géneros más característicos del sector de los videojuegos con un género relativamente nuevo basado en la rejugabilidad.

# Objetivos del Proyecto

Nuestro objetivo principal es desarrollar un videojuego que fusiona las mecánicas de los juegos de plataformas con los elementos característicos de los juegos roguelike. En esta aventura, buscamos ofrecer a los jugadores una experiencia única en cada partida, donde la rejugabilidad sea la principal protagonista, evitando la monotonía al presentar nuevas situaciones y soluciones en cada sesión de juego.

Una de las piedras angulares de esta experiencia será la aleatoriedad inherente a los juegos roguelike, donde cada partida se generará de manera procedimental, asegurando que ningún recorrido sea igual. Además, pondremos énfasis en la diversidad y la importancia de los objetos dentro del juego. Estos objetos serán clave para otorgar ventajas estratégicas a los jugadores, agregando profundidad y emoción a la exploración del mundo del juego.

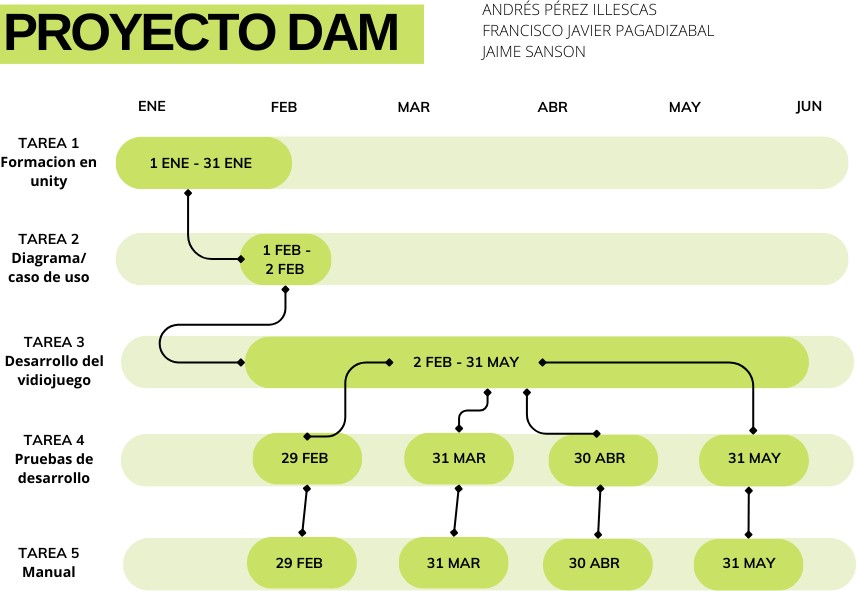
# Metodología

Scrum

Hemos optado por la metodología ágil Scrum, dividiendo nuestro trabajo en cuatro sprints mensuales. Durante cada sprint, recopilamos información sobre nuestras actividades individuales, buscamos oportunidades para unificar o corregir el trabajo realizado, y asignamos nuevas tareas para el próximo sprint.

Diagrama de Gantt (DESGLOSARLO MAS)

Primera versión



Versión final

Escala de tiempo

Descripción generada automáticamente

# Herramientas de Trabajo

Unity

Elegimos utilizar Unity como plataforma de desarrollo por varias razones clave. Primero, su integración con GitHub facilita un control de versiones robusto, permitiéndonos colaborar de manera eficiente, con cada miembro trabajando en su propia rama para luego fusionar los cambios en la rama principal, garantizando un entorno de trabajo seguro y organizado. Segundo, Unity ofrece una amplia gama de funcionalidades gratuitas. Tras una investigación detallada, descubrimos que pocos otros motores de desarrollo de videojuegos proporcionan tantas herramientas y servicios sin costo alguno, lo que nos permitió desarrollar un juego completo sin restricciones.

UE5

Comparativa Unity / UE5

Unity Control Version (REDUCIR CONTENIDO)

Al principio del desarrollo, optamos por utilizar el servicio de control de versiones de Unity. Sin embargo, encontramos que esta herramienta no se ajustaba completamente a nuestras necesidades y resultaba difícil de comprender. Además, experimentamos problemas al abrir el proyecto en equipos distintos al que se utilizó para su creación. Por estas razones, decidimos descartar el uso de esta herramienta en el proyecto.

GitHub

Este repositorio se ha convertido en nuestra opción definitiva, ya que la capacidad de crear ramas (que no pudimos averiguar cómo hacer en el Control de Versiones de Unity) nos permite agilizar el proyecto. Ahora, el trabajo se divide en cuatro ramas, además de la principal, con una rama de pruebas asignada a cada miembro del equipo. Además, hemos integrado la aplicación de escritorio de Unity, que ofrece una gran compatibilidad con la plataforma de desarrollo. Esta herramienta nos permite cambiar de rama al instante en Unity, facilitando la coordinación y la gestión del proyecto.

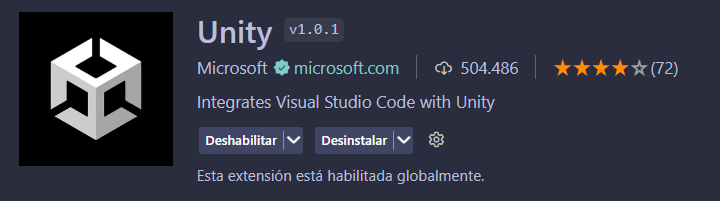
Visual Studio Comunity 2022

Hemos utilizado la versión gratuita de Visual Studio como nuestra herramienta principal para el desarrollo de scripts. Esta versión incluye el paquete de Unity, lo que garantiza la instalación automática de todas las dependencias necesarias para un funcionamiento correcto. Sin embargo, es necesario seleccionar este editor dentro de los ajustes del proyecto en Unity para asegurar una integración adecuada.

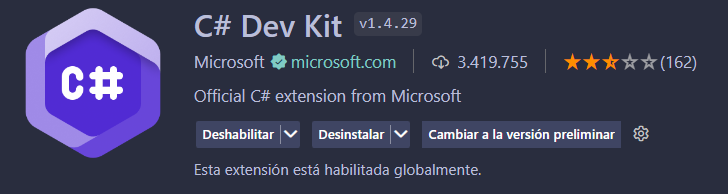
Visual Code

Esta versión también ha sido utilizada por un miembro del equipo, para poder utilizar este tubo que descargar extensiones relacionadas con Unity y habilitar la edición dentro de las preferencias de Unity. La extensión que se han utilizado para el uso de visual code son:

Unity: esta extensión desarrollada por Microsoft nos permite detectar los paquetes de unity y visual code para que no den errores de formato y conectar visual code para que a la hora de abrir los diferentes scripts en cs nos redirija al visual sin problema. La id de la extensión es “visualstudiotoolsforunity.vstuc”. La versión es la 1.0.1.



C# Dev Kit: Como indica su nombre, el kit general para detectar el lenguaje de C#. Con esta extensión podemos administrar el código en el explorador de soluciones, ejecución y detección de errores, así como el desarrollo en diferentes SO, como Windows, Linux o macOS. La versión es la 1.4.29.



# Diseño del Videojuego

Trama

La historia se centra en la supervivencia del personaje principal. Este, se despierta de repente en su casa a las afueras del pueblo en el que vive, viendo como los enemigos saquean y queman su pueblo. En ese momento, decide enfrentarse a estos enemigos para así poder salvar lo que queda de su poblado y lo que queda de su familia. El objetivo de este es llegar hasta el líder de los enemigos y derrotarlo para así poder vivir en paz.

Gráficos

En el proceso de creación de los elementos visuales para nuestro juego, hemos optado por utilizar una herramienta conocida como ASEPRITE. Esta herramienta nos ha proporcionado la facilidad y versatilidad necesarias para realizar pixel art de alta calidad. Con ASEPRITE, hemos podido diseñar personajes, escenarios, objetos y efectos visuales, centrándonos en la estética característica de los juegos de 16 bits.

El término "16 bits" se refiere a una era específica en la historia de los videojuegos, que abarca las décadas de 1980 y 1990. Durante este período, las consolas de videojuegos como la Super Nintendo (SNES) y la Sega Genesis (Mega Drive) tenían procesadores de 16 bits. Estas consolas tenían limitaciones técnicas en comparación con las plataformas modernas, lo que resultaba en gráficos pixelados y una paleta de colores limitada.

Para lograr un estilo visual coherente y atractivo para nuestro juego, hemos realizado una investigación exhaustiva y hemos obtenido referencias de varios juegos que se basan en la estética de los 16 bits. Estas referencias nos han servido como inspiración y guía para diseñar algo propio y original, adaptado a la trama y el estilo del juego que estamos desarrollando. Al estudiar y comprender los elementos visuales de los juegos de 16 bits, podemos capturar la esencia y el encanto de esa época en nuestro propio trabajo.

Sonido

En cuanto al aspecto del sonido, hasta el momento no hemos iniciado su desarrollo.

Personajes

Jugador

El jugador es el único personaje que el usuario puede controlar. Equipado con un arma para el combate, posee habilidades especiales que pueden mejorar o adaptarse para enfrentarse a los enemigos durante el juego.

NPC

Los NPC’s son personajes no controlados por el usuario que cumplen diversas funciones dentro del juego. Estos personajes suelen estar relacionados con la trama y pueden proporcionar diálogos significativos que contribuyen al desarrollo de la historia. Además, algunos NPC’s pueden ofrecer sistemas de intercambio de objetos.

Enemigos (añadir listado de enemigos)

Los enemigos son los antagonistas del jugador. Equipados para el combate cuerpo a cuerpo, estos enemigos siguen rutas predefinidas de patrulla y exhiben una variedad de comportamientos durante el juego, ofreciendo desafíos y enfrentamientos únicos para el jugador.

Jefes

Los jefes son enemigos más poderosos que se encuentran al final del juego. Estos enfrentamientos son considerados más difíciles que el resto debido a su alta defensa y potencia ofensiva. Los jefes suelen requerir estrategias específicas y habilidades avanzadas para ser derrotados con mayor facilidad, proporcionando un punto culminante emocionante para la experiencia del jugador.

Entorno

Plataformas

Las plataformas son elementos del escenario que el jugador puede utilizar para desplazarse o alcanzar áreas específicas. Esto incluye tanto plataformas estáticas como móviles, como las bidireccionales, que ofrecen una variedad de desafíos y oportunidades de juego.

Escaleras

Las escaleras son elementos del escenario que se enfocan en facilitar el movimiento vertical controlado por el jugador. Estas estructuras permiten al jugador ascender o descender entre diferentes niveles de altura dentro de los niveles del juego, ofreciendo una forma intuitiva y fluida de explorar entornos y acceder a nuevas áreas.

Pinchos

Mecánicas

Movimiento

El movimiento se divide en dos secciones: el utilizado por el jugador y el utilizado por los enemigos. El jugador puede moverse en ocho direcciones y cuenta con diversas habilidades que mejoran su movilidad, como el doble salto o el dash. Además, tiene la capacidad de escalar escaleras y subirse a plataformas bidireccionales. Por otro lado, los enemigos tienen un movimiento limitado a dos direcciones, centrado en una zona designada como su patrulla.

Ataque

El jugador tiene un patrón de ataque determinado por el arma elegida al inicio del juego, que puede ser potenciado por objetos obtenibles durante la partida. En contraste, los enemigos realizan ataques por contacto.

Vida

El sistema de vida determina la salud del jugador y de los enemigos. El jugador comienza con 100 puntos de salud, que pueden aumentar o disminuir según los objetos encontrados a lo largo de los niveles. Los enemigos tienen una salud variable adaptada a su diseño y nivel de peligro.

Objetos (Aclarar, como se obtienen y como interactúan con el jugador) (Incluir listado)

Los objetos son elementos que se encuentran dispersos de manera aleatoria por los niveles durante la partida. Estos objetos pueden proporcionar aumentos en estadísticas o habilidades únicas para el jugador.

Arquitectura

Diseño de niveles

En el diseño del juego, se distinguen elementos tanto obligatorios como opcionales. Entre los elementos obligatorios se encuentran el jugador, los enemigos y las plataformas, los cuales conforman la estructura básica del juego.

Por otro lado, los elementos opcionales incluyen objetos especiales, así como también elementos ambientales como escaleras, pinchos u otros obstáculos que añaden profundidad y desafío al diseño del nivel. Estos elementos opcionales permiten una mayor diversidad y complejidad en la jugabilidad, ofreciendo diferentes caminos y estrategias para los jugadores mientras exploran el mundo del juego.

UI

La interfaz del juego se divide en dos partes distintas: los menús y la interfaz dentro del juego.

En los menús, los jugadores pueden acceder a opciones como ajustes, salir del juego y configuraciones de resolución, entre otras. Estos menús proporcionan un entorno donde los jugadores pueden gestionar aspectos técnicos y de configuración antes de iniciar o durante la partida.

Dentro del juego, la interfaz muestra información crucial para la experiencia de juego, como la vida del jugador, un mapa que indica la ubicación actual, y los objetos obtenidos durante la partida. Esta interfaz dentro del juego es fundamental para que los jugadores se mantengan informados sobre su progreso, salud y recursos disponibles mientras exploran el mundo del juego y enfrentan desafíos.

# Desarrollo Técnico

Programación

AÑADIR CAPTURAS DE CODIGO (BREVES)

MainMenu.cs

En este script se basa en cambiar de escena, hay 3 opciones Jugar, la cual nos manda al primer nivel del juego, Ajustes, la cual nos manda a otra escena relacionada con ajustes gráficos y salir, que cierra la aplicación. Dentro de este script también viene incluido los métodos usados en ajustes.

CameraController.cs

Uno de los scripts más importantes del juego, este se usa obteniendo la posición del jugador y unos límites, luego de obtener esto cambiamos la posición de la cámara de manera constante llevándola a la posición den jugador. En esta clase está también el método necesario para que el fondo del videojuego se vaya moviendo a medida que se mueve el jugador y por ende la cámara principal.

LevelEnd.cs

Este script trabaja con colisiones, una vez el jugador entra en contacto con el game object que tiene asociado este script se cambia la escena.

PlayerMovement.cs

Script importante el cual se encarga del movimiento del jugador, en este obtenemos parámetros del jugador y estadísticas, constantemente se comprueban los controles de movimiento horizontal y salto. Calculando así la velocidad de movimiento del jugador para saber hacia dónde mira el Sprite de este. Aquí también está la función que controla la capacidad de realizar doble salto.

PlayerHealth.cs

Está relacionado con la vida del jugador, comprobando constantemente que es lo que le ha hecho daño, ya sean cosas del escenario o enemigos. Cuenta también con un método que vuelve invencible al jugador una vez ha recibido daño, para crear así una oportunidad de escapar o evadir al enemigo. A demás, contamos con la función necesaria para subir la vida del jugador.

PlayerAttack.cs

Este Script cuenta con estadísticas de daño, también con un método de empuje para cuando el jugador es atacado. Este último método lo conseguimos elevando la posición del jugador y ejerciendo una fuerza horizontal, esto se hace con un método el cual obtiene la velocidad de nuestro jugador y la modifica. Esta clase tambien guarda también las funciones relacionadas con la subida de estadísticas de daño de ataque y velocidad de ataque

EnemyStats.cs

Contiene las estadísticas del enemigo, entre estas tenemos la vida, velocidad y el daño.

EnemyLogic.cs

La lógica del enemigo se encarga de la zona de patrulla del enemigo y de la detección del jugador.

El enemigo patrulla entre el punto A y B, esto se hace obteniendo la la posición en el eje x o y (dependiendo del enemigo) y luego con un método hacemos que camine solo hacia esa dirección, para la detección el enemigo cuenta con un rango, por medio de saber la posición del jugador y el enemigo de manera constante. Una vez entra en ese rango el enemigo cambia el foco a la posición del jugador en el plano horizontal, en caso de perder el foco este vuelve a la zona de patrullaje.

En caso de llegar hasta el jugador, por medio de colisiones llama a la clase PlayerHealth.cs y le hace daño.

SpikeDamage.cs

Se encarga de detectar cuando el jugador toca los pinchos y le deja la vida a 0.

Explicar método para randomizar, niveles y objetos

# Control de Calidad

Funcionalidad

En esta fase, nos enfocamos en verificar los controles, mecánicas e interacciones dentro del juego, con especial atención en la experiencia del jugador. Nos aseguramos de que todos los controles respondan correctamente, que las mecánicas de juego funcionen como se espera y que la interacción entre elementos del juego sea coherente y satisfactoria.

Usabilidad

Nuestro objetivo con estas pruebas es garantizar que el juego sea intuitivo y disfrutable para el jugador. Evaluamos el equilibrio entre la dificultad y la recompensa, así como la sensación general al jugar. Nos esforzamos por crear una experiencia de juego que sea fluida, emocionante y satisfactoria para todos los jugadores.

Localización

En esta fase, verificamos que todos los elementos del juego estén en su lugar correcto después de pasar pantallas o realizar cambios, como el patrullaje de los enemigos. Nos aseguramos de que los objetos del juego estén ubicados correctamente y que todas las funciones y comportamientos se mantengan consistentes independientemente de las acciones del jugador o del progreso en el juego.

# Estado

El juego estaría a un 70% - 80% de su compleción, ahora el juego cuenta con:

* Pantalla de inicio.
* Pantalla de ajustes.
* Pantalla de fin del juego.
* Jugador con movimiento en eje x e y, ataques y sistema de salud.
* Enemigos con salud, daño y patrullaje en una zona.
* Detección del jugador por parte del enemigo.
* Plataformas en movimiento.
* Escaleras de mano.
* Gráficos propios.
* Objetos de pruebas (Objetos básicos que cambian estadísticas).

Nos faltaría por implementar:

* Selección de arma al principio.
* Plataformas bidireccionales
* Pulir las físicas.
* Interfaz de objetos
* Refinar los objetos.
* Hacer el sistema de combate jugador a enemigo.
* Batalla contra el jefe

# Referencias

Durante el desarrollo de este proyecto, hemos aprovechado una variedad de recursos para familiarizarnos con el nuevo lenguaje de programación. Entre ellos, hemos realizado cursos en plataformas como Udemy, los cuales nos han brindado un primer acercamiento a la aplicación y sus scripts. Estos cursos han sido fundamentales para adquirir los conocimientos necesarios y comenzar a desarrollar el proyecto con solidez y confianza.

Comunidad de Unity: <https://discussions.unity.com>

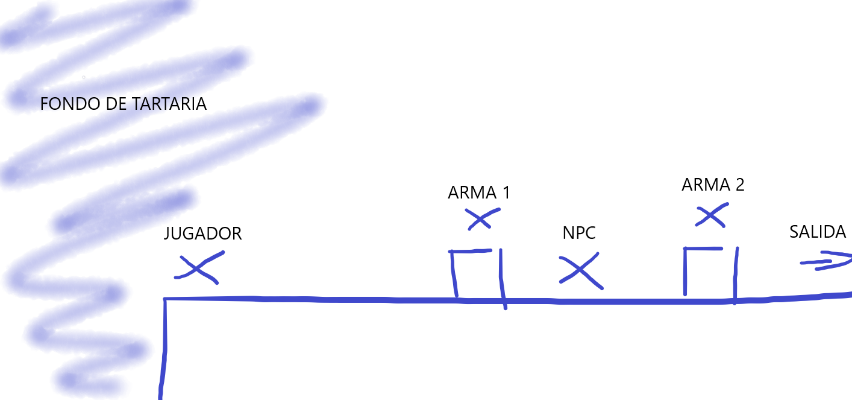
StackOverflow: <https://stackoverflow.com>

Cursos gratuitos YouTube.

# Anexos

Sprites

Bocetos



Interfaz de usuario gráfica, Diagrama, Aplicación

Descripción generada automáticamente con confianza mediaGráfico, Gráfico de cajas y bigotes

Descripción generada automáticamente

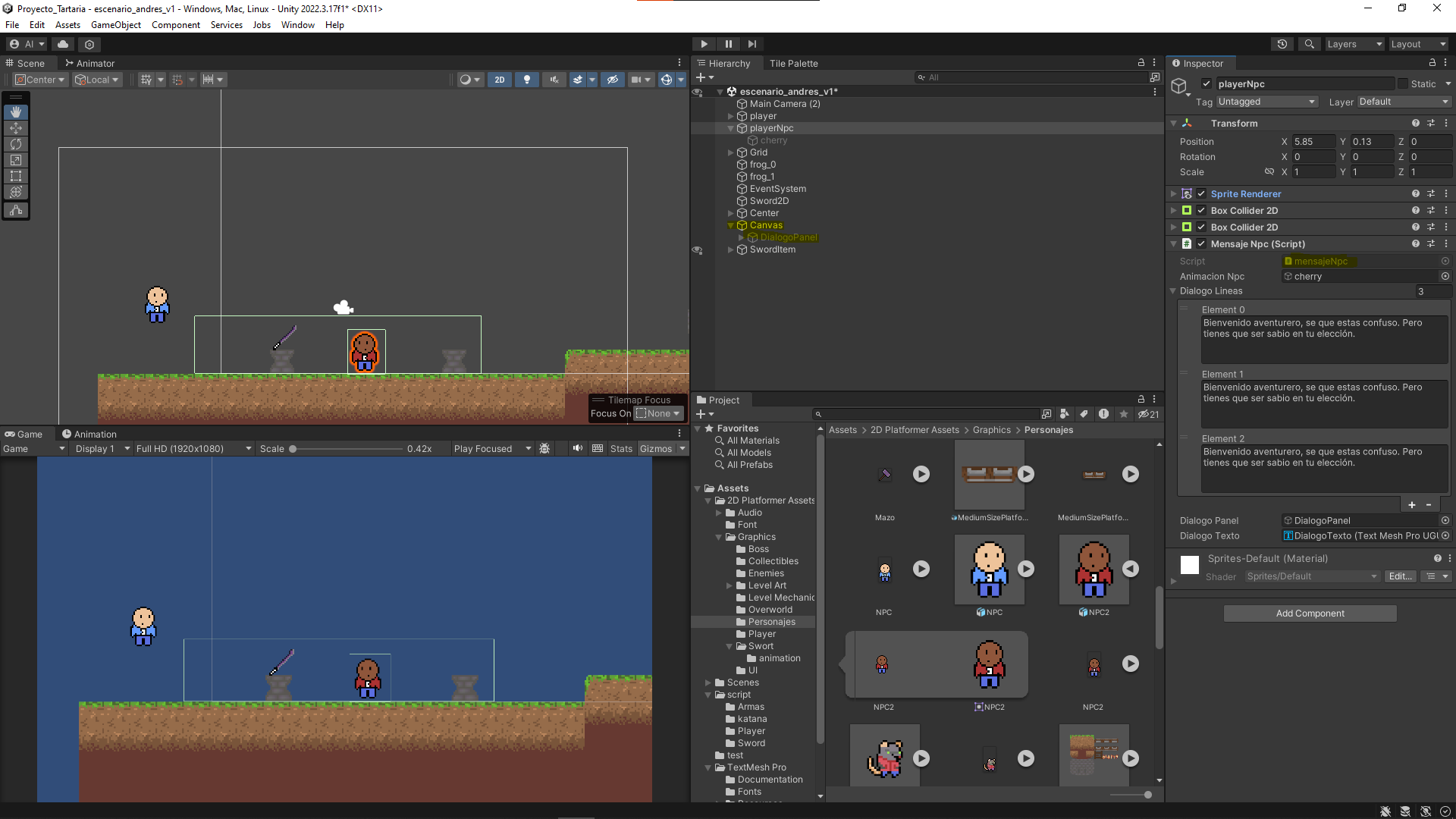
Videos

Video enseñando estructura de niveles y de juego:

<https://www.youtube.com/watch?v=71hsYemCwho&ab_channel=FranciscoJavierPagadizabalS%C3%A1nchez>

Imágenes enseñando funcionamiento de los objetos:

Imágenes enseñando el dialogo con NPC:





# Conclusión

Hablar en la conclusión sobre los problemas y visión del futuro