

**Juego de rol educativo**

“La” “La Misión del Aprendiz”

**Integrantes del grupo:**

* Sofia Obejero
* Melina Ramirez
* Tolaba Walter
* Rojas Gonzalo

**Proyecto final 7mo año**

INDICE

**Introducción** ------------------------------------------------------------------------------------------------- 2

Definición del proyecto ---------------------------------------------------------------------------- 2

¿Qué servicios ofrecerá?¿Cómo? ------------------------------------------------------------- 2

¿A quién sera ofrecido? -------------------------------------------------------------------------- 3

¿Qué necesidad desea cubrir? ----------------------------------------------------------------- 3

Pagina web ------------------------------------------------------------------------------------------ 4

Significado ------------------------------------------------------------------------------------------- 4

Asignación de tareas ----------------------------------------------------------------------------- 5

**Herramientas** ----------------------------------------------------------------------------------------------- 6

Motor gráfico ---------------------------------------------------------------------------------------- 6

LibreSprite ------------------------------------------------------------------------------------------- 8

Visual Code ----------------------------------------------------------------------------------------- 8

Github ------------------------------------------------------------------------------------------------ 9

**Avances** ------------------------------------------------------------------------------------------------------ 11

Estudios ---------------------------------------------------------------------------------------------- 11

Diagrama de Gannt ------------------------------------------------------------------------------- 12

Diseño - Juego -------------------------------------------------------------------------------------- 14

Página web ------------------------------------------------------------------------------------------ 31

Programación del juego --------------------------------------------------------------------------- 32

**Bibliografía** ------------------------------------------------------------------------------------------------ 121

Diseño - juego ------------------------------------------------------------------------------------ 121

Programación del juego ----------------------------------------------------------------------- 121

INTRODUCCION

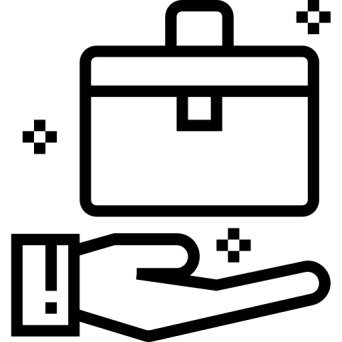
DEFINICIÓN DEL PROYECTO

La misión del Aprendiz" es un juego RPG (Role Play Game), que se caracteriza por ser un juego de rol que se fundamenta en la estructura de la técnica N°5. Su objetivo primordial es impartir conocimientos y repasar los fundamentos básicos de la programación, haciendo uso de diversos puzzles y quizzes. Con el fin de brindar mayor claridad acerca de los juegos de rol, deseamos presentar algunos ejemplos, tales como Forager, Undertale y Oneshot, los cuales pertenecen al género de aventura, historia y fantasía, entre otros elementos temáticos.

Para el desarrollo de nuestro proyecto, hemos optado por basarnos en juegos de rol educativos reconocidos, como "Proyecto Historia", "Monster School Bus" y "Math Quest Adventure", los cuales se centran en la enseñanza de la historia, las ciencias naturales y las matemáticas, respectivamente. Nuestro juego tiene como propósito que el usuario no realice programación directamente dentro del juego en sí, sino que adquiera los conceptos esenciales de la programación, con el objetivo final de poder programar en un auténtico editor de código.

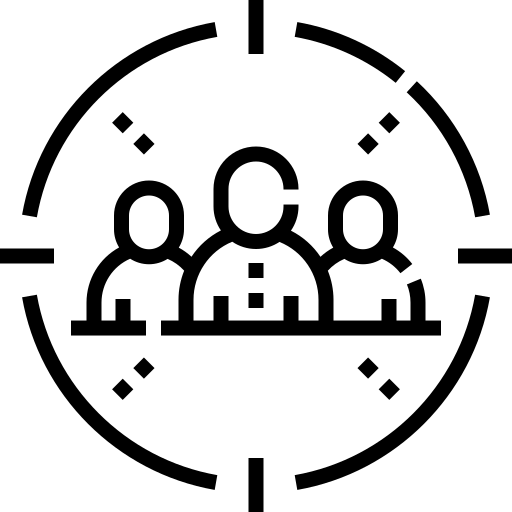
¿QUÉ SERVICIOS OFRECERÁ Y CÓMO?

Nuestra propuesta consiste en un producto que brinda una solución interactiva e intuitiva para adquirir los conocimientos fundamentales de programación, incluyendo los tipos de variables, las estructuras condicionales y los bucles de repetición. Esta herramienta se compone de explicaciones teóricas y desafiantes puzzles diseñados para cada concepto específico. La transmisión de estos contenidos se realizará mediante una simulación de un profesor virtual que proporcionará retroalimentación al jugador, ofreciendo correcciones en caso de cometer errores y recompensas en forma de ítems en caso de realizar correctamente las actividades. Estos ítems serán acumulados a medida que el jugador avance de nivel. Una vez que todos los ítems hayan sido recolectados, se desbloqueará la puerta final que marcará el final del juego.



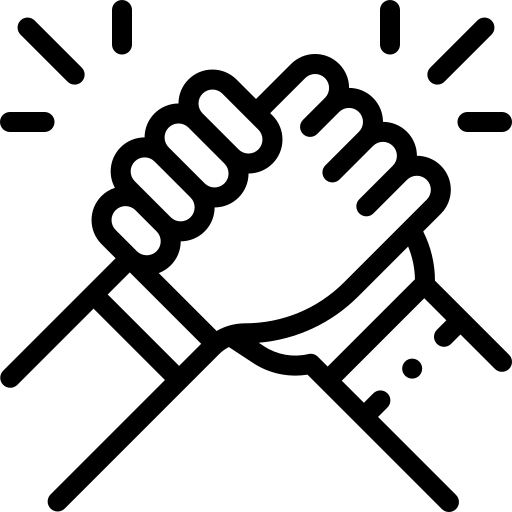
¿A QUIÉN SERA OFRECIDO?

La idea principal es ofrecerlo a los alumnos que ingresan al 4to año de informática para introducirlos a la programación y, al mismo tiempo, a los profesores que deseen utilizarlo como apoyo durante sus clases. Para que esto sea posible, el juego contará con dos modos: uno evolutivo y otro libre. El primero tendrá todos los niveles bloqueados para que el usuario pueda aprender de manera progresiva cada concepto. Mientras que el segundo contará con todos los niveles desbloqueados para que se pueda acceder a cualquiera de ellos en cualquier momento. De este modo, si el jugador desea ver un nivel más complejo, pueda hacerlo sin la necesidad de haber desbloqueado los niveles anteriores.



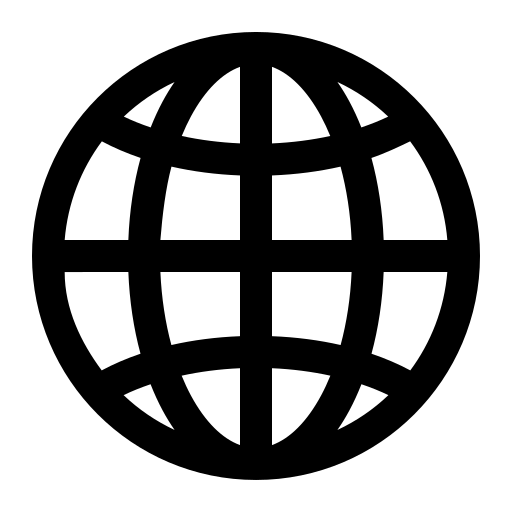
¿QUÉ NECESIDAD DESEA CUBRIR?

Nuestro juego se presenta como una herramienta de refuerzo educativo diseñada específicamente para atender las necesidades de los estudiantes que enfrentan dificultades en la adquisición de los conceptos básicos de programación. Reconocemos que muchos alumnos pueden encontrarse en esta situación, incluso después de haber avanzado en su trayectoria académica, debido a diversos factores que han obstaculizado su aprendizaje en años anteriores, tales como la falta de presencia docente o la incapacidad para mantener el ritmo de la clase.

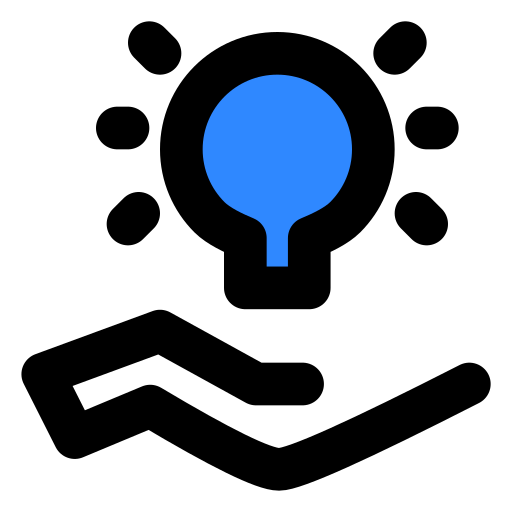


COMPLEMENTO: PAGINA WEB

Dado que nuestro proyecto no tiene fines lucrativos, hemos decidido implementar una página web complementaria como medio para proporcionar información relevante y el instalador del juego. La página web contendrá toda la información necesaria acerca del juego, incluyendo un tutorial para su uso. De esta forma, evitamos tener que publicar el juego en una plataforma externa, ya que el instalador estará disponible en nuestra propia página web.

SIGNIFICADO

El nombre "La Misión del Aprendiz" sugiere un propósito noble y una búsqueda de conocimiento. La palabra "misión" implica una tarea importante y significativa, mientras que la palabra "aprendiz" sugiere una persona que está en busca de conocimiento y habilidades para alcanzar un objetivo. La idea detrás de este nombre es que el aprendizaje es una misión continua en la vida. No se trata simplemente de adquirir conocimientos, sino de buscar constantemente la sabiduría y la experiencia para mejorar como individuos y alcanzar nuestras metas personales y profesionales. Además, la idea de una "misión" implica un sentido de comunidad y colaboración. En una misión, las personas trabajan juntas para alcanzar un objetivo común. En el caso de "La Misión del Aprendiz", la idea es que los estudiantes, mentores y educadores trabajen juntos en una comunidad de aprendizaje para lograr un crecimiento personal y una educación significativa. En resumen, "La Misión del Aprendiz" es un nombre que evoca un sentido de propósito, compromiso, comunidad y búsqueda constante de conocimiento. Es un nombre que puede inspirar y motivar a las personas a buscar el aprendizaje y el crecimiento personal de una manera significativa y emocionalmente satisfactoria.



ASIGNACION DE TAREAS

Con el objetivo de asignar eficientemente las tareas a cada miembro del equipo, hemos tomado en consideración las fortalezas individuales de cada integrante. Esto nos permitirá optimizar el desempeño de cada miembro y lograr una distribución efectiva de las actividades. A continuación, se detalla la asignación de tareas de la siguiente manera:

|  |  |
| --- | --- |
| **Integrante** | **Tarea** |
| Melina Ramirez | Diseño de los sprites que sean necesarios para el juego |
| Sofia Obejero | Programación y diseño del Juego |
| Walter tolaba | Diseño y creación de la pagina web |
| Gonzalo Walter | Diseño y creación de la pagina web |



HERRAMIENTAS

MOTOR GRÁFICO

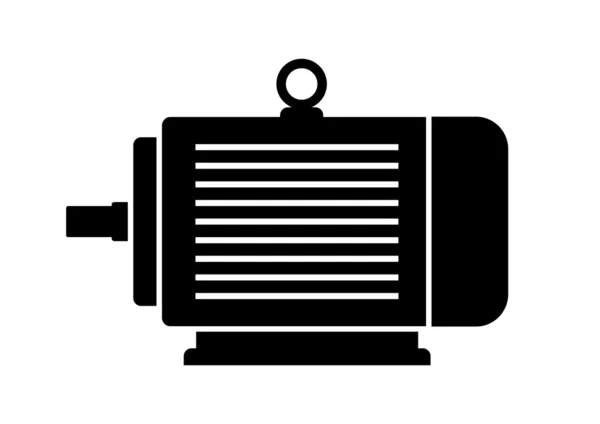
Inicialmente se había contemplado el uso del motor gráfico "Game Maker" para el desarrollo del proyecto. Sin embargo, debido a las limitaciones y restricciones de la versión gratuita de la aplicación, las pruebas del juego resultaron inestables y presentaron deficiencias estéticas que afectaban su apreciación.

Como solución, se tomó la decisión de cambiar el motor gráfico y optar por "Godot Game Engine". Este cambio se basó en la evaluación de su interfaz, la cual proporciona una experiencia mucho más cómoda y estética durante las pruebas del juego. Esto facilita la evaluación de la apariencia visual y la jugabilidad del juego de manera más efectiva.

La elección de "Godot Game Engine" se fundamenta en su capacidad para ofrecer una interfaz amigable y funcional, lo que contribuye a optimizar el proceso de desarrollo y pruebas del juego. La nueva herramienta permite una mayor flexibilidad y control en la creación de los aspectos gráficos y de jugabilidad, mejorando así la experiencia general del usuario.

Con esta decisión, se busca garantizar que el juego alcance los estándares de calidad deseados, tanto en términos de estabilidad como en su aspecto visual y jugabilidad. El cambio de motor gráfico proporciona una base más sólida para el desarrollo del proyecto y permite una mayor eficiencia en la fase de pruebas y ajustes del juego.

En resumen, la elección de "Godot Game Engine" como nuevo motor gráfico se basó en su interfaz más cómoda y estética, que facilita las pruebas y evaluaciones de apariencia y jugabilidad del juego. Esta decisión contribuye a mejorar la calidad y la experiencia global del proyecto.



GODOT GAME ENGINE

Godot Game Engine es un motor de desarrollo de videojuegos de código abierto y multiplataforma. Es una herramienta poderosa y versátil que permite a los desarrolladores crear juegos interactivos de alta calidad para una variedad de plataformas, incluyendo Windows, macOS, Linux, iOS, Android y más.

Godot ofrece un entorno de desarrollo integrado (IDE) con una interfaz intuitiva y fácil de usar. Proporciona un conjunto completo de herramientas para el diseño de juegos, la programación, la creación de gráficos, la animación, el manejo de físicas y la gestión de sonido. Los desarrolladores tienen acceso a un amplio rango de funcionalidades y características, lo que les permite materializar sus ideas creativas de manera eficiente.

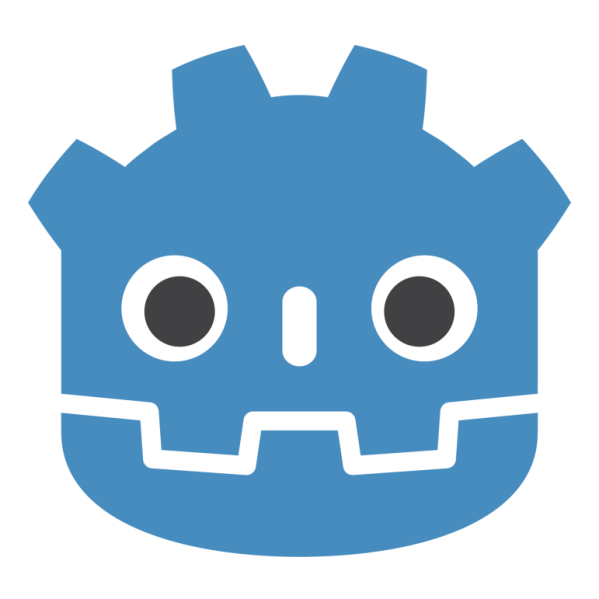
Una de las características más destacadas de Godot es su sistema de programación basado en nodos. Los nodos son objetos predefinidos que representan elementos del juego, como personajes, escenarios, efectos visuales, etc. Los desarrolladores pueden conectar y organizar estos nodos para establecer relaciones y definir el comportamiento de su juego. Esto permite una programación modular y flexible, lo que facilita la creación de juegos complejos y personalizados.

Godot también ofrece soporte para múltiples lenguajes de programación, incluyendo GDScript (su propio lenguaje de script similar a Python), C#, VisualScript y C++. Esto brinda a los desarrolladores la posibilidad de elegir el lenguaje que mejor se adapte a sus necesidades y experiencia.

Además, Godot se destaca por su enfoque en el rendimiento y la eficiencia. Está diseñado para maximizar el rendimiento de los juegos, aprovechando al máximo los recursos del sistema y optimizando el proceso de desarrollo.

Al ser un motor de desarrollo de videojuegos de código abierto, Godot fomenta la colaboración y la contribución de la comunidad de desarrolladores. Esto ha llevado a la creación de una amplia gama de recursos, tutoriales y complementos desarrollados por la comunidad, lo que enriquece aún más las capacidades y opciones disponibles para los desarrolladores.

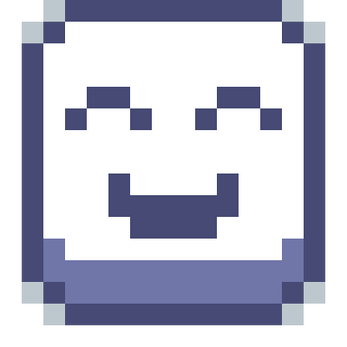
En resumen, Godot Game Engine es un motor de desarrollo de videojuegos de código abierto y multiplataforma que ofrece un conjunto completo de herramientas, un enfoque modular de programación basado en nodos y un rendimiento eficiente. Es una opción popular para aquellos que desean crear juegos interactivos de alta calidad con flexibilidad y facilidad.



PROGRAMA SELECCIONADO PARA EL DISEÑO DE SPRITES: LIBRESPRITE

Este programa es un software libre, gratuito y de código abierto diseñado para la creación y animación de sprites. Cuenta con una interfaz intuitiva y fácil de usar, ofrece herramientas y funciones avanzadas para dibujar y pintar, como pinceles, formas geométricas, degradados y capas, lo que facilita la creación de gráficos detallados y personalizados.

Ademas de estas herramientas, cuenta con características específicas para la animación de sprites. Permite a los usuarios crear fotogramas clave y definir la secuencia de animación para dar vida a los sprites, permitiendo alterar la velocidad de reproducción, la transparencia y aplicar efectos especiales para lograr resultados visuales llamativos. Cabe resaltar que el programa también admite una amplia variedad de formatos de archivos, permitiendo importar y exportar sprites en varios formatos(PNG, GIF, BMP,etc).



VISUAL CODE

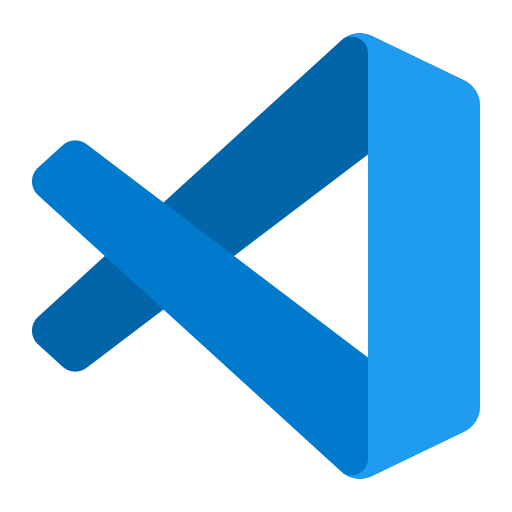
Visual Studio Code (VS Code) es un editor de código fuente desarrollado por Microsoft. Es una herramienta extremadamente popular y ampliamente utilizada por desarrolladores de software de todo el mundo. Se caracteriza por ser gratuito, de código abierto y altamente personalizable.

VS Code proporciona un entorno de desarrollo ligero pero potente que se adapta a una variedad de lenguajes de programación y plataformas. Es compatible con Windows, macOS y Linux, lo que lo hace accesible para una amplia gama de usuarios.

Una de las principales características de VS Code es su capacidad para admitir extensiones. Hay una amplia variedad de extensiones disponibles en el mercado, desarrolladas tanto por Microsoft como por la comunidad de desarrolladores. Estas extensiones amplían las funcionalidades de VS Code y permiten a los usuarios personalizar su entorno según sus necesidades específicas. Algunas de las extensiones populares incluyen aquellas para lenguajes de programación específicos, herramientas de depuración, integración con sistemas de control de versiones y mucho más.

Además, VS Code ofrece características como resaltado de sintaxis, finalización de código inteligente, navegación por el código, refactorización, depuración integrada, control de versiones y soporte integrado para Git, entre otros. También cuenta con una interfaz de usuario intuitiva y personalizable, lo que facilita su uso tanto para principiantes como para desarrolladores experimentados.

En resumen, Visual Studio Code es un editor de código fuente altamente flexible y personalizable que brinda a los desarrolladores un entorno de desarrollo completo para escribir, depurar y mantener su código. Su popularidad se debe a su versatilidad, rendimiento y la amplia gama de extensiones disponibles, lo que lo convierte en una opción preferida para muchos desarrolladores en la actualidad.



GIT / GITHUB

Para el desarrollo de nuestra página web, hemos tomado la decisión de utilizar el repositorio de GitHub como plataforma central para la gestión y almacenamiento de nuestro código fuente.

GitHub es un servicio basado en la nube que ofrece un sistema de control de versiones distribuido, específicamente basado en Git. Este sistema nos permite mantener un registro completo y organizado de todas las versiones de nuestro código a lo largo del tiempo, lo cual resulta fundamental para un proyecto de desarrollo web.

Al utilizar GitHub como repositorio, obtenemos numerosos beneficios. En primer lugar, nos brinda la capacidad de colaborar de manera eficiente con otros desarrolladores que participen en el proyecto, ya que permite la gestión de ramas (branches) para trabajar en paralelo y fusionar cambios de forma controlada.

Además, GitHub nos ofrece herramientas para llevar un seguimiento detallado de los problemas (issues) y solicitudes de cambios (pull requests), lo que facilita la comunicación y la resolución de problemas entre los miembros del equipo. También nos proporciona funcionalidades de control de acceso, lo que nos permite definir y gestionar los permisos de cada colaborador en el repositorio.

Otro aspecto valioso de GitHub es su integración con servicios de implementación continua (continuous integration), como Travis CI o Jenkins. Esta integración nos permite automatizar pruebas y despliegues, asegurando la calidad del código y agilizando el proceso de lanzamiento de nuevas versiones de nuestra página web.

En resumen, la elección de GitHub como repositorio para el desarrollo de nuestra página web nos brinda un conjunto completo de herramientas y funcionalidades que nos ayudarán a gestionar de manera eficiente nuestro código fuente, colaborar de forma efectiva y garantizar la calidad del proyecto en todas sus etapas.



AVANCES

ESTUDIOS

El juego tiene como objetivo enseñar a través de la sintaxis de Python, que es el primer lenguaje que los estudiantes de 4to año aprenderán en sus clases. Por lo tanto, decidimos estudiar a fondo el lenguaje de Python para hacer el desarrollo del juego más atractivo y relevante para ellos.

Además, para la creación de la página web, hemos ampliado nuestros conocimientos en los lenguajes de HTML, CSS y JavaScript. Estos lenguajes son fundamentales para el diseño y la funcionalidad de la página, permitiéndonos crear una experiencia interactiva y atractiva para los usuarios.

También hemos adquirido conocimientos en GDScript, el lenguaje propio de Godot Engine Maker. GDScript se asemeja a Python en cuanto a su sintaxis, lo que nos ha facilitado el desarrollo del juego en el motor Godot. Al estar familiarizados con la sintaxis de Python, nos resulta más fácil comprender y escribir el código en GDScript, agilizando así el proceso de desarrollo y optimizando la integración entre el juego y el motor gráfico.

En resumen, nos hemos capacitado en Python para poder enseñar los conceptos del juego de manera efectiva y hemos profundizado nuestros conocimientos en los lenguajes de HTML, CSS, JavaScript y GDScript para el desarrollo del juego y la creación de la página web. Esta amplia base de conocimientos nos ha permitido ofrecer una experiencia completa y coherente tanto en el juego como en la plataforma web.

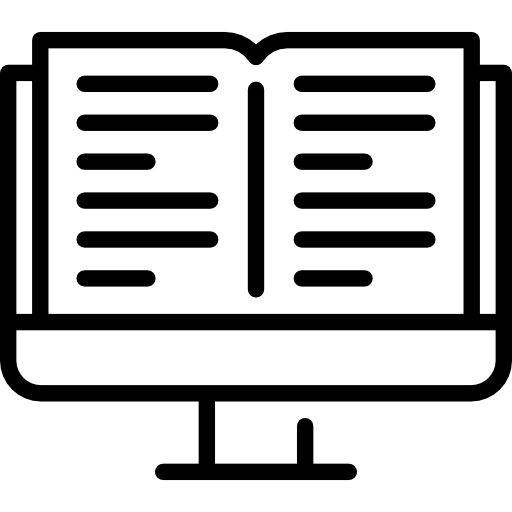


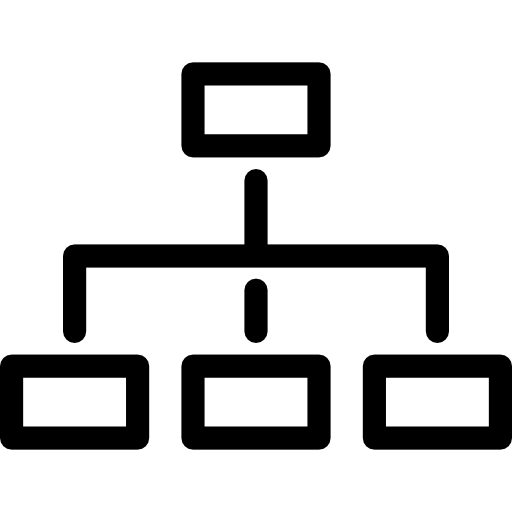
DIAGRAMA DE GANTT

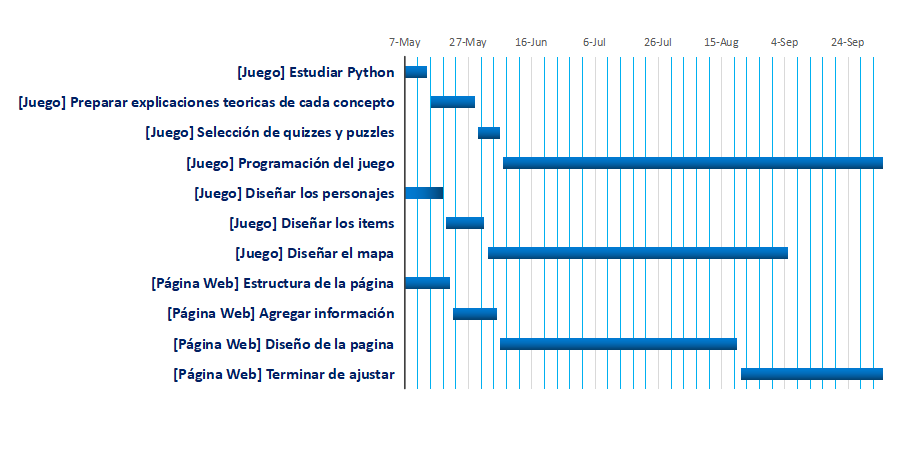
El diagrama de Gantt muestra de manera gráfica las tareas, su duración y la secuencia en la que se deben llevar a cabo. Esto permite visualizar claramente la relación entre las diferentes actividades y el tiempo asignado a cada una. Además, facilita la identificación de posibles solapamientos o retrasos que podrían afectar el cronograma general del proyecto.

El uso de esta herramienta tecnológica proporciona una visión global de las tareas a realizar, permitiendo una mejor asignación de recursos y una mayor eficiencia en la ejecución. También facilita la comunicación entre los miembros del equipo, ya que todos pueden consultar el diagrama de Gantt y comprender fácilmente las fechas y plazos establecidos.

Además, el diagrama de Gantt proporciona una base sólida para el seguimiento del progreso del proyecto, ya que se pueden registrar y actualizar periódicamente los avances y las fechas reales de finalización de cada tarea. Esto permite realizar ajustes y tomar decisiones informadas en caso de desviaciones o cambios en el plan original.

En resumen, el diagrama de Gantt es una herramienta esencial en la gestión de proyectos tecnológicos, ya que proporciona una estructura clara y visual para planificar, organizar y monitorear las actividades. Su uso contribuye a una mejor coordinación, eficiencia y control en el desarrollo del proyecto.

Con el fin de lograr una gestión eficiente de las actividades necesarias para el desarrollo del proyecto, se ha creado el siguiente diagrama de Gantt como una herramienta para mejorar la organización y planificación:



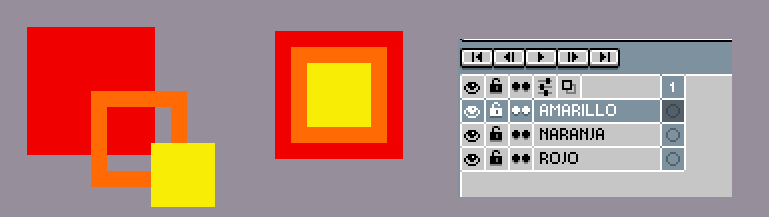
DISEÑO - JUEGO

PERSONAJE

1. **Instalación del programa y uso de sus herramientas:**

La primera etapa consistió en la descarga e instalación del programa y aprender a utilizar correctamente sus herramientas. Principalmente el uso de capas, las cuales permiten la manipulación y edición independiente de diferentes elementos de un sprite al momento de realizar animaciones.

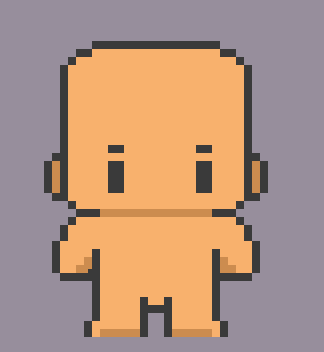
A continuación un muestra del uso de capas:

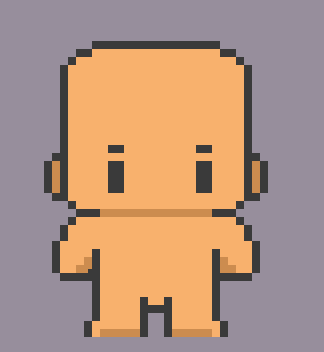
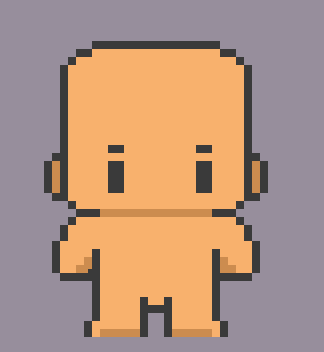


1. **Investigación de modelos de diseños de personajes para juegos:**

En esta segunda etapa, investigamos acerca de los diferentes tipos de modelos de personajes utilizados en juegos de rol o RPG. Mediante la vista y comparación de muchos diseños populares, se llego a la conclusión de que nuestro personaje buscaría tener un diseño simple pero atractivo a la vista.

A continuación una muestra del modelo base de diseño de nuestro personaje



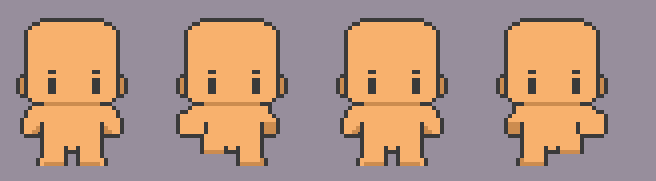


1. **Creación de animaciones de movimiento:**

En esta etapa se llevaron a cabo pruebas de animaciones para nuestro personaje utilizando la herramienta de creación de frames. Los frames consisten en una secuencia de imágenes individuales que, al ser combinadas, crean la ilusión de movimiento. Cada frame representa un momento específico en el que nuestro personaje adopta una determinada pose.

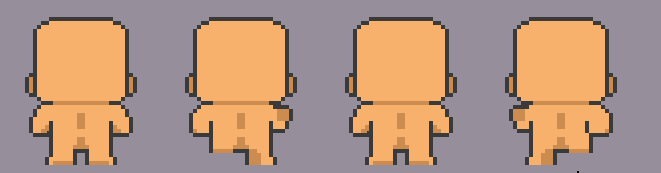
Por ejemplo, cuando el personaje da un paso al caminar hacia adelante, se observa que una pierna se adelanta a la otra, al igual que uno de los brazos se agranda o se hace mas notable. Esto se debe reflejar mediante una imagen diferente a la del personaje en reposo, seguida de una en la que la posición de la pierna y el brazo se intercambian.

A continuación, se presenta una muestra gráfica:

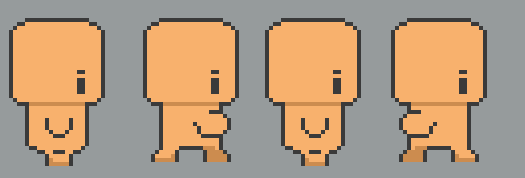


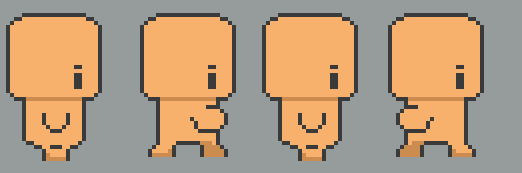
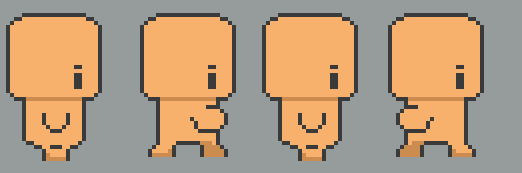
En el caso de caminar hacia atrás, una de las piernas se colocara frente al personaje, por lo que estará oculta a la vista, por lo tanto la otra pierna se vuelve mas notable. Esto mismo se aplica para los brazos del personaje, el brazo contrario a la pierna oculta se vuelve mas notable y el brazo contrario es menos visible

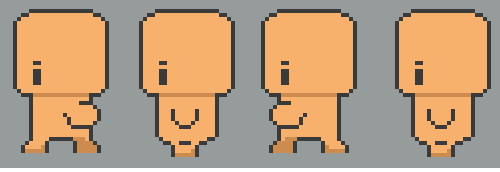
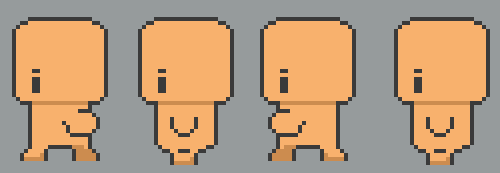
A continuación, se presenta una muestra gráfica:

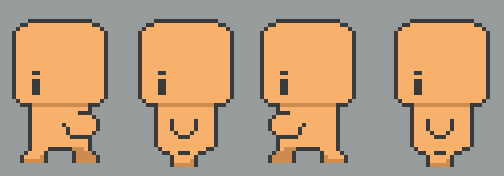
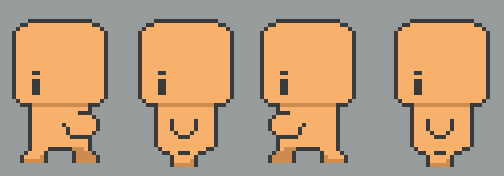


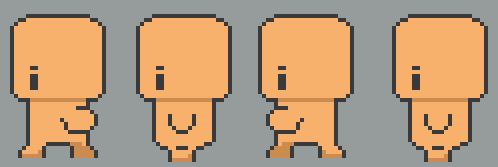
Para animar al personaje caminando hacia la derecha, se comienza con la imagen completa del personaje en perfil derecho, con las piernas juntas y los brazos estirados hacia el suelo. En la siguiente imagen, se mueven las piernas, mostrando una pierna hacia adelante y la otra hacia atrás, mientras que los brazos correspondientes se mueven en sentido contrario. Luego, se regresa al estado inicial y se intercambian las posiciones de brazos y piernas. A continuación se muestra esta secuencia de manera gráfica:



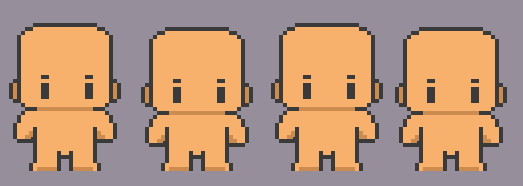
En el caso de animar al personaje mirando hacia la izquierda, se sigue el mismo procedimiento, pero con el personaje en perfil izquierdo. Se muestra la secuencia de los brazos y piernas que estaban ocultos en el perfil derecho. A continuación se muestra esta secuencia de manera gráfica:



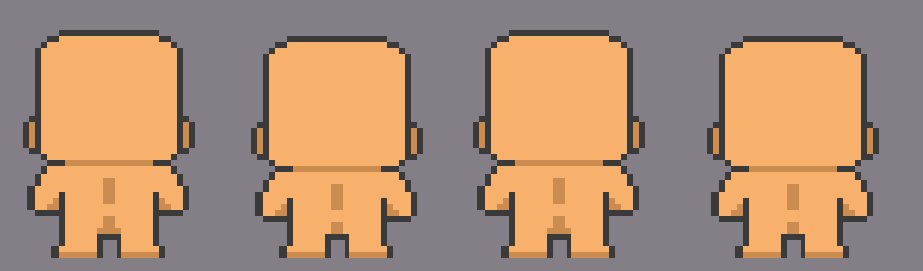


Para animar la secuencia de reposo de nuestro personaje es muy sencillo ya que las únicas partes del cuerpo que se encuentran en movimiento son la cabeza y los brazos. Representando un leve movimiento de exhalación, por lo que la cabeza se moverá unos pixeles hacia abajo al igual que los brazos, simulando movimiento de relajación.

A continuación se muestra esta secuencia de manera gráfica:



Esto mismo sucede al representar al personaje descansando desde la vista trasera, a continuación se muestra esta secuencia de manera gráfica:



1. **Diseño del personaje:**

Al establecer la plantilla base a utilizar para la creación y animación de nuestro personaje, se realizaron varias pruebas de diseño, con el objetivo de que el personaje posea un aspecto simple pero llamativo para el usuario.

A continuación se muestra los modelos de diseño:

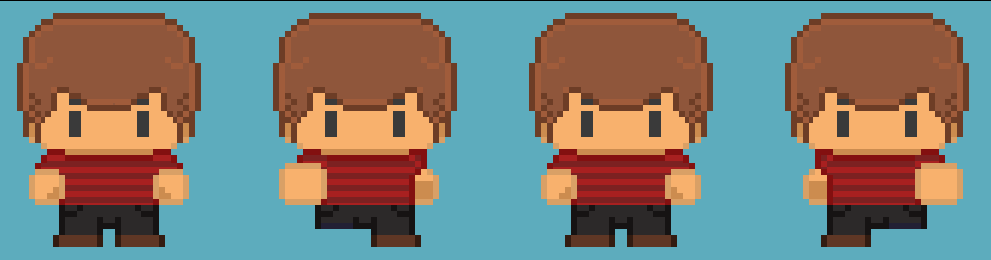


De los diseños mostrados anteriormente se determino que el mejor por el momento es el siguiente:

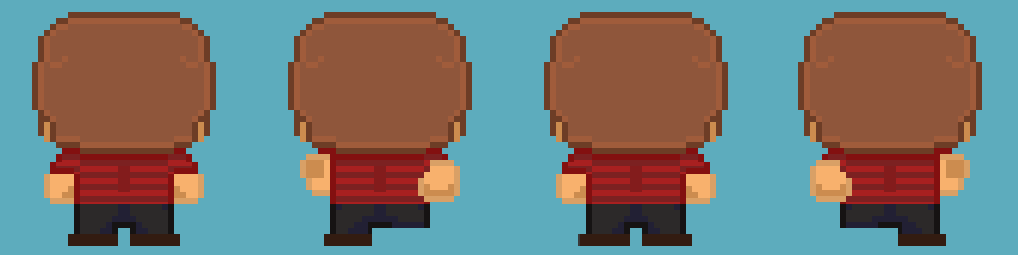


A continuación se muestra las secuencias de movimiento con el aspecto de nuestro personaje:

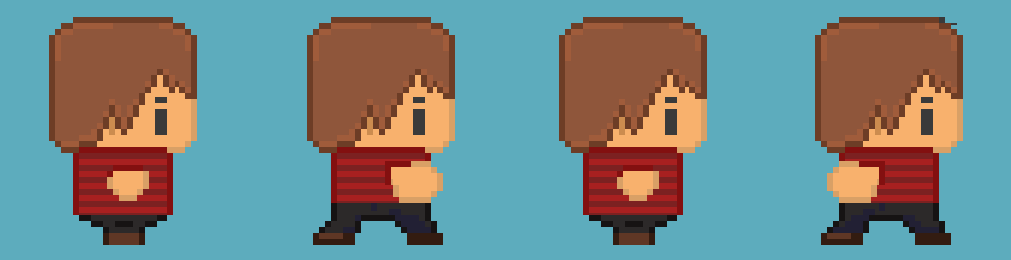
* **Personaje caminando hacia el frente:**



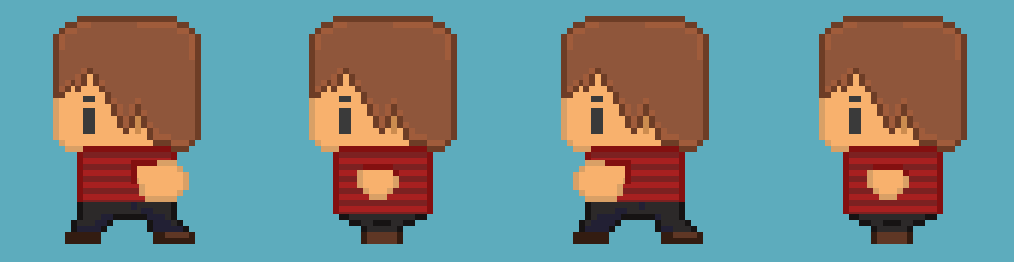
* **Personaje caminando de espaldas:**



* **Personaje caminando hacia la derecha:**



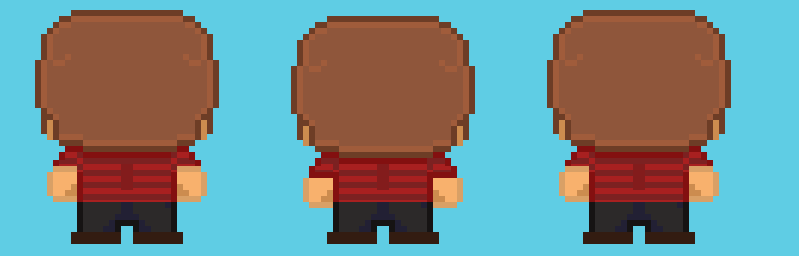
* **Personaje caminando hacia la izquierda:**



* **Personaje en reposo:**



* **Personaje en reposo de espaldas:**

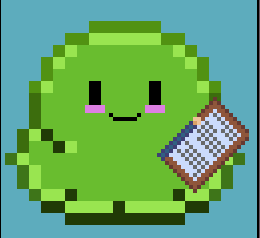


1. **Diseño del guía:**

Para realizar el diseño del personaje encargado de acompañar y guiar al alumno durante el transcurso de los niveles, decidimos utilizar un diseño distintivo capaz de capturar la atención del jugador de manera instantánea, que tenga un aspecto simple y, a la vez, amigable. Lo llamamos Limo.

Además, tenemos la intención de usar en el futuro este personaje como símbolo distintivo, que nos identifique y diferencie de los demás juegos.

En base a esto, realizamos el siguiente diseño:

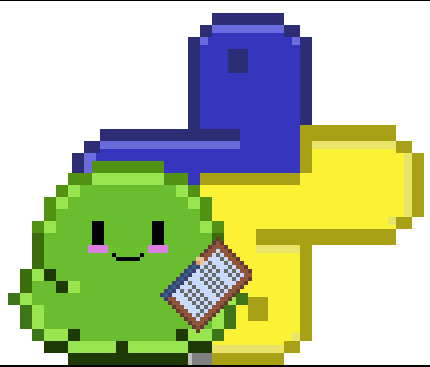


Las animaciones correspondientes a nuestro guía son las siguientes:



1. **Diseño del logo:**

Como mencionamos anteriormente, para realizar nuestro logo utilizaremos a nuestro guía, Limo. Ademas agregamos el logo de Phyton debido a que es el lenguaje en el que vamos a enseñar a programar.



1. **Diseño de ítems:**

Lo siguiente a realizar fueron los ítems que ganara nuestro personaje al superar los distintos niveles o tareas que se le presenten. En total son tres ítems principales, uno por cada unidad de estudio.

Los Ítems que hemos elegido fueron los siguientes:

* **Galletas de gato**



* **Bebida energética**



1. **Diseño de mapas:**

El diseño de mapas para nuestro juego es una parte crucial de nuestro proyecto. Hemos tomado la decisión de no replicar completamente la estructura de la Escuela Técnica N°5 en el juego debido a preocupaciones de seguridad para el establecimiento. En su lugar, hemos optado por utilizar únicamente la vista básica de la escuela, omitiendo áreas como salones, auditorio, comedor, etc.

A continuación, detallamos las áreas que conformaran nuestro mapa completos:

* **Diseño de la Entrada:**

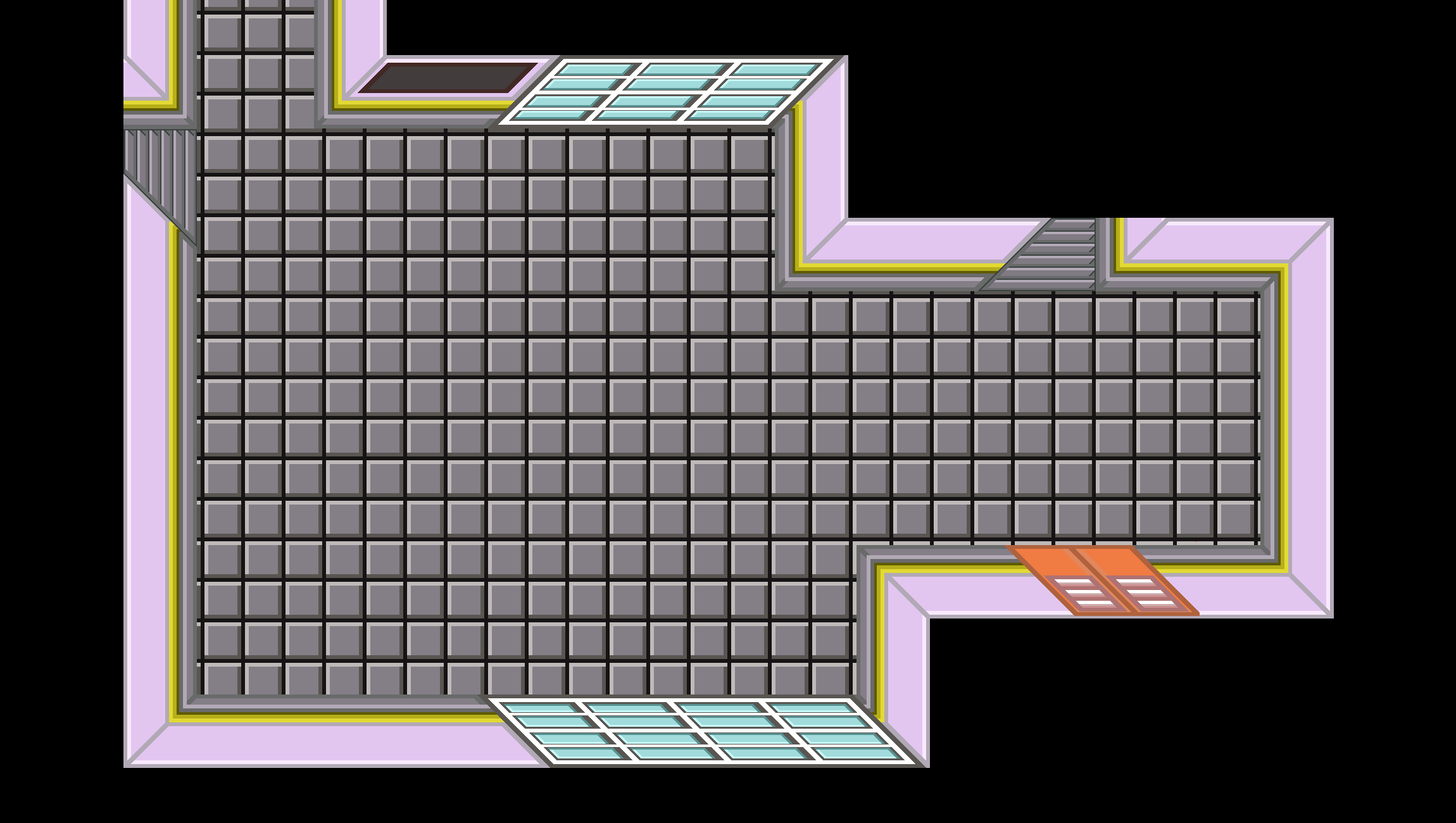
En esta sección, representamos la entrada principal al establecimiento, que también es el punto de inicio del juego para el jugador. Para crear este mapa, dividimos los elementos en diferentes capas para facilitar su manipulación.

Las capas utilizadas incluyen:



Las medidas que se utilizo par a realizar el mapa fueron de 2300 pixeles de ancho y 1300 pixeles de alto.

Aquí presentamos el primer modelo de mapa creado para la primera muestra de avances:



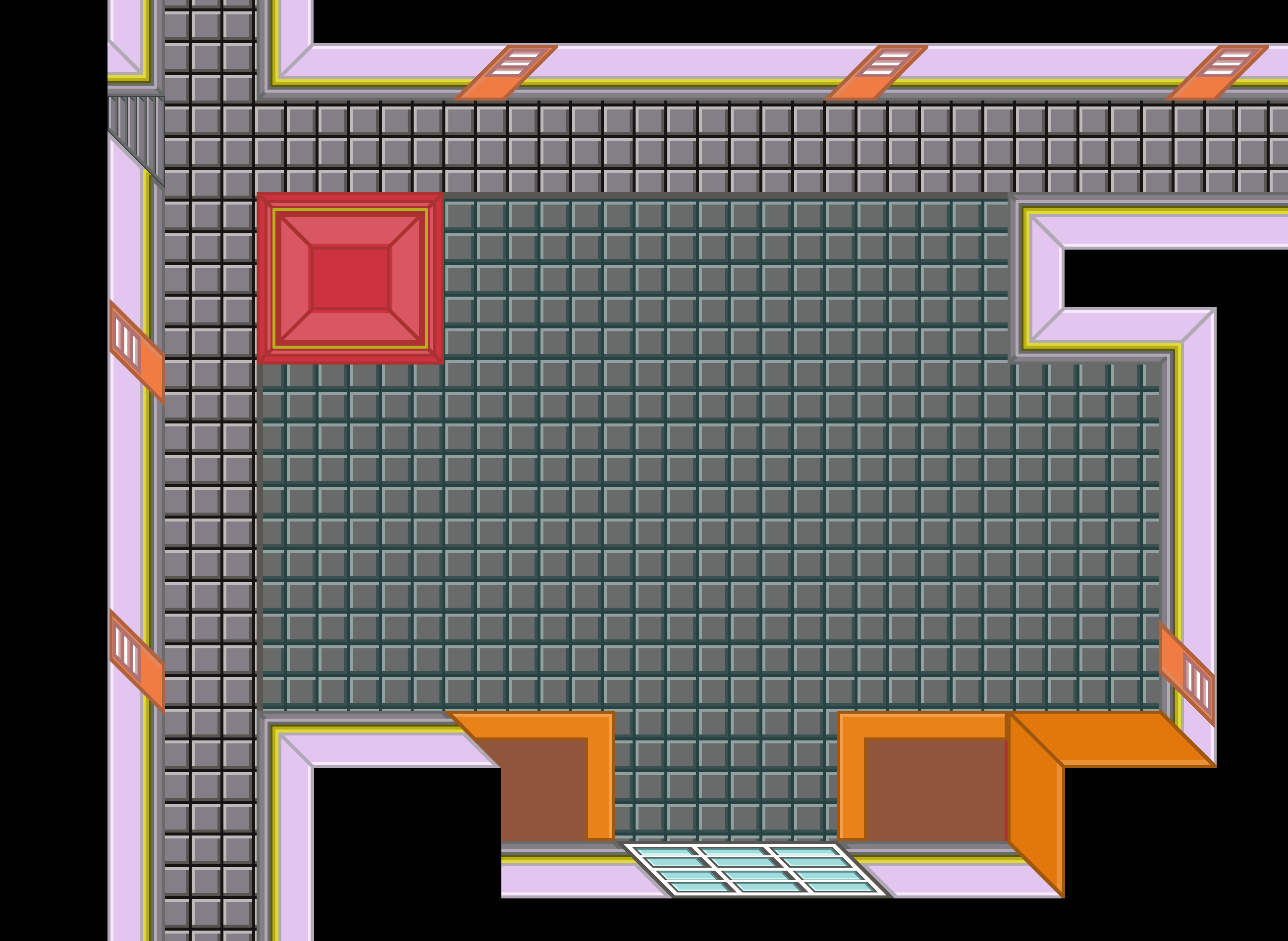
* **Diseño del Patio Central:**

El patio central es un espacio simple por el cual el jugador debe pasar para llegar a las aulas. Al igual que en el mapa de entrada, utilizamos capas para facilitar la manipulación de elementos. Las capas en este mapa son:



Las medidas que se utilizo par a realizar el mapa fueron de 2600 pixeles de ancho y 1900 pixeles de alto.

A continuación se muestra el primer modelo de mapa que se realizo para la primera muestra de avances:

****

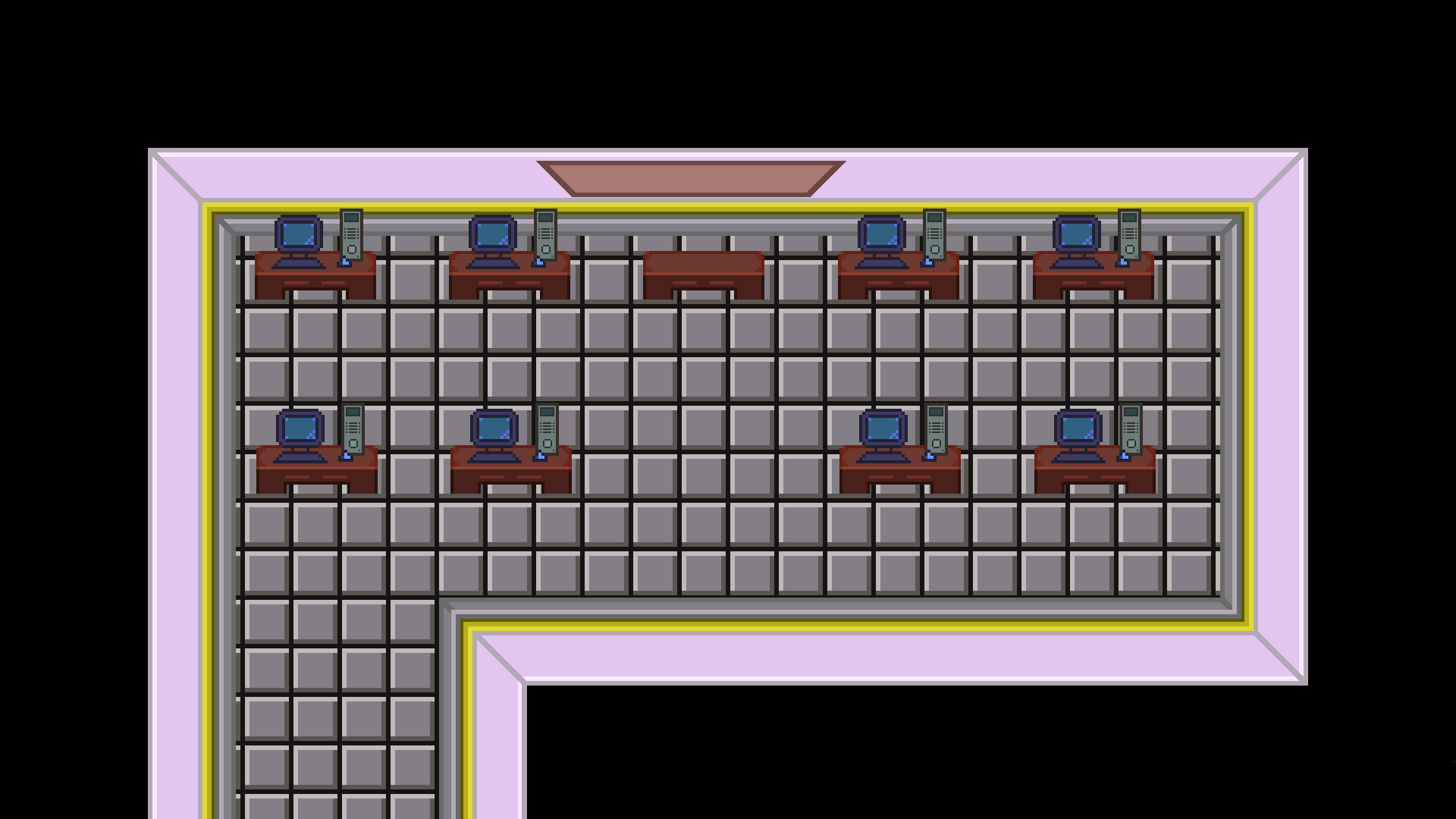
* **Diseño de la Sala de Computo:**

La sala de computo es esencial en nuestro juego, ya que es donde el personaje del jugador llevará a cabo diversas actividades utilizando las computadoras disponibles. Para este mapa, se utilizó un número menor de capas en comparación con los anteriores. Las capas incluyen:



Las medidas que se utilizo par a realizar el mapa fueron de 1920 pixeles de ancho y 1080 pixeles de alto.

Este es el primer modelo de mapa que se realizo para la primera muestra de avances:

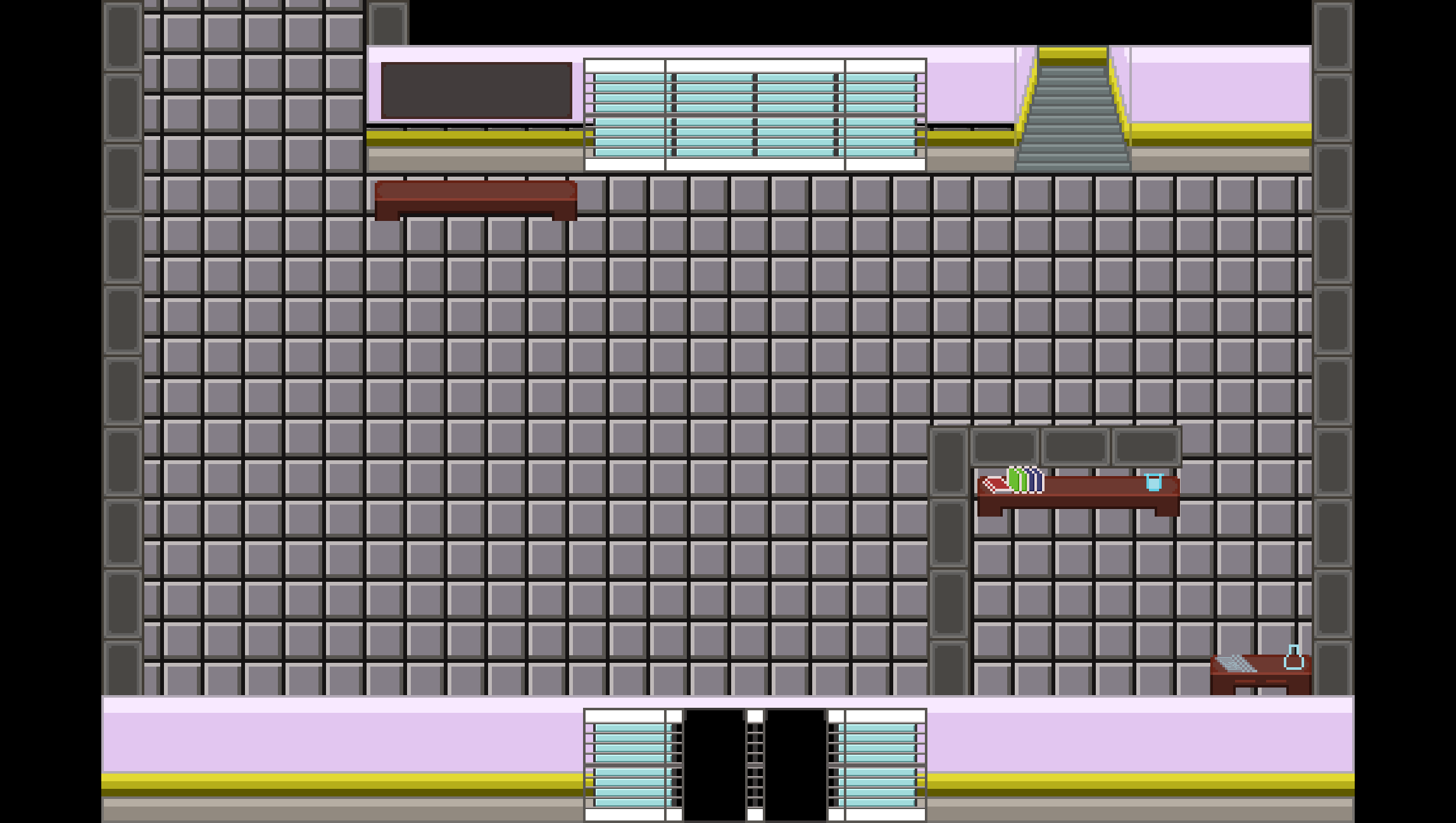
****

* **Actualizaciones:**

Debido a incongruencias entre las perspectivas de nuestros mapas de prueba decidimos cambiar la apariencia de estos, reutilizando las anteriores medidas, las capas y objetos presentes en estos.

A continuación se muestran el diseño de nuestros mapas finales:

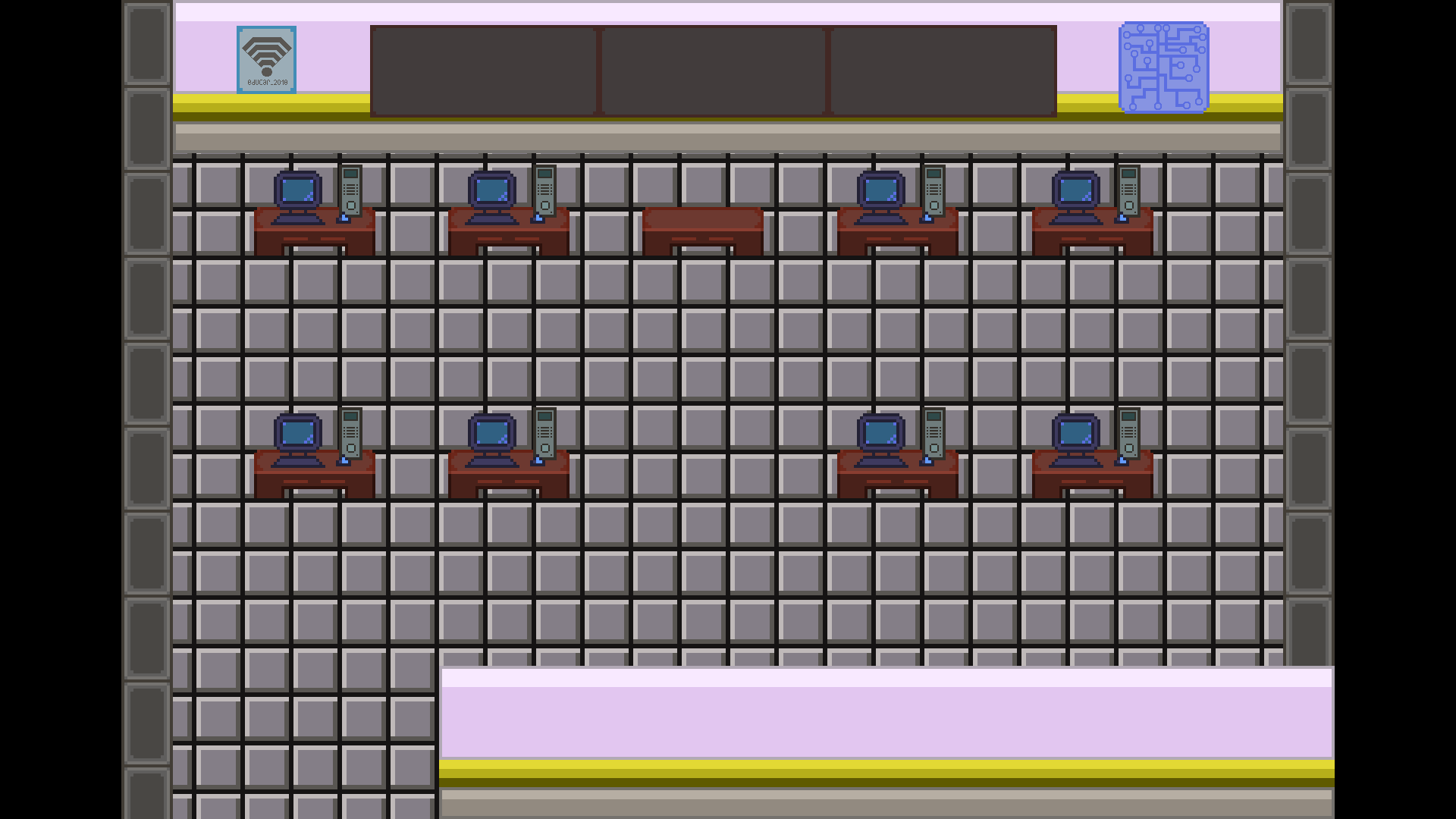
* **Diseño de la Entrada:**

****

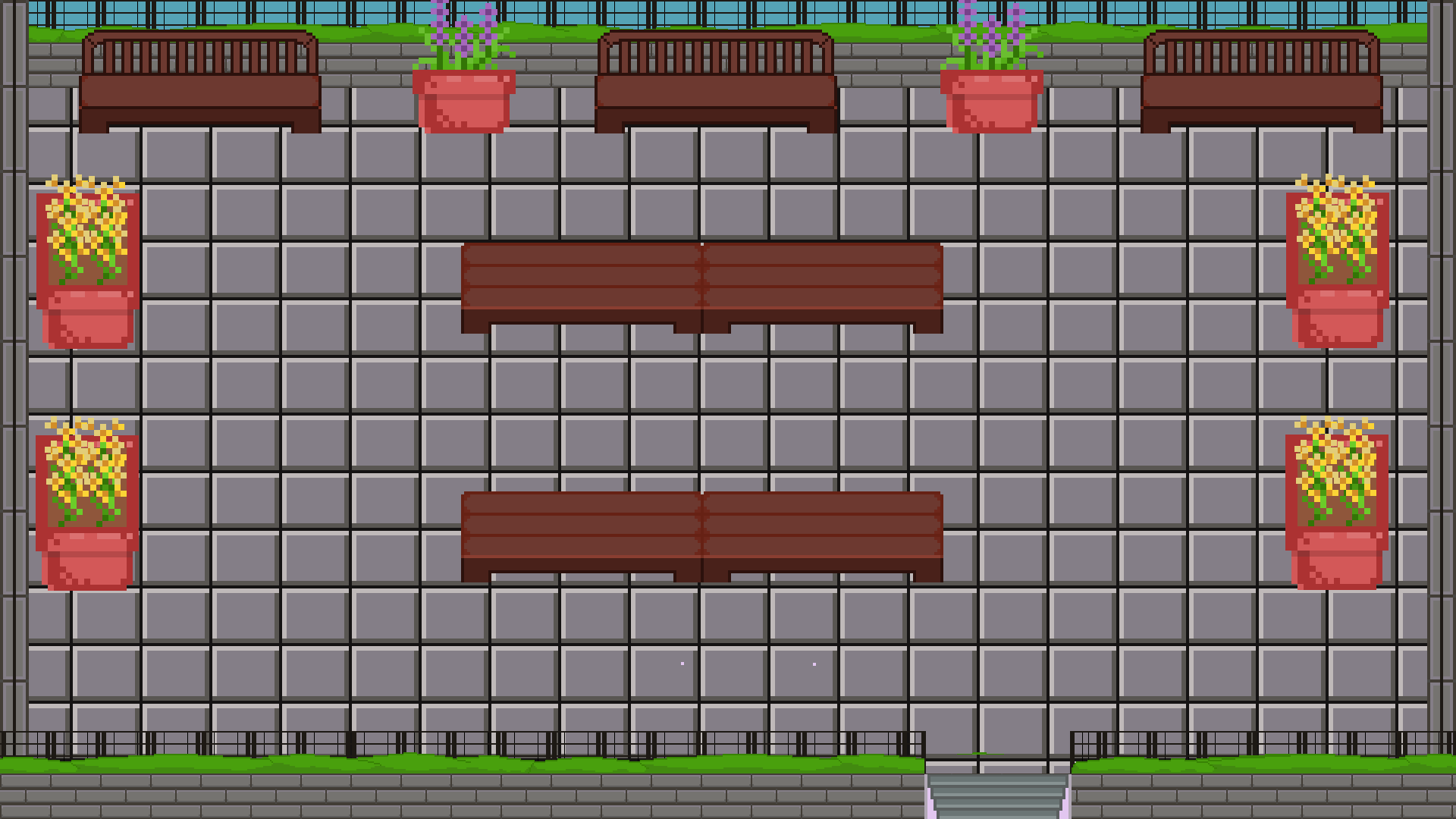
* **Diseño del Patio Central:**

****

* **Diseño de la Sala de Computo:**

****

* **Diseño de salón ultimo nivel:**

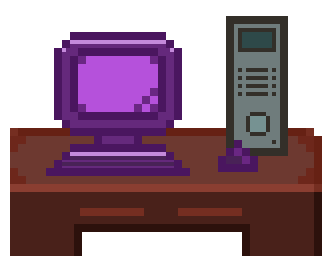
****

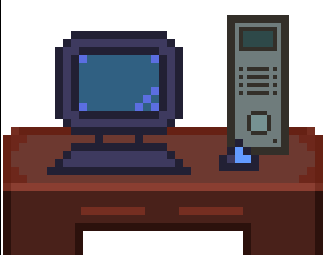
1. **Objetos para los mapas:**

Para rellenarlos mapas y crear un ambiente atractivo a la vista se crearon distintos objetos con animaciones sencillas.

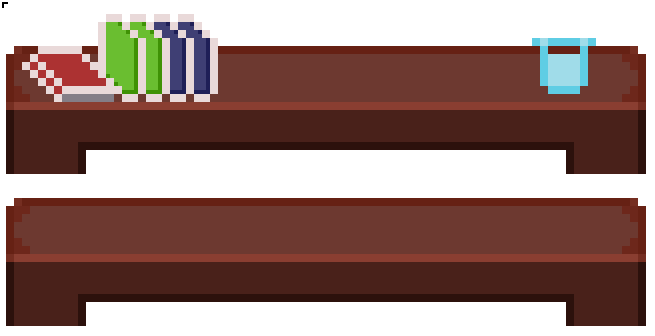
A continuación se mostraran los objetos:

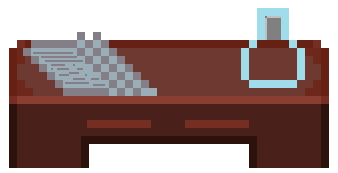
* **Computadores:**

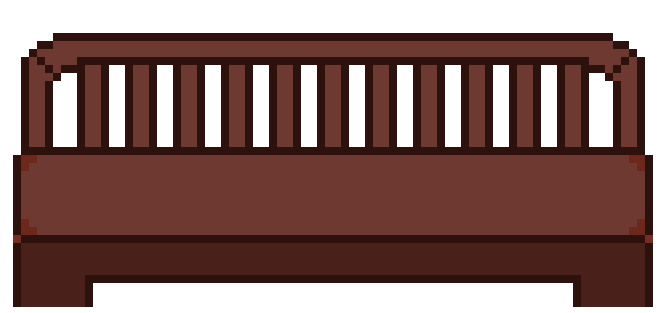




* **Mesas y sillas:**







* **Gatos:**





* **Macetas:**



1. **Menú:**

PAGINA WEB

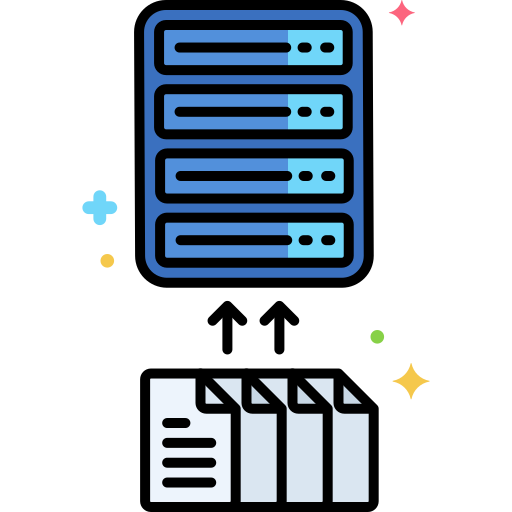
CREACION DEL REPOSITORIO

Un repositorio web es una herramienta de gestión y almacenamiento de código fuente utilizada en el desarrollo de software. Se trata de un sistema que permite almacenar, controlar y organizar las versiones de los archivos y recursos utilizados en un proyecto de desarrollo de software.

En un repositorio web, los archivos y recursos son almacenados en un servidor centralizado al que los desarrolladores pueden acceder y realizar cambios de manera controlada. Estos cambios son registrados y documentados, lo que facilita el seguimiento de la evolución del proyecto a lo largo del tiempo. Además, un repositorio web permite a los desarrolladores trabajar de forma colaborativa, ya que múltiples personas pueden acceder al repositorio, realizar modificaciones y fusionar sus cambios de manera ordenada.

Uno de los sistemas de control de versiones más populares utilizado en los repositorios web es Git. Git permite el control de versiones distribuido, lo que significa que cada desarrollador puede tener una copia completa del repositorio en su máquina local y realizar cambios sin necesidad de una conexión constante a un servidor central. Esto proporciona mayor flexibilidad y autonomía a los desarrolladores.

El uso de un repositorio web proporciona beneficios significativos en el desarrollo de software, como el seguimiento histórico de cambios, la colaboración eficiente entre desarrolladores, la posibilidad de revertir cambios no deseados y la gestión efectiva de ramas y versiones paralelas del proyecto. En resumen, un repositorio web es una herramienta esencial en el proceso de desarrollo de software, que facilita la organización, control y colaboración en el trabajo con el código fuente de un proyecto.

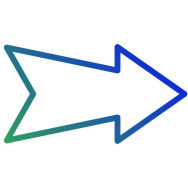


PROGRMACION DEL JUEGO

En un principio, el juego estaba destinado a ser desarrollado en el motor gráfico "Game Maker". Por lo tanto, dedicamos tiempo a estudiar dicho motor gráfico, realizando proyectos pequeños, como juegos sencillos, como un juego de plataformas corto y una imitación de Pac-Man, con el fin de familiarizarnos con el entorno de esta herramienta que nos permitiría desarrollar el juego.

Sin embargo, después de una investigación más profunda sobre los motores gráficos existentes de uso libre y gratuito, decidimos optar por el motor gráfico "Godot". Este motor es portable, multiplataforma y lo que lo hizo nuestra elección final, en relación a lo anteriormente mencionado, es que a diferencia de "Game Maker", permite exportar y realizar pruebas del juego en una aplicación de escritorio sin necesidad de pagar por este servicio. Además, cabe mencionar que la posibilidad de exportarlo en HTML nos permite incluir demos en la página web del proyecto.





Para iniciar el proceso de familiarización con las herramientas que ofrece Godot, llevamos a cabo un estudio y la creación de proyectos sencillos. Esto nos permitió adquirir un entendimiento de la funcionalidad de los nodos, así como de las distintas funciones y señales específicas de cada nodo. Asimismo, nos adentramos en la creación de escenas y en la estructura jerárquica del árbol de ejecución dentro de Godot.

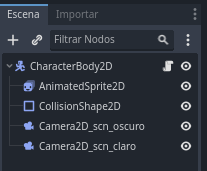
PERSONAJE

Para la creación del personaje, empleamos los siguientes nodos nativos de Godot, cada uno de los cuales posee una funcionalidad específica que contribuye a un atributo distinto del nodo principal, el cual representa al personaje en sí. El nodo que utilizamos como principal es:

* **CharacterBody2D**: representa el cuerpo de personaje en un entorno 2D. Es un componente especializado que facilita el movimiento, la física y la interacción del personaje en el motor de juego.

Este tiene como hijos los nodos:

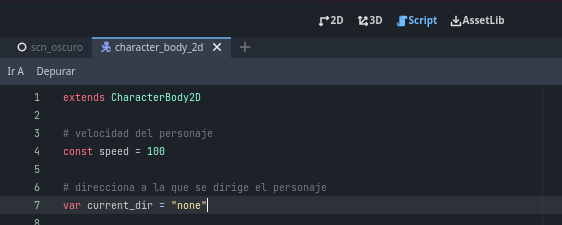
* **AnimatedSprite2D**: es un componente que permite reproducir animaciones en un entorno 2D, utilizando una secuencia de imágenes o frames para crear una animación fluida y visualmente atractiva.
* **CollisionShape2D**: es un componente utilizado para definir la forma y el tamaño de un objeto que participa en la detección de colisiones en un entorno 2D, permitiendo interacciones físicas y detectar colisiones con otros objetos en el juego.
* **Camera2D**: es un componente que simula una cámara en un entorno 2D, permitiendo controlar la perspectiva y el enfoque de la vista del juego, y facilitando el seguimiento de objetos y la implementación de efectos visuales como el desplazamiento suave y el zoom.



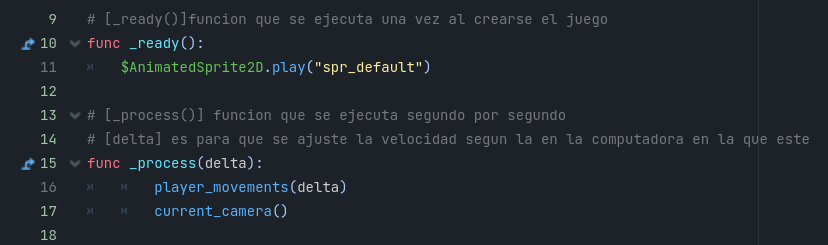
La imagen muestra la estructura del personaje. Basándose en la explicación previa de cada nodo, el personaje está compuesto por un nodo CharacterBody2D como nodo principal, el cual tiene como nodos hijos un AnimatedSprite2D para proporcionar su apariencia visual, un CollisionShape2D para permitir colisiones con objetos y límites del mapa, y una cámara que sigue al personaje en todo momento durante su recorrido en el mapa.

MOVIMIENTOS DEL PERSONAJE

En el código propio del nodo principal, se ha declarado una variable para almacenar la velocidad a la cual el personaje se desplazará, así como otra variable para almacenar la dirección hacia la cual el personaje se está dirigiendo, para luego utilizar esta como guía para crear la animacion del personaje.



Es relevante destacar que Godot cuenta con dos funciones integradas de importancia. Una de ellas es la función **‘\_ready()’**, que se ejecuta una única vez al iniciar el programa, y la otra es la función **\_process()**, que se ejecuta de forma periódica, es decir, segundo a segundo, durante la ejecución del programa. Por lo que, dentro de la función **‘\_process’** se ejecutó la función ‘**player\_movement’**, la cual verifica la entrada por teclado para identificar en que dirección deberá moverse el personaje, y finalmente moverse con la función integrada de Godot **‘move\_and\_slide()’.** Se incluyo una nacionalización que permitirá que la velocidad del juego, en frames por segundo, sea la misma en cualquier computadora, así como también controlara que el jugador no pueda moverse a mayor velocidad en diagonal.



Código del la función **‘player\_movements’**:

Verifica la entrada por teclado, declara a que dirección se dirige el personaje en la variable **‘current\_dir’**, y mueve al personaje en la dirección correspondiente.

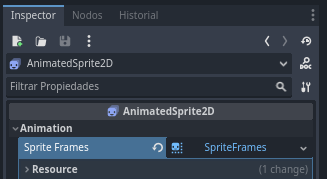


Normalización y movimiento:



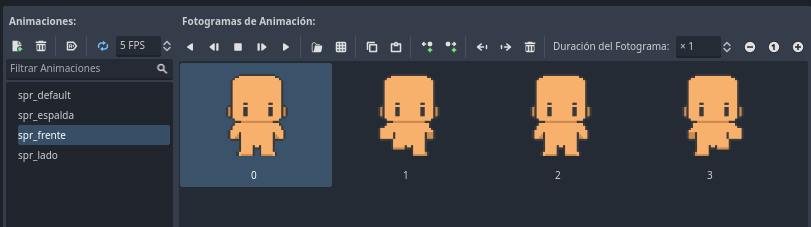
ANIMACION

Para animacion, desde la propiedad dentro del inspector correspondiente al nodo **‘AnimatedSprite2D’**,accediendo a la opción **Animation > SpriteFrames**

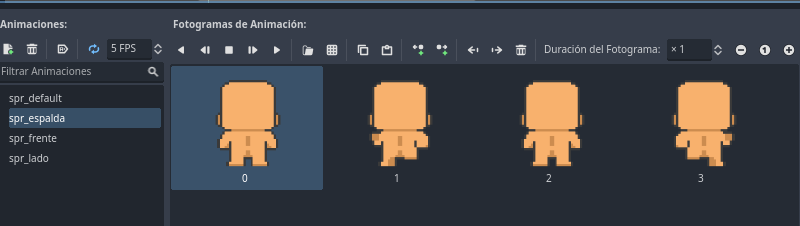


En esta ventana se crearan las animaciones que variaran según ala dirección a la que se mueva el personaje:

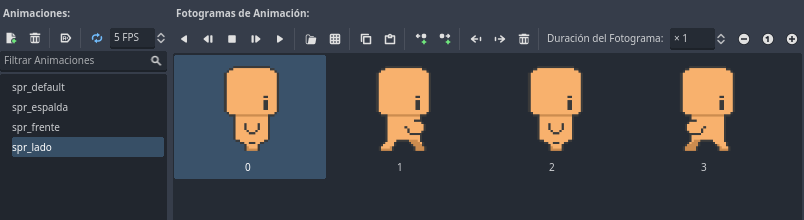
* Frente:

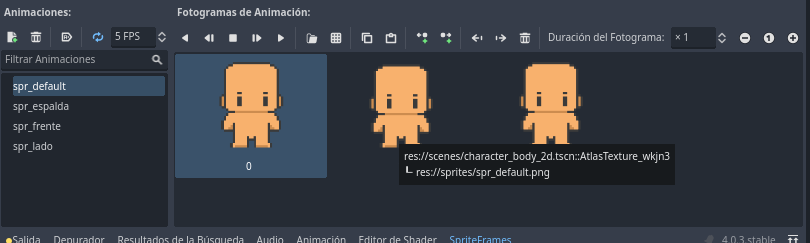


* Espalda:

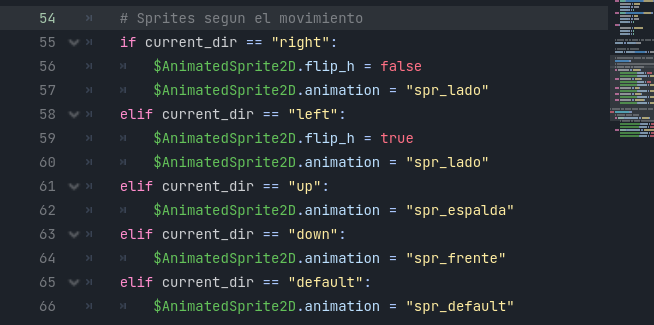


* Lado:

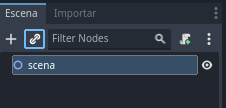
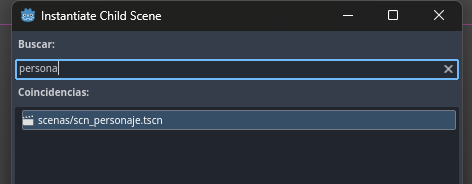


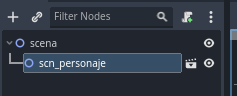
* En espera:

Dentro de la función **‘player\_movement’** se cambiara la apariencia del personaje dependiendo la dirección a a la que dirija con el siguiente código:



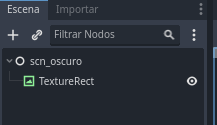
Esta escena del personaje se puede incorporar a diversas escenas de manera similar a como se hace con los nodos, instanciandola, lo que nos permite programarlo una sola vez y utilizarlo las veces que sean necesarias.



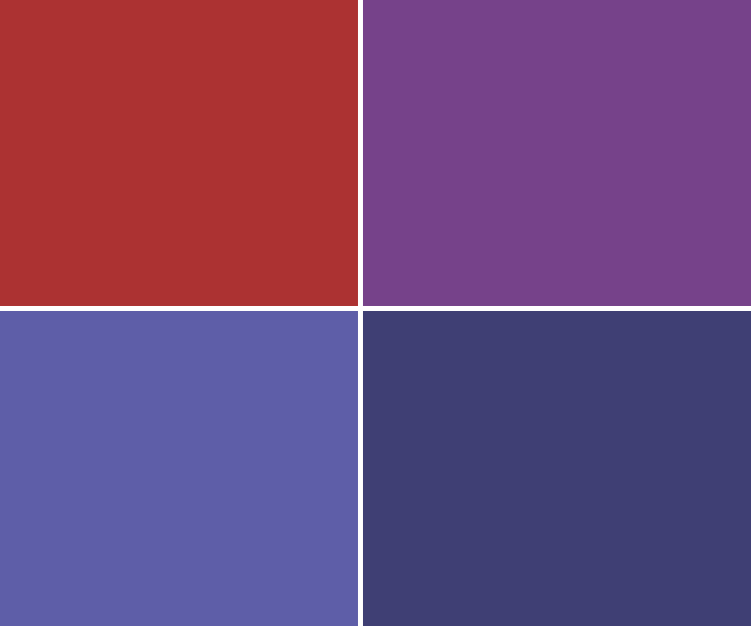


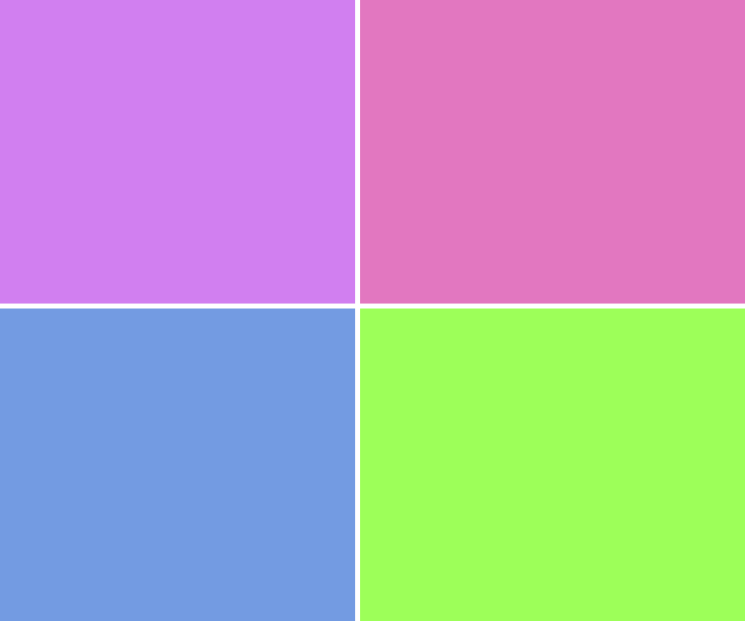
AGREGAR UN MAPA ESTATICO

El nodo necesario en una escena para asignarle una apariencia estática es el ‘**Texture\_rect’** que es el que utilizamos para las pruebas pruebas del personaje.



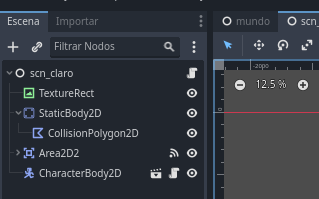
Para notar el cambio de escena, en la prueba del personaje, utilizamos las siguientes plantillas:



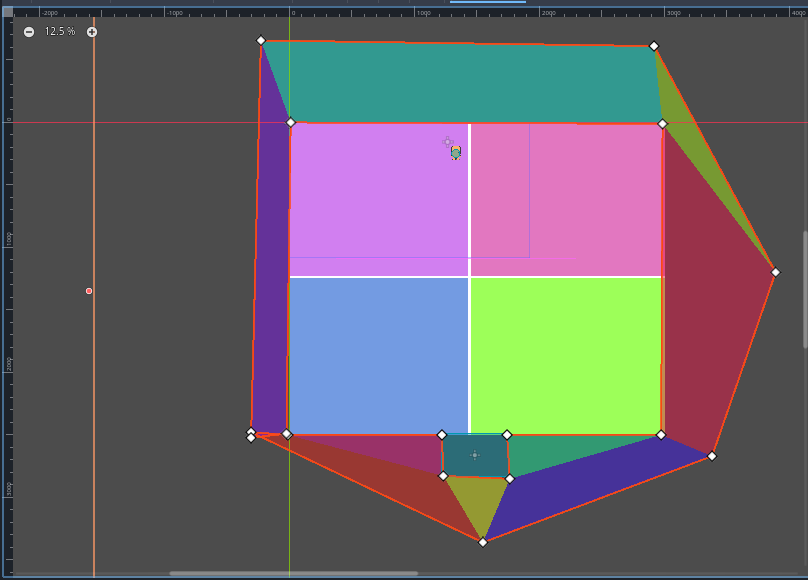


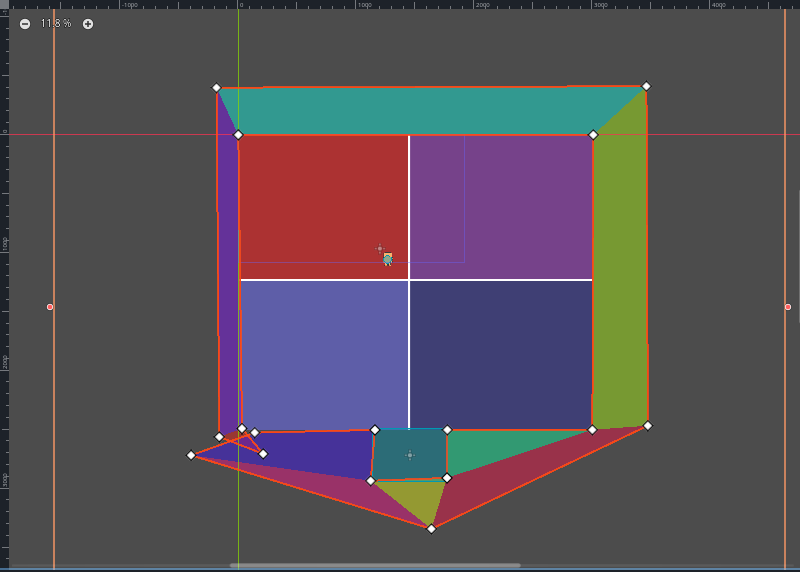
COLISIONES CON EL MAPA

El nodo que utilizamos para crear las colisiones con el mapa fue ‘StaticBody2D’ que tiene como hijo al nodo ‘CollisionPolygon2D’



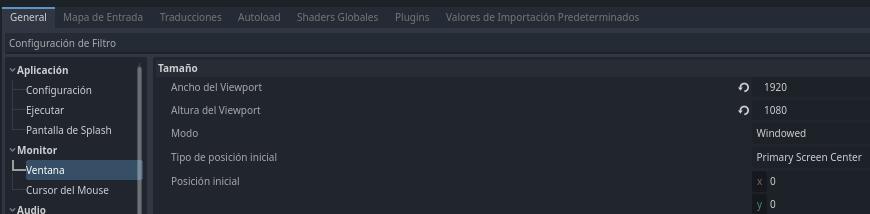
El mapa con el limite definido por la colisión del nodo **‘CollisionPolygon2D’**:





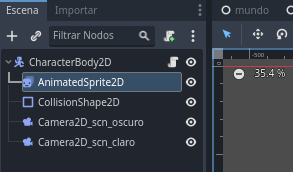
TAMAÑO DEL VIEWPORT

Desde la configuración del proyecto decidimos como medida del viewport una dimensión de 1080px de alto por 1920px de ancho, que por defecto sera el tamaño de la cámara.



AGREGAR UNA CAMARA QUE SIGA AL JUGADOR

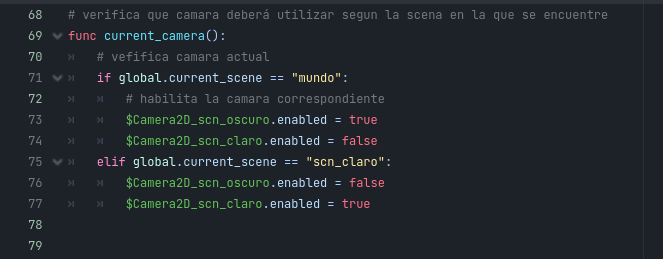
Para poder hacer esto utilizamos los nodos **‘Camera2D’** los cuales son hijos del nodo principal del personaje ya que es a este al que debe seguir. Y definiremos sus limites segun corresponda desde el inspecctor propio del nodo.





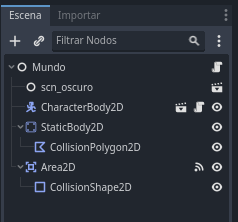
CAMARA SEGUN EL MAPA

Dado que no todos los mapas tendrán la misma medida, vimos correspondiente el asignar una cámara a cada mapa. El siguiente código identifica en que mapa se encuentra el jugador para asignarle una cámara:



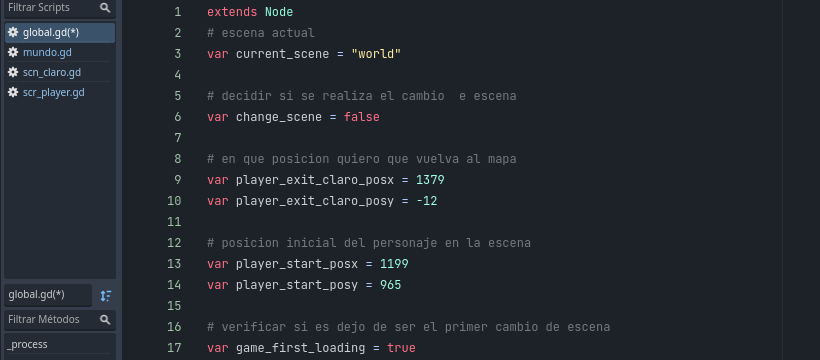
CAMBIO DE ESCENA

Para llevar a cabo la transición de escena, empleamos el nodo **'Area2D'** con el propósito de delimitar precisamente el área que permitirá dicha transición. Este nodo se encuentra anidado bajo el nodo **'CollisionPolygon2D'**, el cual posibilita la colisión entre el personaje y el área, desencadenando así el cambio de escena.

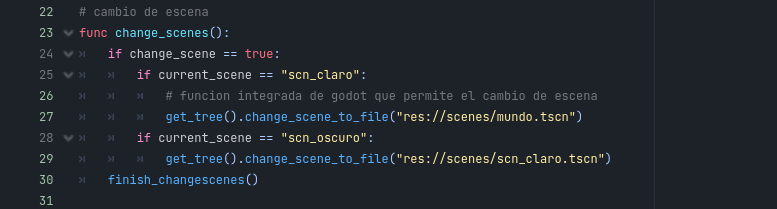


Código para realizar el cambio de escena. Este se encunetra en un scrip apartado global.

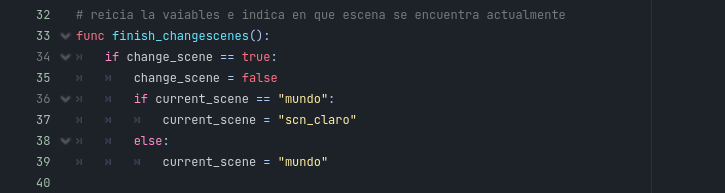
Definición de variables:



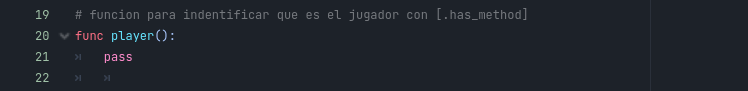
Función para el cambio de escena:



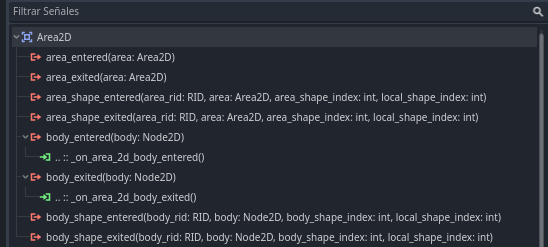
Función para reiniciar variables:



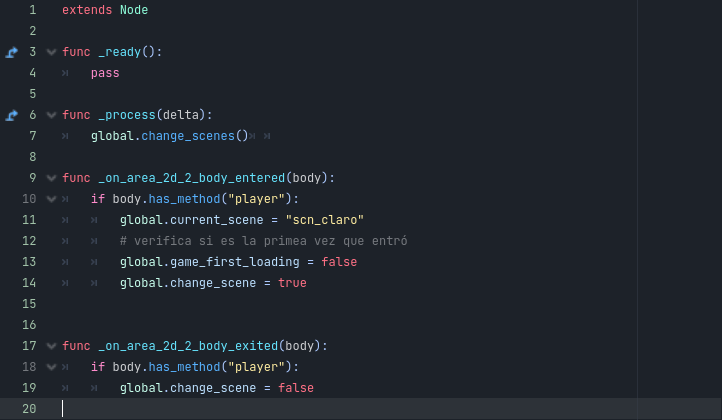
En el código del jugador agregamos una función que tiene como único fin, identificar al nodo del personaje como tal, cuando ingrese a el área que se encuentra designada para realizar el cambio de escena.



Conectamos dos funciones propias del nodo **´Area2D’** al las escenas donde se realizaran los cambios:



Código dentro de las funciones:



ESCENAS FINALES DEL JUEGO

AUTOLOADS

Los Autoloads en Godot son una característica que permite cargar y acceder a script y objetos de manera automática y global en todo el juego. Son utiles para gestionar reecursos compartidos, variables globales, o cualquier tipo de logica que se necesite acceder desde diferentes partes del proyecto sin tener que crear instancias manualmente.

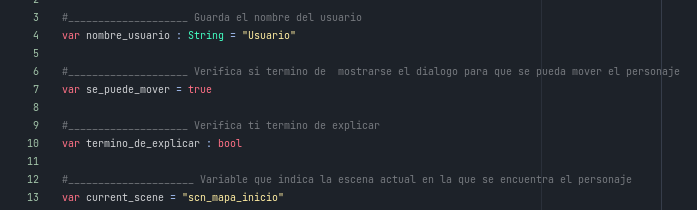
En el proyecto de ‘La Misión del Aprendiz’, utilizamos 3 autoloads:

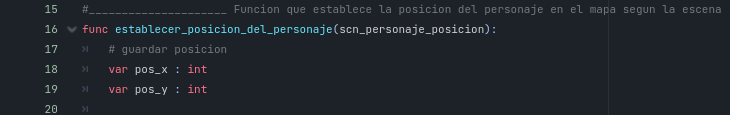
* Global
* ControladorAudio
* ControladorMusica
* DialogManager
* Transición



GLOBAL

En el ‘Global’ establecimos multiples variables globales, junto con funciones relacionadas con el cambio de escena entre otras funcionalidades:

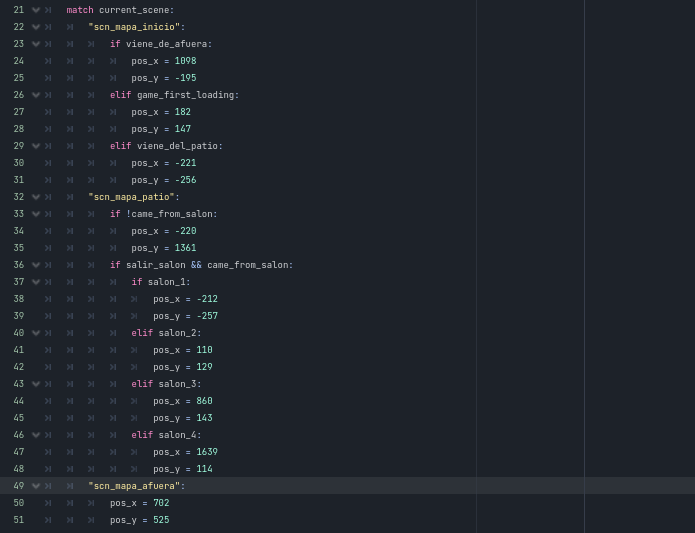




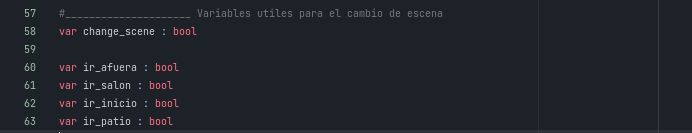
En la misma función establecemos la posicion, en un inicio esta era la estructura, ya que fue construida sobre la marcha, sin muchos conocimientos, y con el fin de ser lo mas funcional posible, sin tomar en cuenta su eficiencia:



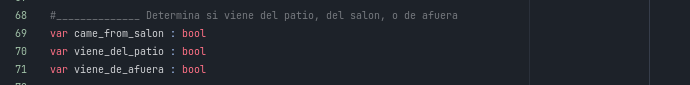
Y terminamos cambiandola por la siguiente:



**Variables**:

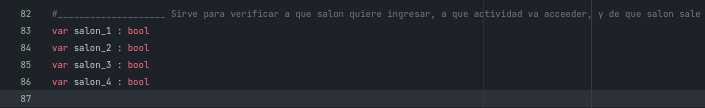






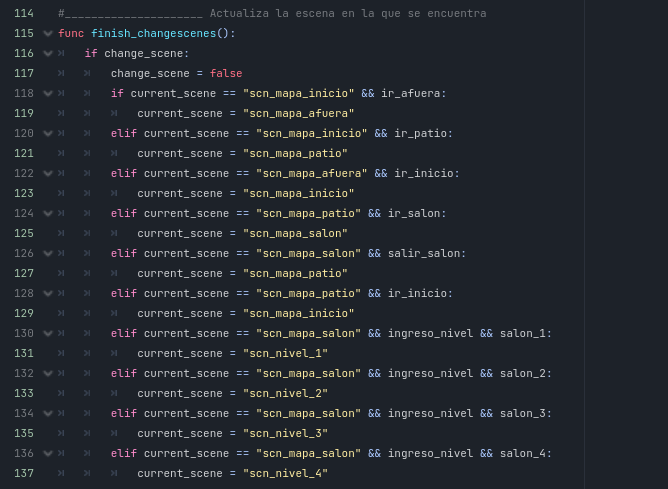






De igual manera como ocurrió con la **función** de la posicion del personaje, la estructura inicial era la siguiente:





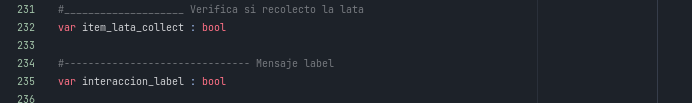
Y la **función** final termino estructurada de la siguinete manera:





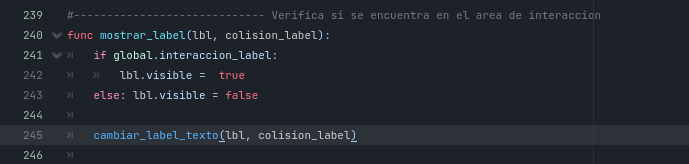
**Variables**:

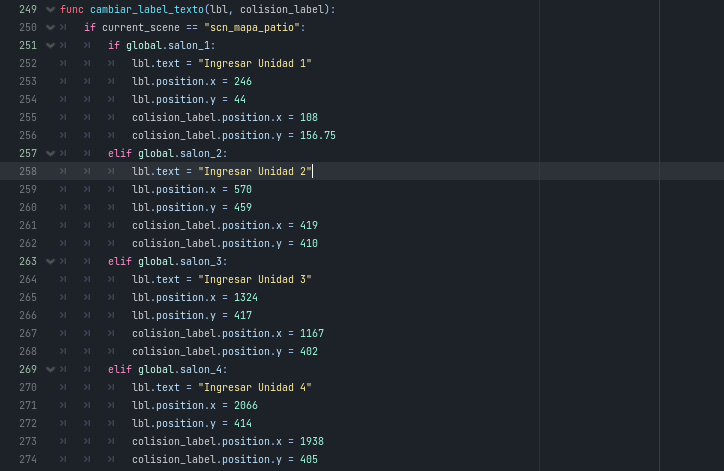


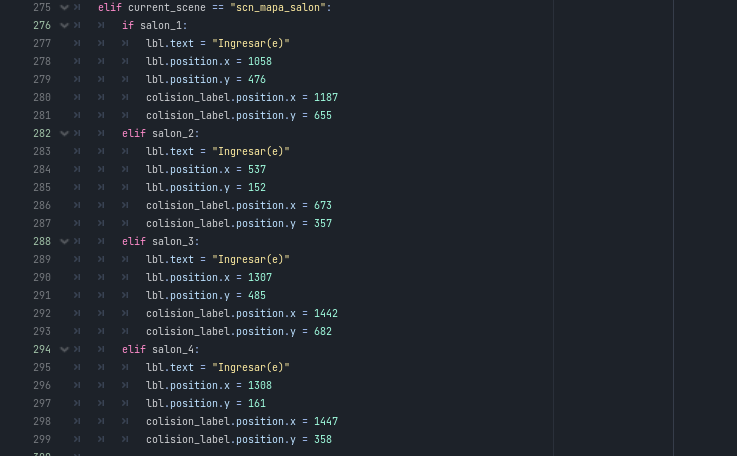


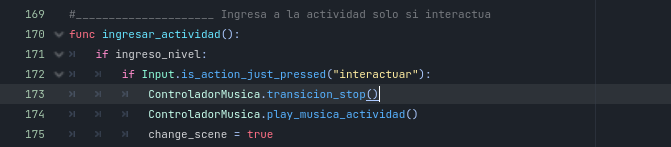


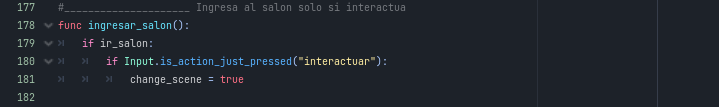
**Funciones**:

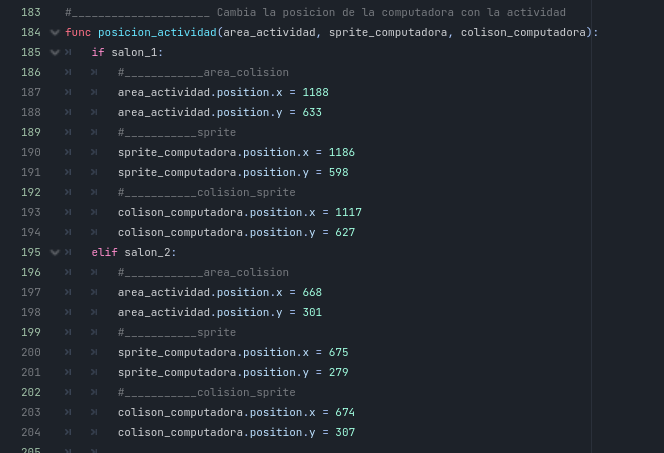


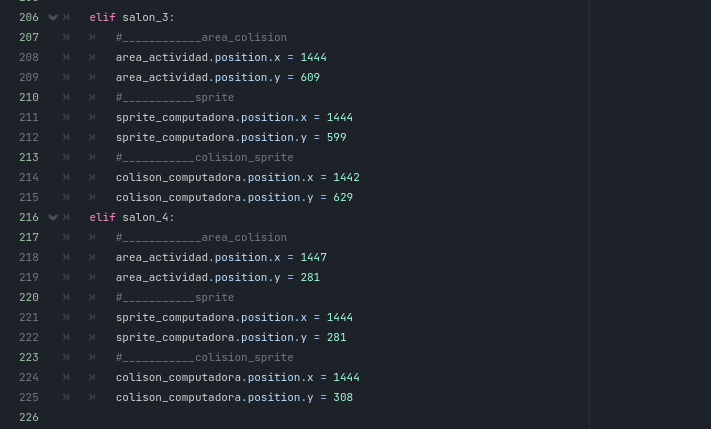


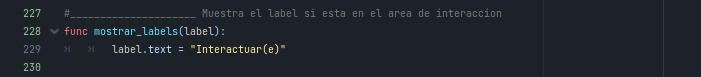












Al comienzo estas **variables** junto con la **función** estaban establecidas en el ‘Global’, pero esto hacia que consumiera muchos recursos a la hora de llamar la función en la escena del profesor, ya que esta se ejecutaba en el ‘process’. Por lo que al final decidimos que esta función solo exista en la escena propia del profesor.



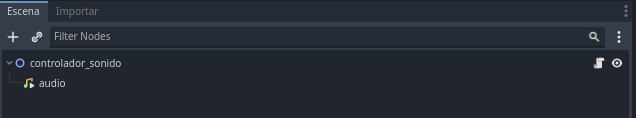
CONTROLADOR AUDIO

En este “autoload” establecimos los distintos sonidos que utilizaríamos durante el desarrollo del juego, para que de esa manera no nos viéramos en la necesidad de agregar constantemente los nodos locales de sonido en cada escena.

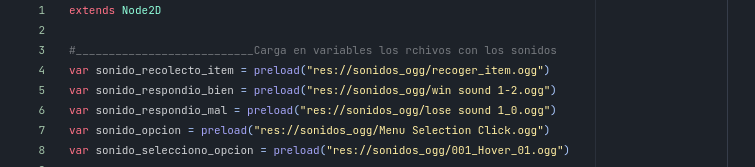
**Nodos**:

Los nodo mas importante que contiene el Nodo2d es justamente ‘AudioStreamPLayer’ que nos permite introducir música dentro del Godot.

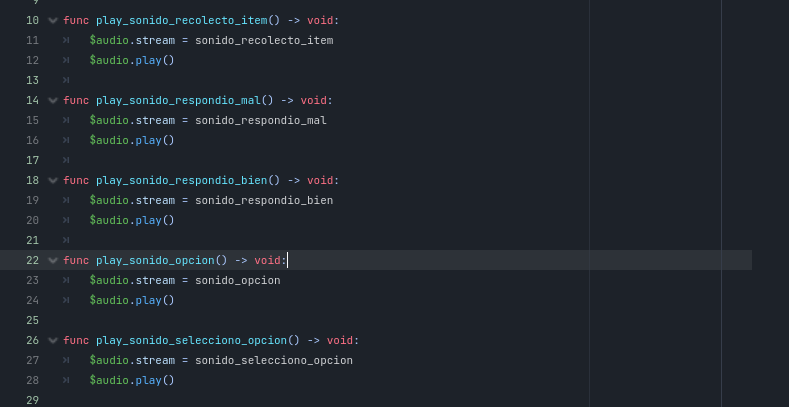
Cabe destacar que todos los audios utilizados en el juego son formato ‘.ogg’, dado que es el formato que menos espacio ocupa en la memoria, sumado a que Godot unicamente con este formato permite la funcionalidad de ‘loop’ para repetir el audio constantemente, una función que se verá presente en el ‘Controlador de música’.



**Código**:



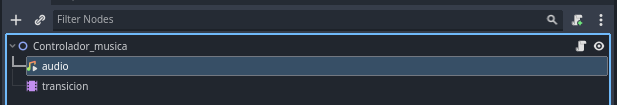
Las siguientes funciones nos permiten establecer el sonido desde cualquier escena:



CONTROLADOR MUSICA

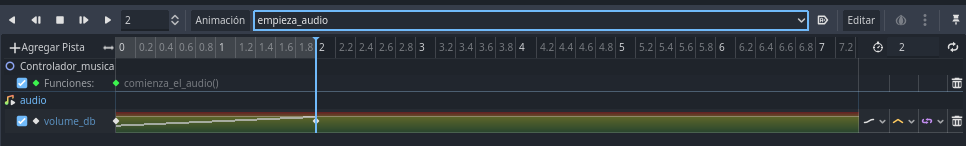
**Nodos**: En este caso, aparte del nodo del sonido que ya se había mencionado en el *‘Controlador de sonido’*, sumamos en nodo ‘AnimationPlayer’, el cual es un nodo que como su nombre nos indica, nos permite crear animaciones, con las propiedades de los nodos.

En este caso lo utilizamos para decrementar y aumentar el sonido al principio y al final de cada reproducción, para que de esa manera el cambio entre una pista y la otra no sea brusco o repentino.

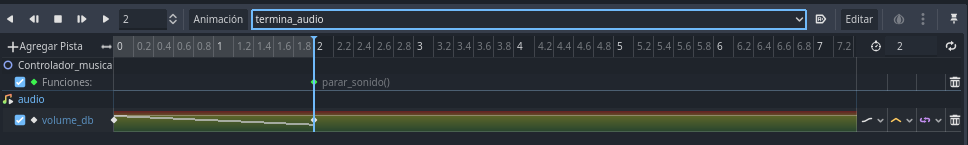


**Animaciones:**

Animacion de inicio:

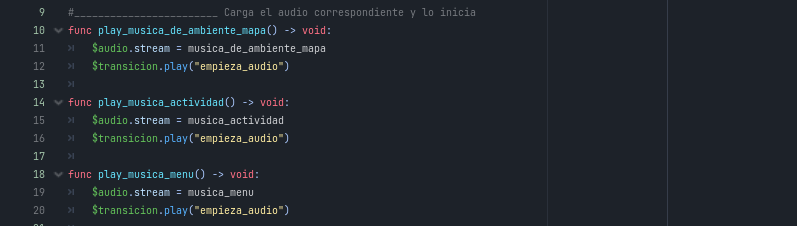


Animacion final:



Codigo:







DIALOG MANAGER

DialogManager es en realidad un ‘Plugin’ dentro de Godot.

Los ‘Plugins’ son módulos de software adicionales que agregan funcionalidad personalizada al motor grafico. Dado a que Godot es de codigo abierto, estos suelen o son en su mayoría creados por la comunidad. Teniendo esto presente. Lo que hicimos en nuestro proyecto fue incorporar este Plugin para los diálogos que requieran de una interacción, a la hora de tener opciones para crear distintas rutas de dialogo.

Para este proyecto exportamos ‘DIalogManager’, creando una copia, para así poder acceder a sus nodos y scripts, y de esta manera modificar su apariencia y agregarle una animacion al final y al principio de cada dialogo, así como también modificar variables desde su codigo.

AGREGADO DE UN PLUGGIN (COMPLEMENTO) PARA LOS DIALOGOS

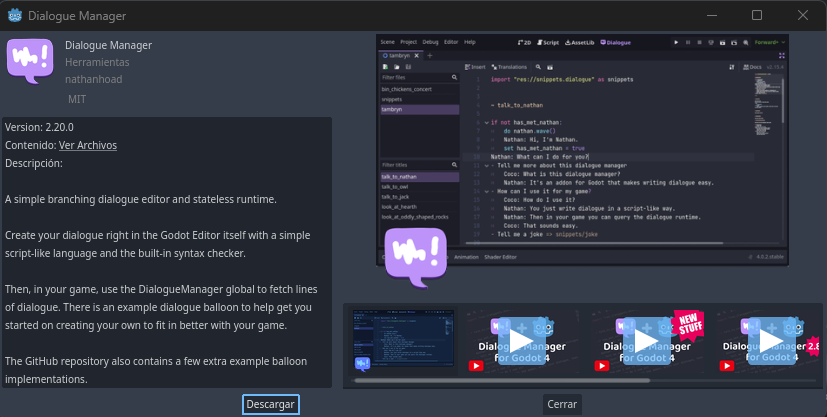
Primero descargamos el plugin desde la opción ‘AssetLib’ propia de Godot:



Buscamos en ella el Plugin ‘Dialogue Manager’



La seleccionamos y la descargamos:



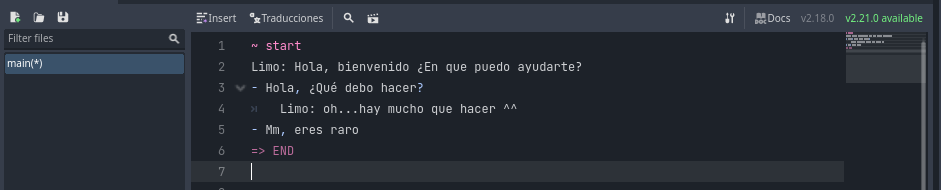
Ahora aparece una nueva opción en la parte superior, ‘Dialogue’, la que nos permitirá acceder a las funcionalidades del pluggin, trabajar con el, y así establecer y modificar dialogos y varíables booleanas.



Al accerder tendremos esta interfaz que nos permite crear los dialogos y ejecutarlos para hacer pruebas sin necesidad de ejecutar todo el proyecto ni establecerlo en ningún lugar.

Solo por mostrar un ejemplo, se presenta el siguiente dialogo:

En codigo:



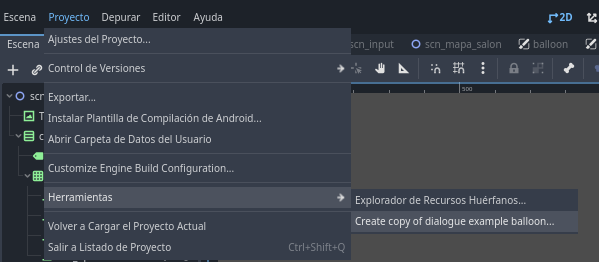
Ejecutado:



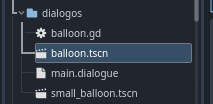
Al seleccionar la primera opción:



Desde los ajustes del proyecto realizaremos una copia de ‘dialogue example balloon’

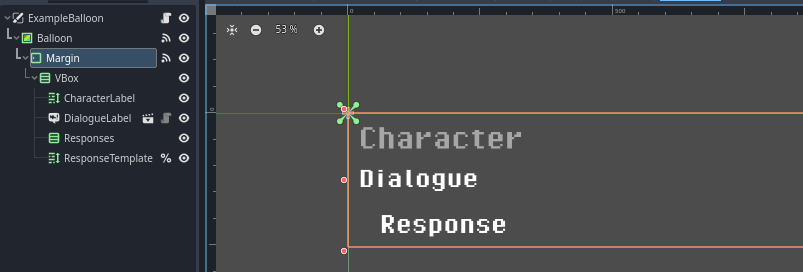


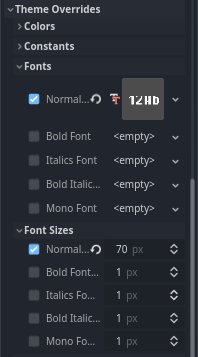
Esto nos permitirá acceder a las siguientes propiedades, donde la mas importante sera la escena ‘balloon.tscn’, que nos permitirá modifica el entorno grafico.



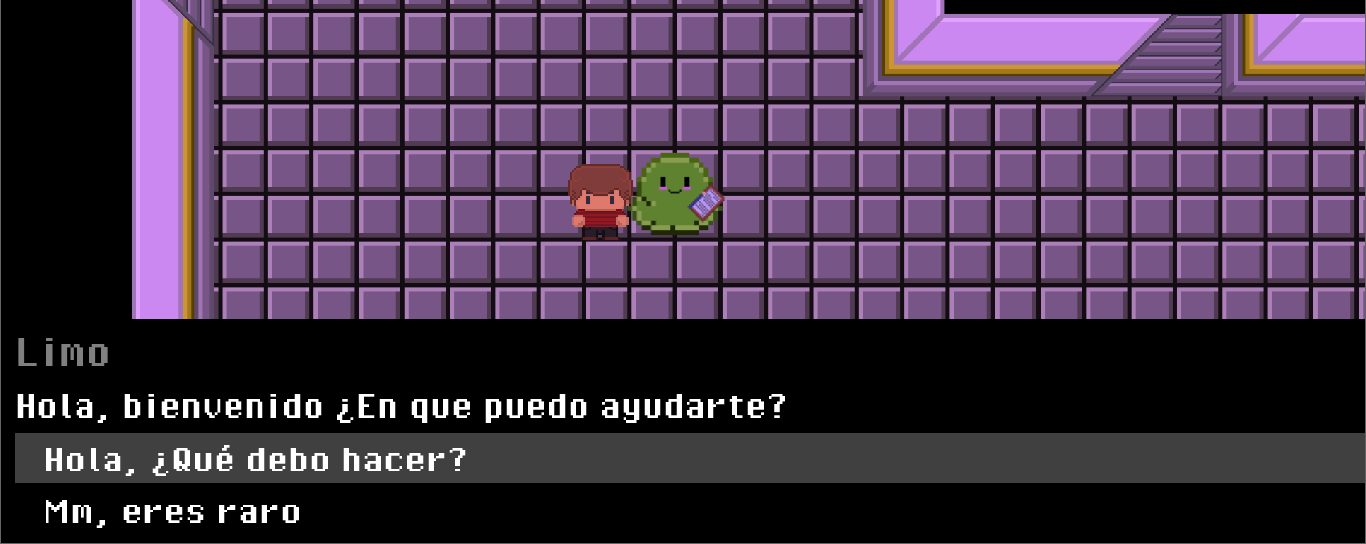
Como se puede observar la resolución del texto y tipografía, junto con el tamaño, no es la ideal, por lo que es necesario realizar ciertos ajustes.

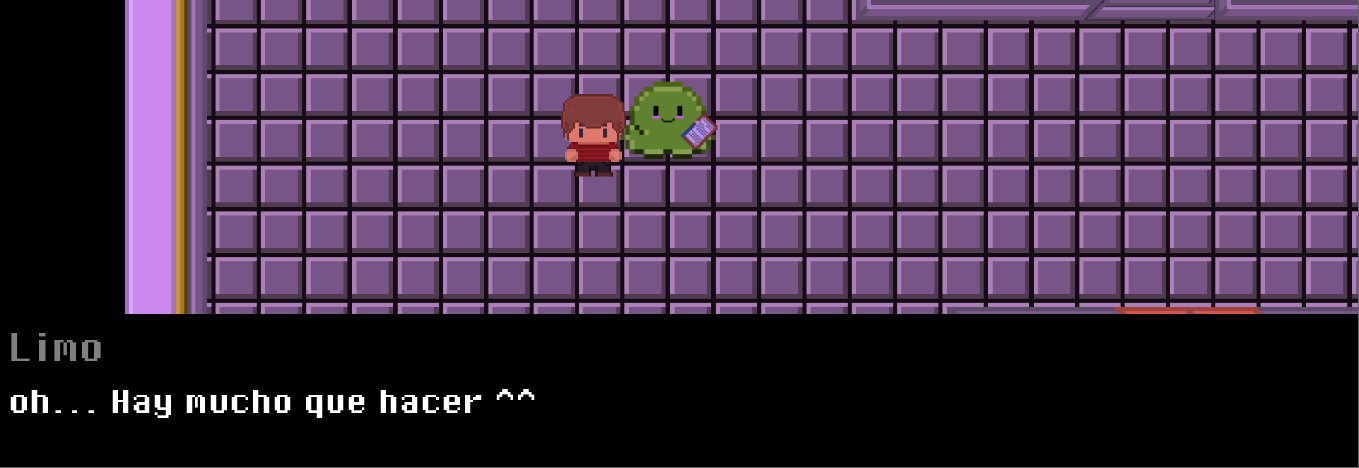
Acá modificamos la tipografía por ‘Pixel Operator HB’, y su tamaño dependiendo de si mostrara un nombre o el dialogo en si.





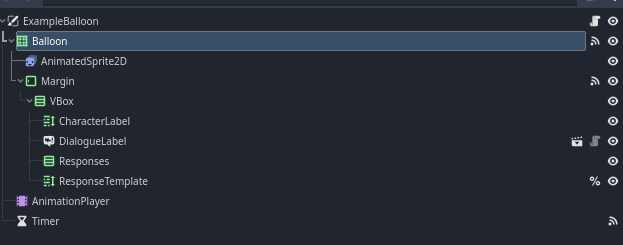
Por lo que ahora ya modificado pasaría a verse de esta manera, resultando así mas agradable a la vista, sin mencionar que es mas fácil de leer y se adapta mejor a la resolución del juego.





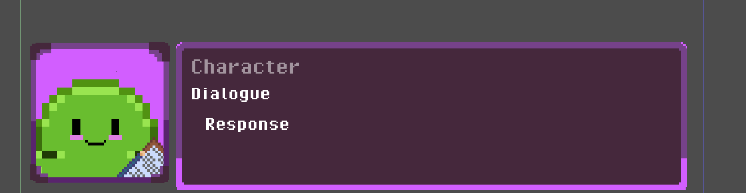
Con el fin de hacer la apariencia mas atractiva, modificamos el fondo del dialogo ya gregamos una animacion, con los nodos ‘NinePatchRect’,’ AnimatedSprite2d’ y ‘AnimationPlayer’, sumando un ‘Timer’ para que una vez finalizada la animacion se elimine la escena.

**Nodos**:

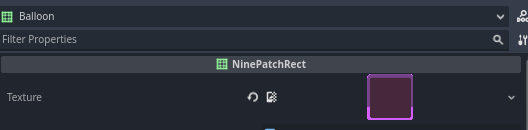


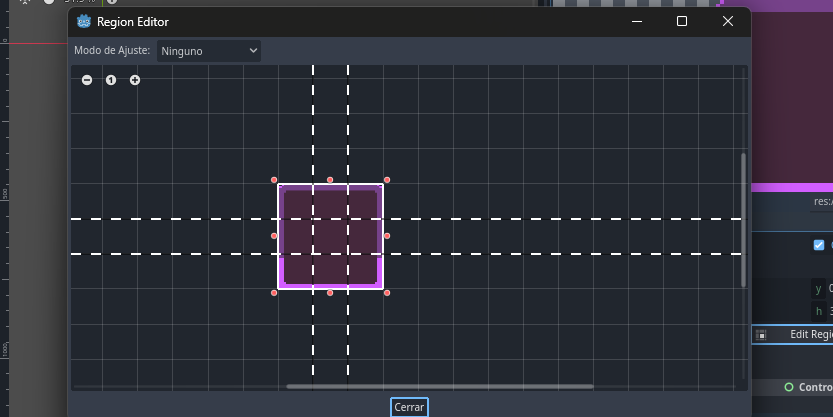
**Fondo**:

Con el nodo ‘AnimatedSprite2d’ agregamos el rostro del profesor, para que se muestre en cada dialogo:



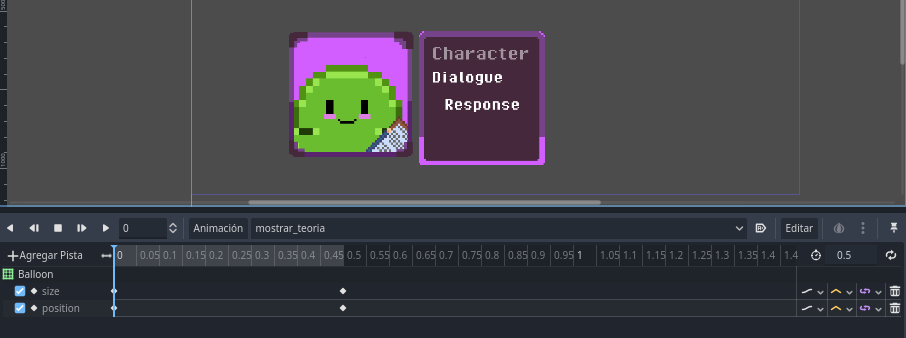
El hecho de agregar el fondo en ‘NinePatchRect’ hace que el fondo se ajuste al tamaño que ocupa su contenido sin problemas con la propiedad ‘region’.

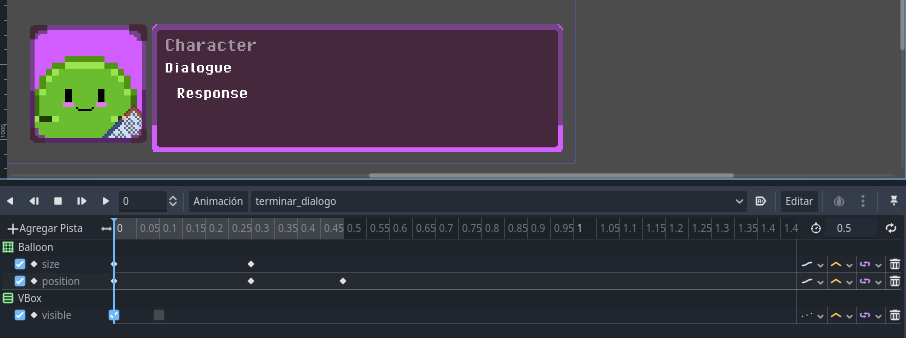


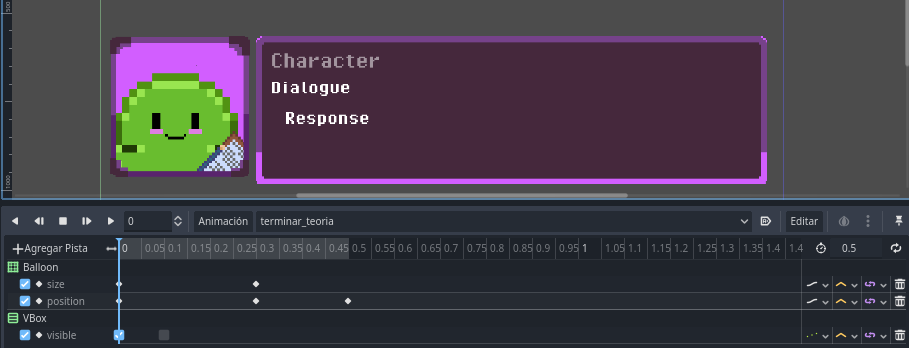


**Animacion**:



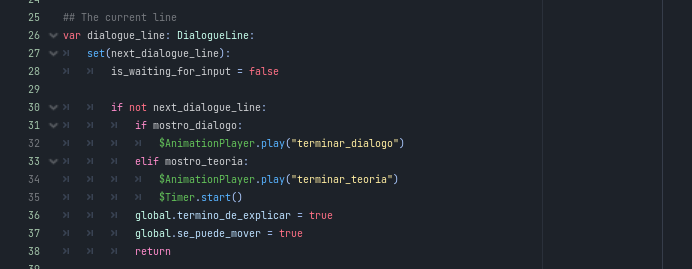




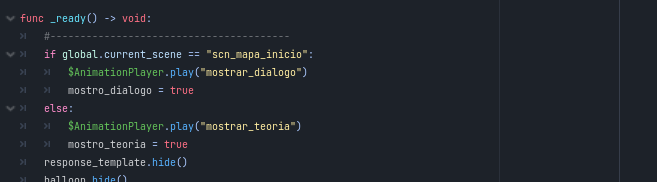


**Agregados en el codigo:**

Sumamos una variable que determina a partir de que momento el personaje se puede mover, siendo este cuando termine el dialogo.



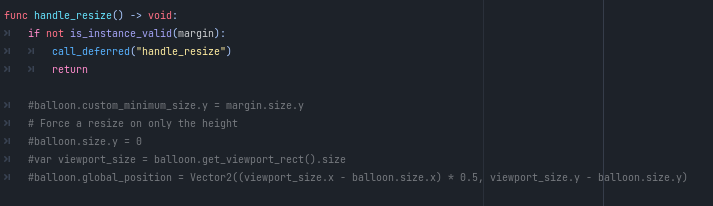
En este caso indicamos que animacion debería mostrarse:





Así como agregamos código, también eliminamos(o en este caso comentamos) codigo para que no interfiera con las modificaciones.

Por ejemplo:

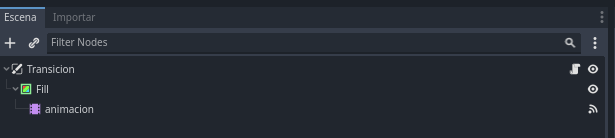


TRANSICIÓN

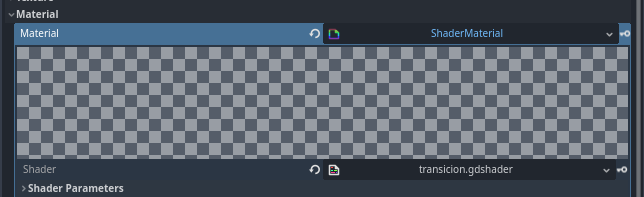
Tal y como su nombre lo indica, el propósito de este ‘Autoload’ es mostrar la animacion que utilizamos en el juego como transición.

**Nodos**:

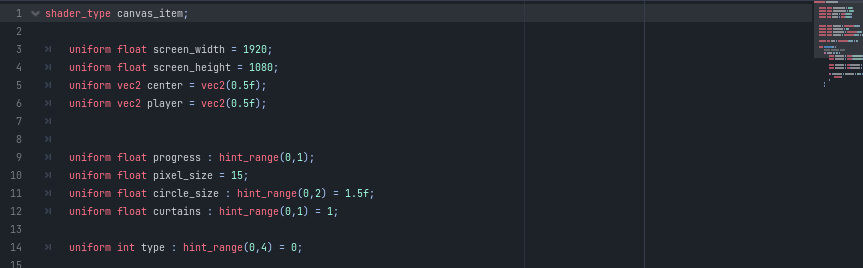
El nodo principal es un ‘CanvasLayer’ que nos permite dibujar la transición junto con el nodo ‘ColorRect’.

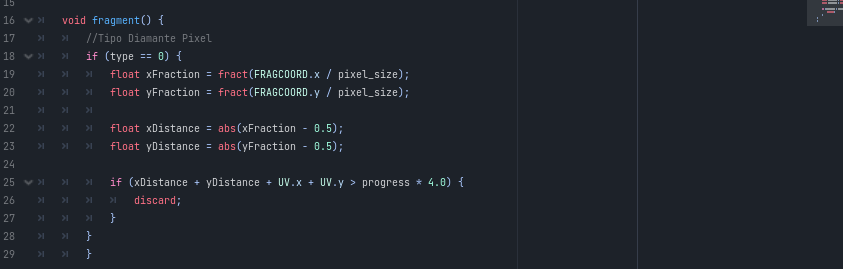


En un inicio creamos un ‘Shader’ que se encuentra en la propiedad ‘Material’ del nodo ‘ColorRect’:



Agregándole el siguiente **codigo**:

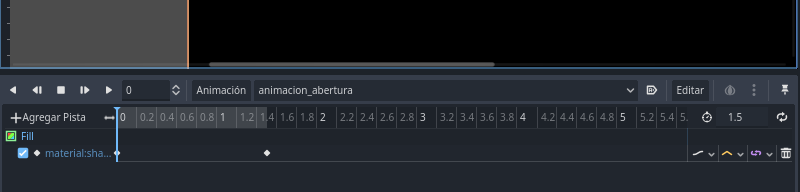


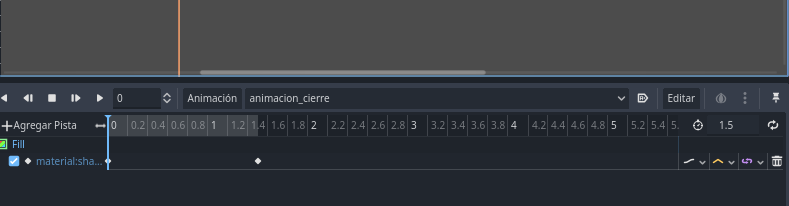


Que nos permite acceder a los siguientes **parámetros**:



Y con ellos crear las **animaciones**:





**Código de la escena:**

En un principio en la funciones establecimos que el nodo se elimine una vez finalizada la aniamación, porque nos habíamos encontrado con el problema de que, si bien el nodo no se veía todavía seguía existiendo, y al estar superpuesto al resto de escenas, este impedía la interacción del mouse con los distintos botones y cajas de texto. Sin embargo el hacer que lo elimine hacia que no se pudiera llamar a la animacion nuevamente porque esta ya no existía, y el hacerlo ‘no visible’ con la propiedad ‘visible’ no era una opción porque obtenía el mismo resultado, ’así que luego de investigar un poco mas sobre las propiedades que incluía Godot, finalizamos en ‘esconder’ el nodo con el método ‘.hide()’ en vez de eliminarlo.

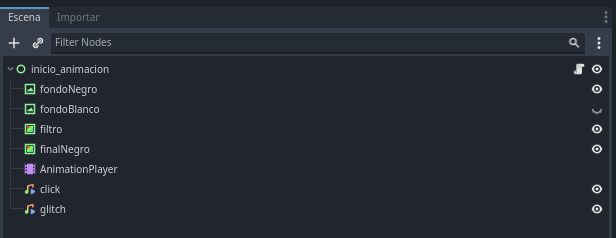


ESCENAS GENERALES

INTRODUCCION

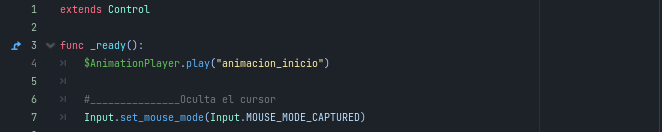
Agregamos una animación al inicio del juego que muestra el logo en su version blanca y negra, junto con efectos de sonido.

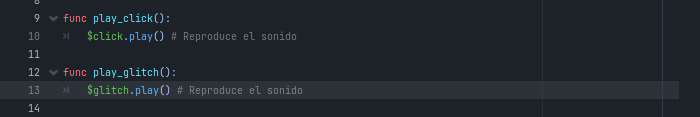
**Nodos**:

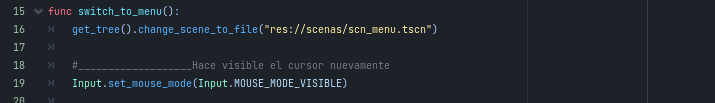




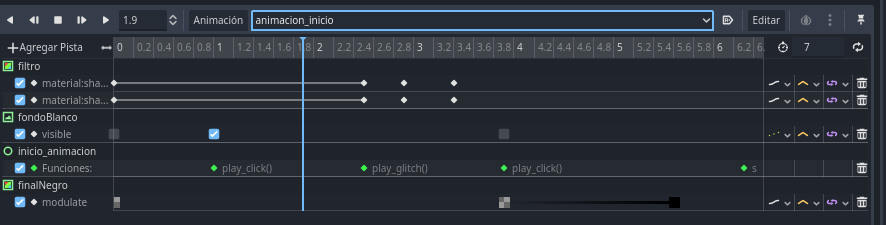
**Código**:







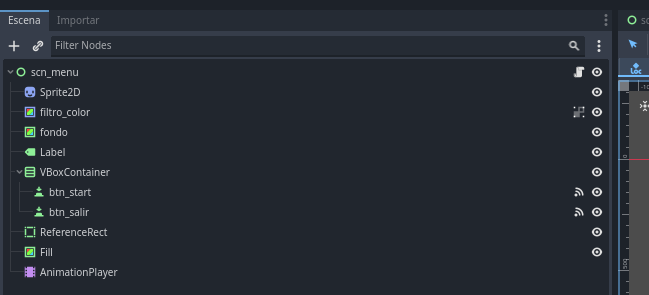
**Animación:**



MENÚ

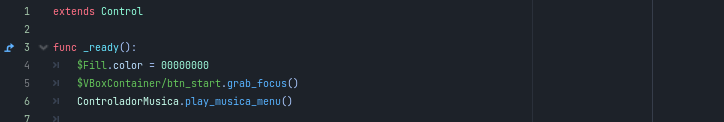
El menú cuenta con las opciones ‘Start’ y ‘Exit’, que tienen como función unicamente iniciar el juego o salir de este.

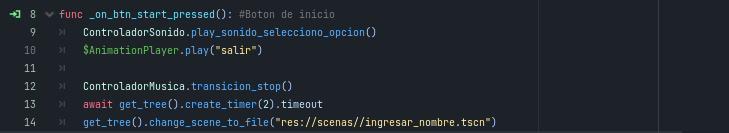
**Nodos**:

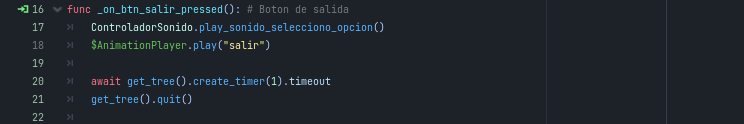




**Codigo**:

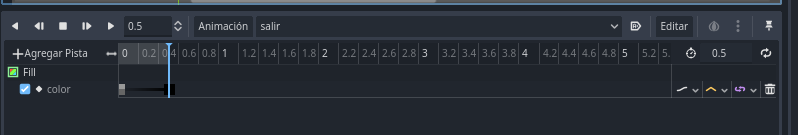








**Animacion**:

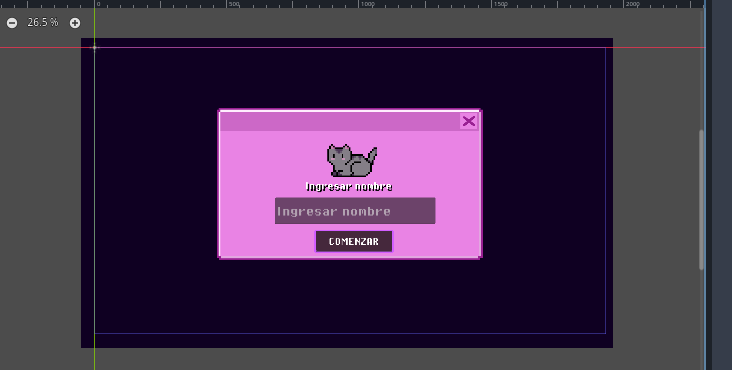


INGRESAR NOMBRE

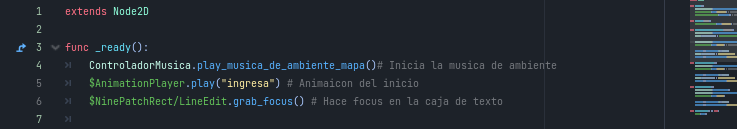
En esta el jugador tiene que ingresar su nombre, que será utilizado en ocasiones en el transcurso del juego.

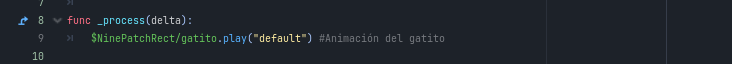
**Nodos:**

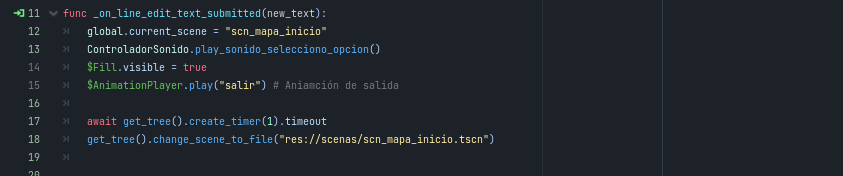


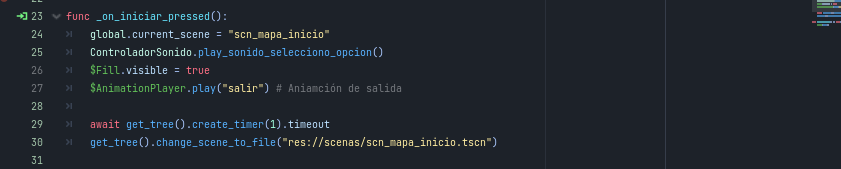


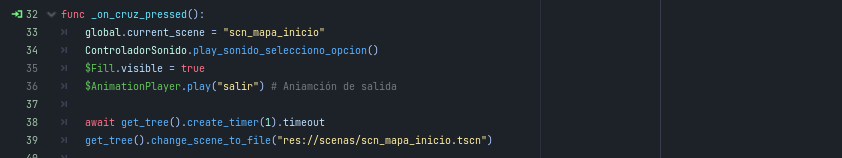
**Código:**





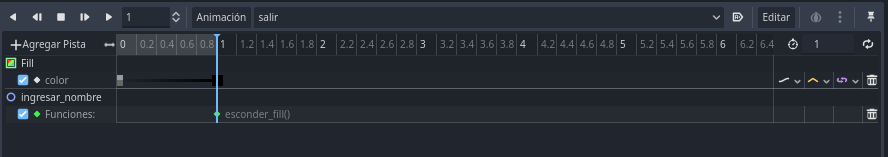


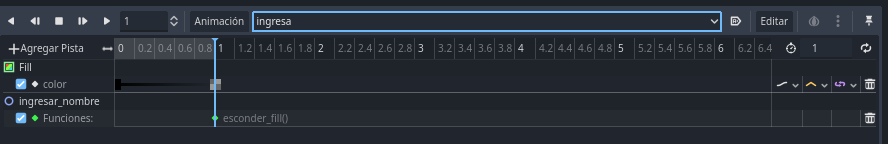






**Animaciones:**

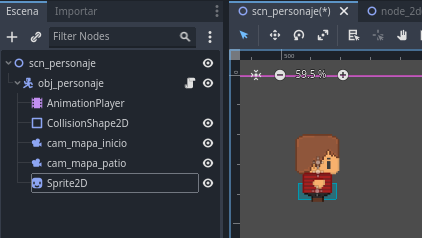


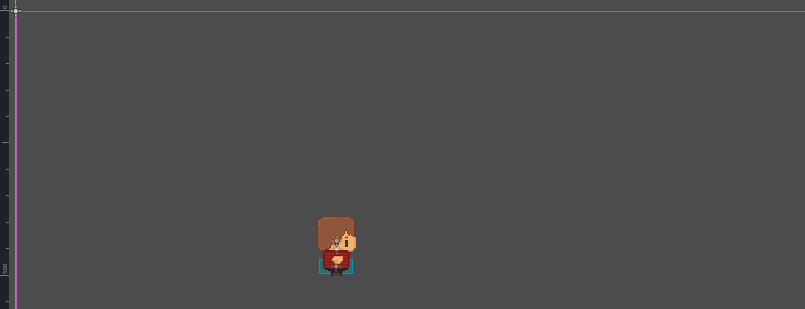


PERSONAJE

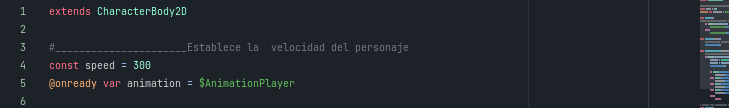
Para la creación del personaje que utilizamos los nodos:

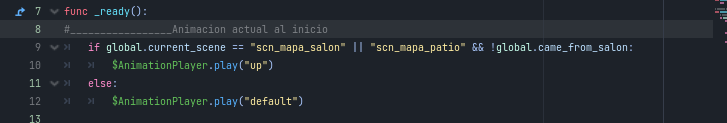
* **Node2d**: Se utiliza para crear y gestionar diversos nodos bidimensionales.
* **CharacterBody2d**
* **AnimationPlayer**: Almacena las animaciones de movimiento.
* **CollisionShape2d**: Establece las colisioes con los objetos dentro del mapa.
* **Camera2d**: Es la cámara que define lo que se mostrará en la pantalla, siguiendo al personaje.
* **Sprite2d**: Almacena la apariencia del personaje.

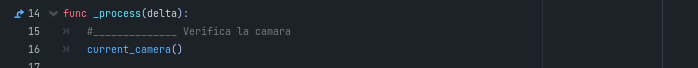




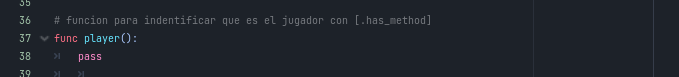
**Codigo**:

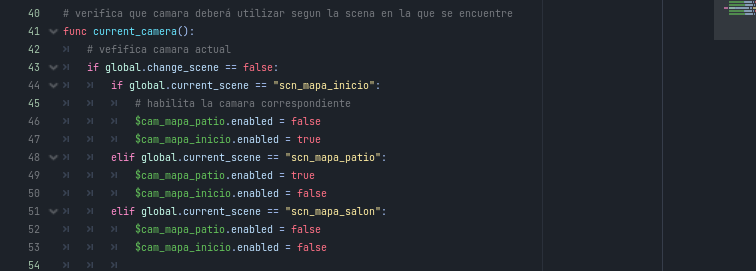






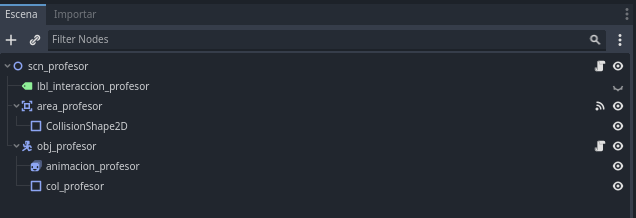




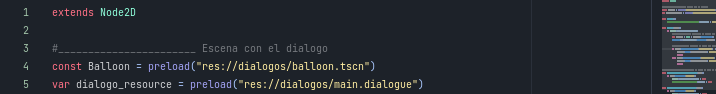


PROFESOR

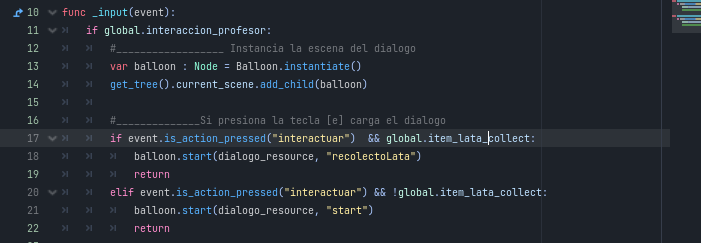
**Nodos**:

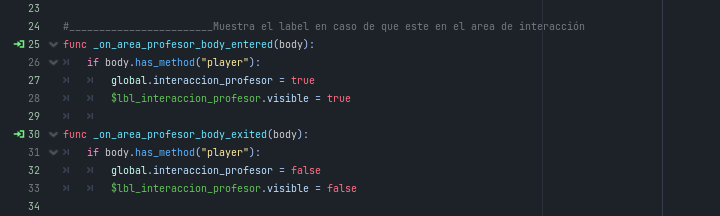


**Código**:





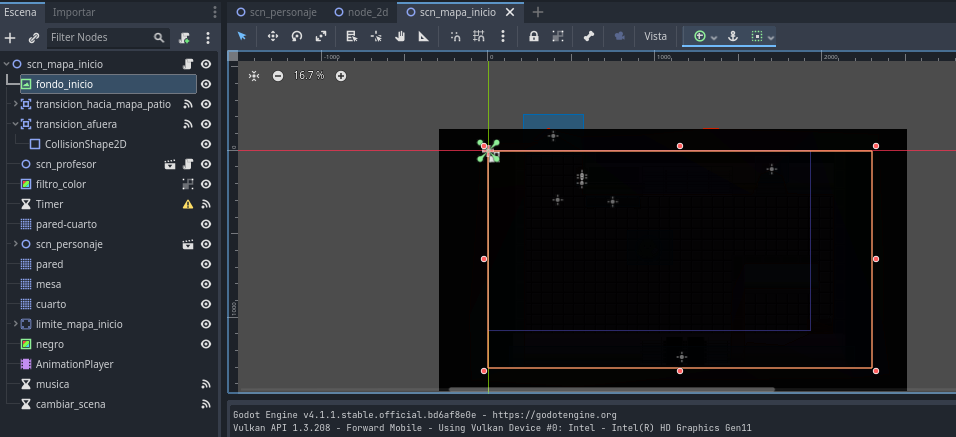




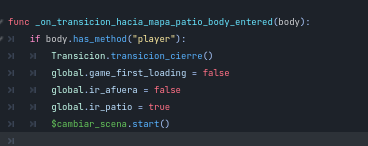
MAPA INICIO

El mapa de inicio sera el primer mapa que el jugador podrá recorrer, así como la única escena en la que se puede establecer un dialogo con el profesor hasta el momento.

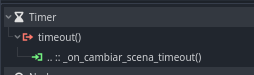
En un inicio los nodos que componían a la escena del mapa de inicio eran siguientes:

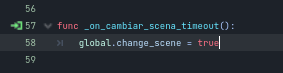


Observamos que era ineficiente crear ‘Timers’ repetidas veces, por lo que reemplazamos el código que habíamos establecido,por ejemplo, como se muestra en la imagen, que consistía en inicializar el tiempo:



Y luego en la señal de cada nodo indicar que acción se iba a realizar:





Lo modificamos suplantando el método anterior por la siguiente linea de código:

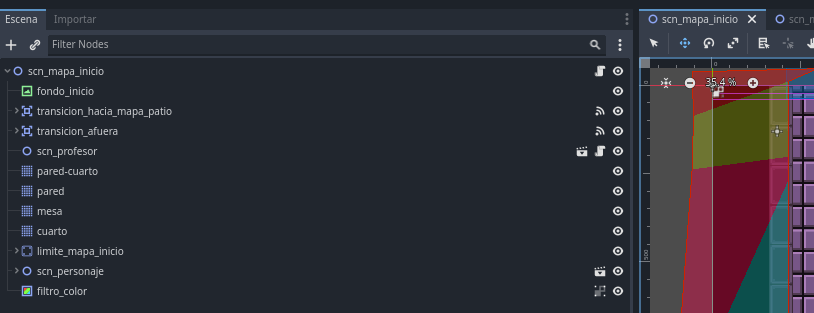


Lo que nos permite disminuir la cantidad de nodos utilizados y la cantidad de lineas de codigo, obteniendo el mismo resultado.

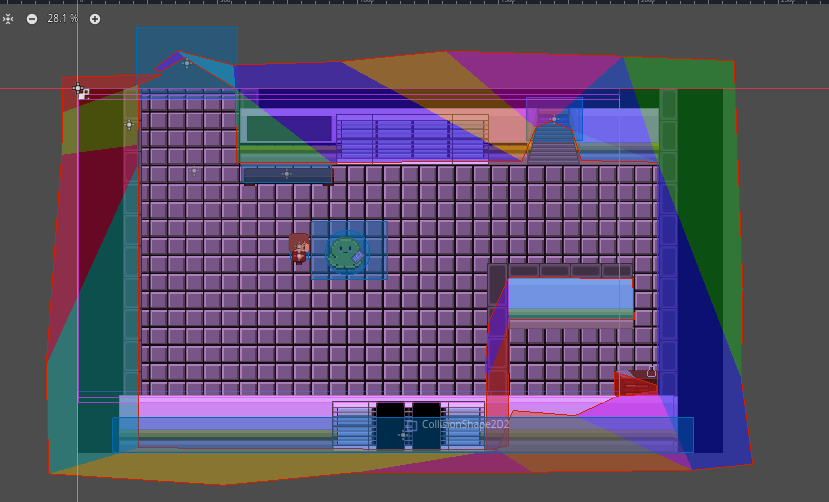
Así como también eliminamos el ‘ColorRect’ de color negro, que tenia como único propósito hacer una animacion en el inicio diferente al de resto de escenas, asignando así la misma transición que las demás.

Por otro lado, habíamos establecido que la música se iniciaría en el mapa de inicio, pero luego de realizar pruebas, concluimos que lo mejor era iniciarla en la escena en la que se ingresa el nombre del usuario, para que de esta manera no se generen cortes en el audio, dado que si la música estuviera establecida en la función de inicio del mapa, esta se reiniciaría cada vez que ingrese a este, lo que es inconveniente, ya que a la hora de recorrer el mapa, ingresa en este en reiteradas ocasiones.

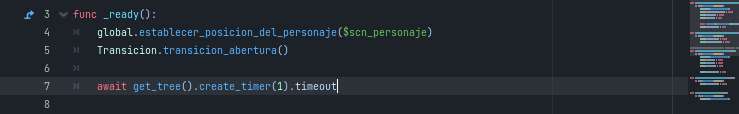
Los **nodos** finales son:

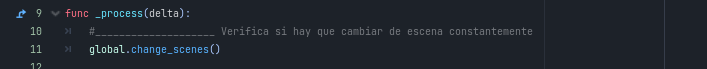


Algunos objetos del mapa se agregan aparte, como las paredes y los muebles, para agregar un poco mas de perspectiva al juego respecto a la profundidad. Y las colisiones igualmente, se adaptana este propósito:

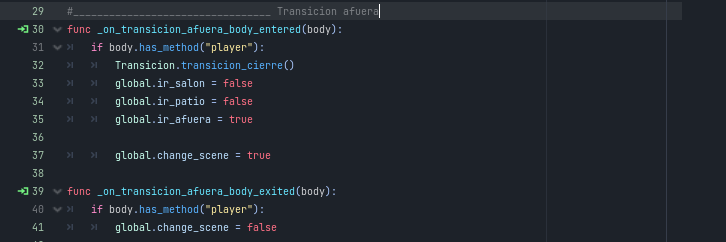


**Código**:





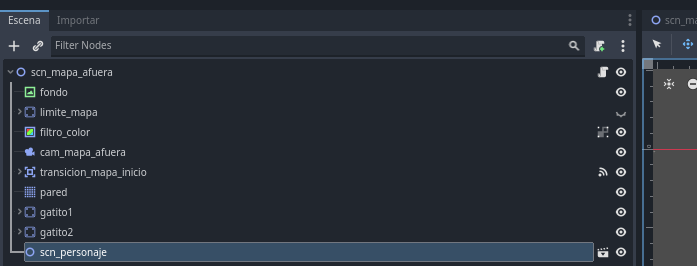


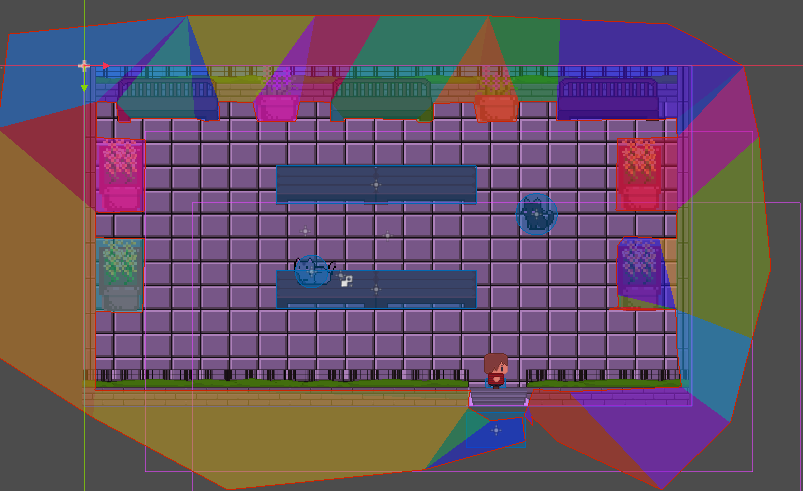


MAPA AFUERA

El propósito de esta escena es, momentáneamente, ofrecerle al jugador un nuevo espacio para recorrer, aunque existe la posibilidad de que sea el foco de recolección de items en el futuro.

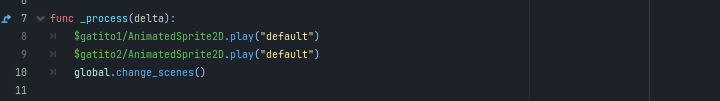
**Nodos**:

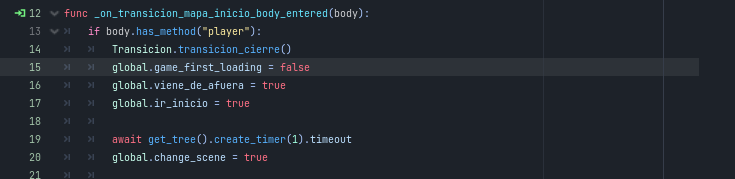


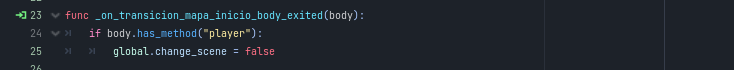


**Código**:





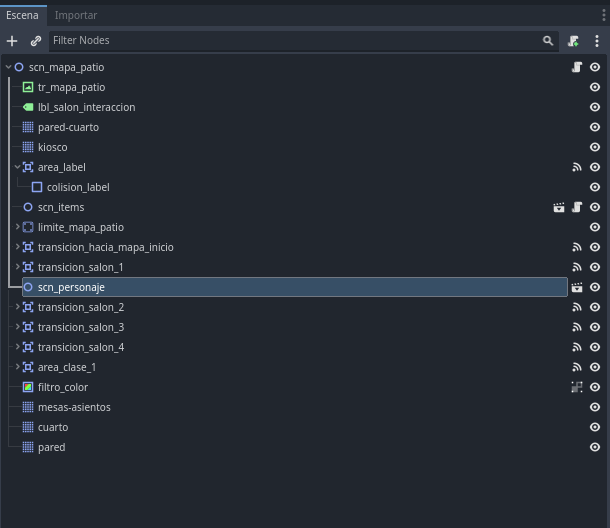


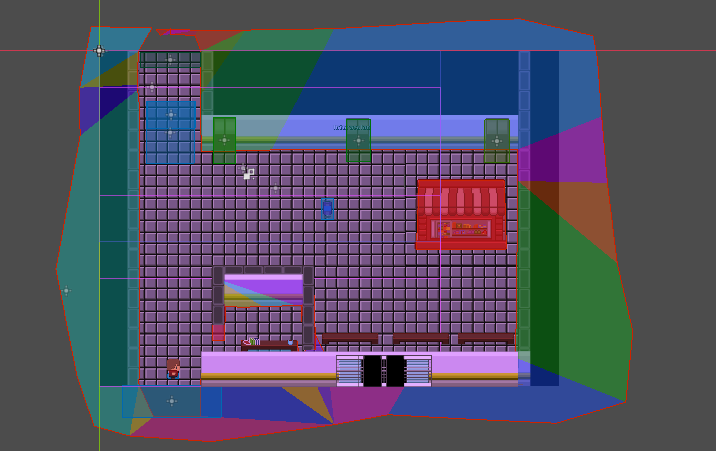


MAPA PATIO

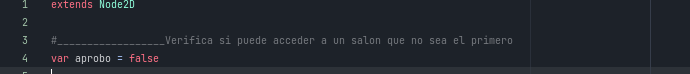
Este mapa conecta con las escenas de los diversos salones y el inicio.

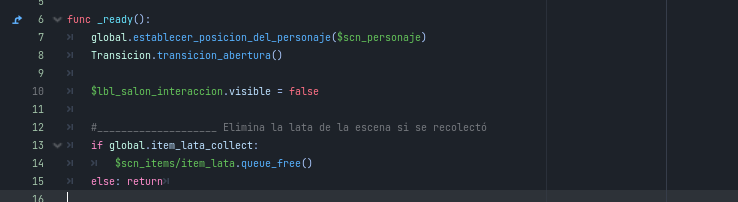
**Nodos:**

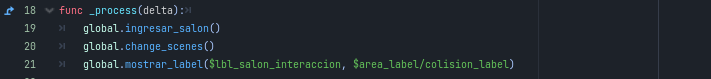


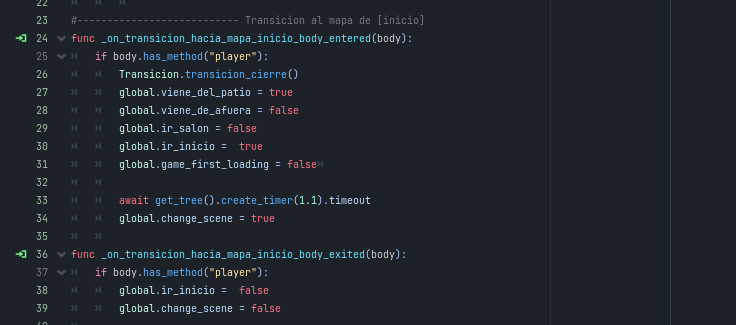


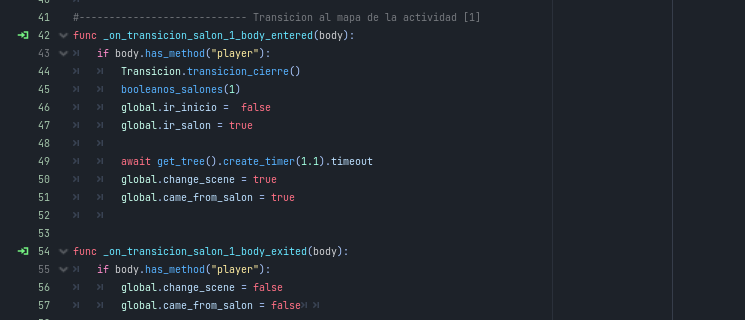
Código:



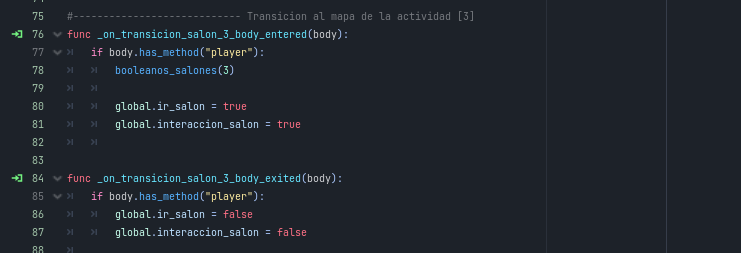


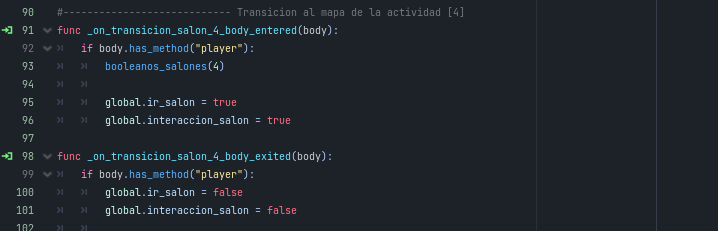


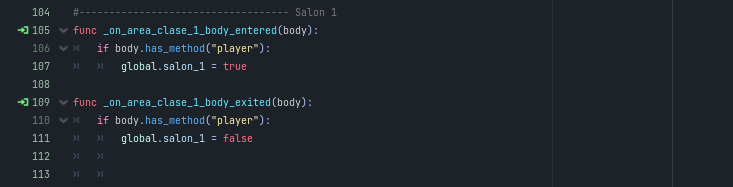


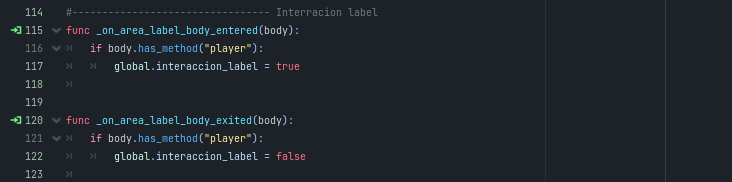


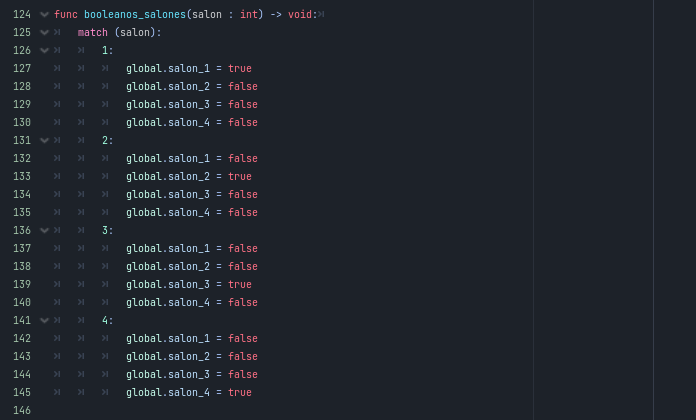






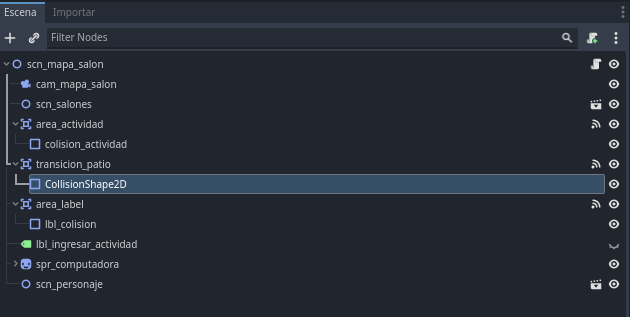


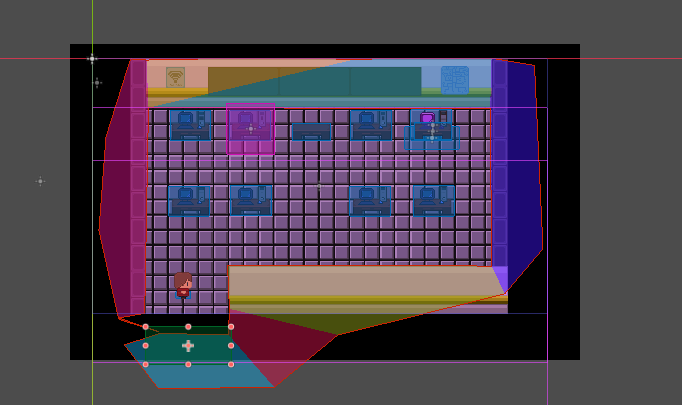




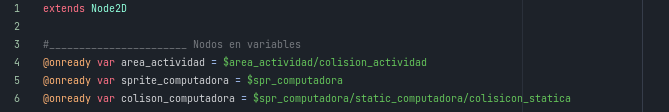
MAPA SALON

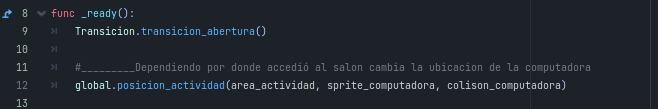
Esta escena conecta con los diversos niveles y actividades.

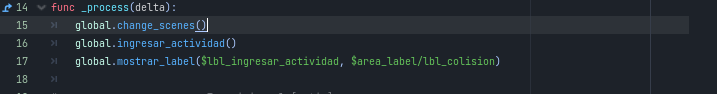


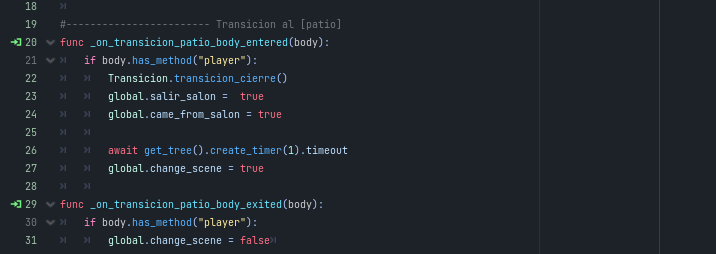


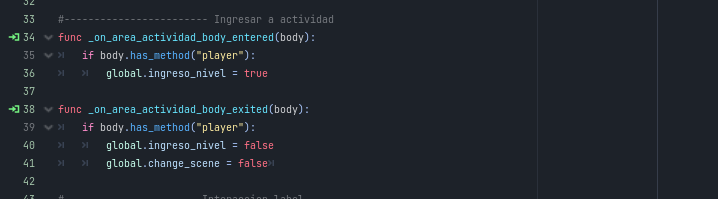
**Código:**

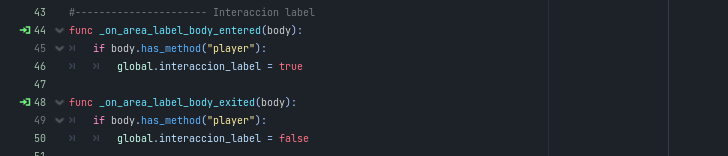












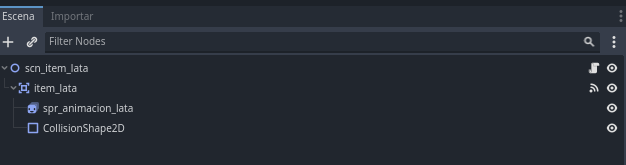
ITEMS

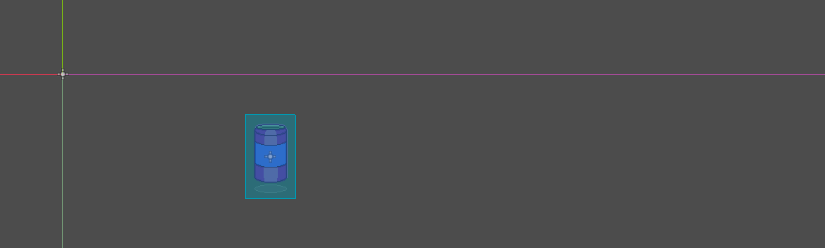
Los items son aquellos que el jugador puede encontrar al recorrer el mapa, por el momento no tienen otro propósito que generar diversos diálogos con el profesor. Sin embargo, luego serán utilizados como llaves dentro del juego para determinar el final de este.

Hasta el momento se encuentran disponibles 2 Items, una galleta y una lata de bebida energética.

**LATA**

**Nodos**:



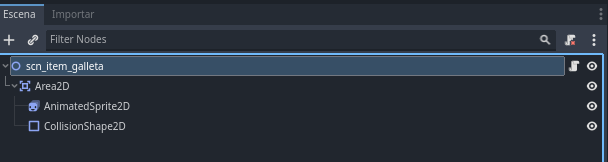


**Código**:



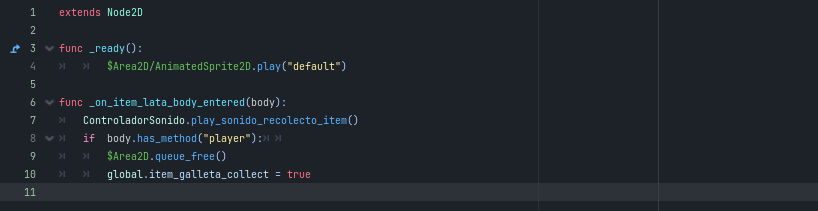
**GALLETA:**

**Nodo:**

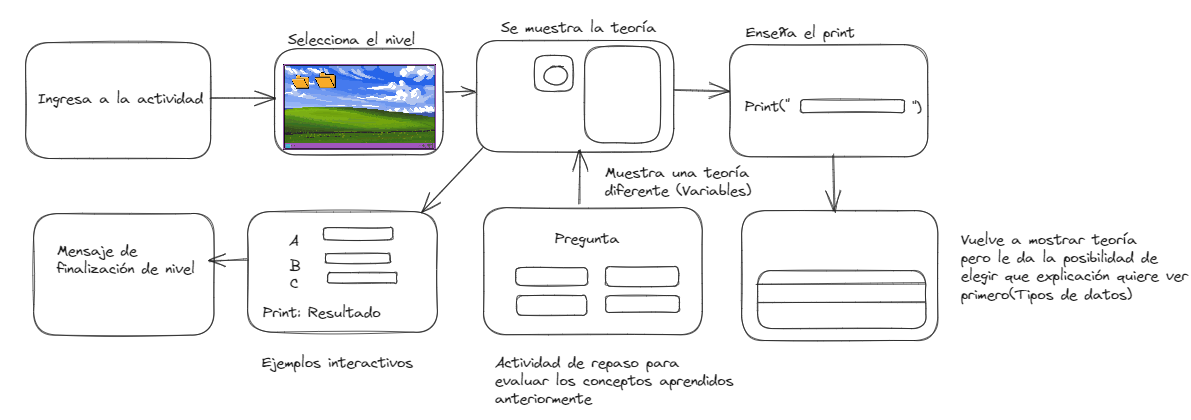




**Código:**



**Maquetado, simplificado, del primer nivel:**



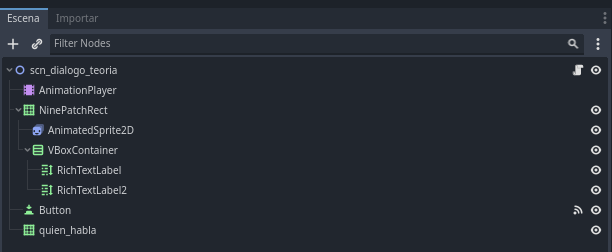
Con el fin de hacer el juego los mas eficiente posible a la hora de armar los niveles, decidimos crear escenas que puedan reutilizarse. Esto lo conseguimos cargando las explicaciones teóricas y actividades en archivos JSON. Por lo que bastaria con cambiar este archivo para cambiar todo el contenido de la escena funcionando de la misma manera.

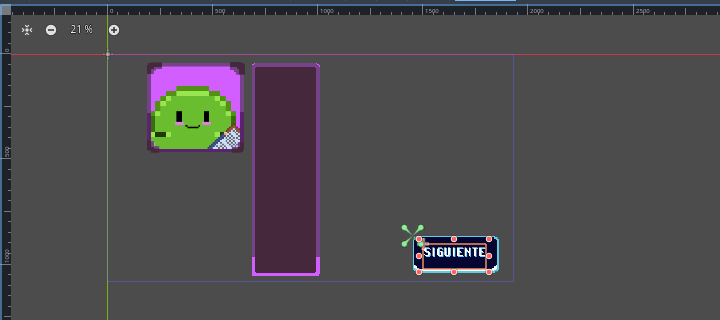
ESCENA TEORIA

Para mostrar las expliacciones teóricas, que en el juego se muestran como diálogos del profesor, esta vez no utilizamos el ‘Plugin’ de ‘DialogManager’ ,sino , que creamos nuestra propia escena en la que leemos un archivo JSON y mediante a funciones las muestra en pantalla. Con la única diferencia de que solo se mostrará texto y no habrá opciones de dialogo.

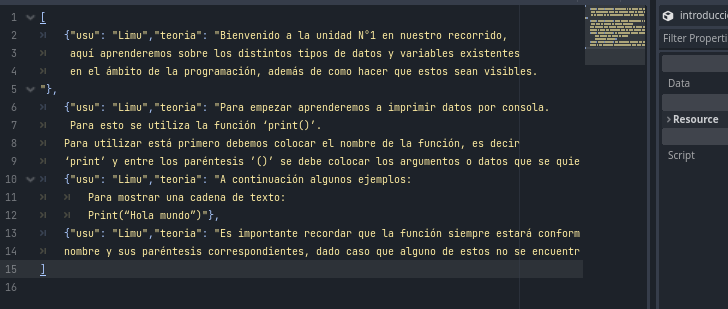
Aunque igualmente utilizamos el ‘Plugin’ en aquellas escenas en las que el jugador puede seleccionar que explicación quiere leer primero.

**Nodos**:

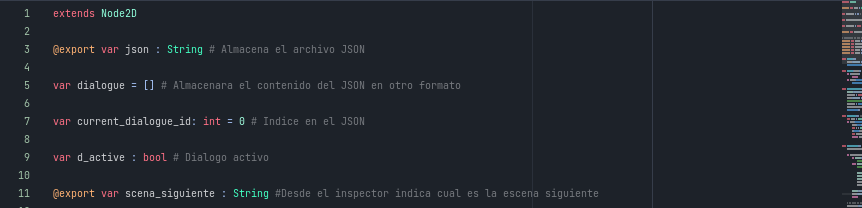


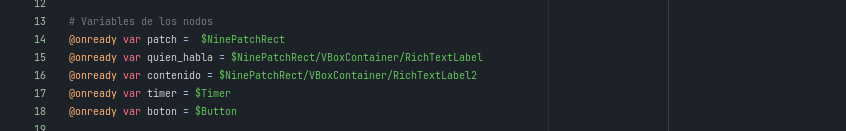


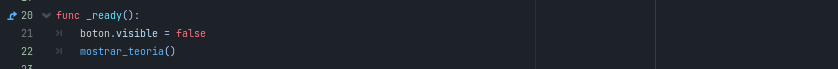
**JSON - Ejemplo**:

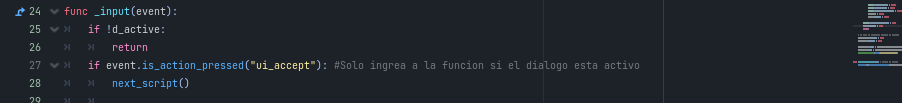


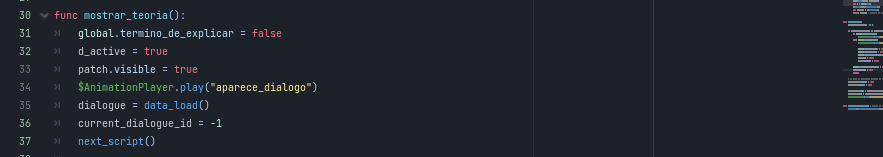
**Código**:

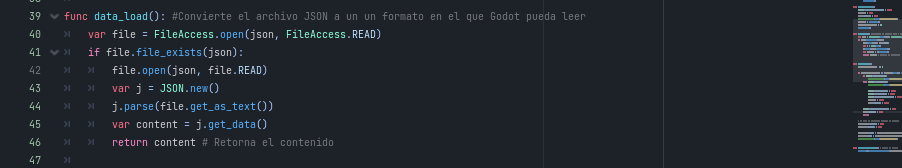


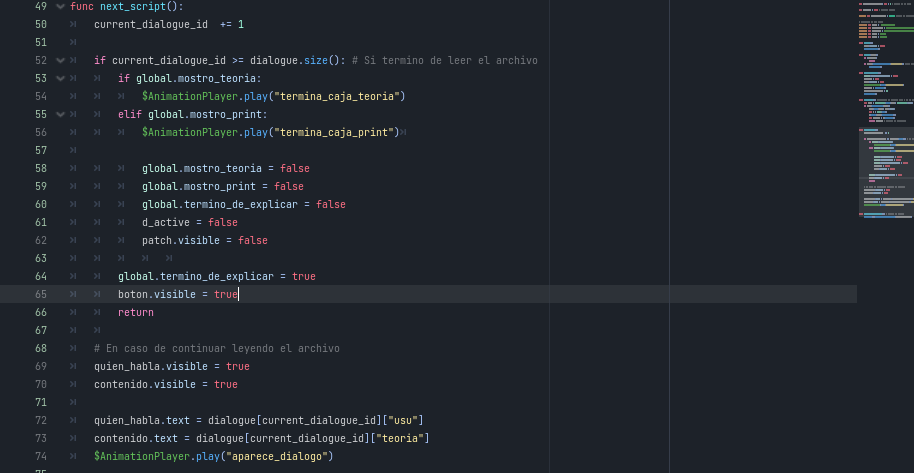






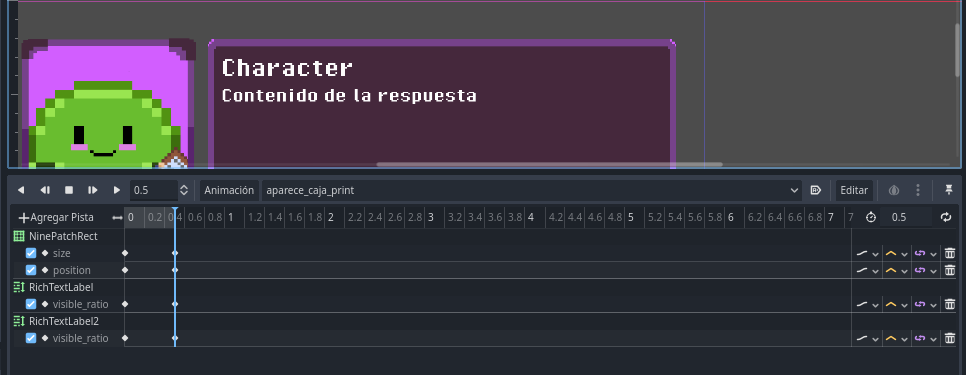




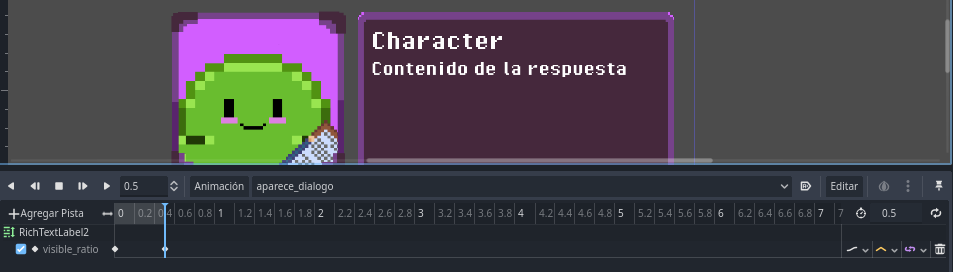


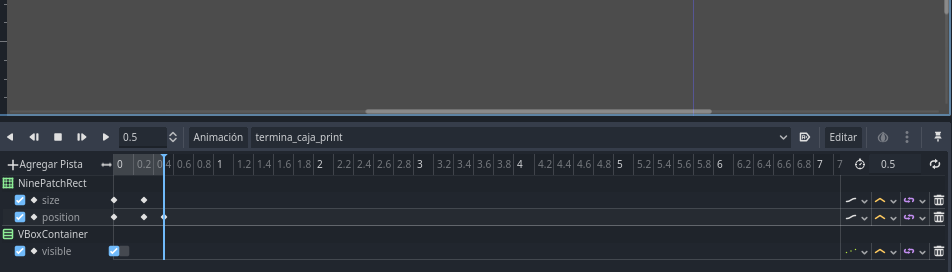


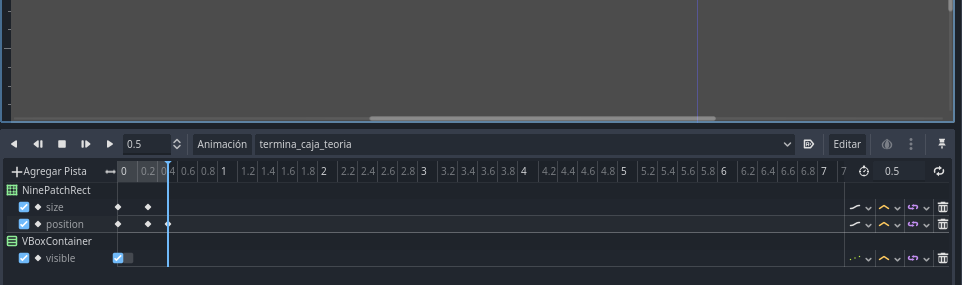
**Animaciones:**







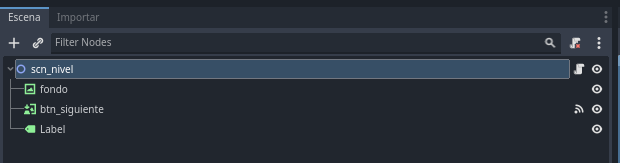


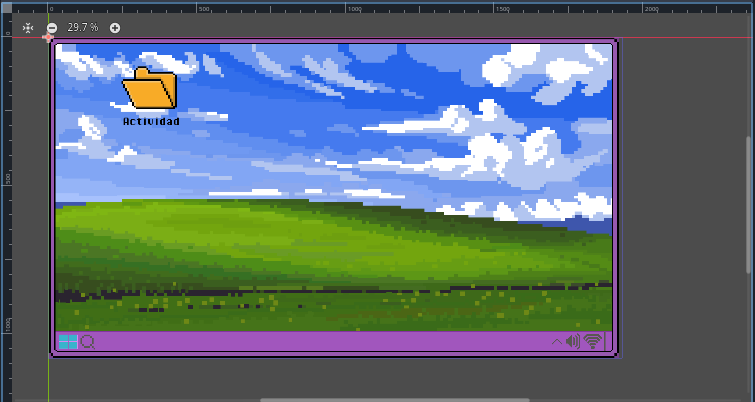


INICIO DE LA ACTIVIDAD

Una vez que ingreso a la computadora desde el salon, se cargará una escena en la que aparecerá la simulación de un escritorio. En este habrá una carpeta que al seleccionarla lo llevará a la primera explicación teórica.

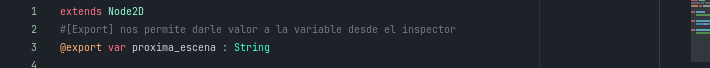
**Nodos**:

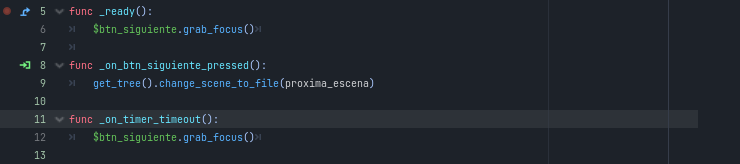




**Código**:

Como se menciona en el comentario del codigo, la propiedad [EXPORT] nos permite modificar el valor de la variable desde el inspector.

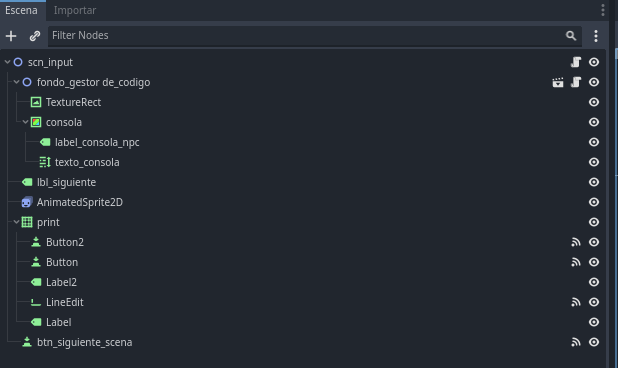


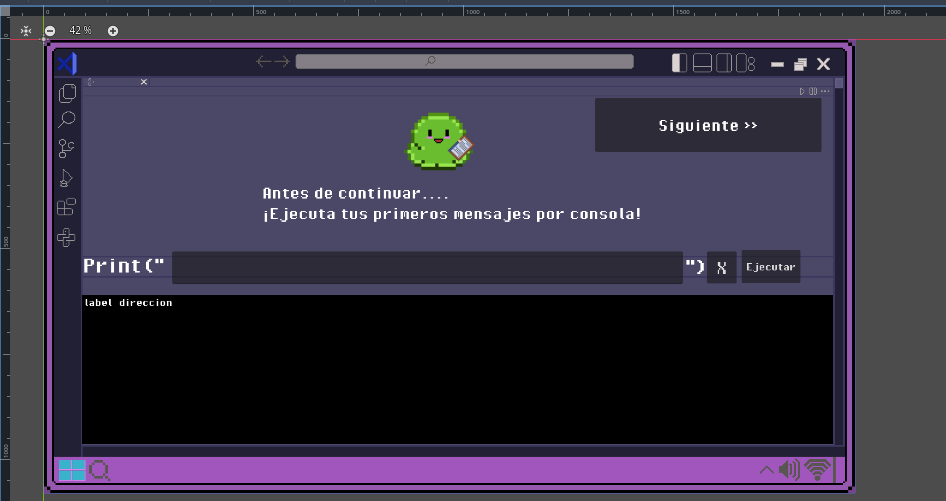


ESCENA PRINT

En esta escena se presenta una simulación de consola dentro del conocido gestor de codigo ‘Visual Code’ para que el jugador se de una idea de como luciría este codigo, justamente, cpdi codificado en este gestor.

**Nodos**:



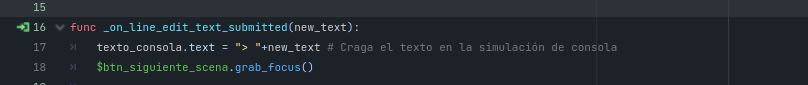


**Código**:

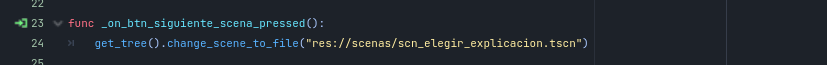








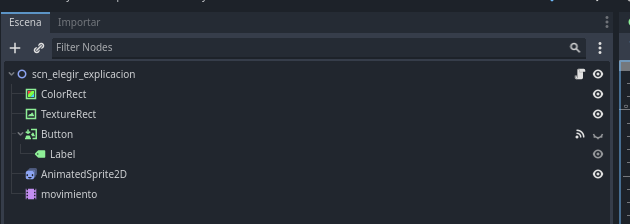


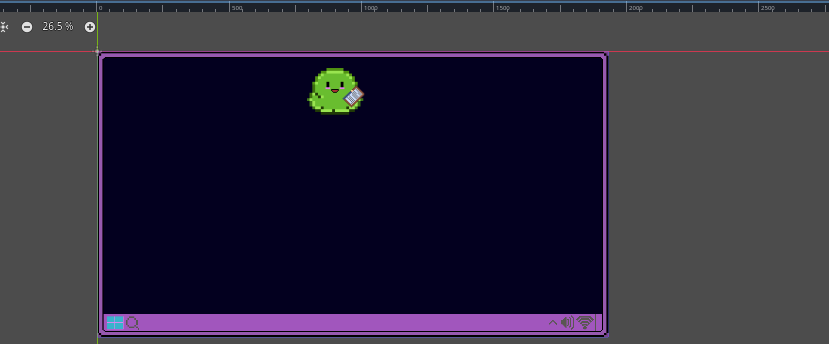


ESCENA SELECCIONAR EXPLICACIÓN

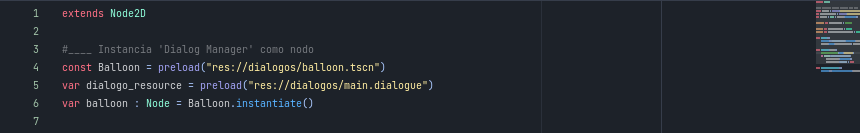
Esta escena busca ser mas interactiva, por lo que le permitirá al jugador decidir que explicación teórica desea leer primero. Todo esto mediante a las opciones que el ‘Plugin’ de ‘Dialog manager’ nos permite incluir.

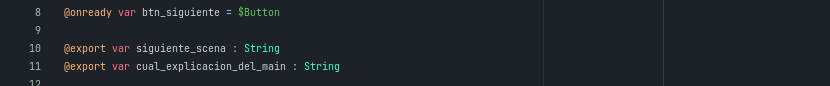
**Nodos**:

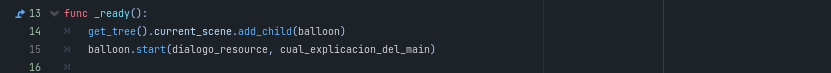




**Código**:











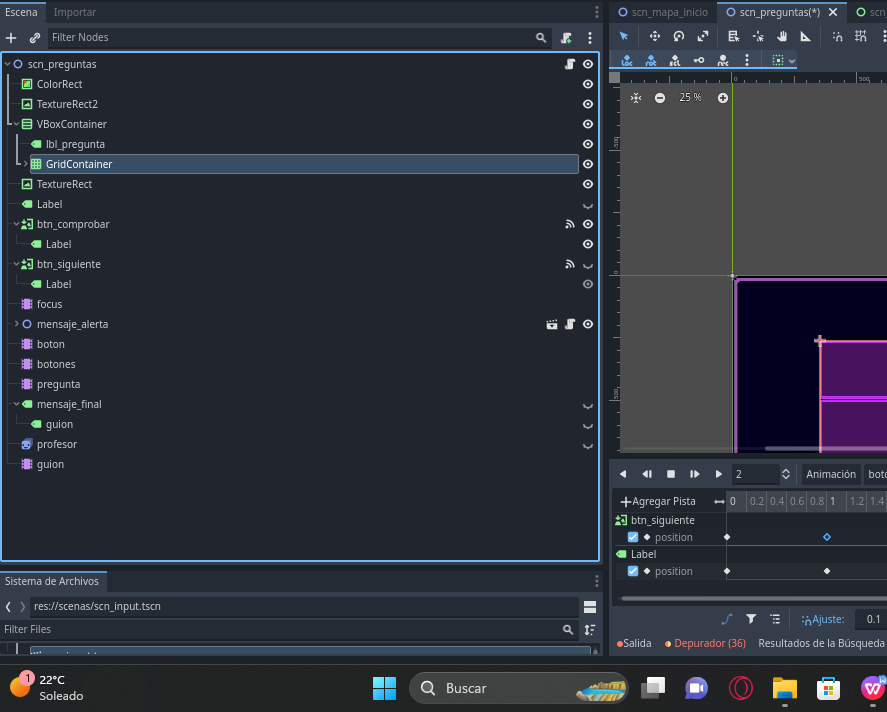
ESCENA PREGUNTAS

En esta escena se presenta una pregunta con sus 4 opciones, donde solo una respuesta es correcta. El fin de esta escena es realizar una evaluación y repaso de los conceptos aprendidos.

En caso de responder correctamente se cargará la siguiente pregunta, y en caso contrario, se le dará la posibilidad de responder nuevamente, luego de leer la alerta que indicará la respuesta correcta.

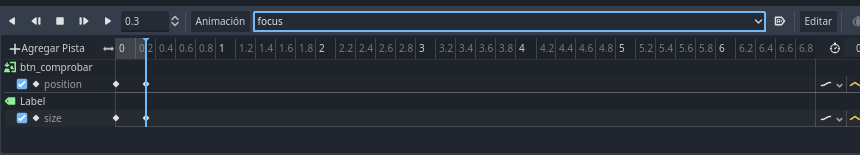
Al igual que la escena de los diálogos, el contenido de la escena se cargará de un archivo JSON.

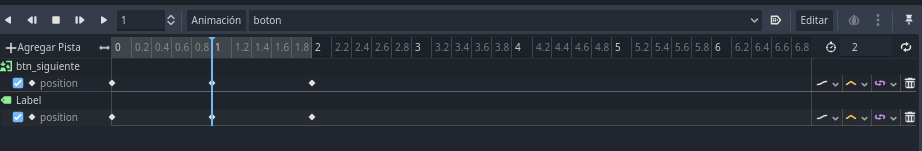
**Nodos**:

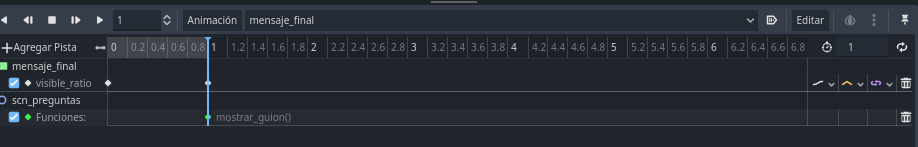


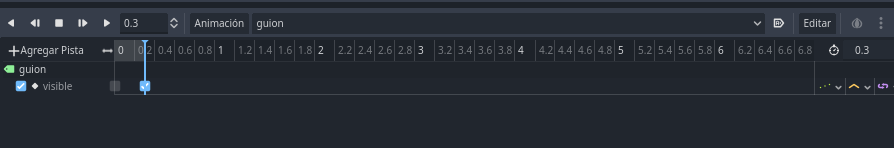


**Animaciones**:



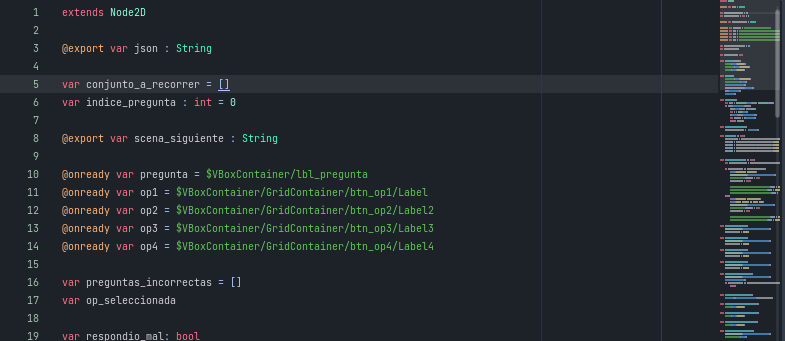


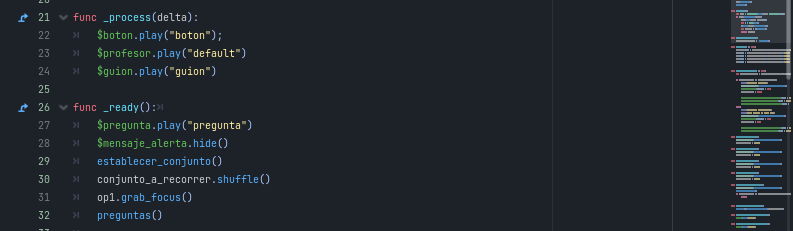


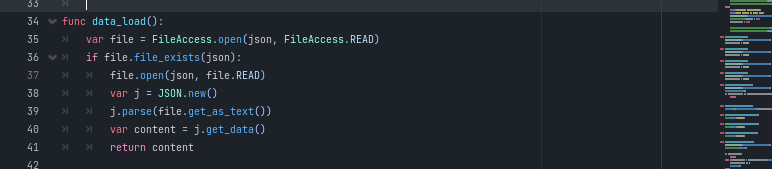


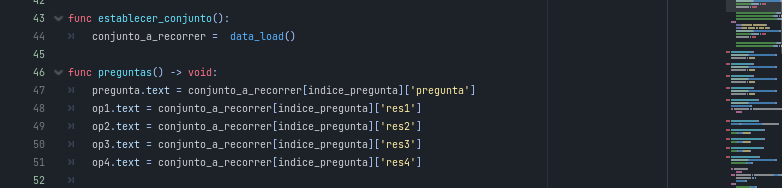


**Código**:

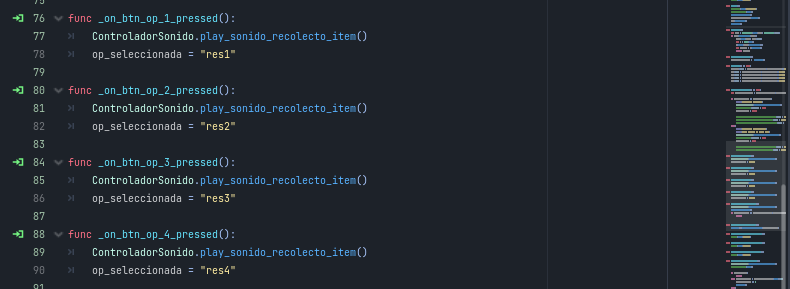


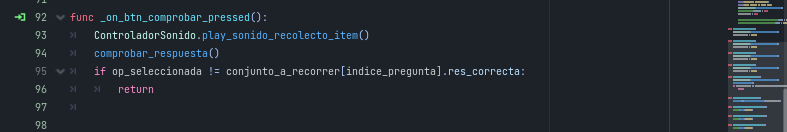


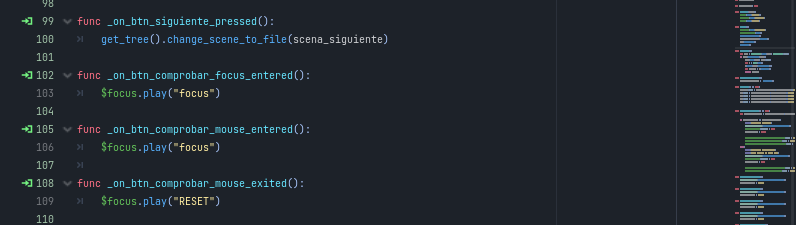


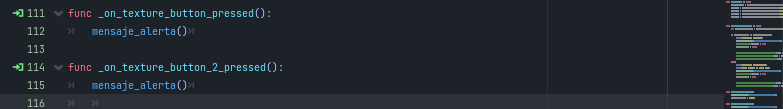


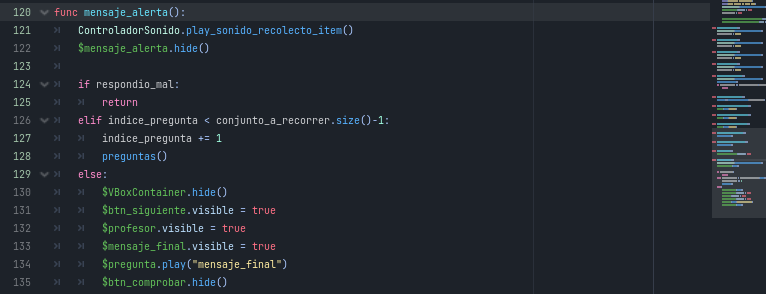






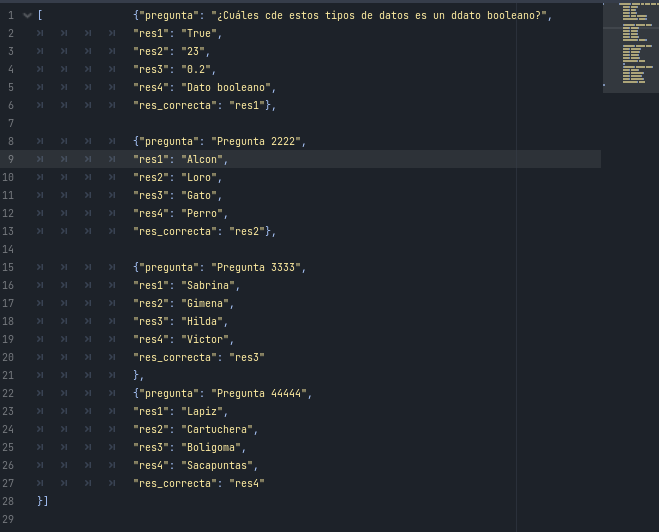








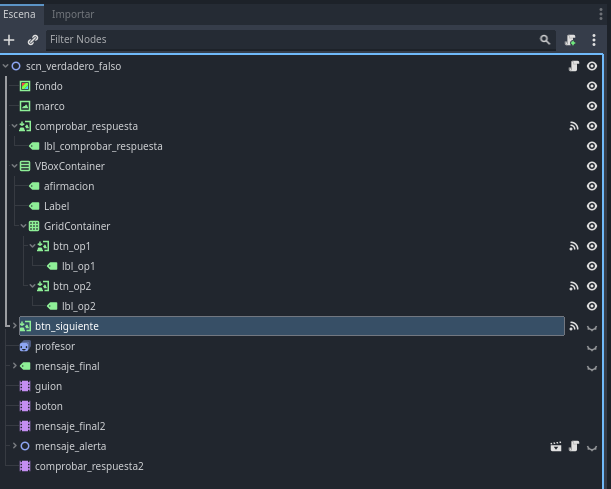
**JSON**:



ESCENA VERDADERO O FALSO

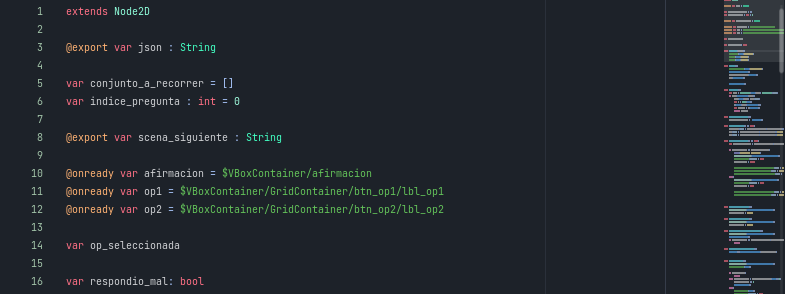
Esta escena esta basada en la escena de las preguntas, solo que modificada para que se adapte al formato de ‘Verdadero o Falso’

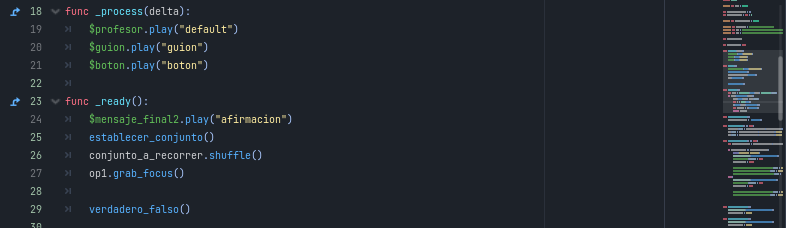
**Nodos**:

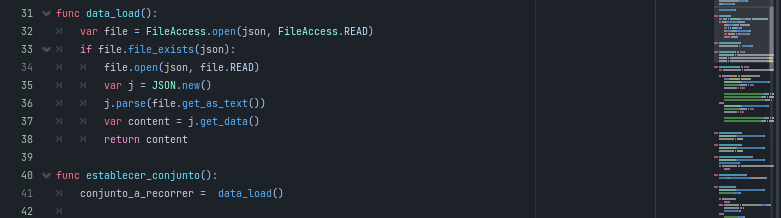


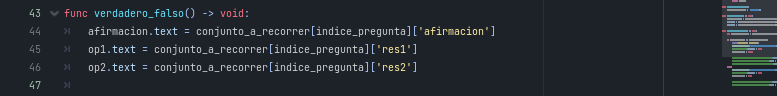


**Código**:

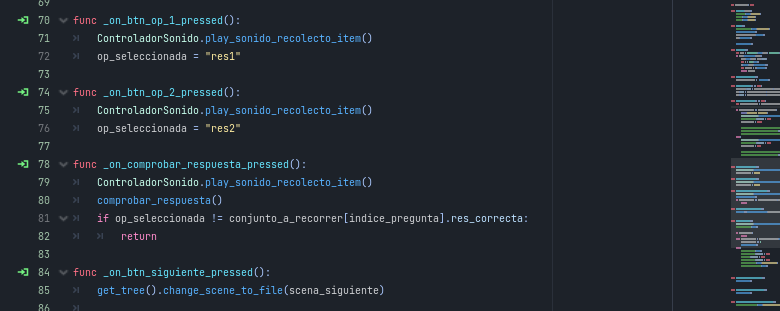


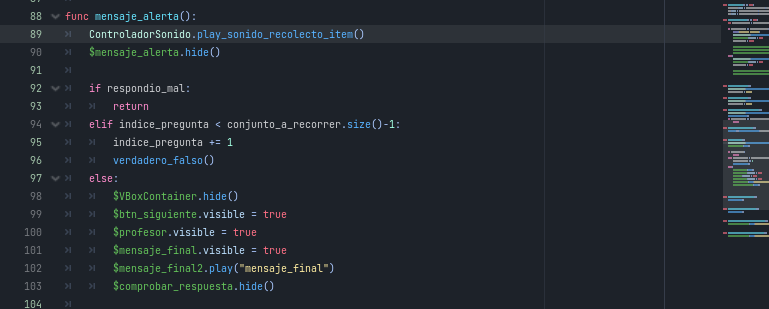






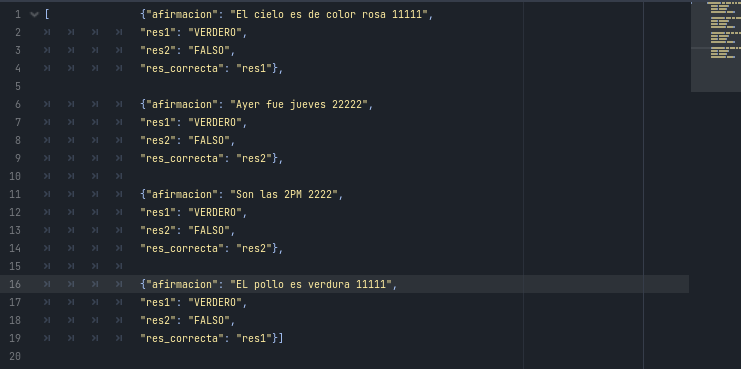








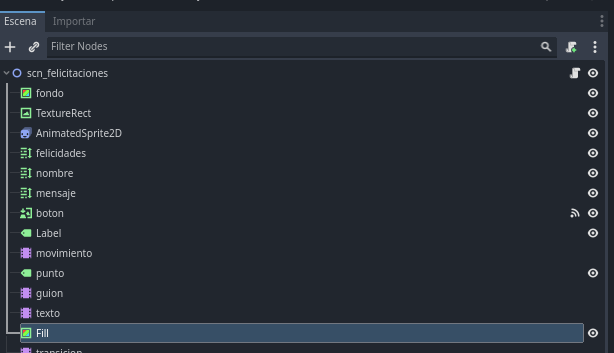
**JSON**:

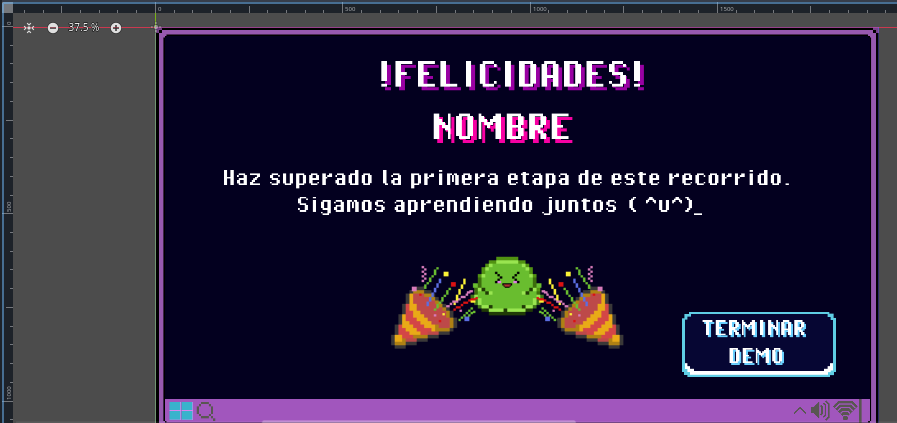


ESCENA FELICITACIÓN

Esta escena solo aparecerá al final de cada nivel completado exitosamente. En este caso como solo estará de muestra el primer nivel y este mensaje solo se mostrará una vez, el botón final tendrá como mensaje ‘Terminar demo’, y lo redirigirá a la introducción del juego.

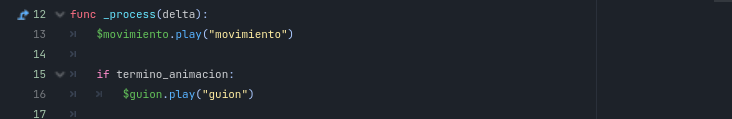
**Nodos**:

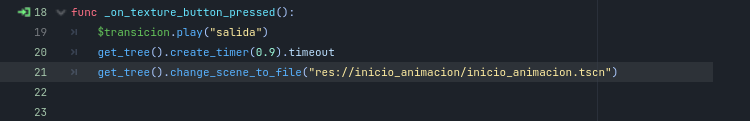


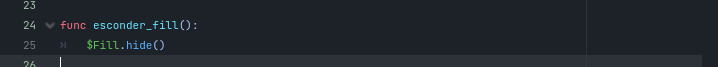


**Código**:

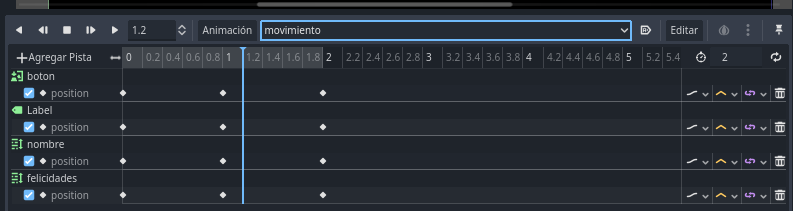


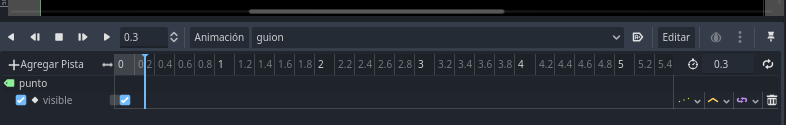


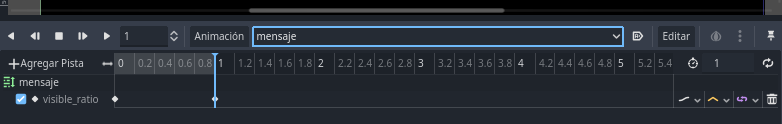


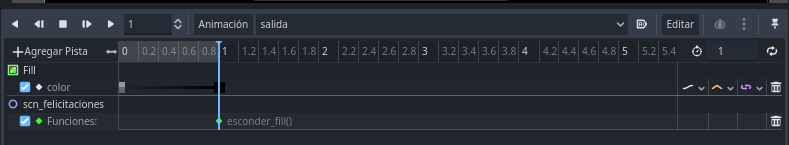


**Animaciones:**









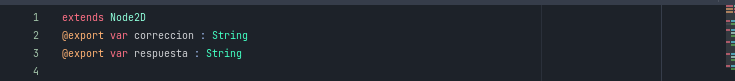
ESCENA ALERTA

Este mesaje se mostrará en caso de que el jugador se equivoque o responda correctamente a lo largo del juego.

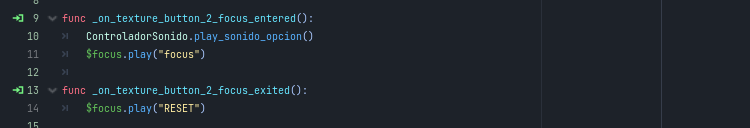


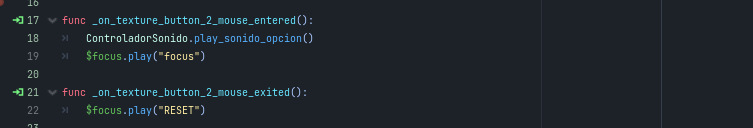


**Código**:





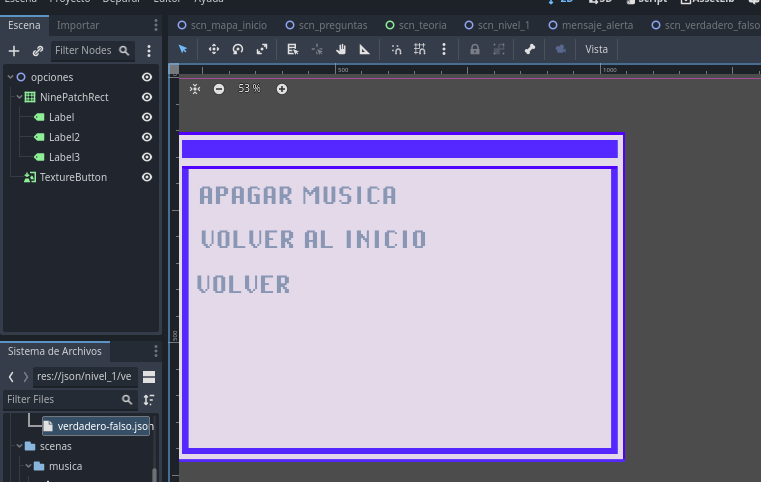




ESCENA: OPCIONES

Esta escena busca ser un menu de opciones, al cual se accederá presionando la tecla [ESQ], y en el cual podrá, apagar la música, o salir por completo de la actividad.

.

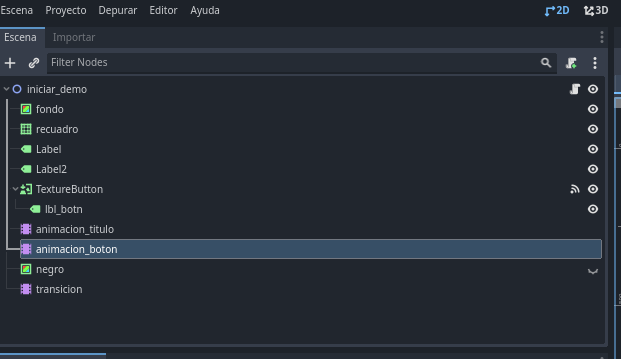


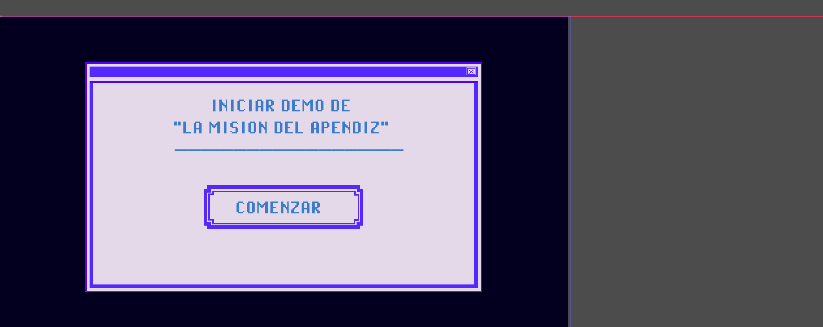
ESCENA: INICIAR DEMO

Con el fin de, en la exposicion del proyecto, hacer la muestra mas llevadera, sumamos una escena que se denomina “Iniciar demo”, para que quien desee probar el juego, dado que el juego ya va estar iniciado, vea como seria ingresar realmente por primera vez en este, es decir, con la animación del inicio, seguido del menu y el ingreso del nombre.

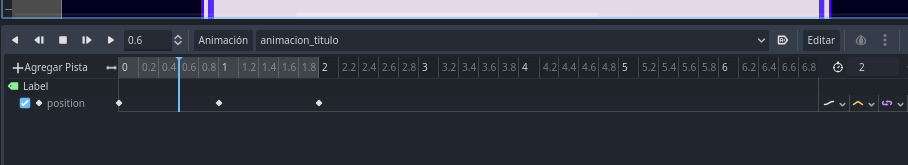
Se planea adicionar un botón que lo lleve directo al mapa de inicio, y uno mas que lo lleve a la muestra de una actividad.

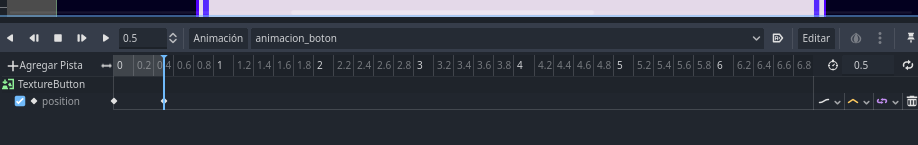
**Nodos**:

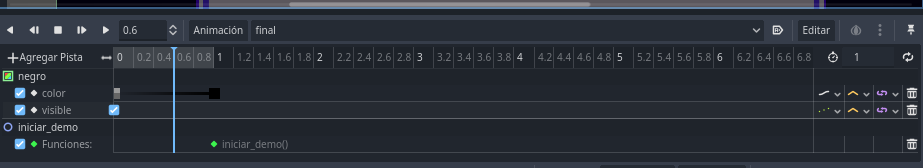




**Animaciones**:







**Código:**





BIBLIOGRAFÍA

DISEÑOS DE PERSONAJES Y MAPAS

* MortMort(2017)*[Guia de Asesprites para principiantes](https://youtu.be/Md6W79jtLJM?si=l4c8JiuslCRPtwO_)*
* Omel pixela(2022) *[Hacer pixelart](https://youtube.com/@omelpixela2906?si=l4ud-Frc-sHVLWcq)*
* Reece Geofroy(2021) *[How to pixelart](https://youtu.be/0I_OZ4qQJfY?si=tpWABkj1CWDWcDcf)*
* Los Cafes de Chivic(2021)  *[Graficos](https://youtu.be/hSvXW08TfME?si=C0MhlUhFcyq-69T7)*
* Nico Pardo(2020) *[Crear personaje](https://youtu.be/9mHMh3P0W68?si=I3fPygtTGeTcnOOz)*
* Mundoalexo Play(2023) *[Crear y animar Sprites](https://youtu.be/ziHRuQWfzaY?si=DuzDcJfN3z4N0OVp)*
* Omel Pixela(2022) *[Como dibujar un TileSet](https://youtu.be/Zg10c0zycVw?si=47Aj9ENhZliTIi6e)*

ESTUDIO DE PHYTON

* Dalto (2023) *[Curso de Python](https:/youtube.com/@soydalto?si=ECZV0E7S5VcGkbL_ )*

PROGRAMACION EN GODOT

* Adderly Céspedes(2018) *[Como hacer un juego en Godot 3](https://www.youtube.com/watch?v=vFbkF0JhSuI&list=PL5K_XeigIfdJeaJ3-_YgnzD711StmJkZh&pp=iAQB)*
* DevWorm(2023) *[How to make a RPG in Godot 4](https://www.youtube.com/playlist?list=PL3cGrGHvkwn0zoGLoGorwvGj6dHCjLaGd)*
* RayuseRP (2021) *[Start menu keyboard selection and shortcouts](https://youtu.be/hXXSWhsjp6M?si=i8XJXszFnoJNtmVP)*
* Leedeo Studio (2023) *[Curso Godot 4](https://www.youtube.com/watch?v=L3pFEk1HPCQ&list=PL5PTqiCIVoiVyA2qed1NE4uKejXEWM60e&pp=iAQB)*

PAGINA WEB

* Dalto (2020) *[Curso CSS](https://youtu.be/OWKXEJN67FE?si=k1cuABY-qT4ugz0A)*
* Dalto(2020) *[Curso HTML](https://youtu.be/kN1XP-Bef7w?si=EX7a0n8g7P1FzrQ7)*