Farming RPG

Vadillo Gutiérrez, Atenea

Informática y Comunicaciones

Desarrollo de aplicaciones multiplataforma

CIFP Juan de Colonia

21/05/2024

Contenido

1. Introducción

El siguiente proyecto final de desarrollo de aplicaciones multiplataforma se desarrollará en el mundo de los videojuegos de granjas que nos permitirá visualizar una propuesta atractiva. Con el objetivo de crear una experiencia de juego compacta pero completa, accesible en escritorio, se centra en la creación de una demostración técnica de un videojuego de granjas.

La demostración técnica presenta un entorno virtual detallado donde los jugadores pueden sumergirse en la gestión y el crecimiento de una granja a través de mecánicas estratégicas y elementos de simulación. A pesar de su alcance limitado, el objetivo es proporcionar una muestra representativa de las funcionalidades clave del juego completo. Para lograrlo, se enfrentará a desafíos técnicos significativos, como la optimización del rendimiento y la adaptación de la interfaz de usuario, asegurando una experiencia fluida y satisfactoria.

A medida que la demanda de juegos de simulación, como los de granjas, continúa en aumento, esta iniciativa busca ofrecer una experiencia de juego envolvente.

La demostración técnica se basa en un entorno virtual detallado donde los jugadores pueden probar lo que sería el resultado final de este juego de granjas con opcion a teclado o mando dentro del propio juego. A pesar de su limitada extensión, la demo pretende proporcionar una muestra representativa de las funcionalidades clave del juego completo.

1.1 Descripción del proyecto y finalidad del mismo

El proyecto tiene como objetivo principal explorar el entorno de desarrollo de Godot Engine 4, un motor de juego de código abierto altamente utilizado. Se llevará a cabo a través de la creación de un juego de granjas, lo que permitirá explorar las funcionalidades y herramientas que ofrece este motor, desde su interfaz de usuario hasta su capacidad para renderizar gráficos en 2D, así como otras funciones como el mapeo de entradas.

Sin embargo, más allá de simplemente crear un juego divertido, el propósito último es aprender y perfeccionar habilidades técnicas y de diseño. Se espera que la aplicación práctica de conocimientos en programación, diseño gráfico, animación y música se convierta en un aspecto central del desarrollo. Además, se enfatiza la importancia de la colaboración y comunicación efectiva entre los miembros del equipo para garantizar la cohesión y la calidad del producto final.

Este proyecto se ha inspirado en una amplia gama de juegos, entre los que se destacan títulos como **Stardew Valley, Potion Permit y Harvest Moon**, todos ellos célebres por su enfoque en la gestión de granjas y la vida rural. La influencia de estos juegos se reflejará en la mecánica y la estética del juego en desarrollo. Además, se explorará la integración de elementos de RPG por turnos, tomando inspiración de otras franquicias de renombre como **Pokémon, Undertale y Delta Rune**. Estos juegos no solo han sido referentes en sus respectivos géneros, sino que también ofrecen una rica fuente de ideas para enriquecer la experiencia de juego que se busca crear.

En resumen, el proyecto no solo busca ofrecer entretenimiento a los usuarios, sino también servir como plataforma de aprendizaje y desarrollo profesional para los participantes. Se aspira a crear una experiencia de juego sólida y atractiva, respaldada por un enfoque metodológico y una ejecución técnica impecable, lo que guiará cada paso del proceso de desarrollo.

Entre los objetivos de este proyecto están:

- Aprender sobre el motor de juegos “Godot Engine”.
- Implementar la lógica en C#.
- Crear una demo técnica de un juego de granjas.
- Crear un menú con ajustes funcionales.
- Dar al usuario un menu de pausa donde poder salir al menú o al escritorio y guardar.
- Guardar configuraciones.
- Menú con aspecto completamente personalizado.

Y partiendo de ahí los objetivos de la demo serían

- Mecanismos para de arar, regar y plantar.
- Tiendas funcionales.
- Enemigos con peleas por turnos
- Ajustes de controles, accesibilidad, gráficos y de sonido.
- Pop ups informativos para el usuario de cuando no puede plantar o vender

2. Análisis y diseño del proyecto

En esta sección se presentará el análisis de requisitos u objetivos y diseño de la aplicación desarrollada.

2.1 Planificación y especificación de requisitos

Los juegos de simulación de granjas son un subgénero de los videojuegos de simulación que se centran en la gestión y el desarrollo de una granja. Estos juegos permiten a los jugadores experimentar la vida agrícola, incluyendo la siembra de cultivos, la cría de animales y la administración de recursos. La popularidad de estos juegos radica en su capacidad para ofrecer una experiencia relajante y gratificante.

La historia de los juegos de simulación de granjas se remonta a la década de 1980, con títulos como **Harvest Moon**, que fue lanzado en 1996 para la consola SNES. **Harvest Moon** se convirtió en uno de los juegos más influyentes del género, estableciendo muchas de las mecánicas y temas que se han convertido en estándar.

En años recientes, el género ha visto una expansión significativa con la llegada de juegos como **Stardew Valley**, lanzado en 2016, que revitalizó el interés por el género en las consolas y PC. **Stardew Valley** es particularmente notable por su profundidad y la gran cantidad de actividades que ofrece, lo que ha llevado a una base de jugadores dedicada y ha inspirado a muchos desarrolladores a crear sus propias versiones de simuladores de granjas.

Godot es especialmente reconocido por sus capacidades robustas y eficientes para el desarrollo de juegos en 2D. El motor de juegos ofrece una serie de características diseñadas específicamente para facilitar la creación de juegos en 2D, como un sistema de nodos flexible, soporte para físicas, un editor de tiles intuitivo, y herramientas para animación y GUI. Godot permite a los desarrolladores trabajar en un entorno visual amigable, donde pueden construir y organizar sus escenas y assets de manera lógica y ordenada.

Como se verá más adelante, está disponible para Linux y para Windows, ya que Godot da la oportunidad de crear el ejecutable para ambos sistemas operativos. No es necesario ningún permiso tampoco, ya que no necesita instalación.

2.2 Construcción

Una vez planteados los requisitos que debe cumplir la aplicación en el presente apartado se explicará el planteamiento inicial y cómo fueron abordados.

2.2.1 Diseño

Inicialmente, este proyecto empezó como un juego de rol por turnos al estilo clásico, inspirado en la mecánica de enfrentar diversos enemigos a lo largo de un camino, siguiendo la dinámica que popularizó la franquicia Pokémon. Sin embargo, durante las primeras fases de desarrollo, cambio la dirección del videojuego al descargar los assets del juego, siendo estos de un juego de granjas.

Al ver los assets de un juego de granjas, cambie de perspectiva a un subgénero completamente diferente: el simulador de granjas. La inspiración vino de la mano de juegos mencionados anteriormente como Harvest Moon, Stardew Valley y Potion Permit, por lo que se decidió adoptar este enfoque y transformar el proyecto en un juego que fusionara elementos de RPG por turnos con la gestión de una granja.

El cambio de dirección no solo amplió los horizontes creativos, sino que también se ofreció la oportunidad de probar campos como el diseño y la programación de videojuegos. A través de este proyecto, el objetivo es adquirir experiencia práctica en el desarrollo de juegos, explorando las complejidades del motor de juego Godot y familiarizándome con sus nodos y funcionalidades.

Tomando en cuenta los contenidos del curso, decidí usar C# en lugar de GDScript. Aunque GDScript está integrado de forma nativa en Godot, opté por C# debido a su amplia adopción en la industria de los videojuegos. Esta elección me permitirá aplicar los conocimientos del curso y adquirir habilidades transferibles valoradas en la industria, aprovechando las ventajas de su sistema de tipos estáticos y su sólido soporte en Godot.

También, se usó el nodo TileMap que simplifica la creación de niveles al hacer una creación de mapas a través de azulejos en capas, los cuales son TileSets. Los TileSets son colecciones de imágenes con opciones de configuración como colisiones, animaciones (no usadas en este proyecto en principio) y datos personalizados llamados CustomData. Hay 4 capas en este proyecto, siendo estas: césped, tierra, semillas y colisiones.

La primera capa, llamada "césped", proporciona una superficie visual para que el personaje camine. Para lograr variedad visual, se empleó la función de aleatoriedad del TileMap, permitiendo que los TileSets se coloquen de manera aleatoria al dibujar un rectángulo, asignándoles diferentes probabilidades de aparición para cada TileSet.

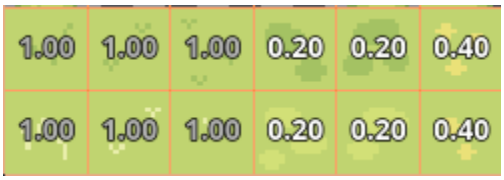

Figura 2.1: Probabilidades de aparición del Tile en el Tilemap en generación aleatoria

La segunda capa, denominada "tierra", desempeña un papel crucial en el juego, ya que es aquí donde se observa un cambio en el TileMap cada vez que se realiza la acción de arar. Esta acción es esencial para preparar el suelo ya que está relacionada con la siguiente capa.

La tercera capa, denominada “semillas”, es la capa designada para la plantación de semillas. Esta capa incorpora TileSets con CustomData que compuestos de un booleano (puede_semillas), el cual sirve para verificar si existen TileSets en la capa “tierra” justo en la posición donde está el jugador. Si detecta que hay TileSets presentes, automáticamente coloca un TileSet correspondiente a la primera fase de crecimiento de la planta.

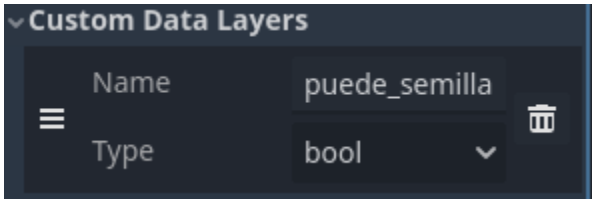

Figura 2.2: Imagen sobre el Custom Data (boolean) aplicado en la capa de semillas

La cuarta capa, llamada "colisiones", es esencial para la estructura del juego, ya que contiene todos los elementos que poseen colisión, como vallas. Estos componentes son cruciales porque establecen los límites del mapa.

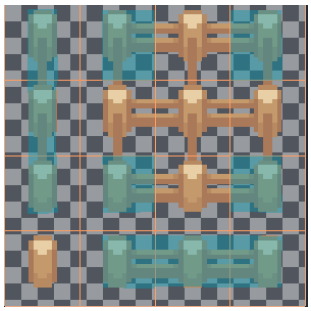

Figura 2.3: Colisiones (en azul) de las vallas utilizadas para las colisiones

Las vallas en específico tambien tienen un bitmask para poder pintarlas en el mapa y que se ajuste el Tile solo al dibujo, como por ejemplo al dibujar dos líneas conectadas y que se dibuje sola la esquina. El bitmask es una secuencia de bits usada para operar y manipular bits específicos dentro de un dato mediante operaciones lógicas.

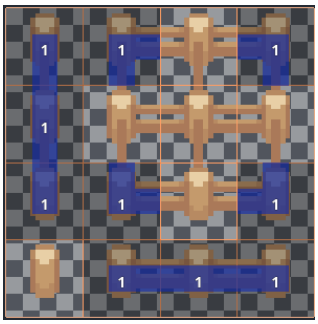

Figura 2.4: Bitmask usado para el cambio de tile a la hora de pintar las vallas

2.2.2 Implementación

El siguiente apartado detalla el proceso de implementación de código en el proyecto de videojuego, del cual se expondrá una demo técnica, abarcando desde la planificación inicial hasta la ejecución final. Exploraremos las herramientas utilizadas y desafíos encontrados durante el desarrollo.

En el desarrollo de videojuegos, la arquitectura no sigue un patrón rígido, sino que se basa en la asociación de scripts a nodos clave que representan componentes y acciones del juego. Estos scripts definen el comportamiento y las interacciones de los nodos, mientras que se puede conectar algunos métodos en el código para gestionar eventos específicos del juego.

La primera fase consistió en la integración de los assets, que abarcó desde la selección de recursos gráficos hasta la creación de animaciones y la definición de su lógica. Lo primero en las animaciones fue la construcción de las mismas a través del nodo AnimationPlayer, que sirve para crear animaciones, en este caso a través de los frames que se encontraban en las hojas de sprites. Para facilitar la implementación de lógica a la hora de cambiar la animación en función de la acción que realice el sprite, dependiendo esta de la entrada de teclado, se juntaron la hoja de animaciones base con las de acción. En acción al final solo se conservó la de arar por motivos de tiempo.

Antes de implementar las animaciones se mapearon las primeras acciones del jugador, dándoles un nombre de referencia para luego instanciarlo en el codigo, un input en teclado y otro en distintos mandos de consola (en el caso de las pruebas, Xbox).

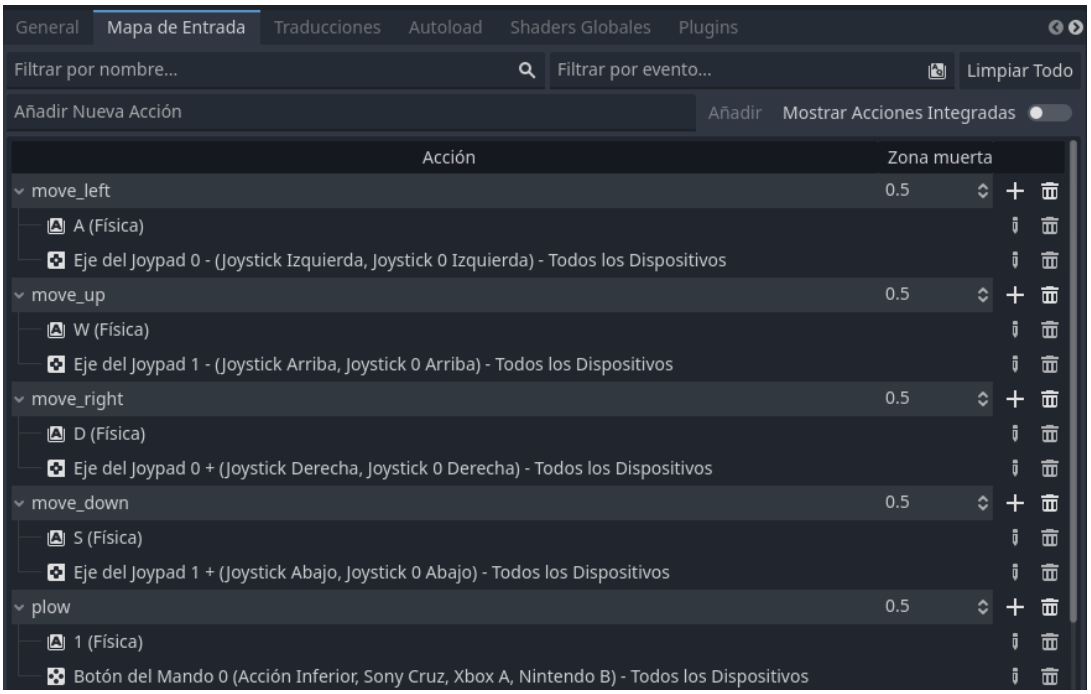

Figura 2.5: Mapeado en teclado y mandos

Para mejorar la gestión de las animaciones y simplificar su transición, se integró con eficacia el nodo AnimationTree que es un nodo que facilita la gestión de animaciones en proyectos de desarrollo de juegos. En esta herramienta se destacó su propiedad BlendSpace2D, que permite mezclar y cambiar entre diferentes animaciones bidimensionales.

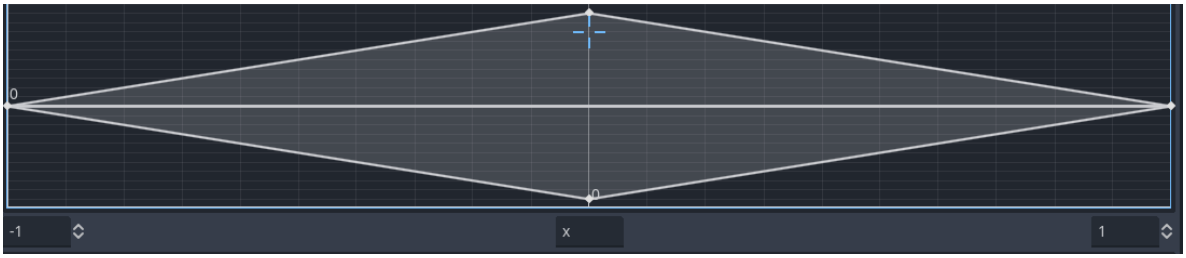

Figura 2.6: BlendSpace2D con sus coordenadas configuradas para el manejo de animaciones

Aquí se ha utilizado para representar el movimiento del personaje con movimientos como caminar, arar e idle en diferentes direcciones. Esta herramienta ha proporcionado una forma de manejar animaciones basadas en dos parámetros, como la velocidad horizontal y vertical, sin la necesidad de crear múltiples animaciones separadas para cada caso.

El code-behind detrás del AnimationTree para gestionar la lógica de cambio de animación es:

```
private void ActualizarParametros()
{
                        if (velocity == Vector2.Zero)
                        {
                           tree.Set("parameters/conditions/idle", true);
                           tree.Set("parameters/conditions/is_moving", false);
                        }
                        else
                        {
                           tree.Set("parameters/conditions/idle", false);
                           tree.Set("parameters/conditions/is_moving", true);
                        }

                        if (Input.IsActionJustPressed("plow"))
                        {
                           tree.Set("parameters/conditions/arar", true);
                           arar.Play();
                        }
                        else
                        {
                           tree.Set("parameters/conditions/arar", false);
                        }

                        if (direction != Vector2.Zero)
                        {
                           tree.Set("parameters/Idle/blend_position", direction);
                           tree.Set("parameters/Walk/blend_position", direction);
                           tree.Set("parameters/Arar/blend_position", direction);
                        }

}
```

En cuanto al codigo de las capas, a parte de lo mencionado en el apartado anterior sobre el diseño de las mismas, algunas como la capa “tierra” y “semillas” llevan codigo por detrás ya que es necesario para su modificación dinámica.

Aquí se ve parte del codigo usado para modificar la capa de tierra para implantar un Tile determinado en la posición del personaje en la capa “tierra”

```
if (Input.IsActionJustPressed("plow"))
{
	int hojaSpritesID = 12;
	Vector2I coordTierra = new(1, 1);
	mapa.SetCell(capaTierra, local, hojaSpritesID, coordTierra);
}
```

En la siguiente figura se ve la lógica de cómo se ponen semillas solo donde haya un Tile determinado en la capa “tierra”

```
if (Input.IsActionJustPressed("seeds"))
{
	TileData tileData = mapa.GetCellTileData(capaTierra, local);
	Variant semillas = false;

	if (tileData != null)
	{
		semillas = tileData.GetCustomData("puede_semillas");
	}
	else
	{
		GD.Print("No se puede plantar ahi"); //Convertir a popup
	}
	Vector2I cordSemilla = new(1, 0);

	if (((bool)semillas) == true)
	{
    mapa.SetCell(capaSemillas, local, plantasID, cordSemilla);
    CicloPlantas(local, cordSemilla, 3);
    GD.Print("Planta");
	}
}
```

Lo siguiente a la lógica en las capas, fueron los ciclos de crecimiento de las plantas, en los que se exploraron dos enfoques. Inicialmente, se intentó implementar un Timer con un evento "**timeout**", con el objetivo de avanzar a la siguiente etapa del crecimiento de la planta cada vez que el temporizador del nodo Timer concluyera.

Sin embargo, al encontrar dificultades con el funcionamiento del evento del Timer, se decidió optar por la creación de un método asíncrono. Este método, luego de una espera de 1 segundo, procedía a avanzar a la siguiente fase del crecimiento de la planta.

El método denominado "CicloPlantas" se encarga de modificar la apariencia del TileSet asignado, cambiándolo a una imagen correspondiente a la fase actual de la planta. Este proceso implica el incremento del valor de X en el Vector que determina las coordenadas de las diferentes fases de crecimiento dentro del TileSet.

```
private async void CicloPlantas(Vector2I posPlanta, Vector2I cordSemilla, int finalPlanta)
{
                                mapa.SetCell(capaSemillas, posPlanta, plantasID, cordSemilla);

   for (int i = 0; i <= finalPlanta; i++)
   {
      await Task.Delay(1000);
      Vector2I nuevaFase = new((cordSemilla.X + i), cordSemilla.Y); //Cambia el aspecto de la
planta para que crezca
      mapa.SetCell(capaSemillas, posPlanta, plantasID, nuevaFase);
   }
}
```

Paralelo a los ciclos de crecimiento de las plantas, se hizo un nodo para el menú principal para añadirlo al principio del videojuego. Este menú principal se compone de 3 botones: **Jugar**, **Ajustes** y **Salir**.

Figura 2.7: Menu de inicio

El botón de **Jugar** lleva directamente al mapa donde se desarrolla el juego, y está ligado a un evento "pressed".

```
public void _OnBtnJugarPressed()
{
   GetTree().ChangeSceneToFile("res://Scenes/Mapa1.tscn");
}
```

El botón de **Salir** cierra la aplicación y está ligado a un método "button_down". Es el único método ligado a este evento ya que con otros eventos no funcionaba el botón por la naturaleza de "Quit()"

```
public void _OnBtnSalirPressed()
{
   GetTree().Quit();
}
```

El botón de **Ajustes** oculta la parte del menú con estas opciones y las cambia por las de ajustes. Este método también está ligado a un evento "pressed".

```
public void _OnBtnAjustesPressed()
{
   main.Visible = false;
```

```
    ajustes.Visible = true;
}
```

En los ajustes es donde está la mayor parte del código del nodo **menú**, ya que los métodos anteriores no tienen mucha complejidad. Al entrar en el apartado de ajustes se ve un botón Atrás que devuelve al menu principal, revirtiendo el método del botón **Ajustes**, y un TabBar con los ajustes de gráficos y sonido.

Figura 2.8: Pantalla de ajustes graficos

Los ajustes de gráficos son los primeros que se ven. Cuentan con dos tipos de opciones: **Modo de ventana** y **Resolución** (solo en modo Ventana). Ambos controles son un Combobox con las opciones pertinentes. Se controlan con un evento ItemSelected asociado a un método que cambie el modo de ventana o la resolución dependiendo de la opción seleccionada.

```
private void _OnModoSelected(long index)
{
   switch (index) {
      case 0: //Pantalla completa
         DisplayServer.WindowSetMode(DisplayServer.WindowMode.Fullscreen);
         break;
      case 1:
         DisplayServer.WindowSetMode(DisplayServer.WindowMode.Maximized);
         break;
      case 2:
         DisplayServer.WindowSetMode(DisplayServer.WindowMode.Windowed);
         break;
   }
```

```
}
```

```
private Dictionary<string, Vector2I> resolucionesPantalla = new Dictionary<string, Vector2I>()
{
   {"720p", new Vector2I(1280, 720)},
   {"1080p HD", new Vector2I(1920, 1080)},
   {"1440p 2K", new Vector2I(2560, 1440)},
   {"2160p 4K", new Vector2I(3840, 2160)},
};

private void _OnResolucionSelected(long index)
{

DisplayServer.WindowSetSize(resolucionesPantalla.Values.ElementAt<Vector2I>((int)index));
}
```

Los ajustes de sonido tienen 3 opciones, coincidiendo con los buses de sonido (Master, Música y Sfx). En esta etapa fue cuando se incluyó el sonido del juego a través de los nodos **AudioStreamPlayer** para la música ambiental (añadidas al bus "Música") y **AudioStreamPlayer2D** para los efectos de sonido (añadidos al bus "Sfx")

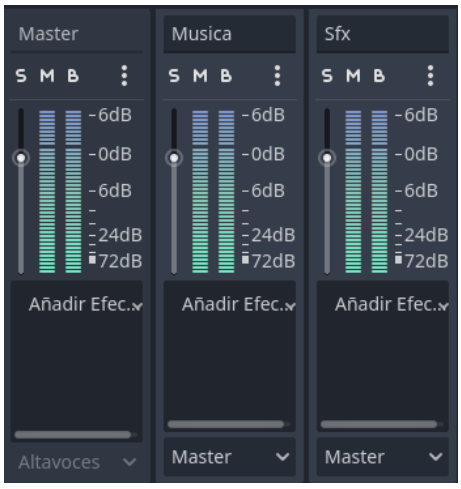

Figura 2.8: Buses de sonido

```
public void ValorSlider()
{
   volMusica.Value = Mathf.DbToLinear(AudioServer.GetBusVolumeDb(1));
   CambiarNumVolumen();
}
```

```
public void _OnVolMaestroValueChanged(float volumen)
{
   AudioServer.SetBusVolumeDb(1, Mathf.LinearToDb(volumen));
   CambiarNumVolumen();
}
```

La siguiente implementación es la de la acción de cosechar. Esta acción identifica el TileSet de la posición del personaje en la capa de semillas para ver si coincide con el del trigo completamente crecido (coordenada 4,0 en la hoja del TileSet) y elimina este Tile para dar impresión de que se ha cosechado. También añade 1 de trigo al inventario.

```
if (Input.IsActionJustPressed("crop"))
{
        Vector2I fase1 = new(4, 0);

        if(mapa.GetCellAtlasCoords(capaSemillas, local) == fase1)
        {
                mapa.EraseCell(capaSemillas, local);
                trigo++;
                GD.Print("La cantidad actual de trigo es: " + trigo);

   }
   else
        {
                GD.Print("Debes plantar trigo para cosecharlo");
        }
}
```

Al terminar con las acciones del Jugador, entró la tienda. La tienda es un StaticBody2D con un Area2D que si pasas por encima vende todo el trigo en el inventario, haciendo que las monedas sean trigo * 10. Esto se hizo con un evento en el Area2D del tipo BodyEntered:

```
public void _OnTiendaEntered(Node2D node)
{
	if (node.HasMethod("JugadorVender"))
	{
		if (trigo == 0)
		{
			GD.Print("No hay trigo para vender");
		}
		else
		{
			monedas += trigo * 10;
			trigo = 0;
			contador.Text = "= " + trigo;
			monedero.Text = "= " + monedas.ToString();
		}
	}
}
```

En esta fase tambien se añadió un método para cerrar el juego desde la escena de Mapa1. Los inputs aquí son Esc en el teclado y la tecla Atrás en el mando de Xbox y sus similares en otros mandos.

```
if(Input.IsActionJustPressed("salir"))
{
	GetTree().Quit();
}
```

En la fase actual tambien se añadió el sonido de error al llevar a cabo una acción que no se puede hacer como plantar donde no se ha arado para que el usuario sepa que son acciones “incorrectas”.

La ultima fase fue la de exportación a binario tanto para Windows como para Linux. Ambas exportaciones tienen los mismos requisitos para la máquina que los ejecute.

2.2.3 Pruebas

Las pruebas son esenciales en el desarrollo de videojuegos para garantizar calidad y estabilidad. Este apartado detalla los tipos de pruebas, procedimientos, herramientas y resultados obtenidos en cada fase de pruebas. Todas las pruebas se han ejecutado en tiempo real.

Prueba	Resultado esperado	Resultado real
Funcionamiento de las animaciones	Las animaciones responden a la entrada de teclas	La animación "Idle" se intercala con las animaciones de andar
Funcionamiento de las animaciones con el codigo corregido	Las animaciones responden a la entrada de teclas	Las animaciones si se reproducían cuando debían
Entrada de mando Xbox 360	El juego responde a la entrada de mando	El juego responde a la entrada de mando igual que a la de teclas
Implementación de las animaciones "plow" (arar)	La animación responde a la entrada de espacio (tecla) y A (Xbox)	No se reproduce la animación/el Sprite se queda en el fin de la animación hasta que se mueve
Creación del nodo AnimationTree	Solucion de los problemas con las animaciones "plow"	El Sprite se queda en el fin de la animación hasta que se mueve
Implementación de StateMachine en el nodo AnimationTree	Solucion de los problemas con las animaciones "plow"	Se ejecutan todas las animaciones a la vez
Cambio de StateMachine a BlendMachine2D	Solucion de los problemas con las animaciones "plow"	El eje Y esta invertido. El sprite andaba de espaldas hacia arriba y abajo
Corrección del eje Y	Solucion al movimiento invertido	Las animaciones "plow" se ejecutan cuando deben y el movimiento deja de estar invertido
Indicador de posición	Se imprime por consola la posición del Sprite al dar a espacio/A	Referencia nula al Sprite
Corrección en codigo de la referencia al Sprite	Se imprime por consola la posición (Vector2i) al dar a espacio/A	Se imprime la posición relativa en integer por consola
Cambiar el TileSet indicado	Se cambia el TileSet donde está el personaje por uno de tierra	Se cambian TileSets con la posición del personaje anterior
Funcionalidad botones menú	Los botones jugar, ajustes y salir funcionan según deben	Jugar abre la escena del juego, ajustes abre el menu de ajustes, pero salir no cierra la aplicación

Cambio de evento pressed a button_down en el boton salir	El boton salir cierra el juego	El boton salir cierra el juego
Funcionamiento del combobox de ajustes (modo ventana)	El tamaño de la ventana corresponde a la opcion seleccionada	El tamaño de la ventana corresponde a la opcion seleccionada
Funcionamiento del combobox de ajustes (resolución)	La resolución corresponde a la opcion seleccionada	La resolución corresponde a la opcion seleccionada
Funcionamiento del CustomData para las plantas	Las plantas solo están en TileSets de tierra	Las plantas solo están en TileSets de tierra
Colisiones	El sprite no puede pasar por las vallas donde tienen zona de colisión	El sprite no puede pasar por las vallas donde tienen zona de colisión
Crecimiento de plantas	Las plantas cambian de aspecto cada segundo (impresión de crecimiento)	Las plantas no cambian de aspecto o directamente aparecen en su fase final de crecimiento
Música del menu principal	La música suena cuando se inicia el videojuego	La música suena cuando se inicia el videojuego
Música del juego	La música suena cuando se pasa a la escena del juego	La música suena cuando se pasa a la escena del juego
Sonido de arar	Suena un sonido cuando se presiona la acción “plow”	Suena un sonido cuando se presiona la acción “plow”
Ajustes de volumen maestro	El volumen general se controla con un slider	El volumen general se controla con un slider
Ajustes volumen por separado	El volumen de la música y los efectos se ajustan por separado	El volumen de la música y los efectos se ajustan por separado
Eliminar la recursividad del método CicloPlantas	Las plantas crecen cada segundo	Las plantas no crecen o solo crecen hasta la segunda fase
Cambio de evento de Timer (timeout) a hilo	Las plantas crecen cada segundo	Las plantas crecen cada segundo
Crecimiento paralelo	Las plantas crecen, aunque haya otras creciendo	Las plantas crecen, aunque haya otras creciendo
Función para detectar trigo crecido	Al pulsar la entrada establecida, se debe imprimir por pantalla si el TileSet donde está el personaje es trigo crecido o no	Se imprime “trigo” si el TileSet se puede cosechar, y “no trigo” si no ha crecido del todo
Sonido de cosecha	Al presionar la acción “crop”, se reproduce un sonido	Al presionar la acción “crop”, se reproduce un sonido
Posición correcta de la tienda	La parte de abajo tiene colision y la de arriba se superpone al jugador	La parte de abajo tiene colision y la de arriba se superpone al jugador

Inventario	Se queda fijo a la camara	Se queda fijo a la camara
Cantidad de trigo	La cantidad de trigo sube con cada cosecha valida	La cantidad de trigo sube con cada cosecha valida
Tienda	Al pasar por la tienda, las monedas son trigo x 10 y el trigo es 0	Detecta siempre que el trigo es 0, a pesar de que en la etiqueta no
Tienda arreglada	Al pasar por la tienda, las monedas son trigo x 10 y el trigo es 0	Las monedas son trigo x 10 y el trigo es 0
Cerrar desde el juego	Al dar Esc, el juego se cierra entero	Al dar Esc, el juego se cierra entero
Sonido de error	Al cosechar sin trigo o al intentar plantar sin arar, deberia sonar un sonido de error	El error suena, pero es casi imperceptible
Subida de volumen de error	El sonido se puede percibir	El sonido se puede percibir

Tabla 2.1: Pruebas de aplicación

3. Recursos

A continuación, se procederá a exponer los diferentes recursos de hardware y software que utiliza Farming RPG para su funcionamiento.

3.1 Recursos hardware

Los recursos de hardware necesarios para este proyecto son un ordenador con una pantalla de 1920 x 1080px (aunque la aplicación tiene ajustes para cambiar la resolución, como se ha dicho en puntos anteriores), teclado y/o un controlador de Xbox 360.

El ordenador ha sido esencial tanto para la estructuración de nodos, cambios en sus características y programación de la lógica de algunos, como para la ejecución del juego y sus pruebas.

El controlador de Xbox 360 ha sido crucial para comprobar el mapeado de entrada del usuario. Su uso ha permitido garantizar una experiencia de juego fluida y cómoda, asegurando controles optimizados para aquellos que prefieren jugar con este tipo de dispositivos.

3.2 Recursos software

Entre las herramientas software utilizadas, se destaca:

- Godot 4.2.2: Una plataforma versátil y robusta, fundamental para la creación y desarrollo de elementos clave del videojuego, tales como las animaciones, el diseño del mapa y la configuración de la entrada del usuario.
- Visual Studio 2022 Community: Una herramienta de programación integral, imprescindible para la elaboración de scripts y la implementación de la lógica subyacente en el videojuego, garantizando así su funcionalidad y coherencia interna.
- Git y GitHub: Sistemas de control de versiones y almacenamiento, respectivamente, que han proporcionado un entorno seguro y eficiente para gestionar el desarrollo del proyecto, ofreciendo una solución alternativa de almacenamiento en caso de ser necesario.

El sistema operativo no importa, ya que el videojuego cuenta con ejecutable portable para Windows y para Linux

4. Planificación

En el siguiente apartado se analizará tanto la forma en la que fue planificado temporalmente el proyecto como el plan económico propuesto.

4.1 Planificación temporal

Durante la etapa inicial del proyecto, se priorizó la integración de los activos en el entorno del videojuego. Esta fase implicó la elaboración de los TileSets destinados al TileMap, así como la creación de animaciones y escenas pertinentes. Se diseñaron TileSets para garantizar la coherencia visual y funcionalidad en el TileMap, mientras que las animaciones se elaboraron mediante la disposición secuencial de frames, aprovechando la estructura proporcionada por la hoja de sprites.

La implementación de las animaciones abarcó diversos aspectos del juego, desde los movimientos del personaje hasta las interacciones con el entorno. Para gestionar estas animaciones de manera eficiente, se optó por la creación de un nodo AnimationTree, proporcionando así un enfoque estructurado para la lógica de animación. Esta fase duró todo el mes de marzo y la primera semana de abril.

Además, se desarrollaron escenas específicas para el jugador (utilizando el nodo CharacterBody2D) y para el mapa (mediante el nodo Node2D), asegurando así una organización coherente y modular del proyecto. La fase inicial culminó con la integración exitosa de estos elementos en el videojuego, sentando así las bases para las etapas subsiguientes del desarrollo. En esta fase también se implementó la lógica del TileMap explicada anteriormente.

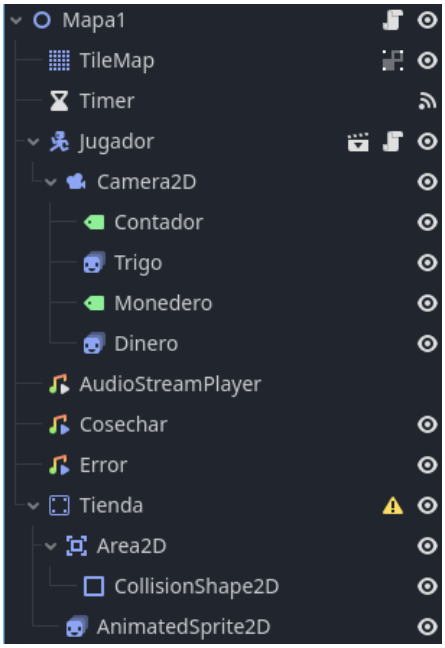

Figura 4.1: Árbol de nodos usados en la escena “Mapa”

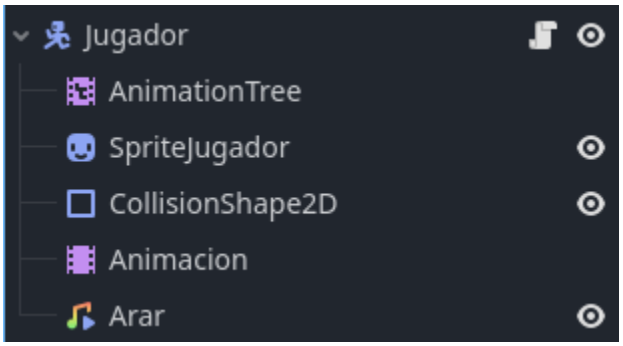

Figura 4.2: Árbol de nodos de la escena “Jugador”

Figura 4.3: Árbol de nodos de la escena “Menu”

La segunda fase del proyecto, se concentró en la implementación de la funcionalidad del juego mediante la creación y desarrollo de scripts específicos. Estos scripts fueron diseñados para controlar el movimiento y las acciones del personaje principal, así como para determinar cuándo se reproducen las animaciones correspondientes a cada acción. Se necesitaron mas o menos 4 dias para ello.

Además, se dedicó especial atención a la integración de la entrada por mando, lo cual constituye un aspecto crucial para la accesibilidad y la experiencia de juego. Se desarrollaron mecanismos que permiten al jugador controlar el personaje utilizando un mando de forma intuitiva y eficiente, asegurando así una experiencia de juego fluida y adaptable a diferentes preferencias de control.

En la tercera fase del desarrollo, se dedicó esfuerzo significativo a la creación de un menú principal completo y funcional, junto con la implementación de opciones de ajustes para mejorar la experiencia del usuario. Al no ser complejo, pero si necesitó gran cantidad de controles, se necesito una semana para terminar la parte grafica. Con los scripts se necesitaron tambien varios dias.

El menú principal se diseñó con el objetivo de ofrecer una interfaz intuitiva y atractiva que permitiera a los jugadores navegar fácilmente por las distintas opciones disponibles. Se incluyeron los botones Jugar, Ajustes y Salir para realzar la experiencia del usuario, así como una disposición clara y coherente de las diferentes secciones del menú.

En cuanto a los ajustes, se implementan opciones personalizables que permiten a los jugadores modificar diversos aspectos del juego según sus preferencias individuales. Esto incluye ajustes video y sonido, contribuyendo a una experiencia de juego más personalizada y satisfactoria para cada jugador.

En la sección de gráficos, se incluyen tres opciones de modo de ventana (pantalla completa, maximizado y ventana), junto con varias opciones de resolución para el modo de ventana sin maximizar. En la sección de sonido, se encuentran tres apartados de volumen destinados a controlar los distintos volúmenes: maestro, música y efectos o Sfx.

La etapa de desarrollo subsiguiente abordó la implementación de un inventario vinculado a la cámara y la creación de una tienda en el juego. El inventario se compone de dos nodos hijos de la cámara: un Label y un AnimatedSprite2D. El Label muestra la cantidad de trigo en posesión del jugador. Esta sección tardo 3 semanas.

Figura 4.4: Inventario y dinero

4.2 Planificación económica

En este apartado se analizará el coste de desarrollo de la demo técnica del videojuego Farming RPG.

En cuanto al desarrollo directo del proyecto, no se ha gastado dinero ya que los assets son sin royalties (siempre y cuando no se usen para comercializarlos directamente o a través de proyectos) y Godot Engine es software libre.

En caso de querer comercializar el videojuego, los assets visuales usados cuestan 3,99$ (3,70€). A parte del paquete de los assets usados para el videojuego, la misma diseñadora tiene un paquete de assets para la interfaz en su versión premium, ya que traen más diseños y es para uso comercial, costando también 3,99$ (3,70€).

Para su publicación en la tienda de Steam y Epic Store se deben pagar 100$ (92,75€) como tasa única, aunque se te devuelva el dinero al generar ingresos de 1000$ (927,54€). Al ser una demo técnica, se pondría gratis para que el cliente pruebe el juego antes de jugarlo.

A la hora de terminar el videojuego, se pondría un precio de más o menos 20€ como pago único (no habría suscripción mensual como puede existir en los juegos MMORPG), aunque se podría hacer un descuento de lanzamiento. También se patrocinaría en redes para llegar a más jugadores.

El total de la documentación sobre Godot (investigación sobre el software) y el proceso de construcción del videojuego (tanto en codigo como en manejo de nodos y assets en el entorno) han sido 3 meses, por lo que como sueldo de programadora cobraría 5.220€ netos.

5. Conclusiones finales

Para concluir la memoria del presente proyecto se analizará el grado de cumplimiento de objetivos fijados y las posibles ampliaciones futuras.

5.1 Grado de cumplimiento de objetivos

En el transcurso de este proyecto, se han alcanzado importantes objetivos relacionados con el aprendizaje y la implementación en Godot Engine, así como el uso de C# en lugar de GDScript. Se ha logrado una comprensión sólida de este motor de juegos, lo cual ha permitido desarrollar una demo técnica funcional de un juego de granjas.

La demo incluye exitosamente las mecánicas básicas de arar y plantar, proporcionando una base interactiva esencial para la experiencia del usuario. Además, la tienda implementada permite vender el trigo, enriqueciendo la jugabilidad y añadiendo profundidad al entorno virtual.

Otro logro significativo ha sido la implementación de ajustes de gráficos y sonido. Estos ajustes ofrecen una experiencia de usuario adaptable y personalizable, lo que es fundamental para atender a las diversas necesidades y preferencias de los jugadores. Esta flexibilidad mejora la satisfacción del usuario, contribuyendo positivamente a la experiencia general del juego.

El objetivo de utilizar C# en lugar de GDScript se ha cumplido exitosamente. Este enfoque ha permitido una programación más estructurada y familiar, facilitándome la implementación de funcionalidades esenciales del juego, sobre todo con la api de diferencias entre C# y GDScript mencionada en las referencias.

Aunque algunos objetivos como el guardado de configuraciones, el menú de pausa y los pop-ups informativos no se han completado, los avances realizados constituyen una base sólida para futuras mejoras. La creación de un menú personalizado ha comenzado a delinear un estilo único y coherente para la interfaz del juego.

La experiencia adquirida durante este proyecto ha sido invaluable, proporcionando un conocimiento práctico y aplicable tanto en el uso de Godot Engine como en la programación con C#. Este aprendizaje continuo asegura que los objetivos pendientes puedan abordarse con mayor eficacia en próximas iteraciones del proyecto.

En resumen, aunque quedan áreas por perfeccionar, los logros alcanzados hasta ahora demuestran un avance significativo en el desarrollo del juego de granjas y establecen un punto de partida robusto para continuar mejorando y refinando la demo técnica. La base construida permite visualizar un camino claro hacia la finalización y el perfeccionamiento del juego, asegurando una experiencia de usuario rica y envolvente.

5.2 Mejoras a futuro

En el marco de las futuras expansiones, se incorporarán los objetivos hasta ahora no alcanzados, para completar lo que sería una versión con una interfaz de usuario completa, entre otras cosas. Asimismo, se proyecta la introducción de mejoras sustanciales destinadas a diversificar y enriquecer el entorno de juego.

Entre estas mejoras, se destacan la inclusión de la capacidad de cultivar una variedad de cultivos en la extensión de tierra de la granja, brindando al jugador la oportunidad de experimentar con diferentes tipos de plantaciones con distintos tipos de crecimientos basados en el ciclo día/noche. Además, se contempla la mejora de la tienda para que los jugadores podrán adquirir suministros agrícolas, mejoras en herramientas y otros elementos esenciales para el desarrollo de su granja.

Adicionalmente, se planea la incorporación de la ganadería como una faceta fundamental de la vida en la granja, permitiendo a los jugadores criar y cuidar una variedad de animales, como vacas y ovejas, ofreciendo así nuevas oportunidades de gestión y generación de ingresos. Por otra parte, se concibe la introducción de una trama narrativa inmersiva que sumerja al jugador en la historia de la granja y la comunidad circundante, ofreciendo desafíos, misterios y emocionantes descubrimientos a lo largo de su progreso.

Como parte integral de estas ampliaciones, se contempla la creación de un pintoresco pueblo donde los jugadores podrán interactuar con una variedad de personajes, desde vecinos amistosos hasta comerciantes y artesanos locales, cada uno con su propia historia y personalidad única. A través de estas interacciones, los jugadores podrán desbloquear nuevas misiones, obtener valiosos recursos y descubrir secretos ocultos que enriquecerán su experiencia en la granja.

En resumen, estas futuras mejoras prometen elevar la experiencia de juego a nuevas alturas, ofreciendo una experiencia agrícola más profunda, inmersiva y satisfactoria para los jugadores.

6. Guía de uso

No es necesaria la instalación, ya que tanto para Windows como para Linux son portables.

Al ejecutar se encuentra la pantalla de inicio con 3 botones: el primero abre el juego, el segundo el menu de ajustes y el tercero cierra la aplicación. El menu de ajustes tiene dos opciones: Gráficos y Sonido, los cuales se explican a continuación.

- Gráficos: La sección de gráficos se enfoca en como quiere el jugador que se vea el juego, contando con ajustes para elegir el modo de ventana. En caso de elegir el modo ventana hay ajustes para elegir entre distintas soluciones estándar (HD, 2K, etc.). Por defecto la aplicación se encontrará en modo pantalla completa, ignorando la resolución elegida.
- Sonido: La sección de sonido controla los volúmenes de la aplicación. Se puede ajustar el volumen maestro, bajando o subiendo el volumen general de la aplicación, o se pueden ajustar los volúmenes de música y efectos de sonido (Sfx) por separado.

En el apartado directo del juego, lo primero que se ve es el jugador, el cual se mueve con las teclas WASD (W para arriba, A para la izquierda, S para abajo y D para la derecha), o en caso de usar mando, con el joystick izquierdo. El jugador tambien podrá arar el suelo (tecla 1 en teclado, boton A en Xbox o su equivalente en otros mandos), plantar semillas (espacio en teclado, boton B en Xbox o su equivalente en otros mandos) y cosechar cuando la planta haya crecido (tecla 2 en teclado, boton X en Xbox o su equivalente en otros mandos).

Para plantar semillas es necesario que el suelo este arado, ya que si no la semilla no entrará en el suelo, y aparecerá un sonido de error para indicar que no se puede plantar ahí. Lo mismo ocurre cuando se intenta cosechar trigo que no ha terminado de crecer.

Con el trigo ya cosechado se puede pasar por el puesto de la tienda (al iniciar el juego, encima del jugador) y venderlo por 10 monedas.

Para salir, se debe presionar escape en teclado o boton atrás en Xbox o su equivalencia en otros mandos.

7. Bibliografía

05/03/2024 Juego de Plataformas 2D/Godot Tutorial/Introducción Godot/1-Capitulo/Programacion Videojuegos www.youtube.com/watch?v=F3T_ZhllzJs&t

05/03/2024 Juego de Plataformas 2D/Godot Tutorial/Crear el Mapa/Godot/2-Capitulo/Programacion videojuegos www.youtube.com/watch?v=SR7mdh0_i6Q&t

07/03/2024 GODOT Desde 0/Movimiento Jugador/Primer Script/Tutorial/ 4-Capitulo/Programación Videojuegos https://www.youtube.com/watch?v=fya91wv1OPI&t

09/03/2024 Juego de Plataformas 2D/Godot Tutorial/Movimiento Personaje/Godot/3-Cap/Programacion videojuegos https://www.youtube.com/watch?v=NeUS3Ytjty4

12/03/2024 Juego de Plataformas 2D/Godot Tutorial/Recoger Monedas/Godot/4-Cap/Programacion videojuegos www.youtube.com/watch?v=u99myfBJDlc&t

20/03/2024 How to Create WORKING CROPS in Godot www.youtube.com/watch?v=QK_uI-m6bpA&t

20/03/2024 How to Make a 2D FARMING Game in Godot (step by step) www.youtube.com/watch?v=QnOQNkglXso&t

27/04/2024 INTRODUCCIÓN A GODOT 4 [10] ANIMATIONPLAYER, Sistema de ANIMACIONES | Indie Game Dev https://www.youtube.com/watch?v=v5uZuyelKRQ

27/03/2024 INTRODUCCIÓN A GODOT 4 [11] ANIMATIONTREE, Gestor de Transiciones | Indie Game Dev https://www.youtube.com/watch?v=hfQkI5zQY3w

30/03/2024 Hiding the Mouse Cursor in Godot 4 https://www.youtube.com/watch?v=QIiS5WK2z0Q

30/03/2024 063 - Viewports - Múltiples Resoluciones - Godot 3 www.youtube.com/watch?v=ZveIbgizF28&t

30/03/2024 Godot 4 Making the Game Fullscreen and adding Splash Screen https://www.youtube.com/watch?v=-iGL2wkARj0

30/03/2024 Main Menu (Godot 4) https://www.youtube.com/watch?v=oDtDuwCPasg

06/04/2024 Animation Tree State Machine Setup w/ Conditions & BlendSpace2D - Godot 4 Resource Gatherer Tutorial www.youtube.com/watch?v=WrMORzl3g1U&t

09/04/2024 INTRODUCCIÓN A GODOT 4 [13] TILEMAP, Sistema de Tiles 2D | Indie Game Dev https://www.youtube.com/watch?v=XVSbjqjJhUQ&t

16/04/2024 Godot 4 TileMap Tutorial Ep 2 | Terrains / Autotilling https://www.youtube.com/watch?v=uXZuitdUPP8&t

16/04/2024 Godot 4 TileMap Tutorial Ep 3 | Placing Tiles with a Mouse Click https://www.youtube.com/watch?v=PSEPHO8ukjI&list=PLflAYKtRJ7dwtqA0FsZadrQGal8lWp-MM&index=3

22/04/2024 ¿Como cambiar la pantalla de inicio en Godot? (Splash Screen) | Tutorial Godot 4 https://www.youtube.com/watch?v=4FyH-wZLuOU

23/04/2024 How To Change The Window Mode And Resolution In Godot 4 https://www.youtube.com/watch?v=YsdkcPV0BAo

23/04/2024 How To Create An Options Menu In Godot! https://www.youtube.com/watch?v=fFIST_4wmyI

26/04/2024 Get access to custom tile data in C# in Godot 4 Beta https://www.reddit.com/r/godot/comments/xjdarh/get_access_to_custom_tile_data_in_c_in_godot_4/

26/04/2024 Documentación TileMap https://docs.godotengine.org/en/latest/classes/class_tilemap.html#class-tilemap-method-get-cell-tile-data

27/04/2024 Godot4: Como agregar colisiones a un TileMap (Curso intensivo del TileMap) https://www.youtube.com/watch?v=0Q2t_oHzZFA

28/04/2024 Godot 4 TileMap Tutorial Ep 6 | Plant Growth https://www.youtube.com/watch?v=6h4QSpccqIY&list=PLflAYKtRJ7dwtqA0FsZadrQGal8lWp-MM&index=6

29/04/2024 ¡5 Tips en Godot que quizás no conozcas! https://www.youtube.com/watch?v=z76J58ecAFM

30/04/2024 Godot 4 Timer Node Tutorial https://www.youtube.com/watch?v=Zf6awHRr7bU

30/04/2024 Como crear un timer! (Godot Engine 4) https://www.youtube.com/watch?v=TyPalRr_jpo

30/04/2024 How to Create WORKING CROPS in Godot https://www.youtube.com/watch?v=QK_uI-m6bpA&list=PL3cGrGHvkwn3zyVj-lHM1aGYhNv8E0HBS&index=2

01/05/2024 Every Node In Godot 3.5+, Episode 5 - Timer | Godot C# Timer Node | Mono | .NET | Tutorial | Example https://www.youtube.com/watch?v=2k4aoZm742o

02/05/2024 Importar muestras de audio - Godot Docs https://docs.godotengine.org/es/4.x/tutorials/assets_pipeline/importing_audio_samples.html

02/05/2024 Música de Fondo - Sunny Land en Godot - Cap: 18 https://www.youtube.com/watch?v=A2GYVhqo7WQ

02/05/2024 How To Create Sound Settings In Godot https://www.youtube.com/watch?v=yWvqvjLJ5Ko&list=PLhBqFleCVBkXQiE8Nm4Co_1iJJ4L7UIzr&index=16

04/05/2024 C# API differences to GDScript https://docs.godotengine.org/en/stable/tutorials/scripting/c_sharp/c_sharp_differences.html

06/05/2024 Documentación TileMap https://docs.godotengine.org/en/stable/classes/class_tilemap.html#class-tilemap-method-set-cell

06/05/2024 Scripting de varios lenguajes https://docs.godotengine.org/es/4.x/tutorials/scripting/cross_language_scripting.html

14/05/2024 How to Make a SELL ZONE in Godot https://www.youtube.com/watch?v=knmOs61W-lI&list=PL3cGrGHvkwn3zyVj-lHM1aGYhNv8E0HBS&index=6

19/05/2024 Introducción a GODOT 4 [15] 👉 EXPORTAR a WINDOWS, LINUX y HTML5 | Desarrollo de Videojuegos https://www.youtube.com/watch?v=pXK-CotnVIM

Assets:

08/03/2024 Sprout Lands - Sprites - Basic pack https://cupnooble.itch.io/sprout-lands-asset-pack

03/05/2024 Sonidos granja arar https://depositphotos.com/es/sound-effects/farming-dig.html

03/05/2024 Musica de fondo https://pixabay.com/es/music/

05/05/2024 Musica de granja https://pixabay.com/es/music/search/farm/

13/05/2024 Sonidos cosechar https://depositphotos.com/es/sound-effects/cosecha.html

13/05/2024 Asset para la tienda https://josie-makes-stuff.itch.io/pixel-art-farming-assets

19/05/2024 Sonido error https://pixabay.com/es/sound-effects/search/error/

(No usado, pero si mencionado) 20/04/2024 Sprout Lands UI Pack https://cupnooble.itch.io/sprout-lands-ui-pack

Investigación:

19/05/2024 Cuanto cobra un programador de videojuegos https://www.bing.com/chat?q=Microsoft+Copilot&FORM=hpcodx

19/05/2024 Stardew Valley Wiki https://es.stardewvalleywiki.com/Stardew_Valley_Wiki

19/05/2024 Harvest Moon https://vandal.elespanol.com/sagas/harvest-moon