UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADUADO EN INGENIERÍA DE LA SALUD

Desarrollo de Terra-EVO, un videojuego de evolución de especies y entornos cambiantes.

Development of Terra-EVO, a video game about species evolution and changing environments.

Realizado por

Daniel Gómez Buenestado

Tutorizado por

Antonio J. Fernández Leiva

Departamento

LENGUAJES Y CIENCIAS DE LA COMPUTACIÓN

UNIVERSIDAD DE MÁLAGA

UNIVERSIDAD DE MÁLAGA
MÁLAGA, Agosto de 2025

Fecha defensa: Septiembre de 2025

Resumen

Este proyecto consiste en el desarrollo de un videojuego 2D estilo *sandbox* en Unity, utilizando C# como lenguaje de programación. El objetivo principal es integrar y conectar distintas técnicas de inteligencia artificial, como adaptaciones de algoritmos evolutivos y matrices de influencia, con generación procedural de terreno.

El juego se centrará en la modificación de diferentes parámetros de un planeta (el entorno de juego) para observar cómo se adaptan diversas plantas y animales, así como sus interacciones entre sí. Su elemento central se basa tanto en la inteligencia individual de los miembros de cada especie como en su capacidad de evolución, poniendo a prueba su adaptación a un entorno en constante cambio mediante un sistema de ADN, cruces y mutaciones realistas para aportar mayor inmersión al jugador.

Palabras clave:

Unity

Videojuego

ADN

Inteligencia Artificial

Abstract

This project consists of the development of a 2D sandbox-style video game in Unity, using C# as the programming language. The main objective is to integrate and connect various artificial intelligence techniques, such as adaptations of evolutionary algorithms and influence matrices, with procedural terrain generation.

The game will focus on modifying different parameters of a planet (the game environment) to observe how various plants and animals adapt, as well as their interactions with each other. Its central element is based both on the individual intelligence of the members of each species and on their capacity for evolution, testing their adaptation to a constantly changing environment through a system of DNA, breeding, and realistic mutations to provide greater immersion for the player.

Keywords:

Unity

Video Game

DNA

Artificial Intelligence

Índice

1

Introducción

Para iniciar con el proyecto, se va a redactar una breve introducción que dará a conocer los objetivos, motivaciones y estructura de memoria de este proyecto.

1.1 Motivación

Desde 1947, con la aparición del “Dispositivo de entretenimiento con tubo de rayos catódicos”, considerado el primer juego electrónico, y cinco años más tarde, en 1952, con “OXO”, el primer videojuego de computadora con una pantalla gráfica digital, la industria del videojuego ha experimentado un crecimiento exponencial.

Durante los siguientes años, el avance en la tecnología con hitos tan importantes como la creación de microprocesadores y, posteriormente los ordenadores y consolas domésticas, impulsó la industria del videojuego, haciendo estos más accesibles al público.

La irrupción de internet durante los años 90 abrió la puerta a la creación de los primeros juegos en línea y de las primeras comunidades. Estas interconexiones no solo transformaron la forma de jugar, sino que también facilitaron el acceso a herramientas y conocimiento informático que hasta entonces estaban reservados a unos pocos.

Esta accesibilidad derivó en la creación de la industria "Indie", donde pequeños grupos de desarrolladores, e incluso programadores individuales, comenzaron a crear videojuegos de calidad sin depender de grandes distribuidoras. Gracias a motores gráficos como Unity o Unreal

Engine, bibliotecas de recursos y abundante material de aprendizaje en línea gratuito, el desarrollo de videojuegos se ha vuelto más fácil que nunca.

En este escenario, algunos videojuegos han destacado por ofrecer experiencias abiertas y rejugables gracias a la aleatoriedad y la simulación, como es el caso de *WorldBox*. Según su página oficial de descarga (*www.*superworldbox.com), *WorldBox* se describe como un juego donde "puedes construir tu propio mundo y llenarlo de vida" [13]. Este permite al jugador crear y modificar mundos enteros, poblarlos con diferentes especies, alterar el clima, provocar catástrofes naturales y observar cómo las civilizaciones evolucionan o colapsan con el tiempo. Gran parte de su atractivo reside en que cada partida es única, gracias a un sistema de generación procedural, inteligencia artificial y reglas internas que producen interacciones complejas e impredecibles.

Fue precisamente esta combinación lo que inspiró el nacimiento de Terra-EVO. La idea surgió de una pregunta recurrente: "¿cómo sería la evolución en diferentes mundos, cada uno con condiciones únicas y eventos impredecibles?" A partir de esta curiosidad, se concibió un proyecto que no solo generara entornos de forma procedural, sino que, mediante técnicas de inteligencia artificial, permitiera al jugador crear ecosistemas y verlos evolucionar o colapsar ante cambios.

1.2 Objetivos

El desarrollo del videojuego Terra-EVO ha tenido como objetivo cumplir con diferentes metas.

Primeramente, se ha pretendido crear un juego al estilo "God simulator", con un sistema de guardado que permita el almacenamiento de diferentes partidas o planetas utilizando JSON para esta tarea.

Para continuar, se planeó un sistema de generación de terreno de manera procedural utilizando "Perlin Noise" en una matriz bidimensional de alturas, donde el valor máximo era 10 y el menor -10. Rigiendo así la generación de planetas y los métodos de alteración de terreno que expondremos más adelante.

El siguiente objetivo consistió en que, una vez generado el mundo, el usuario pueda visualizar y modificar una serie de valores planetarios que servirán en futuras ediciones para establecer ciertos parámetros que afectarán a la experiencia de juego.

Una vez establecidas la generación y modificación de variables de mundo, indicaremos las tres dinámicas de juego iniciales y principales del juego.

Para empezar, uno de los primeros objetivos fue una mecánica de alteración de terreno. Aprovechando la matriz bidimensional de alturas, se programó un par de mecánicas donde el jugador puede elevar o bajar el terreno de ciertas formas e intensidades.

Posteriormente, se implementó una adaptación de algoritmos evolutivos como base del sistema de evolución, técnica inspirada en procesos biológicos y ampliamente descrita en la literatura . [1]. De modo similar, se incorporó un sistema de máquinas de estado, un enfoque clásico que permite modelar comportamientos mediante transiciones entre estados.[3] Finalmente, se integraron mapas de influencia, recurso habitual en el desarrollo de videojuegos de estrategia en tiempo real para representar cómo el entorno afecta la toma de decisiones.[4][5]

Nº	**Objetivo**
1	Desarrollar un juego al estilo *God simulator*, incorporando un sistema de guardado que permita almacenar diferentes partidas o planetas mediante el uso de archivos JSON.
2	Implementar un sistema de generación procedural de terreno utilizando *Perlin Noise* en una matriz bidimensional de alturas, con valores de -10 a 10, para la creación de planetas y métodos de alteración de terreno.
3	Permitir que, una vez generado el mundo, el usuario pueda visualizar y modificar una serie de variables planetarias que servirán en versiones futuras para establecer parámetros que afecten la experiencia de juego.
4	Diseñar mecánicas de alteración de terreno basadas en la matriz bidimensional, de forma que el jugador pueda elevar o disminuir el relieve con distintas formas e intensidades.
5	Implementar algoritmos evolutivos como base del sistema evolutivo, junto con un sistema de máquinas de estados y matrices de influencia aplicadas a objetos, con el fin de simular la inteligencia de los seres animales.

TABLA 1. OBJETIVOS DEL DESARROLLO

1.3 Estructura de la memoria

La estructura de la memoria a continuación se divide en diferentes partes:

- Capítulo 2. Antecedentes
  En este capítulo se presentan las tecnologías, herramientas y conceptos teóricos que se han utilizado para comprender el proyecto, incluyendo el motor de desarrollo, el

lenguaje de programación y los fundamentos de algoritmos evolutivos y matrices de influencia.

- Capítulo 3. Análisis y diseño
  Se muestra cómo se desarrolló la idea inicial del videojuego mediante documentos de diseño como el *High Concept Document* y el *Game Design Document*, se detallan las principales mecánicas previstas y se explica la metodología de trabajo utilizada.

- Capítulo 4. Desarrollo
  Se describe el proceso de implementación siguiendo una línea temporal de *sprints*. Se incluyen las pantallas iniciales, el sistema de guardado, la generación de terreno, la selección de parcelas, la interfaz de usuario, las mecánicas de terreno, evolución de plantas y animales, y el sistema de toma de decisiones.

- Capítulo 5. Conclusiones y líneas futuras
  Se exponen las conclusiones alcanzadas tras el desarrollo del videojuego y se señalan posibles mejoras y ampliaciones que podrían implementarse en futuras versiones.

- Referencias y Apéndices
  Para finalizar, en estos apartados, se listarán las referencias utilizadas y se añadirán en los apéndices algunos documentos relacionados con esta memoria.

2

Antecedentes

En este capítulo se darán a conocer los conceptos, herramientas y técnicas utilizadas para realizar el proyecto.

2.1 Motor de desarrollo y lenguaje utilizados

Para poder desarrollar Terra-EVO ha sido necesario apoyarse en una serie de tecnologías y conceptos teóricos que han permitido sentar las bases del proyecto.

Como motor de desarrollo se ha utilizado *Unity*, presentado y lanzado en 2005. Su éxito se atribuye a su accesibilidad y a la facilidad que otorgó a los desarrolladores independientes, quienes anteriormente enfrentaban altos costes y complejidades técnicas con otras herramientas. Esta democratización del desarrollo permitió a Unity convertirse en una de las plataformas más populares en la industria del videojuego.[6] Hoy en día, Unity es uno de los motores de desarrollo más utilizados, junto con *Unreal Engine* y *Godot*.

Se ha escogido Unity como motor debido a su versatilidad y compatibilidad con multitud de programas. Otros factores claves que influyeron en esta decisión son la experiencia previa y la existencia de la tecnología *MLAgents* en este motor, debido a una primera aproximación al *machine learning* para la inteligencia animal del juego.

Unity se caracteriza por utilizar mayoritariamente C# como lenguaje de programación. Este lenguaje, desarrollado por Microsoft en la década de los 90, está orientado a objetos. [7]

Los ML-Agents son un conjunto de herramientas de código abierto que permiten a los desarrolladores integrar técnicas de aprendizaje automático en sus proyectos. Este kit facilita la creación de entornos de simulación donde agentes inteligentes pueden entrenarse utilizando métodos como el aprendizaje por refuerzo y el aprendizaje por imitación. Los agentes entrenados pueden aplicarse en diversos escenarios, como el control de comportamientos de NPCs, pruebas automatizadas de juegos y evaluación de decisiones de diseño antes del lanzamiento.[9][10]

2.2 Otras herramientas utilizadas

Para la creación de elementos gráficos para este proyecto se ha utilizado íntegramente Aseprite, un programa especializado en la creación y edición de arte pixelado.[8]

Otras herramientas como Project, Word y PowerPoint se han utilizado con el fin de agilizar tareas de planificación y presentación del proyecto.

2.3 Conceptos teóricos importantes

Para entender en su totalidad el proyecto realizado tendremos que explicar detenidamente algunos conceptos clave, como son los algoritmos evolutivos y las matrices de influencias.

Los algoritmos evolutivos son técnicas de inteligencia artificial que se basan en el funcionamiento general de la evolución. Mediante procesos de selección natural, cruces y mutaciones, se puede simular en una población un proceso de adaptación que favorezca aquellas características que mejor se adecuen al entorno.

Las matrices de influencia por otro lado, se utilizan como un mecanismo de toma de decisiones, donde una cosa, en este caso un objeto, está asociado a un parámetro que influenciará a una inteligencia artificial a la hora de interactuar con él. Más adelante se explicarán como se han introducido y adaptado estos conceptos.[4][5]

2.4 Metodología de trabajo "Scrum"

Esta metodología es un conjunto de normas, roles y eventos que establecen una estructura de trabajo ágil, que permite desarrollar proyectos complejos. Se basa en espacios de trabajo cortos, llamados *sprints*, que aportan valor al trabajo realizándose de manera iterativa y adaptando el plan de desarrollo a los constantes cambios de manera versátil.[11]

La metodología Scrum establece unos roles que facilitan la organización y comunicación. El *Product Owner* establece los objetivos y prioridades de lo que quiere el cliente, actuando como un "traductor", el Scrum Master acompaña al equipo y asegura que el proceso se lleve a cabo correctamente, y los desarrolladores se encargan de transformar las necesidades en resultados tangibles.[11]

Los *sprints* son las fases en las que se realizan durante el desarrollo, son espacios de trabajo de alrededor de 1 a 4 semanas. Al inicio de cada uno, se lleva a cabo una reunión de planificación llamada *Sprint Planning*, en la que el equipo define qué tareas se van a realizar y cómo se abordarán para alcanzar los objetivos planteados. Durante el Sprint se realizan reuniones diarias o *Daily Scrum*, estas son muy breves y cada uno de los integrantes comparte qué hizo, qué hará y si existe algún impedimento que limite su trabajo.[11]

Al finalizar el Sprint se realizan dos procesos, el primero de ellos es la *Sprint Review* donde el equipo presenta el incremento del producto, es decir, el resultado tangible de ese Sprint, y recibe retroalimentación de los interesados. El segundo de estos procesos es la *Sprint Retrospective*, que se enfoca en analizar cómo ha trabajado el equipo, identificando fortalezas y oportunidades de mejora para el siguiente ciclo, todo este entramado puede verse resumido en la Figura 1.[11]

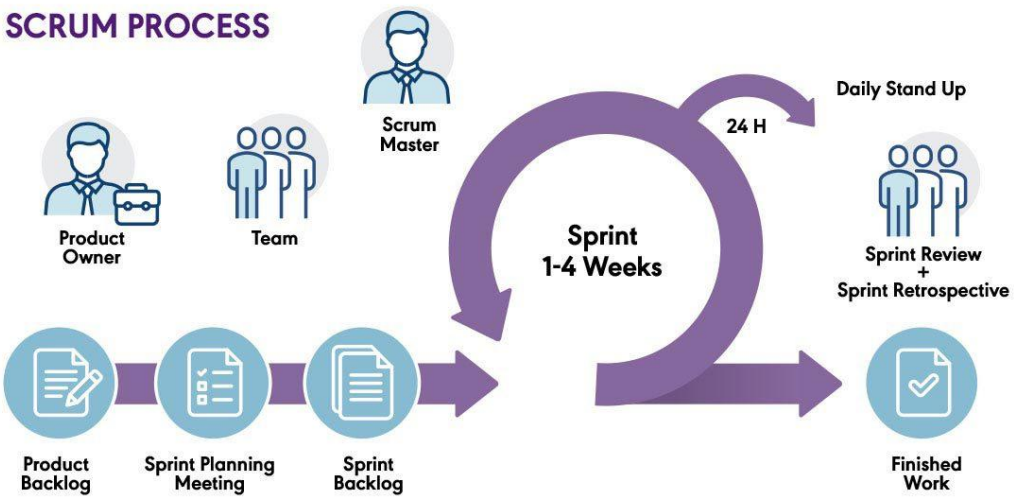

FIGURA 1. METODOLOGÍA SCRUM

3

Análisis y diseño

En este capítulo se muestra cómo se desarrolló la idea inicial del videojuego, se detallarán las principales mecánicas previstas y se explica la metodología de trabajo utilizadas.

3.1 High Concept Document

Se ha decidido empezar la fase de diseño puliendo la idea inicial mediante varias técnicas. Para empezar se realizó un *High Concept Document*. En este documento se resumió el concepto principal del juego, el género, el público objetivo y el estilo visual.

En este documento se ha definido el concepto inicial de Terra-EVO, donde se establece como un juego creativo que intenta simular de manera realista la evolución tanto de las especies como de los ecosistemas, adaptándose a los parámetros impuestos por el jugador.

El *High Concept Document* recoge además el prototipo de la estructura inicial del juego. En un principio se pretendía tener dos modos de juego, el modo historia y el modo creativo. En el modo historia el jugador podría generar únicamente dos especies, una animal y otra vegetal, que irían evolucionando según el entorno, dando lugar a nuevas especies que el jugador podría generar. Mientras tanto en el modo *sandbox*, el jugador podía utilizar todas las especies que haya desbloqueado en cualquiera de las partidas, dándole un control mayor sobre el sistema evolutivo.

Por último, este documento define brevemente algunas mecánicas y el diseño de la interfaz de usuario.

3.2 Game Design Document (GDD)

Con un primer pulido realizado, se ha llevado a cabo un *Game Design Document*, un documento donde se encuentra redactada la idea de manera mucho más detallada, definiendo desde las ideas que dieron lugar al juego hasta las mecánicas, interfaz de usuario y otros aspectos fundamentales para el desarrollo. Aportando la parte más gruesa del conjunto del diseño en el proyecto.

Objetivos

Para empezar se han definido los objetivos y definiciones del juego. Se ha definido Terra-EVO como una simulación de evolución y comportamiento, donde el jugador se convierte en el arquitecto de la vida. Con un enfoque más serio y riguroso que otros juegos del género, el título utiliza *machine learning* para generar un mundo vivo y dinámico donde las especies aprenden de su entorno, evolucionan y se adaptan en un ecosistema en constante cambio.

Se ha establecido como objetivos a largo plazo desarrollar dos tipos de juego, el modo historia y el modo *sandbox*. El modo historia se caracteriza por una progresión estructurada con terraformación, introducción de especies y evolución guiada mientras que el modo *sandbox* permite libertad total para experimentar con el ecosistema y las fuerzas de la naturaleza.

También se han marcado unas directrices base que se tendrán en cuenta a la hora del diseño del proyecto. A continuación se muestran las siguientes:

- Simulación biológica profunda: Las especies no solo mutan, sino que aprenden y se adaptan en función de su entorno y de sus interacciones.
- Generación procedural del mundo: En cada partida se genera un mundo único, que puedes modificar a tu gusto, parcela a parcela.
- Interacción diversificada y entretenida: El jugador puede modificar el entorno mediante fenómenos naturales como tornados, erupciones volcánicas o meteoritos, pero también puede modificar el entorno utilizando herramientas de terraformación y una multitud más.
- Desarrollo de civilizaciones (futuro): La evolución puede llevar a la aparición de seres inteligentes capaces de modificar el terreno y crear sociedades.

Además se establecieron las mecánicas del juego en un proceso tan creativo como ambicioso, por lo que se recalcarán aquellas más importantes, que se han tomado como objetivos:

- **Mecánicas de mundo**, que definen cómo influyen variables como la temperatura, la gravedad o la atmósfera en cada parte del planeta.

- **Mecánicas adaptativas de los animales**, donde la inteligencia artificial permitirá que los individuos aprendan del entorno a través de sentidos como la vista, el olfato o el oído, reaccionando a estímulos y desarrollando comportamientos sociales.

- **Mecánicas de evolución**, que hacen que la aparición de nuevas especies surja de manera emergente, a partir de mutaciones y selección natural, generando incluso "sprites" modulares y únicos para cada criatura.

Más adelante se explicarán todas las mecánicas planteadas para este proyecto.

Por último y con el fin de optimizar el juego, se decidió compartimentar el planeta en diferentes parcelas. Estas parcelas consisten en una zona del planeta interactivo, una a la vez, donde se simulará y se ejecutará todo el juego. El usuario podrá cambiar de parcela y aquellas parcelas no activas tendrán un proceso de simulación reducido con el fin de ofrecer una experiencia optimizada.

Diseño de pantallas

En este documento se detalla además un diagrama de flujo inicial donde se establecen las pantallas que tendrá el proyecto (véase en la Figura 2).

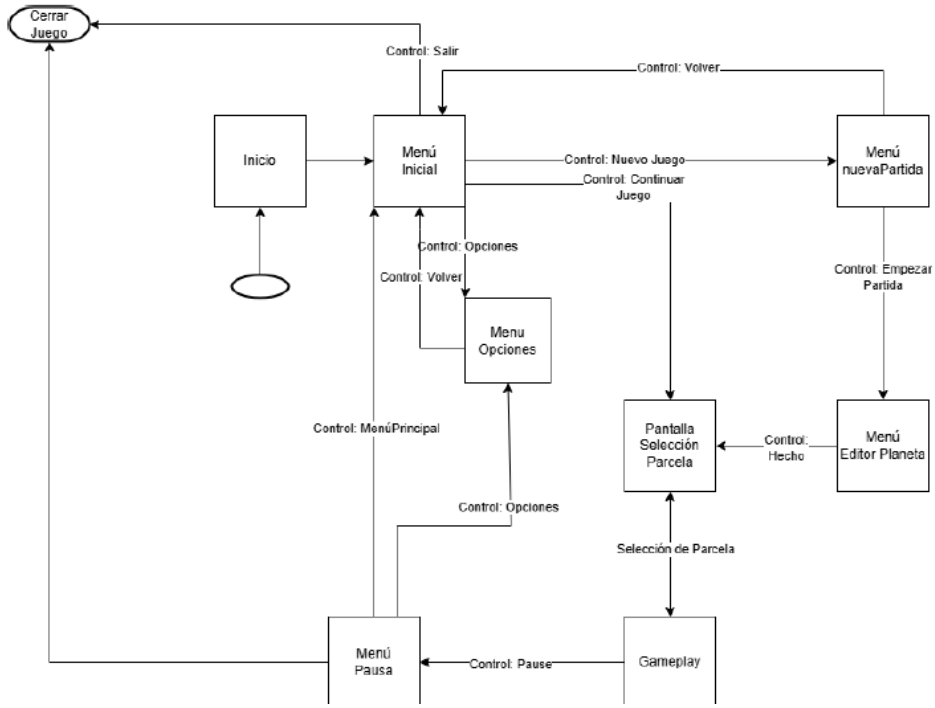

FIGURA 2. DIAGRAMA DE FLUJO INICIAL

Este es un diseño inicial que se cambió durante la fase de desarrollo con el fin de simplificarlo o por decisiones técnicas, este diseño inicial consta de 8 pantallas:

El inicio contaría con una simple animación que daría pie al juego (véase en la Figura 3).

FIGURA 3. BOCETO DEL INICIO

El inicio daría lugar automáticamente al menú inicial, este no tendrá mucha diferencia con el de inicio. Este poseerá espacio hasta para cuatro botones, siendo estos lo de “Nueva Partida”, “Cargar Partida”, “Ajustes” y “Salir del juego”, que llevarían a los menús de “NuevaPartida”, “CargarPartida” y “Ajustes”, respectivamente. El botón de salir del juego cerraría el juego (véase en la Figura 4).

FIGURA 4. BOCETO DEL MENÚ INICIAL

El menú nueva partida se diseñó inicialmente para tener un *TextBox* donde poder colocar el nombre de la partida y el planeta y un botón para confirmar, sin embargo, como se mostrará más adelante se le añadieron los botones de “Modo historia” y “Modo sandbox” para poder distinguir y darle la oportunidad al jugador de seleccionar qué modo de juego quiere disfrutar (tal y como se muestra en la Figura 5).

Al confirmar el nombre de la partida, el juego nos llevaría directamente al menú de “Editor Planetario.

FIGURA 5. BOCETO DEL MENÚ DE EMPEZAR PARTIDA

El menú de editor planetario (véase la Figura 6), en un principio, se decidió que constase de un *ScrollView* con un catálogo de propiedades a cambiar que afectarían a las condiciones del planeta. Inicialmente se incluyeron las siguientes:

- Radio
- Gravedad
- Periodo de rotación
- Temperatura Media
- Nivel del agua
- Oblicuidad planetaria
- Gases atmosféricos, entre los que se incluían:
    - Nitrógeno
    - Oxígeno
    - Dióxido de Carbono
    - Argón

Sin embargo conforme avanzó el desarrollo se decidió sustituir los presentes por:

- Gravedad
- Periodo de rotación
- Temperatura de la estrella
- Distancia con la estrella

- Nivel del agua
- Gases de efecto invernadero
- Oxígeno
- Ozono

En esta nueva versión, la temperatura media, uno de los elementos más importantes a futuro en el juego, se calculará teniendo en cuenta los elementos anteriores. Este sistema de ecuaciones se explicará detalladamente en el apartado 4.2.

Como se explicará en el apartado de desarrollo, durante este, el menú de editor planetario ha sufrido diferentes transformaciones.

FIGURA 6. BOCETO DEL MENÚ DEL EDITOR PLANETARIO

Una vez se han guardado los cambios, este menú debería llevarnos al menú de selección de parcela. En un principio se diseñó utilizando como concepto el mundo con parcelas hexagonales de *RimWorld* un juego de gestión de bases con bastante éxito (véase Figura 8).

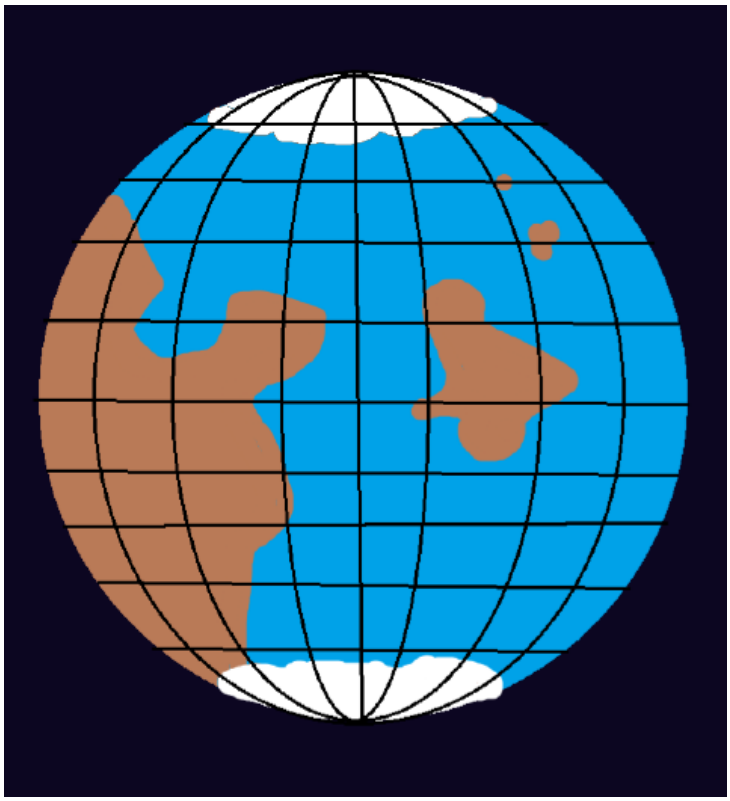

FIGURA 8. BOCETO DEL MENÚ DE SELECCIÓN DE PARCELA

Con este concepto en mente se diseñó un boceto donde el jugador pueda ver su mundo y escoger en qué parcela comenzar su partida, esta decisión se tomó con el propósito de evitar problemas con la capacidad del ordenador al simular un mundo tan grande (véase Figura7).

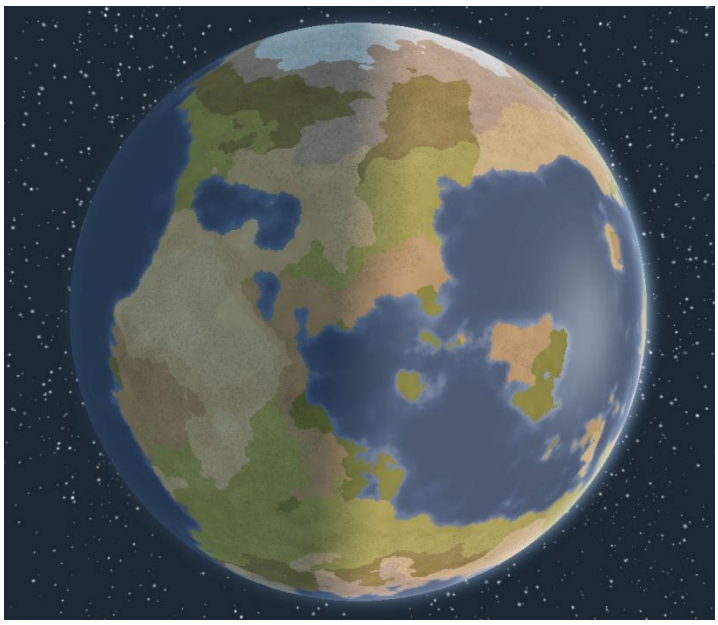

FIGURA 7. IMAGEN DE UN MUNDO CON PARCELAS HEXAGONALES DE RIMWORLD

Una vez fuera seleccionada la parcela, el juego debe llevarnos a la sala de *gameplay*, donde pasará la acción principal, a continuación se muestra un arte conceptual en la Figura 9.

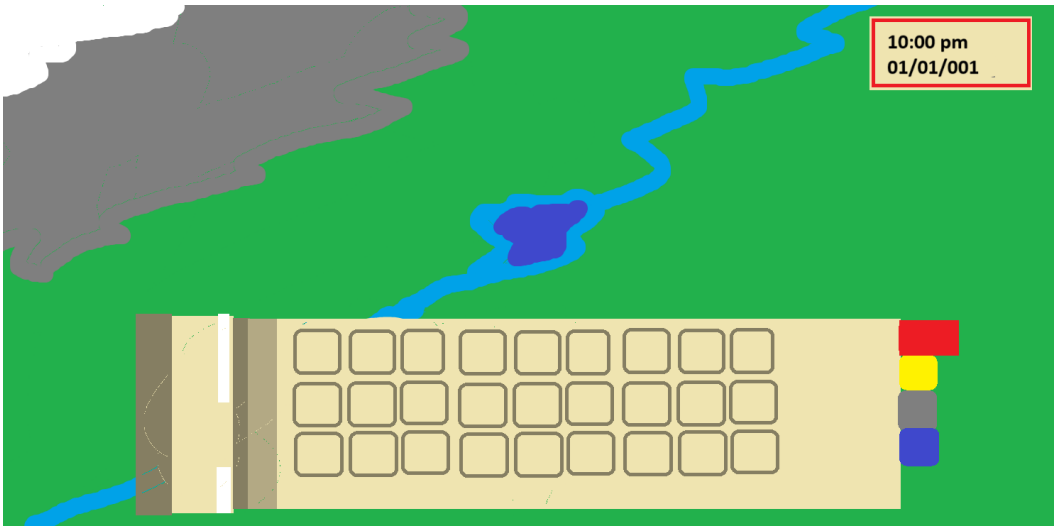

FIGURA 9. ARTE CONCEPTUAL DE LA IU DEL GAMEPLAY

Al ser el resto de pantallas más funcionales y técnicas, como puede ser la pantalla de carga y la de ajustes, no se ha realizado ningún proceso de diseño sobre ellas.

Interfaz de usuario

La interfaz sigue un estilo PixelArt, como se puede ver en la Figura 10, está compuesta principalmente por un pergamino (1*), el cual estará compuesto por diferentes botones (2*) con los que se podrán generar diferentes eventos, especies etc. También podemos apreciar que el pergamino cuenta con tiras de diferentes colores (rojo, azul y verde), estas servirán para "cambiar la hoja del pergamino" y revelar otros botones (2*), acordes con el símbolo de cada tira (La tira roja albergará los desastres naturales y relacionados, la azul tendrá las distintas especies y la verde los ajustes). El apartado 3*, indicará la edad del mundo, y constará de 3 botones: Avanzar (5 días por segundo), pausa, Avanzar rápido(20 días por segundo). El apartado 4* es una propuesta para tener un control de los hitos que cumple el jugador.

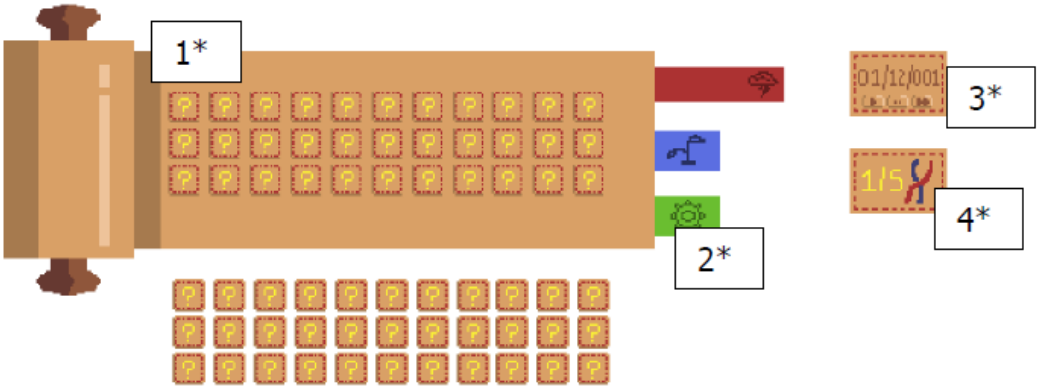

FIGURA 10. ILUSTRACIÓN DE UN BOCETO DE LA IU

Herramientas del jugador

Todas las herramientas del jugador se dividirán en tres grandes menús, para hacer más intuitivo y fácil la búsqueda de estas.

Menú de eventos

En el menú de eventos ,que se puede ver en la Figura 11, se reunirán todos aquellos eventos que impliquen generar un fenómeno natural u objeto, todo aquello que no sea una herramienta o una especie:

FIGURA 11.BOTONES/HERRAMIENTAS DISPONIBLES PARA EL JUGADOR EN EL MENÚ EVENTOS

A continuación, se describirán cada uno de los botones y sus funciones, empezando por el primer botón de la primera fila y acabando en el último de la segunda:

- **Botón Base**: Botón básico, utilizado como plantilla.
- **Botón Volcán**: Generará una erupción volcánica de carácter variable y aleatoria. Los volcanes se medirán en una magnitud del 1 al 10. Mientras mayor sea la magnitud del volcán, mayores serán sus consecuencias. Una erupción volcánica conllevará:
    - Un aumento del efecto invernadero (Temperatura media aumentada)
    - Disminución de la luz solar en la parcela
    - Incendios
    - Terremotos (a partir de magnitud 3 y gradual posteriormente)

        - Desprendimiento de ceniza y lava (a partir de magnitud 2 y gradual posteriormente
- **Botón Meteorito**: Generará un meteorito de tamaño y composición variable. Los meteoritos se medirán en una magnitud del 1 al 10. Mientras mayor sea la magnitud, mayores serán sus consecuencias. El impacto de un meteorito conllevará:
    - Un aumento del efecto invernadero
    - Terremotos
    - Incendios
    - Onda de impacto
- **Botón Lluvia de Fuego**: Algo más informal que el resto, este botón generará nubes que dejarán caer lava o fuego del cielo, provocando incendios.
- **Botón Lluvia Ácida**: Generará una nube que precipitará ácido, haciendo daño a las especies que se encuentran en su camino.
- **Botón Lluvia**: Generará una nube dando lugar a lluvia
- **Botón Maremoto**: Provocará un maremoto (si se activa en una casilla de mar o playa) o una riada (si se activa en una casilla de playa o terrestre con algún río), de diferentes magnitudes (del 1 al 10) aleatorias.
- **Botón Huracán**: Generará un Huracán o tornado de diferentes magnitudes (del 1 al 5), el huracán también generará lluvias y rayos.
- **Botón Terremoto**: Este botón provocará un terremoto en la zona seleccionada, el seísmo, de diferentes magnitudes (del 1 al 10), conllevará:
    - Depresión o elevación del terreno donde se genere
    - Generación de un volcán
    - Caída y muerte de diferentes especies (Plantas, sobre todo)
    - Capacidad de generar maremotos en las parcelas de Mar profundo, Mar poco profundo y playa
- **Botón Fuego**: Todo aquello que el jugador toque mientras este botón esté seleccionado arderá (si algún objeto posee el atributo ignífugo no arderá)
- **Botón Rayo**: Caerá un rayo donde el jugador toque, esto implicará la destrucción de un pequeño perímetro y la ignición de aquellos materiales combustibles que estén en la zona.
- **Botón Granizo**: Generará una nube que arrojará granizo a su paso, el granizo infligirá daño a todas las especies que no estén a cubierto.
  Botón Nieve: Generará una nube que arrojará nieve a su paso, aquel lugar donde caiga nieve generada por el usuario bajará la temperatura local y temporalmente.
- **Botón Enfermedad**: El jugador podrá propagar una enfermedad, esta afectará principalmente a la especie que seleccione el jugador, aunque con el tiempo pueda mutar a otras especies relacionadas con la inicial. Cada enfermedad dispondrá de un nombre y características aleatorias. La enfermedad se regirá por una curva de contagios, y se mantendrá latente una vez que esta haya llegado a su fase final, no provocando infecciones pero sí pudiendo volver a propagarse.

Menú de Herramientas

En el menú de herramientas se reunirán todos aquellos utensilios que permitan al usuario interactuar, modificar y observar el mundo.

FIGURA 12. BOTONES DEL MENÚ DE HERRAMIENTAS

A continuación, se describirá cada uno de los botones y sus funciones, empezando por el primer botón de la primera fila y acabando en el último de la segunda de la Figura 12:

- **Botón de Ajustes**: El botón de ajuste llevará al jugador al menú de pausa o ajuste.
- **Botón Mundo**: El botón de ajuste llevará al jugador a la pantalla de selección de parcela, donde también se podrán modificar los atributos generales del planeta (en Modo *sandbox*).
- **Botón de depresión del terreno**: Solo estará activo cuando el jugador se encuentre en el nivel superficie o nivel subterráneo. Este botón activa un pincel con el que el jugador podrá disminuir la elevación del terreno en el nivel superficie, en el nivel subterráneo podrá crear cavidades.
- **Botón de elevación del terreno**: Solo estará activo cuando el jugador se encuentre en el nivel superficie o nivel subterráneo. Este botón activa un pincel con el que el jugador podrá aumentar la elevación del terreno en el nivel superficie, en el nivel subterráneo podrá rellenar cavidades.
- **Botón de Agua**: Al activar este botón, el usuario podrá generar agua.
- **Botón de Lava**: Al activar este botón, el usuario podrá generar lava.
- **Botón de Selección/Arrastre**: Este botón tiene dos usos principales:
- **Selección**: Con un clic izquierdo sobre una especie, podremos acceder a más información sobre esa especie y ese individuo en específico.
- **Arrastre**: Con un clic izquierdo mantenido podremos mover a un espécimen de un sitio a otro.
- **Botón de inspección**: Si se selecciona una especie en el momento indicado con este botón activado, se abrirá un minijuego o se descubrirá una característica del comportamiento de esta.
- **Botón de Eliminar**: Con este botón se eliminarán entidades
- **Botón Cura**: Con este botón se restaura la vida de las entidades seleccionadas.
- **Botón Linterna**: Mientras este botón esté activado el ratón de jugador brindará luz.
- **Botón de subir Nivel/Capa**: De acuerdo con la mecánica de 3 niveles, este botón servirá para subir de nivel
- **Botón de bajar Nivel/Capa**: De acuerdo con la mecánica de 3 niveles, este botón servirá para bajar de nivel

Menú de especies:

En el menú de especies, siguiendo la temática y dinámica de los anteriores, se mostrarán cada una de las especies que el jugador ha desbloqueado (Modo sandbox), o de las que dispone (Modo Historia), en un botón independiente

Mecánicas

A continuación se muestran las mecánicas que se plantearon para este proyecto:

Mecánicas de Mundo:

Terra-Evo tomará lugar en un planeta con variables cambiantes como las que se pueden ver en la. Estas variables afectarán de una manera u otra a las especies que habitan en este, Radio, Gravedad, Periodo de Rotación, Temp Media, Nivel del Agua, Oblicuidad Planetaria y Gases Planetarios serán los atributos originales. El planeta se generará procedimentalmente y se dividirá en pequeñas "parcelas" que podrán ser de diferente tipo:

- Tierra Firme
- Playa
- Mar poco profundo
- Mar profundo

En cada una de estas parcelas es donde se llevará acabo el grueso del juego, en cada de estas, el jugador podrá crear y evolucionar a su antojo especie utilizando las mecánicas de a continuación. Los atributos locales de las parcelas dependerán de las del planeta y de su posición en este. Los atributos locales que cambiarán según la parcela serán los siguientes:

- Temperatura (Mientras más se acerque a los polos más frío)
- Humedad (Mientras más cercano al mar y más alejado de los polos y el ecuador más humedad)

Las especies podrán colonizar parcelas foráneas, priorizando siempre aquellas parcelas que favorezcan a la especie. Por ejemplo, un animal con pelaje preferirá trasladarse a parcelas más frías que cálidas.

Mecánicas realistas de Fluidos:

Se pretende que los fluidos tengan físicas realistas, y que los fenómenos relacionados con ellos también. Se propone implementar un sistema como el utilizado en el juego "City Skylines" para la simulación el agua.

Mecánica de Erosión de Suelo:

Relacionada con la mecánica realista de fluidos, se pretende que los líquidos puedan erosionar ciertos materiales cuando estos van a altas velocidades y que puedan depositarlos más adelante, en zonas de poca velocidad.

Mecánica de enfermedades:

La mecánica de enfermedades tendrá como objetivo realizar una simulación lo más realista posible de como una enfermedad afecta a una especie o varias. Se pretende que al generar el jugador un evento de enfermedad, o al generarse por sí misma, se cree una enfermedad con

nombres y efectos aleatorios, afectando a una especie en concreto. Esta infectará aquellos no inmunes y dejará a los infectados sobrevivientes con una inmunidad temporal. Esta enfermedad no se eliminará una vez nadie esté contagiado, quedará latente y podrá ser reutilizada al generarse una enfermedad.

Mecánica de Niveles o Capas:

Con el fin de añadir un punto más de complejidad y realismo a Terra Evo se ha decidido dividir el espacio donde se moverán las especies en tres niveles:

Nivel Subterráneo:

En el nivel subterráneo se podrá ver la capa que subyace a la superficie. En el caso de las parcelas de "Tierra Firme", "Playa" y "Mar poco profundo" se podrá ver un conjunto de cavidades con orificios de salida a la superficie o no, también se pretende que haya organismos capaces de crear sus propios túneles. El jugador podrá alterar este especio utilizando la herramienta de elevación o depresión. En el caso de las parcelas de "Mar Profundo" se mostrará al jugador un espacio oscuro donde habitarán criaturas marinas con la característica "Subterránea" que, en este caso, serán criaturas abisales.

FIGURA 13. CONCEPTO DE NIVEL SUBTERRÁNEO PARCELA TIERRA FIRME

En este tipo de parcelas, los seres abisales podrán pasar al nivel superior sin necesitar una salida. En la *Figura 13* podemos ver un arte conceptual de lo que sería el nivel subterráneo. Se pueden ver aquí un conjunto de túneles y grietas, algunas con agua y otras artificiales, se ha pretendido mostrar la entrada de salida de las cavidades indicando con un color más luminoso.

FIGURA 14. DIBUJO CONCEPTUAL DE NIVEL SUBTERRÁNEO EN PARCELA MAR PROFUNDO

Por otro lado, en la *Figura 14* podemos ver otro arte conceptual donde se nos muestra lo que sería el nivel subterráneo en una parcela de Mar Profundo, donde se ha querido hacer especial énfasis en la capacidad que va a tener el jugador de utilizar el botón linterna para ver las especies que se encuentran en las sombras.

Nivel Superficial:

En el nivel superficial se encontrarán el grueso de las criaturas. Estas podrán acceder tanto al nivel subterráneo como al nivel aéreo si cumplen con ciertas características (“Subterráneo”, “Excavador” para el nivel Subterráneo o “Volador” para el nivel aéreo). Según las diferentes parcelas, en el nivel superficial encontraremos:

- Parcela Tierra Firme: Ríos, Montañas, Desiertos, … Lo que sería la superficie.
- Parcela Mar Profundo y Mar Poco Profundo: Aquí se podrán la superficie del agua, es decir, desde especies que habiten la superficie del mar hasta las que habitan a una cierta altura por debajo o en el lecho marino en el caso del Mar Poco Profundo.

- Parcela Playa: Se encontrará una mezcla entre Tierra Firme y Mar Poco Profundo (Figura 15).

FIGURA 15. ARTE CONCEPTUAL DEL NIVEL SUPERFICIE EN UNA PARCELA DE PLAYA

Nivel Aéreo:

En el nivel aéreo se encontrar aquellas especies que puedan volar. Se pretende que este nivel refleje a la lejanía el nivel superficial (Figura 16).

FIGURA 16. ARTE CONCEPTUAL DEL NIVEL AÉREO

Mecánicas adaptativas de los Animales:

El individuo será capaz de reaccionar estímulos y aprender de su entorno utilizando Machine-Learning. Se pretende que la inteligencia artificial pueda asociar objetos a experiencias, es decir, asociar un olor a un sabor, a una experiencia, a un objeto (Visión), etc. También se asociarán diferentes características propias y aleatorias a cada individuo que determinarán su toma de decisiones, ellas son: Valiente, Miedoso, Solitario, Social, Curioso, Cauteloso, que se sumarán a las características propias de cada especie. Se adjunta un esquema del flujo de esta IA en la Figura 17.

La inteligencia artificial tendrá como inputs los sentidos del animal:

- **Olfato**
  El olfato se basará en un “Sphere Collider” que detectará la presencia de otras entidades. Cada entidad tendrá un olor que el “Sphere Collider” detectará, este olor será gradual, aumentando mientras más cerca esté. Se pretende que la IA sea capaz de saber de qué especie o entidad es ese olor dependiendo de si ha interactuado antes con él o no, y su reacción dependerá las experiencias pasadas con este.
- **Gusto**
  Cada entidad comestible poseerá un atributo “Sabor”, cada atributo “Sabor” tendrá asociado un número del +5 al -5, que hará reaccionar positivamente o negativamente a la IA.
- **Tacto**
  Se pretende recoger con el tacto el daño recibido. Mientras más grande sea, más grande será el input a la IA.
- **Visión**
  Para la visión se utilizará un “RayCast”, solo los objetos a cierta distancia (dependiendo del animal) serán detectados.
- **Oído**
  Cada vez que un objeto haga un ruido calculamos la distancia al animal y ajustamos la intensidad del sonido. Será igual que el Olfato, salvo que el olor es un output constante de las entidades y el sonido no.

Los outputs pensados para los animales serán los siguientes:

- **Sonido**
  Se pretende que cada animal tenga una serie de sonidos que emitan en diferentes situaciones posibilitando así que otros individuos puedan asociar sonidos diferentes a eventos diferentes. Por ejemplo, supongamos una situación, en la sabana se encuentran un depredador y dos herbívoros de la misma especie, uno de los herbívoros es atacado por el carnívoro y este emite un sonido de queja (ha recibido un golpe), el otro herbívoro debe saber que ese sonido (si lo ha asociado anteriormente) implica que están atacando a un miembro de su especie.
- **Diferentes Animaciones**
  El individuo tendrá diferentes animaciones o estado, que ayudarán a un observador saber lo que le está pasando.
- **Atacar**
  Provocar daño a algo
- **Ingerir**
  Comer algo.

No se plantea utilizar técnicas de inteligencia artificial en el comportamiento de plantas.

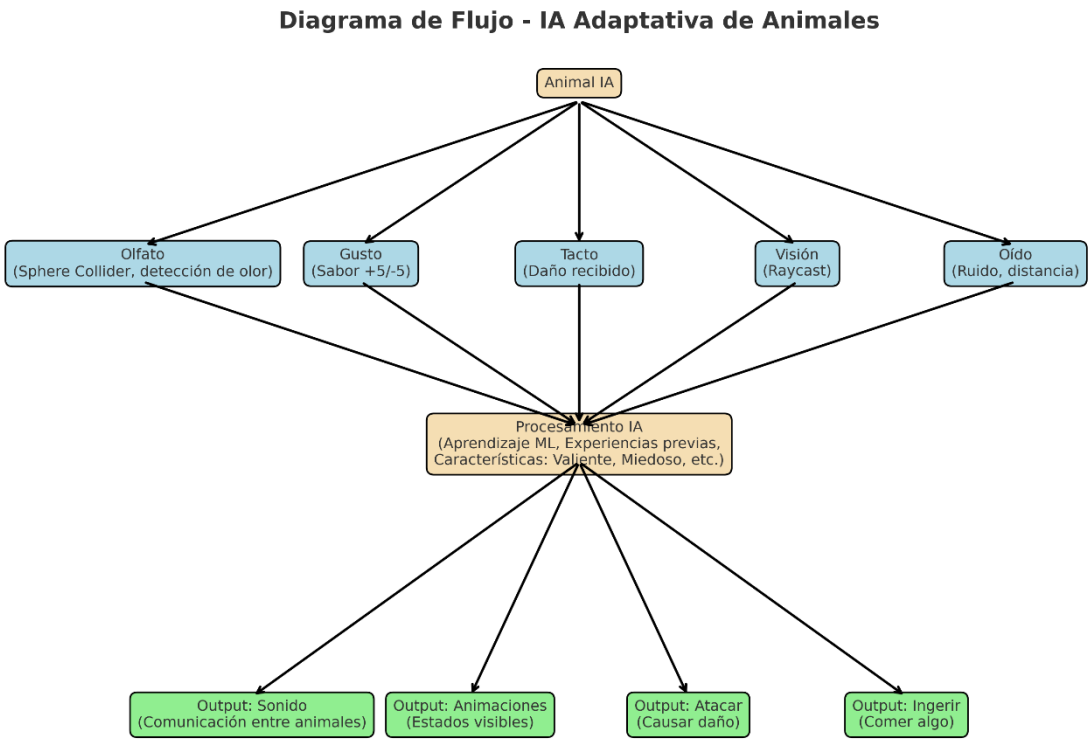

FIGURA 17. DIAGRAMA DE FLUJO IA ADAPTATIVA DE LOS ANIMALES

Mecánicas de Evolución:

En Terra-EVO, la evolución de las especies se basa en un sistema de inteligencia artificial que analiza el desarrollo de los individuos dentro de un ecosistema. La evolución no ocurre de forma predefinida, sino que surge de manera emergente en respuesta a factores ambientales, interacciones y mutaciones aleatorias.

- Principios de la Evolución en Terra-EVO

    - **Mutaciones Individuales**

      Cada individuo dentro de una especie puede sufrir pequeñas mutaciones aleatorias en las cualidades que se definirán más adelante. Estas mutaciones pueden ser beneficiosas, neutras o perjudiciales, dependiendo del entorno de la parcela, el mundo y las interacciones con otras especies.

    - **Selección Natural y Acumulación de Diferencias**

      Si un número significativo de individuos dentro de una misma población ha acumulado suficientes diferencias respecto a su especie original, la IA evalúa si esas variaciones representan una divergencia evolutiva.
      La probabilidad de evolución aumenta si los individuos con mutaciones tienen mayor éxito en la supervivencia y reproducción que los no mutados.

- **Creación de Nuevas Especies**

  Cuando se cumplen los requisitos del apartado anterior, el juego genera una nueva especie derivada.
  Esta nueva especie tendrá características heredadas de su predecesora, pero con adaptaciones que le permitirán sobrevivir mejor en su entorno.
  La IA asignará un comportamiento en función de sus atributos dominantes, el jugador podrá ponerle un nombre "común". Mientras que la IA se encargará del nombre "científico"

- **Ecosistema y Factores de Evolución**

  Factores ambientales como temperatura, disponibilidad de alimento, presencia de depredadores o eventos naturales influirán en qué mutaciones se conservan y cuáles desaparecen, mediante el proceso de selección natural.
  Las especies pueden desarrollar especializaciones, como adaptación a climas extremos, bioluminiscencia en entornos oscuros, camuflaje para evitar depredadores o incluso comportamientos sociales avanzados.

- **Evolución Emergente y No Lineal**

  No todas las mutaciones generan nuevas especies; muchas pueden desaparecer si no representan una ventaja evolutiva.
  Se pueden dar eventos de especiación en paralelo, donde dos poblaciones de la misma especie evolucionan de forma diferente en función de su entorno.

- Generación de la nueva Especie (Apartado Gráfico)

  - **Sistema de Sprites Modulares**

    En lugar de generar un Sprite completamente nuevo, se utilizarán Sprite modulares. La idea es que cada criatura esté formada por piezas intercambiables, como:
    - Cabeza
    - Cuerpo
    - Extremidades
    - Cola o apéndices adicionales

    La IA será capaz de colocar estas partes dependiendo de las características del animal en este caso, y generar cada una de estas por separado, teniendo en cuenta las características gráficas de su sucesor. Esto sucederá en animales

- **Generación Procedimental de Sprites a partir de Patrones**

  Para especies que no necesitan de una animación compleja como pueden ser plantas y hongos, se utilizará esta técnica para generarlos.

- **Cualidades de Especies Animales (Actualizable):**

Atributo	Valores posibles
Número de apéndices	0 - 10
Tamaño	0 - 999
Alimentación	Carnívoro / Herbívoro / Omnívoro
Daño de ataque	0 - 300
Velocidad	0 - 300
Durabilidad	0.01 - 1
Vida	0 - 999
Características	Cuernos, Caparazón, Escamas, Púas, Sordo, Ciego, Visión Nocturna, Pico, Alas, Excavador, Exoesqueleto, Veneno, Tenazas

TABLA 2. ATRIBUTOS DE LAS ESPECIES ANIMALES

Estas variarán aleatoriamente cada vez que nazca un nuevo individuo, teniendo en cuenta los atributos de los padres.

Mecánica de Terraformación:

La terraformación va a depender principalmente de los dos botones de elevación de terreno y depresión de terreno. La elevación del terreno tendrá 21 niveles, siendo un número entero del -10 al 10 siendo del nivel -10 al 0 una capa de sedimentos, con un color más amarillento, el nivel 0 el nivel del mar, del 0 al 6 nivel terrestre y del 7 al 10 el nivel de montaña, distinguiéndose este nivel por un color gris.

Mecánica del Bestiario:

El bestiario es un libro que se asemejará a una “pokedex”, donde el jugador podrá registrar a todos las especies que vaya descubriendo y creando. A medida que el jugador complete minijuegos utilizando el botón de Inspección sobre estas especies se irán desbloqueando en el bestiario tanto las características propias de cada especie (Cualidades de especies animales) como los hábitos característicos y alimentación.

3.3 Metodología (Scrum)

Para la metodología de desarrollo de software se ha seguido una configuración de tipo Scrum, con el fin de conseguir una comunicación directa y clara con la otra parte.

Esta elección se ha visto sesgada debido a lo ambicioso y la dificultad de la idea inicial, por lo que era necesario tener un marco de trabajo ágil para evitar retrasos y dificultades en el desarrollo del videojuego.

Esta manera de trabajar, organizada en *sprints*, ha permitido mantener una visión clara del estado del desarrollo en todo momento, detectar posibles problemas y corregirlos antes de que afectaran al conjunto del proyecto. Además, la planificación en iteraciones facilitó el nacimiento de nuevas ideas y cambios sin necesidad de alterar de manera sustancial el trabajo ya realizado.

Scrum, junto con el uso de herramientas de planificación como Microsoft Project, ha permitido marcar un ritmo de trabajo constante y mantener la idea inicial entre las distintas fases del desarrollo. Cada sprint incluía revisiones periódicas para garantizar que los objetivos se cumplieran y que el proyecto avanzara de manera controlada.

Para facilitar el seguimiento y la coordinación, se elaboró una tabla de reuniones en la que se registraban los encuentros planificados durante cada sprint. Esta tabla incluía:

Reuniones de planificación del sprint: se definían las tareas a realizar y se estimaba el esfuerzo requerido.

Daily stand-ups (reuniones diarias): breves encuentros para reportar avances, identificar obstáculos y coordinar acciones inmediatas.

Revisiones de sprint: al final de cada sprint, se evaluaba el progreso de las tareas, se mostraban los resultados obtenidos y se recogían sugerencias para mejoras o cambios.

Retrospectivas: reuniones para analizar el funcionamiento del equipo, identificar puntos de mejora en la metodología y ajustar la planificación de los siguientes *sprints*.

La tabla permitió tener un registro claro de todas las reuniones, asegurando que cada miembro del equipo conociera los compromisos y el estado de las tareas. Además, facilitó la toma de

decisiones rápidas y la incorporación de nuevas ideas sin afectar negativamente el flujo de trabajo.

Nº	Tipo de reunión	Fecha estimada	Objetivo principal
1	**Sprint Planning 1**	21/04/2025	Definir backlog inicial, alcance del *High Concept Document* y primeras prioridades.
2	**Daily Scrum (semanal con cliente)**	25/04/2025	Revisión breve de avances iniciales del *Game Design Document*.
3	**Sprint Review 1**	01/05/2025	Presentar avances del *Game Design Document* y obtener retroalimentación.
4	**Sprint Planning 2**	05/05/2025	Planificar desarrollo de pantallas y sistema de guardado inicial.
5	**Sprint Review 2**	15/05/2025	Mostrar prototipo de interfaz y guardado básico al cliente.
6	**Sprint Planning 3**	19/05/2025	Definir tareas de generación de terreno y selección de parcelas.
7	**Sprint Review 3**	29/05/2025	Validar mecánica de terreno implementada.
8	**Sprint Planning 4**	02/06/2025	Planear desarrollo de mecánica de elevación de terreno y primer IU funcional.
9	**Sprint Review 4**	13/06/2025	Presentar elevación de terreno y recibir feedback sobre jugabilidad.
10	**Sprint Planning 5**	16/06/2025	Establecer backlog de evolución y crecimiento de plantas.
11	**Sprint Review 5**	03/07/2025	Mostrar crecimiento dinámico de plantas en la simulación.
12	**Sprint Planning 6**	04/07/2025	Planear mejoras del sistema de guardado II y segunda interfaz de usuario.
13	**Sprint Review 6**	16/07/2025	Revisar interfaz II y funcionalidad de guardado.
14	**Sprint Planning 7**	17/07/2025	Planear mecánicas de evolución y aprendizaje animal.
15	**Sprint Review 7**	13/08/2025	Entregar mecánica final de evolución animal y cierre del proyecto.

TABLA 3. TABLA DE REUNIONES CON EL CLIENTE

A continuación se muestra el diagrama de Gantt que se ha seguido, reflejando la distribución de tareas y los *sprints* realizados.

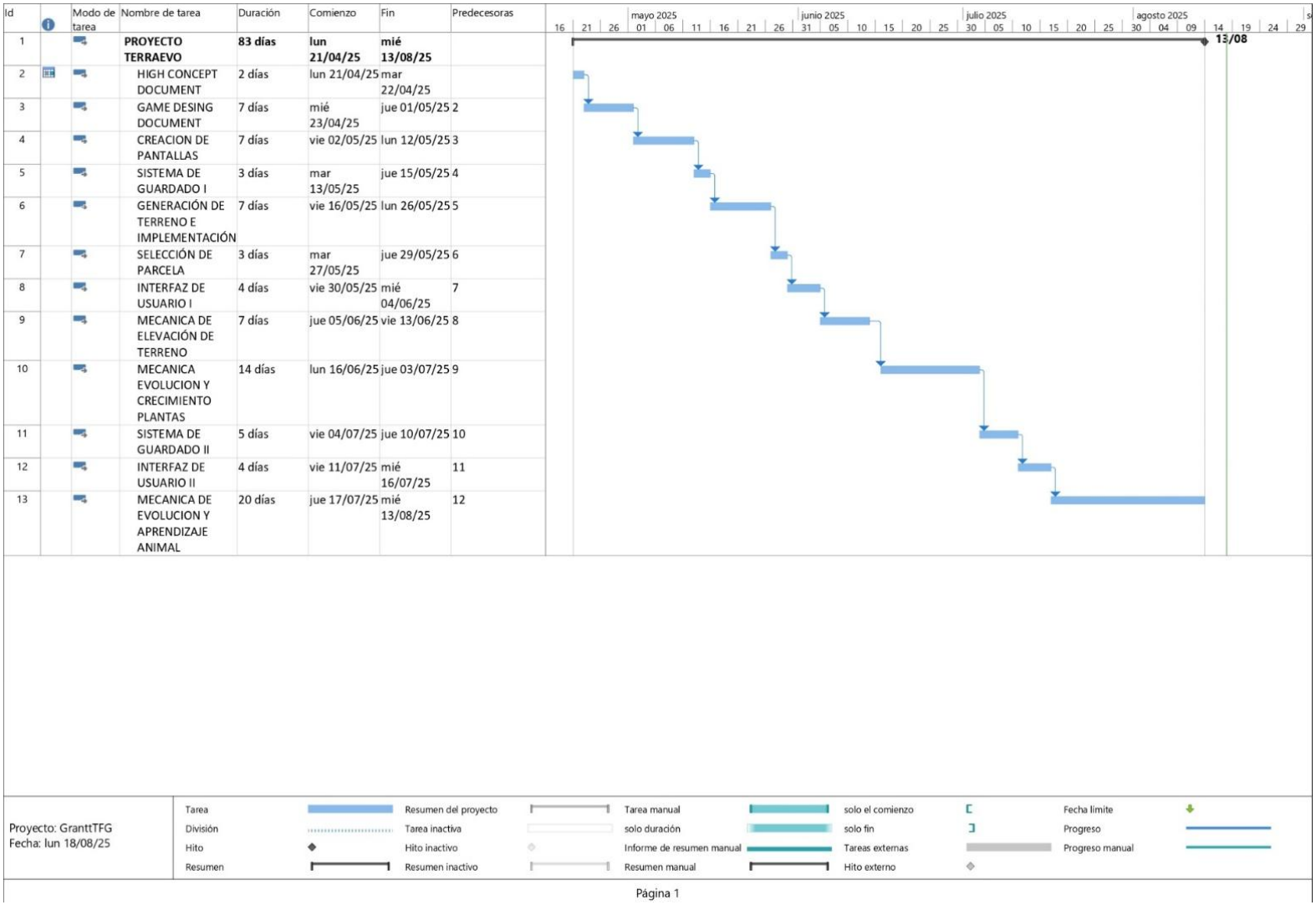

FIGURA 18. DIAGRAMA DE GANTT

4

Desarrollo

En esta sección se detallará paso por paso cada proceso o sprint del desarrollo, explicando tanto como se ha llevado a cabo el proceso, dificultades y resultados de cada uno de ellos.

4.1. Desarrollo de la idea inicial

Durante un periodo de tiempo de 9 días laborables se ha dispuesto un espacio de trabajo para el desarrollo y el pulido de la idea inicial. Desarrollando dos documentos como el *High Concept Document* y el *Game Design Document*, como se explica en el apartado 3.1, se ha conseguido establecer los principales objetivos del juego, mecánicas y otros parámetros. Aunque la idea ha ido transformándose según las fases del desarrollo inicial de la idea, finalmente, con la adecuada tutorización, se ha conseguido amenizar para el trabajo y su tiempo de producción, una idea que, aun tras el proceso de desarrollo, ha seguido siendo bastante ambiciosa.

4.2. Pantallas y prototipo del sistema de guardado

Pantallas

Para empezar a desarrollar las pantallas que nos guiarían por nuestro juego, primero se necesitó un diagrama de flujo, este será una versión algo mejorada del que se realizó inicialmente en el proceso de diseño.

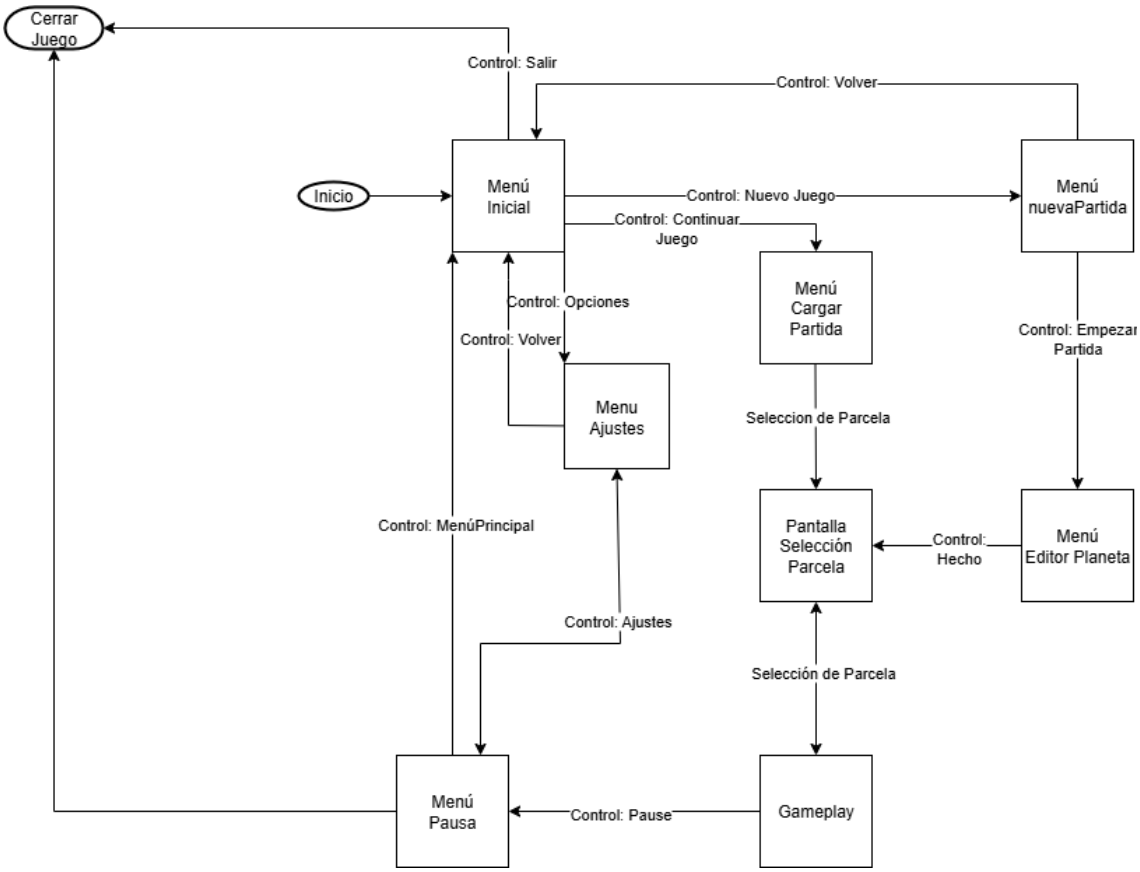

FIGURA 19. DIAGRAMA DE FLUJO FINAL

En la Figura 19 se muestra el diagrama de flujo final utilizado, con alguna pequeña modificación con respecto al GDD. Como se puede observar, se utilizarán un total de 8 pantallas:

- Menú Inicial:

  En esta pantalla se muestra el título del videojuego y bajo este aparecen 3 botones, un botón Nueva Partida, Cargar Partida y Ajustes, que nos llevarán a los menús NuevaPartida, CargarPartida y Opciones respectivamente (Figura 20)

  Esta pantalla se iniciará con una animación inicial antes de que empiece el juego. Más concretamente, el título del juego se desplazará hacia arriba , dejando ver posteriormente los botones indicados, uno por uno.

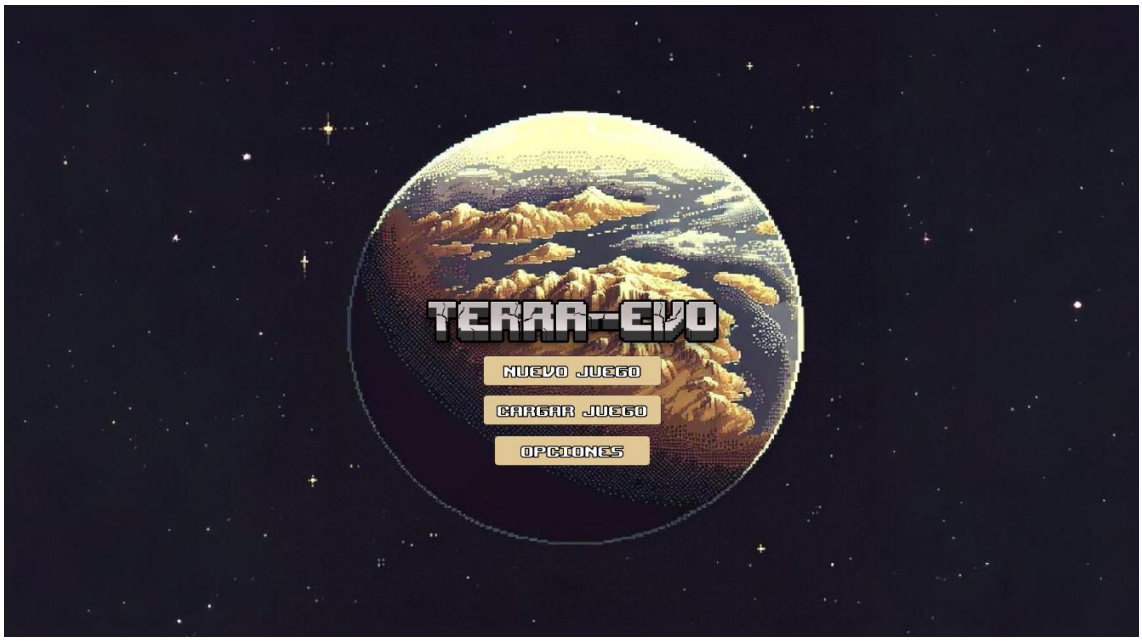

FIGURA 20. IMPLEMENTACIÓN FINAL DEL MENÚ INICIAL

- Menú NuevaPartida:

  En el Menú de Nueva Partida podemos ver un "TextBox" donde podremos introducir el nombre del nuevo mundo o nombre de partida, bajo este, se encontrarán paralelamente dos botones, que nos darán la opción de jugar en modo historia o modo *sandbox*. Una vez seleccionado uno de estos, se habilitará un botón debajo de estos dos, donde podremos empezar el juego, este botón nos llevará al menú EditorPlanetario (Figura 21).

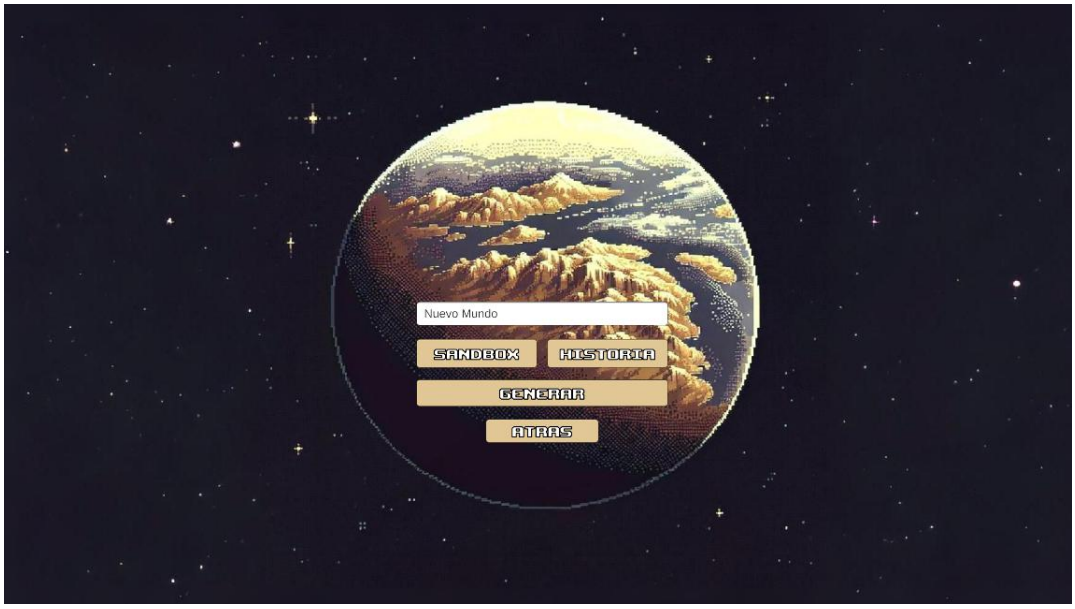

FIGURA 21. BOCETO DEL MENÚ INICIAL

- Menú EditorPlanetario:

  En este menú podremos encontrar una esfera rotando con el mapa generado procedimentalmente a la derecha. A la izquierda encontraremos un "ScrollView" con todas las variables planetarias a cambiar, además del botón Restablecer, que establecerán unos valores similares a los terrestres. Al lado de este botón, podemos encontrar el botón Hecho, que nos llevará al menú SelecciónParcela (Figura 22).

  Estos parámetros afectarán a la evolución de las especies y a la temperatura media del planeta. Podemos observar en la mitad inferior cómo se nos indica este parámetro, que cambiará de manera realista según los datos planetarios.

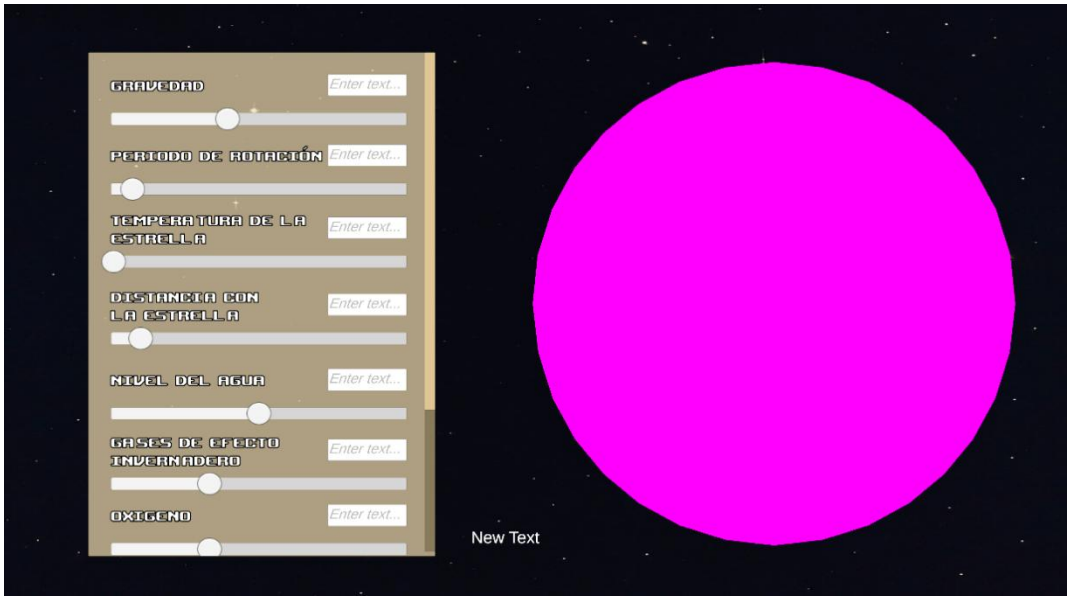

FIGURA 22. PRIMERA IMPLEMENTACIÓN DEL MENÚ DE EDITORPLANETARIO

- Menú SelecciónParcela:

  En el menú de SelecciónParcela se encuentra el mapa generado, esta vez, en un plano. Este plano será dividido en ciertas partes o parcelas, pudiendo elegir entre todas ellas en cuál empezar la partida. Cada una de ellas será un botón, que nos llevará a Gameplay (Figura 23).

FIGURA 23. IMPLEMENTACIÓN INICIAL DEL MENÚ SELECCIONPARCELAS

- Gameplay:

  El Gameplay constará de 3 elementos clave (Figura24).
  - *Tilemap* asociado a un mapa de alturas
  - Elementos sobre el Mapa
  - Interfaz de Usuario

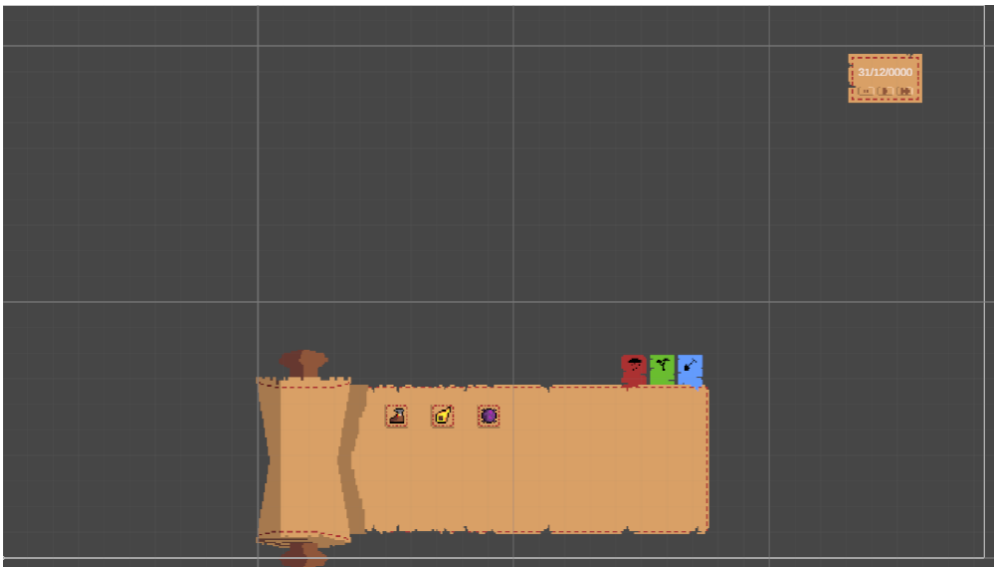

FIGURA 24. PRIMERA IMPLEMENTACIÓN DEL GAMEPLAY

Estos elementos darán lugar a la actividad principal del juegos.

- Menú Pausa:

  Podremos acceder al menú de Pausa mediante un botón en la interfaz de usuario. Este se nos mostrará superpuesto al “gameplay”, y nos dará la opción de guardar partida, cerrar el juego, volver al menú inicial o ir al menú de Opciones . Estas son las opciones planteadas por el momento.

- Menú Ajustes:

  Para el menú de ajustes se tiene pensado algo sencillo con opciones de cambio de sensibilidad y brillo. También podría contar con una opción para activar las notificaciones de todo lo que pasa en el mundo.

  A este menú se podrá acceder desde el menú de pausa o desde el menú de inicio, y siempre, al volver de ese menú, nos llevará al del que procedemos.

- Menú CargarPartida:

  En este menú se puede observar un *ScrollView* con varios *items*, cada *item* corresponderá con un mundo guardado, este mostrará tanto su nombre como su mapa de mundo y un botón de eliminar en caso de querer borrar alguna partida guardada.

Con estos requisitos ya configurados se procedió a la creación de cada una de las pantallas. Establecer las pantallas y añadir todas las funcionalidades iniciales tomó 7 días. La conexión entre estas no supuso gran problema, sin embargo, algunas implementaciones supusieron un contratiempo.

Procesos a destacar en este paso serían:

- Algoritmo de cálculo de temperatura

  Para lograr simular un cambio realista en la temperatura del planeta según los datos dados, primeramente, se ha decidido seleccionar aquellos parámetros que afecten directa o indirectamente a la temperatura. Por lo que, inicialmente, se han descartado los datos de oxígeno, ozono, velocidad de rotación y gravedad.

  Conociendo que las variables que disponemos son:
    - Distancia a la estrella (UA)
    - Gases de efecto invernadero (%)
    - Temperatura de la estrella (K)
    - Nivel del mar (valor entre -10 y 10, siendo 0 el nivel del mar terrestre)

se ha ido deduciendo la fórmula de la temperatura media, inicialmente como:

$$Tmedia = \ Teq * Fi$$

Siendo Tmedia la temperatura media, Teq la temperatura de equilibrio sin atmósfera y Fi el efecto invernadero.

Una vez conociendo de qué dependía la temperatura media de un planeta, se ha ido ampliando esta ecuación tomando algunos supuestos para utilizar como variables los datos que tenemos. Llegando finalmente a esta fórmula:

$$\text{Tfinal} = \text{Testrella} * \sqrt[4]{\frac{1-A}{4}} * \sqrt[2]{\frac{R}{d}} * (1 + 0.01\text{G})$$

donde:

- Testrella = temperatura de la estrella (K)
- R=6.9634×108 m (radio de referencia del Sol)
- d=Distancia a la estrella·1.496×1011 (distancia a la estrella en metros)
- A=albedo=max(0.1,min(0.3−(nivelagua/1000), 0.6))
- G=gases_invernadero (%)

Se ha dado esta fórmula por buena ya que cumple dos requisitos principales, que la temperatura mínima sea -273 °C, y que si ajustamos las variables a las del planeta, nos sale un resultado similar respecto al total.

Sistema de guardado.

Para el sistema de guardado se ha decidido utilizar dos procesos diferentes, uno volátil basado en un "GameDataHolder" y otro no volátil utilizando JSON como guardado. En ambas se ha utilizado una clase “PlayerData”, la cual contiene todos los atributos a guardar. Se ha hecho así para facilitar el guardado de variables y constantes conforme avance el juego. Al empezar esta clase estaba casi vacía e incompleta, teniendo solamente el nombre de guardado de la partida y una variable "mapadealturas" que no es más que un array bidimensional con alturas entre el -10 y el 10, en este caso, uno de prueba, que nos servirá para probar ciertas funciones. En este array basaremos todo el algoritmo de generación procedimental y mecánicas de modificación de terreno.

4.3. Generación de terreno e implementación

Para entender la generación de Terra-EVO debemos entender su base. La base del terreno en este proyecto se basa en un array bidimensional de alturas, donde cada casilla puede tomar un valor entre -10 y 10, siendo 0 el nivel del mar. Este array por defecto tendrá un tamaño de 510x255.

Para la generación de terreno se ha pensado en utilizar "Perlin Noise" (Figura 25) como base para la generación de alturas, debido a que produce patrones suaves y naturales, evitando irregularidades bruscas que darían un aspecto poco realista al mapa.

FIGURA 25. IMAGEN DE "PERLIN NOISE"

Para implementar este ruido se han inicializado los parámetros de ancho, alto y escala del mapa. Estos se han definido como 510, 255 y 10 respectivamente. Estos datos se han definido y han quedado sin cambio desde el principio, salvo por el valor de escala, con el fin de conseguir un mapa lo más realista a una topografía planetaria.

Para mostrar por pantalla el nuevo mapa creado se han asignado distintos colores a diferentes alturas. Siguiendo el modelo donde 0 será el nivel del mar:

- Altura ≤ -4 → Agua profunda (azul oscuro)
- -4 < Altura < 0 → Agua superficial (azul claro)
- Altura = 0 → Arena (amarillo pastel)

- 1 ≤ Altura <5 → Pasto (verde)
- 6 ≤ Altura < 8 → Montaña baja (marrón)
- Altura = 8 → Pico nevado (gris claro)

Para llevar a cabo el renderizado se ha creado una textura 2D vacía y se ha pintado cada pixel con su valor correspondiente en el array bidimensional que actúa como mapa de alturas (Figura 26).

Una vez generada esta imagen, se va a guardar en la carpeta "Worlds" del proyecto, con el nombre que puso el usuario al mundo y su fecha de creación, para evitar duplicados. Este nombre o *path* se guarda en el PlayerData y en la memoria no volátil asignada a cada mundo además de en el GameDataHolder, para poder acceder a ella cada vez que sea necesario.

Para llevar a cabo este proceso se han necesitado dos funciones auxiliares indispensable:

- SaveTextureAsPNG(Texture2D, string):

  Esta función guarda una textura como un PNG con el nombre que ha indicado.

- BorrarImagen(string):

  Esta función comprueba que existe un archivo guardado en el path asignado. Si existe el archivo, se elimina.

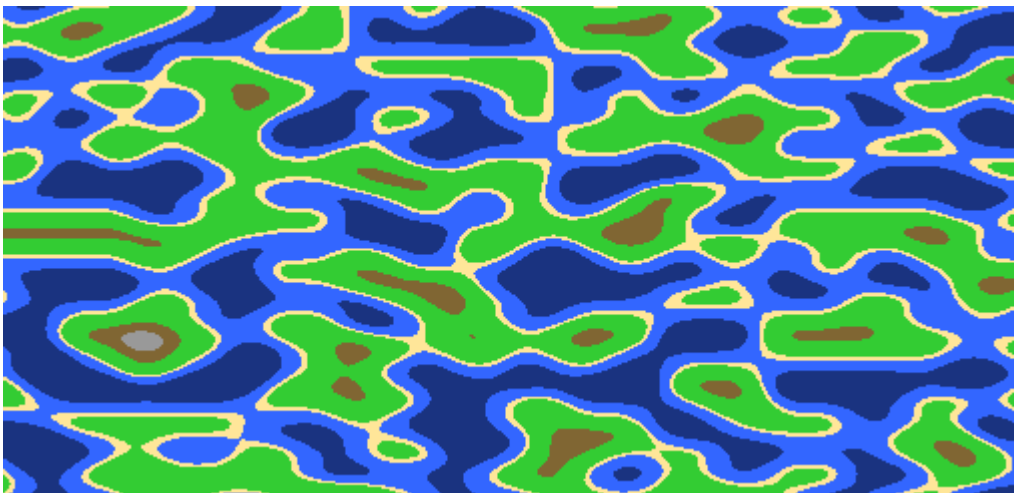

FIGURA 26. MAPA GENERADO UTILIZANDO "PERLIN NOISE"

Con estos sistemas funcionando, fuimos capaces de mejorar el menú de “CargarPartida”, añadiendo el mapa de cada planeta junto a su nombre en cada “ítem” de partida (Figura 27).

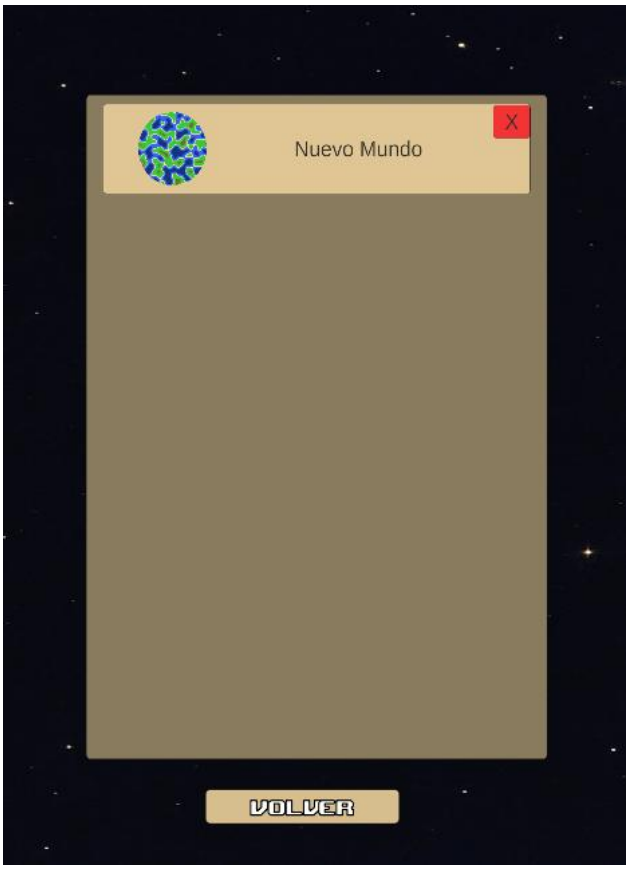

FIGURA 27. MENÚ CARGARPARTIDA

Se intentó añadir este mapa en una esfera, sin embargo, algunos problemas con el formato de la IU no permitieron esto. La solución a este problema fue añadir directamente la imagen con una máscara en forma de círculo.

Además de mejorar el menú de cargar partida, se pudo terminar el menú “EditorPlanetario”, pudiendo añadir esta textura a una esfera rotativa, esta no presentó problemas como la anterior, sin embargo, se nota la "costura" de la unión de los lados opuestos del mapa (Figura 28).

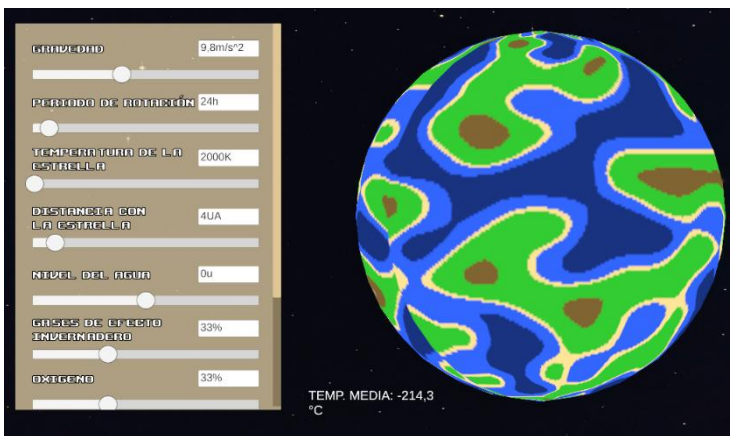

FIGURA 28. MENÚ EDITORPLANETARIO

4.4. Selección de parcela y prototipo de IU

Selección de parcela.

En el siguiente sprint se ha dedicado a completar el menú SelecciónParcela además de generar el mundo en el "gameplay" y hacer un prototipo de la interfaz de usuario (Figura 29).

Se tenía previsto realizar particiones en el mapa general, estas particiones o parcelas permiten correr el juego en una parte concreta de todo el mapa, para que sea menos pesado su procesamiento y simulación, para ello necesitábamos este menú, que nos dará a elegir entre 25 parcelas(5 filas y 5 columnas).

FIGURA 29. MENÚ SELECCIÓNPARCELA

En esta pantalla se recoge la ruta (path) de la imagen del terreno del planeta guardada en la clase "PlayerData" del "GameDataHolder". Una vez obtenida la imagen, se calculan sus dimensiones para, posteriormente, crear 25 texturas independientes extrayendo los píxeles asociados a cada parcela.

Una vez creada la textura, se utiliza un prefabricado, el "prefabparcela", que no es más que un objeto con un "SpriteRenderer" y un BoxCollider2D, el cual se instancia en posiciones fijas de la pantalla. A este objeto se le asignan las texturas extraídas, además de su correspondiente mapa de alturas, que llamaremos mapa de alturas de parcela.

El *prefab*, además, cuenta con un script independiente que le permite iluminarse y elevarse para indicar que se está seleccionando. Una vez que se hace clic sobre él, se actualiza el "GameDataHolder" con el "PlayerData", el cual contendrá el submapa de alturas que hace referencia únicamente al trozo de mundo que comprende la parcela, es decir, el mapa de alturas de parcela, para posterior llevarnos a la escena Gameplay.

Prototipo de IU

El flujo de trabajo llevó directamente a la sala Gameplay, por eso mismo se pensó en iniciar el prototipado de la IU en esta sala. Tras revisar la idea inicial, se decidió dividir la IU en dos partes:

- Pergamino de interacción

  Este pergamino contendrá todas las opciones con las que el jugador puede interactuar con el entorno. Para aumentar el espacio, se han dividido estas herramientas en tres menús.

  - Menú de Ajustes

    El menú de ajustes contendrá un surtido de herramientas ajustes y opciones, como pueden ser las mecánicas de elevación de terreno.

  - Menú de Eventos

    El menú de eventos tendrá implementaciones futuras como la generación de volcanes, lluvia o enfermedades.

  - Menú de Especies

    El menú de especies integrará a todas las especies que se puedan generar y que el jugador haya ido creando.

- Reloj

  Esta pequeña pantalla llevará un contador de los días "in game", además de tres botones para parar, reanudar y aumentar el tiempo. Veremos la implementación de estos más adelante.

Basándose en la idea inicial, utilizando la herramienta Aseprite, se dibujó a mano los menús y los botones que utilizaríamos y que se pueden observar en la Figura 30.

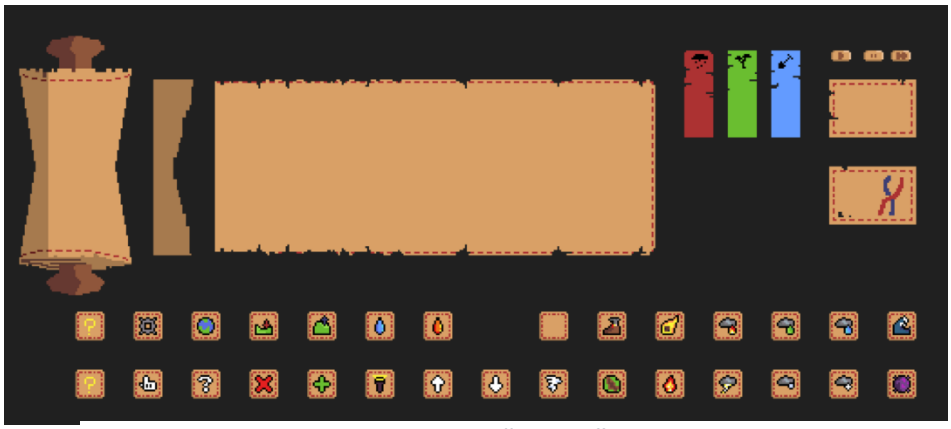

FIGURA 30. "SPRITES" PARA LA IU

En este prototipado solo se colocaron en estos "sprites" y se añadieron funciones básicas como que, para despejar el espacio visual, al clicar en el rollo del pergamino, este baje para dejar ver.

En este sprint se realizó el diseño de los ratones personalizados, también en Aseprite, con el fin de indicar qué función estaba llevando en ese momento el ratón, además de su lógica de cambio y arreglo de bugs (Figura 31).

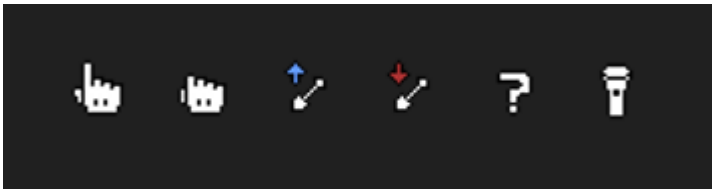

FIGURA 31. IMÁGENES PARA EL RATÓN

Inicialmente se decidió empezar por estos, dando especial importancia a los dos primeros, que serán los ratones por defecto y a los dos del medio, aquellos que representan una pala, porque serán los correspondientes a la mecánica de modificación de terreno que iniciará su desarrollo en el siguiente sprint.

4.5. Mecánica de elevación de terreno

A continuación, se procede a la implementación de la mecánica de elevación de terreno, para ello, necesitamos primero representar el mapa de alturas de la parcela en "gameplay".

Con el fin de representar este mapa de alturas con más detalle que el anterior general, vamos a establecer nuevos colores siguiendo una regla parecida a la anterior:

- Altura ≤ -6 → Agua profunda (azul oscuro)
- -6 < Altura ≤ -3 → Agua (azul)
- -3 < Altura < 0 → Agua superficial (azul claro)
- Altura = 0 → Arena (amarillo pastel)
- 1 ≤ Altura ≤ 2 → Tierra baja
- 2 < Altura ≤ 4 → Tierra media baja
- 4 < Altura ≤ 5 → Tierra media alta
- 5 < Altura ≤ 6 → Tierra alta
- 6 < Altura ≤ 7 → Montaña baja (gris)
- 7 < Altura ≤ 8 → Montaña (gris oscuro)
- Altura = 8 → Pico nevado (blanco)

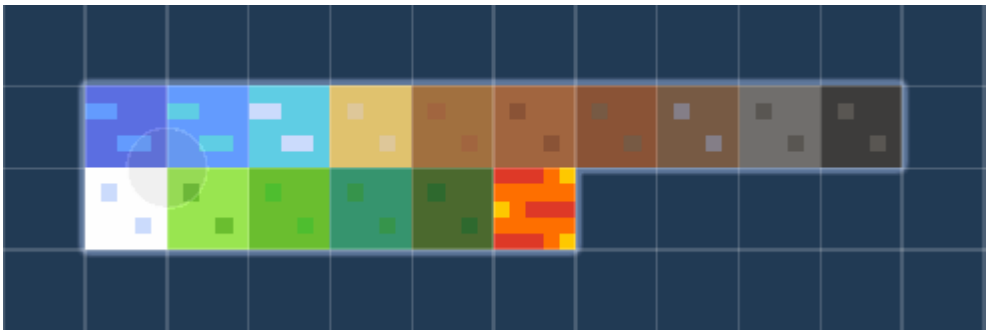

FIGURA 32. IMÁGENES PARA EL *TILEMAP*

A cada rango de alturas se le ha asignado un Tile, como podemos ver en la imagen Figura 32, contando de arriba a la izquierda hacia la derecha, podemos observar cada uno de estos. A partir del Tile blanco correspondiente al "Pico nevado", podemos observar un misceláneo de texturas que se tienen pensado añadir más adelante.

Para presentar este mapa por pantalla se ha creado un *grid* y una función llamada "ActualizarTileVisula(int x, int y, int altura)".Este método permite colocar el tile correspondiente a la altura conociendo solo la posición del *tile* en el *grid* y el valor de su altura. Siguiendo la estandarización anterior, se recoge el *tile* correspondiente a su altura por su nombre y se presenta por pantalla.

Esta función se va a ejecutar una vez con cada altura en el mapa de alturas de la parcela, solo una vez al comenzar el juego. Posteriormente, solo volverá a funcionar si se detecta que se han utilizado las mecánicas de elevación o depresión de terreno. Como aclaración, el mapa de altura de la parcela se ha obtenido gracias a la actualización de la variable "mapadealturasparcela" del “PlayerData” guardado en el “GameDataHolder” en la sala de selección de las parcelas.

Una vez que se ha conseguido representar el mapa de la parcela seleccionada con éxito se decidió empezar a desarrollar la mecánica correspondiente.

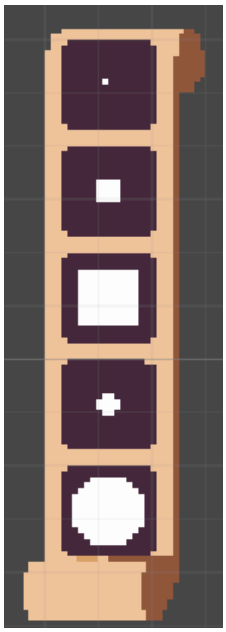

FIGURA 33. SELECCIÓN DE TAMAÑO

La idea inicial para esta funcionalidad era permitir que el jugador adaptara el terreno generado procedimentalmente a su gusto. Sin embargo, al empezar a desarrollarla, se impuso una serie de contratiempos.

El primero de estos contratiempos resultó ser sencillo, el usuario debía saber mínimamente cuál era el tile que estaba a punto de elevar o excavar. Por lo que, para suplir esta necesidad, se dispuso de una función que, conociendo la posición del ratón del jugador, hacía visible un nuevo tile blanco semitransparente que se instanció como deshabilitado encima de todo el mapa, y que solo se habilitaba con la condición anterior, creando el efecto visual de un tile resaltado.

El segundo de estos problemas residió en que cambiar uno por uno los “sprites” podría afectar a la dinámica del juego, por lo que, para ajustarlo, se introdujo un nuevo menú lateral, que se deslizaría hacia la pantalla solo cuando el jugador activase las mecánicas correspondientes. Aclarar que no es lo mismo activar que utilizar, ya que para activarla debes clicar en el botón de la mecánica, y una vez activada, puedes utilizarla sobre el mapa.

Este menú, diseñado también con Aseprite (Figura 33), pretende hacerle más sencillo al jugador los procesos de elevación y excavación, aplicando estos sobre un área más grande y modificable dependiendo de la tarea a hacer.

Una vez activada la mecánica y después de que este menú se haya desplazado hacia el área visible de la pantalla, el jugador podrá seleccionar entre estos 5 botones qué tamaño y forma desea utilizar.

Para llevar esto a cabo se ha realizado para cada botón una matriz correspondiente al tamaño y forma del terreno a modificar. Estas matrices se restarán al mapa de alturas de la parcela y se actualizarán los tiles correspondientes al detectar el cambio.

Con estos y algunos ajustes de menor importancia se ha conseguido implementar la mecánica deseada con éxito.

4.6. Mecánica de evolución y crecimiento vegetal

Esta mecánica de evolución y crecimiento de las plantas es uno de los puntos más importantes del proyecto de Terra-EVO, ya que el eje de este mismo trata de la evolución y el cambio de especies a diferentes entornos, por lo que se ha tratado de completar de la manera más satisfactoria posible.

Para ello se empezó desarrollando el código genético, sus normas y atributos. Este sistema nos permitirá recoger la plenitud de las características de las plantas, tales como:

- La vida de la planta (su resistencia general).
- El tipo de especie vegetal a la que pertenece
- El tamaño y la anchura, que definen su forma y crecimiento físico.
- La capacidad de producir frutos y, en caso de ser frutal, el color de dichos frutos.
- La tasa de crecimiento y la fertilidad, que determinan la velocidad con la que se expande y su capacidad reproductiva.
- Un conjunto de cualidades especiales, como resistencia al fuego, a la sequía, la presencia de espinas, propiedades venenosas, entre otras.

Estas características iniciales fueron transcritas a código binario. Al tratarse de valores numéricos, este paso no supuso un gran problema. Sin embargo, era necesario establecer un conjunto de reglas que permitieran almacenar esta información de forma ordenada y, al mismo tiempo, recuperarla correctamente más adelante.

Para lograrlo, se tomó como referencia el funcionamiento del código genético real, implementando conceptos como iniciadores, terminadores e incluso intrones, con el fin de dotar al sistema de una estructura sólida y coherente.

Con el fin de hacerlo más robusto se le añadió un nuevo componente denominado como "identificadores". Estos identificadores se situarán tras el gen iniciador de la lectura para indicar a qué característica corresponde el número que se va a leer a continuación.

Finalmente, nos quedó un código genético funcional en el 80% de los casos probados (aproximadamente) y con una resistencia a mutaciones efectiva gracias a los intrones.

En la Figura 34 se muestra el código genético binario de una planta.

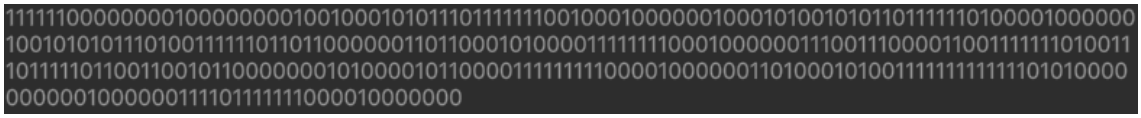

FIGURA 34. ADN BINARIO

Una vez realizadas las pruebas de funcionamiento. Se dispuso a generar el resto de métodos auxiliares necesarios para realizar correctamente la adaptación a los algoritmos evolutivos que teníamos en mente. Para ello se crearán métodos de mutación y cruces lo más biológicamente realistas posible.[12]

Para realizar la función que simule las mutaciones debemos entender qué es una mutación.

Podemos definir mutación como toda alteración permanente en el ADN, según el nivel en el que se vea afectado el ADN, podemos distinguir entre 3 mutaciones:

- Mutaciones puntuales o génicas, que afectan al ADN al nivel más básico, a la secuencia de las bases nitrogenadas en un gen.
- Mutaciones cromosómicas, que son aquellas que afectan a la estructura de un solo cromosoma, alterando grandes cantidades de ADN.
- Mutaciones genómicas, que producen cambios en el número total de cromosomas de un organismo, como ocurre en fenómenos de poliploidía o aneuploidía.

Debido a que el ADN realizado en este proyecto no posee estructuras como los cromosomas debido a su simplicidad, se ha podido simular 4 mutaciones génicas (Figura 35):

- Sustitución de bases: Se invierte un solo bit del ADN, simulando la sustitución de una base por otra.
- Inserción: Se añade un bit aleatorio en una posición específica aleatoria, causando un aumento de longitud en la cadena.

- Eliminación: Se borra un bit de la secuencia, reduciendo su longitud y alterando la información genética.

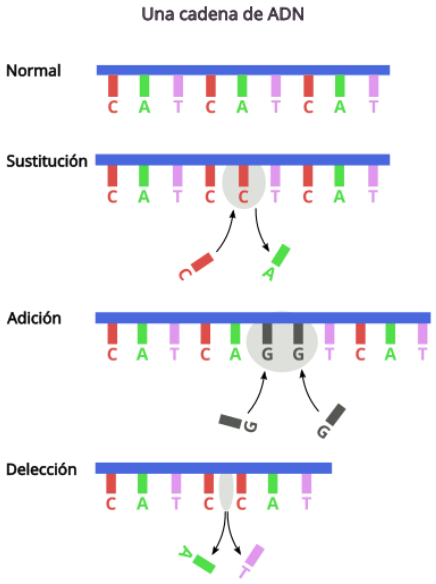

FIGURA 35. MUTACIONES GÉNICAS

Para añadir algún tipo de referencia a las mutaciones cromosómicas (Figura 36), se ha realizado un pequeño guiño añadiendo a la lista de mutaciones las siguientes dos:

- Inversión: Se toma una sección de bits y se invierte. Se pretende simular a pequeña escala una inversión cromosómica donde un fragmento del cromosoma se rompa y se vuelva a insertar en el sentido inverso.

- Duplicación: Se copia una sección de bits y se inserta en la secuencia, simulando la repetición de un fragmento del ADN.

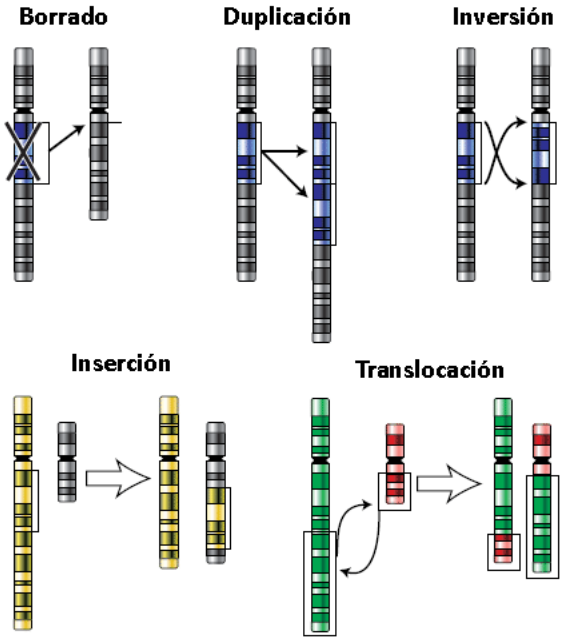

FIGURA 36. MUTACIONES CROMOSÓMICAS

Cada vez que se ejecuta el algoritmo de mutación, la cadena de ADN tiene una probabilidad de un 20% de sufrir una de las 5 mutaciones disponibles.

Para realizar la función de cruce de dos individuos se ha decidido recoger el ADN del padre y de la madre, sacar las características de ambos y seleccionar mediante un método aleatorio qué características obtendrá el nuevo individuo.

La vida de la planta se ha optado por dividirla en dos inicialmente. Un proceso de crecimiento y otro de maduración y reproducción.

El individuo crecerá según las tasas de crecimiento y los límites establecidos por su código genético.

Una vez maduro, dependiendo de su sexo, que será asignado aleatoriamente, si es macho, generará aleatoriamente polen. El polen es un pequeño "prefab" que contiene el código genético del individuo que lo genera, y que cae durante un tiempo predefinido por la altura del generador.

El polen se moverá siguiendo la velocidad y dirección del viento. Se ha generado para ello un algoritmo que genera un vector y va cambiando tanto su magnitud como su ángulo poco a poco.

Por otro lado , si el individuo es hembra, se habilitarán unos espacios llamados slots, el número de estos espacios dependerá de su tasa de fertilidad. Cuando el polen de una planta ajena toca a un árbol hembra con espacios en sus slots se genera una fruta, que va creciendo en la copa de este, con cierto color, va madurando hasta que cae al suelo y tras unos días en el juego se genera un nuevo árbol con los genes cruzados del padre y de la madre.

4.7. Finalización del sistema de guardado y la IU del juego

Para completar la finalización del sistema de guardado, se han arreglado algunos fallos relacionados con la eliminación de mundos y el cargado de partidas, que lo hacían incompatible con los propósitos del juego.

Para finalizar con la IU del juego se han llevado a cabo una serie de implementaciones:

- Implementación del sistema de avance del tiempo, se han sustituido los métodos de las plantas y ,más adelante, animales para que su crecimiento y actividad vayan correspondidos con el tiempo dentro del juego, es decir, que si paramos el tiempo en el juego, estas actividades se paren también.

- Implementación de un tutorial, con el fin de hacer intuitiva y sencilla la experiencia en el juego se ha propuesto un pequeño tutorial que puede ser fácilmente ampliable en futuras versiones (Figura 37).

FIGURA 37. TUTORIAL ACTIVO

Se ha puesto especial interés en el diseño de una pequeña mascota carismática que nos ayude en el principio del juego y se pueda reciclar más adelante.

Basándose en la estructura de un cromosoma, se le ha llamado "CromosoMan". Dándole diferentes diseños a utilizar dependiendo de la situación. Estas imágenes también han sido diseñadas utilizando "Aseprite" (Figura 38).

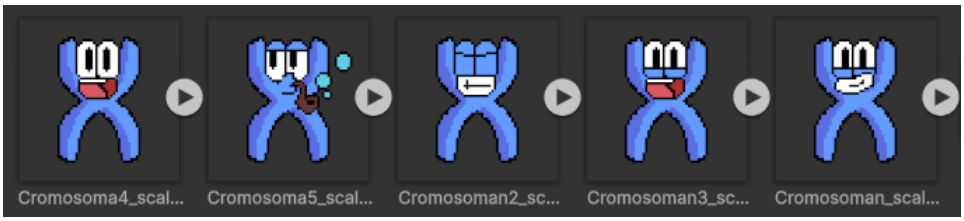

FIGURA 38. IMÁGENES PARA CROMOSOMAN

- Se han añadido los botones de generación de seres vegetales y se ha dejado implementado el botón para los seres animales que serán desarrollados más adelante.

- Se han arreglado algunos fallos relacionados con los interfaces, que no dejaban interactuar con el mundo. Se han detectado y eliminado algunos fallos que provocaban errores en las animaciones de cambio de menú, etc.

- Se ha desarrollado un prototipo del menú de pausa, que pausa el tiempo en el juego y nos da opciones de guardado de partida, volver al menú inicial, cerrar el juego o ir al menú de ajustes.

- También se ha diseñado un prototipo del menú de ajustes para implementaciones futuras.

4.8. Mecánica de evolución, crecimiento y toma de decisiones de animales

Esta última etapa de desarrollo ha sido sin duda la más intensa, ya que se ha llevado a cabo tanto trabajos con algoritmos evolutivos como estudio e implementaciones de diferentes técnicas de inteligencia artificial.

Como ya se ha comentado, la idea inicial para el funcionamiento de los animales era bastante ambiciosa. Se pretendía que, mediante una inteligencia basada en el *machine learning* utilizando *ML Agents*, se crease una máquina que tuviera 5 "inputs" basándose en los 5 sentidos, es decir, gusto, visión, olfato, tacto y oído, creando una simulación realista de las actuaciones del animal. Cada uno de los animales estaba regido por 5 parámetros que afectarían a su comportamiento algunos de ellos, como Valiente o Solitario, querían reflejar la capacidad de un animal para tomar una decisión arriesgada o su capacidad de socialización con seres de su especie, respectivamente.

Para ello se generó un sistema inicial básico donde solo existiera el sentido de la visión, para ir añadiendo progresivamente el resto. Sin embargo, problemas en el entreno del modelo, sumados a la falta de tiempo, llevaron al replanteamiento de todo este sistema.

La nueva mecánica de toma de decisiones se tornó mucho más sencilla. Se planteó una máquina de estado con los 6 procesos básicos del animal. Dormir, Comer, Merodear, Atacar, Reproducirse y Morir (Figura 39). Además se utilizó un sistema basado en matrices de influencias sobre los objetos con los que el animal podría interactuar, esto permitió continuar con el proyecto con un resultado satisfactorio.

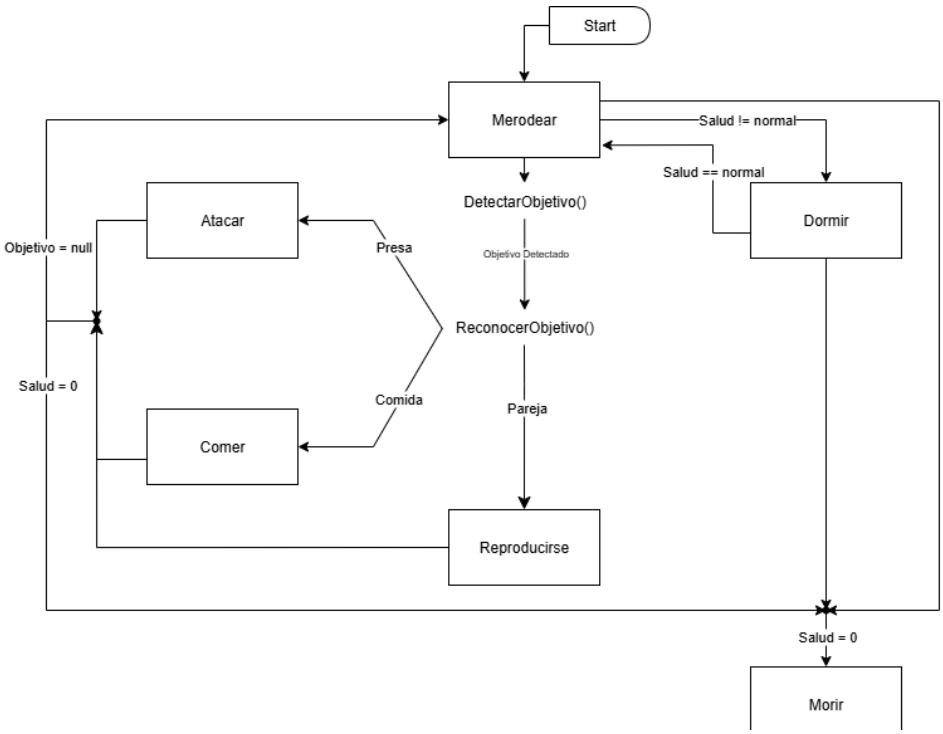

FIGURA 39.REPRESENTACIÓN VISUAL DE LA MÁQUINA DE ESTADOS

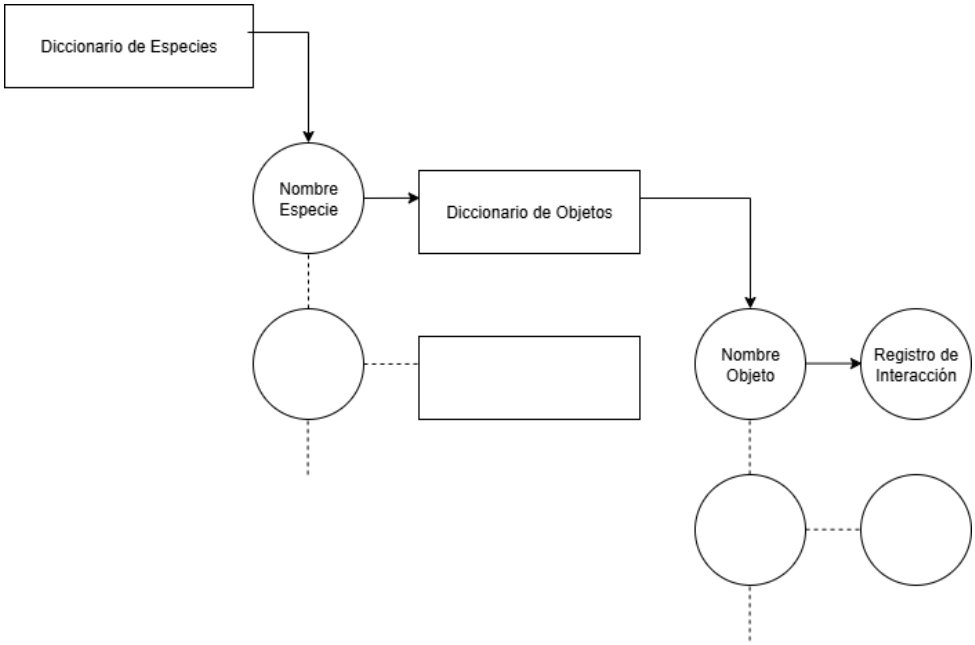

FIGURA 40. IMAGEN REPRESENTATIVA DEL DICCIONARIO DE OBJETOS

Esta matriz de influencias se basa en un diccionario dentro de un diccionario, el cual es identificado por el nombre de la especie, a este diccionario interno lo llamaremos "Diccionario de Objetos", que corresponde con la lista de Objetos con los que ha interactuado la especie que lo identifica. Este Diccionario de Objetos tiene en su interior una clase llamada "RegistroInteracción", identificada por el nombre del objeto al que pertenece. Esta clase posee dos atributos, interacciones y muertes, con lo que, dividiéndolas, podemos sacar fácilmente el porcentaje de muertes por cada interacción, un parámetro clave que ayudará en la toma de decisiones (véase la estructura en la Figura 40).

Cada vez que cualquier individuo de una especie interactúe con un objeto, los valores de "RegistroInteracción" cambiarán, dando a cada especie un tipo de inteligencia colectiva que les ayudará a adaptarse a su entorno.

Una vez desarrollado y resuelta la toma de decisiones, se decidió comenzar con el apartado evolutivo.

Se pudo ahorrar trabajo en el sistema de transcripción de cualidades a ADN binario ya que se utilizará el mismo mecanismo que se utilizó en las plantas, sin embargo, se tuvieron que hacer ajustes mayores para adaptar el código a los nuevos parámetros de los animales.

Estos nuevos parámetros, además de su vida, altura, crecimiento o fertilidad como se hacían con las plantas contarán además con diferentes parámetros como los de la idea inicial para que cada uno de los individuos afronte la toma de decisiones de manera distinta.

Una vez realizados estos procesos vamos a programar el ciclo de vida del animal. Este se debe principalmente a dos cosas, comer y dormir. Si el individuo no come puede morir o ir perdiendo vida, mientras que durmiendo, es capaz de recuperarla. El resto de sus estados dependerán solamente de situaciones y de los parámetros dados al individuo.

5

Conclusiones y Líneas Futuras

A continuación se expondrán las conclusiones tras el desarrollo de este proyecto junto a las posibles mejoras futuras aplicables a este juego.

5.1 Líneas Futuras

Este videojuego, aun en una fase muy avanzada, dispone de un gran potencial. El poder simular la vida con el detalle pretendido puede aportar nuevos puntos de vista sobre la evolución y el comportamiento de especies y concienciar sobre la fragilidad de estos sistemas.

Algunos puntos en los que se pretende avanzar son los siguientes:

- Añadir un método diferenciador y modelos de especies que sean capaces de distinguir cuando un individuo o población deja de formar parte de la especie inicial.
- Implementación de *machine learning* para el sistema de conocimiento de estas especies.
- Implementación de eventos, catástrofes y otros herramientas con los que el jugador pueda interactuar con el entorno.

- Ecosistemas más complejos, añadir capas de simulación como enfermedades, parásitos, dinámicas de fluidos o incluso migraciones de especies, con el fin de hacer más profundo y variado el ciclo de vida del planeta.
- Perfeccionar el sistema de parcelas, cumpliendo con la simulación a menor escala en aquellas que no están siendo jugadas como se propuso en el GDD.
- Mejorar el sistema de ADN, creando estructuras de ADN binario más complejas, como podrían ser los cromosomas.

5.2 Problemas encontrados

Durante el desarrollo de este proyecto han surgido diferentes problemas como se ha ido explicando a lo largo de este documento. Algunos se han podido resolver completamente, se han amortiguado o se ha cambiado el diseño para evitarlo.

Entre todos ellos podemos destacar el uso de los “MLAgents” como inteligencia artificial para darle vida a los componentes animales del juego. Pues tras semanas de desarrollo utilizando y programando estas IAs ocurrieron errores a la hora del entreno de esto. Aunque esto no pareciera un problema muy grave, tras casi una semana de intenso trabajo sin obtener ningún resultado, se decidió utilizar otros tipos de inteligencia artificial con el fin de cumplir con las metas requeridas.

Estos tipos de problemas han surgido de manera continuado, aunque a diferencia del anterior, se han podido solucionar a costa de algunos días de retraso.

Sin embargo, la raíz de todos estos problemas ha sido un fallo en el diseño, ya que el proyecto ha sido muy ambicioso. Aunque se planteó como un proyecto a continuar tras la finalización del Trabajo de Fin de Grado, la magnitud de trabajo esperada ha superado con creces las expectativas.

5.3 Aprendizaje

El aprendizaje que se ha obtenido realizando este proyecto ha sido muy importante. Contextualizando, no se ha tenido experiencia profesional ni de formación previa en el uso de Unity y C# en el ámbito universitario, por lo que la experiencia otorgada, ya no solo en la gestión de proyectos de gran envergadura, sino también en el uso de las tecnologías anteriores, ha sido de vital importancia.

Gracias al desarrollo de este prototipo de videojuego se ha facilitado la adquisición de conocimientos que permiten una mayor facilidad en el uso de otros motores y lenguajes, como Godot o GameMaker. Además, la experiencia obtenida ha posibilitado la participación en

proyectos adicionales, tanto con Unity como con GameMaker, incluyendo la colaboración en iniciativas externas a este trabajo.

5.4 Conclusiones

El desarrollo de Terra-EVO ha cumplido con los objetivos principales planteados al inicio. Se han conseguido integrar técnicas de inteligencia artificial mediante algoritmos evolutivos y máquinas de estados, dando lugar a lo que, con algún tiempo más de desarrollo, podríamos llamar un ecosistema dinámico en el que plantas y animales muestran comportamientos diversos y adaptativos. La evolución de las especies a partir de un ADN binario permite sentar una base sólida sobre la que continuar ampliando la complejidad biológica del sistema.

No hace falta destacar que el proyecto necesita más tiempo para poder pulir, añadir, y perfeccionar ciertas mecánicas para dar el máximo de sí mismo. Sin embargo, con esta implementación inicial se asientan unas bases que pueden traspasar las barreras del entretenimiento.

En el ámbito personal, este proyecto ha abierto mi interés en la industria del videojuego, ofreciéndome las herramientas necesarias para desarrollar juegos de este estilo y parecidos. Además, me ha permitido experimentar con la gestión de un proyecto complejos, enfrentándome a problemas técnicos y de diseño que me aportaron un aprendizaje importante que será de gran utilidad en futuros desarrollos.

De esta forma, Terra-EVO no solo supone un primer acercamiento al diseño de ecosistemas virtuales, sino también el inicio de un camino en el que seguir investigando e innovando dentro del campo del desarrollo de videojuegos. Con más tiempo y dedicación, este prototipo podría evolucionar hacia un producto más completo y ambicioso, capaz de combinar el entretenimiento con la divulgación y la concienciación sobre la fragilidad de los sistemas naturales.

Referencias

[1]. Yusoff, M. H., & Husain, H. (2023). *Evolutionary Algorithm in Game – A Systematic Review*. ResearchGate. https://www.researchgate.net/publication/371251298_Evolutionary_Algorithm_in_Game_-_A_Systematic_Review

[2]. Deepgram. (s. f.). *Evolutionary Algorithms*. Deepgram AI Glossary. https://deepgram.com/ai-glossary/evolutionary-algorithms

[3]. Brooks, R. A. (1986). *A robust layered control system for a mobile robot*. IEEE Journal on Robotics and Automation, 2(1), 14–23. https://people.csail.mit.edu/brooks/papers/how-to-build.pdf

[4]. Hunt, A. (2018). *Influence Maps*. Andrewshunt.com. https://www.andrewshunt.com/influence-maps

[5]. Hunt, A. (2010). *The Core Mechanics of Influence Mapping*. GameDev.net. https://www.gamedev.net/tutorials/programming/artificial-intelligence/the-core-mechanics-of-influence-mapping-r2799/

[6]. Shah, M. (2023). *Unity gaming engine: A case study in platform dominance*. Medium. https://shahmm.medium.com/unity-gaming-engine-a-case-study-in-platform-dominance-962e919b76e6

[7]. Microsoft. (2025). C# programming language. Microsoft. https://learn.microsoft.com/dotnet/csharp/

[8]. Aseprite. (2025). Aseprite: Animated sprite editor & pixel art tool. https://www.aseprite.org/

[9]. Unity ML-Agents. (2025). Machine learning agents toolkit. Unity. https://github.com/Unity-Technologies/ml-agents

[10]. Unity Technologies. (s. f.). *ML-Agents Overview*. https://docs.unity3d.com/Packages/com.unity.ml-agents@3.0/manual/index.html

[11]. Scrum.org. (2025). The Scrum guide. https://scrumguides.org/

[12]. Unitips. (2025). *Tipos de mutaciones*. Unitips. Recuperado de https://blog.unitips.mx/contenido-de-examen-unam-tipos-de-mutaciones

[13]. WorldBox. (2025). WorldBox – God Simulator. Maxim Karpenko. https://www.superworldbox.com/

Apéndice A

Manual de Instalación

Este manual describe los pasos necesarios para la instalación, configuración y correcta ejecución del proyecto desarrollado. Su objetivo es facilitar el proceso tanto para usuarios finales como para “testers”, asegurando que el sistema funcione de acuerdo con los requerimientos especificados.

Requerimientos:

Hardware mínimo recomendado:

- Procesador: Intel i5 / AMD Ryzen 3 o superior
- Memoria RAM: 8 GB
- Almacenamiento: 2 GB libres en disco
- Tarjeta gráfica: Compatible con DirectX 11 / OpenGL 4.5
- Resolución de pantalla: 1280x720 o superior

Software necesario:

- Sistema operativo: Windows 10/11 (64 bits) o equivalente en Linux (Ubuntu 20.04+)
- .NET SDK 8.0 o superior (para ejecución de C#)
- Unity versión 2022.3 LTS (si aplica al proyecto)
- Drivers actualizados de la tarjeta gráfica
- Conexión a internet

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga

E.T.S. DE INGENIERÍA INFORMÁTICA