Desarrollo del videojuego “La Tienda De Goblina”

Autor: Josep Burgués Calvo
Tutor: Gus Marcos Ballester
Profesor: Joan Arnedo Moreno

Ingeniería Informática
Videojuegos

18 de junio de 2023

Créditos/Copyright

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Desarrollo del videojuego “La Tienda De Goblina”*
Nombre del autor:	*Josep Burgués Calvo*
Nombre del colaborador/a docente :	*Gus Marcos Ballester*
Nombre del PRA:	*Joan Arnedo Moreno*
Fecha de entrega (mm/aaaa):	*06/2023*
Titulación o programa:	*Ingeniería Informática*
Área del Trabajo Final:	*Videojuegos*
Idioma del trabajo:	*Español*
Palabras clave	*Videojuegos, Juego casual, Puzle*

Resumen del Trabajo (máximo 250 palabras):

Este proyecto pretende cubrir todas las fases de desarrollo del videojuego “La Tienda De Goblina”, deteniendo la producción una vez obtenido un vertical slice que cumpla con los estándares de calidad impuestos por un proyecto de final de grado.

En “La Tienda De Goblina” el jugador ayudará a Goblina a descubrir el encanto detrás de los artefactos que recolectará durante su aventura, y de esta manera, obtener la mejor venta posible. Este proceso tendrá un enfoque de tipo puzle en el que, mediante el uso de pictogramas, el jugador intentará adivinar la frase o descripción qué hace dicho artefacto atractivo.

Videojuego 2D desarrollado mediante el motor gráfico Godot. Su desarrollo no solo plantea desafíos de la Ingeniería del Software, también de Gestión de Proyectos y Diseño de Videojuegos. Este documento cubre las motivaciones detrás de este proyecto, la estructura del desarrollo de un videojuego, análisis de los sistemas del juego y las cuestiones técnicas relacionadas con su implementación.

Abstract (in English, 250 words or less):

This project aims to cover all the phases in the development of the video game “Goblin’s Shop”, stopping the game’s production once attained a vertical slice that fulfills the quality standards imposed by a bachelor’s degree final project.

In “Goblina’s Shop” the player will assist Goblina in discovering the charm behind the artifacts that she will collect during her adventure, and in this manner, to obtain the best possible selling price. This process will have a puzzle like approach in which, through the usage of pictograms, the player will try to deduce the sentence or description that makes said artifact attractive.

2D videogame developed using Godot engine. It’s development not only arises challenges regarding

Software Engineering, but also of Project Management and Game Design. This document covers the motive of this project, the structure of a game develppment, an analysis of the game systems plus their intentions and the technical issues encountered during its implementation.

Dedicatoria/Cita

Me gustaría dedicar este trabajo a mis padres. Siempre me han apoyado en mis estudios y me han animado a perseguir mis metas. Espero en el futuro seguir progresando como desarrollador para poder dedicarles muchos más videojuegos.

Agradecimientos

Agradezco a mi tutor de TF sus consejos a la hora de orientar el proyecto por el buen camino y distribuir la tareas necesarias. También a mi profesor de la academia de música Bad Fader por brindarme los recursos necesarios para grabar las dos músicas del videojuego. Finalmente, a todas mis amistades que me han brindado su apoyo e interés durante el desarrollo del proyecto.

Resumen

Este proyecto pretende cubrir todas las fases de desarrollo del videojuego “La Tienda De Goblina”, deteniendo la producción una vez obtenido un *vertical slice* que cumpla con los estándares de calidad impuestos por un proyecto de final de grado.

En “La Tienda De Goblina” el jugador ayudará a Goblina a descubrir el encanto detrás de los artefactos que recolectará durante su aventura, y de esta manera, obtener la mejor venta posible. Este proceso tendrá un enfoque de tipo puzle en el que, mediante el uso de pictogramas, el jugador intentará adivinar la frase o descripción qué hace dicho artefacto atractivo.

Videojuego 2D desarrollado mediante el motor gráfico Godot. Su desarrollo no solo plantea desafíos de la Ingeniería del Software, también de Gestión de Proyectos y Diseño de Videojuegos. Este documento cubre las motivaciones detrás de este proyecto, la estructura del desarrollo de un videojuego, análisis de los sistemas del juego y las cuestiones técnicas relacionadas con su implementación.

Palabras clave

Videojuegos, Juego casual, Puzle

Abstract

This project aims to cover all the phases in the development of the video game “Goblin’s Shop”, stopping the game’s production once attained a vertical slice that fulfills the quality standards imposed by a bachelor’s degree final project.

In “Goblina’s Shop” the player will assist Goblina in discovering the charm behind the artifacts that she will collect during her adventure, and in this manner, to obtain the best possible selling price. This process will have a puzzle like approach in which, through the usage of pictograms, the player will try to deduce the sentence or description that makes said artifact attractive.

2D videogame developed using Godot engine. It’s development not only arises challenges regarding Software Engineering, but also of Project Management and Game Design. This document covers the motive of this project, the structure of a game develppment, an analysis of the game systems plus their intentions and the technical issues encountered during its implementation.

Keywords

Videogames, Casual game, Puzzle

Notaciones y Convenciones

Formato de negrita: Elementos clave o de estructura de la exposición.

Formato de cursiva: Términos, en su mayoría anglosajones, empleados con frecuencia en la industria de videojuegos para describir un videojuego o elementos de su desarrollo. (Consultar anexo Glosario)

Formato de subrayado: Nombres de clases, variables, métodos y nombres de archivo. Empleado durante la descripción del diseño en apartado Diseño.

Índice

Figuras y tablas

Índice de figuras

Índice de tablas

1. Introducción

1.1. Prefacio

Este TF propone el desarrollo de un videojuego para escritorio titulado “La Tienda De Goblina”. El juego propone un *gameplay loop* claro y definido que combina sistemas habituales en la plataforma móvil (sistema *merge*) con sistemas más esotéricos (“negociar” por medio de un sistema pictográfico y desbloquear así el encanto de los artefactos).

El juego se encontraría enmarcado dentro del género casual. En este género proliferan propuestas en las que las sesiones de juego son cortas pero entretenidas. Además, el desafío de concentración exigido es bajo, ya sea mediante el planteamiento de normas simples o controles fáciles de operar. Suelen ser una forma habitual de escapar el estrés o aburrimiento diario sin sentir que se está asumiendo un compromiso que entra en conflicto con las responsabilidades externas al juego.

“Personalmente, es un desafío tanto a nivel técnico como de diseño.”

Desde hace tiempo dedico esfuerzos al análisis de la plataforma de juegos casuales en búsqueda de mecánicas y propuestas que consigan ese equilibrio tan complicado entre mantener el interés del jugador por su desafío pero no exigir de este una mayor cantidad de recursos que podría utilizar en las cosas que de verdad le importan. A su vez, como jugador habitual del género RPG, siempre he querido explorar más sobre la gestión del inventario. De estas dos vertientes aparece la mecánica *merge*.

Por otro lado, considero que esa mecánica por sí sola no es lo suficientemente atractiva como para mantener el interés del jugador. Pero si es un gran vehículo para conducir su comportamiento. Por eso, he decidido combinarlo con un sistema de acertijos en el que el jugador deberá adivinar el verdadero atractivo de su mercancía y en el que se le facilitarán muchas oportunidades para volver a intentarlo. Con ello, quiero fomentar un proceso de prueba y error en el que se minimiza la sensación de fracaso. Este hecho, beneficia el carácter iterativo de los videojuegos casuales.

Pienso que el desarrollo de una idea original me permite demostrar las aptitudes adquiridas hasta ahora en el grado. Esta propuesta de TF me obliga a ser concreto con mis ideas, valorar por encima de todo la implementación, el prototipado y el producto entregable. Quiero demostrar entender las mecánicas en el diseño del juego por las que se opta mediante el análisis y tener las capacidades técnicas para producir un *vertical slice* con un nivel de calidad adecuado.

1.2. Definición

Este trabajo parte como punto de partida con: mis conocimientos adquiridos gracias al grado, los conocimientos de programación en *Pyhton* y en *C++* adquiridos por cuenta propia y una limitada experiencia en el uso de algunos *engines* para el desarrollo de videojuegos. A nivel de recursos, la idea ha sido desarrollada durante el periodo de propuesta de TF y no se ha producido de forma previa ningún recurso, por lo que serán fabricados o adquiridos durante el desarrollo del proyecto.

La intención de este TF a nivel personal es la de crear currículo para en el futuro dedicarme a nivel profesional en la industria del videojuego. Creo que a pesar de la simplicidad de muchos juegos casual, crear un diseño simple pero entretenido es un desafío mayor a lo que puede aparentar en un inicio. A su vez, quiero adquirir experiencia en el uso de Godot para en el futuro emplearlo como herramienta de prototipado de ideas.

Este proyecto explora la implementación de un sistema pictográfico como medio para las negociaciones que realizará el jugador durante la sesión de juego. En la actualidad, los sistemas que involucran el acto de “negociar” en los videojuegos suelen ser representados mediante el uso de pruebas y mecánicas que no guardan relación directa con la acción que abstraen. En este TF se aspira a crear innovación bajo la siguiente premisa, ¿Y si el precio de venta obtenido dependiese de los conocimientos que posee el proveedor sobre su mercancía? De esta forma, se lanza al jugador el desafío de descubrir qué hace único a cada una de sus mercancías. Cabe destacar, que un elemento muy importante de la negociación como son las preferencias del consumidor ha sido apartado por la enorme complejidad del sistema final.

Esto se justifica en otras de las finalidades del proyecto, producir una experiencia de juego casual en la que los sistemas son intuitivos y fáciles de comprender. Se espera que el videojuego final sea auto explicativo, reduciendo así el coste de entrada. El juego ha de invitar a sesiones cortas, en las que independientemente de la duración, es posible extraer entretenimiento. Tanto la interfaz de usuario, diseño artístico y de sonido estará enfocado a todos los públicos, por lo que busca ser agradable y simpático.

Como resultado final de este trabajo, se espera publicar en la plataforma web el juego “La Tienda de Goblina” y que este pueda ser jugado desde cualquier escritorio.

1.3. Objetivos generales

1.3.1. Objetivos principales

Objetivos de la aplicación/producto/servicio:

- El producto final ha de ofrecer una experiencia completa del *gameplay loop* propuesto.
    - Han de estar presentes todos los elementos mostrados en el apartado 3.2 Gameplay Loop.
- El producto ha de cumplir con las especificaciones técnicas requeridas por la plataforma en la que será publicado (Itch.io HTML5 game)
    - Página de requisitos (Itch IO).

Objetivos para el cliente/usuario:

- Los sistemas de juego han de ser intuitivos.
    - El juego no recurrirá a tutoriales. Las user test han de demostrar que es autoexplicativo. Se busca incentivar un proceso de prueba y error para el sistema basado en acertijos.
- Conservar el progreso en el juego entre sesiones.
    - El juego ha de proporcionar un sistema de carga y guardado de datos que permita al jugador reanudar el juego donde lo dejo. A su vez, el jugador puede emplear los conocimientos adquiridos en sesiones anteriores para seguir progresando (deducción basada en los acertijos resueltos).

Objetivos personales del autor del TF:

- Proponer un diseño y proyecto que se ajusta a las limitaciones de tiempo.
    - Verificado por el nivel de calidad de la propia entrega.
- Emplear el prototipado como medio para la toma de decisiones
    - Consultar 4.2.2 Evolución del diseño.

1.3.2. Objetivos secundarios

Objetivos adicionales que enriquecen el TF.

- Demostrar un uso adecuado de las herramientas y artefactos creados para gestionar el proyecto.
- Estructura del código de forma limpia y estandarizada.

1.4. Metodología y proceso de trabajo

1.4.1. Consideración previa

Me gustaría abrir esta sección con una distinción a tener presente durante el resto del proyecto; El TF es un proyecto académico que encapsula el proyecto de desarrollo de un videojuego. No es un hecho que deba afectar a la calidad del proyecto encapsulado (el videojuego), pero creo que influye sobre el proceso de trabajo y la metodología.

El proceso de trabajo se verá afectado, porque antes que el desarrollo del videojuego viene las necesidades que exija la gestión del TF. Es una demostración de conocimientos adquiridos, por lo que se espera documentar todas las decisiones y su justificación. También incluirá elementos que no tendrían por qué estar presentes en el desarrollo de un videojuego, como análisis de los sistemas de progresión, de los sistemas interactivos, … Finalmente, el proyecto de desarrollo del videojuego no tiene razón de ser o validez sin la consecución del TF.

1.4.2. Distribución del trabajo

Fase de Preproducción

Cubierta en PEC1 y PEC2

- Propuesta
- Análisis de mercado
- Planificación

Producción

Inicia en PEC2 hasta PEC4

Prototipo

Versión cruda. Comprueba experiencia de usuario, *gameplay*, mecánicas, …

Versión jugable

Sustituye *assets* previos por unos de mayor calidad. Se pule la experiencia de juego, la deuda técnica, … Se cubre casi todas las funcionalidades.

Vertical slice

Versión completa y jugable del juego. Se espera un aspecto y una calidad de producto final. Como jugar al juego final pero una versión reducida a un máximo de media hora. Suele emplearse para captar la atención de inversores y convencer a interesados.

El desarrollo de un videojuego continua, con más fases de producción y todo un mundo de postproducción (Stefyn, Nadie, 2022) (Alex Tsekhansky, 2023). Sin embargo, el alcance de este TF se limita a obtener un *vertical slice*.

1.4.3. Proceso de trabajo

Para el TF haré uso de los materiales de la asignatura GP[1] y seguiré muy de cerca las indicaciones de la guía PMBOK[2]. Se espera así:

- Establecer todas las actividades para alcanzar los objetivos.
- Definir los tiempos y recursos necesarios para completar el proyecto.
- Establecer los mecanismos de seguimiento y control para que el proyecto se cumpla dentro del marco de tiempo establecido.
- Contextualizar los progresos en base al panorama.
- Asumir el nivel de compromiso necesario.

Las tareas del TF son divididas en 2 tipos.

- Relacionadas con la Memoria
- Relacionadas con el Videojuego

Esta distinción puede apreciarse en el cronograma (1.5).

El desarrollo del videojuego se ejecutará en paralelo con el desarrollo de la memoria. La mayoría del trabajo relacionado con la memoria se centra en los periodos previos a una entrega y tras obtener retroalimentación de una entrega previa.

Para el desarrollo del videojuego valoro la flexibilidad de plazos, emplear los esfuerzos en tareas concretas, aprendizaje continuo y entrega incremental. Tomaré métodos y prácticas de la filosofía ágil, pero no me ceñiré a ella.

Lo más importante para el proceso propuesto será obtener un backlog bien definido.

- Saber que incluirá el juego y que no incluirá
- De donde se obtendrán los recursos y cuales se producirán
- Cuál es el esfuerzo estimado de cada tarea

A partir del backlog aplicaré parte del *framework Kanban*. Emplearé una *Kanban board* que extraerá sus tareas del backlog y el plan del TF. He decidido emplear Kanban porque

- Es un modelo de flujo continuo en vez de *sprints* con una duración determinada.
- Permite visualizar el trabajo y entender el progreso actual.
- Limita la cantidad de *work in progress*.

[1] Gestión de Proyectos
[2] Guía de los Fundamentos para la Dirección de Proyectos

1.5. Planificación

La planificación del proyecto se ha realizado en base a la guía PMBOK. En esta sección se cubre la lista de entregables, alcance, roles y duración del proyecto. Los aspectos que no son cubiertos en esta sección han sido cubiertos en otras secciones que pertenecen al apartado 1.

1.5.1. Dimensiones del proyecto

Lista de entregables

- Pruebas de evaluación continua
    - Cuatro pruebas repartidas en el cronograma.
- Ejecutable del videojuego.
- Repositorio del videojuego.
- Tráiler del videojuego.
- Presentación de defensa.

Alcance

El videojuego ha de incluir:

- Sistema Merge (gameplay interactivo)
- Sistema Negociar (gameplay interactivo)
- Sistema Inventario (permanencia)
- Sistema Libro Comerciante (progresión)[3]

Mi idea es que jugar y probar todos los sistemas sea más que asequible en 10 minutos. Completar el juego en 30 minutos es una cuestión de sistemas de progresión, contenido y balance.

La memoria, además de los apartados ya estipulados, incluirá el análisis de los sistemas del juego.

Roles

Estudiante: Desarrollador y responsable del proyecto. Debe actuar siguiendo las indicaciones del tutor. Trabaja de forma individual.
Tutor TF: Actúa como consultor a lo largo de las entregas de la evaluación continua, y como evaluador al final de esta.

[3] También llamado “colección”

Cronograma

Planificación inicial

Desarrollo TF

Name	Date	Fase	Parent item	Sub-item	Tipo
Propuesta	@March 1, 2023 → March 5, 2023	PEC1			Videojuego
Apartado 1	@March 6, 2023 → March 10, 2023	PEC1			Memoria
⚠️ENTREGA PEC1	@March 11, 2023 → March 12, 2023				ENTREGA
Prototipo	@March 13, 2023 → April 16, 2023	PEC2			Videojuego
Apartado 1 : Revisión	@March 27, 2023 → March 30, 2023	PEC2			Memoria
Apartado 2	@April 3, 2023 → April 9, 2023	PEC2			Memoria
Apartado 3	@April 10, 2023 → April 14, 2023	PEC2			Memoria
Video	@April 14, 2023	PEC2			ENTREGA
⚠️ENTREGA PEC2	@April 15, 2023 → April 17, 2023				ENTREGA
Apartado 1, 2 y 3 : Revisión	@April 18, 2023 → April 23, 2023	PEC3			Memoria
Versión jugable	@April 17, 2023 → May 21, 2023	PEC3			Videojuego
Apartado 4	@May 8, 2023 → May 14, 2023	PEC3			Memoria
⚠️ENTREGA PEC3	@May 20, 2023 → May 21, 2023				ENTREGA
Apartado 4 : Revisión		PEC4			Memoria
Vertical Slice	@May 22, 2023 → June 7, 2023	PEC4			Videojuego
Apartado 5	@June 8, 2023 → June 10, 2023	PEC4			Memoria
Apartado 6	@June 9, 2023 → June 10, 2023	PEC4			Memoria
Apartado 7	@June 11, 2023	PEC4			Memoria
Versión final memoria	@June 12, 2023 → June 13, 2023	PEC4			Memoria
Defensa del proyecto	@June 14, 2023 → June 18, 2023	PEC4		Trailer, Presentación	ENTREGA
⚠️ENTREGA PEC4	@June 17, 2023 → June 18, 2023				ENTREGA
Presentación	@June 14, 2023 → June 18, 2023		Defensa del proyecto		
Trailer	@June 17, 2023 → June 18, 2023		Defensa del proyecto		

Ejecución

Desarrollo TF

Name	Date	Fase	Parent item	Sub-item	Tipo
Propuesta	@March 1, 2023 → March 5, 2023	PEC1			Videojuego
Apartado 1	@March 6, 2023 → March 10, 2023	PEC1			Memoria
⚠️ENTREGA PEC1	@March 11, 2023 → March 12, 2023				ENTREGA
Prototipo	@March 18, 2023 → April 26, 2023	PEC2			Videojuego
Apartado 1 : Revisión	@April 3, 2023 → April 6, 2023	PEC2			Memoria
Apartado 2	@April 3, 2023 → April 11, 2023	PEC2			Memoria
Apartado 3	@April 12, 2023 → April 14, 2023	PEC2			Memoria
Video	@April 14, 2023	PEC2			ENTREGA
⚠️ENTREGA PEC2	@April 15, 2023 → April 17, 2023				ENTREGA
Apartado 1, 2 y 3 : Revisión	@April 27, 2023 → April 29, 2023	PEC3			Memoria
Versión jugable	@April 30, 2023 → May 25, 2023	PEC3			Videojuego
Apartado 4	@May 21, 2023 → May 24, 2023	PEC3			Memoria
⚠️ENTREGA PEC3	@May 20, 2023 → May 21, 2023				ENTREGA
Apartado 4 : Revisión	@June 8, 2023 → June 13, 2023	PEC4			Memoria
Vertical Slice	@May 26, 2023 → June 13, 2023	PEC4		Fase recolección, Fase tienda	Videojuego
Fase recolección	@May 26, 2023 → June 1, 2023		Vertical Slice		
Fase tienda	@June 2, 2023 → June 9, 2023		Vertical Slice		
Apartado 5	@June 13, 2023 → June 15, 2023	PEC4			Memoria
Apartado 6	@June 14, 2023 → June 15, 2023	PEC4			Memoria
Apartado 7	@June 14, 2023 → June 15, 2023	PEC4			Memoria
Versión final memoria	@June 14, 2023 → June 15, 2023	PEC4			Memoria
Defensa del proyecto	@June 16, 2023 → June 18, 2023	PEC4		Trailer, Presentación	ENTREGA
⚠️ENTREGA PEC4	@June 17, 2023 → June 18, 2023				ENTREGA
Presentación	@June 14, 2023 → June 18, 2023		Defensa del proyecto		
Trailer	@June 17, 2023 → June 18, 2023		Defensa del proyecto		

1.5.2. Seguimiento y control

Mediante la aplicación web **Notion** haré todo el seguimiento del proyecto.

Emplearé un diario de proyecto que indicará a que tarea destino mi tiempo cada día (independientemente si es exitosa).
Haré uso de una Kanba board para el desarrollo del videojuego.
Conservaré un diario de trabajo para anotar decisiones (y luego pasarlas a memoria), anotar ideas, …
Mantendré una sección de cambios e incidencias para así documentar de forma adecuada la memoria.
Mantendré una sección de recursos para facilitar el trabajo bibliográfico.
Gestionaré la duración de las tareas en el cronograma 1.5.1.

1.6. Presupuesto y herramientas

Presupuesto

Para el presupuesto se tendrán en cuenta las licencias adquiridas con anterioridad al proyecto, pero que han sido empleadas en este. También se hará un seguimiento de las horas invertidas en el desarrollo de código. En cuanto a los *assets* para el videojuego final, estos han sido adquiridos en las fases finales, siempre en favor de recursos libres de uso. Creo que es recomendable la adquisición de productos acabados porque encargar la producción de nuevos productos podría comprometer el desarrollo del proyecto.

Tiempo invertido

200 horas en desarrollo técnico
16 horas producción de audio
40 horas producción de arte
No incluye el tiempo invertido en redacción de memoria, estudio del engine, revisión de conceptos, diseño del juego…

Costes de licencias

Notion : Licencia de pago anual a 8$
Microsoft Office : Licencia incluida en los estudios en la UOC (precio pago anual para uso personal de 70$)
Procreate : Licencia de pago único a 16€ en la Appstore española
StableDiffusion : Uso de créditos gratuitos.
Lucid Chart : Versión de prueba
Apple Logic Pro: Licencia de pago único a 200$

Costes de hardware

Personal Computer con licencia Windows 10, 16 GB RAM, Intel Core i5 9600, NVIDIA GTX 1060 : 1260€
iPad Pro11 : 1100€
Apple Pencil : 150€

Herramientas

Gestión de proyecto, producción de documentos, …

- Notion
    - Software de gestión de proyectos online.
- Microsoft Office
    - Paquete ofimático capaz de cubrir todas las necesidades de documentación del proyecto.

- Google Drive
    - Soporte de copias de seguridad y compartido de archivos.
- Google Keep
    - Parte de la aplicación web Google Docs para el apunte de notas y seguimiento de tareas.
- Lucid Chart
    - Elaboración de diagramas.

Creación y gestión de assets

- Procreate
    - Editor de gráficos para arte digital. Lo utilizo para uso personal y profesional.
- Inkscape
    - Código Libre
    - Editor de gráficos de vectores
- DragonBones
    - Software libre
    - Animación basada en esqueleto.
- StableDiffusion
    - Generación de arte mediante IA.
- Apple Logic Pro
    - Estación digital de producción de audio.

Hardware

- iPad Pro11 + ApplePen
- PC personal

1.7. Estructura del resto del documento

En la sección 2. Estado del arte, analizaré la industria de desarrollo de videojuegos independientes limitando el alcance a proyectos que contaban con un solo integrante o un equipo muy reducido. De este análisis extraigo conclusiones y expongo consejos que expertos en el sector han compartido. Finalmente cito las influencias de este proyecto, ya sea por las herramientas que han empleado o por su diseño del videojuego final.

En la sección 3. Propuesta, expongo el TF que va a ser realizado. Los rasgos generales del videojuego que lo definen y lo hacen particular. De esta sección destaca el uso de lemas de desarrollo, una práctica muy recomendable para asegurar que los subobjetivos establecidos durante el desarrollo sean coherentes unos con los otros.

En la sección 4. Diseño, entro en todo lo relacionado con la implementación del producto. De esta forma, cubrir todos los elementos de diseño; diseño de los sistemas, diseño del juego, diseño de los gráficos y recursos empleados.

En la sección 5. Conclusiones y líneas de futuro, la sección final, expongo mi valoración sobre el trabajo realizado y los resultados obtenidos en este proyecto. En especial, un ejercicio autocrítico sobre qué se ha hecho bien y qué ha salido mal, así como la extracción de posibles líneas de futuro para el producto.

2. Estado del arte

Esta sección pretende ser un análisis del ecosistema en el que se desarrolla el proyecto. En ella espero:

- Formar la base bibliográfica del proyecto.
- Reconocer el estado actual del área.
- Aprender de los éxitos y fracasos previos.
- Crear una referencia para la toma de decisiones informadas.
- Conocer los problemas habituales en el sector.

2.1. Industria indie dev

En esta sección se da énfasis al desarrollo de proyectos de videojuego conocidos como *solodev* (un solo desarrollador). Dado el carácter multidisciplinar de la producción de videojuegos, no es fácil encontrar videojuegos en los que algunas de sus fases no se hayan visto involucradas más personas, por eso algunas de las referencias a continuación han contado desde el principio con un equipo de dos personas o han escalado a medida que los recursos en su disposición incrementaban. Defiendo que pese a ese factor, siguen siendo referencias válidas a este proyecto porque siguieron un proceso de desarrollo *indie* (enfrentaron las mismas problemáticas de un proyecto *solodev*) y porque es muy habitual (y a menudo inevitable) delegar ciertas partes del proyecto en terceros como podría ser la producción de arte, de sonido, marketing, …

Primero introduciré una serie de títulos que son grandes éxitos de la industria y han seguido un desarrollo indie. Junto a ellos, recursos que permiten conocer cómo se han llevado a cabo, conclusiones por los autores sobre el desarrollo, problemas que enfrentaron, … Mi objetivo es así mostrar:

- Qué el perfil de desarrollador indie es uno indefinido, con enorme variedad de antecedentes y competencias distintas.
- Que los tiempos y filosofías de desarrollo pueden variar completamente en función del individuo, y no solo del proyecto.
- La dificultad en llevar a cabo un proyecto de estas magnitudes, por la necesidad de capacidades de gestión, producción artística, producción técnica y largas jornadas de trabajo. No hay un camino fácil.
- Qué todos ellos nacen por un genuino interés en la creación. Si se aspira a la estabilidad económica o el beneficio, indie dev no es el camino.

Seguiré con una recopilación de fuentes que permiten conocer las mejores prácticas en la industria y analizar las dificultades que los proyectos previos superaron, así como una compilación de las que considero que vale la pena destacar.

Finalmente, puesto que este proyecto está basado en Godot, considero beneficioso destacar dos de sus antecesores más recientes los cuales he jugado y disfrutado, sobre los cuales es posible extraer conclusiones dada su trayectoria.

Spelunky

Ilustración 1: Portada Spelunky

Spelunky es un juego de plataformas 2D creado por el desarrollador Derek Yu y publicado en 2008.

Gran parte de su éxito lo debe, además de a su diseño y calidad, a ser de los primeros juegos en combinar elementos *roguelike* con acción en tiempo real de plataformas, creando así una fórmula en la que el jugador se enfrenta en cada iteración a una nueva combinación de problemas conocidos, para los que es necesario elaborar nuevas estrategias.
(Noclip - Video Game Documentaries, 2017) y (GDC, 2017)

Tal ha sido su marca en la industria que en los años venideros muchos desarrolladores tomaron influencia de esta combinación de géneros y llevaría al lanzamiento de grandes títulos como The Binding of Isaac, FTL: Faster Than Light, Rogue Legacy, … La naturaleza procesal de los juegos *roguelike* permite que pequeños equipos de desarrolladores indie produzcan una experiencia de juego de larga duración (e incluso compleja) sin la necesidad de invertir los muchos recursos que requiere elaborar un mundo completo y detallado. Además, gracias a la enorme creatividad de los muchos desarrolladores, propuestas que comparten la etiqueta *roguelike* ofrecen experiencias completamente distintas y variadas.

Spelunky fue desarrollado empleando Game Maker, hecho que ha dado gran popularidad al engine. Su diseño fue fruto de un fuerte compromiso por el prototipado, en el que Derek Yu trata los prototipos

como bocetos para combinar y probar ideas, de esta forma, descubrir qué funciona y qué no. Otro consejo en el que hace especial énfasis su creador es en la publicación de ideas y versiones de prueba, exponer a la opinión del público el videojuego incluso en sus versiones más tempranas.

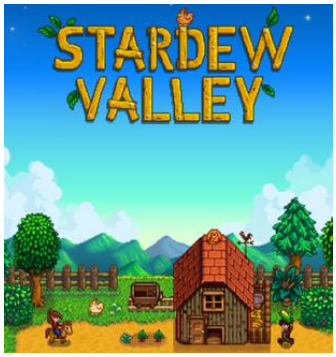
Tras su lanzamiento, Spelunky ha sido trasladado a las consolas modernas de la época por lo que cuenta con versiones en XBOX360 y PS3. Eso también demuestra que el proyecto no termina tras el lanzamiento, en el mundo de los videojuegos actual es muy importante cuidar el mantenimiento del producto final, y en este caso, estar atento a posibilidades para ampliar su mercado.

Derek Yu ha publicado un libro que compila sus ideas con respecto diseño de juegos, análisis de desarrollo y sus influencias. (Yu, Spelunky by Derek Yu, 2016)

Otro recurso de especial interés es su blog en el que detalla consejos y reflexiones sobre la industria, además de recursos de aprendizaje (por ejemplo un tutorial en la producción de pixel art). (Yu, Tumblr)

Stardew Valley

Ilustración 2: Portada Stardew Valley



Stardew Valley es un juego de simulación en el que el jugador administra su propia granja, y en el proceso, se sumerge en un mundo con vida propia. Fue publicado en 2016 y la historia de cómo su creador, Eric Barone, consiguió producirlo es una demostración de talento y perseverancia. (Singal, 2016), (Lin, 2016), (White, 2018) y (Baker, 2016)

Sin experiencia previa en el desarrollo de videojuegos, Eric Barone se propuso crear una réplica de la primera entrega en la saga Harvest Moon, de la cual él era un fan desde pequeño, para así mejorar sus habilidades (en la creación de videojuegos) y ofrecer un pequeño tributo para el resto de los fans. A medida que el desarrollo avanzó, empezó a ver mayores posibilidades con respecto qué podía ser su juego. Esta convicción persistió hasta que finalmente decidió comprometerse a tiempo completo al desarrollo del proyecto, dedicando así una media de diez horas al día los siete días de la semana, en lo que finalmente fue un desarrollo de 4 años.

En este desarrollo destaca su meticulosidad y capacidad autodidacta. Ha producido por sí solo todo el código, arte, historia y música del videojuego. Exceptuando su experiencia previa en composición musical, aprendió el resto de las disciplinas de forma autodidacta. El desarrollo fue cambiando de plataforma de desarrollo a medida que sus habilidades mejoraban, finalmente programando en C# mediante el *framework* Microsoft XNA.

No fue un desarrollo fácil. Además de las largas jornadas de trabajo, se encontró en tediosos ciclos en los que tuvo que solucionar todo tipo de problemas que nunca había enfrentado y sin ninguna ayuda externa. De hecho, meses antes del lanzamiento, estuvo a punto de abandonar el proyecto. Uno de los factores, y posible error de desarrollo, fue que todo el *playtest* era realizado por él mismo y las impresiones que consultaba a su novia. Llegó un momento que no disfrutaba del juego y se veía incapaz en detectar qué era divertido y qué no. Esta carencia de feedback fue suplida tras ponerse en contacto con 3 *streamers* de Twitch por medio de su publicadora Chucklefish. Esto le permitió obtener mucha más información tanto de sus impresiones como ampliar el registro de errores, y a su vez, recuperar la confianza qué estaba perdiendo.

Incluso tras el lanzamiento del juego, continúo trabajando; en un inicio solventando los problemas que los jugadores podrían encontrar y posteriormente añadiendo nuevo contenido. Si bien ahora ha encontrado tiempo para descansar, continúa mejorando Stardew Valley y en la actualidad es el mismo quién gestiona la publicación de su juego. Es posible seguir su trabajo en https://www.stardewvalley.net/blog/ .

Return of the Obra Dinn

Ilustración 3: Portada Return of the Obra Dinn

Return of the Obra Dinn es un juego de rompecabezas y misterio en primera persona publicado en 2018. Ha sido desarrollado por Lucas Pope, también autor del aclamado Papers Please y con mucha experiencia en el sector; ha trabajado para Realtime Associates en 2003 y para Naughty Dog en 2007. Además, su primer lanzamiento (en el qué colaboro) data del año 1999, Gearhead Garage.

He escogido este título y este desarrollador porque aporta contraste a los anteriores. Si bien los anteriores son muy bueno desarrolladores, Lucas Pope cuenta con mucha experiencia en el sector trabajando para grandes empresas. Dedicó su tiempo en Naughty Dog a la construcción de herramientas de desarrollo, y esos conocimientos también los aplica a los juegos que desarrolla, consiguiendo así la estética que hace único a este título. El juego ha sido desarrollado empleando Unity, y en su blog de desarrollo entra en detalle sobre las técnicas de *rendering* qué ha empleado para conseguir el aspecto final. También lo hace sobre el diseño de sonido, modelado de personajes, modelado del barco, … Considero que el blog de por sí es una fuente muy valiosa de conocimientos de desarrollo. (Pope, 2014)

Al igual que en el resto de los casos, tampoco fue un desarrollo fácil, le llevó 4 años y medio. El mismo explica como el hecho entre elegir un barco de 4 cubiertas en vez de 3 significó mucho más trabajo. También conectar las historias de los 60 tripulantes del barco, a los que el jugador tendrá qué identificar durante la partida. O en ocasiones, como en escenas de violencia, el efecto visual 1-bit que hace distintivo al juego desaparecía. (Machkovech, 2019)

Sin embargo, creo necesario destacar las dificultades que enfrento con el trabajo de localización. Con la globalización, todos los videojuegos aspiran a poder ser jugados por cualquier individuo de cualquier país, pero eso significa sortear las barreras lingüísticas, jurídicas y culturales (e incluso afecta al software, música, arte, …). Para Return of the Obra Dinn, en el que el jugador ha de describir en palabras la muerte de los tripulantes, el idioma no solo guía la historia sino que es parte de la mecánica principal del juego. Esto demuestra los muchos aspectos que ha de considerar un desarrollador, y si bien es habitual dejar el trabajo de localización para las fases finales del proyecto, el propio Lucas Pope saca como lección de las problemáticas encontradas haber involucrado antes en el proyecto a los encargados de localización. (Ars Technica, 2019)

Return of the Obra Dinn ha sido aclamado por la crítica tras su lanzamiento, y en el presente, Lucas Pope se encuentra desarrollando su nuevo proyecto Mars After Midnight. Siguiendo su filosofía de trabajo, continúa compartiendo en su respectivo diario de desarrollo las herramientas que produce para llevarlo a cabo.

Muchos otros

La lista de juegos que han sido producidos por un solo desarrollador, o así fue como se desarrolló gran parte del proyecto, es una lista enorme en la que continúan apareciendo nuevas entradas. A su vez, el sector experimenta la llegada de nuevas herramientas y la mejora de las presentes, cada vez más avanzadas (u otras simplificadas, haciendo el desarrollo más accesible). La industria cuenta con un montón de historias dignas de análisis como las que hay detrás de Minecraft, Kenshi, Undertale, Dwarf Fortress, …

Cómo una de las finalidades de esta sección del TF es aprender de los éxitos y fracasos previos, creo que eso es posible no solo analizando esas historias, sino también haciendo uso del conocimiento compartido que ha desarrollado la comunidad en un proceso de prueba y error. Me gustaría destacar portales como Game Developer, LOSTGARDEN, DEV Community (portal para desarrolladores de todo tipo de disciplinas), … La saga de libros Boss Fight Books, el foro de desarrolladores TIGSource, presentaciones realizadas en al Game Developers Conferences, … De esta última me parece de especial interés la saga Postmortems.
(https://www.gamedeveloper.com/about-game-developer, s.f.), (https://lostgarden.home.blog/about/, s.f.), (https://dev.to/about, s.f.), (Boss Fight Books, s.f.), (TIGForums, s.f.), (https://www.gdcvault.com/help.php, s.f.) y (GDC, s.f.)

Consejos del sector

Referencias: (Yu, Tumblr), (https://lostgarden.home.blog/about/, s.f.) y (https://www.gamedeveloper.com/about-game-developer, s.f.)

Al igual que en los medios creativos, la producción y puesta práctica de ideas está por encima de la abstracción y esperar a la idea perfecta. Es muy recomendable encontrar herramientas que permitan un prototipado rápido y dominarlas. Tratar el prototipado como el medio para valorar la viabilidad de las ideas.

Comunidad de desarrolladores y jugadores muy extensa. Mucha gente dispuesta a colaborar y probar nuevos juegos. Pedir ayuda y ofrecerla cuando es posible permite mantenerse activo en la comunidad, de esta forma crear contactos y puntos de apoyo. Este aspecto tiene gran relevancia en todo lo relacionado con *playtesting*. Por defecto, la experiencia del desarrollador no es indicativa del estado del juego puesto que existe unos conocimientos previos sobre los sistemas del juego y su propósito.

Emplea el anonimato o falta de exposición a tu favor. A la hora de publicar y compartir ideas, no hay que cumplir con los estándares de calidad que se exigen a las grandes desarrolladores. Las ideas tienen un valor muy limitado si no se llevan a la práctica, el ocultismo perjudica más que beneficia. Evitar la gran revelación. Es muy importante poder obtener feedback externo con frecuencia y así iterar. El propio desarrollador no puede hacer una valoración objetiva sobre la experiencia de juego dados sus conocimientos previos (conoce los sistemas y su propósito).

Distinguir cuando es necesario desarrollar tus propias herramientas y cuando existen soluciones preexistentes. Qué beneficiará en mayor medida al proyecto. Por ejemplo, gráficos de alta calidad requieren de altos requisitos técnicos, por lo que suelen usarse herramientas ya desarrolladas.

Mecánicas o técnicas realmente innovadoras suelen exigir la creación de herramientas por el coste de tiempo y recursos que puede significar adaptar las soluciones preexistentes al modelo deseado.

Existe una fuerte unanimidad en el uso de fechas límites. Es algo que también aparece en la producción de películas, música, … Aseguran que habrá un entregable final y exigen así identificar las limitaciones del proyecto. En la industria indie dev, se recomienda aprovechar fechas de eventos, competiciones y premios para así asumir el compromiso necesario.

Gran énfasis en finalizar los proyectos. Evita volver a empezar, porque si es así, lo más probables es que las mismas dificultades aparezcan en el siguiente proyecto. Si es inevitable, muy importante analizar que hizo que el último proyecto fracasará y plantear el siguiente proyecto con escala reducida. De hecho, en la industria la habilidad para terminar los proyectos es muy respetada y codiciada.

Desarrollar un videojuego comercial lleva tiempo independientemente de las habilidades personales. Involucra muchas tareas tediosas pero necesarias por lo que requiere perseverancia. Recomiendan saber delegar aquellos aspectos del proyecto siguiendo un balance de costes en el que tiempo y dinero son recursos muy valiosos. Esto no significa descuidar la calidad del producto, pero sí asegurar su consecución.

Trabajar en equipo no es fácil y puede llegar a crear más problemas que soluciones. Es una decisión que debe tomarse con mucha consideración y va más allá de las competencias o el carácter de los individuos. Muchos desarrolladores recomiendan emplear contratistas antes que empleados.

Volviendo sobre los puntos anteriores. La gestión de recursos es muy importante. Reconocer los puntos fuertes del proyecto e invertir en ellos. Encontrar un balance saludable a largo plazo, específicamente el bienestar personal, es una industria donde los proyectos son de larga duración y de nada sirve doblar el ritmo productivo si se sacrifica la estabilidad (un proyecto inestable es un proyecto cuya consecución es incierta).

Dados los recursos limitados, es importante estar dispuestos a reducir la escala. Quitar características y así respetar la planificación. Tomar riesgos calculados considerando las competencias y recursos disponibles. Es precisamente estas limitaciones las que hacen diferentes a unos juegos de otros.

El proyecto no termina necesariamente tras el lanzamiento. Es importante el mantenimiento del juego, gestión de la publicación, … De hecho, puede desencadenar en futuros proyectos como ofrecer nuevos sistemas o ampliar mercados desarrollando en distintas plataformas.

Godot

Brotato es un lanzamiento reciente (septiembre de 2022) inspirado en el éxito indie del año pasado Vampire Survivor. Ha sido desarrollador y distribuido por su creador bajo el seudónimo de Blobfish.

Es un videojuego que ha sabido aprovechar una tendencia en la industria, demostrando así no solo la importancia de una buena idea y su ejecución, sino planear una buena fecha de lanzamiento. De forma similar a la industria cinematográfica, en la que las películas compiten por aparecer en las carteleras, en la industria de los videojuegos hay una competencia muy feroz por ser la novedad y ocupar el tiempo de sus jugadores. Dado al carácter de los videojuegos, capaces de ofrecer horas y horas de entretenimiento, el consumidor se ve forzado a elegir entre el enorme catálogo que ofrecen las plataformas. A su vez, a pesar de la existencia de videojuegos que han conseguido triunfar tiempo después de su lanzamiento, el éxito de muchos videojuegos suele venir dado por su rendimiento las primeras semanas tras su publicación. La competencia entre títulos con un género o propuesta similar es directa; los jugadores tienden a elegir un favorito y ello significa que el resto de titulos similares se ven perjudicados.

La última tendencia en la aparición de juegos similares a Vampires Survivor no es nuevo en la industria, así ha pasado con el género *battle-royale, roguelike, MOBA*, y muchos otros tanto antes y después. Y si algo ha quedado demostrado, es que ningún desarrollador puede esperar ser el único que la aproveche. Seguir una tendencia no es garantía de éxito porque pronto aparecen muchos títulos en respuesta a ella.

Brotato ha apostado por un enfoque basado en las estadísticas, en cada partida se le ofrece al jugador amplias posibilidades para probar nuevas estrategias y se pone a su disposición más información sobre la evolución de su personaje (mejora de estadísticas). Otro punto fuerte de este título es que ha dado hincapié a una buena optimización y experiencia de juego. En este tipo de juegos de “escalado infinito”, a medida que la partida progresa es más fácil experimentar *lagspikes* porque el trabajo computacional incrementa con la aparición de mayores hordas de enemigos y mayores cálculos computacionales.

Un recurso que me ha parecido muy interesante en la siguiente entrada del desarrollador en su blog en la que comparte 68 reflexiones sobre el desarrollo indie. (Blobfish)

Endoparasitic es un juego de horror desarrollado por Narayan Walters, quién cuenta con un canal de Youtube bajo el usuario de Mizizizi en el que cubre su desarrollo además de ofrecer tutoriales muy prácticos para Godot.

Si bien no ha conseguido obtener las cifras de ventas que ha obtenido Brotato, ha obtenido muy buenas valoraciones en Steam y se estima que ha vendido más de 40mil copias. Su número de jugadores concurrentes es bajo, pero eso es debido a que no es un juego con carácter “rejugable”. Uno de sus lauréles es que ha sido jugado por el famoso Youtuber Markipiler, video que cuenta con más de 5 millones de reproducciones y significa una oportunidad de exposición al público valiosísima a la que muchos desarrolladores aspiran.

Canal creador en el que también sube tutoriales
https://www.youtube.com/@Mizizizi
Markipiler gameplay
https://www.youtube.com/watch?v=AtxYF4mXK7w
Estadísticas
https://steamdb.info/app/2124780/

2.2. Influencias

Ilustración 4: Pantalla de juego Overloot

A continuación juegos que han influenciado el diseño del videojuego para el TF.

Overloot

Overloot es un juego RPG puzle desarrollado por BKOM Studios, empresa francesa desarrolladora de juegos, aplicaciones, etc.

Su *gameplay loop* está basado en su sistema de combate en el que el jugador recoge equipamiento, el cual guarda en su inventario, y al combinarlo con equipamiento del mismo tipo y calidad, este mejora. A

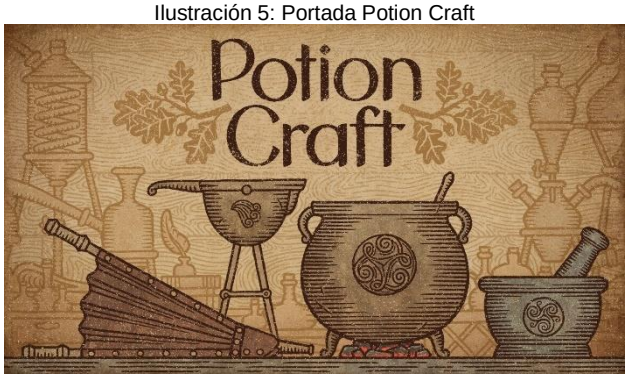
medida que el jugador progresa por los diferentes niveles las estadísticas del equipamiento recogido mejora, así como las estadísticas de los enemigos. Sigue un modelo free to play en el que el jugador puede pagar para progresar más rápido.

Actualmente el juego ya no se encuentra disponible en ninguna de sus dos plataformas de publicación, la tienda Google Play y la App Store.

El juego propuesto en este TF busca ampliar sobre este sistema. Pienso que es una mecánica de juego entretenida, pero es fácil que resulte repetitiva por si sola. Además, no se pretende obtener una réplica exacta. Se pretende extraer el concepto de aplicar la mecánica *merge* habitual en muchos juegos móvil junto a la administración de inventario RPG.

Potion Craft

Ilustración 5: Portada Potion Craft



Potion Craft es un juego de simulación en el que el jugador toma control de una tienda de alquimista medievo. Ha sido desarrollado por niceplay games y publicado por tinyBuild.

Es un juego relativamente simple pero muy satisfactorio de jugar. En él el jugador recolecta hierbas que planta en su jardín, crea pociones con ellas en un sistema donde las decisiones que tomes durante su creación alteran el producto final, y finalmente, vende las pociones a los aldeanos locales con la opción de negociar un precio diferente al establecido.

Su principal influencia sobre esta propuesta de TF es el control sobre toda la cadena de valor en la administración de una tienda. Cabe destacar, que a diferencia de este juego, mi propuesta no busca ser un simulador ni replicará el sistema de creación y venta de pociones. Sin embargo, creo que es una influencia digna de mención porque ambos presentan un *gameplay loop* muy similar.

3. Propuesta

Este proyecto no aspira al beneficio o éxito comercial, y a pesar de que implica la creación de un producto, no planeo comercializarlo. De ahí justifico el enfoque de la sección anterior y la presente desde un punto de vista académico y no comercial.

3.1. Ambientación

El juego se desarrolla en un mundo de fantasía no especificado. En el juego el jugador acompañará a nuestra protagonista Goblina. Goblina es el ojito derecho de su abuelo el Gran Gobulus Golagola VII, un reconocido magnate de Goblinópolis. Pero a pesar de esta preferencia, no se toma en serio a su nieta como verdadero contendiente a heredar la posición de matriarca de los Golagola. Para demostrar su valor, Goblina decide embarcarse en una aventura en la que planea amasar una fortuna y demostrar ser una gran comerciante. Sin embargo, Goblina desconoce el valor de su mercancía, por lo que tendrá que descubrirlo por medio de un proceso de prueba y error en sus negociaciones.

Al tratarse de un *vertical slice*, no habrá desarrollo de la historia. Sin embargo, la ambientación y caracterización del personajes permite ampliar el videojuego de forma cohesiva.

Ilustración 6: Concepto base Goblina

Al pensar en “goblins” gran parte del imaginario cae en criaturas desalmadas y feas (unos orcos en miniatura). Pero ese no tiene porque ser el caso. Este diseño similar a un ratoncillo y su enorme mochila acompaña mucho mejor el tono que quiero que tenga el juego.

Créditos: Ilustración creada por el usuario PrideAndEnvy de la plataforma Reddit para https://dungeonsandbargains.com.

3.2. Gameplay Loop

Loot phase

Tabla 1: Gameplay Loop 1

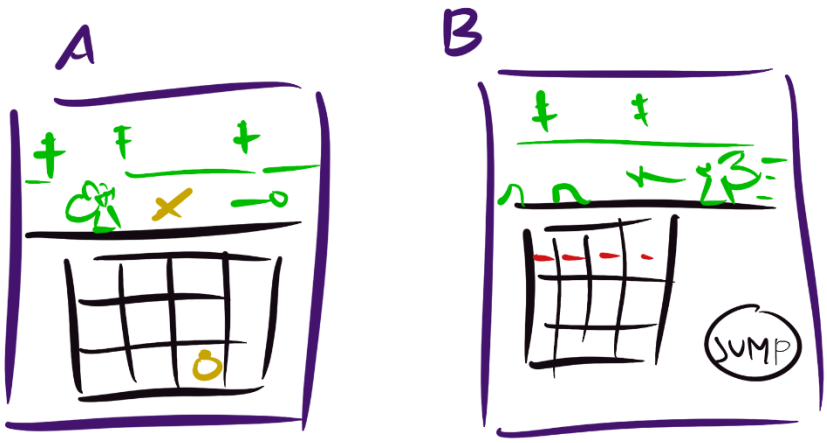


Ilustración 7: Overloot UI

A

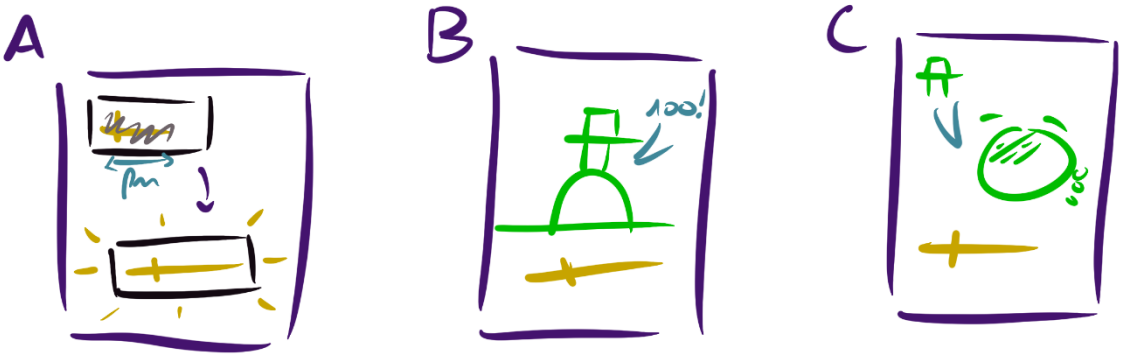
Merge: Minijuego de gestión de inventario. Goblina recoge artefactos y los añade al inventario. El jugador ha de crear el espacio suficiente en el inventario. También puede juntar aquellos artefactos del mismo tipo y calidad para formar uno mejor. Mayor la calidad, mejores probabilidades de obtener buen producto.

B

Goblina es descubierta y alguna amenaza le persigue. No hay combate en el juego por lo que nunca es atrapada. En un inicio se trataba de un minijuego tipo *runner*. Con el avance del proyecto, se ha decidido prescindir de esta fase.

Shop phase

Tabla 2: Gameplay Loop 2



A

Revelar: Los artefactos que ha recolectado en la *loot phase* son *blueprints*. A partir de ellos se obtiene los artefactos que conforman el libro.

B

Sistema de encanto: Es la base del juego, una especie de sistema de acertijos en la que el jugador empleará cartas de encanto para intentar adivinar cual es el encanto del artefacto elegido.

C

Tasar: Presenta una alternativa para cuando el jugador tiene un exceso de oro o se encuentra atascado en su progreso. Similar a los juegos RPG cuando el jugador evita el enfrentamiento con el jefe final para adquirir mejores habilidades y superarlo así con mayor facilidad.

El jugador obtiene el producto base en la **loot phase**.

En la **shop phase** lo revela (A) y lo vende (B), progresa en el juego al vender y también al invertir el oro ganado (C).

3.3. Objetivo del juego

Las medidas de progresión del juego son el oro y el libro de comercio. El objetivo del jugador es conseguir revelar todas las entradas del libro, hecho para el que entonces tendrá un flujo de ganancia de oro más que considerable.

Durante la **loot phase** el jugador recolecta artefactos sin revelar. En la **shop phase**, el jugador revela estos artefactos y comercia con ellos.

Cada artefacto tiene un encanto* oculto en el libro. Vender un artefacto haciendo uso de las cartas de encanto adecuadas desbloquea su entrada en el libro. El juego es completado una vez se ha desbloqueado el encanto de todos los artefactos.

**Encanto: Conjunto de cualidades que hacen que una cosa o una persona sea sumamente atractiva o agradable.*

La mayoría de las veces no se realizará la venta óptima, o se venderán artefactos cuyo encanto ya ha sido revelado, esta ganancia de oro se emplea para progresar.

De esta forma el jugador está incentivado a ganar dinero, al igual que Goblina, porque es una moneda de progreso aunque el objetivo final sea completar el libro.

3.4. Lemas

Premisas que guían el desarrollo del videojuego y facilitan la toma de decisiones para un objetivo común.

1.Las mecánicas que propone el videojuego son familiares y basadas en interacción de arrastre.

Los controles están basados en una interacción de seleccionar más arrastrar que sigue el mismo patrón que los controles táctiles en la mayoría de las aplicaciones de smartphone. Esto también genera la presencia de elementos visuales familiares que guían el comportamiento.

2.La ambientación del juego es ligera, apelando así al jugador casual (a todos los públicos).

Se apuesta por un diseño de dibujo colorido y dentro del género fantástico, capaz de apelar a diferentes demografías basadas en su edad. También se hace un fuerte uso de pictogramas para representar diferentes conceptos, tanto características de objetos o emociones de los personajes, de esta forma reducir las barreras idiomáticas.

3.Proporcionar una experiencia “*play from the start*”.

Mediante el resto de los puntos y el desarrollo de una interfaz de usuario simple, se busca reducir la necesidad de largas explicaciones para comprender los sistemas del juego. De esta forma, proporcionar un videojuego de rápido despliegue en el que es posible empezar a jugar desde el primer momento. A su vez, se reduce el miedo al fracaso minimizando las penalizaciones que el jugador sufre por fallar, para que este pueda experimentar y descubrir el funcionamiento de los sistemas por si solo.

4.Progreso de juego y experiencia tipo “juega sin compromiso”.

El videojuego busca ser consumido en intervalos irregulares, en los que el jugador juega en espacios de tiempo reducidos durante su día a día. Es posible disfrutar de todas las mecánicas del videojuego desde el principio, por lo que el jugador no se ve coaccionado a jugar para la obtención de un recurso limitado.

3.5. Publicación

Decisión de plataforma

Para esta versión del videojuego (vertical slice), se ha decidido publicar en plataforma de escritorio. Esta decisión está motivada por garantizar un despliegue lo más rápido posible (sin necesidad de descarga), y asegurar una experiencia de juego lo más cercana posible a tal y como se ha refinado la interacción durante las diferentes iteraciones de desarrollo.
Su objetivo inicial era la plataforma móvil. Este cambio de rumbo se debe a la adopción de una estrategia ejecutada por otros videojuegos en los que el proyecto es publicado en su versión de escritorio limitando durante el desarrollo los controles del juego y el uso de recursos de memoria y procesador. En este caso, siempre basar los controles en mecánicas de ratón y el desarrollo gráfico cómo si se tratara de un lanzamiento para smartphone. Dos ejemplos de gran presencia en la industria, ejecutados por empresas con amplios recursos, son Hearthstone (de Blizzard) y Teamfight Tactics (de Riot Games). Esta estrategia busca facilitar en futuras iteraciones el redespliegue en plataforma móvil.

También hay que destacar múltiples consideraciones de la plataforma web, alineadas con las especificaciones del desarrollo para smartphone. Estas son las cuestiones del tamaño de la aplicación, rendimiento y diseño adaptativo. En cuánto el tamaño de aplicación, se seguirá el tamaño máximo impuesto por la plataforma en la que se publicará (Itch IO). Este proyecto no cubre el análisis de medidas de rendimiento. Se intentará obtener un diseño adaptativo durante las fases finales.

Estrategia de publicación

No existen objetivos comerciales detrás de la publicación, por lo que no se cubren consideraciones de marketing, estrategias de venta, modelo de negocio, …

Los objetivos de este proyecto tienen una finalidad académica y profesional. Se espera que el producto final sea un recurso que pueda; sumar a mi currículo para el mundo laboral, compartir con otros desarrolladores interesados y permitir a otros estudiantes del grado de Ingeniería Informática probar con facilidad un producto final del TF en Videojuegos.

4. Diseño

4.1. Arquitectura de la aplicación y subsistemas

4.1.1. Arquitectura general

Godot como sistema central

Toda la aplicación se encuentra centralizada en un proyecto del *engine* Godot. Por lo tanto, la arquitectura del sistema y subsistemas estará basada en cómo Godot estructura los proyectos, escenas y scripts. La experiencia de juego es guiada mediante una estructura de escenas, dónde cada escena representa una pantalla del juego. El jugador navega de pantalla en pantalla e interactúa con las operaciones disponibles en cada pantalla. Sin embargo, una escena es mucho más que tan solo una pantalla de juego.

Escena

En Godot, una escena es una jerarquía en forma de árbol que agrupa distintos nodos. Las escenas se pueden instanciar, heredar y reutilizar; de forma similar a las clases en el paradigma OOP. Al ejecutar el programa, las escenas son instanciadas, y cómo objetos, contienen atributos que indican su estado y métodos que permiten su funcionalidad. Esta estructura permite aplicar conceptos OOP como el de encapsulación o el principio de responsabilidad única (Godot Engine). Por lo tanto, las escenas pueden ser utilizadas para crear pantallas de juego, jugadores, enemigos, balas, elementos GUI, …

Nodo

Los nodos son los componentes que forman el árbol de una escena, actúan como unidades funcionales y son su unidad atómica. Es importante entender que si bien se utiliza el término componente debido a que los nodos y su jerarquía componen la escena, la relación de un nodo con su padre (nodo en un nivel superior de la jerarquía) es una relación de agregación. Un nodo por si solo es una entidad independiente, e incluso, el mismo puede ser otra escena contenida (PackedScene).

Scripts

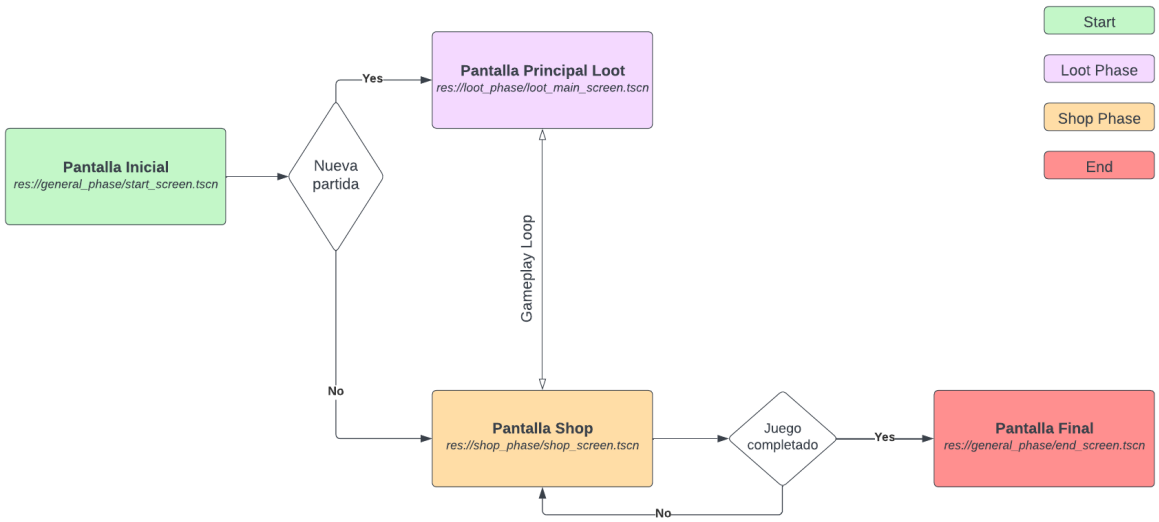
Los scripts son recursos que aprovechan las funcionalidades del *engine* para extender la funcionalidad de los nodos en los que son incorporados. Todos los scripts heredan de alguna clase base del engine. Esto permite que el script tenga acceso a sus métodos y atributos. El programador, mediante la definición de nuevos métodos, redefinición de los métodos virtuales y el uso de los atributos y métodos heredados, puede crear nueva funcionalidad.

Implicaciones en sección 4.1

Para una mejor comprensión del programa, en concreto de su estructura y lógica, los diagramas no reflejarán todos los nodos contenidos en una escena, a pesar de que pueden ser considerados objetos. Lo que si se hará es adjuntar una imagen con la jerarquía que forma la escena y una explicación de esta. Del mismo modo, se tratará como a una unidad funcional a los nodos junto a su script, o en otras ocasiones, se indicará cuando un nodo contiene clases internas. En los diagramas solo se incluirán los nodos relevantes para su funcionamiento.

Diagrama general del flujo de escenas

Tabla 3: Diagrama flujo de escenas



Cada pantalla es una escena que se encargará de coordinar sus nodos invocando los diferentes contextos de la aplicación. Las pantallas de carga y las transiciones de escena se ejecutarán al cambiar la pantalla actual.

Consultar apartado 4.2.1 para una mejor comprensión de la navegación.

Consultar apartado 4.1.2 para el funcionamiento específico de cada escena.

Sistema de archivos y estructura del proyecto

Tabla 4: Estructura carpetas *root*

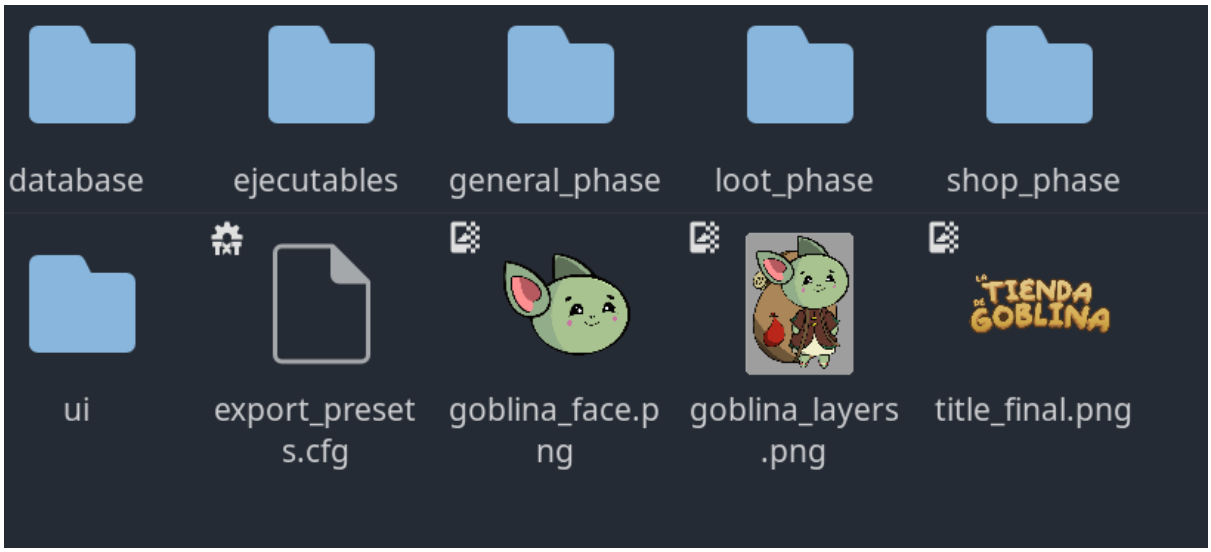
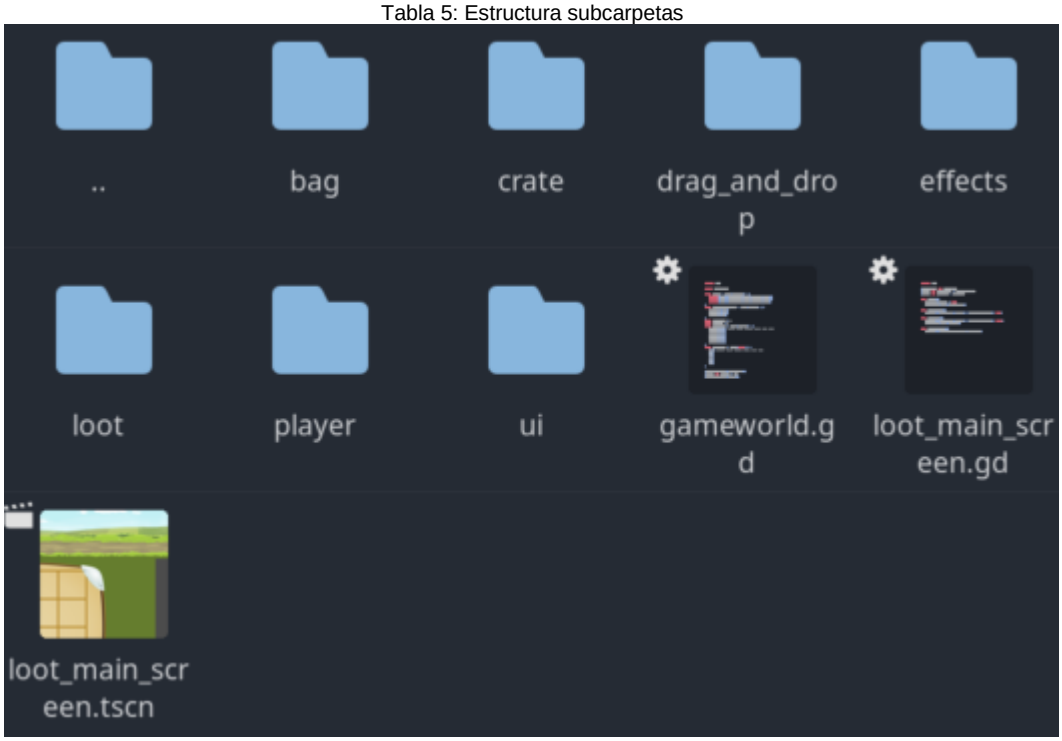


Tabla 5: Estructura subcarpetas



El nombramiento de archivos respeta la guía de estilo de Godot (Godot Engine). A la hora de modificar los archivos del proyecto (desplazamiento, renombrado, …), es muy importante utilizar el sistema de archivos incorporado en el editor de Godot para así evitar problemas de dependencia.

Está en manos del desarrollador emplear la estructura de archivos más conveniente para su programa. Existen diferentes preferencias, una habitual es en la estructura general destinar una carpeta a los recursos (*assets*), otra para scripts y otra para las escenas. Después en estas subcarpetas emplear la misma estructura para así establecer relaciones entre recursos, scripts y escenas.

Para este proyecto he considerado oportuno estructurar las carpetas en base a las fases del videojuego.

- Directorio principal -> elementos generales del juego
- Fase de recolección -> loot_phase
- Fase de venta -> shop_phase
- Pantallas de inicio, final y transición -> general_phase

En lo que respecta a las subcarpetas, el objetivo es agrupar las escenas a los recursos y scripts utilizados lo más cerca posible. En algunos casos un recurso o script puede ser reutilizado en diferentes escenas, en tal caso, es habitual destinar una carpeta que contiene su funcionalidad en el directorio principal.

Problemas en el cambio de escena

Si en ocasiones la imposición del *engine* Godot de un sistema de archivos propio es útil para el control de dependencias, esto también puede generar problemas con los programas externos de control de versión (VSC).

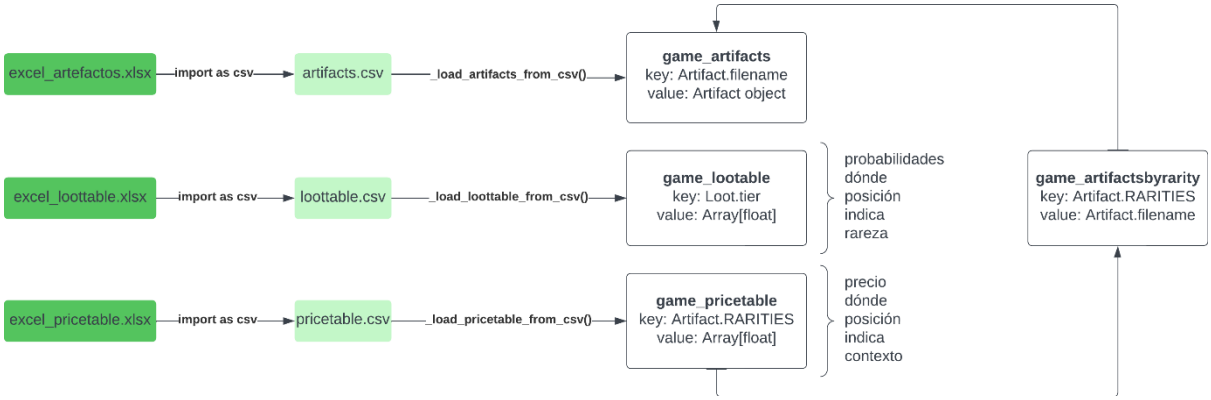
Así fue el caso de un problema cuyo origen fue bastante difícil de encontrar durante el desarrollo (porque el problema no estaba en la escena actual). Para el cambio de escena, se había empleado una variable tipo *export* (variables asignables y visibles desde el editor) encargada de gestionar uno de los próximos cambios de escena. Al ejecutar el proyecto, aparecía la pantalla de carga de Godot y poco después el programa se cerraba sin ningún mensaje de error. Para encontrar su origen fue necesario replicar el mismo estado de la aplicación, ejecutar el engine junto a su consola, e informarme sobre como Godot gestiona el control de dependencias.

En Godot los recursos no solo pueden ser referenciados por su ruta de archivo, sino que Godot también emplea una clase *singleton* llamada ResourceUID encargada de asignar un UID para cada recurso en el proyecto. El UID es un identificador único que permite al engine conservar las referencias al recurso incluso si su ruta de archivo cambia (es renombrado o desplazado). Al operar con distintas versiones de control, es posible que el UID de un recurso cambie debido a las modificaciones del programa externo VSC (modifica algunos datos caché). Cuando el engine intenta iniciar la variable al ejecutar la aplicación, a pesar de que la ruta del archivo es la misma, no encuentra la UID proporcionada y detiene la ejecución (indicando el error en la consola externa).

Estructura BD

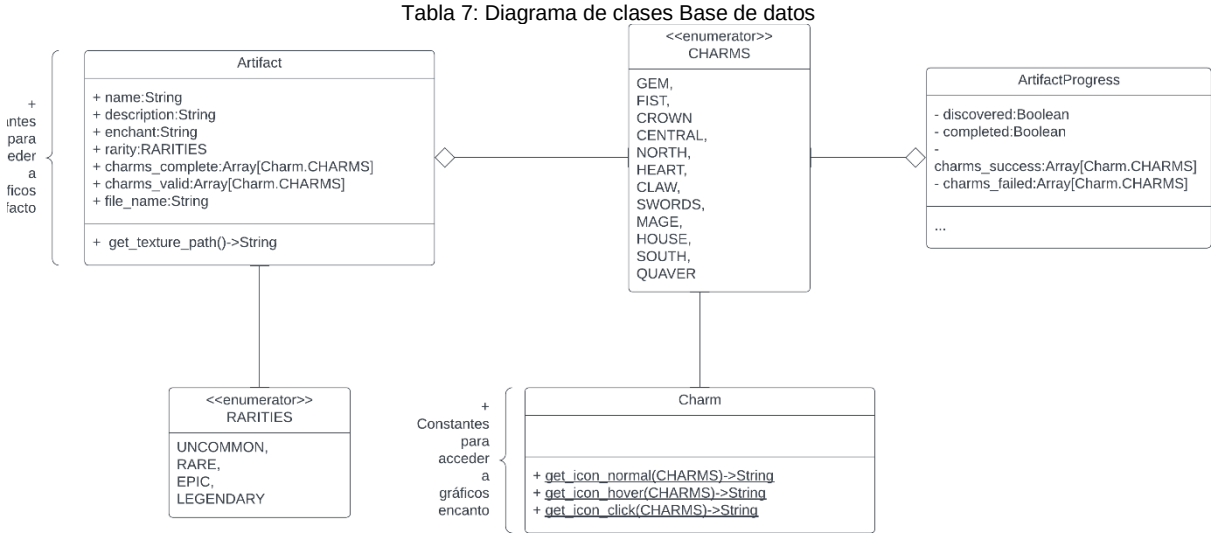
Para su implementación en Godot, he empleado el patrón Singleton donde un script con el nombre de clase Database es accesible desde cualquier otro punto del programa. Este script carga la información necesaria, prepara las estructuras y variables que contendrán el estado del juego y permiten acceder a la información mediante su interfaz.

Tabla 6: Diagrama de tablas



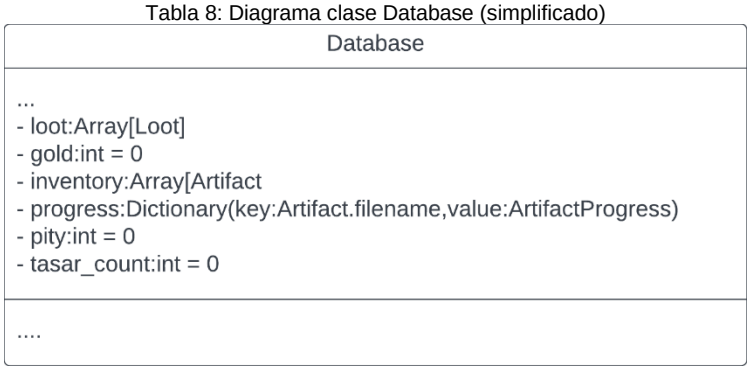
La base de datos está construida en código GDScript y se apoya de varios archivos de extensión CSV para la información relacionada con las características del artefacto (game_artifacts), las probabilidades de obtener un artefacto según su rareza (game_loottable) y una tabla de precios según su rareza (game_pricetable). Estas estructuras han sido generadas en diccionario por la eficiencia de esta estructura en Godot y, en general, en lenguajes de tipado débil (Godot Engine). Un cuarto diccionario (game_artifactsbyrarity) actúa de forma similar a como lo hacen las vistas en las bases de datos relacionales para apoyar las consultas basadas en la rareza del artefacto.

Tabla 7: Diagrama de clases Base de datos



La clase Artifact contiene toda la información necesaria sobre un artefacto. La clase Charm permite almacenar los encantos y acceder a los recursos de imagen. La clase ArtifactProgress da soporte a la base de datos para controlar el estado de la partida.

Tabla 8: Diagrama clase Database (simplificado)



Por lo tanto, la base de datos es una clase Database que contiene:

- Constantes para acceso a los recursos CSV.
- Constantes para control del progreso de la partida.
- Las tablas en Tabla 6: Diagrama de tablas.
- Variables con el estado de la partida en Tabla 8: Diagrama clase Database (simplificado).
- Operaciones de acceso y gestión.

Las constantes para el control del progreso de la partida gestionan elementos relacionados con el *gameplay* que son tratados en Arquitectura subsistemas(Progresión) junto a una explicación del uso de las tablas para calcular precios, determinar artefacto obtenido, …

Los métodos contenidos en Database son el mecanismo mediante el cuál:

- Se carga y almacena la información relacionada con el funcionamiento del juego.
  - Carga de tablas, preparación de diccionarios, …
- Se accede a las operaciones que permiten controlar el progreso de la partida.
  - Acceder a los artefactos en la BD, acceso al inventario del jugador, actualizar progreso de la partida, …

Lista de assets

Letrero título del juego

Ruta del archivo

res://title_final.png

Autor

Creado por mí

Herramientas utilizadas

Inkscape

Uso

Pantalla de título

Pack GUI para botones, paneles, indicadores de estado, …

Enlace

https://pzuh.itch.io/free-fantasy-game-gui

Autor

pzUH

Licencia

CC0

Uso

La mayoría de los elementos GUI. Modificado en base a las necesidades de cada pantalla.

Gráfico de ‘La Tienda de Goblina’ vista desde el exterior

Ruta del archivo

res://general_phase/tienda_from_outside.png

Autor

Creado por mí

Herramientas utilizadas

StableDiffusion + Procreate

Uso

Pantalla de título y transiciones (exterior<->interior)

Fondo parallax

Ruta del archivo

res://general_phase/bg/

Enlace

https://www.gamedevmarket.net/asset/hand-painted-parallax-background-1

Autor

oleekconder

Licencia

Gratuito

Uso

Pantallas en exteriores

Personaje 'Goblina'

Ruta del archivo

res://goblina_layers.png, res://goblina_face.png, res://loot_phase/player/

Autor

Creado por mí

Herramientas utilizadas

Procreate

Uso

Casi todas las pantallas de juego

Caja que recolecta el jugador

Ruta del archivo

res://loot_phase/crate/, res://shop_phase/crate/

Autor

Creado por mí

Herramientas utilizadas

StableDiffusion + Procreate

Uso

Pantalla de recolección y pantalla trastienda

Indicador de interacción

Ruta del archivo

res://loot_phase/ui/click_indicator/

Autor

Creado por mí

Herramientas utilizadas

Inkscape

Uso

Pantalla de recolección

Fondo del interior de la tienda

Ruta del archivo

res://shop_phase/bg_storage.jpg, front_l1.jpg, front_l2.png

Autor

Creado por mí

Herramientas utilizadas

StableDiffusion + Procreate

Uso

Pantallas de tienda (interiores)

Artefactos del juego

Ruta del archivo

res://shop_phase/artifacts/

Autor

Creado por mí

Herramientas utilizadas

StableDiffusion + Procreate

Uso

Pantallas de tienda

Iconos de encanto

Ruta del archivo

res://shop_phase/charms/

Enlace

https://game-icons.net/

Autor

Lorc, Skoll, Delapouite

Licencia

CC BY 3.0

Uso

Pantallas de tienda. Modificado patrón de colores y forma del exterior.

Personaje mercader

Ruta del archivo

res://shop_phase/merchant/

Enlace

https://kenney.nl/assets/toon-characters-1

Autor

Kenney

Licencia

CC0

Uso

Pantalla de venta. Modificado patrón de colores y bordes.

Reacciones del mercader

Ruta del archivo

res://shop_phase/sell/reaction_indicator/

Enlace

https://kenney.nl/assets/emotes-pack

Autor

Kenney

Licencia

CC0

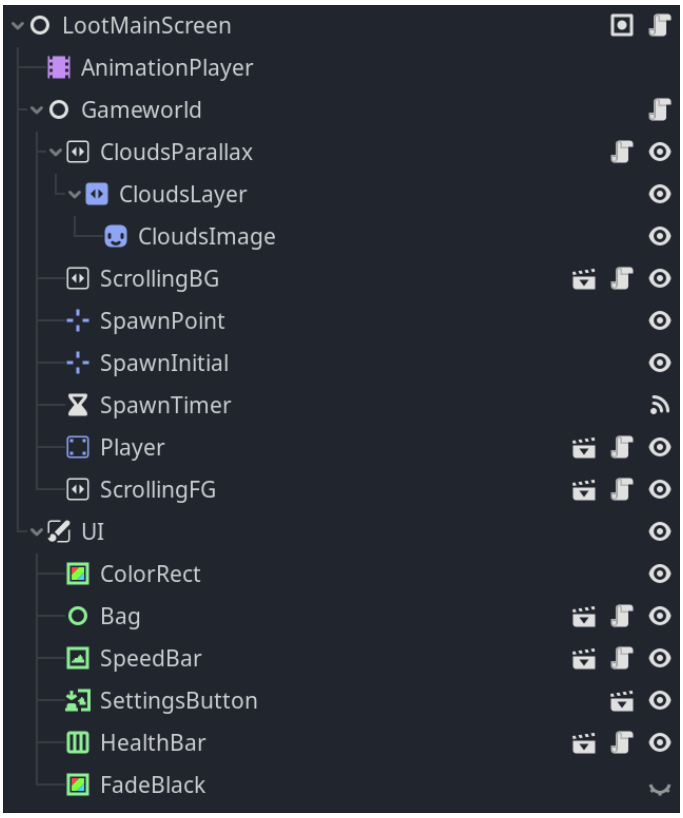
Uso

Pantalla de venta. Modificado tamaño y proporciones.

4.1.2. Arquitectura subsistemas

Pantalla Loot: Recolección y merge

Tabla 9: Jerarquía de nodos Loot



La pantalla ha sido dividida en dos categorías. El Gameworld en la parte superior es dónde se encuentra el mundo de juego, las acciones del jugador se ven reflejadas en este. La UI en la parte inferior gestiona la interacción. La idea es crear una frontera en la que el jugador transfiere los elementos del mundo de juego a la interfaz, pero es el mundo de juego el que dicta “qué está ocurriendo” en la pantalla. Esto también se ve reflejado en el aspecto técnico, el nodo Gameworld gestionará todos los elementos del mundo de juego y se comunicará con el nodo Bag para conseguir la interacción expuesta.

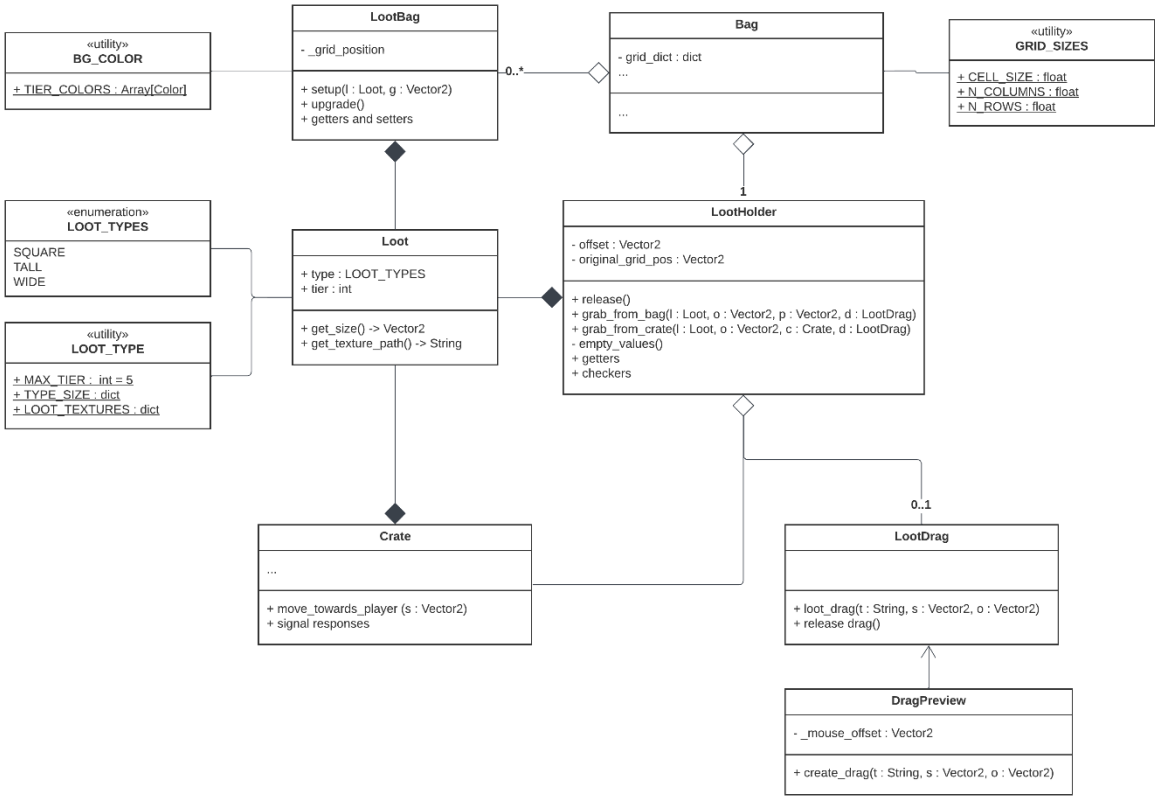
Estructura Bag

La Bag actúa como un sistema de inventario habitual en los juegos del género RPG. El jugador por medio de un sistema *drag and drop*[4] ha de poder recolectar los objetos para añadirlos a su inventario y combinar dichos objetos para mejorar su calidad. Del mismo modo, deshacerse de los objetos menos valiosos para crear espacio (en este aspecto están las consideraciones del tamaño de la cuadrícula, tratadas en el apartado 4.2.2 sobre el diseño). Godot ofrece un sistema *drag and drop* implementado en el engine (*built-in*), pero tras diversas pruebas este no era capaz de responder a mis necesidades.

[4] O sistema de arrastre.

Por lo tanto, la solución no solo debía cubrir el almacenamiento de los objetos, también el sistema de interacción.

Tabla 10: Diagrama de clases entorno a Bag

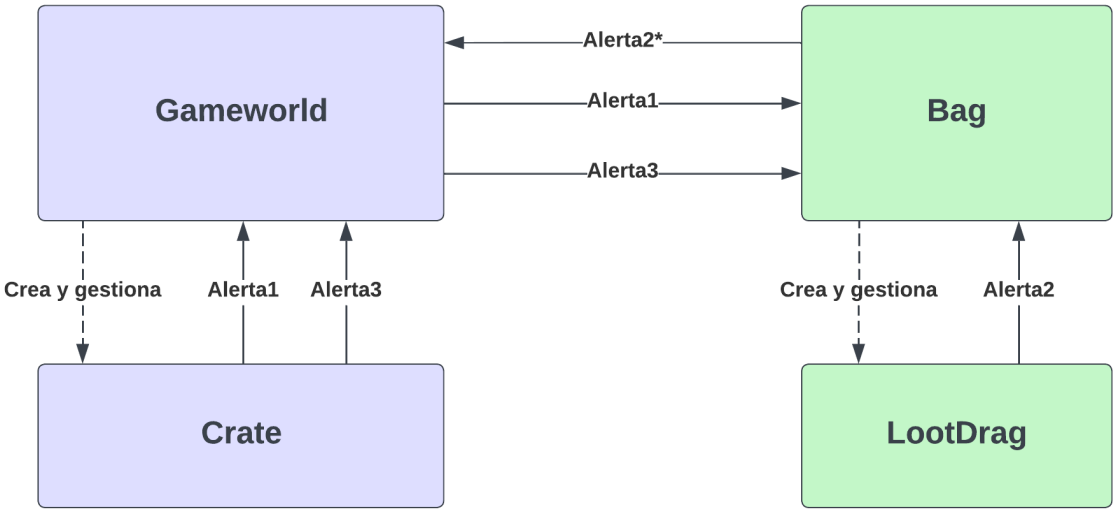


Ya se ha expuesto al inicio de este apartado el diseño de Godot entorno a la OOP, este permite aplicar el paradigma pero impone ciertas restricciones en su aplicación. Este diagrama no es un diagrama ni mucho menos completo de todo el subsistema, de los atributos y métodos contenidos, pero facilitará más adelante la exposición de como funciona la Bag.
Hubiese preferido establecer una relación de herencia entre Loot (padre) LootBag y LootHolder (hijos), pero la funcionalidad del engine se veía limitada porque se sacrificaba otras funciones virtuales necesarias (como las contenidas en TextureRect). Godot tampoco acepta la sobrecarga de funciones, esto se ve reflejado en grab_from_bag y grab_from_crate.

Loot es la clase base que almacena información sobre un objeto. LootBag permite visualizar dicho objeto en la interfaz, instanciándolo en la mochila y es almacenado en Bag. LootDrag es un objeto instanciado durante el uso de drag and drop para indicar al jugador que objeto arrastra y a donde. LootHolder es la pieza central a la hora de coordinar el sistema de arrastre.

Sistema de arrastre

Tabla 11: Diagrama Drag and drop



Ya con una noción de la estructura de Bag, procedo a describir la interacción drag and drop. Cuando el jugador pulsa en la mochila, lo hace para arrastrar un objeto contenido en ella. En el caso de que lo haya (ha pulsado en un objeto LootBag), el objeto es eliminado de la mochila y un objeto LootDrag es creado representando la interacción. LootDrag por si solo no contiene la información que el programa necesita, por lo que Bag actualiza su miembro LootHolder para gestionar que objeto está siendo arrastrado. Al liberar el ratón mientras se usa un objeto LootDrag, una señal es enviada a Bag para determinar el resultado del arrastre (señal 2).

- Si el objeto fue arrastrado fuera de la mochila es tirado.
- Si es arrastrado a un espacio vacío es insertado en ese espacio.
- Si es arrastrado a un objeto compatible, el objeto compatible incrementa su calidad.
- Si las dos operaciones anteriores no son válidas, el objeto es devuelto a su posición original.

Este escenario inicial es relativamente sencillo, pero su complejidad aumenta al incluir Gameworld y Crate. Gameworld generará cajas en el mundo de juego que avanzan hacia el personaje de juego. Estos objetos pueden ser recolectados para obtener objetos en la mochila. Si uno de estos objetos llega al personaje sin haber sido interactuado con éxito, impacta con él y pierde una vida. Gameworld, de forma similar a la Bag con LootDrag, crea y gestiona los objetos Crate en su movimiento e interacción.

La alerta 1 hace referencia a cuando se interactua con un objeto Crate. Este manda una señal a Gameworld, y Gameworld informa a Bag de la interacción. Bag creará un objeto LootDrag en la ubicación del ratón (encima de la caja) y tomará las riendas de la interacción.
La alerta 3 se produce cuando un objeto Crate impacta con el personaje (Player). Manda una señal a Gameworld, y una vez más Gameworld informa a Bag del evento.

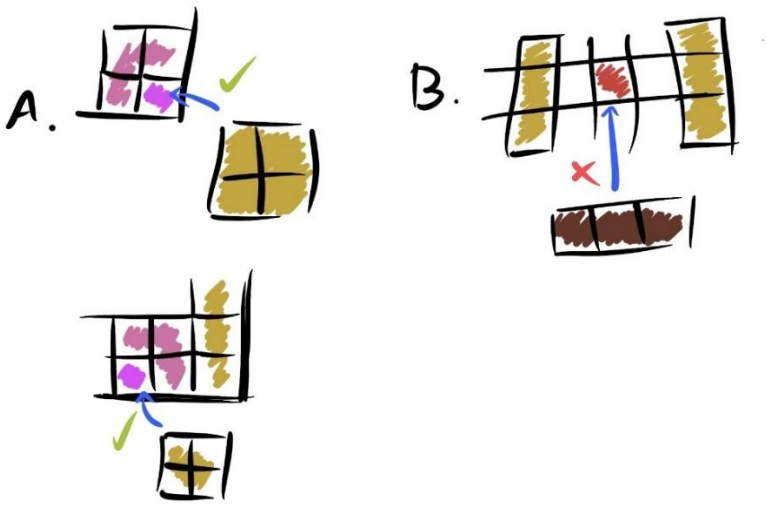
Estas dos alertas hacen que la complejidad de LootHolder se incremente. Bag necesita una referencia a la instancia Crate que ha originado la interacción, para si la operación es exitosa, mandar una señal a Gameworld y este hacerla desaparecer (alerta2*). También es necesaria una referencia a la instancia LootDrag que el mismo genera, porque al producirse la alerta 3 el objeto es devuelto a su posición original y se cancela la operación de arrastre (el ciclo de vida normal de la instancia es interrumpido).

Es importante destacar en esta explicación la distinción entre alerta y señal. En Godot la comunicación entre instancias se produce por medio del uso de referencias o el uso del sistema de señales que proporciona el engine basado en el patrón de observador. Cuando hablo de alertas hablo del flujo de ejecución que desencadena debido a un evento asíncrono en la pantalla. Cuando hablo de señal hablo del sisteam de señales de Godot.

Una práctica recomendada es que los nodos superiores en la jerarquía empleen referencias para comunicarse con los nodos inferiores, y los nodos inferiores empleen señales para comunicar su estado a los nodos superiores. Por ejemplo, Crate se comunica con Gameworld por medio de señales, mientras que Gameworld gestiona Crate por medio de referencias. Hay muchas formas de obtener la referencia a una instancia en Godot, ya sea por medio de su jerarquía en el SceneTree, el uso de grupos o una referencia almacenada entre otros. Este último método es el menos recomendado porque la instancia referida puede desaparecer, invalidando así la variable que almacena esa información. Se recomienda obtener la referencia justo cuando va a ser utilizada, y deshacerse de ella tan pronto la operación ha sido completada.

Antes de terminar esta sección quiero dedicar parte de la exposición a cómo Bag determina la casilla donde el jugador quiere soltar el objeto y cómo verifica su posición resultante. Se ha dedicado una porción de tiempo considerable a hacer la interacción rápida y predecible. Para ello se han empleado 2 técnicas; el uso de un algoritmo recursivo buscador de caminos (Bag. _grid_find_space_available), y el uso de un ancla adaptativa (Bag. _grid_drop_coordinates).

Ilustración 8: Boceto



El algoritmo, dado un nodo de inicio, un número de saltos por dirección válida y longitud mínima del recorrido comprueba si es posible encajar la pieza en la posición deseada. El primero depende del ancla adaptativa, los otros dos de las proporciones de la pieza (forma y tamaño). Me gusta esta solución porque es escalable (se adapta a cualquier pieza válida para el tablero siempre y que un nodo no tenga un grado mayor a 2) (A). En vez de desarrollar líneas de código específicas para cada pieza, dada una casilla de inicio válida sabrá determinar si existe una solución correcta. Siempre busca en el mismo orden haciendo así la función determinista.

El ancla adaptativa me permite proporcionar en la mayoría de los casos el nodo inicial más apropiado para el algoritmo. Dado un objeto del inventario y una posición deseada, el programa divide el tablero en cuatro cuadrantes y comprueba si alguna de las esquinas del objeto sobresale. Si es así, selecciona el ancla apropiada y se selecciona la casilla más cercana. De esta forma es posible evitar que el jugador lance por accidente un objeto del inventario a la vez que se conserva una interacción fluida.

Sin embargo, la solución propuesta no es perfecta (B). El nodo seleccionado ha de tener grado 1 en el camino solución o el algoritmo no encontrará espacio para la pieza. En las piezas 2x2 cualquier nodo puede serlo, pero en las piezas 1x3 y 3x1 no.

De las diversas pruebas realizadas sobre esta interacción, se destacan dos conclusiones:
Siguiendo el principio de Gutenberg (Sinha), el ojo tiende a posarse a la izquierda de la parte superior del objeto, por lo que es recomendable priorizar esa ancla dada que la expectativa del jugador es situar el objeto en base a ese punto.
Si el usuario desplaza un objeto a lo largo de una cuadrícula, permitir un movimiento fluído en vez de restringir la posición del objeto a la cuadrícula (grid snap).

Pantalla tienda: Libro, inventario y acertijos

Tabla 12: Jerarquía de nodos Tienda

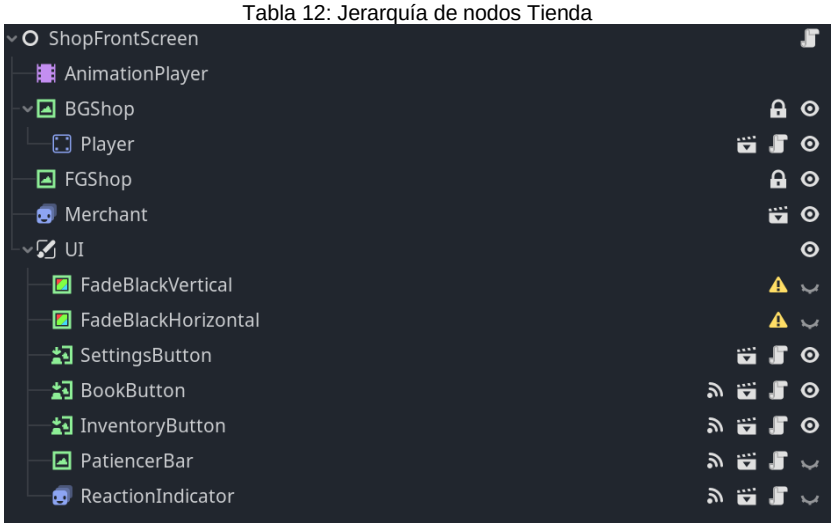
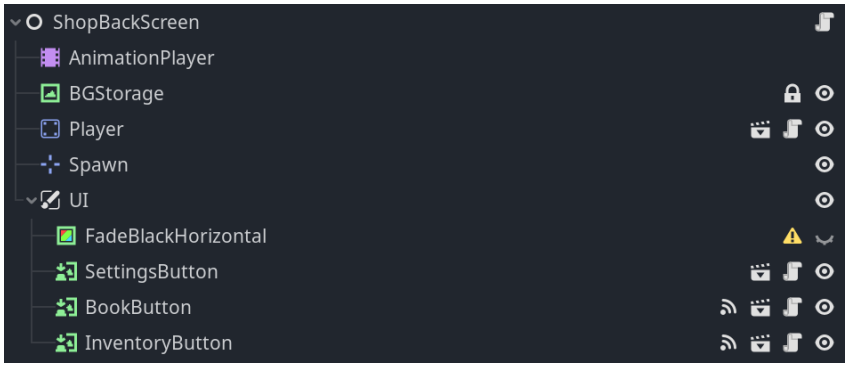


Tabla 13: Jerarquía de nodos Tienda(Almacén)



Las dos pantallas de la tienda siguen el mismo modelo de estructura.

El nodo raíz contiene un script encargado de gestionar el flujo de interacción.

Flujo Tienda(Almacén)

- Entrada desde Tienda
- Revelar
- Tasar
- Salida a Tienda

Flujo Tienda

- Entrada desde exterior para ir directo a Tienda(Almacén)
- Entrada desde Tienda(Almacén)
- Vender
- Salida a exterior

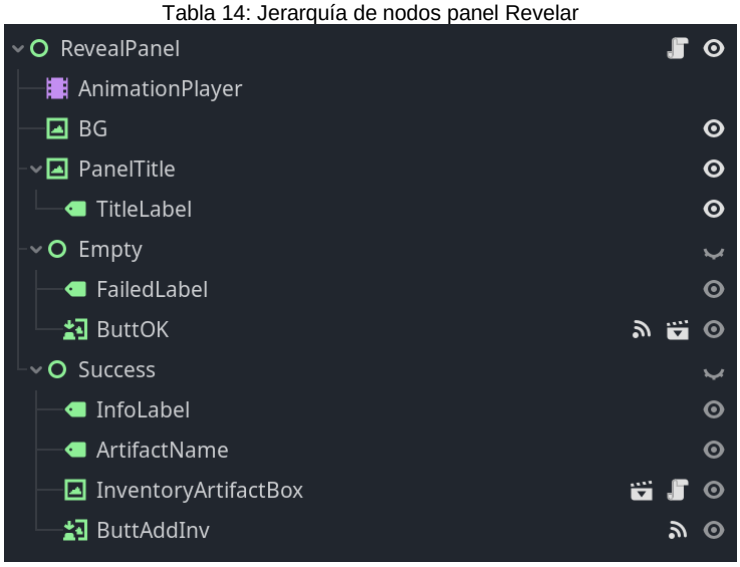
Estas transiciones son gestionadas en gran parte mediante llamadas a el nodo AnimationPlayer que contiene el estado inicial, la animación de transición y el estado final. Los botones permiten acceder a los ajuste, el inventario o la colección (libro) en cualquier momento de la ejecución. El resto de los elementos, incluido el jugador, son elementos visuales para dar soporte a lo que está ocurriendo en pantalla y dar una mayor sensación de interactividad.

La interacción en cada uno de los tres sistemas (revelar, tasar y vender) es llevada a cabo mediante el uso de paneles. Al iniciar una de las fases pertinentes, el nodo raíz instancia un panel que contiene la lógica necesaria para llevar a cabo la interacción. Este panel comunicará por medio de señales del Godot Engine su estado al nodo raíz. Ya sea para que su estado se vea reflejado en el resto de los elementos de la pantalla o para indicar el final de fase (cierre del panel), tras el cuál se prepara la próxima transición.

Por su parte, los paneles son máquinas de estado controladas por su nodo raíz. Del mismo modo, la transición de estado es gestionada por su nodo AnimationPlayer. El nodo raíz modificará los nodos de un estado para que reflejen el contexto de programa actual y delegará en el nodo de transiciones la visibilidad del estado.

Revelado

Tabla 14: Jerarquía de nodos panel Revelar



El panel de revelado es un panel puramente informativo y de lógica simple. En la fase de revelado el nodo raíz instancia las cajas que ha obtenido el jugador durante la recolección. Para abrir la caja, el jugador pulsa repetidamente en ella y esta mandará una señal al nodo raíz conforme empieza el revelado. El panel de revelado dada la rareza de la caja empleará el método draw_artifact() de la base de datos para determinar el resultado e informar al usuario.

Tasar

Tabla 15: Jerarquía de nodos panel Tasar

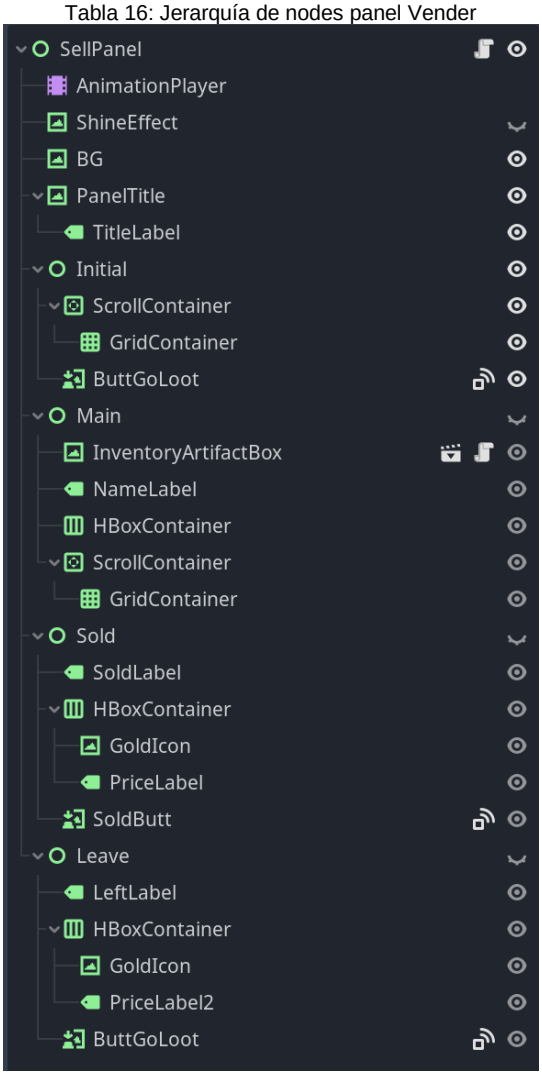
El panel de tasar gestiona la interacción durante la fase homónima. Tiene tres estados; selección de artefacto, coste de tasar y revelado de encanto. Los elementos por destacar son el filtro que aplica en la selección de artefacto y el efecto máquina tragaperras durante el revelado. Como tasar revela una propiedad única de cada artefacto, de los artefactos en el inventario muestra una única instancia. A su vez, solo es posible revelar un encanto por artefacto durante una única fase de tasar (cuestiones de diseño en Diseño del juego).

Ilustración 9: Curva Bézier en visualizador de Godot

Para crear un mayor *engagement* y expectativa una vez se ha realizado el pago, se ha utilizado un nodo que contiene como si fueran fotogramas el icono de todos los encantos en el juego. La velocidad de transición entre fotogramas emplea una curva de Bézier cúbica que hará que la velocidad de cambio de imagen haga este efecto habitual en juegos de azar dónde inicialmente las imágenes cambian con una alta frecuencia y cuando la animación se aproxima a su fin empieza a desacelerar.

Vender

Tabla 16: Jerarquía de nodes panel Vender



El panel de venta es el más complejo tanto por número de estados y por su funcionamiento. Para la selección de artefacto reutiliza el código empleado en el inventario. Durante la fase principal de venta, se coordina mediante señales con el nodo raíz de la escena para indicar el estado de la venta, elemento vital para que el jugador sepa si va por buen camino a la hora de resolver el acertijo. La pieza base es la clase CharmSlot que se instancia durante la ejecución. Este objeto actúa tanto como ranura así como botón. Al pulsar en el botón, la lógica de programa verifica que encanto contiene y cuál es el resultado de la acción dado el artefacto seleccionado. La base de datos almacena que encantos han sido probados y su resultado, esto se refleja en el panel de botones de encanto dónde los útiles son colocados en primer lugar, después los no descubiertos y finalmente los inválidos (en los que ya no se puede volver a pulsar).

Las operaciones de la base de datos están diseñadas para minimizar la necesidad del resto del programa en actualizar el progreso del juego. Esto se ve reflejado al completar un artefacto, dónde es la operación de añadir un intento la que verifica si el artefacto ha sido completado.

Tras completar la venta, las ranuras de encanto permiten saber si era una venta perfecta o una venta válida. Del mismo modo, si el artefacto ha sido vendido de forma perfecta por primera vez aparece un panel de artefacto completado. Es importante dar valor a este evento mediante la posición del panel emergente, su tamaño y elementos visuales porque significa que el jugador está más cerca de completar el juego.

Inventario y libro

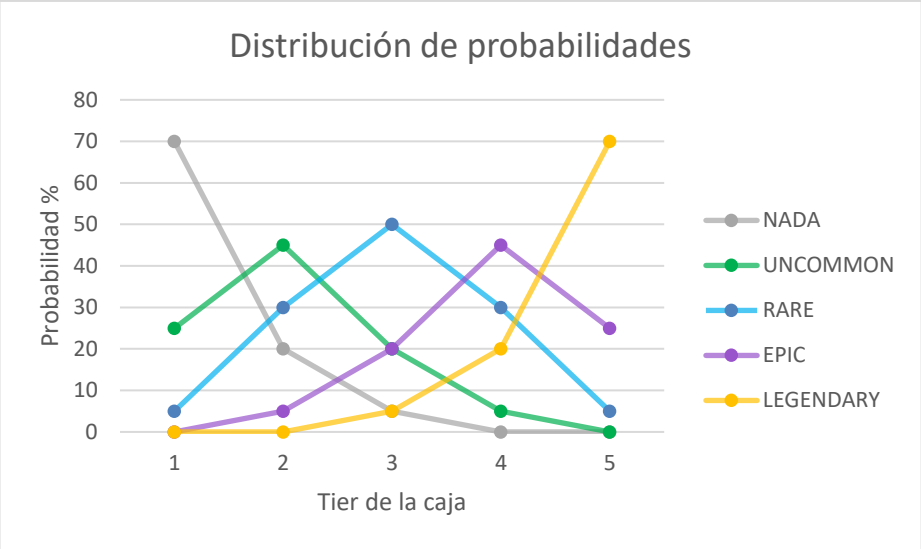
No es necesario ampliar sobre los paneles de inventario y libro porque emplean las mismas técnicas empleadas en los tres anteriores. Son paneles informativos que contienen el estado del juego.

Progresión

Existen dos elementos en el juego que gestionan la velocidad de progreso del jugador. La tabla loottable.csv que determina a partir de las cajas recolectadas el artefacto obtenido y la tabla pricetable.csv que determina el coste de tasar más el oro obtenido al vender un artefacto.

Tabla de recolección

Tabla 17: Distribución de probabilidades loottable



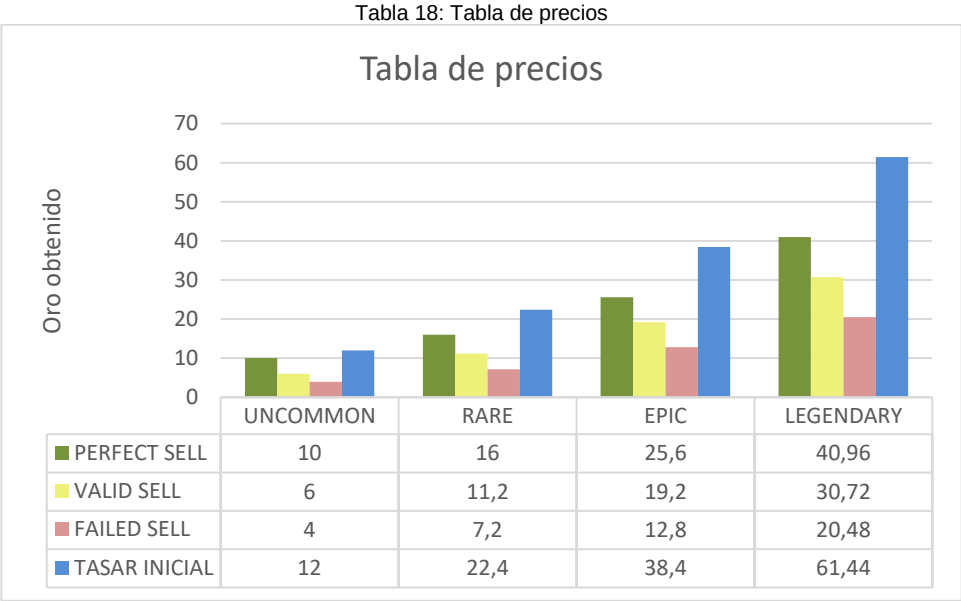
La distribución de probabilidad aplicada es simétrica en el punto central (misma probabilidad para sucesos con la misma distancia respecto el eje central) y con casos extremos en los bordes de la distribución. El propósito detrás de esta implementación es penalizar al jugador en el estado inicial (caja tier 1), recompensar al jugador si consigue un estado final (caja tier 5) y añadir un elemento de azar pero equilibrado para los estados intermedios.

Para garantizar que el jugador pueda seguir progresando independientemente de su suerte, se ha incorporado lo que se llama en muchos videojuegos de móvil un sistema de compasión. El oro tiene un cierto grado de utilidad para el progreso, pero no garantiza que el jugador obtenga los artefactos que

necesita para terminar el juego. Este sistema garantiza que tras tres tiradas independientes en las que ha obtenido un artefacto ya completado, la siguiente tirada obtendrá un artefacto por completar de categoría igual o más alta al que hubiese obtenido sin la intervención del sistema. No es un sistema perfecto porque puede incentivar a que el objetivo del usuario sea acumular cajas de una determinada rareza, ignorando así la premisa de obtener una caja de la mayor rareza posible. Sin embargo es un sistema oculto y pienso que en diseño de juegos suele ser más beneficioso ayudar al jugador pasivo sin castigar al jugador activo.

Tabla de precios

Tabla 18: Tabla de precios



	UNCOMMON	RARE	EPIC	LEGENDARY
PERFECT SELL	10	16	25,6	40,96
VALID SELL	6	11,2	19,2	30,72
FAILED SELL	4	7,2	12,8	20,48
TASAR INICIAL	12	22,4	38,4	61,44

La tabla de precios se ha establecido en base a una serie de premisas:

- Fallar tres veces en la venta de un objeto debe permitir tasar ese objeto o uno de esa categoría.
- Dos ventas válidas deben permitirte tasar ese objeto o uno de esa categoría.
- La venta perfecta de un objeto debe permitirte tasar un objeto de cualquier categoría inferior.
- El precio de tasado y el precio de venta perfecta ha de incrementar significativamente cuanto mayor sea la categoría, y este incremento no ha de ser lineal para así generar un excedente.

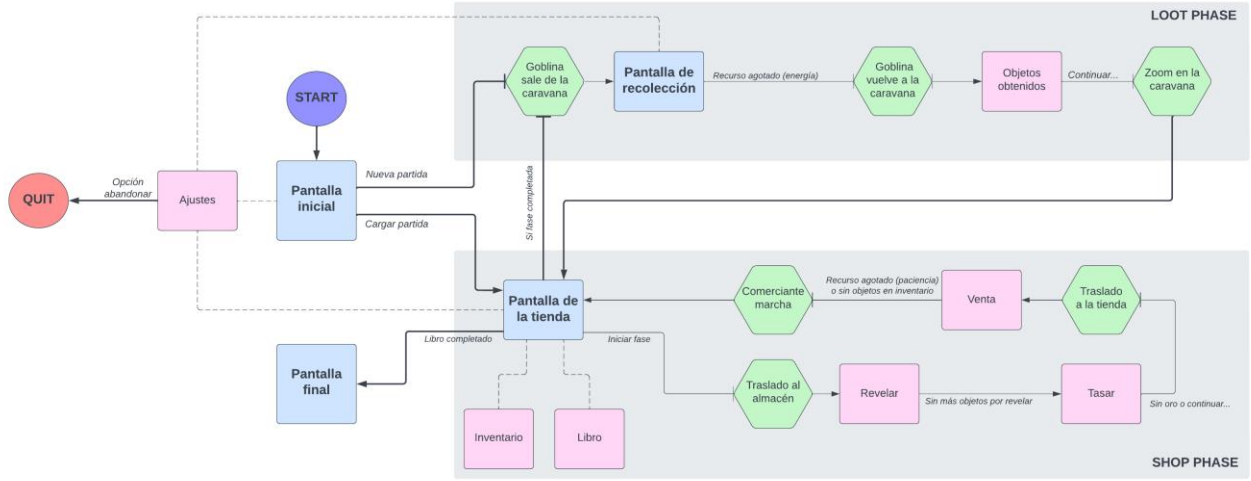
También existe un incremento em el precio de tasar (5%) a medida que el jugador emplea la mecánica, es un incremento suave que en ningún modo busca disuadir el uso del sistema, pero si limitar su abuso (ignorar los acertijos). Los precios se han desarrollado en torno a tasar porque es el único coste de oro en el juego.

4.2. Diseño del juego

4.2.1. Diagrama de navegación

Para ver la navegación inicial consultar Anexo C y 3.2 Gameplay Loop

Tabla 19: Diagrama de navegación



El objetivo de la navegación final es conducir la interacción del jugador en la mayor medida posible. Esto se debe a que el videojeugo ha de ser autoexplicativo, y sin embargo, incluye algunos contenidos como revelar y tasar cuya función puede no ser aparente a primera vista. Mediante este flujo se asegura que el usuario visita las pantallas en el orden adecuado y su funcionalidad es más aparente debido al contexto de la aplicación.

4.2.2. Evolución del diseño

La evolución del diseño puede ser valorada en base a las dos fases principales del videojuego: la fase de recolección y la fase de tienda (o venta).

La fase de recolección ha conservado sus elementos iniciales, en gran parte, porque poseía una referencia clara (2.2 Overloot). Su evolución ha girado entorno a sus parámetros (tamaño de la bolsa y los objetos almacenados en ella) y a su progresión.
La bolsa inicial era demasiado grande. Este hecho hacia que la toma de decisiones significativas se demore y reducía la utilidad de la mecánica merge. Una bolsa más pequeña permite que de forma temprana el jugador considere que objetos vale la pena conservar y cuales es mejor tirar para así hacer espacio al próximo objeto. También da mayor utilidad a combinar los objetos, ya no es solo una forma de obtener mejores cajas, sino que garantiza la posibilidad de crear espacio en la bolsa. Las proporciones finales de la cuadrícula y piezas es un reflejo de esta filosofía porque no siguen unos valores convencionales en los que suele ser intuitivo llenar la bolsa sin dejar espacios libres. Su

objetivo es que el jugador ponga atención a la calidad de los objetos recolectados por encima de rentabilizar al máximo el espacio disponible.
En lo que se refiere a progresión, estaba claro desde el principio que la recolección debía verse restringida por algún factor que creara un sentimiento de urgencia. El concepto inicial era un medidor de tiempo que determinaba la duración de la fase de recolección, por lo tanto el jugador debía recolectar la mejor bolsa posible en ese espacio de tiempo. El problema de esta solución es que no tenía en cuenta el rendimiento del usuario durante la partida. La siguiente iteración hizo que el medidor de tiempo incrementara o disminuyera en base a las acciones del jugador. Probé con incrementar el medidor al recolectar un objeto o al combinarlo, o disminuir el medidor cuando un objeto impactaba en el jugador. Finalmente, centré mi atención en el factor más significativo de la interacción, cuando el jugador impacta con la caja porque no ha conseguido crear espacio en la bolsa. A partir de ahí decidí aplicar el sistema de vidas actual. Prefiero esta solución porque es intuitiva, es el clásico sistema de corazones habitual para este perfil de videojuego (2D *sidescroller*). Posteriormente, he añadido el sistema de marchas, cuanto más tiempo está el personaje sin haberse detenido, más rápido avanza el juego.
En los conceptos iniciales había una fase runner tras la fase de recolecta. Fue declinada de los diseños posteriores por varios motivos; suponía una carga de trabajo adicional que no justificaba su presencia, pretendía introducir mecánicas *sidescroller* que ya ha conseguido introducir la fase de recolección y el énfasis de la aplicación está en la fase de tienda.

La fase de tienda, en especial el sistema de venta o negociación, ha sido la que ha experimentado una mayor evolución a lo largo del proyecto.
El sistema de revelado fue generado desde su concepción y respondía a la necesidad de transformar unos objetos homogéneos como son los recolectados en la fase anterior en objetos heterogéneos que serán vendidos. De esta forma, durante la recolección la atención del usuario se centra en acumular objetos y mejorar su calidad, y distinguir estos objetos por su forma o espacio que ocupan en la bolsa y no cualquier otra característica inherente del objeto.
A pesar de ser un juego basado en el comercio, desde el principio quise evitar que el único objetivo del jugador fuese amasar una fortuna. La moneda del juego tiene el valor que tenga los productos o servicios que ofreces a cambio de esta, pero este diseño no contemplaba elementos adicionales, buscaba que todo el énfasis residiera en el acto de comerciar. Por lo tanto, tenía que ser el propio sistema de comercio el que incentivara a jugar, y no su resultado. Tasar es el medio que aporta utilidad al oro, y permite no retrasar el progreso del usuario si un acertijo se le resiste. Los encantos revelados mediante tasar no son desbloqueados, sino que el jugador los debe recordar. Esta decisión sigue la filosofía en la que se basa el juego con el conocimiento como medida de progreso.

En un principio, el sistema de comercio iba a ser uno o más minijuegos de reacción o memoria integrados en una sola unidad. Sin embargo, pronto encontré que esta solución no respondía a los

retos que presenta un TF. Esto me ha conducido al desarrollo de varias iteraciones siguiendo las algunos principios.

- El modelo propuesto debe incluir alguna de las características presentes en una negociación real.
    - En una negociación real intervienen muchos factores como la necesidad del comprador, la capacidad del vendedor para crear esa necesidad o la necesidad intrínseca que es capaz de cubrir el objeto.
- El jugador no debe de ejercer la negociación desde una posición de conocimiento absoluto.
    - Es una premisa tanto basada en la ambientación como en donde reside el desafío del sistema. En la mayoría de los sistemas de comercio en los videojuegos el jugador conoce con exactitud cuanto dinero puede obtener de la venta o quién está dispuesto a comprarlo (algo comprensible ya que suele ser un sistema de soporte, pero no es el caso en este diseño).

En las iteraciones iniciales los objetos poseían estadísticas similares a los objetos en un juego RPG. Estas estadísticas, o algunas de ellas, permanecieron ocultas para el jugador. Además, no se informaba al jugador del valor inicial del objeto, por lo tanto, el valor de sus estadísticas. La idea era que el jugador infiriera el valor del objeto a partir de las ofertas iniciales del comerciante, de las características visuales del objeto o del valor de las estadísticas no ocultas, así como deducir cual podría ser la estadística oculta. De esta forma, conocer que estadísticas eran más codiciadas. Entre los motivos por los que decidí rechazar esta propuesta, fue porque estaba enfocado a una audiencia muy diferente de la que quería apelar mediante la fase de recolecta o la ambientación del videojuego.

La siguiente iteración se centraba en establecer una relación entre el comerciante y el objeto. Habría diferentes comerciantes, y siguiendo el principio que un objeto no tiene el mismo valor para todos, habría distintos niveles de afinidad entre objeto y comerciante. El objetivo del jugador era encontrar el comerciante con mayor afinidad hacia el objeto que desea vender, y obtener el mejor precio posible. Para ello, planteaba un sistema de reacciones similar al actual (también uno de paciencia) y un ciclo de oferta más contraoferta. El comerciante ofrecía un precio inicial por el objeto (el cuál podía no corresponder a su interés), y el jugador podía pedir más, o aceptar la oferta. Durante este ciclo las reacciones del comerciante, el medidor de paciencia y el valor de la oferta del comerciante debían permitir al jugador discernir cual era su grado de verdadero interés por el objeto. Si de verdad se trataba de un objeto afín o no. El principal motivo por el sentí la necesidad de seguir iterando más allá de esta propuesta era el coste de recursos gráficos y la limitada interacción (el sistema podría ser más o menos complejo a nivel interno, pero al final tan solo se le presentaban dos opciones al jugador, y nada obligaba a que su decisión tuviese las consideraciones que yo esperaba).
Tras varias iteraciones adicionales, llegué al sistema actual mediante la limitación de un solo comerciante y centrar la interacción en descubrir el valor del objeto. El objetivo de este sistema es crear

un juego lineal y basado en conocimiento para proporcionar una experiencia completa en una única partida, sin la necesidad de introducir elementos de rejugabilidad. Considero que el uso de acertijos y elementos visuales para sus resoluciones hace este sistema cohesivo con la ambientación y el resto de diseño del juego. En un inicio, y lo que podría ser una línea de futuro, cualquier combinación de encantos válidos generaba una frase de encanto apropiada. De esta forma, no se trataba tanto de un uso específico del objeto, sino de apelar al comerciante con una posible utilidad del objeto.
Lógicamente, la calidad de la experiencia de juego depende en gran medida en la calidad de los acertijos y su dificultad. Pienso que los acertijos actuales son muy mejorables pero permiten tener una idea clara del sistema. El diseño de estos acertijos ha sido una actividad muy interesante. Estoy bastante orgulloso de los encantos elegidos y su iconografía, creo que son flexibles y los iconos cubren con precisión los distintos conceptos que buscan evocar.

4.3. Diseño gráfico e interfaces

4.3.1. Estilos

Título

Ilustración 10: Gráfico del título

Logotipo

Ilustración 11: Gráfico de logotipo

Paletas

Ilustración 12: Paleta título y UI

Ilustración 13: Paleta Goblina

Ilustración 14: Paleta "tiers"

Fuentes

Ilustración 15: Fuente "Take Coffee"

Para el título se ha empleado una transformación de la fuenta Take Coffe de Khurasan. Es una fuente de uso gratuito tanto personal como comercial. (Khurasan)

Ilustración 16: Fuente "Zero Cool"

Para el texto UI se ha empleado la fuente Zero Cool de GGBotNet. Es una fuente de licencia abierta. (GGbotNet)

GUI

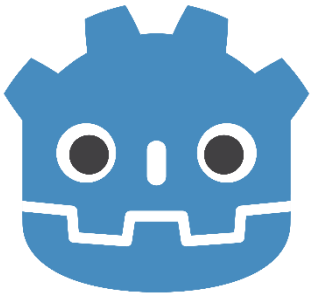
Ilustración 17: Muestra GUI

Para la interfaz gráfica se ha utilizado el pack Free Fantasy Game UI del autor pzUH. Registrado bajo la licencia de dominio público. (pzUH)

4.4. Recursos tecnológicos utilizados

4.4.1. Engine y elección

Ilustración 18: Logo Godot Engine



*Para este proyecto se ha optado por emplear **Godot 4**. A continuación expongo su justificación.*

Desde un punto de vista técnico habría sido recomendable hace uso de librerías, sin embargo, creo que el alcance del videojuego propuesto es lo suficientemente ambicioso como para justificar el uso de un engine. Considero que el uso de librerías y frameworks ha de atender a las características del videojuego; requisitos gráficos, técnicos, de diseño, … Las mecánicas propuestas por este proyecto son muy compatibles con el engine elegido, y a su vez, suponen un desafío de diseño a la altura del TF.

He elegido Godot porque es un software ligero, particularmente útil en proyectos 2D, de código abierto y basado en C++. Emplearé el lenguaje GDScript porque es lenguaje integrado de la aplicación. Esto permite obtener la misma funcionalidad, y a menudo, en líneas de código más simples. He usado con anterioridad Python por lo que me siento cómodo en su uso. El sistema de eventos es fácil de usar y es posible escribir fragmentos de código semi asíncrono. El engine propone una arquitectura basada en OOP que permite controlar la evolución del proyecto y su escalabilidad. Cuenta con una comunidad activa, lo cual es muy importante en la búsqueda de soluciones o al encontrar defectos en el programa. Tiene despliegue multi plataforma, lo que permite mantener mis opciones abiertas de cara al futuro. Una de mis principales motivaciones detrás de su elección es que considero el engine Godot una gran herramienta de prototipado, y este proyecto me permite adquirir las habilidades que en el futuro me permitan un rápido despliegue de prototipos que pongan a prueba mis ideas.

Además, el 1 de marzo de 2023 fue lanzada la versión 4.0. Este lanzamiento supone un gran acontecimiento para la plataforma. Creo que optar por desarrollar el proyecto en esta versión brinda la posibilidad de hacer uso de estas nuevas prestaciones y elaborar código adaptado a la nueva versión. *Listado de cambios* (Wilkes).

A continuación otros *engines* populares que han sido considerados:

- Unity
    - Es el mejor engine multi propósito, pero creo que esa misma virtud a veces actúa en detrimento. Pienso que si tienes claro que producto quieres desarrollar, el proyecto se beneficiará de escoger herramientas especializadas en esas características. Tampoco me agrada la idea de emplear C# para el desarrollo de scripts.
- Unreal Engine
    - Se especializa en gráficos 3D. Tiene una curva de aprendizaje empinada, además del hecho de que este proyecto es para un videojuego 2D.
- Game Maker
    - Su lenguaje propietario, similar a JavaScript, no me convence. Si bien la mayoría de las mecánicas en el videojuego propuesto son bastante convencionales, creo que Godot me permite un mayor grado de flexibilidad y creatividad.
- Construct3
    - No cumple con el desafío técnico que exige el TF.
- Defold
    - Basado en Lua, lenguaje en el que no tengo experiencia previa (como mucho elaborando macros en el videojuego World of Warcraft). Comunidad muy reducida.

4.4.2. Producción de recursos gráficos

Para la elección de *assets* se ha intentado siempre priorizar el uso de gráficos de licencia de dominio público. Incurrir en costes relacionados con el apartado visual se ha considerado innecesario para un proyecto no comercial.

Durante el desarrollo del prototipo se han empleado numerosos recursos que provenían de (The VG Resource, s.f.), sin embargo, los recursos contenido en esta web son para el uso de proyectos personales y no comerciales. Por lo tanto, era necesario en futuras fases la búsqueda de recursos válidos y legales para este proyecto.

El dominio web no esta falto de plataformas en las que adquirir dichos recursos (Game Dev Market, s.f.) (Game Art 2D, s.f.) (Kenney.nl, s.f.)… Pero es una rareza encontrar gráficos de licencia libre y que se ajusten a las características del proyecto para proporcionar una experiencia cohesiva. Ya sea por un estilo visual distinto (*pixel art*, *voxel*, *isommetric game*, …) o por un género distinto (la mayoría de los recursos 2D son para juegos de plataformas).

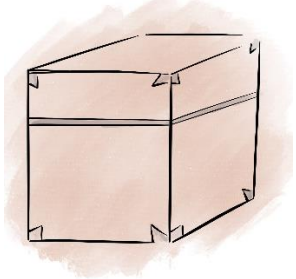
Tras experimentar con la producción de *assets*, he llegado un proceso que hace uso de las nuevas tecnologías para suplir mis carencias artísticas. Para ello empleo el software Stable Diffusion (AI, s.f.), una herramienta desarrollada por Stability AI. Este sistema emplea un modelo de difusión para generar imágenes basadas en; texto o *prompt*, una serie de parámetros configurables y, opcionalmente,

imágenes de referencia. Apoyado por la aplicación para iPad Procreate y el uso del ApplePen, puedo refinar mis bocetos o aportar cohesión a las imágenes resultantes de StableDiffusion. Las imágenes producidas de esta forma caen bajo la licencia Creative ML OpenRAIL-M (Stable Diffusion), por lo que son válidas para este proyecto.

Proceso de creación asset “crate”

Para la fase de recolección, necesitaba una especie de contenedor que el personaje recogería y del que luego obtendría los artefactos. La idea de recoger cofres esparcidos por el mundo de juego me parecía excesiva, por lo que intente crear cajas de madera que no fueran necesariamente lujosas pero con una estética similar a un cofre.

Ilustración 19: Concepto inicial



Introduje el concepto inicial en la aplicación de Open Art para realizar un Img2Img. Decidí emplear el modelo público DreamShaper porque me había dado buenos resultados con anterioridad. El *prompt* que mejor resultados me dio es “a game art of a crate.” Tras varías iteraciones modificando los parámetros de Strength (ruido añadido a la imagen), CFG Scale (similitud con *prompt* e imagen de origen) y Steps (pasos de reducción de ruido) obtuve una iteración que era de mi agrado.

Quiero destacar que mi experiencia y habilidades con esta herramienta son limitadas. Con unos buenos conocimientos en la creación de prompts y ajustes de parámetros es posible conseguir resultados increíbles. Recurso recomendado de aprendizaje (OpenArt AI)*.*

Ilustración 20: Mejor iteración

A partir de esta iteración modifiqué la imagen usando Procreate con el objetivo de:

- Eliminar el efecto pincel de fondo.

- Marcar los bordes a pulso, para añadir imperfección.
- Colorear la cerradura.
- Gestionar el balance de colores.
- Cambiar la parte superior de la caja
- Centrar la imagen

Ilustración 21: Versión modificada

Este tan solo es un ejemplo de las capacidades del software StableDiffusion. También ha sido empleado para generar los fondos de la tienda, reescalar imágenes o para corregir imperfecciones en mis dibujos.

4.4.3. Desarrollo diseño y estructuración de ideas

Ilustración 22: Página Notion (Initial Pitch)

Loot hoarder (UOC - initial)

Planifico

prototipo ipad

pitch

Controlas a un goblin, con un saco enorme en el que acumula todo los objetos valiosos que encuentra.

Aprovecho esta breve sección para destacar la utilidad de la herramienta Notion (Notion, s.f.). No solo ha sido útil para gestionar el desarrollo del proyecto (tareas, contenidos, estructura, …), pero también para el desarrollo del diseño del videojuego. Para este tipo de proceso es necesario estructurar las ideas de forma ordenada, poder apoyarse en imágenes, incorporar vídeos integrados, … Su fácil uso me ha permitido organizar todo el proceso creativo y emplear los recursos generados como referencia en las futuras iteraciones. Cabe destacar, que este proceso no tiene porque limitarse al uso de Notion, en este caso elegido debido a preferencia personal. Otras herramientas recomendables son EverNote, MilaNote, ClickUp, … Finalmente, siempre es recomendado tener a mano un cuaderno y lápiz.

5. Conclusiones y líneas de futuro

5.1. Conclusiones

Estos meses de desarrollo me han permitido adquirir muchas lecciones. Sobre mí proceso de trabajo, sobre el desarrollo de videojuegos, etc.

Me gustaría empezar por la capacidad para perseverar. No es algo que me faltase ni tampoco un absoluto que se pueda tener como tal, pero durante este proyecto me he enfrentado a muchos nuevos desafíos, que en el pasado me podrían haber superado y en esta ocasión he solventado. Contaba con todos los conocimientos adquiridos en el grado para enfrentarlos, pero dependía de mí saber ponerlos en práctica. Era la primera vez que diseñaba un juego de principio a fin, tanto en diseño del juego como en diseño del programa. Hasta ahora mi experiencia se limitaba a sistemas aislados en los que probaba las capacidades de los distintos engine con los que decidía trastear. De todo este proceso (incluido el desarrollo de la memoria) he adquirido muchas lecciones. De algunas ya tenía una noción, y de otras ya había oído el consejo con anterioridad (como algunos de los consejos en 2.1), pero la experiencia tiende a ser el mejor maestro.

Sobre el desarrollo de videojuegos he podido corroborar la importancia del prototipado. De ahí residen algunos de mis errores pero también valoro mi adaptación. Poder establecer un prototipo lo antes posible es la mejor forma de: Determinar que funciona en un diseño de juego, que es factible a nivel técnico, de poder desarrollar una planificación coherente, etc. Durante este proyecto, tardé más de lo que debía en empezar el desarrollo del apartado técnico. Dediqué demasiado tiempo al diseño del juego, un diseño que ha ayudado en el desarrollo de la memoria, pero cuya validez como he comprobado siempre depende del prototipo. Es mucho más recomendable establecer unas bases y referencias, y a partir del prototipo resultante, desarrollar el diseño final. También considero que ha sido equivocado el análisis funcional del prototipo. Se ha invertido tiempo en tareas que eran propias de etapas más avanzadas, como el refinado de código, sistemas avanzados o elementos visuales detallados. El gran problema de este error es que no solo retrasa el seguimiento de la planificación, sino que los recursos invertidos pueden llegar a ser en vano debido a la necesidad de una modificación en la siguiente fase. Estos factores me han llevado a asumir una deuda técnica que se ha visto arrastrada hasta las fases finales del proyecto. A pesar de este grave error y como ha perjudicado al proyecto (se detalla más adelante), valoro positivamente mi capacidad para adaptarme a esta situación, modificando el plan técnico y el diseño para conseguir cumplir con las fechas establecidas.

Después del prototipado aparece la versión jugable, y no ha sido hasta la fase de vertical slice que no he visto la importancia de establecer, como mínimo, cómo se obtendrán los recursos audiovisuales finales. Actúe bajo la esperanza de que en la basta red de Internet encontraría amplías opciones, y no

resulta una tarea tan sencilla. Creo que es muy recomendable establecer antes o durante la versión jugable un inventario de assets definitivo, y no esperar a iteraciones finales. Una nota más optimista ha sido el uso de las herramientas seleccionadas. Este proyecto no solo me ha permitido mejorar en gran medida mis capacidades en el uso del engine Godot, también en el resto de las herramientas como Inkscape, DragonBones, StableDiffusion o Procreate.

Volviendo a la deuda técnica, ha significado que prácticamente las fases de versión jugable y vertical slice se han solapado. Es una práctica muy poco recomendable y de cara al futuro evitaré a toda costa. Si ha funcionado el uso de la metodología Kanban para establecer prioridades y tareas durante el desarrollo, y dentro del retraso en la planificación, me ha permitido seguir un desarrollo ordenado.

En líneas generales, no esperaba las muchas dificultades técnicas que podía encontrar. Mi nivel de dedicación ha sido alto, pero desgraciadamente muchas sesiones de trabajo fueron infructíferas al no conseguir las tareas propuestas para esa jornada. Esto me ha permitido aprender mucho, y he pasado de programar por primera vez con Godot a un nivel actual en el que tareas que al inicio podían llevarme más de un día ahora es cuestión de horas.

Sobre el producto final, qué se ha conseguido y de qué se ha tenido que prescindir, destaco los siguientes puntos:
Valoro que ha sido posible llevar a cabo un diseño de videojuego y programa bastante particular, por lo que no se ha recurrido a soluciones genéricas, todo el código y arquitectura fue creado a partir de cero. También estoy satisfecho con el diseño propuesto, considero su complejidad acorde a los requisitos de un TF y de las restricciones temporales.
No se ha podido desarrollar pruebas de usuario, pruebas de accesibilidad o un sistema de guardado debido al retraso en el proyecto. Del mismo modo, el objetivo inicial era desarrollar un videojuego para la plataforma móvil, pero creo que la adaptación de controles ha conseguido respetar los motivos por los que había escogido la anterior plataforma.

5.2. Líneas de futuro

Me gustaría seguir trabajando en el juego en los próximos meses. Desarrollar los sistemas que no han podido estar presentes en el producto final, optimizar las transiciones de escenas, revisar elementos del juego, … Sobre todo porque me permitirá seguir aprendiendo sobre el desarrollo en fases avanzadas del proyecto. En especial, el apartado de sonido para el cuál tenía previsto hacer muchas más cosas.

El objetivo del producto seguirá siendo el mismo que ha sido expuesto en este trabajo.

Referencias

(s.f.). Obtenido de Boss Fight Books: https://bossfightbooks.com/

AI, S. (s.f.). *https://stablediffusionweb.com/#faq*. Obtenido de Stable Diffusion: https://stablediffusionweb.com/

Alex Tsekhansky. (8 de 3 de 2023). *https://www.theknightsofunity.com/about-us*. Obtenido de TheKingsOfUnity: https://blog.theknightsofunity.com/game-production-pipeline/

Ars Technica. (21 de 4 de 2019). *https://about.youtube/.* Obtenido de Youtube: https://www.youtube.com/watch?v=OMi6xgdSbMA

Baker, C. (9 de 3 de 2016). *https://www.gamedeveloper.com/about-game-developer.* Obtenido de Game Developer: https://www.gamedeveloper.com/business/the-4-years-of-self-imposed-crunch-that-went-into-i-stardew-valley-i-

Blobfish. (s.f.). *https://www.blobfish.dev/about/.* Obtenido de Blobfish dev: https://www.blobfish.dev/68-gamedev-ideas/

Game Art 2D. (s.f.). *https://www.gameart2d.com/about.html*. Obtenido de Game Art 2D: https://www.gameart2d.com/

Game Dev Market. (s.f.). *https://www.gamedevmarket.net/about/gdm*. Obtenido de Game Dev Market: https://www.gamedevmarket.net/

GDC. (20 de 4 de 2017). *https://about.youtube/.* Recuperado el 8 de 4 de 2023, de Youtube: https://www.youtube.com/watch?v=RiDy6CgBKqs

GDC. (s.f.). *https://about.youtube/*. Obtenido de Youtube: https://www.youtube.com/playlist?list=PL2e4mYbwSTbbiX2uwspn0xiYb8_P_cTAr

GGbotNet. (s.f.). *https://www.ggbot.net/fonts/.* Obtenido de DaFont: https://www.dafont.com/zero-cool.font

Godot Engine. (s.f.). Obtenido de Godot Engine Manual Reference: https://docs.godotengine.org/en/stable/tutorials/best_practices/what_are_godot_classes.html

Godot Engine. (s.f.). *Godot Docs.* Obtenido de Data Preferneces: https://docs.godotengine.org/en/stable/tutorials/best_practices/data_preferences.html

https://dev.to/about. (s.f.). Obtenido de DEV: https://dev.to/

https://lostgarden.home.blog/about/. (s.f.). Obtenido de Lost Garden: https://lostgarden.home.blog/

https://www.gamedeveloper.com/about-game-developer. (s.f.). Obtenido de Game Developer: https://www.gamedeveloper.com/

https://www.gdcvault.com/help.php. (s.f.). Obtenido de GDC Vault: https://www.gdcvault.com/

Itch IO. (s.f.). *https://itch.io/docs/general/about.* Obtenido de Itch IO: https://itch.io/docs/creators/html5

Kenney.nl. (s.f.). *https://kenney.nl/support*. Obtenido de Kenney: https://kenney.nl/

Khurasan. (s.f.). *https://www.dafont.com/faq.php.* Obtenido de https://www.dafont.com/take-coffee.font: https://www.dafont.com/take-coffee.font

Lin, A. (23 de 2 de 2016). *https://cornellsun.com/about/.* Obtenido de The Cornell Daily Sun: https://cornellsun.com/2016/02/23/stardew-valley-pushing-the-boundaries-of-farming-rpgs/

Machkovech, S. (1 de 12 de 2019). *https://arstechnica.com/about-us/.* Obtenido de Ars Technica: https://arstechnica.com/gaming/2019/01/from-uncharted-to-obra-dinn-lucas-pope-dishes-on-his-illustrious-game-dev-career/

Noclip - Video Game Documentaries. (15 de 4 de 2017). *https://about.youtube/.* Recuperado el 8 de 4 de 2023, de Youtube: https://www.youtube.com/watch?v=jv434Xyybqc

Notion. (s.f.). *https://www.notion.so/about*. Obtenido de Notion: https://www.notion.so/

OpenArt AI. (s.f.). Obtenido de Blog OpenArt AI: https://blog.openart.ai/2023/02/13/the-most-complete-guide-to-stable-diffusion-parameters/

Pope, L. (24 de 5 de 2014). *TIGSource.* Obtenido de TIGSource: https://forums.tigsource.com/index.php?topic=40832.0

pzUH. (s.f.). *https://opengameart.org/content/faq.* Obtenido de OpenGameArt: https://opengameart.org/content/free-fantasy-game-gui

Singal, J. (14 de 3 de 2016). *https://www.vulture.com/about-us/.* Obtenido de Vulture: https://www.vulture.com/2016/03/first-time-developer-made-stardew-valley.html

Sinha, S. (s.f.). *https://medium.com/about?autoplay=1.* Obtenido de UXDesign: https://uxdesign.cc/gutenberg-modern-day-ux-14ac545e151c

Stable Diffusion. (s.f.). Obtenido de Stable Diffusion License: https://huggingface.co/spaces/CompVis/stable-diffusion-license

Stefyn, Nadie. (5 de 9 de 2022). *https://www.cgspectrum.com/contact*. Recuperado el 8 de 3 de 2023, de CGSpectrum: https://www.cgspectrum.com/blog/game-development-process

The VG Resource. (s.f.). *https://www.spriters-resource.com/page/about/*. Obtenido de Spriters Resource: https://www.spriters-resource.com/

TIGForums. (s.f.). Obtenido de TIGSource: https://forums.tigsource.com/

White, S. (20 de 3 de 2018). *https://www.condenast.com/brands/gq.* Obtenido de GQ: https://www.gq.com/story/stardew-valley-eric-barone-profile

Wilkes, A. (s.f.). *https://gdscript.com/about/.* Obtenido de GDScript: https://gdscript.com/articles/godot-4-gdscript/

Yu, D. (2016). *Spelunky by Derek Yu.* Boss Fight Books.

Yu, D. (s.f.). *Tumblr.* Obtenido de Make Games: https://makegames.tumblr.com/

Anexos

Anexo A: Glosario

Asset: Término reducido para hacer referencia a todos los elementos (normalmente audiovisuales) que conforman un videojuego. Ya sean entornos, objetos, sonidos, efectos, …

Gameplay loop: Describe el ciclo de repetición de mecánicas y sistemas durante el desarrollo de una partida del videojuego.

Merge: Mecánica de videojuegos en la que después de combinar dos elementos del mismo tipo o con las mismas características se espera obtener un nuevo elemento con dichas características mejoradas.

Engine: En este caso utilizado como abreviación de *game engine*. Hace referencia a una aplicación para la creación de videojuegos que incluye todas las herramientas y funciones básicas de desarrollo para permitir al desarrollador economizar el proceso de desarrollo.

Indie: Término empleado en muchas industrias creativas cómo la de los videojuegos, la cinematográfica, la musical, … Describe un trabajo o colectivo que no está afiliado a una compañía líder del sector o con amplios recursos.

Solodev: Desarrollo de videojuegos por un solo individuo.

Framework: Estructura de soporte que sirve como base para la construcción de nuevas funcionalidades y servicios.

Sprint: En este caso, término empelado en la filosofía Agile que indica un período de tiempo establecido previamente durante el cuál se ha de completar un trabajo o tarea.

Work in progress: De traducción literal “trabajo en progreso”, es el estado de tareas cuyo trabajo ha sido iniciado pero todavía no han sido completadas.

Roguelike: Género de videojuegos que habitúan a incluir elementos como la generación procesal, sesiones cortas de partida y muerte permanente del personaje.

MOBA: Siglas empleadas para el género de videojuegos *Multiplayer Online Battle Arena* en la que dos equipos de jugadores compiten en un campo de batalla en igualdad de condiciones.

Battle-royale: Género de videojuegos dónde los jugadores son distribuidos en diferentes zonas de un campo de batalla y se proclama vencedor el último jugador

con vida.

Engagement: En análisis de experiencia de usuario hace referencia a la capacidad de un producto para atraer y retener la atención del usuario.

BD: Base de datos

Pixel art: Forma de arte digital en el que las imágenes 2D son construidas empleando píxeles como base de construcción.

Voxel art: Forma de arte digital en la que los gráficos 3D son construidos empleando vóxeles (píxel volumétrico) como base de construcción.

Isommetric game: Videojuego que hace uso de la perspectiva isométrica para la creación de sus gráficos.

Drag and drop: Gesto en el uso de interfaces de usuario que permite “arrastrar” un objetivo virtual de una posición a otra “agarrando” y soltando” .

Playtest: Proceso de prueba de un videojuego en búsqueda de fallos de código o diseño.

Streamer: Persona que retransmite en directo, normalmente desde su domicilio, en plataformas como Twitch, Youtube, …

Rendering: Técnicas de gráficos de computadora que permiten crear imágenes a partir de un modelo 2D o 3D.

Lagspike: Utilizado habitualmente con relación al rendimiento de una conectividad de red, pero también puede aplicarse a tareas de procesamiento de gráficos, audio, … Término que cubre defectos de rendimiento en una aplicación en los que la imagen se congela, la aplicación no responde, etc.

Prompt: En el campo de la inteligencia artificial, es el texto o código que permite a la máquina conocer cuál es el resultado esperado por el usuario.

Sidescroller: Videojuego con vista lateral en el que la cámara sigue al jugador en su movimiento.

Blueprint: Plano o base que sirve para la creación de nuevos elementos en base a unas características definidas previamente.

Built-in: Funcionalidades ya implementadas en una aplicación base.

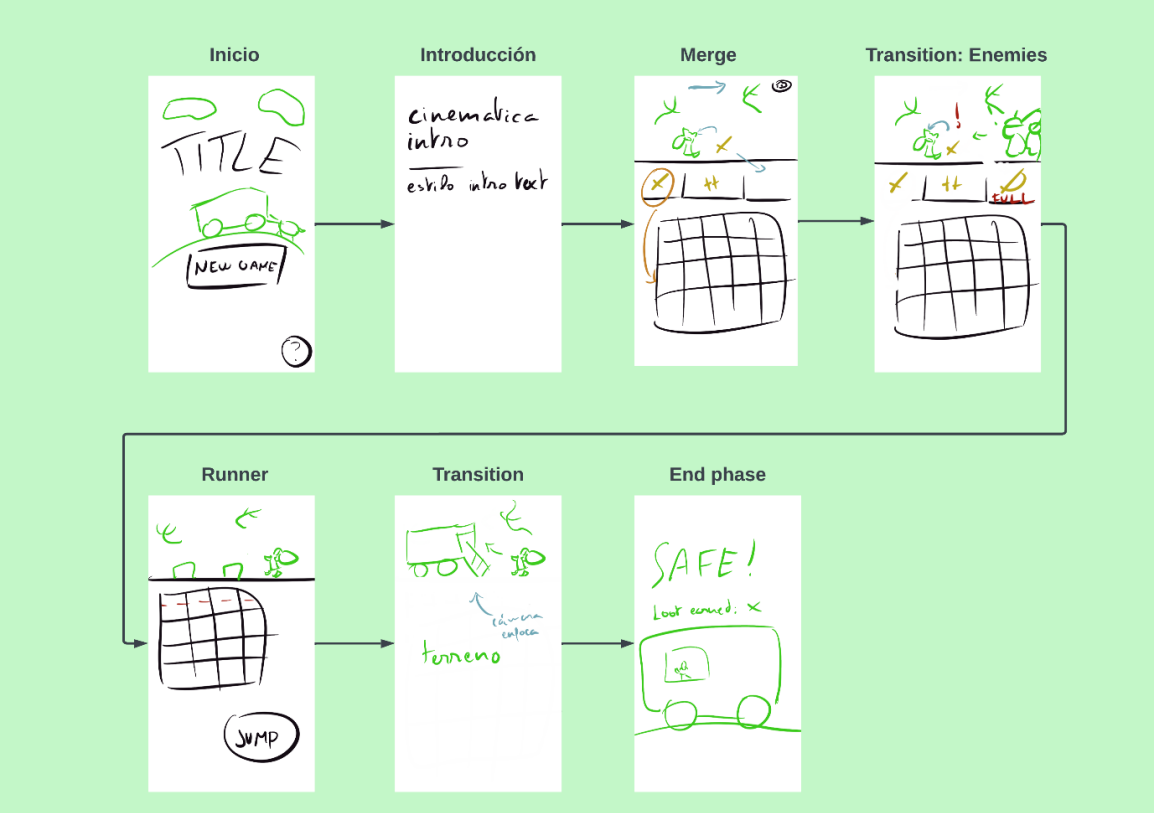
Anexo B: Entregables del proyecto

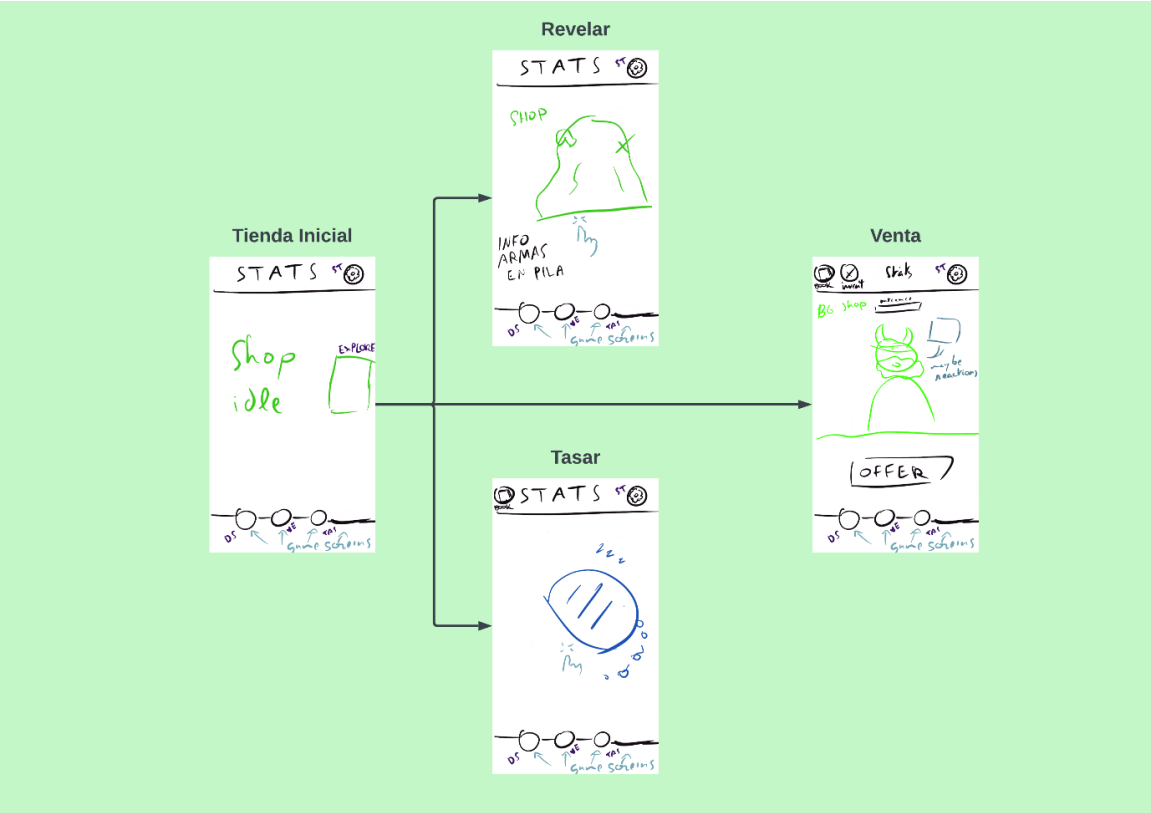
Repositorio GitHub del trabajo: https://github.com/NTCR/goblinaTF

Enlace de descarga Itch.io: https://naitooo.itch.io/la-tienda-de-goblina

Anexo C: Navegación inicial

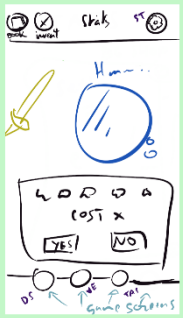
Maqueta del diseño inicial del juego y la navegación entre pantallas.



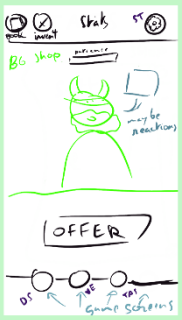


Revelar

Tasar



Venta

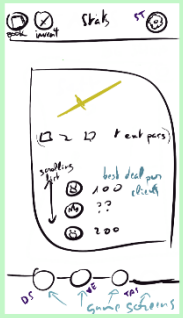


Inventory

Anypoint in shop

Dialogue

Book



Anexo D: Recursos de aprendizaje para Godot Engine

GodotDocs incluye el manual de referencia de la aplicación y números artículos para el desarrollo en GodotEngine.

https://docs.godotengine.org/en/latest/index.html

Foro de preguntas y respuestas de GodotEngine

https://www.gdquest.com/

Portal GDQuest

https://www.gdquest.com/

Herramienta online de aprendizaje del lenguaje propietario GDScript desarrollada por GDQuest.

https://gdquest.github.io/learn-gdscript/?ref=godot-docs

Portal creado por Andrew Wilkes con tutoriales y proyectos basados en GDScript.

https://gdscript.com/tutorials/

Portal de tutoriales gratuitos para GodotEngine.

https://godottutorials.com/

Canales de Youtube con materiales didácticos en el uso de GodotEngine:

PlayWithFurcifer: https://www.youtube.com/c/PlayWithFurcifer

HeartBeast: https://www.youtube.com/@uheartbeast

AlexHoratio: https://www.youtube.com/@AlexHoratio/search?query=godot

AndOne: https://www.youtube.com/@andone3980

FirebelleyGames: https://www.youtube.com/@FirebelleyGames