



UNIVERSIDAD  
**AUSTRAL**

| INGENIERÍA

Trabajo de Grado

*Desarrollo de Videojuego*  
*“Spring Light”*

***Profesora:*** Mariana Falco

***Director:*** Tomás Agustín Fernández Martínez

***Alumno:*** Alejo Ramirez Gismondi

# Índice

<b>Índice</b>	<b>2</b>
<b>Resumen Ejecutivo</b>	<b>5</b>
<b>Introducción</b>	<b>6</b>
Motivación y Objetivo	6
<b>Marco Teórico</b>	<b>8</b>
Breve Historia de los Videojuegos	8
Industria en la actualidad	12
Aplicaciones prácticas de los videojuegos	14
Diferentes Géneros de videojuegos y sus acepciones	16
Puzzle	16
RPG	17
Simulación / Simulator / Sim	17
Farm Simulator (Subgénero)	17
Pixel Art	18
Indie	18
Arte Digital: Pixel Art y Formatos Gráficos	19
Compatibilidad entre Hardwares y Sistemas Operativos	24
Distribución	26
<b>La aplicación</b>	<b>28</b>
Requisitos mínimos de Software	29
Requisitos mínimos de Hardware	29
Marketplace de Assets	29
<b>Tecnologías</b>	<b>30</b>
<b>Unity</b>	<b>30</b>
Cinemachine	31
JetBrains Rider Editor	33
<b>Backend Web</b>	<b>34</b>
<b>Frontend Web</b>	<b>36</b>

<b>Git y GitHub</b>	<b>37</b>
<b>Aseprite y Wacom</b>	<b>39</b>
<b>Arquitectura</b>	<b>40</b>
<b>Investigación</b>	<b>46</b>
Metodologías de Trabajo	46
Etapas/Pipelines	50
Pipeline de Gráficos	51
Pipeline de Animación	51
Pipeline de Scripting	52
Pipeline de Testing	53
Pixel Art	54
Programación Orientada a Objetos	56
Observer Pattern	58
State Pattern	61
Singleton Pattern	62
Strategy Pattern	63
<b>Desarrollo</b>	<b>66</b>
Fase 0: Capacitación	66
Fase 1: Prototipado	74
Inventario: Interactable Objects vs Item Objects	76
CropTiles	80
Puzzles	82
Fase 1.2: Desarrollo de la plataforma web	92
Fase 1.3: Integración de la plataforma web en Unity	94
Fase 1.4: Tests de Carga	102
Fase 2: Integración y pulido	106
Fase 3: Distribución	108
Publicar un juego en Steam	109
Publicar un juego en Itch.io	110
Distribución	111
Plataforma de Distribución	113
<b>Usuarios y Funcionalidades</b>	<b>114</b>
Tipos de usuarios	114
Definición de funcionalidades	115
Especificación de user stories	118

Flujo del juego	138
<b>Interfaz de usuario</b>	<b>151</b>
<b>Dificultades encontradas y soluciones propuestas</b>	<b>167</b>
Cuadro de Diálogo	167
Manejo de Escenas, guardado y serialización	168
Pipeline de Gráficos	170
Generación de Skins de la plataforma web	172
<b>Conclusiones</b>	<b>173</b>
Trabajo con Unity	174
Evolución del Producto	175
<b>Bibliografía</b>	<b>176</b>

## Resumen Ejecutivo

El objetivo del presente trabajo fue presentar una investigación sobre diferentes aspectos del desarrollo de videojuegos y poner a su vez estos conocimientos en práctica para crear un juego propio.

El área de interés en el contenido investigado se enfocó en un principio en la teoría de los videojuegos, su historia y las distintas formas de arte digital que se utilizan en la industria. Además se investigó las diferentes estrategias que se toman a la hora de crear el modelo de objetos de los juegos utilizando patrones de diseño y diferentes metodologías y pipelines de trabajo que se pueden utilizar para mejorar la eficiencia y prolijidad del trabajo realizado.

El desarrollo del juego se llevó a cabo en el motor Unity, complementando con algunas tecnologías de arte digital y para el trabajo con scripts. Se evaluó el proceso y las decisiones que se tomaron, especialmente enfrentando dificultades para las cuales no existían soluciones preparadas, como sucedió en el caso del guardado del progreso, en donde se aplicó un patrón de diseño llamado Strategy para resolver una serialización de objetos compleja que no se podía realizar directamente con las librerías disponibles.

Paralelamente, el desarrollo de la plataforma web se realizó en NestJs y en React. Esta plataforma permite a los usuarios crear sus propias texturas para el personaje del juego e integrarlas a su experiencia.

# 1. Introducción

## 1.1 Motivación y Objetivo

El Trabajo de Grado consiste en realizar una investigación para comprender cómo es el proceso de creación de un videojuego, y la documentación de su proceso de desarrollo tanto en tecnologías como en buenas prácticas de producción y diseño que existan en la industria. Para esto, no solo se investigó, sino que efectivamente se capacitó y desarrolló un juego, utilizando la plataforma Unity<sup>1</sup>, para poner en práctica el conocimiento obtenido.

El videojuego desarrollado es de carácter exclusivamente lúdico, con el objetivo de proveer al jugador con una experiencia inmersiva y divertida. En la industria existen varios juegos de este mismo estilo, que han sido extremadamente exitosos. Ejemplos incluyen las sagas de Pokemon<sup>2</sup>, Stardew Valley<sup>3</sup>, Harvest Moon<sup>4</sup> y Final Fantasy<sup>5</sup>. Adicionalmente, existe un interés personal en investigar e involucrarse en la creación de un videojuego.

Se tomó como inspiración la saga de juegos de Harvest Moon principalmente. Desarrollada originalmente para la Super Nintendo<sup>6</sup> en 1996, fue increíblemente popular tanto para esta plataforma como para las siguientes de Nintendo<sup>7</sup>. Sin embargo, este

---

<sup>1</sup>Unity (motor de videojuego). (n.d.). Wikipedia. Retrieved January 11, 2023, from [https://es.wikipedia.org/wiki/Unity\\_\(motor\\_de\\_videojuego\)](https://es.wikipedia.org/wiki/Unity_(motor_de_videojuego))

<sup>2</sup> Pokémon (serie de videojuegos). (n.d.). Wikipedia. Retrieved February 3, 2023, from [https://es.wikipedia.org/wiki/Pok%C3%A9mon\\_\(serie\\_de\\_videojuegos\)](https://es.wikipedia.org/wiki/Pok%C3%A9mon_(serie_de_videojuegos))

<sup>3</sup> Stardew Valley. (n.d.). Wikipedia. Retrieved January 11, 2023, from [https://es.wikipedia.org/wiki/Stardew\\_Valley](https://es.wikipedia.org/wiki/Stardew_Valley)

<sup>4</sup> Harvest Moon (serie de videojuegos). (n.d.). Wikipedia. Retrieved January 11, 2023, from [https://es.wikipedia.org/wiki/Harvest\\_Moon\\_\(serie\\_de\\_videojuegos\)](https://es.wikipedia.org/wiki/Harvest_Moon_(serie_de_videojuegos))

<sup>5</sup> Final Fantasy (franquicia). (n.d.). Wikipedia. Retrieved February 3, 2023, from [https://es.wikipedia.org/wiki/Final\\_Fantasy\\_\(franquicia\)](https://es.wikipedia.org/wiki/Final_Fantasy_(franquicia))

<sup>6</sup> Super Nintendo. (n.d.). Wikipedia. Retrieved January 11, 2023, from [https://es.wikipedia.org/wiki/Super\\_Nintendo](https://es.wikipedia.org/wiki/Super_Nintendo)

<sup>7</sup> Nintendo. (n.d.). Wikipedia. Retrieved January 11, 2023, from <https://es.wikipedia.org/wiki/Nintendo>

género tremendamente prometedor que combina pixel-art<sup>8</sup>, RPG<sup>9</sup> y simulador de granja no fue explotado particularmente en la industria hasta la llegada de Stardew Valley en 2016, desarrollado para multiplataforma. A partir de este éxito, el género se replicó muchas veces en distintas formas y variantes.

Tomando como referencia estas obras, se tomaron algunos elementos del simulador de granja y se combinó con el género de rompecabezas, para darle una vuelta única distintiva.

---

<sup>8</sup> Pixel art. (n.d.). Wikipedia. Retrieved January 11, 2023, from [https://es.wikipedia.org/wiki/Pixel\\_art](https://es.wikipedia.org/wiki/Pixel_art)

<sup>9</sup> Videojuego de rol. (n.d.). Wikipedia. Retrieved January 11, 2023, from [https://es.wikipedia.org/wiki/Videojuego\\_de\\_rol](https://es.wikipedia.org/wiki/Videojuego_de_rol)

## 2. Marco Teórico

### Breve Historia de los Videojuegos

Se desconoce exactamente cuál fue el primer videojuego creado. Uno de los principales candidatos a dicho título se podría considerar el *Nought and Crosses*, lanzado en 1952 por Alexander S. Douglas. Como se ve en la figura 1.0, este videojuego era una simple representación gráfica del tres en raya, y permitía al usuario jugar contra la máquina. El primer videojuego que tuvo cierto éxito fuera del ámbito universitario fue el Pong<sup>10</sup> en 1972, el cual corría en la consola Atari<sup>11</sup>. Este juego, basado en el deporte del tenis de mesa, consistía de una pelota y dos barras verticales que servían de barreras. Los jugadores debían mover las barras para evitar que la pelota saliera de la cancha. Este juego, junto con el Magnavox Odyssey<sup>12</sup> popularizó el concepto de videojuegos y consolas al público general.

---

<sup>10</sup> Tubb, J. (n.d.). *Pong*. Wikipedia. Retrieved January 11, 2023, from <https://es.wikipedia.org/wiki/Pong>

<sup>11</sup> Tramiel, J. (n.d.). Atari. Wikipedia. Retrieved February 3, 2023, from <https://es.wikipedia.org/wiki/Atari>

<sup>12</sup> Baer, R. (n.d.). Magnavox Odyssey. Wikipedia. Retrieved February 3, 2023, from [https://es.wikipedia.org/wiki/Magnavox\\_Odyssey](https://es.wikipedia.org/wiki/Magnavox_Odyssey)





**Figura 1.0:** Videojuego Pong

**Fuente:** Tubb, J. (n.d.). Pong. Wikipedia. Retrieved January 11, 2023, from <https://es.wikipedia.org/wiki/Pong>

Sin embargo, el verdadero éxito de los videojuegos no llegó hasta 1978 con el lanzamiento del Space Invaders, desarrollado por Toshihiro Nishikado. El Space Invaders (figura 2.0) fue lanzado originalmente en Japón y luego licenciado y lanzado en los Estados Unidos. El Space Invaders era un juego de tipo Arcade, que corría sobre un procesador Intel 8080, muy avanzado para la época, permitiendo el uso de colores y animaciones más complejas.



**Figura 2.0:** Videojuego Space Invaders

**Fuente:** Space Invaders. (n.d.). Wikipedia. Retrieved February 3, 2023, from [https://es.wikipedia.org/wiki/Space\\_Invaders](https://es.wikipedia.org/wiki/Space_Invaders)

Después del lanzamiento de Space Invaders, se produjo un gran aumento en la popularidad de los juegos de arcade y los juegos en general, y se comenzaron a desarrollar y lanzar un gran número de juegos nuevos y empresas de desarrollo de juegos.

A finales de 1970 y principios de 1980, los juegos de arcade alcanzaron su pico de popularidad, y se produjo una explosión en el número de salones de juegos y máquinas de juegos disponibles. Juegos como Pac-Man, Donkey Kong y Centipede también alcanzaron gran popularidad en esta época. A medida que la tecnología avanzaba, las consolas de juegos caseros comenzaron a mejorar y a convertirse en más populares, con la Atari 2600 siendo una de las primeras consolas de juegos caseros en tener éxito comercial.

A finales de los años 80 y principios de los 90, surgieron las primeras consolas de 16 bits, como la Sega Genesis y la Super Nintendo, y los juegos comenzaron a ser más sofisticados en términos de gráficos y sonido. En esta época, también aparecieron los primeros juegos de rol y aventura en consolas y computadoras.

En la década de 1990, los juegos de consolas de 32 bits como la PlayStation de Sony y la Saturn de Sega revolucionaron la industria de los videojuegos con gráficos de mayor calidad y juegos más complejos.

En la actualidad, la industria de los videojuegos ha seguido evolucionando con la aparición de las consolas de videojuegos de nueva generación y la popularidad cada vez mayor de los juegos en línea y móviles. Los videojuegos se convirtieron en una industria global de miles de millones de dólares, con una audiencia global y un gran número de desarrolladores y empresas en todo el mundo.

## Industria en la actualidad

En la actualidad existe una pluralidad de juegos y de compañías que los producen y distribuyen. Muchas de estas compañías ganan e invierten millones de dólares anuales en este tipo de productos. Las compañías de videojuegos más conocidas incluyen:

- Activision Blizzard, Inc. (57.05 mil millones de dólares)<sup>13</sup>
- Electronic Arts Inc. (31.11 mil millones de dólares)<sup>14</sup>
- Tencent Holdings Ltd. (468 mil millones de dólares)<sup>15</sup>
- Nintendo Co., Ltd. (51,5 mil millones de dólares)<sup>16</sup>
- Sony Interactive Entertainment LLC. (114,2 mil millones de dólares)<sup>17</sup>

Las industrias y por lo tanto inversiones son particularmente importantes en Asia, en donde se juegan la mayoría de los juegos competitivos. En Corea del Sur se pronostica un mercado de cerca de veinte mil millones de dólares para el 2023, con un crecimiento esperado del 7,67% anual<sup>18</sup>. Uno de los juegos más populares es el League of Legends, el cual es un juego competitivo desarrollado por la compañía Riot Games<sup>19</sup>, radicada en Corea del Sur. El League of Legends, también conocido por sus siglas como *LOL*, también celebra

---

13

[https://www.google.com/finance/quote/ATVI:NASDAQ?sa=X&ved=2ahUKEwib87TQwlf9AhValJUCHe\\_f\\_B1UQ3ecFegQIORAZ](https://www.google.com/finance/quote/ATVI:NASDAQ?sa=X&ved=2ahUKEwib87TQwlf9AhValJUCHe_f_B1UQ3ecFegQIORAZ)

14

[https://www.google.com/finance/quote/EA:BCBA?sa=X&ved=2ahUKEwjMmO3iwlF9AhXLjZUCHTZiD\\_BkQ3ecFegQIKhAZ](https://www.google.com/finance/quote/EA:BCBA?sa=X&ved=2ahUKEwjMmO3iwlF9AhXLjZUCHTZiD_BkQ3ecFegQIKhAZ)

15

[https://www.google.com/finance/quote/0700:HKG?sa=X&ved=2ahUKEwjkoliAwYf9AhXPqpUCHdewB\\_cIQ3ecFegQIJxAZ](https://www.google.com/finance/quote/0700:HKG?sa=X&ved=2ahUKEwjkoliAwYf9AhXPqpUCHdewB_cIQ3ecFegQIJxAZ)

16

[https://www.google.com/finance/quote/7974:TYO?sa=X&ved=2ahUKEwiY\\_vScwYf9AhUjr5UCHXrW\\_CAYQ3ecFegQIMBAZ](https://www.google.com/finance/quote/7974:TYO?sa=X&ved=2ahUKEwiY_vScwYf9AhUjr5UCHXrW_CAYQ3ecFegQIMBAZ)

17

[https://www.google.com/finance/quote/SONY:BCBA?sa=X&ved=2ahUKEwig4JK1wYf9AhWOrJUCHT\\_UQA20Q3ecFegQILRAZ](https://www.google.com/finance/quote/SONY:BCBA?sa=X&ved=2ahUKEwig4JK1wYf9AhWOrJUCHT_UQA20Q3ecFegQILRAZ)

<sup>18</sup> Video Games - South Korea. (n.d.). Statista. Retrieved February 9, 2023, from <https://www.statista.com/outlook/dmo/digital-media/video-games/south-korea>

<sup>19</sup> Riot Games. (n.d.). Riot Games: Home. Retrieved February 9, 2023, from <https://www.riotgames.com/es>

un torneo mundial anual<sup>20</sup>, en el cual diferentes equipos compiten por la copa y un premio de más de dos millones de dólares<sup>21</sup>.

Paralelamente, en el mercado de los smartphones, los juegos se compran y distribuyen a través de google play store, en el caso de dispositivos móviles Android o en el AppStore, en el caso de dispositivos móviles iOS. Según se informa, el porcentaje de ingresos del AppStore que pertenece a los videojuegos es de más de un 60% de su total de ganancias, incluyendo compras directas y microtransacciones en las apps. En el caso de Google Play Store, este número asciende a más de un 70%<sup>22</sup>.

---

<sup>20</sup> League of Legends World Championship. (n.d.). Wikipedia. Retrieved February 9, 2023, from [https://en.wikipedia.org/wiki/League\\_of\\_Legends\\_World\\_Championship#Trophy](https://en.wikipedia.org/wiki/League_of_Legends_World_Championship#Trophy)

<sup>21</sup> Prize Pool for League of Legends Worlds: Check how much money will be distributed this year. (n.d.). The Economic Times. Retrieved February 9, 2023, from <https://economictimes.indiatimes.com/news/international/us/prize-pool-for-league-of-legends-worlds-check-how-much-money-will-be-distributed-this-year/articleshow/94541119.cms>

<sup>22</sup> Home App Data Mobile Games Revenue Data (2023). (2023, January 9). Business of Apps. Retrieved February 9, 2023, from <https://www.businessofapps.com/data/mobile-games-revenue/>  
Báez, F., Burlando, F., & Franco, B. (2013, December 21). Las aplicaciones "freemium" son las más rentables en la AppStore. Infobae. Retrieved February 9, 2023, from <https://www.infobae.com/2013/12/21/1532555-las-aplicaciones-freemium-son-las-mas-rentables-la-appstore/>

Partis, D. (2021, May 19). 62% of all App Store revenue is generated from game transactions. GamesIndustry.biz. Retrieved February 9, 2023, from <https://www.gamesindustry.biz/62-percent-of-all-app-store-revenue-is-generated-from-game-transactions>

## Aplicaciones prácticas de los videojuegos

Además de sus conocidas aplicaciones lúdicas, los videojuegos también tienen muchas aplicaciones prácticas que se utilizan para mejorar la motivación y el fácil acceso de los usuarios al realizar ciertas actividades.

La gamificación se refiere a la aplicación de técnicas y mecanismos de juego en entornos no lúdicos con el objetivo de motivar y enganchar a los usuarios para que realicen tareas o acciones deseadas. La gamificación se utiliza comúnmente en aplicaciones móviles, sitios web y programas de marketing para mejorar la participación y retener a los usuarios. También se utiliza en el aprendizaje, la formación y la productividad, aplicando técnicas de juego para hacer que la experiencia sea más divertida y atractiva.

Por ejemplo, los juegos se han utilizado en el ámbito de la educación para mejorar la motivación y el aprendizaje de los estudiantes. Esto se ha estudiado a lo largo de la historia por diversas ramas de la ciencia, por ejemplo la psicología a la mano de Jean Piaget o Lev Vygotsky<sup>23</sup>. El uso de juegos aumenta las posibilidades de iniciar y persistir en las actividades, así como el rendimiento académico, la capacidad de atención y la motivación extrínseca e intrínseca. Esto es especialmente atractivo para los niños que tienen alguna deficiencia cognitiva, problemas de aprendizaje o dificultades para seguir el esquema de educación convencional, como podrían ser personas con trastornos del espectro autista.

Otras aplicaciones abarcan desde el entrenamiento militar, para mejorar la preparación y el rendimiento de los soldados hasta la investigación médica para simular y analizar procesos biológicos complejos, y en la terapia para ayudar a pacientes con problemas de salud mental o físicos. Por último, también se han visto aplicaciones en el área de negocios, para simular escenarios, y en el entrenamiento de personas mayores.

Todas estas actividades y los productos que se desarrollan para servir a estos mercados atraen mucho a la inversión por parte de instituciones que buscan resolver estos problemas complicados y específicos.

---

<sup>23</sup> Aprendizaje basado en juegos. (n.d.). Wikipedia. Retrieved February 9, 2023, from [https://es.wikipedia.org/wiki/Aprendizaje\\_basado\\_en\\_juegos](https://es.wikipedia.org/wiki/Aprendizaje_basado_en_juegos)

Como se puede ver, el mercado de los videojuegos es un mercado en crecimiento y muy lucrativo. Según estimaciones, el tamaño del mercado de los videojuegos es de varios miles de millones de dólares anualmente, y se espera que continúe creciendo en el futuro. Esto se debe además a que la industria de los videojuegos abarca una amplia gama de plataformas, incluyendo consolas de juegos, PC, dispositivos móviles y realidad virtual, lo que significa que hay muchas oportunidades para que las compañías ganen dinero a través de la venta de juegos, contenido adicional y hardware.

## Diferentes Géneros de videojuegos y sus acepciones

Junto con el surgimiento de los juegos digitales se crearon muchos géneros y subgéneros de juegos que no existían previamente con los deportes o juegos de mesa tradicionales. Si bien estos géneros no están necesariamente estandarizados, se utilizan comúnmente para anticipar a los usuarios de forma sencilla qué esperar de la experiencia de un videojuego antes de comprarlo o también para realizar recomendaciones en base a otros similares. La comunidad de juegos en línea es capaz de reconocer decenas de géneros diferentes en donde se suele tomar un juego o saga de juegos como estándar y se utiliza para comparar con los demás. A continuación se describen algunos de los géneros utilizados en la industria por la comunidad y por las tiendas en línea, incluyendo los géneros a los que pertenece Spring Light.

### Puzzle

Un juego de género Puzzle es un juego en el que el jugador debe resolver desafiantes problemas o situaciones que requieren pensamiento lógico, habilidades matemáticas y estratégicas. Los juegos de puzzle pueden incluir desafíos basados en la lógica, la combinación de elementos, la manipulación de objetos, la resolución de acertijos y la planificación. Estos juegos suelen ser simples en términos de control, pero requieren una gran cantidad de concentración y pensamiento crítico. Ejemplos de juegos de puzzle incluyen Tetris, Bejeweled y Candy Crush.



## RPG

RPG es la abreviatura de "role-playing game". Se trata de un género de videojuegos que consiste en controlar un personaje o grupo de personajes con una historia, objetivos y desafíos que cumplir. Los jugadores deben tomar decisiones y realizar acciones para progresar en la historia y desarrollar el personaje o personajes que controlan. La experiencia RPG suele incluir elementos como la elección de habilidades, la obtención de equipamiento, la resolución de misiones y la exploración de un mundo virtual. Los RPGs pueden ser de estilo japonés o occidental, y pueden ser en línea o para un solo jugador.

## Simulación / Simulator / Sim

Un videojuego de simulación es un juego que trata de imitar un sistema real o ficticio a través de un modelo computarizado. Los juegos de simulación suelen recrear una actividad, un proceso o un ambiente específico y permiten a los jugadores experimentar cómo sería participar en esa actividad o en ese ambiente. Los ejemplos incluyen simuladores de vuelo, simuladores de estrategia militar, simuladores de construcción de ciudades, entre otros. Los jugadores pueden tener control total o limitado sobre las variables y las decisiones que afectan al sistema simulado.

## Farm Simulator (Subgénero)

Un juego de simulación de granja es un juego que imita la experiencia de tener y gestionar una granja. El objetivo puede variar desde simplemente cultivar y cuidar las cosechas hasta expandir la granja, construir estructuras, comprar y vender productos agrícolas, y administrar los recursos económicos de la granja. Los juegos de simulación de granjas suelen ser de estilo de juego de estrategia o simulación en tiempo real.

## Pixel Art

Un juego pixel art es un juego en el que la estética visual está compuesta por imágenes formadas por píxeles. Este estilo se hizo popular en los primeros días de los videojuegos y se ha vuelto popular en la actualidad, especialmente en juegos indie. Un juego pixel art suele tener un estilo retro y característico, con un enfoque en la simplicidad y la funcionalidad en lugar de una alta calidad visual. A menudo, los juegos pixel art incluyen una variedad de géneros, desde aventuras hasta juegos de rol y plataformas.

## Indie

Un juego Indie es un juego desarrollado por un estudio de desarrollo independiente o por un pequeño grupo de desarrolladores sin la ayuda de una compañía de videojuegos grande o un editor. Estos juegos suelen ser más pequeños y más íntimos que los juegos de consola o PC de gran presupuesto, y a menudo ofrecen experiencias más innovadoras o arriesgadas. Muchos juegos Indie se publican en plataformas digitales como Steam o itch.io y pueden ser una fuente importante de creatividad y experimentación en la industria de los videojuegos.

## Arte Digital: Pixel Art y Formatos Gráficos

El estilo llamado *Pixel Art* es una forma de arte digital que se crea mediante el uso de una computadora, caracterizado porque las imágenes son editadas a nivel pixel. Así, las imágenes no se conforman simplemente utilizando formas y colores como ocurre con el arte tradicional, sino que además el artista debe considerar la restricción de tener una unidad de tamaño fija, que conforma tanto un máximo como mínimo a la hora de agregar detalle. Este atributo único le da a esta forma de arte la característica de lograr increíbles niveles de detalle con pocos píxeles.

Los editores de gráficos rasterizados (o en inglés, Raster Graphic Image Editors) son programas que permiten crear y editar imágenes utilizando formatos de archivos gráficos rasterizados. En estos formatos, las imágenes se escriben y se leen en forma de una o más matrices de píxeles, en donde cada posición de la matriz corresponde a un color y posible transparencia. Ejemplos de estos formatos pueden ser JPG, JPEG, PNG y GIF. La figura 3.0 es un ejemplo de una imagen rasterizada en formato JPG.

Alternativamente, existen los formatos de gráficos vectorizados, como se muestra en la figura 4.0. La diferencia clave es que en estos los gráficos se guardan en forma de vectores en el espacio, en lugar de una matriz. Estos formatos no suelen ser buenos para representar imágenes foto realistas, ya que se caracterizan por su aspecto más caricaturesco o artificial. Sin embargo, poseen la ventaja de poder escalar a cualquier resolución sin problema y sin alterar el archivo original. Por este motivo, se suelen utilizar más para campos como el diseño gráfico, tipografías, logos, dibujos animados, etc.



**Figura 3.0:** Imagen Rasterizada

**Fuente:** White-Faced Heron Egretta - Free photo on Pixabay. (2022, September 21).  
Pixabay. Retrieved February 3, 2023, from  
<https://pixabay.com/photos/white-faced-heron-heron-7469267/>



**Figura 4.0:** Imagen Vectorial

**Fuente:** Lemons. (2019, November 18). Free SVG. Retrieved February 3, 2023, from <https://freesvg.org/lemons>

Originalmente el Pixel Art se desarrolló principalmente para videojuegos cerca de la década de los 80. La razón de esto fue, por un lado hacer uso de los píxeles físicos en los monitores antiguos de baja resolución, y por otro lado aprovechar su bajo nivel de detalle para aligerar el tamaño de los archivos gráficos, que constituyen la mayor parte del tamaño de un videojuego. Los primeros juegos en incluir este tipo de gráficos fueron por ejemplo el Space Invaders (1978) y Pacman (1980), y luego las consolas de 8 bits como la NES y Sega.

Usualmente se observan dos variantes del Pixel Art, según la perspectiva desde la cual están dibujadas las imágenes. Cuando los objetos se dibujan desde su lado, ya sea izquierdo o derecho, pero dejando ver parte de su frente o posterior, se considera Pixel Art Isométrico, como se muestra en la figura 5.0. El Pixel Art no isométrico, es aquel que no cae en esta categoría, incluyendo las vistas de frente, lado, detrás, encima (también llamada Top-Down) o en perspectiva. La figura 6.0 es un ejemplo de este tipo de arte.

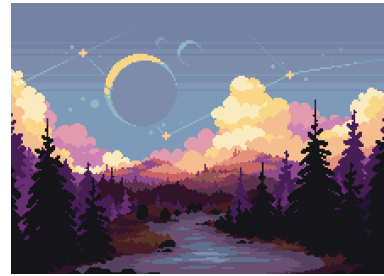


**Figura 5.0:** Ejemplos de Pixel Art Isométrico

**Fuentes:**

isometric pixel art by Me : r/PixelArt. (2022, June 20). Reddit. Retrieved February 3, 2023, from [https://www.reddit.com/r/PixelArt/comments/vg6m2i/isometric\\_pixel\\_art\\_by\\_me/](https://www.reddit.com/r/PixelArt/comments/vg6m2i/isometric_pixel_art_by_me/)

Isometric pixel art by JessPXL on DeviantArt. (2021, July 27). DeviantArt. Retrieved February 3, 2023, from <https://www.deviantart.com/jesspxl/art/Isometric-pixel-art-886987470>



**Figura 6.0:** Ejemplos de Pixel Art no Isométrico

**Fuentes:**

. (2022, August 10). YouTube. Retrieved February 3, 2023, from  
<https://ar.pinterest.com/pin/trix-on-twitter--31666003619300943/>

Schlitter, R. (2021, December 2). Pixelblog - 35 - Top Down Interiors — SLYNYRD.  
SLYNYRD. Retrieved February 3, 2023, from  
<https://www.slynyrd.com/blog/2021/11/30/pixelblog-35-top-down-interiors>

## Compatibilidad entre Hardwares y Sistemas Operativos

Crear videojuegos para diferentes plataformas o sistemas operativos puede ser complejo debido a las diferencias en las especificaciones del hardware, las diferencias en los sistemas operativos y las diferencias en los SDKs (Software Development Kits) y las herramientas de desarrollo.

Para desarrollar un juego en una plataforma específica, se requiere conocer los detalles técnicos de la plataforma en cuestión, como la memoria disponible, el procesador, la tarjeta gráfica, la resolución de pantalla, etc. También es necesario conocer las herramientas de desarrollo y los SDKs disponibles para esa plataforma para poder crear un juego que funcione de manera óptima en ese entorno. Además, adaptar un juego para diferentes plataformas también puede requerir ajustes en el diseño y la implementación del juego, debido a las diferencias en los controles, la interfaz de usuario y las capacidades de red.

Un motor de videojuegos, como Unity, puede ayudar a reducir la complejidad de crear videojuegos para diferentes plataformas al proporcionar una capa de abstracción sobre el hardware y el sistema operativo. Esto significa que, en lugar de tener que escribir código específico para cada plataforma, los desarrolladores pueden escribir código una sola vez y usar el motor para ejecutarlo en diferentes plataformas. Además, los motores de juegos proporcionan un conjunto de herramientas y características que facilitan el desarrollo de juegos, como renderizado, física, iluminación, animaciones, entre otras cosas.

- **PC:** Unity permite exportar juegos para Windows y Mac en diferentes formatos como .exe, .app, .dmg entre otros. En estos casos existen ciertas características o restricciones a tener en cuenta, especialmente para juegos que demanden muchos recursos como los juegos en 3D. Estos incluyen requisitos de hardware mínimos, compatibilidad con diferentes sistemas operativos, compatibilidad con diferentes tarjetas gráficas, configuraciones de pantalla y resolución, etc.





- **Consolas:** Unity permite exportar juegos para consolas de videojuegos como PlayStation, Xbox, Nintendo Switch. En este caso los formatos de archivos dependen de la consola, y suelen ser formatos propietarios. Si bien en estas plataformas es necesario tener en cuenta el sistema de Input personalizado de cada una, corren con la ventaja de que todas las consolas tendrán exactamente las mismas características, con lo cual el desarrollador ya puede saber de antemano que tan bien se ejecuta el juego y lograr optimizaciones muy eficientes.
- **Dispositivos móviles:** Unity permite exportar juegos para iOS y Android en diferentes formatos como .apk y .ipa entre otros. De todas las plataformas posibles, la móvil es la que presenta más variedad de hardwares y potencias, ya que existe un mercado muy amplio. Además, se requiere tener en cuenta que la mayoría de estos dispositivos no son muy potentes en cuanto a poder de procesamiento, con lo cual los factores a tener en cuenta incluyen requisitos de hardware mínimos, compatibilidad con diferentes sistemas operativos, compatibilidad con diferentes tamaños de pantalla y resolución, ajustes de batería, etc. También es común limitar el uso de ciertos recursos del dispositivo, como la memoria y la CPU, para asegurar un rendimiento óptimo en dispositivos con especificaciones limitadas.
- **Web:** Por último, Unity permite exportar juegos para navegadores web utilizando HTML5 y WebGL, los formatos de archivos son .html, .js y .data entre otros.

## Distribución

En el pasado, la distribución de los videojuegos se llevaba a cabo de forma diferente a como se hace actualmente. Antes de la era digital, los videojuegos se distribuían principalmente en formato físico, como cartuchos, discos o cintas. Estos medios físicos se enviaban a las tiendas especializadas y a los salones de juegos, donde los consumidores podían comprarlos o alquilarlos.

En el caso de los juegos de arcade, éstos se distribuían directamente a los centros de arcade, donde los propietarios de los establecimientos los instalaban en las máquinas y ofrecían a los clientes jugar a cambio de monedas o fichas. Para las consolas de juegos, las consolas y los juegos se vendían por separado, y estos últimos se distribuían en tiendas especializadas o grandes cadenas. Con el avance de la tecnología aparecieron nuevas formas de distribución, como los servicios de descarga en línea para consolas y PC, que permitían a los usuarios descargar y jugar juegos digitalmente. Ejemplos de dichos servicios incluyen:

- **Steam:** Es una plataforma de distribución digital de juegos para PC y Mac. Ofrece una amplia variedad de juegos, incluyendo tanto títulos independientes como juegos de gran presupuesto de desarrolladores importantes. A lo largo del tiempo se ha convertido en un estándar de distribución para la industria de videojuegos en PC. La desventaja de Steam es que cobra alrededor de un 30% de las ganancias de los juegos, con lo cual muchas compañías optan por distribuir sus juegos de manera individual a través de sus propias páginas.
- **Epic Games Store:** Es una plataforma de distribución de juegos digitales para PC y Mac propiedad de la compañía desarrolladora de juegos Epic Games. Se centra en ofrecer juegos gratuitos cada semana y promociones especiales. Epic Store cobra un 12% de comisión a los juegos, siendo uno de los stores que menos comisión tiene. Esto es parte de su estrategia para intentar derrocar a Steam, el principal proveedor de juegos para PC.

- **PlayStation Store, Xbox Game Store y Nintendo eShop:** Estas son las tiendas en línea para consolas de juegos de Sony, Microsoft y Nintendo respectivamente, donde los usuarios pueden comprar y descargar juegos digitales para sus consolas.
- **Apple App Store and Google Play:** Estas son las tiendas en línea para dispositivos móviles de Apple y dispositivos con sistema operativo Android, donde los usuarios pueden descargar y jugar juegos móviles. El App Store cobra, al igual que Steam, un 30% de comisión, mientras que el Play Store tan solo un 15%.

### 3. La aplicación

La aplicación desarrollada, como fue mencionado con anterioridad, se trata de un videojuego para un solo jugador. Este cuenta con diferentes aspectos comunes a otros juegos pero los combina dándoles una vuelta original. El producto se cataloga dentro de lo que se conoce como RPG, o "role-playing game", siendo a su vez un simulador de granja y un juego de Puzzles. Además involucra ciertos elementos de la estética japonesa, como por ejemplo las puertas "Torii"<sup>24</sup> o la apariencia del personaje principal siendo muy similar a una "Geisha"<sup>25</sup>.

El juego es de tipo "Open-World" y cuenta con diversos mapas que el jugador puede recorrer libremente. El mapa principal es la granja, en donde el personaje puede cultivar plantas, comprar y vender productos. Recorriendo los otros mapas se pueden encontrar rompecabezas que, al resolverse, otorgan recompensas en forma de ítems para la granja.

El juego estaría disponible para PC a través de alguna plataforma o "store" en línea, con la posibilidad de lanzarla en nuevas plataformas en el futuro gracias a las posibilidades que ofrece Unity. Sin embargo, debido a que las plataformas de distribución consideradas presentan barreras de entrada monetarias o de inspección que dependen de terceros esto no se realizó para esta instancia. La distribución del juego se realizó de forma directa a través del archivo ejecutable.

El proceso de desarrollo y las decisiones tomadas se explican con mayor detalle en la sección 7, titulada "Desarrollo".

Si bien el juego es muy simple y podría ejecutarse en hardwares de muy baja potencia, es importante aclarar algunos requerimientos mínimos que debe cumplir tanto el

---

<sup>24</sup> Torii. (n.d.). Wikipedia. Retrieved February 3, 2023, from <https://es.wikipedia.org/wiki/Torii>

<sup>25</sup> Geisha. (n.d.). Wikipedia. Retrieved February 3, 2023, from <https://es.wikipedia.org/wiki/Geisha>

hardware como el software para poder asegurar la calidad en ese marco. Para Spring Light se definieron:

## Requisitos mínimos de Software

- Sistema Operativo: Windows 10 x64 o superior
- Sistema Operativo Mac 10.13 o superior

## Requisitos mínimos de Hardware

- Espacio de almacenamiento: 300 MB o superior
  - Con un build de 250 MB
- 1 GB RAM o superior
- CPU Intel i3 3era generación, similar o superior

## Marketplace de Assets

Paralelamente a esto, se desarrolló también una plataforma web llamada el “Marketplace de Assets”. Este “Marketplace” consistió en un servicio que permite a los usuarios crear y buscar recursos de tipo textura que se pueden importar al juego “Spring Light”.

El alcance se describe como un MVP para visualizar y experimentar el proceso desde la creación del recurso hasta su puesta en funcionamiento en el juego y el recorrido que hace el usuario. La plataforma se compone de un servidor y un cliente web que manejan y procesan datos referentes a imágenes creadas por los usuarios para poder personalizar el arte visual del juego original.

Para lograr esto, fue necesario capacitarse en varias tecnologías y herramientas, planificar el trabajo a realizar y llevar a cabo la ejecución de ese plan. Se hizo uso asimismo del contenido visto en varias materias de la facultad.

## 4. Tecnologías

Para el desarrollo se utilizaron diversas tecnologías y herramientas en simultáneo que facilitaron el proceso.

### Unity

Como motor de videojuego se utilizó Unity, debido a que es una herramienta popular en la industria con muchos recursos accesibles para aprender a utilizar, inclusive dentro del ámbito de la universidad donde algunos profesores y alumnos conocidos ya la utilizan. Por otro lado también influyó un interés personal en aprender a utilizar esta herramienta.

Unity sirvió de base para el desarrollo, integrando el arte y el código y luego permitiendo crear el ejecutable para diversas plataformas. Unity permite además integrar mediante su Asset Store paquetes creados por otros miembros de la comunidad los cuales agregan funcionalidades o utilidades a la plataforma.

El Asset Store de Unity es una plataforma en línea donde los desarrolladores de juegos y otros profesionales de la industria pueden comprar y vender recursos y herramientas para sus proyectos de Unity. Estos recursos incluyen gráficos, scripts, audio, animaciones, modelos 3D, efectos visuales, entre otros elementos que pueden ser utilizados para agilizar y mejorar el proceso de desarrollo de juegos y aplicaciones en Unity. Los recursos disponibles en el Asset Store están disponibles para descargar e importar directamente en un proyecto de Unity, lo que puede ahorrar tiempo y esfuerzo, acelerando el proceso de desarrollo.

Del Asset Store se utilizaron por ejemplo los paquetes Cinemachine, RPGTalk, JetBrains Rider Editor y Test Framework, entre otros. Estos se explican a continuación.

## Cinemachine

Este paquete, desarrollado personalmente por Unity, es una suite de herramientas que permite utilizar diversas cámaras virtuales, dinámicas e inteligentes para conseguir fácilmente alcanzar efectos de cámaras<sup>26</sup> complejos. El paquete fue lanzado por primera vez junto con Unity 2019. Anteriormente, los efectos de cámaras debían ser creados por el programador manualmente sin soporte de la plataforma, y esto lo hacía un proceso difícil y tedioso. Rápidamente, Cinemachine se ha convertido en un estándar de la industria al desarrollar con Unity.

En Spring Light, se utiliza Cinemachine para crear una cámara virtual anclada al jugador en el centro. Dentro de las opciones que permite esta cámara, se configuró de tal forma que exista un área en el centro de la pantalla dentro de la cual el jugador puede moverse libremente, pero que al salir de este área la cámara se moverá para volver a ubicar al jugador dentro. En la figura 7.0 se muestra un cuadrado blanco representando el espacio de cámara que se muestra al jugador. Luego dentro de este existe un área virtual delimitada por un cuadrado verde dentro del cual el personaje puede moverse libremente sin reposicionar la cámara. La línea punteada verde marca el límite desde el cual si el personaje la cruza se comienza a mover la cámara

---

<sup>26</sup> Un efecto de cámara es un elemento visual en un videojuego que cambia la forma en que se muestra la escena o la perspectiva de la cámara. Estos efectos pueden incluir cosas como shakes, desenfoques, distorsiones, y otros efectos visuales que mejoran la atmósfera o la dramatización de la escena. Los efectos de cámara también pueden ayudar a los desarrolladores a crear una sensación de movimiento o a enfatizar determinados eventos dentro del juego.



**Figura 7.0:** Área de movimiento del jugador


**Fuente:** Elaboración Propia

Otra opción, llamada “Look Ahead<sup>27</sup>”, permite mover la cámara donde se estima que el jugador que se está moviendo estará, mediante un algoritmo de predicción. Por último, se configuró la cámara para que sea “pixel perfect<sup>28</sup>”. Esto es importante, ya que de no hacerlo se estimara la posición de algunos pixeles en la pantalla para ahorrar procesamiento. Normalmente esto es una ventaja para cualquier juego, ya que consume menos recursos de la máquina. Sin embargo, para un juego de pixel art con resolución tan baja (16x16), la estimación de los píxeles se puede notar a simple vista<sup>29</sup> y crea efectos extraños que pueden confundir al jugador. La cámara “pixel perfect” arregla esto calculando la posición de cada píxel de la pantalla individualmente.

<sup>27</sup> Cinemachine. (n.d.). Unity. Retrieved January 11, 2023, from <https://unity.com/es/unity/features/editor/art-and-design/cinemachine>

<sup>28</sup> Se dice que involucra precisión al nivel del píxel

<sup>29</sup> Para ver este efecto en movimiento ver la siguiente referencia:

 How to set up pixel perfect camera with cinemachine in Unity



## JetBrains Rider Editor

Rider es un IDE<sup>30</sup> desarrollado por la empresa JetBrains<sup>31</sup> para el desarrollo multiplataforma .NET<sup>32</sup>. En este proyecto particularmente se utilizó Rider para la edición de los Scripts en el lenguaje C#, que es el lenguaje principal para el cual Unity tiene soporte. Para poder utilizar Rider junto con Unity es necesario además el paquete JetBrains Rider Editor que funciona de puente entre ambos programas.

En la comunidad se observan diferentes IDEs junto con Unity. Además de Rider se utiliza también comúnmente Visual Studio<sup>33</sup> y Visual Studio Code, ambos desarrollados por Microsoft. El principal motivo por el que se utilizan dichas alternativas es porque Rider es un software pago y costoso, sobre todo para compañías pequeñas o desarrolladores individuales de presupuesto limitado. En este caso, siendo que el IDE no afecta el resultado final sino que simplifica el proceso, se eligió utilizar Rider que se provee con el paquete de educación de JetBrains<sup>34</sup>, ya que es el que, personalmente, tenía menor curva de aprendizaje.

---

<sup>30</sup> Integrated development environment. (n.d.). Wikipedia. Retrieved February 9, 2023, from [https://en.wikipedia.org/wiki/Integrated\\_development\\_environment](https://en.wikipedia.org/wiki/Integrated_development_environment)

<sup>31</sup> Herramientas esenciales para desarrolladores de software y equipos. (n.d.). JetBrains. Retrieved February 9, 2023, from <https://www.jetbrains.com/es-es/>

<sup>32</sup> Microsoft .NET. (n.d.). Wikipedia. Retrieved February 9, 2023, from [https://es.wikipedia.org/wiki/Microsoft\\_.NET](https://es.wikipedia.org/wiki/Microsoft_.NET)

<sup>33</sup> Microsoft Visual Studio. (n.d.). Wikipedia. Retrieved February 9, 2023, from [https://es.wikipedia.org/wiki/Microsoft\\_Visual\\_Studio](https://es.wikipedia.org/wiki/Microsoft_Visual_Studio)

<sup>34</sup> <https://www.jetbrains.com/community/education/#students>

## Backend Web

Como tecnología de backend se decidió utilizar NestJS. NestJS<sup>35</sup> es un framework de aplicación Node.js<sup>36</sup> que permite el desarrollo rápido y eficiente de aplicaciones modernas y escalables. Ofrece una amplia gama de características para desarrollar aplicaciones de back-end, como modularidad, seguridad, rendimiento y flexibilidad.

NestJS se basa en la arquitectura del patrón de diseño de aplicaciones web Model-View-Controller (MVC)<sup>37</sup> y utiliza TypeScript<sup>38</sup>, lo que permite una mejor organización y manejo del código. También ofrece una integración nativa con el sistema de dependencias y una infraestructura de pruebas robusta.

Además, NestJS cuenta con una amplia variedad de bibliotecas y herramientas, como GraphQL, gRPC y Swagger, que permiten la integración de tecnologías avanzadas y la creación de aplicaciones altamente sofisticadas.

En el caso de esta aplicación, se decidió utilizar NestJS por las razones mencionadas anteriormente, además de que ya se conocía el lenguaje Typescript y NodeJS presentaba una extensa y completa documentación para poder aprender a utilizarlo en poco tiempo.

Como base de datos de la plataforma se utilizó PostgreSQL<sup>39</sup>. PostgreSQL es un sistema de gestión de bases de datos relacionales de código abierto. Es conocido por su

---

<sup>35</sup> Mysliwiec, K. (n.d.). NestJS. NestJS - A progressive Node.js framework. Retrieved February 13, 2023, from <https://nestjs.com/>

<sup>36</sup> Node.js. (n.d.). Node.js. Retrieved February 13, 2023, from <https://nodejs.org/es/>  
Node.js. (n.d.). Wikipedia. Retrieved February 13, 2023, from <https://en.wikipedia.org/wiki/Node.js>

<sup>37</sup> Model-view-controller. (n.d.). Wikipedia. Retrieved February 13, 2023, from <https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller>

<sup>38</sup> TS. (n.d.). TypeScript: JavaScript With Syntax For Types. Retrieved February 13, 2023, from <https://www.typescriptlang.org/>  
TypeScript. (n.d.). Wikipedia. Retrieved February 13, 2023, from <https://es.wikipedia.org/wiki/TypeScript>

<sup>39</sup> Postgresql. (n.d.). PostgreSQL: The world's most advanced open source database. Retrieved February 13, 2023, from <https://www.postgresql.org/>  
Stonebraker, M. (n.d.). PostgreSQL. Wikipedia. Retrieved February 13, 2023, from <https://es.wikipedia.org/wiki/PostgreSQL>

alta flexibilidad, robustez y rendimiento, y es ampliamente utilizado en aplicaciones empresariales y web. PostgreSQL es compatible con el estándar SQL y proporciona una amplia gama de características avanzadas, incluidas la gestión de datos geoespaciales, la replicación de datos, la integración de lenguajes de programación y la gestión de transacciones.

PostgreSQL es altamente escalable y se puede ejecutar en una amplia variedad de plataformas, incluidas Windows, MacOS y Linux. Además, cuenta con una amplia comunidad de desarrolladores y usuarios que contribuyen a su desarrollo y mejora continua, lo que garantiza que PostgreSQL siga siendo una solución robusta y actualizada.

Tanto el servidor como la base de datos están deployados en Heroku<sup>40</sup>. Heroku es una plataforma en la nube que permite a los desarrolladores implementar, ejecutar y escalar aplicaciones en la web. Ofrece una amplia variedad de opciones de despliegue, incluyendo soporte para una gran cantidad de lenguajes de programación, como Ruby, Java, PHP, Python, Node.js y muchos más. Los desarrolladores pueden usar Heroku para crear, probar e implementar aplicaciones web con facilidad, sin preocuparse por la infraestructura subyacente. Además, Heroku ofrece una interfaz intuitiva para la gestión de aplicaciones, lo que significa que los desarrolladores pueden centrarse en escribir código, en lugar de administrar servidores y otros componentes.

---

<sup>40</sup> Heroku. (n.d.). Wikipedia. Retrieved February 13, 2023, from <https://es.wikipedia.org/wiki/Heroku>

## Frontend Web

Para desarrollar el frontend se utilizó React. React<sup>41</sup> es una biblioteca de JavaScript para construir interfaces de usuario. Fue desarrollado y mantenido por Facebook, y es uno de los frameworks de front-end más populares y ampliamente utilizados en la actualidad.

React se enfoca en el concepto de componentes, que son bloques reutilizables de código que representan elementos individuales en la interfaz de usuario.

Otra característica importante de React es su enfoque en la programación declarativa, lo que significa que se enfoca en describir qué se quiere mostrar en la interfaz de usuario en lugar de cómo hacerlo. Esto permite una codificación más clara y fácil de mantener, y también mejora la experiencia de depuración.

Además, React es compatible con una amplia variedad de tecnologías y bibliotecas, lo que lo hace una opción popular para la construcción de aplicaciones web de alta escalabilidad y rendimiento.

La página web está deployada en Netlify<sup>42</sup>. Netlify es una plataforma de hosting y despliegue para aplicaciones web y sitios estáticos. Ofrece una amplia variedad de características, incluyendo integración con Git, soporte para múltiples idiomas de programación, CDN, implementación continua, gestión de formularios, y más. Netlify es fácil de usar y está diseñado para ayudar a los desarrolladores a ahorrar tiempo y simplificar el proceso de despliegue, lo que les permite centrarse en el desarrollo de sus aplicaciones. Además, Netlify es una solución de hosting escalable, lo que significa que puede manejar un gran tráfico sin interrupciones.

---

<sup>41</sup> React. (n.d.). Wikipedia. Retrieved February 13, 2023, from <https://es.wikipedia.org/wiki/React>  
React. (n.d.). React – Una biblioteca de JavaScript para construir interfaces de usuario. Retrieved February 13, 2023, from <https://es.reactjs.org/>

<sup>42</sup> Netlify. (n.d.). Wikipedia. Retrieved February 13, 2023, from <https://en.wikipedia.org/wiki/Netlify>

# Git y GitHub

Como herramienta de control de versión se decidió utilizar Git<sup>43</sup>. Este permite mantener un seguimiento de los cambios realizados a los archivos del proyecto, lo cual es útil para detectar o revertir cambios indeseados. Además, este sistema es el estándar de la industria de la programación. En el caso de este proyecto se eligió GitHub<sup>44</sup> para cumplir este rol.

Particularmente en la industria de los videojuegos existe otro sistema de control de versiones muy utilizado llamado Perforce<sup>45</sup> Helix Core. Perforce, a diferencia de Git, que es open source, es un software de tipo comercial desarrollado por la compañía del mismo nombre que disfruta también del servicio en la nube.

La ventaja que tiene Perforce sobre Git en este caso, es que Perforce ofrece un producto que contempla dentro de sus casos de uso la producción de videojuegos, con lo cual ya incorpora conceptos como el guardado de archivos multimedia pesados utilizando punteros en lugar de binarios, para poder optimizar el espacio y el tiempo consumido.

A pesar de esto, la razón por la que no se utilizó este producto es que Perforce ofrece una solución muy compleja y particular para empresas en las que el control de versiones y el trabajo en equipo es un problema considerable. En este proyecto, la cantidad de overhead que agregaba en tiempo de aprender una nueva herramienta y navegar la complejidad agregada, no compensaba el retorno de inversión para la funcionalidad que ofrecía.

Los archivos más pesados en el proyecto son los gráficos, que se componen de grillas de píxeles. Al ser un juego en pixel art, la mayoría de las imágenes tienen una resolución de 32x32 píxeles, es decir, 4 kilobytes. Esto equivale a lo que pesan dos

---

<sup>43</sup> Git. (n.d.). Wikipedia. Retrieved February 3, 2023, from <https://en.wikipedia.org/wiki/Git>

<sup>44</sup> GitHub. (n.d.). Wikipedia. Retrieved February 3, 2023, from <https://en.wikipedia.org/wiki/GitHub>,  
Github. (n.d.). GitHub: Let's build from here · GitHub. Retrieved February 3, 2023, from <https://github.com/>

<sup>45</sup> Perforce. (n.d.). Perforce Software | Development Tools For Innovation at Scale. Retrieved February 3, 2023, from <https://www.perforce.com/>

archivos de código del proyecto. Por este motivo, a pesar de estar guardando las imágenes directamente en Git, esto no se considera un problema, ya que no enlentece ninguna operación. En lo que respecta a un proyecto de un solo desarrollador y con el alcance planteado, Git se puede considerar una solución más que satisfactoria.

## Aseprite y Wacom

En lo que respecta a la creación y edición de los recursos artísticos utilizados en el proyecto se hizo uso del software Aseprite<sup>46</sup> como editor de gráficos rasterizados<sup>47</sup>. La razón por la cual los gráficos deben ser rasterizados es por la naturaleza del pixel art. La capacidad de controlar cada uno de los píxeles permite a los artistas crear imágenes detalladas y precisas, y les permite utilizar una amplia gama de técnicas para crear texturas, sombras y otros efectos visuales que simplemente no serían posibles en un editor de gráficos vectoriales para la resolución a nivel pixel.

Aseprite está orientado particularmente a la creación de arte digital en estilo pixel art, con lo cual se considera uno de los mejores softwares para este propósito. Dentro de sus funcionalidades permite además la creación de animaciones. Junto con este programa se utilizó una tableta marca Wacom<sup>48</sup> para simular un entorno de desarrollo en una empresa real de videojuegos, en donde los artistas utilizan tabletas gráficas para dibujar. El uso de esta tableta fue muy útil ya que todos los programas de edición de gráficos serios están pensados para utilizarse de esta manera en lugar del mouse. La tableta permite dibujar de forma más natural, utilizando un lápiz táctil que además detecta cambios de presión y esto se traduce directamente en la imagen.

---

<sup>46</sup> Aseprite. (n.d.). Aseprite - Animated sprite editor & pixel art tool. Retrieved February 3, 2023, from <https://www.aseprite.org/>

<sup>47</sup> Editor de gráficos rasterizados. (n.d.). Wikipedia. Retrieved February 3, 2023, from [https://es.wikipedia.org/wiki/Editor\\_de\\_gr%C3%A1ficos\\_rasterizados](https://es.wikipedia.org/wiki/Editor_de_gr%C3%A1ficos_rasterizados)

<sup>48</sup> Sakai, M. (n.d.). Wacom. home. Retrieved February 3, 2023, from <https://www.wacom.com/es-es>

## 5. Arquitectura

La arquitectura típica de un videojuego desarrollado en Unity se basa en varios componentes clave que trabajan juntos para crear una experiencia de juego completa. La mayoría de estos componentes son provistos directamente por el motor, aunque es posible agregar y quitar según discreción del usuario. Estos componentes incluyen:

- **Scene:** La “Escena” es el contenedor o conjunto principal de todos los elementos de una escena del juego. Un juego de Unity puede tener varias escenas, cada una de las cuales representa una parte diferente del juego, como un menú principal, un nivel o una pantalla de pausa. Las escenas se pueden cargar en memoria de forma aditiva o reemplazando las escenas actuales.
- **GameObject:** Un GameObject es la unidad básica de los elementos del juego. Todo lo que se ve en una escena de Unity es un GameObject. Los GameObjects tienen una jerarquía, lo que permite organizar los elementos del juego en una estructura lógica. Adicionalmente, se le puede agregar a cada GameObject diferentes componentes para cambiar su funcionamiento o comportamiento.
- **Componentes:** Cada GameObject puede tener varios componentes. Estos componentes le dan al GameObject sus funcionalidades, como transformación, comportamiento, gráficos, audio, etc. Unity provee con su paquete por defecto una cantidad de componentes de propósito general que ayudan con las funcionalidades básicas de cualquier juego. Adicionalmente, el usuario es capaz de crear componentes nuevos por medio de Scripts.
- **Assets:** Los assets son los recursos del juego, como texturas, modelos 3D, audio, scripts, entre otros.



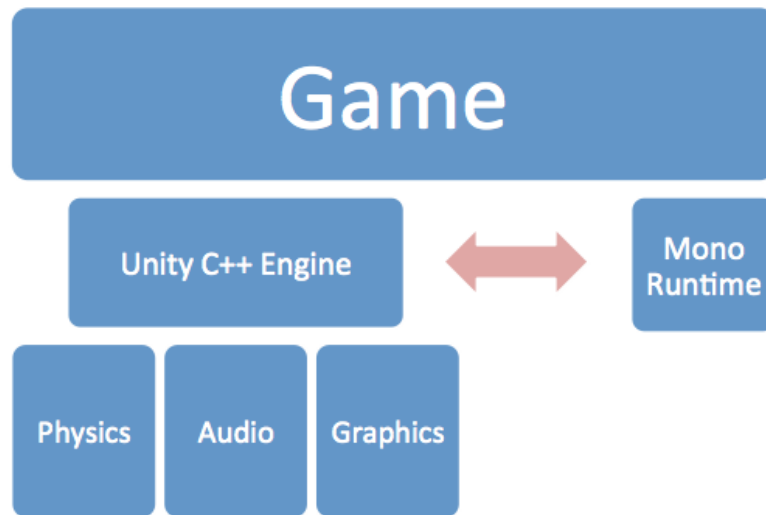
- **Scripts:** Los scripts son archivos de código que se utilizan para controlar el comportamiento de los GameObjects y la lógica del juego. Los scripts en Unity se pueden escribir en el lenguaje de programación C# o UnityScript.
- **Motor de física:** Unity provee un motor de física incorporado llamado Unity Physics. Este motor se utiliza para simular eventos físicos en el juego, como colisiones, gravedad y fuerzas, utilizando componentes que Unity trae por defecto. Este motor es muy potente y se puede utilizar tanto para juegos en 3D como para 2D.
- **Motor de renderizado:** Unity tiene un motor de renderizado incorporado que se utiliza para dibujar los elementos del juego en la pantalla. Este motor permite a los desarrolladores controlar cómo se ven los elementos en el juego, como las luces, las sombras, las texturas, entre otros.
- **Sistema de animación:** El sistema de animación incorporado que permite a los desarrolladores crear animaciones para los modelos 3D en el juego.
- **Sistema de sonido:** Permite a los desarrolladores añadir audio al juego, como música, efectos de sonido y diálogos. Si bien este sistema es bastante complejo y permite incluso calcular la distancia entre los sonidos y el jugador para simular distancia o sonidos surround o estereo, en el caso de Spring Light esto cae por fuera del alcance.

La arquitectura del juego a la hora de ejecución en Unity es muy flexible y está diseñada para soportar múltiples plataformas. Esta se muestra en la figura 8.0. Cuando un juego se construye, el motor genera un archivo ejecutable específico para la plataforma seleccionada. Este archivo ejecutable contiene todo lo necesario para ejecutar el juego en la plataforma, incluyendo el código del juego, los recursos, los assets, los scripts, etc. El código en sí está escrito en C#, pero el motor de Unity se encarga de traducirlo a un lenguaje de máquina nativo para la plataforma específica, ya sea para PC, consolas, dispositivos móviles, entre otros.

Es importante tener en cuenta, de todos modos, que si bien Unity provee una forma para convertir el código fuente en ejecutable de cualquier plataforma, muchas veces la

conversión conlleva un costo adicional. Por ejemplo, en el caso de Spring Light, si se quisiera crear un ejecutable en móvil para Android se tendría que crear todo un sistema nuevo de Input para que el jugador sea capaz de interactuar con los objetos a través de un dispositivo táctil. Además, se tendría que diseñar e implementar esta nueva interfaz de usuario, posiblemente alterando la que se definió para PC. La arquitectura final resultante de compilar el juego para una plataforma específica se muestra en la figura 9.0.

Esto involucra el mantener dos versiones paralelas de la UI para las diferentes plataformas, lo cual agrega overhead de mantenimiento. En proyectos más complejos o más demandantes de recursos también se debe tener en cuenta la marcada baja de recursos que ofrece un dispositivo mobile, por lo que es posible que se deba investigar y probar meticulosamente para ver los requisitos mínimos que necesita el sistema, e incluso realizar cambios para mejorar la eficiencia. Todo esto es trabajo adicional que se debe tener en cuenta dependiendo de cada plataforma nueva en la que se quiera lanzar el producto en paralelo.

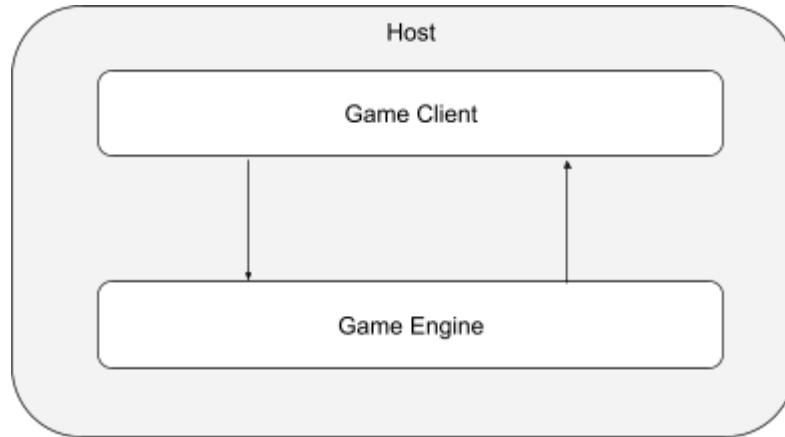


**Figura 8.0:** Arquitectura interna del motor Unity. El Mono Runtime<sup>49</sup> es el motor que se encarga de traducir el código del lenguaje de programación al lenguaje nativo del procesador.

**Fuente:** de Icaza, M. (2011, March 7). GDC 2011 - Miguel de Icaza. tirania.org. Retrieved February 9, 2023, from <https://tirania.org/blog/archive/2011/Mar-07.html>

---

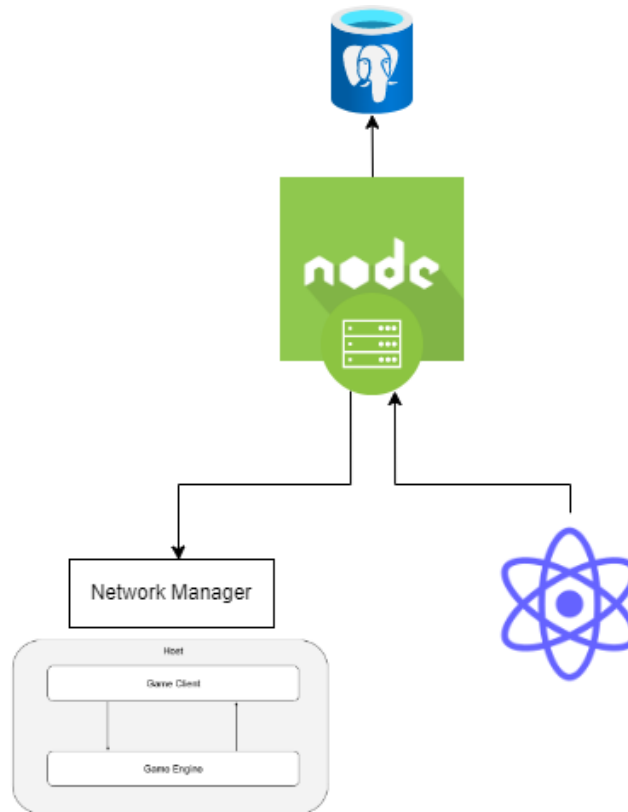
<sup>49</sup> *Mono (software)*. (n.d.). Wikipedia. Retrieved February 10, 2023, from [https://en.wikipedia.org/wiki/Mono\\_\(software\)](https://en.wikipedia.org/wiki/Mono_(software))



**Figura 9.0:** Arquitectura del juego ejecutable corriendo en un sistema cliente

**Fuente:** Elaboración Propia

Por otro lado, la arquitectura del servidor web se compone de dos partes principales. Por un lado se encuentra el backend, el cual se encarga de hacer el procesamiento de las imágenes y manejar su guardado y distribución. Por el otro, se encuentra el cliente web, que funciona a modo de interfaz para que los usuarios puedan crear sus texturas o skins con facilidad, acceder a las otras funcionalidades del sitio y backend. El diagrama de la arquitectura final se muestra en la figura 67.0.



**Figura 67.0:** Diagrama de arquitectura de todo el sistema en su totalidad

**Fuente:** Elaboración Propia

El cliente web, implementado en React y corriendo en un servidor de Netlify, se comunica con el backend, el cual está implementado en NestJs y corriendo en un servidor de Heroku. El backend a su vez, se comunica con la base de datos, hospedada en Heroku también, para guardar la información de los usuarios y de las texturas o sprites que creen.

Cuando los usuarios quieran agregar una nueva skin a su juego, deberán cargar un código. Este código lo utilizará el componente Network Manager para comunicarse con el backend y pedir las imágenes y la información asociada para poder construir la skin en el juego.

## 6. Investigación

Previo y durante el desarrollo de la aplicación se investigó sobre diversos temas pertinentes al desarrollo de videojuegos. Como se carecía de experiencia previa, se investigaron elementos desde los más generales hasta más específicos. Se detallan a continuación algunos de los temas investigados, las decisiones tomadas y sus aplicaciones en el proyecto.

### Metodologías de Trabajo

Uno de los puntos a analizar dentro de la industria del desarrollo de videojuegos es las diferentes metodologías que se utilizan. Al ser los videojuegos un producto de software utilizan comúnmente metodologías muy similares a otros productos. Dentro del ámbito de la programación, existen varias metodologías o modalidades de trabajo que se utilizan para organizar la creación de un software. Estos se utilizan independientemente de la industria que requiera el software ya que trabajan a un nivel de abstracción superior. Algunos ejemplos pueden ser:

- **Waterfall**<sup>50</sup>: Este es un enfoque tradicional en el que se sigue una secuencia lineal de etapas, comenzando con el análisis del juego, seguido del diseño, implementación, pruebas y mantenimiento. Es un enfoque muy planificado y estructurado que es más adecuado para proyectos de gran escala y con un alto nivel de complejidad.

---

<sup>50</sup> Rerych, M., & Wright, S. (n.d.). fit 2002. fit 2002. Retrieved February 13, 2023, from <http://cartoon.iguw.tuwien.ac.at/fit/fit01/wasserfall/entstehung.html>

- **Agile**<sup>51</sup>: Este es un enfoque iterativo e incremental que se basa en el desarrollo ágil de software. Se divide en ciclos de trabajo cortos, conocidos como sprints, en los que se desarrolla una parte del juego, se prueba y se realizan ajustes. Es un enfoque más flexible y adaptable que permite a los desarrolladores adaptarse rápidamente a los cambios en el juego o en el equipo.
- **Scrum**<sup>52</sup>: Es un marco de trabajo ágil que se basa en el enfoque Agile y se enfoca en la gestión del equipo y el proceso de desarrollo. Se compone de roles, ceremonias y artefactos específicos, que permiten una mayor colaboración y transparencia en el equipo.
- **Lean**<sup>53</sup>: Es un enfoque de desarrollo de videojuegos que se basa en los principios del Lean Development, una metodología de producción que se utiliza en la industria para reducir los tiempos de ciclo y aumentar la eficiencia. El enfoque Lean se centra en la eliminación de desperdicios y el aumento de la eficiencia en el desarrollo de juegos.

Existen otras metodologías de trabajo pero estos son los más utilizados en el desarrollo de videojuegos. Dependiendo de las necesidades del juego y del equipo, es posible que se utilice una combinación de estas o se desarrollen nuevas personalizadas.

Además de estas metodologías de desarrollo, también existen procesos específicos para la creación de contenido, como la planificación y la creación de gráficos, audio y animaciones. Estos procesos pueden ser independientes de la metodología de desarrollo general, pero están estrechamente relacionados con ella.

Durante el desarrollo se utilizó la metodología Agile. Si bien tiene unos principios que la definen, esta metodología se caracteriza por ser un paraguas que abarca diversas formas de trabajar, aunque todas enfocadas en cambios incrementales y veloces. En este

---

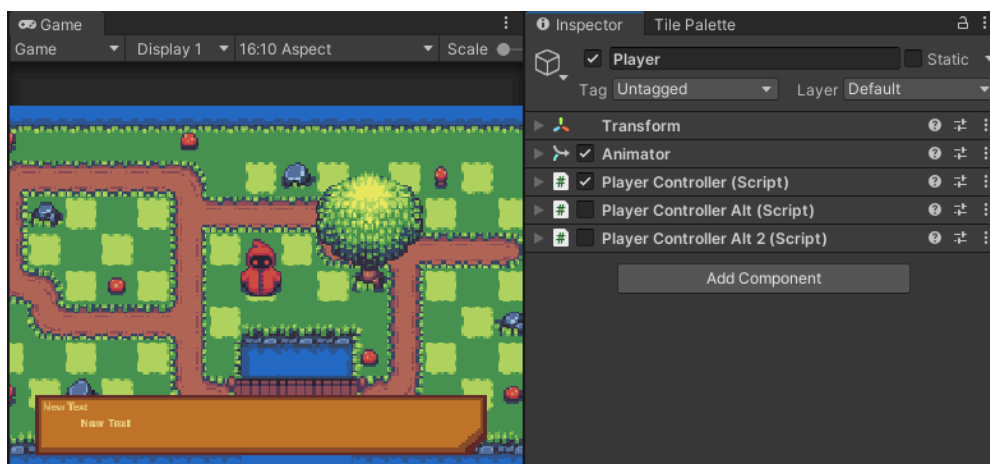
<sup>51</sup> Medinilla, Á., & Gómez, E. (n.d.). Manifiesto por el Desarrollo Ágil de Software. Manifiesto for Agile Software Development. Retrieved February 13, 2023, from <https://agilemanifesto.org/iso/es/manifesto.html>

<sup>52</sup> Sutherland, J. (2015). Scrum: el nuevo y revolucionario modelo organizativo que cambiará tu vida. Planeta.

<sup>53</sup> Poppendieck, T. D., Poppendieck, M., & Poppendieck, T. (2003). *Lean Software Development: An Agile Toolkit*. Addison-Wesley.

caso la modalidad de trabajo consistió en primer lugar en hacer una definición a alto nivel de las funcionalidades que se pretendían incorporar en el juego.

Luego, se comenzó directamente con la capacitación para ya empezar a producir código que fuera útil para el proyecto. Esto es una recomendación que se da mucho en la industria, “Fail fast, fail cheap”. Esto quiere decir que es necesario fallar muchas veces para conseguir hacer un buen juego, por lo que es importante no concentrarse en hacerlo bien en primera instancia sino en probar muchas cosas diferentes y que el hecho de fallar no consuma muchos recursos. Intentando seguir esta metodología, se desarrollaron las primeras 3 capacitaciones como juegos independientes, para probar diferentes estrategias, estilos y herramientas, y poder ver cual era mejor. Esto es fácil de probar en Unity ya que permite activar y desactivar los scripts de los objetos de forma muy sencilla, lo cual sirve para probar el funcionamiento de diferentes implementaciones. Todo esto se puede ver en la figura 10.0.



**Figura 10.0:** Captura de pantalla del último juego de la capacitación. A la izquierda se puede ver al gameobject que representa al personaje que controla el jugador. A la derecha se ven los diferentes scripts que se le aplican a ese gameobject.

**Fuente:** Elaboración Propia



Una vez comenzado el desarrollo se establecieron las funcionalidades más básicas que debía de tener el juego para poder crear un prototipo, y apuntar a lograrlo de la forma más rápida y menos costosa posible. Cada dos semanas se establecían nuevos objetivos y funcionalidades utilizando una herramienta de tipo Kanban en la plataforma Trello, para tener un seguimiento visual de las funcionalidades terminadas, en progreso y por comenzar.

En retrospectiva esta metodología resultó muy útil. La volvería a utilizar para el desarrollo de cualquier videojuego, incluso haciendo iteraciones más pequeñas, como podría ser por ejemplo semanalmente. No obstante, para proyectos más grandes y complejos como son los de empresas con decenas de empleados, esto es mucho más difícil ya que todos los equipos deben coordinar entre sí y tener un estimativo de cuando terminaran cada tarea o se podrían bloquear el progreso mutuamente. El “overhead” de organización y planificación en ese caso crece considerablemente y es más importante tener una certeza sólida de la dirección específica que se quiere lograr con el juego.

## Etapas/Pipelines

Por otro lado, en conjunto con las metodologías de trabajo se consideran diferentes etapas o *pipelines*. Por “pipeline de trabajo” se entiende la secuencia de actividades o herramientas que reciben un *input* y producen un *output*, el cual finalmente concluye con la producción final del producto. Si bien existen muchos pipelines que se pueden implementar dependiendo de la envergadura del proyecto, los requerimientos y los recursos disponibles, durante el desarrollo de Spring Light se utilizaron los siguientes: Gráficos, Animación, Scripting, Testing. Estos se explican en más detalle en la figura 11.0. En cada color se representa un elemento diferente del producto. Si bien en ciertos casos puede admitirse cierta superposición entre pipelines para ahorrar tiempo, lo normal es que se deba terminar una etapa antes de comenzar la siguiente ya que el input de las etapas está estrechamente relacionado con el output de la anterior.



**Figura 11.0:** El diagrama de tipo Gantt representa diferentes pipelines que se dan en la creación de videojuegos y cómo se conectan.

**Fuente:** Elaboración Propia

## Pipeline de Gráficos

Este pipeline se utiliza para crear y producir los gráficos del juego. Usualmente en un proyecto de gran envergadura se compone de diferentes departamentos como podría ser marketing, arte, modelado, etc. En un proyecto pequeño lo puede realizar tan solo una persona. En este caso el trabajo de la persona es obtener o crear los recursos artísticos visuales que se utilizan en el juego, los cuales serán el output del pipeline.

Normalmente incluye modelos 3D, texturas, efectos de iluminación y sombras, y otros elementos visuales. En el caso de Spring Light se utilizó para crear o modificar los Sprites 2D en pixel art que se utilizan como texturas de los objetos en los renderizadores de imágenes. Para esto se utilizó el programa Aseprite. Los gráficos creados o editados con Aseprite se pueden guardar en un archivo .aseprite, y contienen toda la información que el programa necesita, incluyendo máscaras, capas y frames de animación. Luego desde este se puede exportar a cualquier tipo de archivo que se requiera en el proyecto.

## Pipeline de Animación

Este pipeline se utiliza para crear y producir las animaciones del juego, incluyendo personajes, vehículos, efectos especiales y otros elementos que requieren movimiento. Es muy similar al pipeline de gráficos pero la diferencia radica en que el input de este pipeline se conecta al output del pipeline anterior, por lo que se entiende que es necesario primero tener terminados todos los gráficos para poder comenzar con las animaciones del juego.

El output de este pipeline puede ser una combinación de muchas imágenes agrupadas en una animación o una animación de un modelo 3D. Es importante tener en cuenta que los pipelines no son una estructura rígida, sino que pueden tener un ida y vuelta de recursos. Por ejemplo, en el caso de estar realizando animaciones 2D es posible que haya que retocar gráficos o hacer cambios que pertenecen al pipeline de gráficos. En el caso de Spring Light las animaciones incluidas en el juego son muy simples, por lo cual alcanzó con el motor de animaciones que provee Unity para imágenes 2D.

Las animaciones se pueden crear originalmente en Aseprite utilizando el sistema de frames y luego exportar como imágenes en formato PNG en una misma imagen o en imágenes separadas. En este caso se utilizó una misma imagen en la que se tomaron las secciones por tamaño de 16x32 pixeles para el personaje principal y 16x16 píxeles para las demás imágenes. Estas imágenes se agrupan luego utilizando Unity en archivos .anim, los cuales especifican las diferentes propiedades que debe tener un objeto (por ejemplo el personaje) a lo largo del tiempo de la animación, como puede ser su posición, rotación, escala, etc.

## Pipeline de Scripting

Este pipeline se utiliza para crear y producir el código del juego, incluyendo scripts, sistemas de juego y mecánicas, y otros elementos que requieren programación. Se suele llevar a cabo por un equipo de programación que puede ser de una sola persona hasta una combinación de múltiples equipos trabajando juntos. Este pipeline, al igual que los otros, puede tener su propia metodología, por ejemplo utilizar Agile o Scrum. El input y output dependen del proyecto ya que en algunos casos puede requerir arte o animaciones pero en otros casos es posible realizar el código sin que estos estén aún.

Como output suele incluir el código ejecutable y posiblemente también algún documento con detalles o decisiones que se hayan realizado. En Unity, los scripts suelen guardarse y organizarse en una estructura de carpetas dentro de la carpeta "Assets" del proyecto. A menudo, se crean carpetas específicas para diferentes tipos de scripts, como "Scripts", "Controllers", "Managers", etc. Dentro de estas carpetas, los scripts se organizan según su función o su relación con otros elementos del juego.

Por ejemplo, puede haber una carpeta "Player" dentro de "Scripts" que contenga todos los scripts relacionados con el jugador, o una carpeta "AI" dentro de "Controllers" que contenga todos los scripts relacionados con la inteligencia artificial. Es importante tener una buena organización de los scripts para facilitar la colaboración en un eventual equipo y la mantenibilidad del código. Además, cualquier carpeta llamada "Resources" es una carpeta especial que permite acceder a sus contenidos de manera fácil y rápida desde código. Los

archivos contenidos en esta carpeta pueden ser accedidos utilizando el método `Resources.Load()`, que permite cargar un objeto contenido en la carpeta, dándole el nombre del archivo como parámetro. Esta funcionalidad es útil para cargar recursos como imágenes, sonidos, configuraciones, etc. Sin necesidad de tener que especificar la ruta completa del archivo.

## Pipeline de Testing

Este pipeline se utiliza para asegurar la calidad del juego, se realizan pruebas de calidad, para buscar errores y bugs, y asegurar que el juego funcione correctamente en todas las plataformas. El input es el código que se obtuvo del pipeline de scripting y el output es un reporte de la calidad de ese código, lo cual puede incluir más o menos detalles. Además, este pipeline es útil a la hora de atajar posibles errores que se introdujeron al código de forma inadvertida.

De todos los pipelines este es el que se suele automatizar más, ya que se puede configurar una suite de tests automáticos que se ejecuten cada vez que se presenten cambios en el código. Sin embargo, si es necesario introducir tests nuevos a mano mientras progresa el proyecto. En este proyecto el pipeline se implementó utilizando Github Actions. Al realizar un push a la branch dev o master, Github corre la suite de tests automáticamente para alertar al equipo en caso de que ocurra alguna falla y que pueda ser detectada lo antes posible.

Proyectos más grandes pueden contener un pipeline de Integración o deployment para crear entregables automáticamente. En el caso de Spring Light esto no fue necesario, ya que de momento solo se tiene una versión del juego en producción. Esta decisión se explica más en detalle en la sección 7, Fase 3 subtitulada “Distribución”.

## Pixel Art

En un juego que utiliza pixel art, la resolución es un factor importante que afecta a la apariencia y calidad del juego. Existen varias resoluciones posibles, pero las más comúnmente utilizadas son 8, 16, 32, 64 y 128. Estas se pueden ver en la figura 12.0. Si bien una resolución de 8 o 16 píxeles puede parecer la más sencilla o rápida de realizar, hay que tener en cuenta que estas resoluciones tan bajas hacen fuerte uso de técnicas como el espacio negativo o la sugestión para dar a entender la imagen.

Por este motivo, realizar dibujos y animaciones en una resolución tan baja suele ser más difícil de lo que parece. Una resolución muy alta como 128 pierde casi ya el efecto de pixel art, siendo más similar al arte digital convencional. Para el desarrollo de Spring Light se decidió utilizar una resolución de 32 píxeles. Utilizar una resolución de 32 en lugar de 16 pixels tiene varias ventajas:

- **Mayor detalle:** La mayor resolución permite un mayor detalle en los gráficos y una mejor representación de los objetos y personajes.
- **Menos pixelado:** Con una resolución más alta, los objetos y personajes tienen menos pixelado y se ven más suaves y nítidos.
- **Mayor flexibilidad** en la animación: Con una mayor resolución, se pueden crear animaciones más suaves y detalladas.
- **Mejor escalabilidad:** Una resolución más alta permite escalar los gráficos más fácilmente sin perder calidad, lo que es útil si se desea mostrar los gráficos en pantallas de diferentes tamaños.
- **Mayor libertad creativa:** La mayor resolución permite una mayor libertad creativa al diseñar los gráficos, ya que se pueden representar objetos y personajes con más detalle y realismo.



**Figura 12.0:** Comparación de las diferentes resoluciones más utilizadas de pixel art

**Fuente:** Isometric View - Pixel Art Style (+ drawing Room and Store) “Pixel Art Isometric #1” by Cheishiru - Make better art. (2021, May 31). Clip Studio TIPS. Retrieved February 3, 2023, from <https://tips.clip-studio.com/en-us/articles/4969>

## Programación Orientada a Objetos

Para poder comprender el modelo de datos que se utiliza en el sistema que está por detrás del videojuego *Spring Light*, es necesario en primer lugar introducir la programación orientada a objetos. Unity opera con el lenguaje de programación C#, el cual, si bien es un lenguaje multiparadigma, está fuertemente basado en la orientación a objetos.

La programación orientada a objetos es un paradigma de programación que utiliza el concepto de objetos, los cuales contienen información (campos) y funciones (métodos). Concebido por primera vez en 1967 con el lenguaje Simula 67, la programación orientada a objetos difiere de la tradicional en el sentido de que no se trata de instrucciones secuenciales que se le dan a la máquina, sino que se abstrae el comportamiento del sistema en objetos, lo cual permite alcanzar un mejor diseño con beneficios de simpleza, escalabilidad, reutilización y confiabilidad.

Dentro de lo que se conoce como programación orientada a objetos existen las buenas prácticas de diseño, las cuales son principios que se pueden aplicar a la hora de programar para alcanzar mejores resultados. De acuerdo con lo que dice Robert C. Martin<sup>54</sup>, algunos de los principios de diseño son:

- **Single Responsibility Principle:** Cada objeto o módulo en un sistema debe tener una única responsabilidad. Esto quiere decir también que debe tener una única razón para cambiar.
- **Open/Closed Principle:** Los objetos o módulos deben estar abiertos a extensión pero cerrados a modificación. Algo que se intenta a la hora de diseñar sistemas es adelantarse a la posibilidad de cambio, redefinición o extensión de las funcionalidades del mismo. Lo que este principio dice es que los objetos deben poder extenderse para agregar más comportamientos, pero no cambiar los que ya existen.

---

<sup>54</sup> Martin, R. C. (2000). Design Principles and Design Patterns. Design Principles and Design Patterns. Retrieved February 3, 2023, from [http://staff.cs.utu.fi/~jounsmmed/doos\\_06/material/DesignPrinciplesAndPatterns.pdf](http://staff.cs.utu.fi/~jounsmmed/doos_06/material/DesignPrinciplesAndPatterns.pdf)

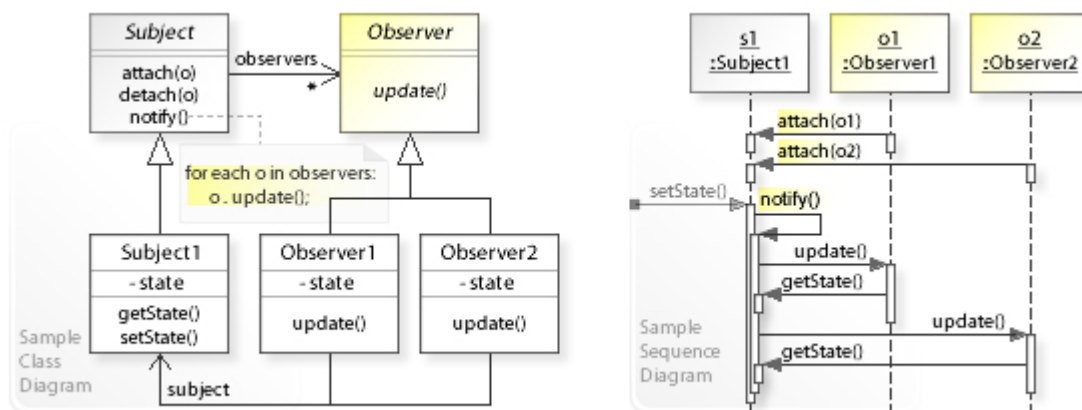


- **Liskov Substitution Principle:** Dice que los objetos o módulos de un programa deberían poder ser reemplazables por instancias o subtipos sin alterar el correcto funcionamiento del mismo. Esto causa que se desacople el objeto que realiza la acción de su implementación, lo cual ayuda a la hora de reutilizar código.
- **Interface Segregation Principle:** Las interfaces deben ser específicas para quienes las estén utilizando, y no tener una interfaz de propósito general. Esto oculta algunas partes o métodos del objeto que se está utilizando dependiendo de quien lo use, para evitar generar dependencias innecesarias.
- **Dependency Inversion Principle:** Se debe depender de abstracciones en lugar de implementaciones. Esto significa que al momento de relacionar dos objetos o módulos, se lo debe hacer a través de una interfaz en lugar de directamente. Esto lo que permite es cambiar la implementación que se utiliza en una de las dos partes sin alterar la otra parte del sistema.

Adicionalmente, sobre estos principios existen los patrones de diseño. Los patrones de diseño son una serie de técnicas o soluciones que se aplican sobre problemas recurrentes en la programación de objetos. A lo largo del desarrollo de *Spring Light* se utilizaron patrones de diseño en diversos momentos para ayudar a resolver dichos problemas de una forma más elegante y eficiente. A continuación se explican algunos de los patrones de diseño que fueron utilizados durante el desarrollo.

## Observer Pattern

Este patrón de diseño se utiliza cuando se tiene uno o más objetos que necesitan reaccionar frente a un cambio de estado de otro. En este caso, se aplica la inversión de dependencias para que el objeto observado mantenga una lista de los objetos que lo observan y los actualice cuando se de dicho cambio de estado. Este patrón es muy útil ya que permite además dinámicamente subscribirse y desuscribirse de el objeto observado, lo cual le otorga una gran flexibilidad. Su diseño UML se puede ver en la figura 13.0.



**Figura 13.0:** Esquema UML del patrón de diseño Observer

**Fuente:** Observer pattern. (n.d.). Wikipedia. Retrieved January 11, 2023, from [https://en.wikipedia.org/wiki/Observer\\_pattern](https://en.wikipedia.org/wiki/Observer_pattern)

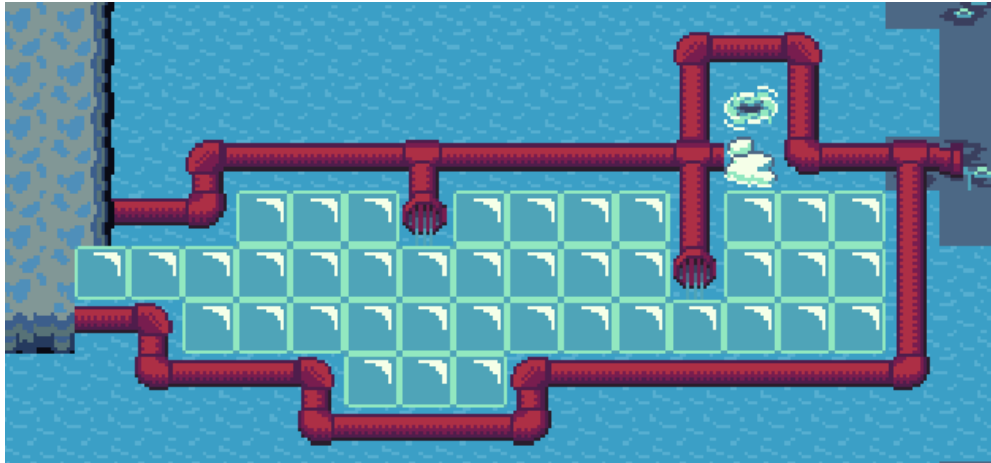
Este patrón fue utilizado en múltiples ocasiones durante el desarrollo. Ejemplos incluyen:

- **Ciclo dia/noche**

Se utilizó un patrón Observer para agregar a todos los componentes que debían ser alertados del cambio de día al objeto responsable de realizar dicha transición. Esto permite desacoplar los objetos entre sí e incluso agregarlos y quitarlos dinámicamente en tiempo de ejecución. Los cultivos, por ejemplo, pueden ser alertados del cambio de día para alterar su estado interno sin necesidad de que el componente que cambia el día conozca ninguna característica acerca de ellos más que un método que le permita alertarles de la transición.

- **Managers de puzzles**

Otra aplicación que se le dio a este patrón es en la gestión de puzzles. Muchas veces los puzzles están compuestos por elementos pequeños con los que el jugador debe interactuar para alcanzar cierto resultado. Estos elementos se agrupan dentro de un manager o gestor, para poder corroborar condiciones de victoria y reiniciar el estado del puzzle. En este caso también se hizo utilidad de un patrón observable para poder agregar los componentes al manager en tiempo de ejecución. Por ejemplo en la figura 14.0 a continuación el jugador debe caminar por sobre todos los bloques de hielo para que el puzzle se considere completo. Utilizando el patrón Observer es posible agregar y quitar cuantos bloques de hielo se necesiten sin necesidad de cambiar una sola línea de código.



**Figura 14.0:** Puzzle #5. El jugador debe caminar sobre todos los bloques de hielo para cumplir las condiciones de victoria.

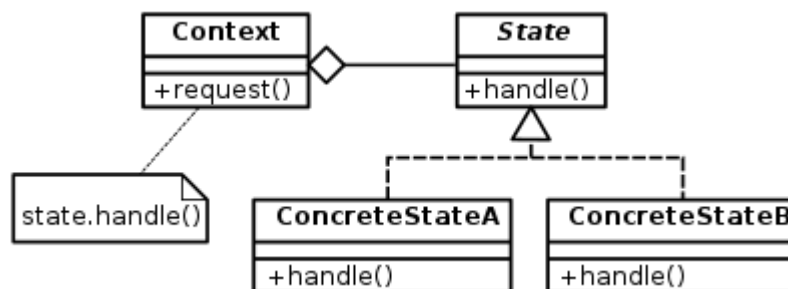
**Fuente:** Elaboración Propia

## State Pattern

Este patrón se cataloga dentro de los patrones de comportamiento. El concepto proviene originalmente de los autómatas de estado finito, los cuales son capaces de cambiar de un estado a otro dependiendo de las interacciones que se realicen con su interfaz. El patrón de diseño State, como se indica en la figura 15.0, permite que un objeto altere su comportamiento al cambiar su estado interno.

Cada estado representa una clase separada y al cambiar de estado, el objeto cambia su clase interna. Esto nos permite delegar responsabilidades y evitar un gran switch o if-else en caso de tener muchos estados diferentes. Permite a los objetos cambiar su comportamiento dinámicamente en tiempo de ejecución, lo cual es útil cuando queremos evitar acoplamientos fuertes y permitir cambios en el comportamiento de manera sencilla.

Este patrón se utilizó también en el desarrollo para el manejo del estado interno de los CropTiles. Esto se explica con mayor detalle más adelante en la sección 7.2, bajo el subtítulo “CropTiles”.

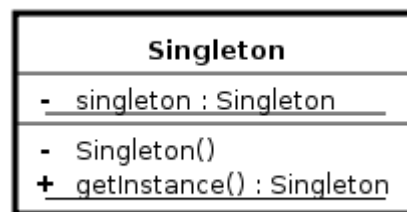


**Figura 15.0:** Esquema UML del patrón de diseño State

**Fuente:** State pattern. (n.d.). Wikipedia. Retrieved February 3, 2023, from [https://en.wikipedia.org/wiki/State\\_pattern](https://en.wikipedia.org/wiki/State_pattern)

## Singleton Pattern

El patrón singleton asegura que solo exista una instancia de una determinada clase en toda la aplicación y proporciona un punto de acceso global a ella. Es útil cuando queremos asegurar que solo haya una conexión a una base de datos o un único administrador de configuración en toda la aplicación, por ejemplo. Es una forma de mantener un control sobre la cantidad de instancias y evitar problemas de uso y acceso concurrente. Su diseño en UML se puede ver en la figura 16.0.



**Figura 16.0:** Esquema UML del patrón de diseño Singleton

**Fuente:** Singleton. (n.d.). Wikipedia. Retrieved February 3, 2023, from <https://es.wikipedia.org/wiki/Singleton>

En Unity en general y específicamente en el caso de Spring Light, este patrón es altamente utilizado para crear Managers o objetos que deben vivir entre varias escenas y que tienen una responsabilidad trascendente a ellas. Por ejemplo se utiliza para el manejo de la Skin<sup>55</sup> del jugador, el manejo de la persistencia del progreso cuando se guarda o carga el mismo, el manejo de la música, etc.

<sup>55</sup> Una "Skin" es una textura o descarga gráfica que cambia la apariencia de los personajes en los videojuegos. Son puramente estéticos: no aumentan las habilidades del personaje ni afectan el resultado del juego.

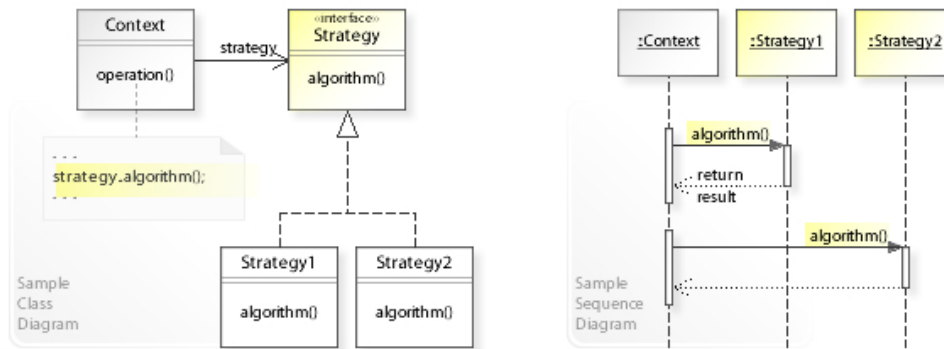
## Strategy Pattern

El patrón de diseño Strategy es un patrón de comportamiento que permite definir una familia de algoritmos, encapsular cada uno de ellos y hacerlos intercambiables. Es una forma de encapsular una lógica específica de una clase en una interfaz, permitiendo que diferentes implementaciones de esa interfaz se intercambien fácilmente en tiempo de ejecución.

El patrón Strategy, como se ve en la figura 17.0. se compone de tres elementos principales:

1. La interfaz "Strategy": Define los métodos que deben ser implementados por cada algoritmo.
2. Las clases "Concrete Strategy": Implementan la interfaz Strategy y proporcionan la lógica específica para cada algoritmo.
3. La clase "Context": Utiliza la interfaz Strategy para delegar el trabajo a una de las clases Concrete Strategy.

Esto permite que el código de la clase Context no dependa de un algoritmo específico, sino de la interfaz Strategy, lo que permite intercambiar fácilmente el algoritmo utilizado en tiempo de ejecución. Es muy útil cuando la lógica a implementar es muy compleja, y necesitamos cambiarla con facilidad.



**Figura 17.0:** Esquema UML del patrón de diseño Strategy

**Fuente:** Strategy pattern. (n.d.). Wikipedia. Retrieved January 26, 2023, from [https://en.wikipedia.org/wiki/Strategy\\_pattern](https://en.wikipedia.org/wiki/Strategy_pattern)

Este patrón fue muy útil principalmente para la serialización de los datos del juego, en donde se debía serializar de formas diferentes dependiendo del tipo de dato que existía. En este caso lo que se hizo es crear un serializador propio para cada tipo de dato que sigue una interfaz común provista por la librería JSON.NET, como se evidencia en la figura 18.0. Estos serializadores se encargan de tomar el objeto, serializarlo y luego deserializarlo cuando se carga el progreso, utilizando uno u otro dependiendo del tipo de objeto.



```
return new List<JsonConverter>()  
{  
    new ScriptableObjectConverter<EmptyObject>( folderPath: "" ),  
    new ScriptableObjectConverter<ProduceObject>( folderPath: "Produce" ),  
    new ScriptableObjectConverter<SeedObject>( folderPath: "Seeds" ),  
    new ScriptableObjectConverter<ToolObject>( folderPath: "Tools" ),  
    new ScriptableObjectConverter<WaterCanToolObject>( folderPath: "Tools" ),  
    new ScriptableObjectConverter<CropObject>( folderPath: "Crops" ),  
    new ItemObjectConverter(),  
};
```

**Figura 18.0:** Serializador Propio utilizando el patrón strategy

**Fuente:** Elaboración Propia

## 7. Desarrollo

### Fase 0: Capacitación

El proceso de desarrollo de Spring Light consistió en tres fases principales, las cuales sirvieron para organizar el trabajo en unidades autocontenidas y poder tener metas parciales alcanzables para cada una.

En primer lugar se realizó una capacitación, tanto en el desarrollo de videojuegos en general como en Unity y Aseprite. Para esto se consultaron múltiples videos<sup>56</sup> y se realizaron tres proyectos más pequeños a modo de poner en práctica los conocimientos aprendidos.

El primer juego creado consiste en una copia del popular “Flappy Bird<sup>57</sup>”. Flappy Bird es un juego casual para dispositivos móviles desarrollado y publicado por el desarrollador independiente Dong Nguyen en 2013. Consiste en controlar a un pájaro que vuela a través de una serie de tuberías sin tocarlas, mientras se desplaza hacia adelante automáticamente. El jugador tiene que tocar la pantalla para hacer que el pájaro salte y evitar las tuberías. El juego es simple, pero adictivo y desafiante debido a su alta dificultad.

Desde el punto de vista técnico este juego es un gran punto de partida para cualquiera que quiera comenzar a aprender un motor de videojuegos, ya que es extremadamente simple de realizar. Esto se debe a que permite una sola acción por parte del usuario y solo posee dos funciones, colisionar contra una tubería y reiniciar el juego. Su simpleza permite experimentar con otro tipo de elementos que normalmente uno necesita

---

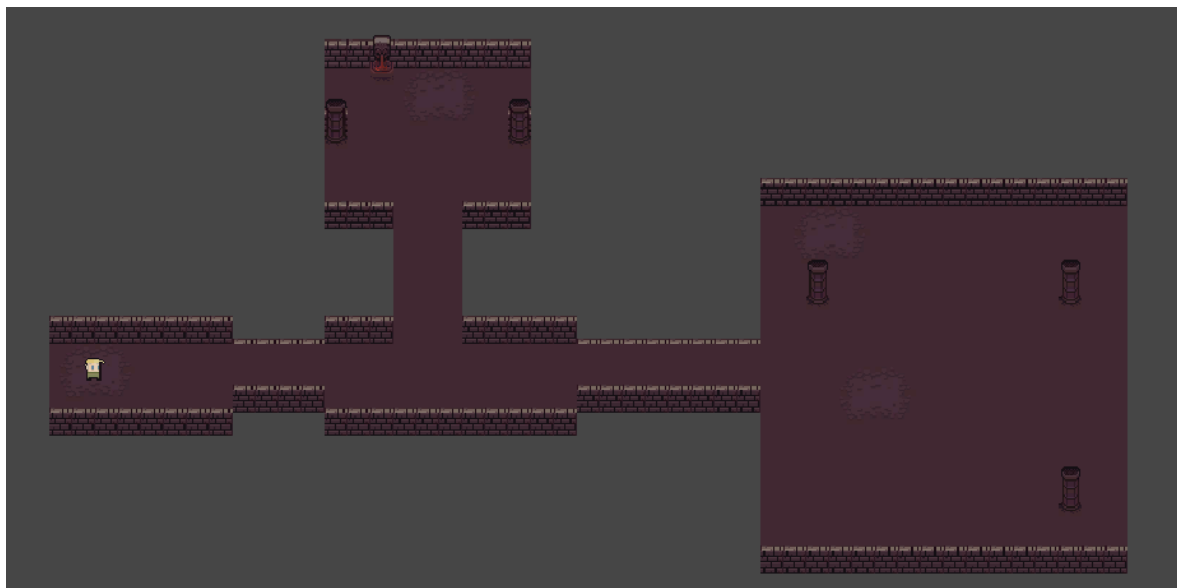
<sup>56</sup> Véase bibliografía y referencias

<sup>57</sup> Flappy Bird. (n.d.). Wikipedia. Retrieved January 26, 2023, from [https://es.wikipedia.org/wiki/Flappy\\_Bird](https://es.wikipedia.org/wiki/Flappy_Bird)

conocer en cualquier juego como la importación de texturas, las colisiones de elementos utilizando el motor de física, la actualización del estado del juego, etc.

Una vez finalizado este, se pasó a un tutorial diferente, el cual consiste en un juego mucho más cercano al del proyecto. Este consiste en un RPG con vista top down en un dungeon, incluyendo combate y animaciones. Este juego fue mucho más desafiante de crear, si bien el tutorial servía de gran apoyo para aprender las herramientas principales, como el mapa de Tile de Unity, el cual permite replicar un área de píxeles a lo largo de una superficie, para realizar con más facilidad los mapas.

Este proyecto fue muy importante para la capacitación, ya que permitió entender diversos elementos de un juego top down como el movimiento, animaciones, la interacción y los cambios de escena. Muchos de los elementos que se desarrollaron en esta etapa se lograron traducir inmediatamente al producto final y eso permitió ahorrar mucho tiempo de desarrollo. Las figuras 19.0, 20.0 y 21.0 muestran el resultado final de este proyecto.



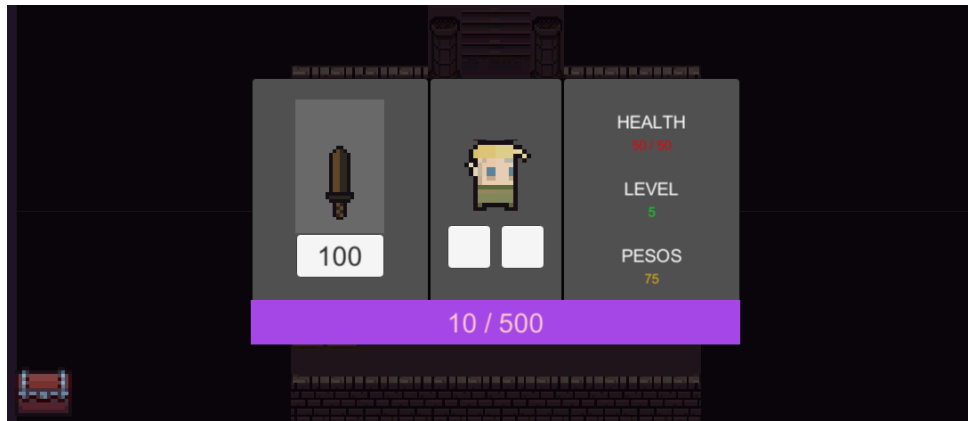
**Figura 19.0:** Juego Top Dungeon creado como capacitación, escena inicial

**Fuente:** Elaboración Propia



**Figura 20.0:** Juego Top Dungeon creado como capacitación, segunda escena

**Fuente:** Elaboración Propia



**Figura 21.0:** Juego Top Dungeon creado como capacitación, Menu UI

**Fuente:** Elaboración Propia

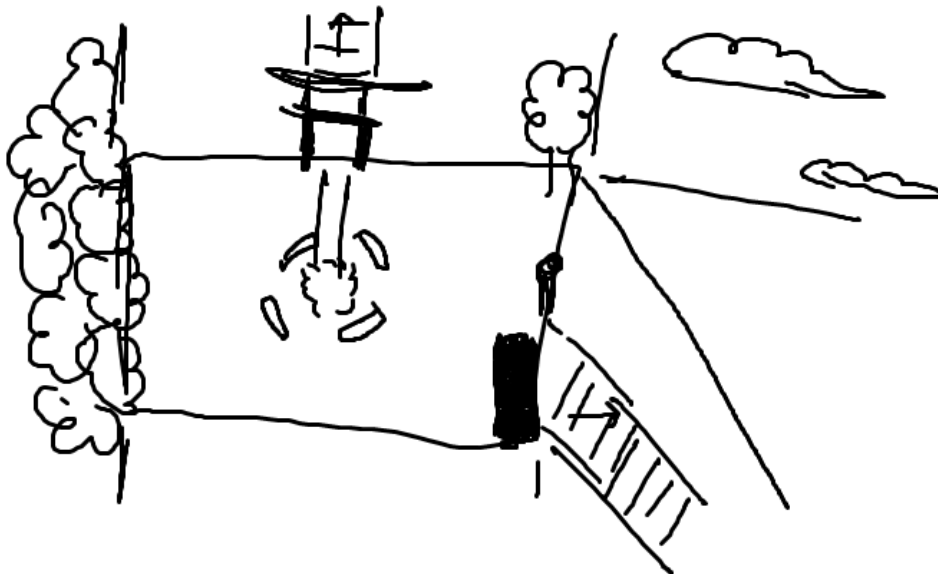
Por último, se siguió un último tutorial para crear un juego con un sistema de grilla para el movimiento, lo cual era la única característica que no estaba incluida en el tutorial anterior. Este también fue de gran importancia. Se probaron diversas formas de controlar el jugador y de interactuar con los objetos del mundo. Una vez probadas las estrategias, se pudo simplemente utilizar el mismo sistema para Spring Light, incluso reutilizando algunos assets visuales. Este proyecto se puede ver en la figura 22.0.



**Figura 22.0:** Juego en cuadrícula sin nombre creado como capacitación

**Fuente:** Elaboración Propia

En lo que respecta al pixel art, se investigó y se capacitó en las tecnologías de arte digital para su producción. Para esto se hizo uso de una tableta Wacom y el programa Aseprite, siguiendo tutoriales de Youtube<sup>58</sup>. Muchos de los assets, como la figura 26.0 y 27.0, creados en esta etapa sirvieron tanto para el proyecto de capacitación como para el producto final. Otros sirvieron de arte conceptual para definir la estética del juego, como la figura 25.0. También se crearon borradores de las escenas del juego como parte de la fase de planeamiento, como se ve en las figuras 23.0 y 24.0.



**Figura 23.0:** Borrador del diseño del mapa, escena del Lobby

**Fuente:** Elaboración Propia

<sup>58</sup> Véase bibliografía y referencias



**Figura 24.0:** Borrador del diseño del mapa, escena de la granja

**Fuente:** Elaboración Propia



**Figura 25.0:** Arte Conceptual inicial del juego

**Fuente:** Elaboración Propia





**Figura 26.0:** Arte conceptual inicial del juego

**Fuente:** Elaboración Propia



**Figura 27.0:** Logo del juego

**Fuente:** Elaboración Propia

## Fase 1: Prototipado

Como segunda fase del proyecto se creó un prototipo del juego utilizando los conocimientos aprendidos durante la capacitación. El motivo de crear un prototipo primero fue para implementar un metodología similar al agile o iterativo, en donde cada funcionalidad que se implementa se lanza a producción para que se pueda probar, evaluar y realizar eventuales correcciones antes de comenzar con una nueva funcionalidad. Esto permite conseguir atrapar fallas e inconsistencias en el producto cuando este es más pequeño y se pueden resolver más fácilmente. Si el producto se implementara todo junto antes de ser probado por los usuarios sería mucho más difícil realizar cambios.

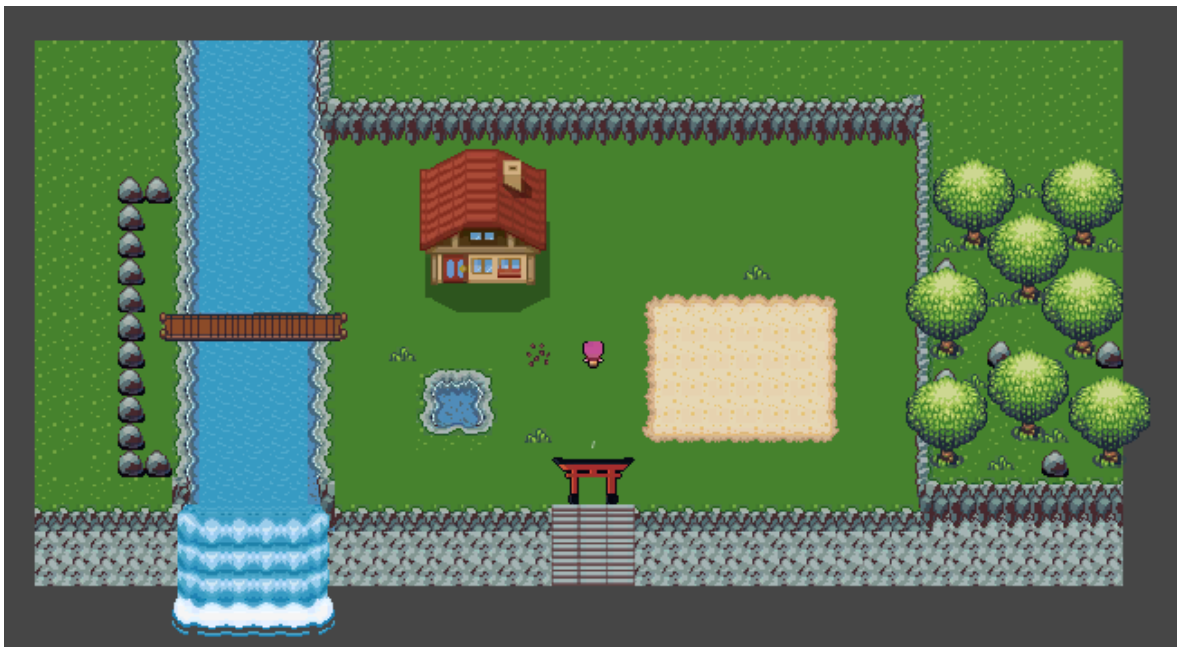
Este prototipo incluye únicamente las funciones más básicas del juego completo, a modo de que pudieran ser probados por usuarios para recibir constante retroalimentación. El primer prototipo incluye solamente un mapa modelo con un personaje el cual se podía mover dentro de una grilla con el teclado. Sin aún tener siquiera un archivo ejecutable se dio a probar a tres usuarios dentro del editor de Unity mismo.

Todos los testers acordaron que el movimiento del personaje se sentía raro, ya que si se estaba moviendo al personaje de dirección era difícil cambiar la dirección del movimiento sin soltar la tecla de caminar. Además, aunque el juego no lo requiriera todos acordaron que era importante poder tener un movimiento en diagonal presionando dos teclas a la vez. Se realizaron los ajustes necesarios y se repitió el proceso hasta que los usuarios quedaron conformes.

Una vez terminado esto fue necesario desarrollar simultáneamente assets pixel art para el juego, animaciones y ciertos elementos como el inventario. Para esto se aplicó la misma fórmula que antes, realizando pruebas con usuarios durante el desarrollo para corregir cómo se sentía el juego para un jugador real. Esta modalidad de trabajo proviene de la filosofía Agile, en donde se intenta crear pequeñas mejoras en forma de iteraciones por sobre lo que ya está creado e ir probando su correcta funcionalidad. La primera versión del juego se puede ver en la figura 32.0.

Así se terminaron las animaciones y las mecánicas del inventario y del ciclo de la granja. Estas dos funcionalidades fueron las que más esfuerzo requirieron, ya que son las mecánicas esenciales del juego y su complejidad es considerable. Afortunadamente, existen muchos tutoriales sobre como lograr cosas similares en Unity, con lo cual fue posible basarse sobre varios de estos para lograr resultados mejores y de mucha mayor calidad.

Para el sistema de inventario se decidió utilizar un sistema que provee Unity llamado Scriptable Object. Un Scriptable Object en Unity es un tipo de objeto que se utiliza para almacenar datos y configuraciones en un formato que puede ser fácilmente modificado y compartido entre diferentes scripts y componentes en un juego o aplicación. Estos objetos pueden ser creados como assets independientes en el proyecto de Unity, lo que permite una mayor flexibilidad y modularidad en el desarrollo. Además, al ser objetos de Unity, pueden ser fácilmente modificados en tiempo de ejecución, lo que permite una mayor capacidad de personalización y adaptación en el juego.

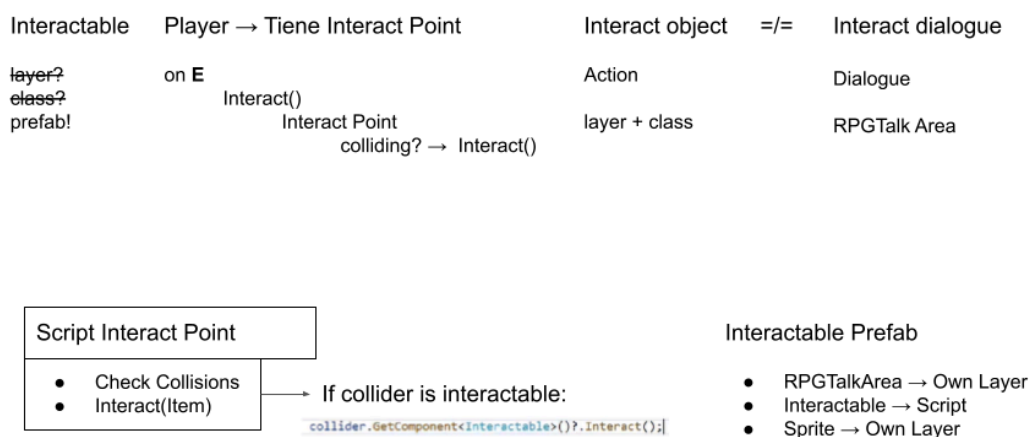


**Figura 32.0:** Escena de la granja en sus primeras versiones

**Fuente:** Elaboración Propia

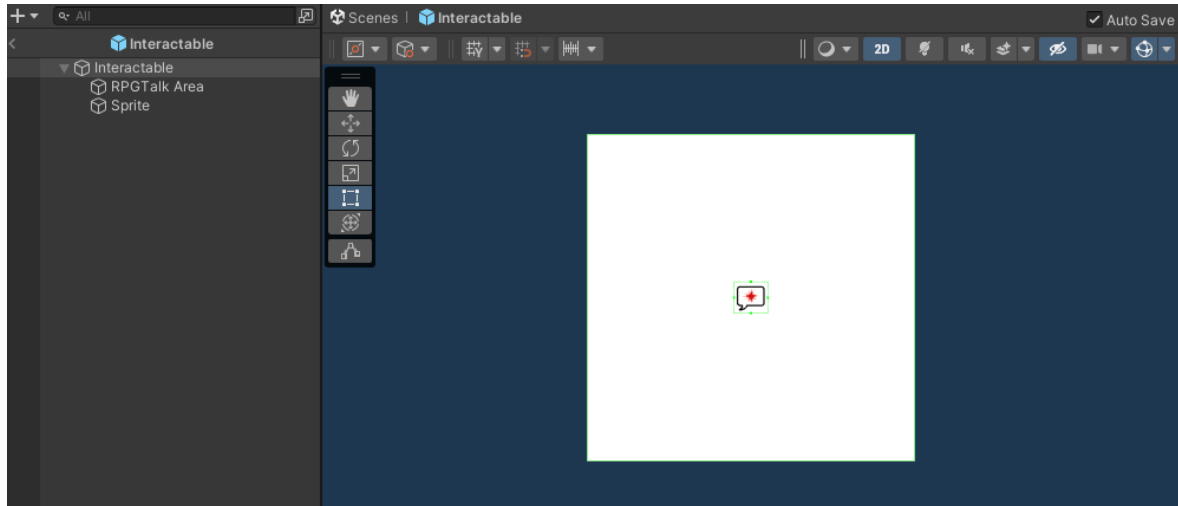
## Inventario: Interactable Objects vs Item Objects

El sistema de inventario, si bien parece una sola mecánica, esconde detrás dos módulos diferentes. En primer lugar existen los objetos en el mundo con el cual el jugador interactúa. Podemos denominar estos objetos “Interactables”. Esto es lo primero que se debió implementar. Se decidió que se utilizaría la tecla E para interactuar con los objetos. El diseño debía ser suficientemente general como para poder aceptar una gran variedad de acciones diferentes que puede tener un objeto al interactuar con el. Algunos objetos interactuables pueden agregarse al inventario, otros pueden activarse o tener efectos en el mundo, mientras que otros simplemente pueden abrir un cuadro con texto. En la figura 33.0 se puede ver un borrador de lo que se planeó para este sistema. Originalmente se había realizado un bosquejo a mano alzada y luego se pasó en computadora para el informe. El resultado de este mecanismo se puede ver en la figura 34.0 y 35.0.



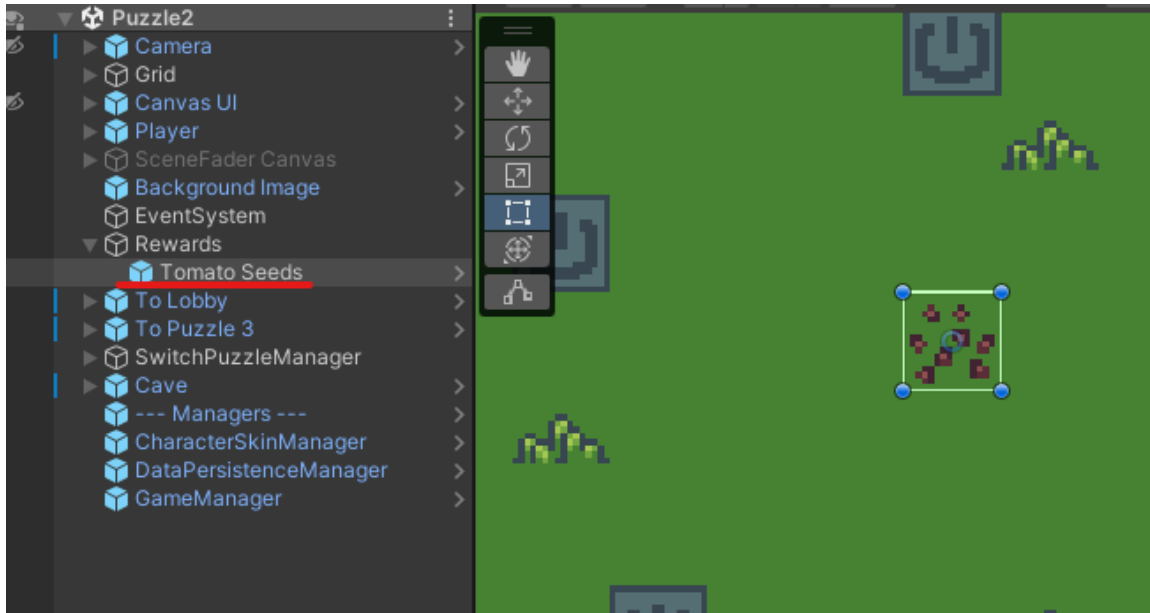
**Figura 33.0:** Borrador del esquema de objetos interactuables

**Fuente:** Elaboración Propia



**Figura 34.0:** Resultado final del Prefab Interactable.

**Fuente:** Elaboración Propia



**Figura 35.0:** Utilizando el Prefab Interactable para representar unas semillas de tomate. En este caso se le aplica la imagen y el componente Item que le otorga las características y permite que el jugador lo obtenga en su inventario.

**Fuente:** Elaboración Propia

El jugador cuenta con un objeto que se encuentra delante suyo en todo momento en la dirección que está viendo. Este objeto se denomina “Interact Point” y es invisible al jugador, únicamente utilizado como mecánica. Si este objeto se encuentra colisionando con un objeto interactuable al presionar la tecla E, se dispara la interacción. Para los casos en los que se utiliza un cuadro de diálogo, se decidió utilizar la librería RPGTalk<sup>59</sup>, la cual es muy utilizada en estos casos. El objeto interactuable se convierte así en un *Prefab* además de una clase de objeto, lo cual permite una alta personalización de la acción que se realiza.

<sup>59</sup> RPGTalk | GUI Tools. (n.d.). Unity Asset Store. Retrieved January 11, 2023, from <https://assetstore.unity.com/packages/tools/gui/rpgtalk-73392#content>

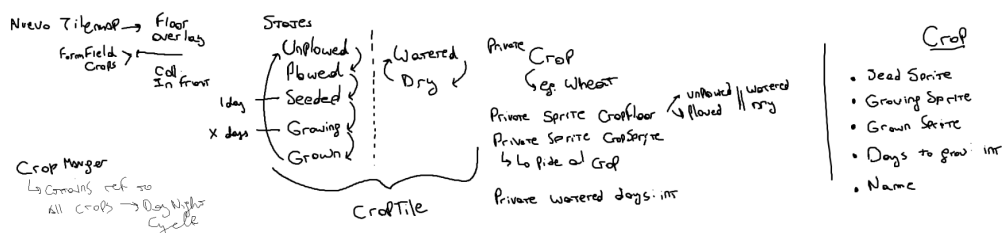
Un prefab en Unity es una forma de almacenar objetos y componentes en un solo archivo y poder reutilizarlos en diferentes partes del juego sin tener que crearlos desde cero cada vez. Es como una plantilla que puedes guardar y volver a usar repetidamente. Se considera como una instancia de un GameObject y sus componentes son guardados como un asset independiente.

Un prefab puede ser reutilizado varias veces en diferentes partes de un juego o escena. Los cambios realizados en una instancia de prefab se reflejarán en todas las instancias del mismo. Es una herramienta muy utilizada para crear objetos de juego reutilizables como por ejemplo, balas, explosiones, enemigos, etc. También se utilizan para crear un sistema de configuraciones de un juego. De esta forma, se puede ahorrar tiempo y asegurar de que todos los objetos sean consistentes en cuanto a su apariencia y comportamiento.

Una vez completado esto solo restaba la segunda mecánica para completar el sistema de inventario. Esta consistía en un contenedor que pudiera registrar los objetos guardados y con qué objeto se estaba interactuando en el mundo. Para esto se creó el concepto de ItemObject, el cual también hace uso de los Scriptable Objects. Un ItemObject es un objeto abstracto el cual puede concretizarse en diferentes tipos de objetos según su comportamiento. Se dividieron los Items del juego en Tools, Produce, Seeds y Crops. Al utilizar Scriptable Objects, se pueden crear muchos assets de cada tipo con facilidad a través del menú de Unity.

## CropTiles

Para el sistema de granja se decidió utilizar un State pattern (máquina de estado) para representar las diferentes etapas que debe atravesar un cultivo para crecer y poder ser cosechado y plantado nuevamente. Esto se planeó abstractamente antes de su implementación, como se muestra en la figura 36.0



**Figura 36.0:** Borrador original del esquema de objetos de tipo cultivo

**Fuente:** Elaboración Propia

El objeto "CropTile" es el encargado de representar una celda para plantar. El estado de este se mantiene mediante un State pattern y la información del Crop que se está plantando se guarda en un Scriptable Object separado que registra la información como el nombre, la imagen de la semilla, la imagen durante el crecimiento, los días que tarda en crecer, etc.

Los CropTiles siempre comienzan en estado Unplowed. En este estado no se puede realizar ninguna operación sobre ellos a excepción de ararlos con la herramienta "Hoe". Esto es porque se considera tierra infértil que no es apta para cultivos. Luego de arar la tierra, se puede interactuar con cualquier objeto de tipo semillas para plantarlas. Además, permite interactuar con la regadera, en cualquier estado del CropTile. Este estado de regado desaparece con la transición de día. Si se riegan las semillas y se pasa de día,



estas cambian a estado Growing. Este estado presenta una imagen del cultivo que se plantó diferente para cada tipo de semilla y se debe regar cada día para que avance su crecimiento. Si se pasan los días sin regarlas, la planta no crecerá. Después de una determinada cantidad de días regados, el CropTile en estado Growing cambiará a estado Grown y así lo hará también su imagen. En este estado, el jugador puede utilizar la herramienta “Scythe” para cosechar la planta y obtener su cosecha, la cual puede luego vender. Este ciclo se muestra en la figura 37.0.



**Figura 37.0:** CropTiles en la granja. De izquierda a derecha se pueden ver cada uno de los estados posibles que tiene un mismo CropTile.

- 1: Unplowed
- 2: Plowed
- 3: Seeded
- 4: Growing
- 5: Grown
- 6: Vuelve a comenzar

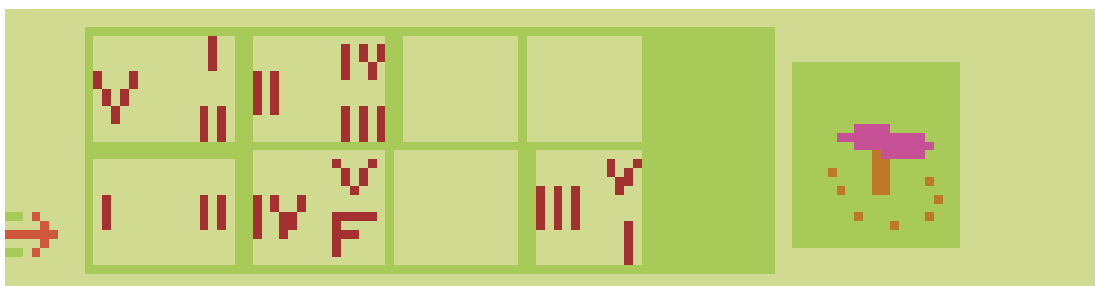
**Fuente:** Elaboración Propia

## Puzzles

El juego cuenta con cinco distintivos rompecabezas (o puzzles) que el jugador puede resolver para obtener distintos resultados. Estos fueron planeados de antemano previa a su implementación:

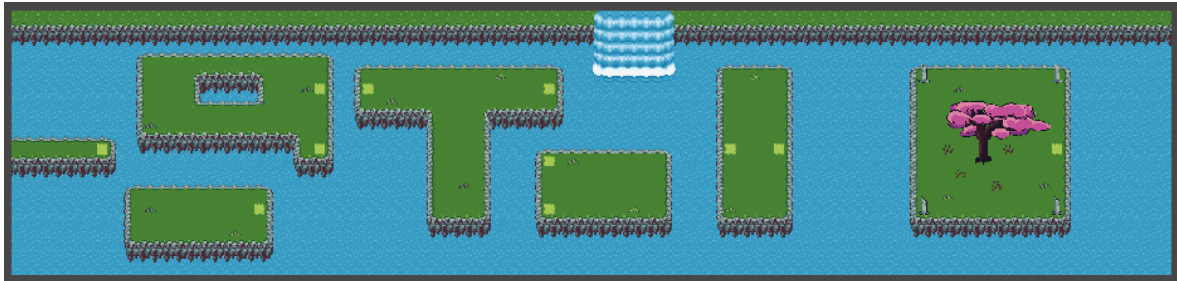
El Puzzle número uno se planeó como un laberinto de islas, en las cuales el jugador debe decidir entre opciones de plataformas que lo mueven a otra isla. Este puzzle está inspirado en uno de los puzzles del Pokemon Ruby, en donde una serie de túneles bajo tierra se conectan entre sí por aberturas que a veces retroceden y a veces avanzan en el progreso del camino.

En la figura 37.0 se puede ver el plan original de las islas con las opciones del jugador. Este comienza en la isla marcada con un “I” a su izquierda en donde se encuentra la flecha naranja. Las conexiones se realizan por número, es decir, por el número “II” a la derecha se llega a la isla que tiene marcado “II” a la izquierda y así sucesivamente. El camino marcado con una “F” es el camino que lleva a la isla Final y esta se representa con el árbol cerezo rosado y las semillas que son las recompensas del jugador. La figura 38.0 muestra el resultado final del puzzle.



**Figura 37.0:** Borrador del diseño del Puzzle numero 1

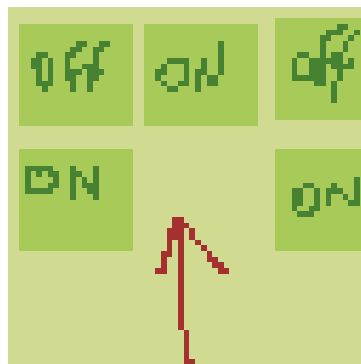
**Fuente:** Elaboración Propia



**Figura 38.0:** Resultado final del Puzzle numero 1

**Fuente:** Elaboración Propia

El segundo puzzle se diseñó como un juego de rompecabezas en el que se encuentran en el mundo ciertos dispositivos que pueden estar encendidos o apagados. Este diseño se muestra en la figura 39.0. Al interactuar con ellos, estos se cambian de estado a si mismos y cambian a su vez a sus dos vecinos. El objetivo del juego es lograr prender todos los dispositivos. Al hacerlo se revela en el centro la recompensa del jugador. El resultado final se puede ver en la figura 40.0.



**Figura 39.0:** Borrador del diseño del Puzzle numero 2

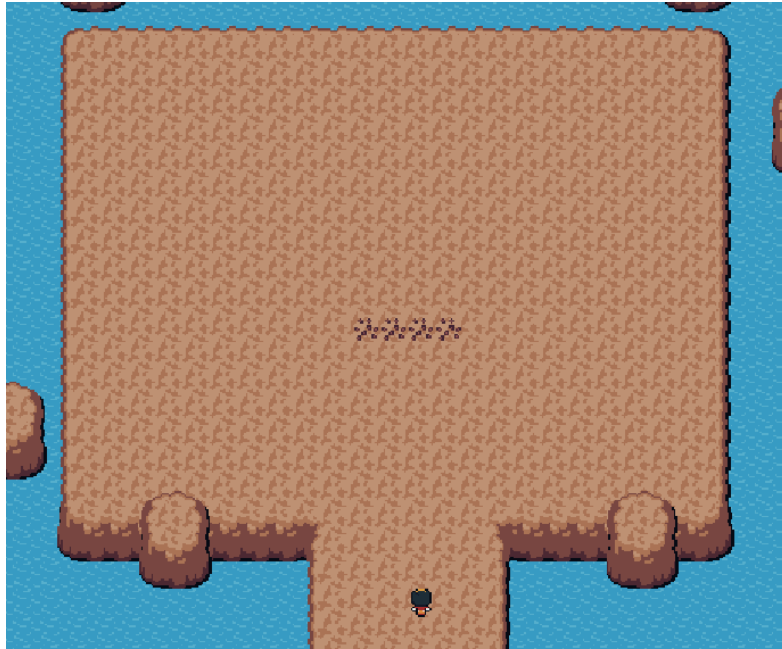
**Fuente:** Elaboración Propia



**Figura 40.0:** Resultado final del Puzzle numero 2

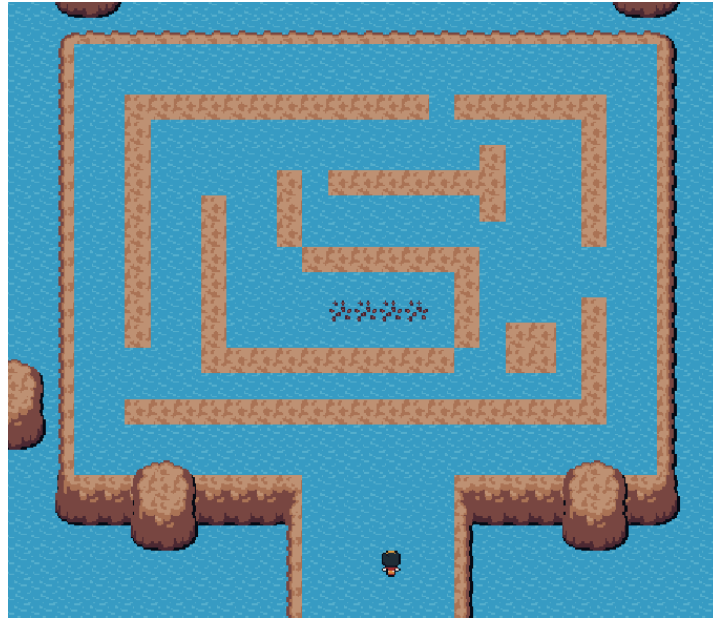
**Fuente:** Elaboración Propia

El tercer puzzle fue inspirado también en uno de los juegos de la saga de Pokemon, y consta de un laberinto invisible contra el que el jugador colisiona al intentar alcanzar la recompensa, la cual es plenamente visible. Para resolverlo, este debe maniobrar alrededor de las paredes utilizando el personaje para probar si hay un camino o una pared. Este se puede ver en la figura 41.0 y 42.0.



**Figura 41.0:** Puzzle numero 3

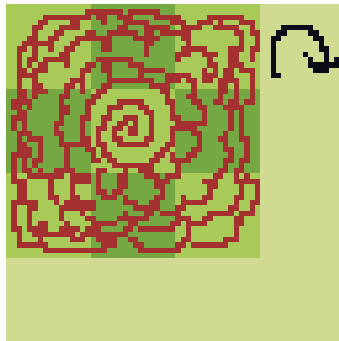
**Fuente:** Elaboración Propia



**Figura 42.0:** Puzzle número 3 en donde se muestran las paredes que normalmente son invisibles al jugador

**Fuente:** Elaboración Propia

El cuarto puzzle se trata de un rompecabezas de estilo giratorio. El jugador se encuentra con una imagen de una flor y debe rotar sus piezas para lograr que se forme la figura. Al hacerlo, aparece la recompensa. El borrador se muestra en la figura 43.0 mientras que el resultado final se puede ver en la figura 44.0 y 45.0.



**Figura 43.0:** Borrador del diseño del Puzzle numero 4

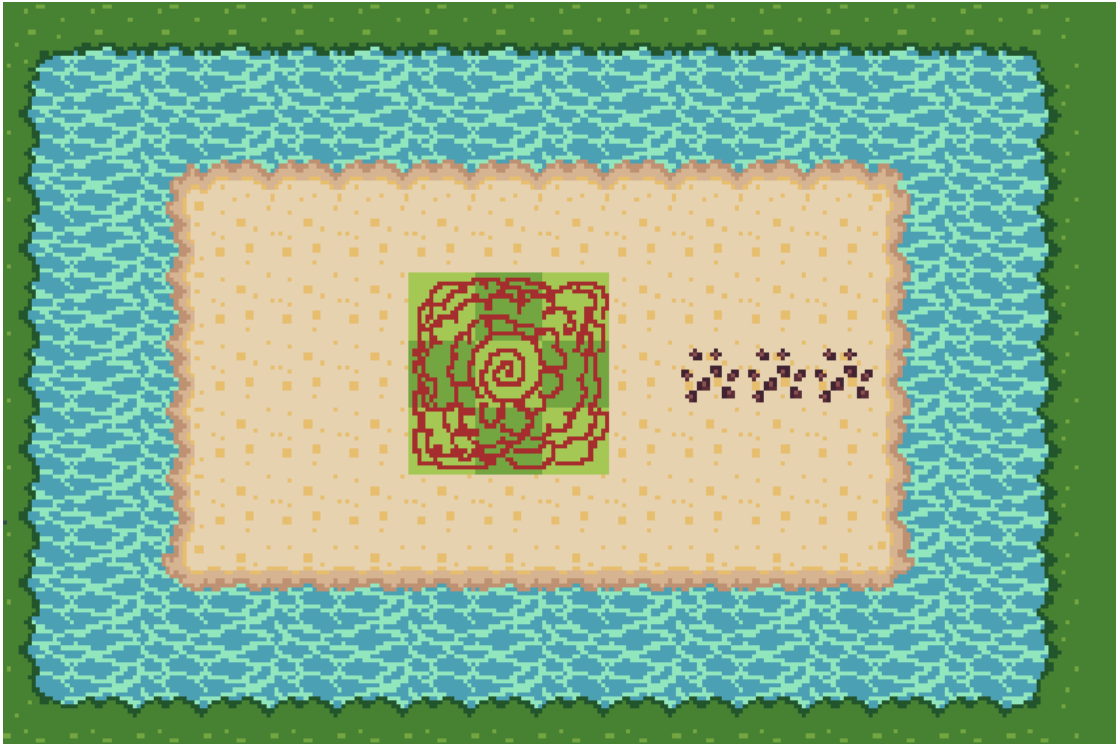
**Fuente:** Elaboración Propia



**Figura 44.0:** Puzzle #4

**Fuente:** Elaboración Propia

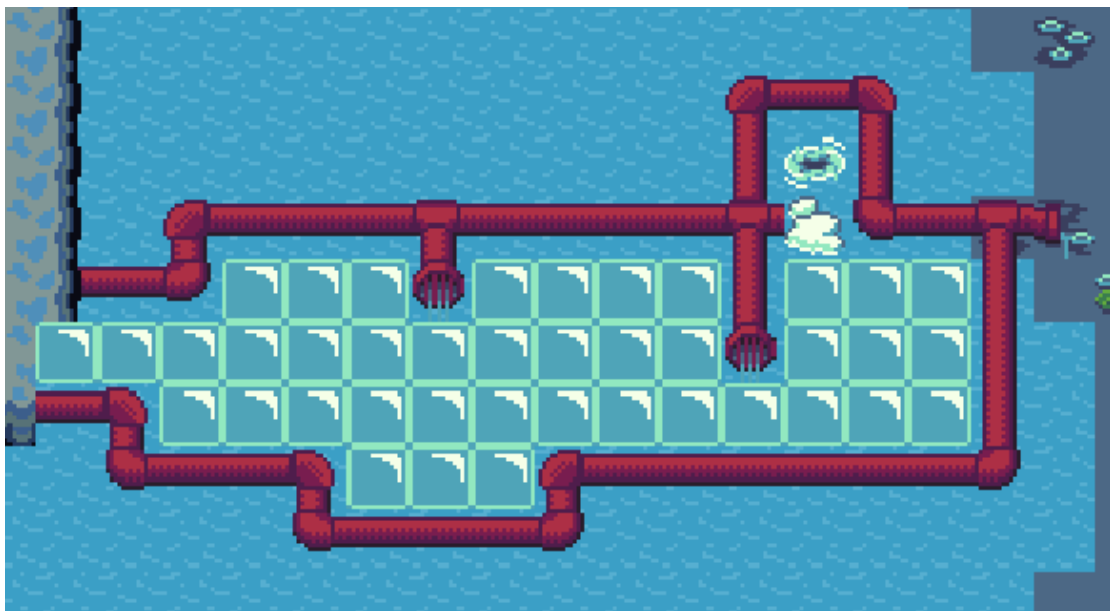




**Figura 45.0:** Puzzle #4 resuelto con sus recompensas

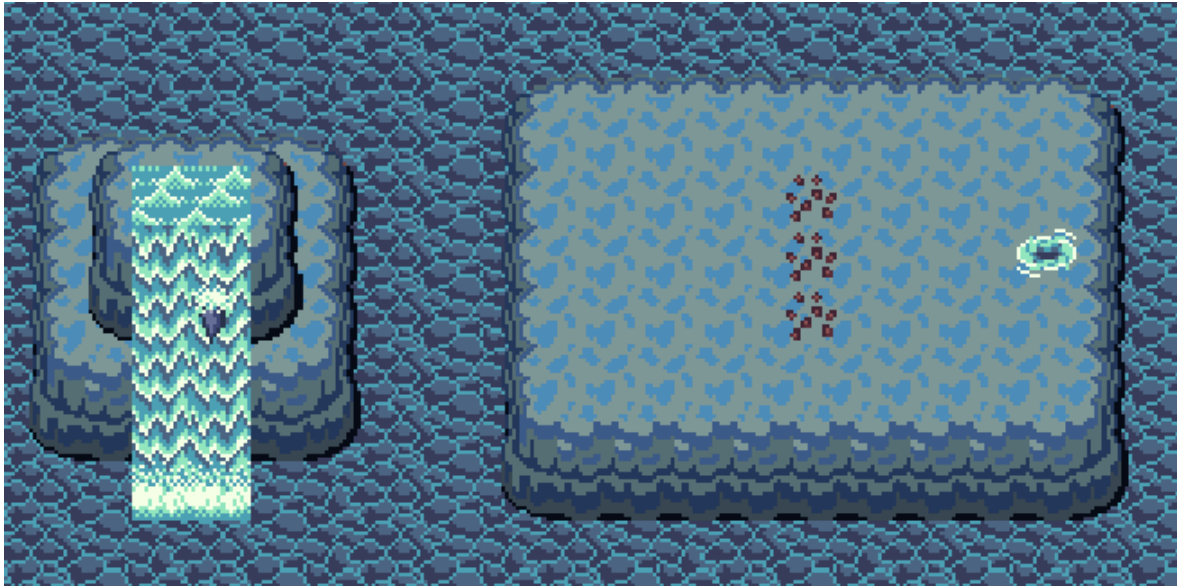
**Fuente:** Elaboración Propia

Por último, para el Puzzle #5 se decidió hacer un laberinto de plataformas de hielo, las cuales desaparecen luego de caminar sobre ellas, como se muestra en la figura 46.0. Al llegar al final del camino, si se hacen desaparecer todas las plataformas el jugador será teletransportado a una isla en donde está la recompensa, lo cual se ve en la figura 47.0.



**Figura 46.0:** Puzzle #5

**Fuente:** Elaboración Propia



**Figura 47.0:** Isla final

**Fuente:** Elaboración Propia

Los puzzles son una parte importante del juego, ya que proveen una sensación de mundo abierto, permitiendo al jugador recorrer los distintos mapas en el orden en que quiera y, al resolverlos, lo recompensa otorgándole Items que no podría conseguir de otra manera.

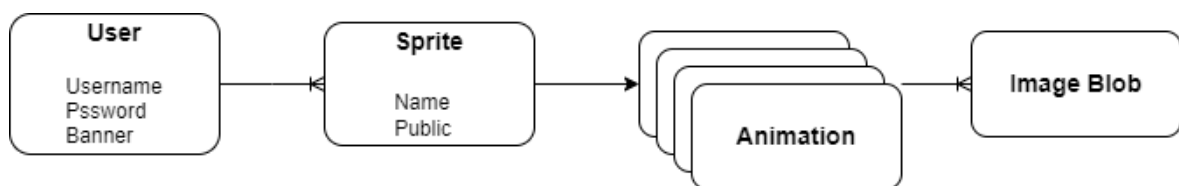
Se consiguió asimismo dos jugadores nuevos para probar los puzzles y dar su opinión respecto a su implementación, tanto en dificultad como en intuitividad de la resolución. Así se determinó, por ejemplo, que era importante ajustar las posiciones del jugador cuando se realizaban teletransportaciones, ya que si no se volvía exactamente al principio del puzzle a veces no quedaba claro que había sucedido.

## Fase 1.2: Desarrollo de la plataforma web

La plataforma web, llamada “Marketplace de Assets”, se desarrolló en dos partes. La primera parte consta del desarrollo del servidor web. Para esto fue necesario, al igual que con el videojuego, capacitarse primero en NestJs, ya que es un framework con el cual nunca se había trabajado. Además, fue necesario capacitarse en su integración con PostgreSQL. Esto se realizó utilizando el tutorial documentado que figura en la página oficial del framework, el cual guía al desarrollador por las diferentes funcionalidades que provee el software.

En este caso afortunadamente ya se conocía el lenguaje de programación Typescript, con lo cual no hubo necesidad de aprenderlo nuevamente.

Luego de realizar esto se comenzó rápidamente con la implementación de las funcionalidades más importantes, el dibujador y la carga de Sprites por medio de imágenes. Para esto se tomó una modalidad de trabajo similar a la del videojuego, en el sentido de que se hicieron primero las funcionalidades más importantes para poder ir probándolas a medida que se trabajara. Se definió el modelo de datos para las entidades que maneja la aplicación. Esto se puede ver en la figura 64.0 a continuación.



**Figura 64.0:** Modelo de Datos para el backend de la plataforma web

**Fuente:** Elaboración Propia

Además, se creó desde el comienzo el servidor en Heroku, el cual gracias al plan de estudiantes se pudo utilizar de forma gratuita. Por medio de un pipeline de deployment se

utilizó Github Actions para realizar la actualización del código del servidor de Heroku cada vez que se realizaban cambios al servidor. Esto se hizo de la misma manera también con la página de React en Netlify, la cual se comenzó al mismo tiempo para poder probar la integración end-to-end de todo el producto. El desarrollo local se llevaba a cabo entonces en un contenedor de Docker, mientras que el producto se publicaba de manera constante.

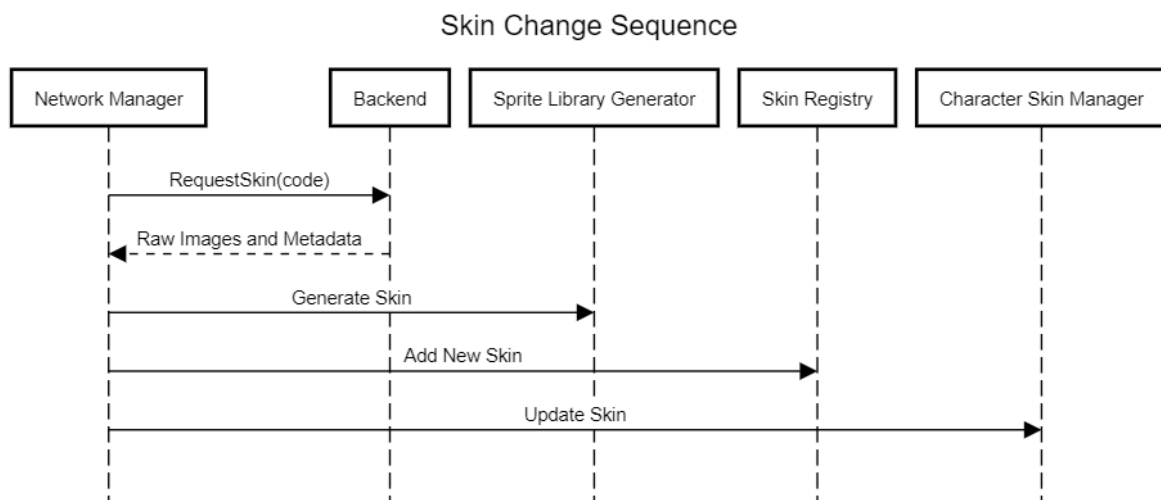
El dibujador presentó ciertas dificultades a la hora de implementarse, ya que se debía crear múltiples imágenes para poder formar animaciones en cada una de las cuatro vistas del personaje. Además, se debe tomar estos datos en forma de valores RGBA (Red, Green, Blue y Alpha) y convertirlos en una imagen rasterizada, con su codificación particular. Esto finalmente se logró convirtiendo cada píxel de la imagen en una cadena de bytes y utilizando el conjunto de los bytes para transformarlo en imagen.

De manera similar se creó también la funcionalidad de subir imágenes para formar sprites y la de edición de sprites. En esta etapa fue que se descubrió una poderosa funcionalidad que no se había tenido en cuenta originalmente. Por la forma en que se había implementado la creación y edición de las texturas la plataforma permitía cargar una textura en forma de imágenes sueltas y luego editarlas utilizando el dibujador para realizar retoques. Esto es de gran utilidad para el usuario, ya que puede suceder que las imágenes originales no sean exactamente lo que el usuario quería lograr, y es capaz en este caso de realizar las modificaciones que quiera.

Las demás funcionalidades se implementaron sin contratiempos. Para todas las funcionalidades se utilizó el mismo sistema que para el desarrollo del juego. Es decir, se establecieron los objetivos de las próximas dos semanas de trabajo y se llevaron a cabo, realizando el deploy al finalizar las funcionalidades para poder probarlas en el ambiente integrado.

## Fase 1.3: Integración de la plataforma web en Unity

La segunda parte del desarrollo de la plataforma web se realizó en Unity. En esta fase la clave consiste en realizar la conexión entre el juego y el servidor web. Esto se pudo implementar a través de una librería que provee Unity para realizar conexiones remotas a través de la web. Se le asignó a cada skin creada por un usuario un código único. Este código, al ser ingresado en el videojuego, se le envía al servidor para que este responda con las imágenes y toda la información necesaria para que el juego logre integrar una skin utilizable como componente. El flujo de secuencia que realiza el juego al recibir un código se puede ver en la figura 65.0.

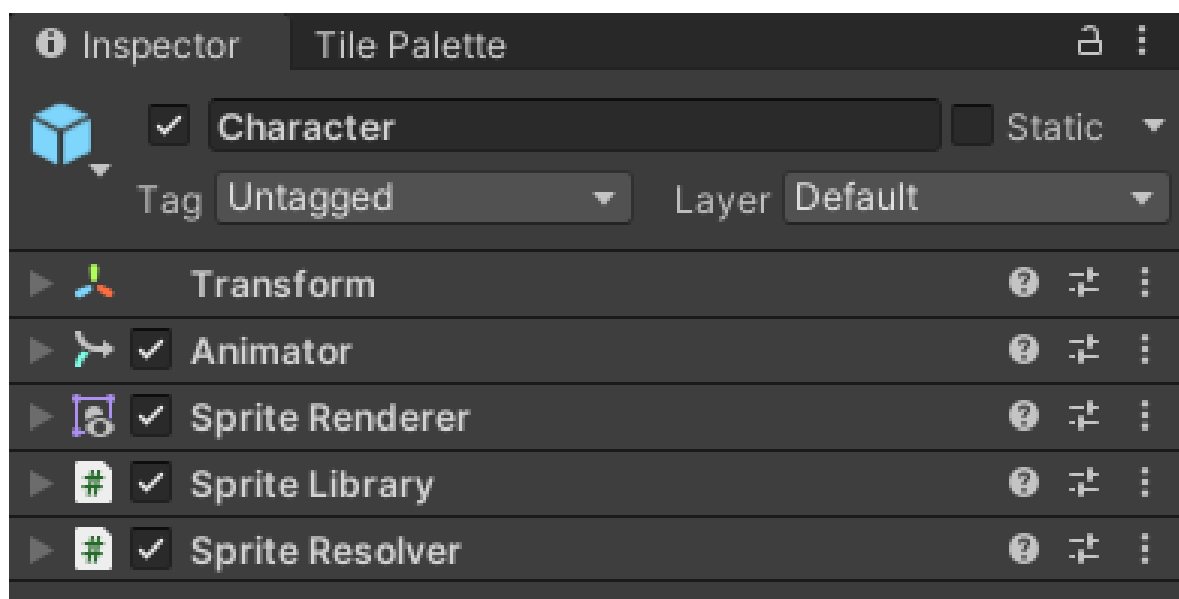


**Figura 65.0:** Diagrama de secuencia para la descarga de una nueva skin desde el backend

**Fuente:** Elaboración Propia

En la creación de la skin intervienen tres componentes principales. Estos son el Sprite Library Generator, el Skin Registry y el Character Skin Manager. Luego de crear la skin hay otros tres componentes que juegan un rol importante en utilizar esa skin. Provistos

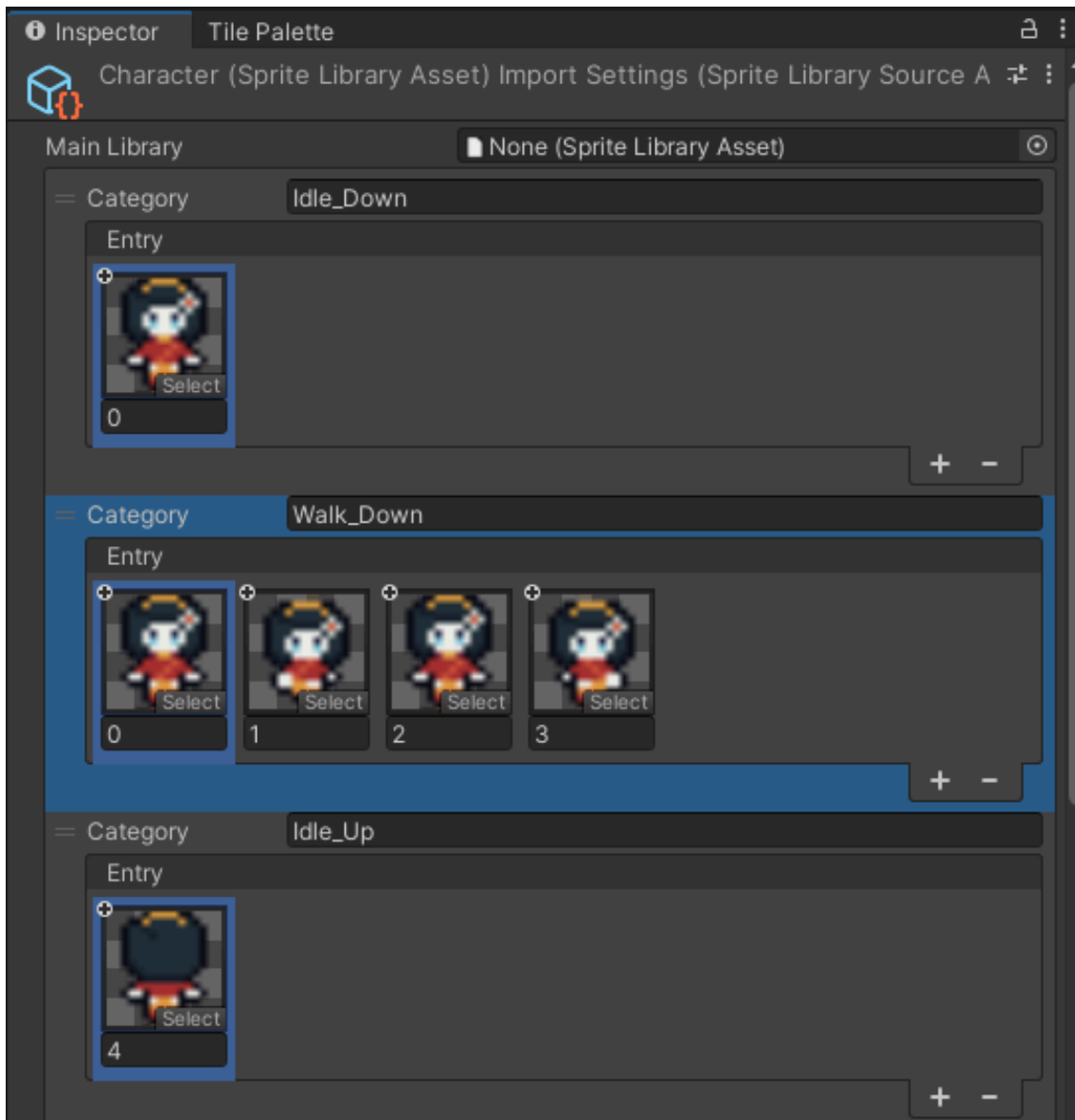
por Unity y se adecuan muy bien a las necesidades del proyecto. Estos son el Sprite Resolver, Sprite Library y Sprite Renderer. Todos estos componentes deben agregarse al objeto del jugador para que puedan trabajar en conjunto para manejar el estado de la skin. Esto se muestra en la figura 66.0.



**Figura 66.0:** Objeto en Unity que representa el jugador

**Fuente:** Elaboración Propia

Cuando el juego obtiene las imágenes del servidor, el Sprite Library Generator genera con ellas lo que se conoce como un Sprite Library Asset. Este es un tipo de Asset especial que cataloga las imágenes en una estructura predeterminada de categorías y etiquetas. Estas categorías se asignan para cada vista y cada animación diferente que tenga la skin y las etiquetas se utilizan para marcar el número de frame al que corresponde cada imagen dentro de su categoría. Esto se muestra en la figura 67.0.

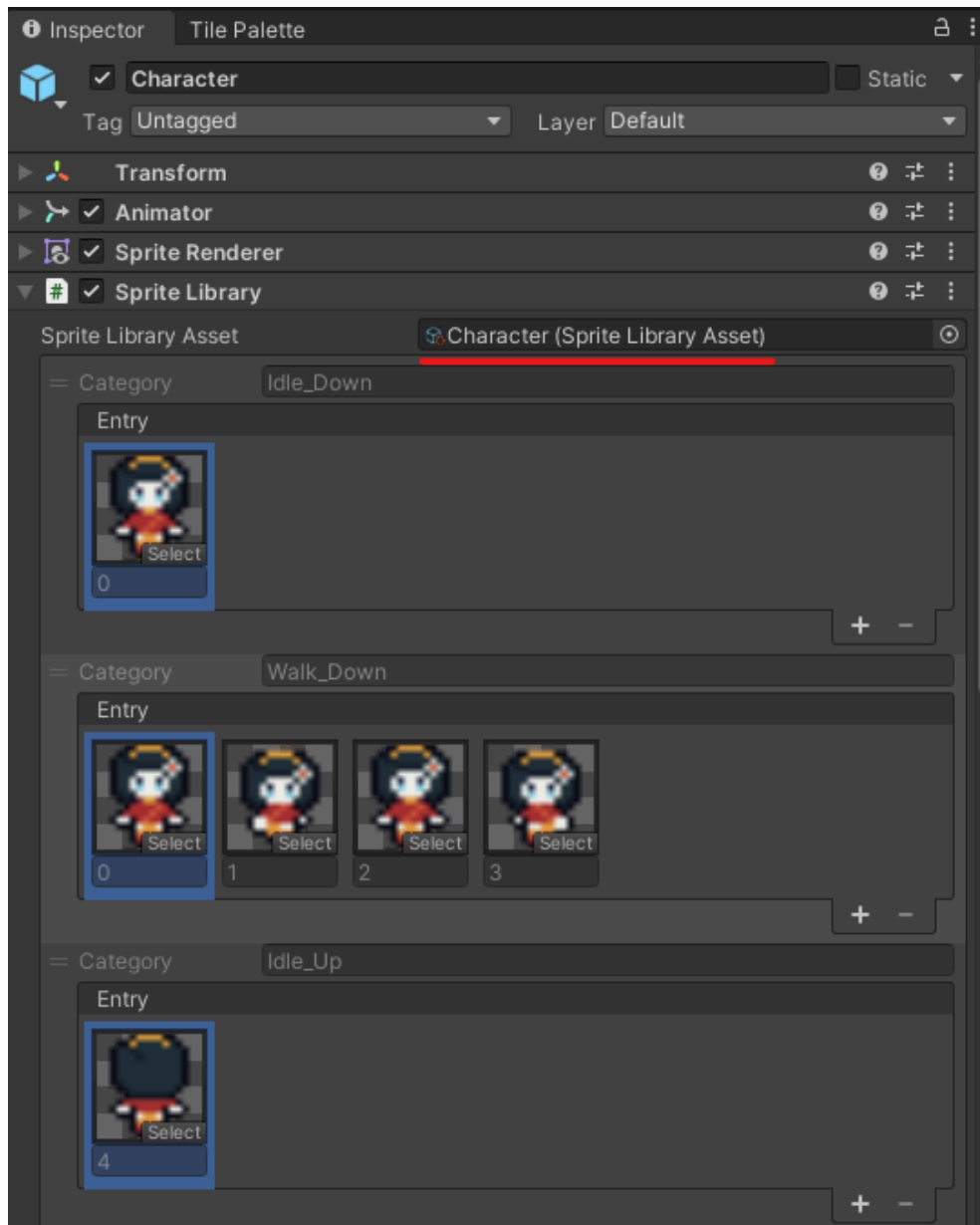


**Figura 67.0:** Sprite Library Asset correspondiente a la skin default del personaje de Spring Light

**Fuente:** Elaboración Propia



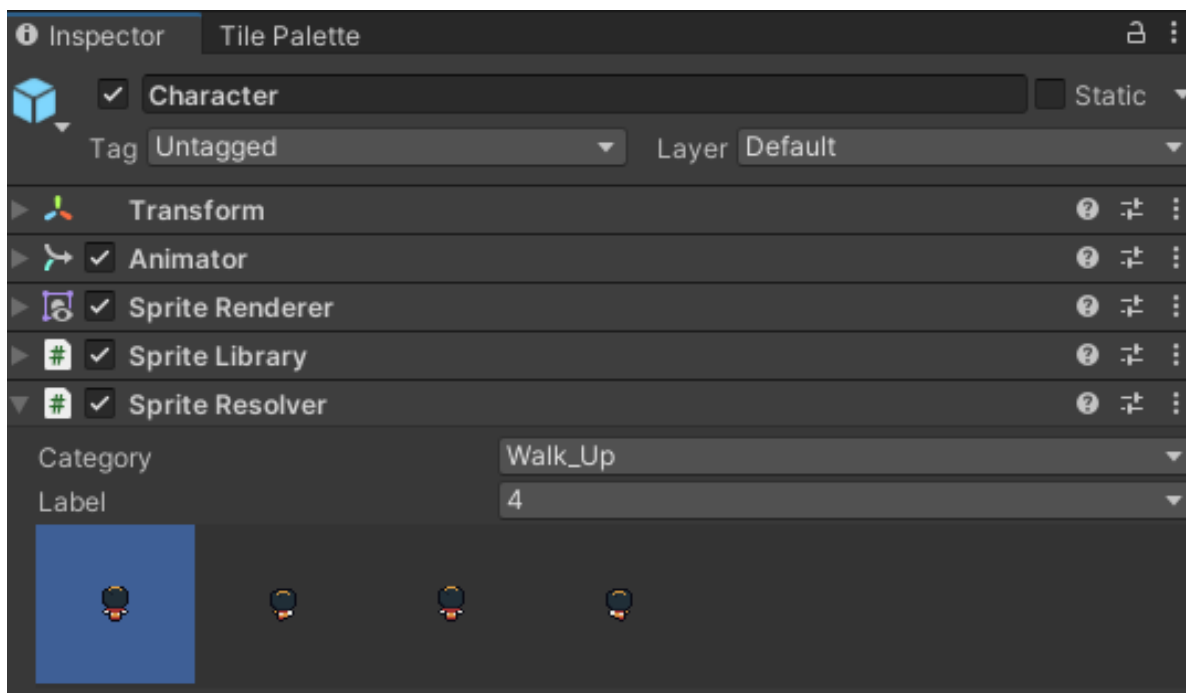
Dentro del juego se puede tener tantos Sprite Library Assets como el jugador quiera, uno por cada skin que tenga descargada. Estos Assets se asignan al componente Sprite Library que se encuentra en el objeto del jugador, como se muestra en la figura 68.0.



**Figura 68.0:** Sprite Library correspondiente a la skin default del personaje de Spring Light

**Fuente:** Elaboración Propia

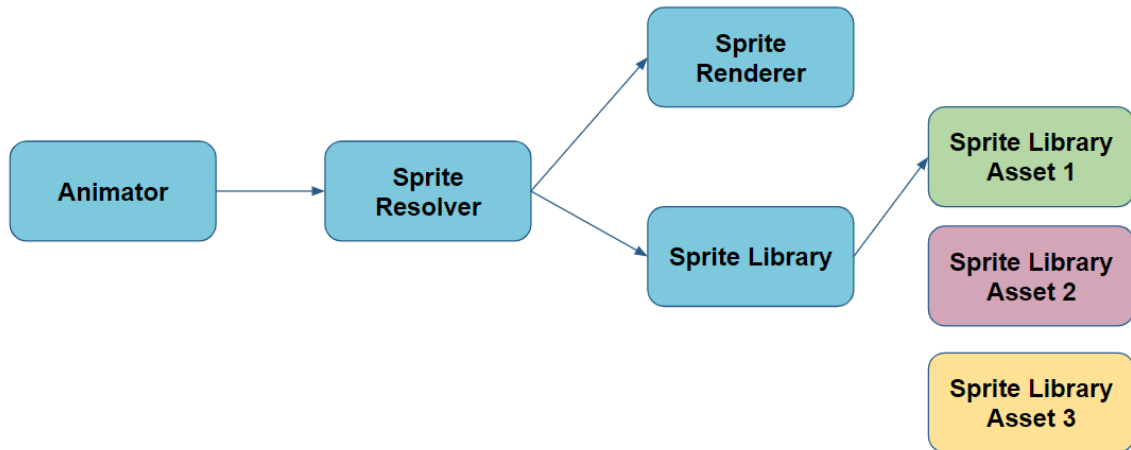
Entonces es el turno del Sprite Resolver de realizar la conexión entre el componente de Sprite Library, quien tiene la skin, y el componente Sprite Renderer, quien se encarga de pintar el gráfico en la pantalla en cada frame del juego. El Sprite Resolver se muestra en la figura 69.0.



**Figura 69.0:** Sprite Resolver

**Fuente:** Elaboración Propia

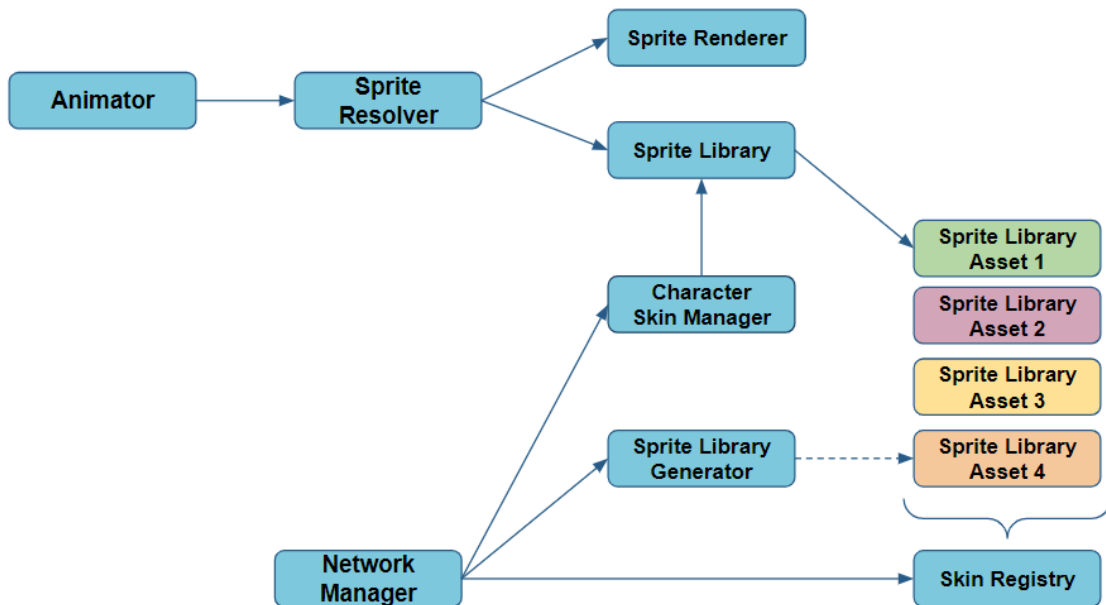
El flujo completo de utilización de una skin se resume en el cuadro de flujo correspondiente a la figura 70.0. Se marcan en azul los componentes que trabajan activamente en el renderizado de la skin y en colores los diferentes Assets que se pueden utilizar para representar diferentes skins.



**Figura 70.0:** Diagrama de flujo de llamados para cada frame de animación del personaje

**Fuente:** Elaboración Propia

Agregando a este diagrama los componentes necesarios para generar una nueva skin cuando el jugador ingresa un nuevo código, se tiene como flujo resultante el de la figura 71.0.



**Figura 71.0:** Diagrama de flujo tanto para la animación del personaje con cualquier skin y para la generación de una skin nueva

**Fuente:** Elaboración Propia

Si bien este sistema es un tanto complejo a primera vista, el diseño permite una gran versatilidad a la hora de agregar nuevos componentes y Assets por lo que se considera que se alcanzó un buen resultado.

Luego de realizar esta integración se realizó también el caso especial para actualizar las skins que el usuario puede tener en el juego pero que fueron actualizadas en la plataforma.

## Fase 1.4: Tests de Carga

Al finalizar todas las funcionalidades del sitio, se llevó a cabo un test de carga sobre el servidor para poder tener una métrica de cuánta carga de usuarios podría soportar. Este test no se puede realizar sobre el servidor real ya que este se quedaría fácilmente sin puertos de conexión y la prueba fallará. Por este motivo, se utilizó en cambio un contenedor de Docker, al cual se le limitaron las capacidades. Se le asignaron 512mb de memoria y 1 único core de CPU Intel i5 8va generación.

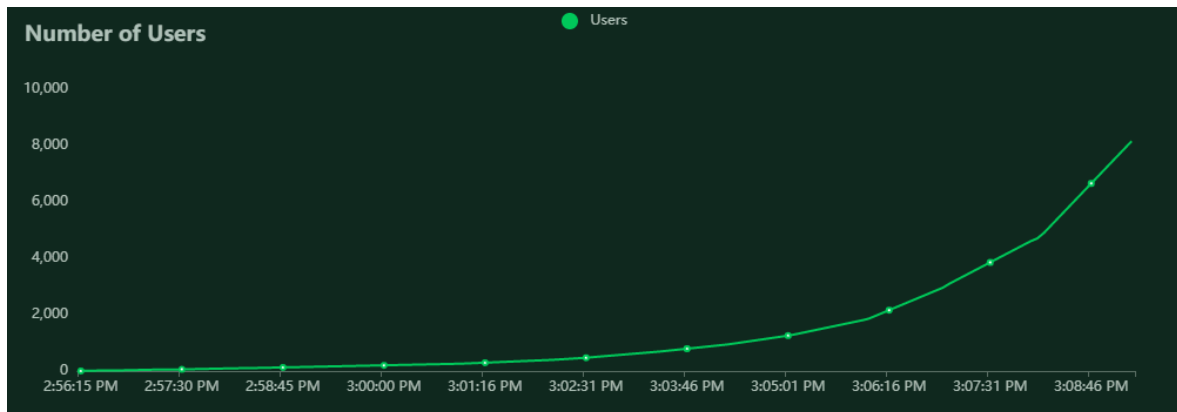
Los tests se realizaron de manera local utilizando una librería muy popular implementada en Python, llamada Locust.py. El test se llevó a cabo durante aproximadamente 15 minutos como se muestra en la figura 72.0.



**Figura 72.0:** Diagrama de flujo tanto para la animación del personaje con cualquier skin y para la generación de una skin nueva

**Fuente:** Elaboración Propia

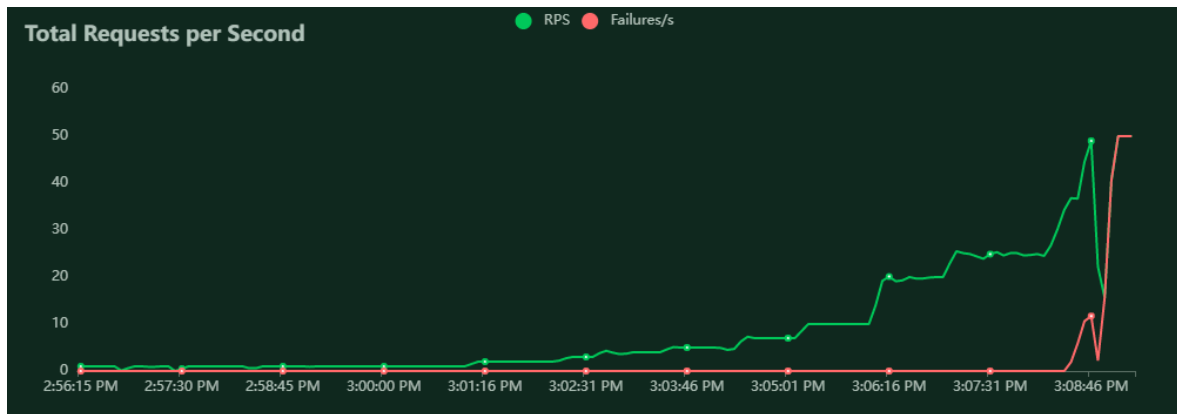
Este test arrojó resultados muy positivos. Se descubrió que en estas condiciones el servidor es capaz de soportar una concurrencia de aproximadamente 6000 usuarios, los cuales realizan una request cada 2 o 3 segundos. El servidor fallará si la cantidad de usuarios llega a aproximadamente 8000. El número máximo de usuarios concurrentes que registró el test fue de 8155. Esto se puede ver en el gráfico de la figura 73.0.



**Figura 73.0:** Cantidad de usuarios concurrentes en función del tiempo en el test de carga realizado con Locust

**Fuente:** Elaboración Propia

Respecto a las requests por segundo, en total se realizaron 7555 requests, de las cuales fallaron 1603 luego de que la cantidad de requests por segundo subiera de aproximadamente 40. Esto se puede ver en la figura 74.0.

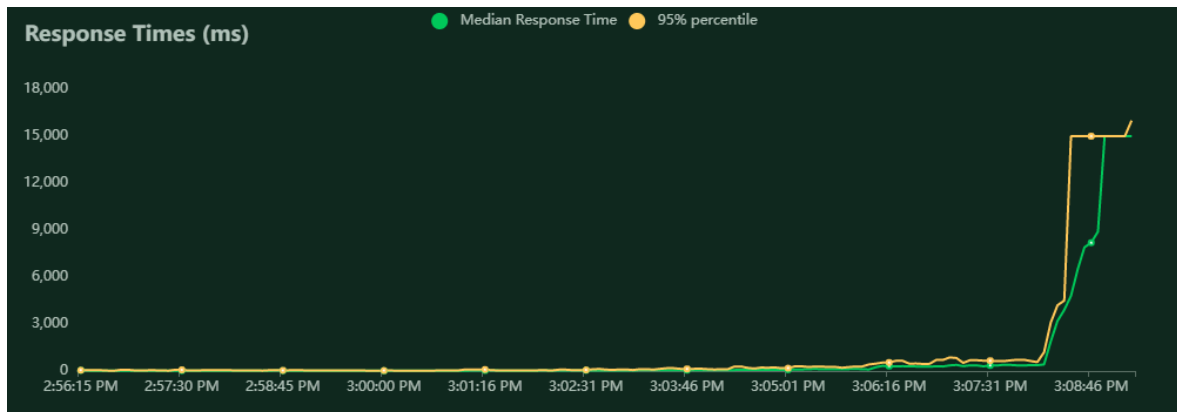


**Figura 74.0:** Cantidad de requests concurrentes en función del tiempo en el test de carga realizado con Locust

**Fuente:** Elaboración Propia

Por último, se registró también el tiempo de respuesta de las requests y se determinó un tiempo de respuesta mínimo nominal de 15 ms. El promedio para el test fue de 4135 ms y el tiempo máximo registrado fue de 15576 ms. Sin embargo, como se ve en la figura 75.0, hay que tener en cuenta que la mayoría del tiempo, hasta que el servidor comienza a fallar, el tiempo de respuesta no estaba ni cerca del valor promedio del test, sino más bien cerca del valor mínimo.





**Figura 74.0:** Cantidad de requests concurrentes en función del tiempo en el test de carga realizado con Locust

**Fuente:** Elaboración Propia

## Fase 2: Integración y pulido

En esta fase se mantuvo el foco en mejorar lo más posible las funcionalidades existentes del producto. Se agregaron las mecánicas de compra/venta de Items y se integraron todos los mapas dentro del mundo para que el jugador fuera capaz de caminar de uno a otro. En un principio se había concebido que todo el juego ocurriera sobre una montaña. Sin embargo, luego esta idea fue descartada ya que constringía mucho las opciones de personalización de los ambientes. Además, se trabajó en el arte para lograr que sea más estéticamente placentero.

Fue en esta etapa además en donde se consiguió crear el archivo ejecutable con todos los assets necesarios para poder realizar el proceso de distribución. El ejecutable se creó para el sistema operativo Windows arquitectura x64, ya que este era el sistema operativo de desarrollo. Este se envió a usuarios para que lo prueben en sus propias máquinas y así recibir feedback de diferentes entornos.

Normalmente esto es muy importante para ver que el juego ejecute correctamente con diferentes CPUs y/o tarjetas gráficas debido a que los distintos hardware y sistemas operativos pueden tener características y requisitos específicos que pueden afectar el rendimiento y la funcionalidad del juego. Por ejemplo, un juego que funciona perfectamente en una computadora con un procesador potente y una tarjeta gráfica de última generación puede tener problemas de rendimiento en un dispositivo móvil con hardware más limitado.

Probar el juego en diferentes plataformas permite identificar y corregir cualquier problema relacionado con la compatibilidad antes de su lanzamiento al mercado. Algunos de estos problemas pueden incluir diferente velocidad de procesamiento, pérdida de fps<sup>60</sup>, pérdida de calidad de las texturas o incluso simplemente inhabilidad para ejecutar o frecuentes crashes. Sin embargo, al ser Spring Light un juego de muy bajos recursos este

---

<sup>60</sup> "FPS" es un acrónimo que significa "Frames Per Second", o "Cuadros Por Segundo", en español. Se refiere a la cantidad de imágenes (cuadros) que se muestran por segundo en un videojuego o en cualquier otro tipo de animación. Una tasa de FPS más alta generalmente resulta en una experiencia más suave y fluida para el usuario. Los juegos y aplicaciones de realidad virtual a menudo requieren tasas de FPS más altas para proporcionar una experiencia inmersiva y sin interrupciones.

punto no era tan importante como sí lo era, por ejemplo, el probar la resolución de la UI en diferentes tamaños de pantalla. Si bien Unity permite tener una previsualización de cómo se vería el juego en diferentes resoluciones, es importante también tener constancia de cómo se ve realmente en un entorno real.

## Fase 3: Distribución

Para esta fase se investigó las diferentes formas posibles que existen y se utilizan para distribuir copias digitales de un videojuego. Existen varios medios o canales disponibles para realizar esto, dependiendo de la audiencia a la que se quiera apuntar y las características del juego.

- **Tiendas:** Steam, Epic Games Store, itch.io, entre otras. Estas tiendas ofrecen un lugar para que los desarrolladores suban y promuevan sus juegos, y los jugadores puedan comprarlos o descargarlos gratuitamente. La ventaja de estos sitios es que permiten alcanzar audiencias de muchos jugadores, y que son un estándar de la industria. Un juego que se distribuye por sí mismo pero no por Steam, por ejemplo, se ve como algo extraño desde el punto de vista de los jugadores, por lo cual a veces las compañías se sienten presionadas a publicar sus juegos en estas plataformas. Sin embargo, estas páginas obligan a quienes publican juegos en ellas a adherirse a políticas que suelen ser innegociables, como por ejemplo recaudar un porcentaje de las ganancias que alcance el juego.
- **Descarga directa:** los desarrolladores pueden proporcionar un enlace de descarga directa para su juego en su sitio web o en una plataforma de medios sociales. Esto muchas veces incluye un launcher, que es un programa externo que permite realizar una descarga menos pesada para el servidor en primera instancia. Además, un launcher permite gestionar más fácilmente la actualización y eventual desinstalación del juego. La desventaja de este canal es que incurre en un gran costo monetario y de infraestructura para el desarrollador, por lo que normalmente se reserva para juegos de empresas más grandes, quienes tienen el poder adquisitivo y los medios para crear y mantener dicha infraestructura.
- **Suscripciones:** algunos servicios de suscripción, como Xbox Game Pass o Apple Arcade, ofrecen un catálogo de juegos a los suscriptores. Estos juegos suelen ser

específicos para una plataforma y se suele cobrar a los publicantes una tarifa para entrar en el catálogo en primera instancia.

## Publicar un juego en Steam

Para publicar un juego en Steam, primero es necesario tener una cuenta de desarrollador, la cual tiene un costo anual de \$50 dólares estadounidenses. Una vez que se tiene la cuenta, se puede utilizar el Steam Direct, un sistema de publicación para juegos independientes, para subir el juego.

Este proceso requiere de información básica sobre el juego, como el título, una descripción, imágenes y un video promocional. También es necesario especificar los sistemas operativos y las plataformas compatibles. Finalmente, se debe pagar una tarifa de \$100 dólares estadounidenses por cada juego publicado. Una vez que el juego ha sido revisado y aprobado por Steam, estará disponible para ser comprado y descargado por los usuarios.

Además, Steam se reserva un 30% de las ganancias que genere un juego por medio de ventas o microtransacciones en su plataforma. El proceso de publicación de un juego en Steam ha ido variando a lo largo de los años y existen muchas formas que fueron deprecadas con los años.

## Publicar un juego en Itch.io

Para publicar un juego en itch.io primero se debe crear una cuenta en la plataforma y subir tu juego a tu perfil. Esto puede ser mediante un archivo ejecutable o mediante un enlace a una página web donde se pueda jugar. Una vez subido, se puede configurar varios aspectos del juego, como su precio, su sistema de juego, su descripción, capturas de pantalla y vídeos promocionales.

Después de configurar todos estos aspectos, se pone en publicación el juego y estará disponible para que los usuarios lo descarguen o compren. Si bien itch.io no tiene el alcance ni las herramientas que provee Steam, este proceso es totalmente gratuito, por lo cual es una alternativa muy popular entre desarrolladores individuales o proyectos de poca envergadura.

Sin embargo, lamentablemente en itch.io se encuentra muy viralizado el problema del copyright o derechos de autor para los software que se suben al sitio. La compañía no toma suficientes recaudos para proteger los derechos de autor de los usuarios por lo cual subir contenido a esta página conlleva ciertos riesgos que se decidieron evitar.

## Distribución

Lo primero que se tuvo en cuenta a la hora de preparar el archivo ejecutable fue el público para el cual estaba preparado en juego. En primera instancia el plan fue darlo a probar a conocidos personalmente para que estos dieran una devolución de las primeras versiones y poder hacer correcciones y mejoras. Para esto se conocía que todos los usuarios poseían PCs tanto Windows como Mac, por lo que se decidió crear el ejecutable para esos dos sistemas operativos.

La integración del juego en ejecutables se realiza en la máquina que se utilizó para el desarrollo en lugar de un pipeline de building o de integración desde un repositorio remoto. Esto es por dos motivos.

En primer lugar, el building de Unity se caracteriza por ser un poco lento. En la máquina de desarrollo tarda aproximadamente 1 minuto y 13 segundos en crear el build de 250Mb, teniendo esta 12Gb de RAM y un procesador Intel i5 8va generación. En un pipeline de Github Actions, por ejemplo, las máquinas que se utilizan tienen 7 Gb de RAM y una CPU de tan solo 2 cores<sup>61</sup>, por lo que el tiempo de integración aumentaría significativamente. A pesar de esto, Github Actions permite hasta un límite de 6hs por trabajo de ejecución<sup>62</sup> por lo cual, a menos que el juego crezca en tamaño considerablemente esto no sería un problema en principio.

En segundo lugar, para agregar un pipeline de integración al pipeline existente existen dos caminos: Utilizar GameCI o crear un pipeline de integración desde cero.

### Utilizar GameCI

Esta es una herramienta que provee un Github Action, el cual puede ejecutar testing y compilar el proyecto. Como se mencionó anteriormente, este se está utilizando para realizar el testing automático cada vez que se suben cambios al repositorio remoto. Sin embargo, para la pipeline de building, GameCI requiere algunos datos adicionales que

---

<sup>61</sup> About GitHub-hosted runners. (n.d.). GitHub Docs. Retrieved February 9, 2023, from <https://docs.github.com/en/actions/using-github-hosted-runners/about-github-hosted-runners>

<sup>62</sup> Usage limits, billing, and administration. (n.d.). GitHub Docs. Retrieved February 9, 2023, from <https://docs.github.com/en/actions/learn-github-actions/usage-limits-billing-and-administration>

necesita por motivos internos de Unity para realizar la compilación. Estos son el usuario y la contraseña de la cuenta dueña del proyecto. Si bien esta información es sensible, el riesgo de que se exponga no es mucho ya que lo hacen a través de una clave privada en el repositorio llamada “Repository Secret”<sup>63</sup>.

Sin embargo, el problema radica en que Unity permite crear cuentas no solo con usuario y contraseña sino también con social login, como es el caso de google. En el caso de Spring Light esto fue lo que se utilizó para crear la cuenta, y GameCI no parece tener soporte para este método de autenticación.

### **Crear un pipeline de integración para juegos de Unity desde cero**

La alternativa pareciera ser crear un pipeline de integración desde cero. Este permitiría realizar la integración del juego en alguna plataforma remota como puede ser un Github Actions o incluso configurar una máquina en un servidor remoto de Amazon o cualquier otro proveedor de servicios de nube. Esto implica un esfuerzo considerable y sería una funcionalidad muy avanzada para una etapa del proyecto en donde el tiempo de compilación y su distribución sea crítica.

Esto podría ser por ejemplo en el caso de una empresa grande en donde se tiene muchos empleados que deben probar el juego y crear versiones nuevas todos los días para realizar correcciones y devolver al área de aseguramiento de calidad. En estos casos, tanto el tiempo de compilación como la distribución sería crucial que se hagan de forma rápida y automática.

---

<sup>63</sup> Encrypted secrets. (n.d.). GitHub Docs. Retrieved February 9, 2023, from <https://docs.github.com/en/actions/security-guides/encrypted-secrets>



## Plataforma de Distribución

Tomando en cuenta las características de los diferentes medios de distribución disponibles, no se logró realizar la distribución del juego por medio de una plataforma online como Steam o Itch.io. En Steam se encontró que la barrera económica de entrada era demasiado alta y además posee un proceso de aprobación por parte de la compañía misma que en gran parte no está bajo control del publicante. En el caso de Itch.io si bien no existían barreras de entrada costosas ni verificaciones por parte de la compañía, existe un riesgo grande al subir el contenido ya que no se puede asegurar del todo el respeto de los derechos de autoría de los archivos, los cuales pueden ser redistribuidos o replicados sin autorización por otros medios.

La distribución se realizó en cambio en forma de archivo ejecutable en una carpeta comprimida enviada directamente a los usuarios que pidieron una copia. Esta se encuentra publicada en el repositorio git del juego en la plataforma Github marcada como un release. Gracias a esto es posible compartir un link para que los usuarios se lo descarguen pero al ser Github una plataforma orientada hacia la programación, esta solución no es muy atractiva para juegos fuera de la etapa de desarrollo.

Como expansión del proyecto, sin embargo, sería una buena idea intentar lanzar el juego en alguna tienda de modo que se pueda distribuir de manera masiva sin tener que conocer a los individuos personalmente.

## 8. Usuarios y Funcionalidades

### 8.1 Tipos de usuarios

La audiencia objetivo del juego Spring Light son jugadores de preferencia muy similar a la audiencia del Harvest Moon o el Stardew Valley. Las mecánicas son simples y no requieren coordinación en tiempo real, con lo cual no se requiere experiencia previa en el mundo de los videojuegos, a diferencia de otros ejemplares como por ejemplo el Dark Souls o juegos de alta gama que asumen que el jugador ya posee experiencia y elevan el nivel de dificultad para este tipo de usuario. Si bien la audiencia objetivo está más cerca de adolescentes, puede ser disfrutado por cualquier edad, ya que presenta mecánicas y desarrollo relativamente neutro para cualquier edad o sexo.

## 8.2 Definición de funcionalidades

Se detallan a continuación las funcionalidades que presenta el videojuego Spring Light:

- Sistema de interacción por caja de diálogo
  - Al interactuar con objetos del mundo se abre una caja de diálogo que muestre un texto referente al objeto con el que se interactúa
- Ciclo de días
  - Se puede cambiar de día para avanzar el progreso de los cultivos
- Sistema monetario y de inventario
  - Capacidad de guardar una cantidad limitada de objetos en el inventario
  - Sistema monetario para el jugador
- Compra de productos
  - El jugador puede comprar objetos que le sirvan en la granja mediante el uso de monedas
- Venta de productos
  - La venta de los productos producidos en la granja se realiza mediante una caja en donde se puede ingresar todos los productos para vender.
- Producción de cosecha
  - Las plantas crecerán y producirán atadas al paso del ciclo de los días.
  - Plantar semillas
  - Regar
  - Cosechar productos
- Puzzles
  - 5 puzzles con diferentes mecánicas, resoluciones y recompensas únicas.
- Guardado de progreso
  - Sistema de guardado de progreso automático cada vez que se cambia de escena o se sale del juego a través del menú.

Se detallan a continuación las funcionalidades que presenta la plataforma web:

- Dibujador
  - El dibujador permite crear desde la misma página web cada una de las imágenes que componen todo el Sprite de una nueva skin. Para esto permite realizar dibujos en un cuadro de tamaño 16 x 32 píxeles, aceptando imágenes para las cuatro vistas: arriba, abajo, izquierda y derecha; con un máximo de 4 frames por vista. Permite las opciones de dibujar con color, transparencia, modo arcoiris y borrador. Además, requiere un nombre para el nuevo sprite creado.
- Subir Sprite
  - En caso de tener las imágenes del nuevo sprite ya creadas, por ejemplo en otro programa de edición, la plataforma permite también crear un nuevo sprite subiendo las imágenes de cada frame de cada vista directamente.
- Buscar Sprites
  - Los usuarios pueden filtrar los Sprites del juego utilizando un buscador por nombre.
- Editar Sprite
  - Esta funcionalidad permite a los usuarios editar un Sprite que hayan creado tanto por subida como por dibujador. Esto es muy poderoso ya que permite edición cruzada entre los dos tipos de edición, pudiendo subir imágenes y luego editarlas a mano en la misma plataforma.
- Registro y autenticación de Usuario
  - Los usuarios son capaces de registrarse con un nombre de usuario y contraseña, lo cual les proporciona acceso a la plataforma y a crear Sprites.
- Perfil de Usuario
  - Cada usuario posee un perfil el cual muestra todos los Sprites creados por ese usuario que sean públicos, así como también el nombre de la cuenta y una imagen banner.
- Modificar Perfil de Usuario

- La página permite también editar el nombre de usuario, la contraseña y el banner del perfil.
- Transferir Skin al juego
  - Cuando un usuario encuentra una skin que le gusta, este es capaz de previsualizar el movimiento en todas las vistas y luego, si así lo quisiera, presionar un botón para copiar a su portapapeles un código que le permita agregar esa Skin a su copia de Spring Light.
- Cambiar de Skin en Spring Light
  - Dentro del juego, se permite alternar entre las diferentes Skins que el jugador haya descargado por medio de la plataforma web, así como también las skins que trae el juego por defecto.

## 8.3 Especificación de user stories

### Movimiento 2D

**Descripción:** El personaje deberá poder moverse utilizando el sistema de Input de Unity. Este movimiento será de traslación hacia arriba, abajo, izquierda y derecha de la pantalla en forma de grilla. El movimiento de rotación será la posición hacia la cual está mirando el personaje y cambiará dependiendo de hacia donde se esté moviendo. El movimiento podrá ser bloqueado por ciertos objetos como paredes y terrenos. Además, se podrá apreciar una animación del personaje que denote que está moviéndose.

### User Story

**Como** jugador

**Quiero** poder mover mi personaje en el mundo

**Tal que** pueda explorar el mundo e interactuar con sus objetos

### Criterios de Aceptación

**Dado** que el personaje se encuentra habilitado para moverse

**Cuando** utilizo el sistema de Input estándar definido por Unity (por ejemplo las flechas en PC)

**Entonces** el personaje se mueve en esa dirección.

**Sistema de Interacción por caja de diálogo**

**Descripción:** Para poder interactuar con los objetos del mundo, el personaje deberá posicionarse enfrente del objeto interactuable y presionar la tecla de interacción. El personaje debe estar rotado orientado hacia el objeto para que la interacción sea exitosa. La interacción puede causar diversos efectos, uno de ellos siendo la apertura de un cuadro de diálogo con un texto referente a esa interacción.

**User Story**

**Como** jugador

**Quiero** poder interactuar con determinados objetos en el mundo

**Tal que** aparezca un cuadro de diálogo que indique cierto texto referente a ese objeto

**Criterios de Aceptación**

**Dado** un objeto de tipo interactuable que tenga un texto asociado

**Cuando** interactuo con este

**Entonces** aparece un cuadro de diálogo con el texto asociado a ese objeto

**Ciclo de días**

**Descripción:** El juego debe contener un mecanismo para el paso del tiempo. Esto se debe a que existen ciertos objetos, como por ejemplo los cultivos, que dependen del paso del tiempo para completar su propósito. Sin embargo, el progreso no se medirá conforme al tiempo real, sino que se establecerán unidades discretas de tiempo en forma de días que el jugador podrá hacer pasar libremente mediante la interacción con algún objeto, por ejemplo una casa o una cama.

**User Story**

**Como** jugador

**Quiero** hacer pasar los días del mundo

**Tal que** el paso del tiempo pueda afectar a los objetos pertinentes (por ejemplo las plantas)

**Criterios de Aceptación**

**Dado** un conjunto de objetos que son afectados por el pasar de los días

**Cuando** interactuo con el objeto designado a controlar el paso de los días

**Entonces** se notifica a todos los objetos pertinentes del cambio de día



**Sistema Monetario**

**Descripción:** Deberá existir un mecanismo que registre la cantidad de monedas que el jugador posee en un determinado momento, junto con la posibilidad de consultar su cantidad, agregar o sustraer monedas. Estas se podrán ver fácilmente en la interfaz de usuario.

**User Story**

**Como** jugador

**Quiero** poder tener un registro de las monedas que poseo

**Tal que** pueda ganar, gastar o consultar su cantidad

**Criterios de Aceptación**

**Dado** que se está dentro del juego

**Cuando** compro o vendo items

**Entonces** puedo utilizar las monedas en mi cuenta

**Sistema de Inventario**

**Descripción:** Se debe poder llevar un registro de los objetos que el jugador colecciona durante su aventura en el juego, para que pueda revisarlos y utilizarlos cuando sea necesario. Además, se debe poder tener una vista general de todos los objetos que se tiene disponibles.

**User Story**

**Como** jugador

**Quiero** poder cargar cierta cantidad de Ítems

**Tal que** pueda utilizarlos para interactuar con objetos del mundo

**Criterios de Aceptación**

**Dado** un objeto de tipo interactuable que se pueda guardar en el inventario y espacio libre en el inventario

**Cuando** interactuo con este

**Entonces** se guarda en el inventario

**Sistema de Interacción con Ítem del inventario**

**Descripción:** Se podrá seleccionar cualquier espacio del inventario para interactuar con el mundo. Si este espacio contiene un Ítem, este se utilizara para interactuar. En algunos casos determinados objetos necesitaran determinados ítems para que la interacción tenga un efecto. Por ejemplo, si se tiene un objeto de tipo “semilla” y se interactúa con tierra arada entonces se consumirá el ítem y se plantarán las semillas.

**User Story**

**Como** jugador

**Quiero** poder interactuar con determinados objetos en el mundo utilizando un ítem

**Tal que** estos tengan un efecto especial

**Criterios de Aceptación**

**Dado** un objeto de tipo interactuable que acepte interacción con un ítem

**Cuando** selecciono un ítem en el inventario e interactúo

**Entonces** se dispara el efecto deseado de la interacción

<b>Compra de Productos</b>
<b><u>Descripción:</u></b> Se podrá hacer uso de las monedas recolectadas para comprar ítems definidos.
<b><u>User Story</u></b>  <b>Como</b> jugador  <b>Quiero</b> poder utilizar las monedas acumuladas en una tienda  <b>Tal que</b> obtenga recompensas a cambio
<b><u>Criterios de Aceptación</u></b>  <b>Dado</b> una cantidad de monedas suficientes  <b>Cuando</b> compro un ítem en la tienda  <b>Entonces</b> se agrega este ítem a mi inventario

**Venta de Productos**

**Descripción:** Al colocar los productos cosechados en la caja de ventas se podrán vender dichos Ítems por monedas, las cuales se agregaran a la cuenta del jugador. Estos Ítems se venderan únicamente cuando cambie el día.

**User Story**

**Como** jugador

**Quiero** poder vender los Ítems que produzco en mi granja

**Tal que** gane a cambio una cantidad de monedas por cada uno

**Criterios de Aceptación**

**Dado** un Ítem que tenga un valor monetario

**Cuando** lo introduzco en la caja de ventas y el día cambia

**Entonces** se acredita la cantidad de monedas que vale en mi cuenta

## Producción de cosecha

**Descripción:** En la granja habrá un espacio en donde se podrán cosechar plantas para vender. Este espacio permitirá al jugador utilizar las herramientas a su disposición para arar la tierra, regarla y cosechar productos que ya hayan crecido. En primer lugar la tierra se encontrará en estado infertil. Luego, al ararla con la herramienta adecuada pasará a ser tierra arada. Luego se deberán plantar las semillas interactuando con un objeto de tipo semilla. Luego se deberá regar la tierra y hacer pasar los días. Al cabo de una cantidad de días definida por el tipo de cultivo, el brote se transformará en un cultivo listo para cosechar. En esa instancia se podría cosechar con la herramienta adecuada y la tierra volverá a su estado inicial.

## User Story

**Como** jugador

**Quiero** poder hacer crecer plantas en mi granja

**Tal que** pueda cosechar y vender sus productos

## Criterios de Aceptación

**Dado** una semilla de una planta

**Cuando** aro la tierra, planto la semilla, la riego por la cantidad de días necesarios y la cosecho

**Entonces** obtengo un producto correspondiente a esta planta

**Puzzles**

**Descripción:** En ciertas partes del mapa el jugador podrá encontrar rompecabezas o puzzles que requieran algún tipo de ingenio o habilidad para resolverse. Al cumplir las condiciones de victoria, se le otorgará al jugador algún tipo de recompensa.

**User Story**

**Como** jugador

**Quiero** poder resolver diferentes puzzles que encuentre en el mundo

**Tal que** reciba una recompensa a cambio

**Criterios de Aceptación**

**Dado** que el jugador encuentra un puzzle en el mundo

**Cuando** lo resuelve

**Entonces** consigue una recompensa

**Guardar y Cargar progreso**

**Descripción:** Cada vez que se cambie la escena o se salga el juego por completo se deberá guardar el progreso de éste, de modo que al volver a ingresar se le pueda dar al jugador la opción de continuar donde lo dejó o comenzar un juego nuevo. En caso de continuar donde lo dejó, se debe poder recuperar el estado que tenían los objetos antes de que el juego se cerrara.

**User Story**

**Como** jugador

**Quiero** poder guardar y cargar el progreso que realice en el juego

**Tal que** pueda continuar mi partida en otro momento

**Criterios de Aceptación**

**Dado** haber realizado algún progreso en el juego

**Cuando** salgo de este y vuelvo a ingresar

**Entonces** tengo la opción de continuar desde donde lo deje la ultima vez



**Dibujador**

**Descripción:** El dibujador permite crear desde la misma página web cada una de las imágenes que componen todo el Sprite de una nueva skin. Para esto permite realizar dibujos en un cuadro de tamaño 16 x 32 píxeles, aceptando imágenes para las cuatro vistas: arriba, abajo, izquierda y derecha; con un máximo de 4 frames por vista. Permite las opciones de dibujar con color, transparencia, modo arcoiris y borrador. Además, requiere un nombre para el nuevo sprite creado.

**User Story**

**Como** usuario autenticado

**Quiero** poder realizar dibujos en la plataforma web

**Tal que** pueda crear un Sprite sin necesidad de utilizar un programa externo

**Criterios de Aceptación**

**Dado** un usuario autenticado

**Cuando** Dibujo cada uno de los frames de las vistas del sprite, le asigno un nombre y lo guardo

**Entonces** se crea el nuevo sprite que dibuje

**Subir Sprite**

**Descripción:** En caso de tener las imágenes del nuevo sprite ya creadas, por ejemplo en otro programa de edición, la plataforma permite también crear un nuevo sprite subiendo las imágenes de cada frame de cada vista directamente. Estas imágenes no pueden superar los 4 frames por vista ni ser mayores a 32 x 16 píxeles.

**User Story**

**Como** usuario autenticado

**Quiero** poder cargar imágenes en la plataforma web

**Tal que** pueda crear un Sprite con esas imagenes

**Criterios de Aceptación**

**Dado** un usuario autenticado

**Cuando** Cargo en la página cada uno de los frames de las vistas del sprite, le asigno un nombre y lo guardo

**Entonces** se crea el nuevo sprite

**Buscar Sprite**

**Descripción:** Los usuarios pueden filtrar los Sprites del juego utilizando un buscador por nombre. Los resultados incluyen únicamente los sprites públicos que coincidan en nombre parcialmente con la palabra buscada

**User Story**

**Como** usuario

**Quiero** poder filtrar los Sprites por nombre

**Tal que** pueda encontrar uno específico o un grupo de sprites de acuerdo al nombre que ingrese

**Criterios de Aceptación**

**Dado** un usuario

**Cuando** escribo una palabra en el buscador y presiono buscar

**Entonces** se filtran los sprites por nombre

**Editar Sprite**

**Descripción:** Esta funcionalidad permite a los usuarios editar un Sprite que hayan creado tanto por subida como por dibujador. Esto permite edición cruzada entre los dos tipos de edición, pudiendo subir imágenes y luego editarlas a mano en la misma plataforma. Además, se puede cambiar el nombre del sprite y su visibilidad de privado o público.

**User Story**

**Como** usuario autenticado dueño de un sprite

**Quiero** poder editar mi sprite dibujando o cargando imagenes

**Tal que** se guarden esos cambios

**Criterios de Aceptación**

**Dado** usuario autenticado dueño de un sprite

**Cuando** realizo cambios a las imágenes del sprite, al nombre o a la visibilidad

**Entonces** se guardan los cambios realizados

**Registro y Autenticación de Usuario**

**Descripción:** Los usuarios son capaces de registrarse con un nombre de usuario y contraseña, lo cual les proporciona acceso a la plataforma y a crear Sprites. Luego son capaces de hacer Login en la plataforma para autenticar su identidad. Por último, pueden salir de su cuenta para borrar la identidad del navegador.

**User Story**

**Como** usuario no autenticado

**Quiero** poder crear una cuenta y luego hacer login con ella

**Tal que** pueda acceder a las funcionalidades del sitio

**Criterios de Aceptación**

**Dado** usuario no autenticado

**Cuando** se registra y luego hace login utilizando nombre y contraseña

**Entonces** puede acceder a su perfil y las funcionalidades del sitio

**Perfil de Usuario**

**Descripción:** Cada usuario posee un perfil el cual muestra todos los Sprites creados por ese usuario que sean públicos, así como también el nombre de la cuenta y una imagen banner.

**User Story**

**Como** usuario

**Quiero** poder ver el perfil de otro usuario

**Tal que** pueda ver su banner, nombre y los sprites públicos creados por este

**Criterios de Aceptación**

**Dado** usuario

**Cuando** entro en el perfil de otro usuario

**Entonces** puedo ver su banner, nombre y los sprites públicos creados por este

**Modificar Perfil de Usuario**

**Descripción:** Se permite editar el nombre de usuario, la contraseña y el banner del perfil.

**User Story**

**Como** usuario autenticado

**Quiero** poder editar mi nombre, contraseña o banner de mi perfil

**Tal que** se actualicen estos datos

**Criterios de Aceptación**

**Dado** usuario

**Cuando** entro en la seccion de edicion y edito mis datos

**Entonces** mis datos se actualizan

**Transferir Skin al juego**

**Descripción:** Cuando un usuario encuentra una skin que le gusta, este es capaz de previsualizar el movimiento en todas las vistas y luego, si así lo quisiera, presionar un botón para copiar a su portapapeles un código que le permita agregar esa Skin a su copia de Spring Light.

**User Story**

**Como** usuario

**Quiero** poder utilizar una skin que encontré en la plataforma en Spring Light

**Tal que** se cambie la skin con la que juego

**Criterios de Aceptación**

**Dado** usuario

**Cuando** copio el código del Sprite desde la plataforma y lo ingreso en el juego

**Entonces** se descarga esa skin desde el servidor y puedo utilizarla para jugar



**Cambiar de Skin en Spring Light**

**Descripción:** Dentro del juego, se permite alternar entre las diferentes Skins que el jugador haya descargado por medio de la plataforma web, así como también las skins que trae el juego por defecto.

**User Story**

**Como** usuario

**Quiero** poder alternar entre las skins que haya descargadas en mi juego

**Tal que** se cambie la skin con la que juego

**Criterios de Aceptación**

**Dado** usuario

**Cuando** presiono alguna de las flechas para cambiar de skin en el menú

**Entonces** se cambia la skin con la que puedo jugar

## 8.4 Flujo del juego

El mundo del juego Spring Light est  dividido en diferentes escenas, entre las cuales se puede cambiar obteniendo una ilusi n de unidad entre ellas. En primer lugar, la primera escena que se muestra al jugador al abrir el juego es el men  principal. Este muestra el nombre del juego y presenta dos opciones: Crear un juego nuevo o continuar la partida anterior, en caso de tener alg n progreso guardado, en cuyo caso se contin a desde donde se dej  al cerrar el juego por  ltima vez.

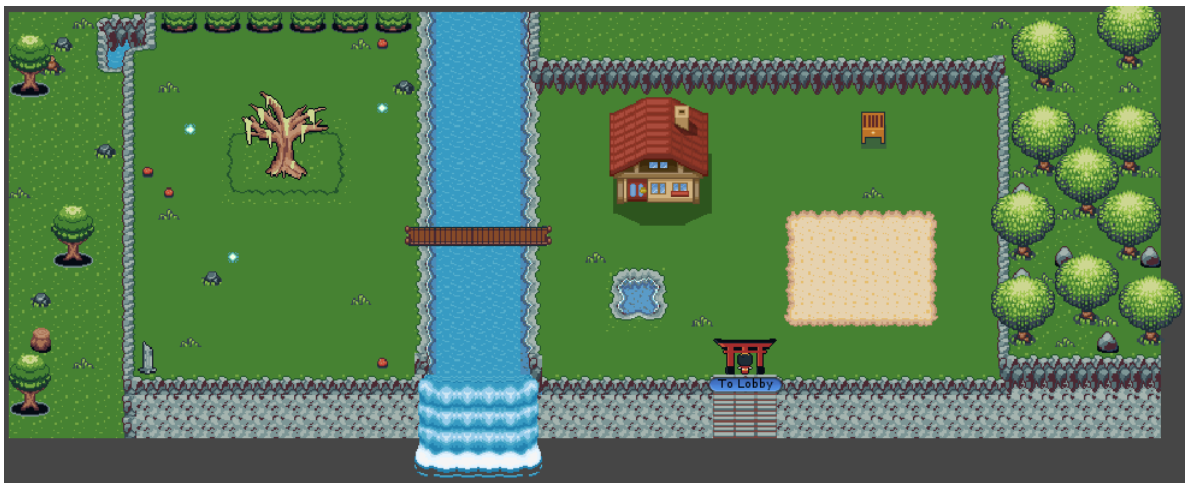
Al hacer click en el bot n de crear un juego nuevo se carga la primera escena, llamada "Lobby", mostrada en la figura 48.0. Ambientada en forma de bosque al pie de una monta a, la escena de lobby sirve a modo de puente entre la granja y los diferentes puzzles.



**Figura 48.0:** Escena del Lobby

**Fuente:** Elaboración Propia

Si se camina hacia arriba al pasar por una puerta de color rojo se cambia nuevamente la escena a la granja, la cual se puede ver en la figura 49.0. Esta cuenta con la parte fértil en donde se puede crecer cultivos, la casa, la caja de ventas y, cruzando un puente, la tienda.



**Figura 49.0:** Escena de la granja

**Fuente:** Elaboración Propia

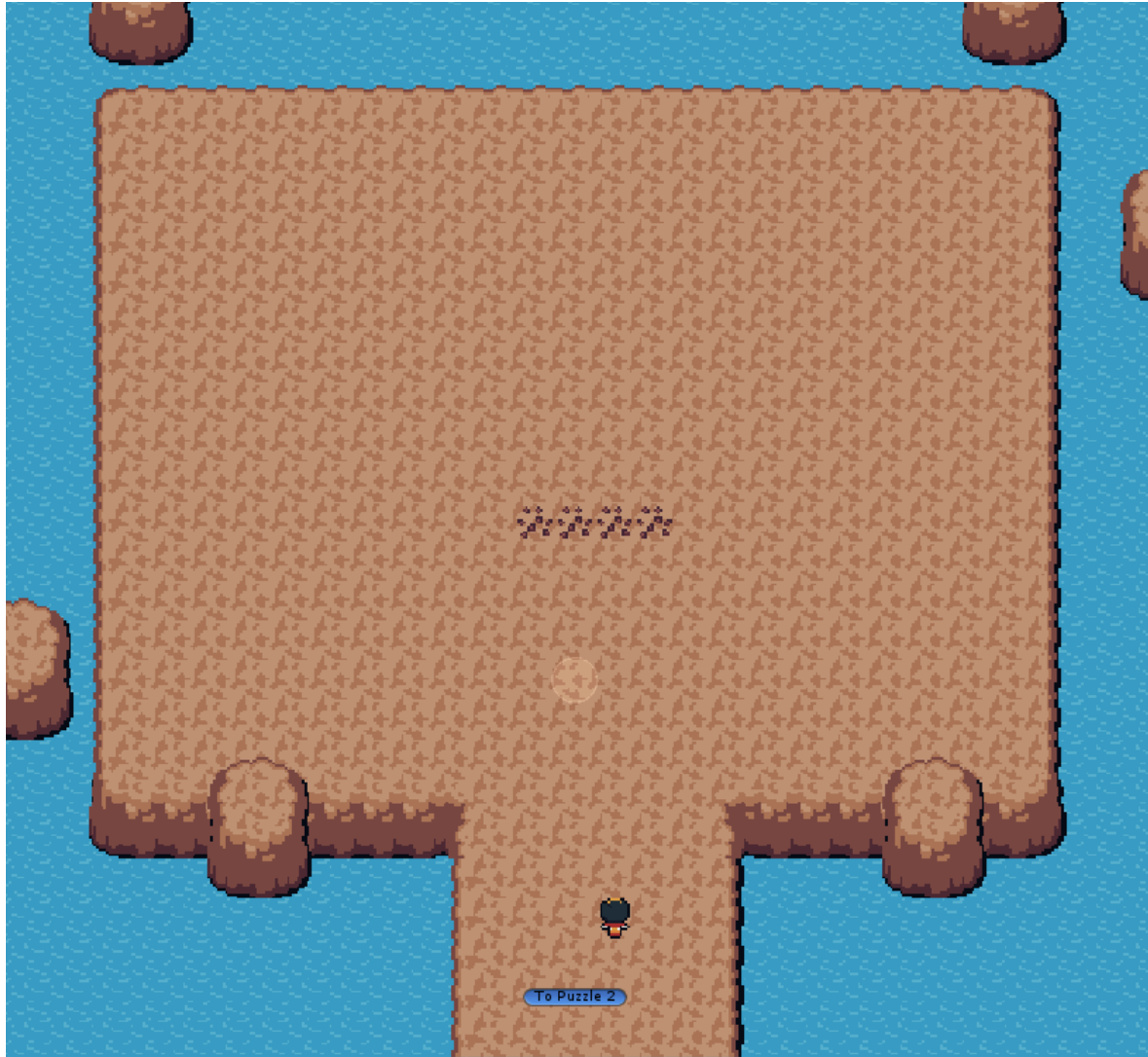
Yendo hacia abajo en la escena del lobby, el mapa se divide en cuatro posibles rutas. Hacia la izquierda se encuentra el puzzle número 2. Este puzzle consiste en varias placas con las cuales el jugador puede interactuar para prender o apagar su luz. Estas a su vez pueden afectar las placas contiguas y el puzzle se considera resuelto cuando todas las placas están prendidas. La escena se muestra en la figura 50.0.



**Figura 50.0:** Escena del Puzzle 2

**Fuente:** Elaboración Propia

Desde este se puede ver una cueva que es el acceso al puzzle número 3, ambientado dentro de una caverna, como se ve en la figura 51.0. Al entrar en la cueva se puede ver a simple vista la recompensa de las semillas del puzzle. Sin embargo, al intentar el jugador ir a recolectarlas, se da cuenta rápidamente de que existe un conjunto de paredes invisibles que debe sortear con astucia para llegar al centro del mapa.

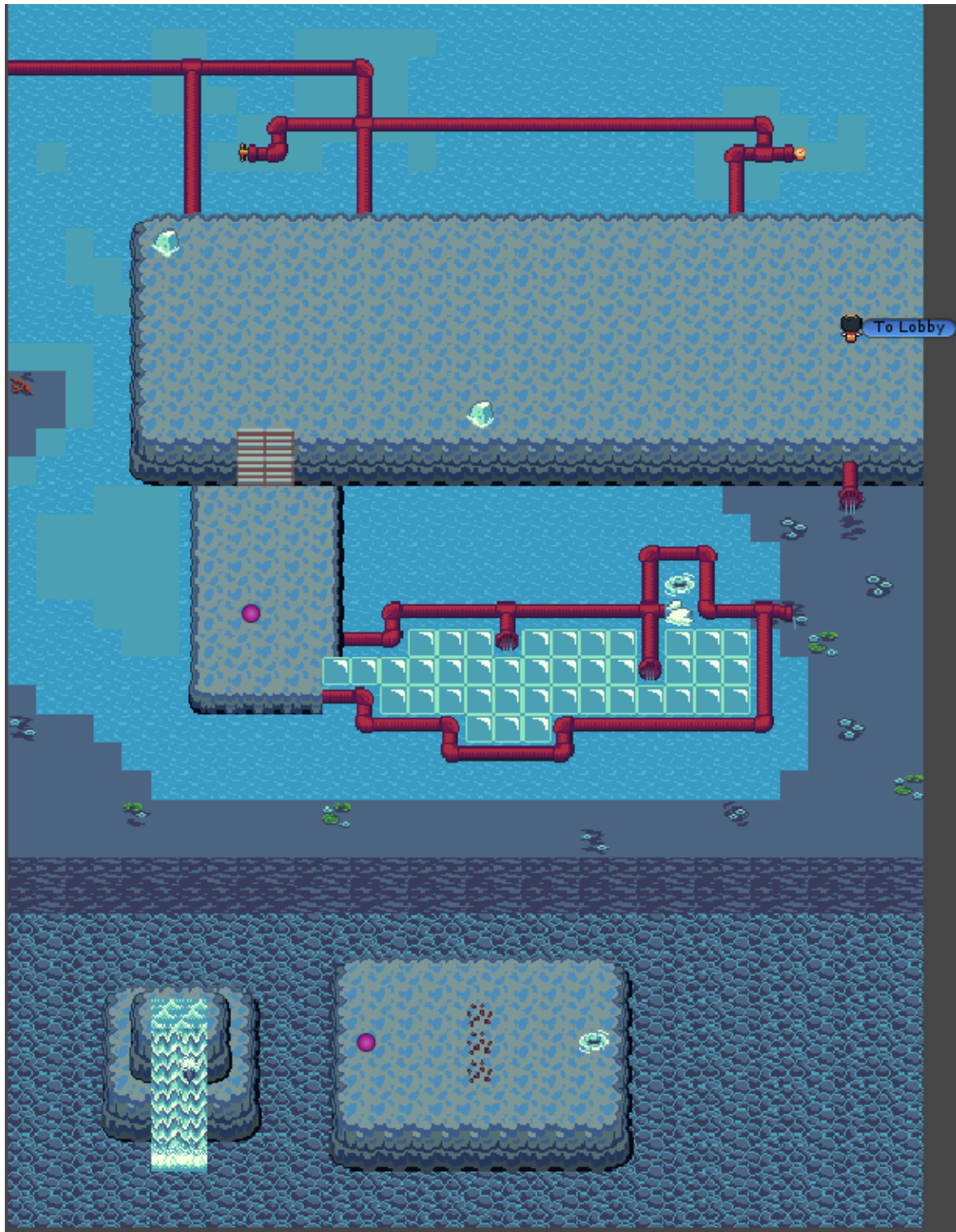


**Figura 51.0:** Escena del Puzzle 3

**Fuente:** Elaboración Propia

Caminando hacia abajo a la izquierda en el Lobby se accede al puzzle número 5, ambientado dentro de una zona de agua, hielo y tuberías. Para completar este puzzle el jugador debe caminar en un conjunto de hielos que se rompen al pasar por arriba. Si el jugador camina sobre el agua o llega al final sin haber cubierto la totalidad de los hielos, el puzzle lo transporta al principio. Si se camina sobre todos los hielos antes de llegar al final, en cambio, el puzzle transporta al jugador a otra isla en donde se puede recolectar la recompensa. La escena completa se puede ver en la figura 52.0.





**Figura 52.0:** Escena del Puzzle 5**Fuente:** Elaboración Propia

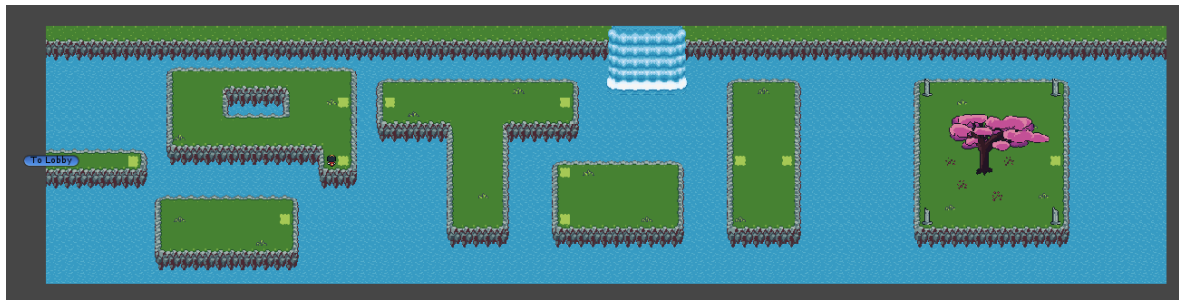
Comenzando nuevamente desde el Lobby, caminando hacia abajo a la derecha, se encuentra una escena similar a un estuario o una zona con cuerpos de agua. Esta se muestra en la figura 53.0. En este caso se trata del puzzle número 4, en el que se debe rotar ciertos paneles con el dibujo de una flor para formar la figura al derecho. Una vez que todas las placas se encuentren en la posición correcta aparecerá la recompensa.



**Figura 53.0:** Escena del Puzzle 4

**Fuente:** Elaboración Propia

Finalmente, si se avanza desde el Lobby hacia la derecha se encuentra con el puzzle número 1, graficado en la figura 54.0. Este consiste de una serie de islas flotantes en donde el jugador debe resolver un laberinto de plataformas que lo transportan de una a otra. Al encontrar el camino correcto, se llega a la última isla, la cual contiene un árbol grande y la recompensa del puzzle.



**Figura 54.0:** Escena del Puzzle 1

**Fuente:** Elaboración Propia

En cualquier momento de cualquier escena, el jugador puede presionar el botón de “Menú” arriba a la derecha para abrir el menú de opciones, el cual contiene los botones para salir al menú principal o salir del juego. Este menú se puede ver en la figura 55.0.



**Figura 55.0:** Escena del Menú Principal

**Fuente:** Elaboración Propia

En el caso de la plataforma web, la experiencia de usuario comienza por

## 9. Interfaz de usuario

La interfaz de usuario, mostrada en la figura 56.0, se puede considerar en tres partes o secciones principales. Por un lado se encuentra la sección de medidores que contiene el contador de monedas (figura 57.0) y el medidor de agua (figuras 58.0, 59.0, 60.0). El contador de monedas es simplemente un número que indica la cantidad de monedas que se tiene actualmente, el cual se puede consultar en cualquier momento a simple vista. El medidor de agua, por otro lado, mide el nivel de agua presente en la regadera. Este se representa con una barra de color celeste. El nivel del agua se actualiza cuando se riega cultivos o cuando se recarga la regadera para que el jugador pueda saber cuando ir a recargar.



**Figura 56.0:** UI jugador completa

**Fuente:** Elaboración Propia



**Figura 57.0:** Contador de Monedas en la UI

**Fuente:** Elaboración Propia





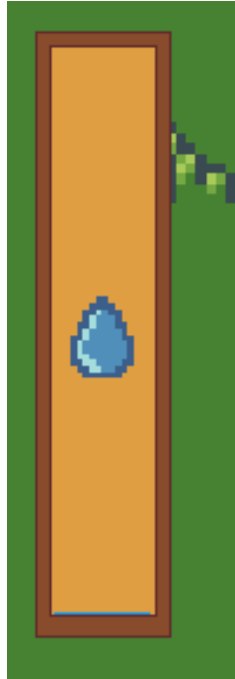
**Figura 58.0:** Medidor de Agua Lleno en la UI

**Fuente:** Elaboración Propia



**Figura 59.0:** Medidor de Agua a medio consumir en la UI

**Fuente:** Elaboración Propia



**Figura 60.0:** Medidor de Agua vacío consumir en la UI

**Fuente:** Elaboración Propia

Por otro lado se tiene la barra de inventario, como se ve en la figura 61.0. Esta presenta lugar para 9 tipos de Ítem distintos, pero puede contener cualquier cantidad de ítems idénticos. Para poder utilizar estos ítems se tiene una posición del inventario seleccionada, marcada con una flecha de color roja. Este marcador se puede cambiar entre las diferentes posiciones para cambiar el ítem con el que se está interactuando.



**Figura 61.0:** Inventario en la UI

**Fuente:** Elaboración Propia

Finalmente, se tiene el menú en la esquina superior derecha. Este se muestra en las figuras 62.0 y 63.0 Este menú se puede abrir y cerrar y presenta las opciones de salir al menú principal o salir del juego.



**Figura 62.0:** Boton de Menu en la UI

**Fuente:** Elaboración Propia



**Figura 63.0:** Menú Abierto en la UI

**Fuente:** Elaboración Propia

Paralelamente a estas interfaces de usuario, existe un último menú que se abre únicamente cuando se interactúa con la tienda en la escena de la granja. Este menú muestra los diferentes Ítems disponibles en la tienda, con su foto y su precio. Al hacer click en cualquiera de estos, se corrobora que el jugador tenga los fondos suficientes y se realiza la compra, agregándole al inventario y bloqueando este Ítem de ser comprado nuevamente. Esta acción se ve en las figuras 64.0 y 65.0.



**Figura 64.0:** UI de la tienda con todos sus ítems

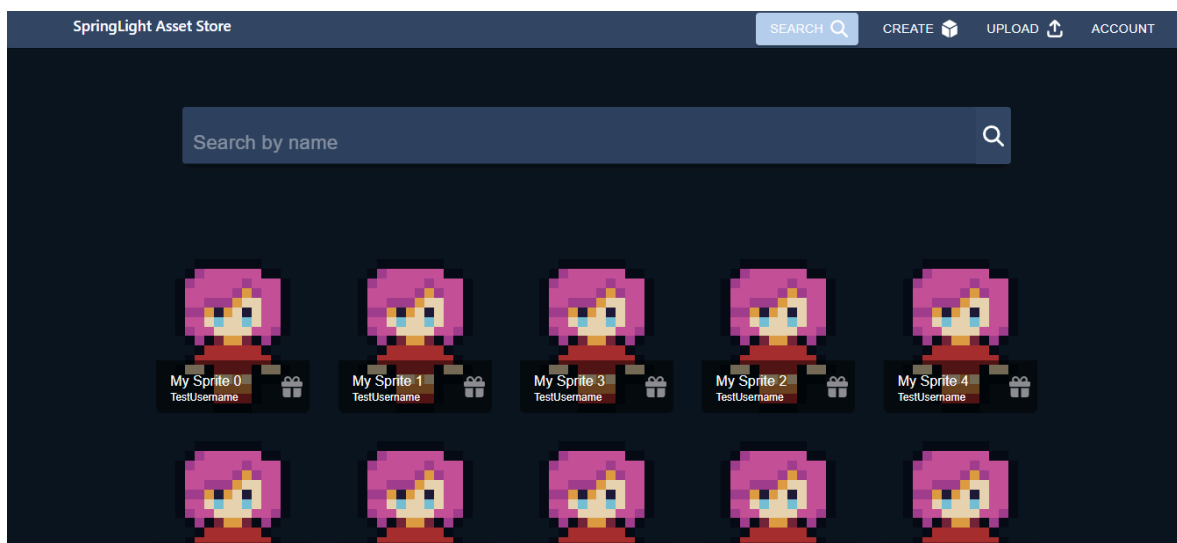
**Fuente:** Elaboración Propia



**Figura 65.0:** UI de la tienda con un ítem comprado

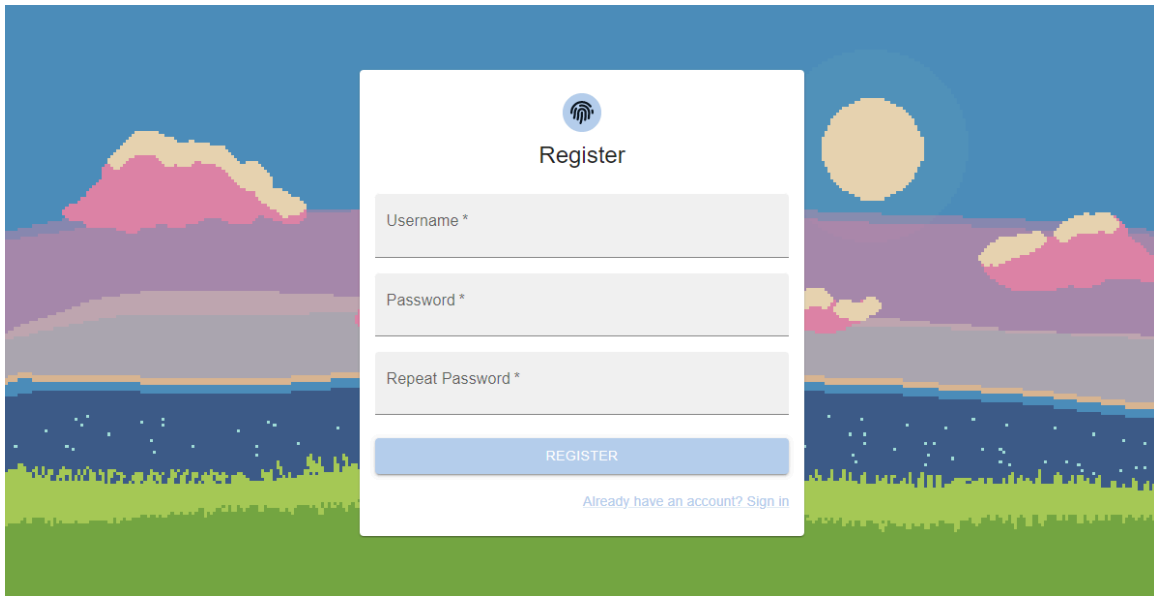
**Fuente:** Elaboración Propia

En lo que respecta a la plataforma web, esta presenta dos interfaces de usuario distintas. En primer lugar se encuentra el cliente web, desde el cual se crean y buscan las nuevas skins. Las diferentes secciones de este se muestran en las figuras 75.0, 76.0, 77.0, 78.0, 79.0, 80.0, 81.0, 82.0 y 83.0.



**Figura 75.0:** Página principal del Marketplace web

**Fuente:** Elaboración Propia



Register

Username \*

Password \*

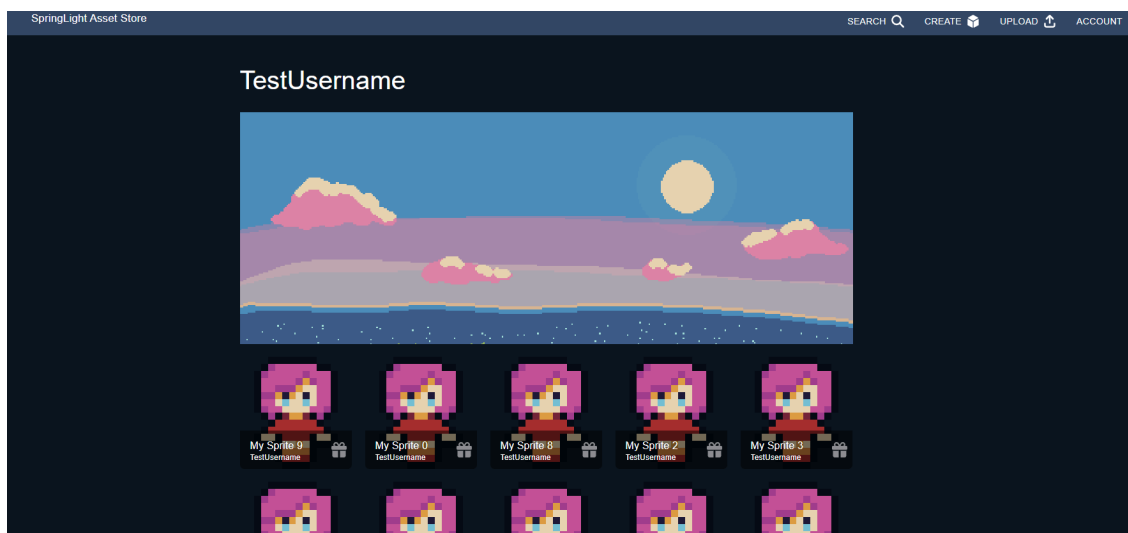
Repeat Password \*

REGISTER

[Already have an account? Sign in](#)

**Figura 77.0:** Formulario de Registro del Marketplace web

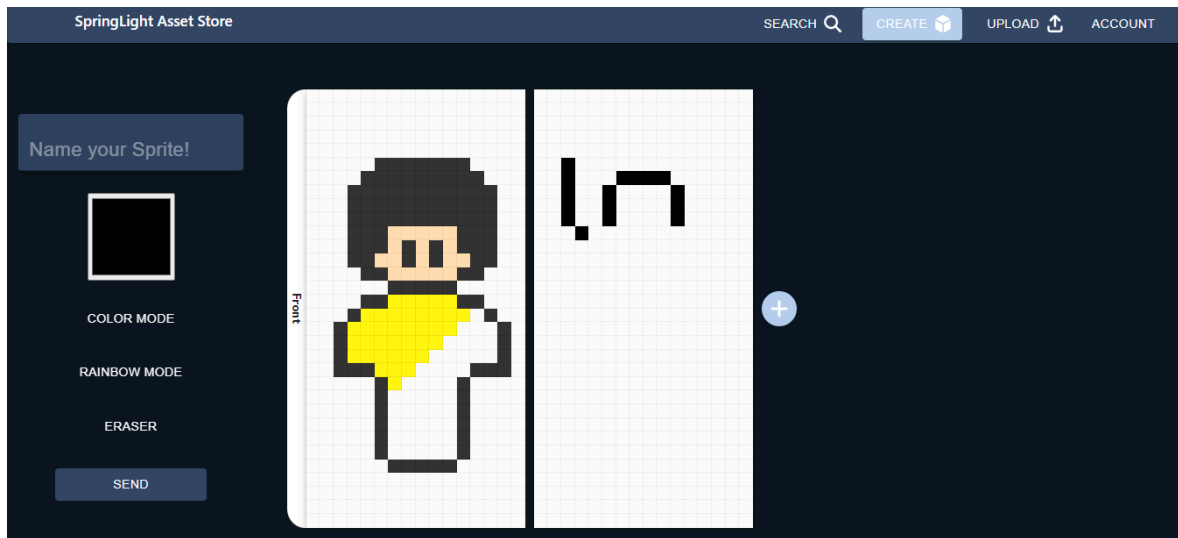
**Fuente:** Elaboración Propia



**Figura 78.0:** Perfil de un usuario

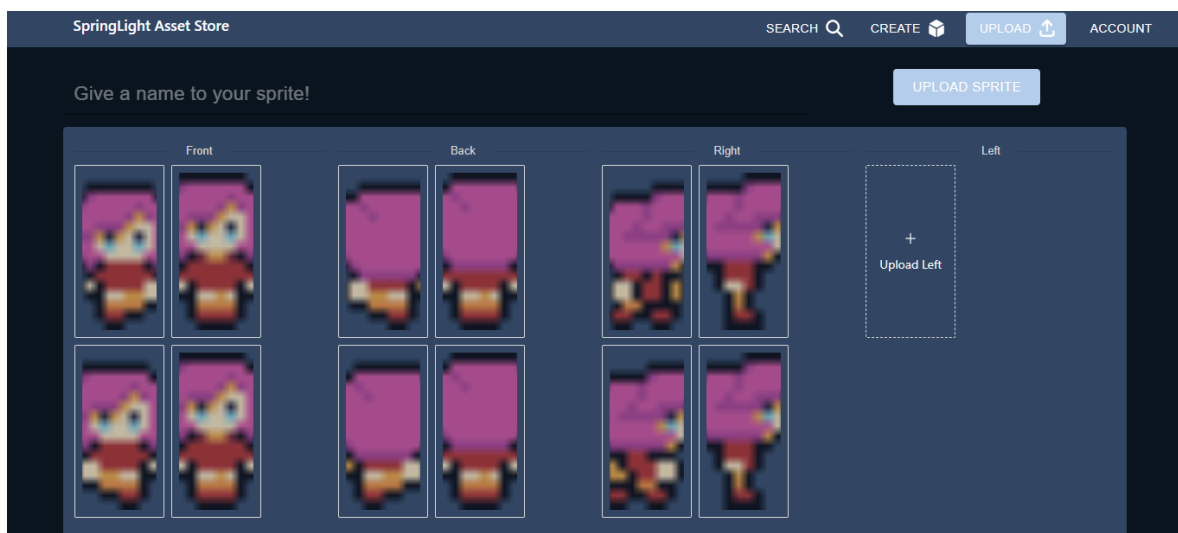
**Fuente:** Elaboración Propia





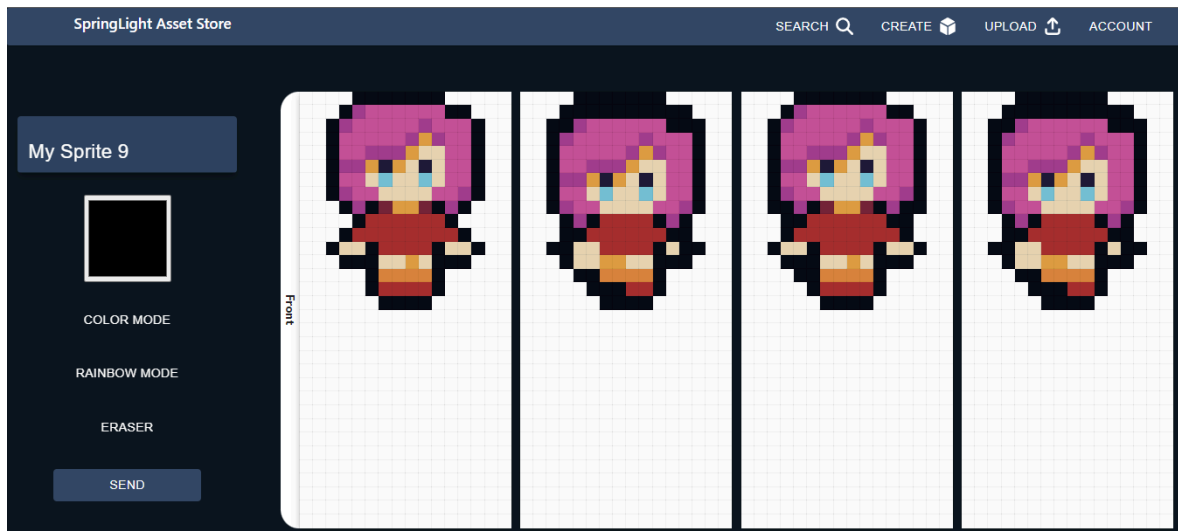
**Figura 79.0:** Creación de Sprite por medio de Dibujador

**Fuente:** Elaboración Propia



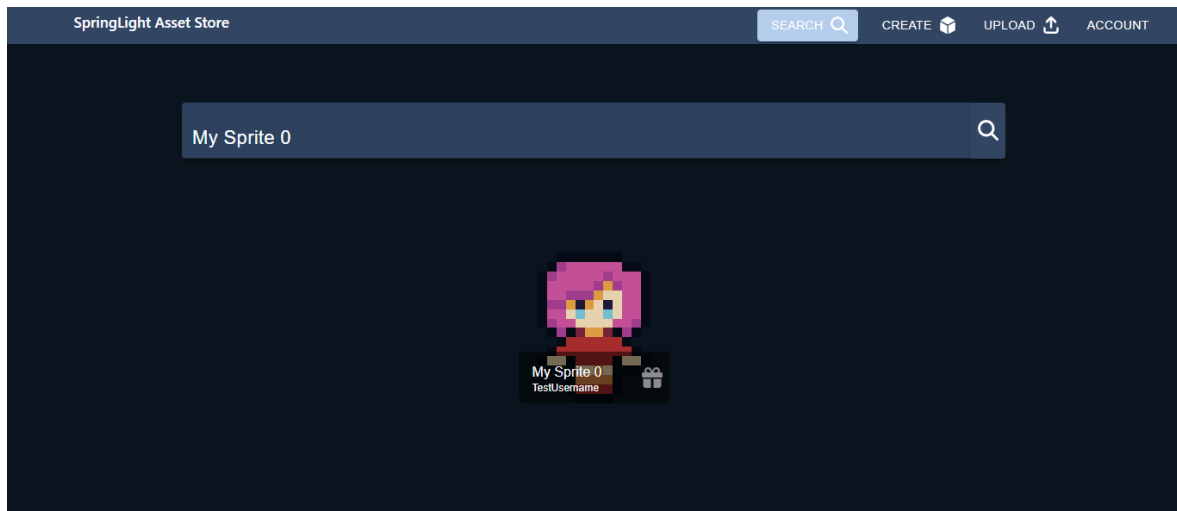
**Figura 80.0:** Creación de Sprite por medio de Carga de Imágenes

**Fuente:** Elaboración Propia



**Figura 81.0:** Edición de un Sprite previamente creado

**Fuente:** Elaboración Propia



**Figura 82.0:** Búsqueda de Sprites

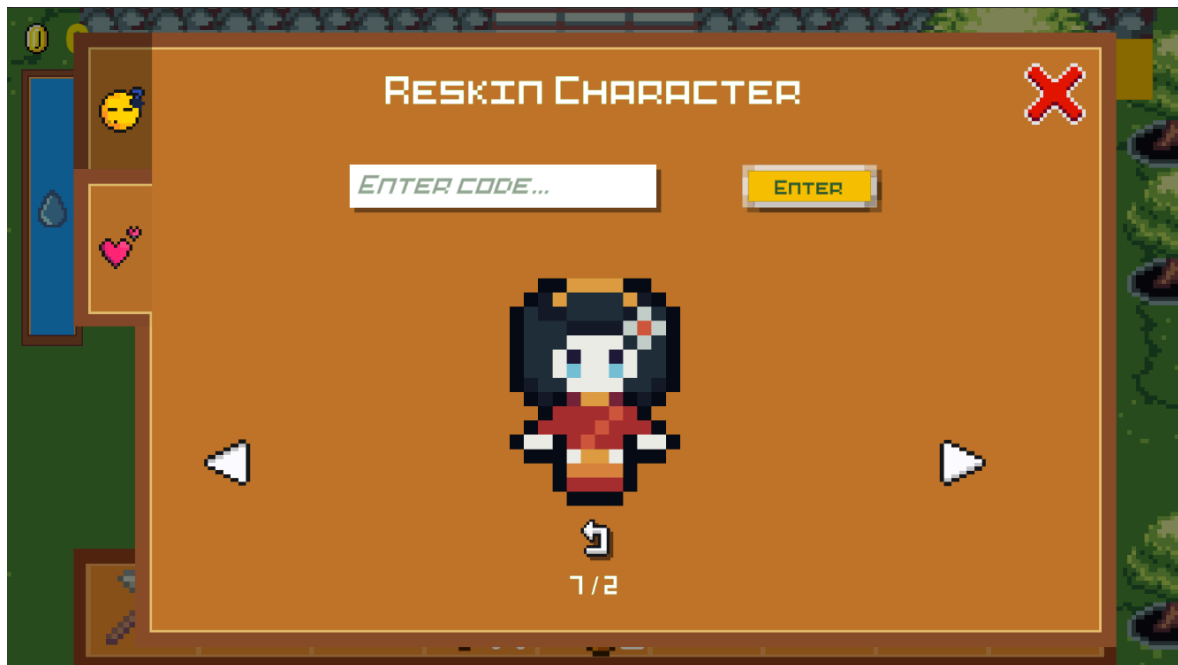
**Fuente:** Elaboración Propia



**Figura 83.0:** Vistas de un Sprite y su obtención de código

**Fuente:** Elaboración Propia

Por otro lado, se encuentra la interfaz de usuario que se encuentra dentro del juego Spring Light. En la figura 84.0 se muestra el menú con la Skin que trae por defecto. La cantidad de skins que posee el usuario se puede observar en la parte inferior, junto con el número de la skin actual. El usuario es capaz de alternar entre estas Skins utilizando las flechas a la izquierda y derecha de la previsualización del personaje.



**Figura 84.0:** Menu UI en Spring Light para el cambio de Skin

**Fuente:** Elaboración Propia

Al cargar un código en la barra de arriba y presionar el botón de Enter, el juego activa el mecanismo de descarga de Sprite, pidiendo la información al servidor y compilando una nueva Sprite Library Asset. El resultado se puede ver en las figuras 85.0 y 86.0.



**Figura 85.0:** Menu UI en Spring Light al descargar una skin

**Fuente:** Elaboración Propia



**Figura 86.0:** El personaje utilizando la Skin previamente descargada desde el servidor web

**Fuente:** Elaboración Propia

## 10. Dificultades encontradas y soluciones propuestas

### Cuadro de Diálogo

A lo largo del desarrollo del videojuego surgieron diversas dificultades que hubo que sortear para continuar con el desarrollo. Una de las funcionalidades a desarrollar, el cuadro de diálogo, presentó algunas tomas de decisiones al momento de implementación. En primer lugar se encontraba la decisión sobre si realizar la implementación a mano o conseguir algún asset, paquete o extensión de terceros que ya proveyera esta funcionalidad.

La ventaja de realizar la implementación a mano era que se podría adaptar perfectamente a las necesidades del producto y estaría abierta a cambios si lo fueran necesarios. La desventaja de esto es que consumiría mucho tiempo y además existiría una pieza más de software que mantener. Por otro lado, al utilizar un paquete de terceros, se podría implementar esta funcionalidad mucho más rápido y quitando la responsabilidad de hacer ajustes o arreglos.

Luego de investigar qué opciones había para utilizar se encontraron principalmente dos paquetes en el Unity store que ofrecían estas funcionalidades de forma gratuita: RPGTalk e Ink. RPGTalk provee más funcionalidades que Ink, ya que este último solo se encarga de los diálogos mientras que el primero ofrece también toda la funcionalidad de manejo de UI. Analizando las ventajas finalmente se tomó la decisión de utilizar RPGTalk. Si bien esto resulta en una marcada pérdida de flexibilidad respecto a las otras dos opciones, la envergadura del proyecto habilita tal decisión ya que los requerimientos funcionales del cuadro de diálogo son de los más simples.

## Manejo de Escenas, guardado y serialización

Para lograr integrar las diferentes escenas en un solo mapa, se utilizó un sistema que cambia de escena según la posición del jugador. Así, al llegar al borde de un mapa o entrar en una zona de cambio de escena, como una puerta, se dispara dicho mecanismo en el motor de Unity, dando al jugador la ilusión de haber cambiado de mapa. A este objeto especial del juego se lo llamó Scene Transition Position.

Un problema interesante que apareció fue el de guardar el progreso cuando se realiza la transición. Para esto es necesario guardar los datos de esa escena en particular para luego serializarlos. Esto presentó dos problemas:

En primer lugar, los objetos deben recibir un aviso para guardarse en el momento de la transición. Sin embargo, Unity no provee un evento para el momento antes de cambiar la escena, solo para después. En este caso la solución fue hacer que el objeto SceneTransitionPosition implemente el patrón Observable, para así suscribir todos los objetos pertinentes a este y que les avise antes de realizar la transición.

El segundo problema radicaba en que cada escena tiene objetos propios que se deben guardar, con lo cual era difícil saber cuáles objetos deben guardarse en cada momento. La solución para esto fue que los objetos puedan guardarse y luego cargarse a sí mismos, utilizando un PersistenceManager con el patrón Observable, el cual les informa a los objetos cuando guardarse y cuando cargarse.

El siguiente problema grande vino por el hecho de persistir la información guardada al disco. Esto se realiza mediante la serialización de la información guardada a un string de texto. En Unity, se pueden utilizar varios formatos para persistir la información, como JSON, XML, y binario. JSON y XML son formatos de texto plano que son fáciles de leer y editar manualmente. Además, son compatibles con una amplia variedad de lenguajes y plataformas, lo que los hace ideales para el intercambio de datos entre diferentes sistemas.

Sin embargo, su tamaño puede ser más grande que el de los formatos binarios y pueden ser un poco más lentos al cargar y guardar datos. Binario es un formato de datos



compacto y rápido para cargar y guardar datos. Sin embargo, no es legible por humanos y no es tan fácil de editar manualmente como los formatos de texto. También es menos compatible con diferentes plataformas y lenguajes.

Al final se tomó la decisión de utilizar el formato Json ya que el tamaño y velocidad de la información de progreso era irrelevante para el proyecto. En este caso se intentó primero con la utilidad por defecto de Unity, JsonUtility, pero rápidamente demostró no ser adecuada ya que solo puede serializar los tipos de objeto más simples como los tipos primitivos o clases simples sin herencia.

En este caso era necesario serializar objetos complejos con referencias cíclicas, polimorfismo, clases abstractas y algunas referencias a ScriptableObjects. Se investigó sobre este punto qué alternativas de serialización existían. Principalmente existen dos opciones, JSON.NET y Odin Serializer.

JSON.NET es un serializador open source que permite serializar objetos complejos en cualquier lenguaje o entorno .NET. Si bien no fue creado específicamente para Unity, es muy popular debido a su gran capacidad y el hecho de ser gratuito. Odin Serializer, por otro lado, es un software comercial de la familia de productos Odin. Está orientado específicamente a Unity y provee funciones un poco más complejas que JSON.NET. Sin embargo, es un software comercial y su precio es bastante elevado (aprox. U\$S 55). Luego de ver estas opciones se optó por el serializador JSON.NET, el cual cubre bastante bien las necesidades del proyecto.

No obstante, al no ser una librería preparada para Unity, hay algunas clases de objetos que no puede serializar de forma nativa, como los ScriptableObjects. En estos casos provee una forma, mediante el patrón Strategy, de implementar un serializador propio para cada clase que uno quiera y realizar la serialización a mano. Aprovechando que los ScriptableObjects existen en el directorio como assets, se pueden serializar simplemente como su nombre y luego deserializar cargándolos desde una carpeta de Resources con el proceso inverso. Esta forma es muy sencilla de implementar e incluso ahorra espacio de almacenamiento.

## Pipeline de Gráficos

Si bien el modelo del pipeline cambió a lo largo del proyecto, se logró alcanzar un modo de trabajo eficiente aprendiendo ciertas lecciones. En primer lugar conviene comenzar los gráficos con algo simple para probar si funciona en el juego. Esto se puede hacer con assets de internet o con formas y colores simples dibujados en Aseprite a mano. Para ilustrar esto, durante las primeras semanas de desarrollo la imagen del jugador fue la de la figura 66.0.



**Figura 66.0:** Imagen del jugador que se utilizó a modo de prueba durante las primeras semanas del desarrollo

**Fuente:** Elaboración Propia

Esta imagen fue muy sencilla de crear pero permite tener una imagen con el tamaño máximo que se le asignó al jugador para poder probar con facilidad las mecánicas de movimiento y de colisiones con objetos. Más tarde, a lo largo del proyecto se reemplazó por nuevas versiones del personaje cuando se necesitó comenzar con las animaciones de movimiento en las cuatro direcciones.

Otra lección aprendida respecto al pipeline de gráficos fue el guardado de los archivos en directorios separados. Conviene tener un directorio con los archivos .aseprite catalogados según si forman parte del mundo, son objetos, personaje, UI, etc. Luego por otro lado, tener otro directorio separado en donde se guarden las versiones exportadas a .png que utiliza el juego realmente. Cuando el juego se compila no hace uso de los archivos .aseprite, con lo cual no es necesario incluirlos en el build. Sin embargo, si es una buena idea incluirlos en el control de versiones, en este caso el repositorio de Git.

## Generación de Skins de la plataforma web

En un primer momento, la forma de crear los Sprite Library Assets en el juego en Unity era mediante el AssetDatabase. Este es un módulo de Unity que permite crear assets en cualquier directorio y realizar cambios a los archivos del proyecto. Este componente se utilizó debido a que nunca se había utilizado esta funcionalidad en Unity previamente, con lo cual proveyo una solución posible al problema. Sin embargo, entrado en el desarrollo, fue evidente que se había comprendido mal el funcionamiento de este componente, ya que su objetivo no era agregar funcionalidades al juego en tiempo de ejecución, sino agregar funcionalidades al editor de Unity en tiempo de desarrollo. Por este motivo este componente no funciona fuera del editor cuando se compila el juego.

Frente a este dilema se tuvo que rehacer todo el módulo de creación de Sprites manualmente, lo que finalmente se acabó llamando el Sprite Library Generator.

## 11. Conclusiones

En conclusión se lograron cumplir los objetivos propuestos desde el punto de vista práctico y teórico durante el desarrollo del videojuego. Se investigó mucho sobre los distintos aspectos que involucran la creación de un videojuego y se pusieron luego en práctica para no solo consolidarlos sino también probar su efectividad en la práctica.

Se logró cumplir con todas las funcionalidades planteadas desde un principio, aunque algunos se implementaron de forma diferente a la imaginada o planeada en un principio. Esto es una característica inherente al proceso de creación de cualquier videojuego e incluso cualquier programa de software, ya que el proyecto evoluciona y toma su curso a medida en que se implementa. Por ejemplo, en un principio el mapa del mundo se había imaginado en una montaña, pero luego ese concepto se cambió por un bosque para dar más versatilidad al ambiente de las escenas.

## Trabajo con Unity

Trabajar con Unity fue una experiencia satisfactoria. La plataforma es intuitiva y fácil de aprender, lo que permite a los desarrolladores centrarse en la creación de sus juegos sin tener que preocuparse por problemas técnicos complejos. Además, Unity cuenta con una amplia documentación y comunidad, lo que significa que siempre es posible encontrar soluciones a cualquier problema que pueda surgir durante el desarrollo.

Otro aspecto positivo de Unity es su sistema de paquetes, que permite que los desarrolladores compartan módulos de código reutilizables entre diferentes productos. Esto lo diferencia de otras soluciones o frameworks ya que no todos los motores poseen un ecosistema tan vasto. Esto es posible gracias a que el motor es abierto a la extensión por parte de los desarrolladores, a los cuales permite editar, alterar y agregar muchos tipos de características. Se puede crear menus interactivos y herramientas que funcionen solo en modo de edición para ayudar con el desarrollo mismo del videojuego, sin necesidad de incluirlos en el producto final. Otro paquete provisto por el motor que está muy bien integrado es el Test Framework, el cual se puede agregar a cualquier proyecto y permite correr tests tanto en modo edición como en modo ejecutable. Esto está muy bien integrado con el programa y lo hace muy simple a la hora de correrlo local.

Habiendo dicho esto, es importante tener en cuenta también que si existen algunas cosas que no son tan fácilmente alterables por el usuario y que el framework coerce a utilizar en cierta medida. Ejemplos de esto pueden ser el ciclo de vida de los objetos, ciertos tipos de objetos como los Game Objects o los objetos con transforms.

Adicionalmente, Unity viene acompañado de algunas desventajas. En primer lugar, la programación en Unity está fuertemente inclinada hacia la programación orientada a objetos. Esto significa que uno no puede decidir con facilidad utilizar, por ejemplo, un paradigma funcional, por más que C# si lo permita. Si bien la programación orientada a objetos puede ser muy útil si se trabaja con los principios de diseño adecuados, hay situaciones en las que alguien podría querer utilizar diferentes lenguajes o estrategias que existen en el mundo de la programación.

## Evolución del Producto


Otra cosa a mencionar también son algunas mejoras que se podrían introducir sobre el producto actual. Por ejemplo, se podría agregar una escena dentro de la casa en la granja que permita al jugador amueblarla con Ítems conseguidos en el mundo. También se puede agregar animales en la granja, mejoras a las herramientas, etc.

Paralelamente, a lo largo del desarrollo del juego se descubrió que Unity tiene otras herramientas que permiten crear juegos en 2D que no son las estándar. Una de ellas se denomina URP (Universal Render Pipeline) y permite opciones mucho más complejas a la hora de realizar efectos especiales con luz o texturas. Sería interesante investigar estas opciones y cómo se podrían aplicar para mejorar Spring Light.

Por otro lado, la plataforma web también se podría expandir a nuevas funcionalidades. Por su diseño, sería interesante experimentar permitiéndole al usuario crear diferentes tipos de assets para personalizar su experiencia en el juego. Se podría cargar música, texturas del terreno y eventualmente hasta expansiones del juego realizadas por la comunidad, mods, etc.

## 12. Bibliografía

### Referencias y Bibliografía

-  (2022, August 10). YouTube. Retrieved February 3, 2023, from <https://ar.pinterest.com/pin/trix-on-twitter--31666003619300943/>
- About GitHub-hosted runners.* (n.d.). GitHub Docs. Retrieved February 9, 2023, from <https://docs.github.com/en/actions/using-github-hosted-runners/about-github-hosted-runners>
- Activation.* (n.d.). GameCI. Retrieved February 9, 2023, from <https://game.ci/docs/github/activation>
- Animator Sprite Swap: Sprite Animation Reskin Tutorial.* (2020, January 8). YouTube. Retrieved January 11, 2023, from <https://www.youtube.com/watch?v=3SD9iFN4B8U>
- Aprendizaje basado en juegos.* (n.d.). Wikipedia. Retrieved February 9, 2023, from [https://es.wikipedia.org/wiki/Aprendizaje\\_basado\\_en\\_juegos](https://es.wikipedia.org/wiki/Aprendizaje_basado_en_juegos)
- Are Unity's Tile Features Good Enough to Make Real Games?* (2021, July 24). YouTube. Retrieved January 11, 2023, from <https://www.youtube.com/watch?v=wRkT10D-Quo>
- Are Unity's Tile Features Good Enough to Make Real Games?* (2021, July 24). YouTube. Retrieved January 11, 2023, from <https://www.youtube.com/watch?v=wRkT10D-Quo>
- Aseprite.* (n.d.). Aseprite - Animated sprite editor & pixel art tool. Retrieved February 3, 2023, from <https://www.aseprite.org/>



*The Aseprite 1.3 Update is a Literal Game Changer.* (2021, March 20). YouTube. Retrieved January 11, 2023, from [https://www.youtube.com/watch?v=TL\\_JZluydas](https://www.youtube.com/watch?v=TL_JZluydas)

*The Aseprite 1.3 Update is a Literal Game Changer.* (2021, March 20). YouTube. Retrieved January 11, 2023, from [https://www.youtube.com/watch?v=TL\\_JZluydas](https://www.youtube.com/watch?v=TL_JZluydas)

*An Aseprite Crash Course In 30 Minutes.* (2021, February 7). YouTube. Retrieved January 11, 2023, from <https://www.youtube.com/watch?v=59Y6OTzNrhk>

*An Aseprite Crash Course In 30 Minutes.* (2021, February 7). YouTube. Retrieved January 11, 2023, from <https://youtu.be/59Y6OTzNrhk>

*asp.net - Newtonsoft.Json serialization returns empty json object.* (2015, March 12). Stack Overflow. Retrieved January 26, 2023, from <https://stackoverflow.com/questions/29003215/newtonsoft-json-serialization-returns-empty-json-object>

*Assemblies in .NET.* (2022, September 8). Microsoft Learn. Retrieved January 11, 2023, from <https://docs.microsoft.com/en-us/dotnet/standard/assembly/>

*Assembly Definitions in Unity.* (2020, May 17). YouTube. Retrieved January 11, 2023, from <https://www.youtube.com/watch?v=HYqOSkHl674>

*Auto-Tiling In ~Tiled Editor~ [ Stream Highlight ].* (2019, June 5). YouTube. Retrieved January 11, 2023, from <https://www.youtube.com/watch?v=PU1u1k38w5k>

Baer, R. (n.d.). *Magnavox Odyssey*. Wikipedia. Retrieved February 3, 2023, from [https://es.wikipedia.org/wiki/Magnavox\\_Odyssey](https://es.wikipedia.org/wiki/Magnavox_Odyssey)

Báez, F., Burlando, F., & Franco, B. (2013, December 21). *Las aplicaciones "freemium" son las más rentables en la AppStore*. Infobae. Retrieved February 9, 2023, from

<https://www.infobae.com/2013/12/21/1532555-las-aplicaciones-freemium-son-las-mas-rentables-la-appstore/>

*Basics of Universal Render Pipeline (URP) in Unity 2019.3.* (2020, March 2). YouTube.

Retrieved January 11, 2023, from [https://youtu.be/HqaQJfuK\\_u8](https://youtu.be/HqaQJfuK_u8)

Boylls, T. (2022, December 15). *A Detailed Guide to Getting Your Game on Steam.* wikiHow.

Retrieved January 28, 2023, from

<https://www.wikihow.com/Get-Your-Game-on-Steam>

*Building code-free shaders in Unity with Shader Graph.* (2022, December 18). YouTube.

Retrieved January 11, 2023, from <https://youtu.be/mRDG5sQYjdo>

Ceci, L. (2022, September 13). *Global app store commission rates 2022.* Statista. Retrieved

January 11, 2023, from

<https://www.statista.com/statistics/975776/revenue-split-leading-digital-content-store-worldwide/>

*Character Customization Using Spritesheets - Unity Tutorial.* (2021, August 6). YouTube.

Retrieved January 11, 2023, from <https://www.youtube.com/watch?v=t2UUHl4eudI>

*Cinemachine.* (n.d.). Unity. Retrieved January 11, 2023, from

<https://unity.com/es/unity/features/editor/art-and-design/cinemachine>

*Continuous Integration & Unit Tests | The Open Augmented Reality Teaching Book -*

*Create and Code Augmented Reality!* (n.d.). Code Reality. Retrieved January 11, 2023, from

[https://codereality.net/ar-for-eu-book/chapter/development/tools/unity/advanced/ci\\_unity/](https://codereality.net/ar-for-eu-book/chapter/development/tools/unity/advanced/ci_unity/)

- Custom JsonConvert*. (n.d.). Json.NET. Retrieved January 26, 2023, from <https://www.newtonsoft.com/json/help/html/CustomJsonConverter.htm>
- Custom JsonConvert<T>*. (n.d.). Json.NET. Retrieved January 26, 2023, from <https://www.newtonsoft.com/json/help/html/CustomJsonConverterGeneric.htm>
- de Icaza, M. (2011, March 7). *GDC 2011 - Miguel de Icaza*. tirania.org. Retrieved February 9, 2023, from <https://tirania.org/blog/archive/2011/Mar-07.html>
- Editor de gráficos rasterizados*. (n.d.). Wikipedia. Retrieved February 3, 2023, from [https://es.wikipedia.org/wiki/Editor\\_de\\_gr%C3%A1ficos\\_rasterizados](https://es.wikipedia.org/wiki/Editor_de_gr%C3%A1ficos_rasterizados)
- Encrypted secrets*. (n.d.). GitHub Docs. Retrieved February 9, 2023, from <https://docs.github.com/en/actions/security-guides/encrypted-secrets>
- Fast C# Scripting in Unity with JetBrains Rider! - Overview*. (2019, August 15). YouTube. Retrieved January 11, 2023, from <https://www.youtube.com/watch?v=2CvSooOhlWI>
- Final Fantasy (franquicia)*. (n.d.). Wikipedia. Retrieved February 3, 2023, from [https://es.wikipedia.org/wiki/Final\\_Fantasy\\_\(franquicia\)](https://es.wikipedia.org/wiki/Final_Fantasy_(franquicia))
- 5 Minute DIALOGUE SYSTEM in UNITY Tutorial*. (2021, March 19). YouTube. Retrieved January 11, 2023, from <https://www.youtube.com/watch?v=8oTYabhj248>
- Flappy Bird*. (n.d.). Wikipedia. Retrieved January 26, 2023, from [https://es.wikipedia.org/wiki/Flappy\\_Bird](https://es.wikipedia.org/wiki/Flappy_Bird)
- French, L. (2020, January 9). *How to change a Sprite from a script in Unity (with examples)*. Game Dev Beginner. Retrieved January 11, 2023, from [https://gamedevbeginner.com/how-to-change-a-sprite-from-a-script-in-unity-with-examples/#change\\_sprite\\_from\\_script](https://gamedevbeginner.com/how-to-change-a-sprite-from-a-script-in-unity-with-examples/#change_sprite_from_script)

*Game architecture with ScriptableObjects | Open Projects Devlog.* (2020, December 19).

YouTube. Retrieved January 11, 2023, from <https://youtu.be/WLDgtRNK2VE>  
*game-ci/unity-actions: Github actions for testing and building Unity projects.* (n.d.). GitHub.

Retrieved January 11, 2023, from <https://github.com/game-ci/unity-actions>

*Game Development Isn't a Game.* (2023, January 6). ? - YouTube. Retrieved January 11, 2023, from  
<https://www.youtube.com/watch?v=jIFkxfkICO8&list=PLaWi-C2BWT5q9I7EVXQDCI10DEgyFWl9n&index=2>

*Geisha.* (n.d.). Wikipedia. Retrieved February 3, 2023, from  
<https://es.wikipedia.org/wiki/Geisha>

*Git.* (n.d.). Wikipedia. Retrieved February 3, 2023, from <https://en.wikipedia.org/wiki/Git>

*GitHub.* (n.d.). Wikipedia. Retrieved February 3, 2023, from  
<https://en.wikipedia.org/wiki/GitHub>

*Github.* (n.d.). GitHub: Let's build from here · GitHub. Retrieved February 3, 2023, from  
<https://github.com/>

*Google will reduce Play Store cut to 15 percent for a developer's first \$1M in annual revenue.* (2021, March 16). The Verge. Retrieved January 11, 2023, from  
<https://www.theverge.com/2021/3/16/22333777/google-play-store-fee-reduction-developers-1-million-dollars>

Greer, J. (2022, July 3). *Pixel Art Class - The Importance of Character Scale.* YouTube.  
Retrieved January 11, 2023, from <https://youtu.be/Gwrpw1YtuGo>

*Grid Based Movement in Unity.* (2020, January 30). YouTube. Retrieved January 11, 2023, from <https://www.youtube.com/watch?v=mbzXIOKZurA>

*Grid Based Movement In Unity.* (2023, January 6). ? - YouTube. Retrieved January 11, 2023, from

<https://www.youtube.com/watch?v=mbzXlOKZurA&list=PLaWi-C2BWT5q9l7EVXQDCllODEgyFWl9n&index=1>

*Harvest Moon (serie de videojuegos).* (n.d.). Wikipedia. Retrieved January 11, 2023, from [https://es.wikipedia.org/wiki/Harvest\\_Moon\\_\(serie\\_de\\_videojuegos\)](https://es.wikipedia.org/wiki/Harvest_Moon_(serie_de_videojuegos))

Hejlsberg, A. (n.d.). *C Sharp (programming language).* Wikipedia. Retrieved January 11, 2023, from [https://en.wikipedia.org/wiki/C\\_Sharp\\_\(programming\\_language\)](https://en.wikipedia.org/wiki/C_Sharp_(programming_language))

*Herramientas esenciales para desarrolladores de software y equipos.* (n.d.). JetBrains. Retrieved February 9, 2023, from <https://www.jetbrains.com/es-es/>

*Home App Data Mobile Games Revenue Data (2023).* (2023, January 9). Business of Apps. Retrieved February 9, 2023, from <https://www.businessofapps.com/data/mobile-games-revenue/>

*How much does Steam charge companies to have their game in the store?* (n.d.). Quora. Retrieved January 11, 2023, from <https://www.quora.com/How-much-does-Steam-charge-companies-to-have-their-game-in-the-store>

*How to Add a 2D Follow Camera in Unity 2022.* (2022, June 17). YouTube. Retrieved January 11, 2023, from <https://youtu.be/eHUYlasHvG8>

*How to Build Open Games.* (2023, January 6). How to Build Open Games - YouTube. Retrieved January 11, 2023, from [https://www.youtube.com/watch?v=udF7XX\\_vTUE&list=WL&index=3](https://www.youtube.com/watch?v=udF7XX_vTUE&list=WL&index=3)

*How to make a Save & Load System in Unity | 2022.* (2022, March 9). YouTube. Retrieved January 11, 2023, from <https://youtu.be/aUi9aijvpgs>

*How to make a Save & Load System work across Multiple Scenes in Unity | 2022 tutorial.* (2022, April 18). YouTube. Retrieved January 11, 2023, from <https://youtu.be/ijVA5Z-Mbh8>

*How to make AWESOME Scene Transitions in Unity!* (2020, January 12). YouTube. Retrieved January 11, 2023, from <https://youtu.be/CE9VOZivb3I>

*How to make UI in UNITY - EASY TUTORIAL.* (2018, July 16). YouTube. Retrieved January 11, 2023, from [https://www.youtube.com/watch?v=\\_RIsfVOqTaE](https://www.youtube.com/watch?v=_RIsfVOqTaE)

*How to reuse Animation Clip for other characters in Unity.* (2021, November 30). YouTube. Retrieved January 11, 2023, from <https://www.youtube.com/watch?v=6mNak-mQZpc>

*How to show a Toast message in Unity similar to one in android?* (2018, October 1). Stack Overflow. Retrieved January 11, 2023, from <https://stackoverflow.com/questions/52590525/how-to-show-a-toast-message-in-unity-similar-to-one-in-android>

*How to store references to scene objects in prefabs?* (2017, January 3). Game Development Stack Exchange. Retrieved January 26, 2023, from <https://gamedev.stackexchange.com/questions/135209/how-to-store-references-to-scene-objects-in-prefabs>

*How to Switch Levels with Trigger Area for 2D RPGs in Unity.* (2021, December 29). YouTube. Retrieved January 11, 2023, from <https://www.youtube.com/watch?v=-7lOslJyi8g>

*HOW TO TELEPORT OBJECTS IN UNITY | Teleport Player and GameObjects in Unity |*

*Unity Tutorial.* (2021, December 7). YouTube. Retrieved January 26, 2023, from  
<https://youtu.be/xmhm5jGwonc>

*How to Unit Test Unity Code.* (2017, July 4). YouTube. Retrieved January 11, 2023, from  
<https://www.youtube.com/watch?v=TyxDg70hc3g>

*How to use and adjust 2D collider in Unity.* (2021, November 22). YouTube. Retrieved  
January 11, 2023, from <https://www.youtube.com/watch?v=eDOxDJEtE14>

*How to use Prefabs in Unity.* (2018, June 24). YouTube. Retrieved January 11, 2023, from  
<https://youtu.be/QmhzU8yH908>

*Integrated development environment.* (n.d.). Wikipedia. Retrieved February 9, 2023, from  
[https://en.wikipedia.org/wiki/Integrated\\_development\\_environment](https://en.wikipedia.org/wiki/Integrated_development_environment)

*Is it possible to serialize sprites in unity?* (2019, January 12). Stack Overflow. Retrieved  
January 26, 2023, from  
<https://stackoverflow.com/questions/54159017/is-it-possible-to-serialize-sprites-in-unity>

*Isometric pixel art by JessPXL on DeviantArt.* (2021, July 27). DeviantArt. Retrieved  
February 3, 2023, from  
<https://www.deviantart.com/jesspxl/art/Isometric-pixel-art-886987470>

*isometric pixel art by Me : r/PixelArt.* (2022, June 20). Reddit. Retrieved February 3, 2023,  
from  
[https://www.reddit.com/r/PixelArt/comments/vg6m2i/isometric\\_pixel\\_art\\_by\\_me/](https://www.reddit.com/r/PixelArt/comments/vg6m2i/isometric_pixel_art_by_me/)

- Isometric View - Pixel Art Style (+ drawing Room and Store) "Pixel Art Isometric #1"* by Cheishiru - Make better art. (2021, May 31). Clip Studio TIPS. Retrieved February 3, 2023, from <https://tips.clip-studio.com/en-us/articles/4969>
- JetBrains*. (n.d.). Wikipedia. Retrieved January 11, 2023, from <https://en.wikipedia.org/wiki/JetBrains>
- JSON.NET*. (n.d.). Json.NET - Newtonsoft. Retrieved January 12, 2023, from <https://www.newtonsoft.com/json>
- JSON.NET Error Self referencing loop detected for type*. (n.d.). Stack Overflow. Retrieved January 26, 2023, from <https://stackoverflow.com/questions/7397207/json-net-error-self-referencing-loop-detected-for-type>
- JSON .NET For Unity | Input Management*. (n.d.). Unity Asset Store. Retrieved January 26, 2023, from <https://assetstore.unity.com/packages/tools/input-management/json-net-for-unity-11347>
- JsonSerializerSettings Class*. (2012, March 21). Json.NET. Retrieved January 26, 2023, from [https://www.newtonsoft.com/json/help/html/T\\_Newtonsoft\\_Json\\_JsonSerializerSettings.htm](https://www.newtonsoft.com/json/help/html/T_Newtonsoft_Json_JsonSerializerSettings.htm)
- J-Spencer, M. -. (n.d.). *NetworkManager* example from <https://youtu.be/K9uVHI645Pk>. gists · GitHub. Retrieved January 11, 2023, from <https://gist.github.com/Matthew-J-Spencer/8da232ef3d3ce81a0f4a2cb087eb740b>
- Kenney. (n.d.). Kenney • Home. Retrieved January 11, 2023, from <https://kenney.nl/>



*League of Legends World Championship*. (n.d.). Wikipedia. Retrieved February 9, 2023, from

[https://en.wikipedia.org/wiki/League\\_of\\_Legends\\_World\\_Championship#Trophy](https://en.wikipedia.org/wiki/League_of_Legends_World_Championship#Trophy)

*Learn Unity and C#*. (2023, January 6). ? - YouTube. Retrieved January 11, 2023, from

[https://www.youtube.com/watch?v=b8YUfee\\_pzc&t=8s](https://www.youtube.com/watch?v=b8YUfee_pzc&t=8s)

*Lemons*. (2019, November 18). Free SVG. Retrieved February 3, 2023, from

<https://freesvg.org/lemons>

*Lerping in Unity - You HAVE to know this!!* (2021, September 9). YouTube. Retrieved

January 11, 2023, from <https://www.youtube.com/watch?v=JS7cNHivmHw>

*Make a Game Like Pokemon In Unity*. (2023, January 6). ? - YouTube. Retrieved January 11, 2023, from

[https://www.youtube.com/watch?v=\\_Pm16a18zy8&list=PLLf84Zj7U26kfPQ00JVI2nIoozuPkykDX](https://www.youtube.com/watch?v=_Pm16a18zy8&list=PLLf84Zj7U26kfPQ00JVI2nIoozuPkykDX)

*Make a Game Like Pokemon in Unity*. (2023, January 6). ? - YouTube. Retrieved January 11, 2023, from

[https://www.youtube.com/watch?v=\\_Pm16a18zy8&list=PLLf84Zj7U26kfPQ00JVI2nIoozuPkykDX&index=1](https://www.youtube.com/watch?v=_Pm16a18zy8&list=PLLf84Zj7U26kfPQ00JVI2nIoozuPkykDX&index=1)

*Making a 2D Level in Unity with the Tile Palette*. (2020, May 18). YouTube. Retrieved January 11, 2023, from <https://www.youtube.com/watch?v=-1VNG1jbOul>

*Making Games For Players Under 14*. (2023, January 6). ? - YouTube. Retrieved January 11, 2023, from <https://www.youtube.com/watch?v=NdFw8kvHAY8&list=WL&index=1>

*Making Save Games With The Odin Serializer*. (n.d.). Odin Inspector. Retrieved January 12, 2023, from

<https://odininspector.com/tutorials/serialize-anything/making-save-games-with-the-odin-serializer#odin-serializer>

*Making Your First Game: Basics - How To Start Your Game Development - Extra Credits.*

(2015, January 14). YouTube. Retrieved January 11, 2023, from

[https://www.youtube.com/watch?v=zO6QR-tz1\\_o](https://www.youtube.com/watch?v=zO6QR-tz1_o)

*Making Your First Game: Practical Rules - Setting (and Keeping) Goals - Extra Credits.*

(2015, January 21). YouTube. Retrieved January 11, 2023, from

<https://www.youtube.com/watch?v=dHMNeNapL1E>

*Manual: Writing and executing tests in Unity Test Runner.* (n.d.). Unity - Manual. Retrieved

January 11, 2023, from

<https://docs.unity3d.com/2018.3/Documentation/Manual/PlaymodeTestFramework.html>

Martin, R. C. (2000). *Design Principles and Design Patterns*. Design Principles and Design

Patterns. Retrieved February 3, 2023, from

[http://staff.cs.utu.fi/~jounsmmed/doos\\_06/material/DesignPrinciplesAndPatterns.pdf](http://staff.cs.utu.fi/~jounsmmed/doos_06/material/DesignPrinciplesAndPatterns.pdf)

Maurer, R. (n.d.). *Atari 2600*. Wikipedia. Retrieved January 11, 2023, from

[https://es.wikipedia.org/wiki/Atari\\_2600](https://es.wikipedia.org/wiki/Atari_2600)

Medinilla, Á., & Gómez, E. (n.d.). *Manifiesto por el Desarrollo Ágil de Software*. Manifiesto

for Agile Software Development. Retrieved February 13, 2023, from

<https://agilemanifesto.org/iso/es/manifiesto.html>

*Microsoft .NET*. (n.d.). Wikipedia. Retrieved February 9, 2023, from

[https://es.wikipedia.org/wiki/Microsoft\\_.NET](https://es.wikipedia.org/wiki/Microsoft_.NET)

Microsoft Visual Studio. (n.d.). Wikipedia. Retrieved February 9, 2023, from

[https://es.wikipedia.org/wiki/Microsoft\\_Visual\\_Studio](https://es.wikipedia.org/wiki/Microsoft_Visual_Studio)

Mono (software). (n.d.). Wikipedia. Retrieved February 10, 2023, from

[https://en.wikipedia.org/wiki/Mono\\_\(software\)](https://en.wikipedia.org/wiki/Mono_(software))

Muehsig, R. (2015, March 30). *JSON.NET deserialize to abstract class or interface*. Code Inside Blog. Retrieved January 26, 2023, from

<https://blog.codeinside.eu/2015/03/30/json-dotnet-deserialize-to-abstract-class-or-interface/>

*My Pixel Art Logo Design Process - PIXEL ART TUTORIAL (Aseprite / Libresprite)*. (2019, December 11). YouTube. Retrieved January 26, 2023, from

[https://youtu.be/QDEDkojq\\_SY](https://youtu.be/QDEDkojq_SY)

*Namespaces in Unity*. (2020, April 19). YouTube. Retrieved January 11, 2023, from

<https://www.youtube.com/watch?v=vLrGdx-Ob3g>

Nintendo. (n.d.). Wikipedia. Retrieved January 11, 2023, from

<https://es.wikipedia.org/wiki/Nintendo>

Observer pattern. (n.d.). Wikipedia. Retrieved January 11, 2023, from

[https://en.wikipedia.org/wiki/Observer\\_pattern](https://en.wikipedia.org/wiki/Observer_pattern)

Open Game Art. (2022, October 6). OpenGameArt.org. Retrieved January 11, 2023, from

<https://opengameart.org/>

Partis, D. (2021, May 19). *62% of all App Store revenue is generated from game*

*transactions*. GamesIndustry.biz. Retrieved February 9, 2023, from

<https://www.gamesindustry.biz/62-percent-of-all-app-store-revenue-is-generated-from-game-transactions>

*Patrón de diseño.* (n.d.). Wikipedia. Retrieved January 11, 2023, from

[https://es.wikipedia.org/wiki/Patr%C3%B3n\\_de\\_dise%C3%B1o](https://es.wikipedia.org/wiki/Patr%C3%B3n_de_dise%C3%B1o)

*PAUSE MENU in Unity.* (2017, December 20). YouTube. Retrieved January 11, 2023, from

<https://youtu.be/JivuXdrIHK0>

*Pixel art.* (n.d.). Wikipedia. Retrieved January 11, 2023, from

[https://es.wikipedia.org/wiki/Pixel\\_art](https://es.wikipedia.org/wiki/Pixel_art)

*Pixel Art Class - Art Styles for Indie Games.* (2021, October 10). YouTube. Retrieved

January 11, 2023, from <https://youtu.be/LfTiws-OVss>

*Pixel Art Class - Character Sprite Build!* (2020, March 18). YouTube. Retrieved January 11,

2023, from <https://www.youtube.com/watch?v=UPAHMyN9YeQ>

*Pixel Art Class - Character Sprite Details.* (2020, March 22). YouTube. Retrieved January 11,

2023, from <https://www.youtube.com/watch?v=6Vu5tylaw10>

*Pixel Art Class - Constructing a Character.* (2020, March 15). YouTube. Retrieved January

11, 2023, from <https://www.youtube.com/watch?v=lv4ZgH0bbSY>

*Pixel Art Class - Palettes & Colour.* (2020, June 14). YouTube. Retrieved January 11, 2023,

from <https://youtu.be/hkrK65FPmDI>

*Pixel Art Class - Tile Set Art.* (2020, May 19). YouTube. Retrieved January 11, 2023, from

<https://www.youtube.com/watch?v=Q-vZ6ZvvSys>

*Pixel Art Class - Tips & Scripts for Supercharging Your Aseprite Workflow.* (2021, December

5). YouTube. Retrieved January 11, 2023, from <https://youtu.be/hiMBVCFMj6E>

*Pixel Art Class - Top Down Style.* (2023, January 6). ? - YouTube. Retrieved January 11, 2023,

from

<https://www.youtube.com/watch?v=2JCG4fCmeHk&list=PLaWi-C2BWT5q9I7EVXQDCIIODEgyFWI9n&index=7>

*Pixel Art Class - What is Pixel Art?* (2020, March 8). YouTube. Retrieved January 11, 2023, from <https://www.youtube.com/watch?v=7BWr2tIK-4c>

*Pixel Art Class - What's The Right Canvas Size?* (2021, November 13). YouTube. Retrieved January 11, 2023, from <https://youtu.be/upEGBGCiWEw>

*PlayStation*. (n.d.). Wikipedia. Retrieved January 11, 2023, from <https://es.wikipedia.org/wiki/PlayStation>

*Pokémon (serie de videojuegos)*. (n.d.). Wikipedia. Retrieved February 3, 2023, from [https://es.wikipedia.org/wiki/Pok%C3%A9mon\\_\(serie\\_de\\_videojuegos\)](https://es.wikipedia.org/wiki/Pok%C3%A9mon_(serie_de_videojuegos))

*Pokemon Unity*. (2023, January 6). ? - YouTube. Retrieved January 11, 2023, from <https://www.youtube.com/watch?v=XozaGOtIGys&list=PLR0iv79IRiAGSOuwXMjm6xE7uVEdSILxG>

Poppendieck, T. D., Poppendieck, M., & Poppendieck, T. (2003). *Lean Software Development: An Agile Toolkit*. Addison-Wesley.

*Perforce*. (n.d.). Perforce Software | Development Tools For Innovation at Scale. Retrieved February 3, 2023, from <https://www.perforce.com/>

*Prize Pool for League of Legends Worlds: Check how much money will be distributed this year*. (n.d.). The Economic Times. Retrieved February 9, 2023, from <https://economictimes.indiatimes.com/news/international/us/prize-pool-for-league-of-legends-worlds-check-how-much-money-will-be-distributed-this-year/articleshow/94541119.cms>

*Programación orientada a objetos.* (n.d.). Wikipedia. Retrieved January 11, 2023, from

[https://es.wikipedia.org/wiki/Programaci%C3%B3n\\_orientada\\_a\\_objetos](https://es.wikipedia.org/wiki/Programaci%C3%B3n_orientada_a_objetos)

*Programación orientada a objetos.* (n.d.). IBM. Retrieved January 11, 2023, from

<https://www.ibm.com/docs/es/spss-modeler/saas?topic=language-object-oriented-programming>

*Publish Apps, Games and Software on the Epic Games Store.* (n.d.). Epic Games. Retrieved

January 11, 2023, from <https://store.epicgames.com/en-US/publish>

*Publish your game to itch.io.* (2022, July 18). GDevelop wiki. Retrieved January 28, 2023,

from <https://wiki.gdevelop.io/gdevelop5/publishing/publishing-to-itch-io>

*Question - Create SpriteLibraryAsset by Scripts.* (2021, September 11). Unity Forum.

Retrieved January 11, 2023, from

<https://forum.unity.com/threads/create-spritelibraryasset-by-scripts.1168610/>

*Question - Create SpriteLibraryAsset by Scripts.* (2021, September 11). Unity Forum.

Retrieved January 11, 2023, from

<https://forum.unity.com/threads/create-spritelibraryasset-by-scripts.1168610/>

*Raster graphics editor.* (n.d.). Wikipedia. Retrieved January 11, 2023, from

[https://en.wikipedia.org/wiki/Raster\\_graphics\\_editor](https://en.wikipedia.org/wiki/Raster_graphics_editor)

*Redalyc.Breve historia de los videojuegos.* (n.d.). Redalyc. Retrieved January 11, 2023, from

<https://www.redalyc.org/pdf/537/53701409.pdf>

*ReferenceLoopHandling Enumeration.* (n.d.). Json.NET. Retrieved January 26, 2023, from

[https://www.newtonsoft.com/json/help/html/T\\_Newtonsoft\\_Json\\_ReferenceLoopHandling.htm](https://www.newtonsoft.com/json/help/html/T_Newtonsoft_Json_ReferenceLoopHandling.htm)

- Refresh Unity Assets.* (2022, August 25). JetBrains. Retrieved January 11, 2023, from [https://www.jetbrains.com/help/rider/Refreshing\\_Unity\\_Assets.html#how-unity-editor-reacts-on-asset](https://www.jetbrains.com/help/rider/Refreshing_Unity_Assets.html#how-unity-editor-reacts-on-asset)
- Rerych, M., & Wright, S. (n.d.). *fit 2002*. Retrieved February 13, 2023, from <http://cartoon.iguw.tuwien.ac.at/fit/fit01/wasserfall/entstehung.html>
- Rider: The Cross-Platform .NET IDE from JetBrains.* (n.d.). JetBrains. Retrieved January 11, 2023, from <https://www.jetbrains.com/rider/>
- The right way to pause a game in Unity.* (2022, July 29). YouTube. Retrieved January 11, 2023, from <https://youtu.be/ROwsdfEGFO>
- Riot Games. (n.d.). Riot Games: Home. Retrieved February 9, 2023, from <https://www.riotgames.com/es>
- RPGTalk. (n.d.). Seize Studios. Retrieved January 11, 2023, from <http://www.seizestudios.com/developer/rpgtalk/>
- RPGTalk | GUI Tools. (n.d.). Unity Asset Store. Retrieved January 11, 2023, from <https://assetstore.unity.com/packages/tools/gui/rpgtalk-73392#content>
- Sakai, M. (n.d.). Wacom. home. Retrieved February 3, 2023, from <https://www.wacom.com/es-es>
- Save & Load System in Unity - Bug Fixes, Scriptable Objects, Deleting Data, Backup Files, and More.* (2022, June 20). YouTube. Retrieved January 11, 2023, from <https://www.youtube.com/watch?v=yTWPCimAdvY>
- Save SCRIPTABLE OBJECTS - Unity Tutorial.* (2021, March 6). YouTube. Retrieved January 26, 2023, from <https://youtu.be/Bv-Qie4ISWg>

*Scene helper for Unity additive loading.* · GitHub. (n.d.). gists · GitHub. Retrieved January 11, 2023, from

<https://gist.github.com/kurtdekker/862da3bc22ee13aff61a7606ece6fdd3>

Schlitter, R. (2021, December 2). *Pixelblog - 35 - Top Down Interiors* – SLYNYRD.

SLYNYRD. Retrieved February 3, 2023, from

<https://www.slynyrd.com/blog/2021/11/30/pixelblog-35-top-down-interiors>

*Scriptable Object Inventory System Part 1.* (2023, January 6). ? - YouTube. Retrieved

January 11, 2023, from

[https://youtu.be/\\_lqTeruf3-s?list=PLaWi-C2BWT5q9l7EVXQDCIIODEgyFWl9n](https://youtu.be/_lqTeruf3-s?list=PLaWi-C2BWT5q9l7EVXQDCIIODEgyFWl9n)

*Scripting API: Networking.UnityWebRequest.Get.* (n.d.). Unity - Manual. Retrieved January

11, 2023, from

<https://docs.unity3d.com/ScriptReference/Networking.UnityWebRequest.Get.html>

*Scripting API: SceneManagement.SceneManager.GetActiveScene.* (n.d.). Unity - Manual.

Retrieved January 26, 2023, from

<https://docs.unity3d.com/ScriptReference/SceneManagement.SceneManager.GetActiveScene.html>

*Sending Web Requests in Unity - UnityWebRequest.* (2021, February 8). YouTube. Retrieved

January 11, 2023, from <https://www.youtube.com/watch?v=K9uVHI645Pk>

*Serialization Settings.* (2012, March 21). Json.NET. Retrieved January 26, 2023, from

<https://www.newtonsoft.com/json/help/html/SerializationSettings.htm#ReferenceLoopHandling>

*Singleton.* (n.d.). Wikipedia. Retrieved February 3, 2023, from

<https://es.wikipedia.org/wiki/Singleton>



*Sitio oficial de PlayStation®: Consolas, juegos, accesorios y más.* (n.d.). PlayStation.

Retrieved January 11, 2023, from <https://www.playstation.com/es-ar/>

*SOLID.* (n.d.). Wikipedia. Retrieved January 11, 2023, from

<https://es.wikipedia.org/wiki/SOLID>

*Space Invaders.* (n.d.). Wikipedia. Retrieved February 3, 2023, from

[https://es.wikipedia.org/wiki/Space\\_Invaders](https://es.wikipedia.org/wiki/Space_Invaders)

*Stardew Valley.* (n.d.). Wikipedia. Retrieved January 11, 2023, from

[https://es.wikipedia.org/wiki/Stardew\\_Valley](https://es.wikipedia.org/wiki/Stardew_Valley)

*Stardew Valley Like Episode.* (2023, January 6). ? - YouTube. Retrieved January 11, 2023,

from

<https://www.youtube.com/watch?v=ZIEE-2ZdAxU&list=PLOGUZtUkX6t6wXFOUOWAQNVYL68pYUCZv&index=1&t=68s>

*Stardew Valley Like Game in Unity.* (2023, January 6). ? - YouTube. Retrieved January 11,

2023, from

<https://youtu.be/dcGm81ILbww?list=PLOGUZtUkX6t6wXFOUOWAQNVYL68pYUCZv>

*Stardew Valley Machinations Diagram Tutorial Part 1 - Livestock.* (2020, June 29).

YouTube. Retrieved January 11, 2023, from

<https://www.youtube.com/watch?v=8N8VAfxJdhs>

*State pattern.* (n.d.). Wikipedia. Retrieved February 3, 2023, from

[https://en.wikipedia.org/wiki/State\\_pattern](https://en.wikipedia.org/wiki/State_pattern)

*Strategy pattern.* (n.d.). Wikipedia. Retrieved January 26, 2023, from

[https://en.wikipedia.org/wiki/Strategy\\_pattern](https://en.wikipedia.org/wiki/Strategy_pattern)

*Sunnyside World* by danieldigggle. (n.d.). danieldigggle. Retrieved January 11, 2023, from

<https://danieldigggle.itch.io/sunnyside>

*Super Nintendo*. (n.d.). Wikipedia. Retrieved January 11, 2023, from

[https://es.wikipedia.org/wiki/Super\\_Nintendo](https://es.wikipedia.org/wiki/Super_Nintendo)

Sutherland, J. (2015). *Scrum: el nuevo y revolucionario modelo organizativo que cambiará tu vida*. Planeta.

Tajiri, S. (n.d.). *Pokémon*. Wikipedia. Retrieved January 11, 2023, from

<https://es.wikipedia.org/wiki/Pok%C3%A9mon>

*TDD and Unit Testing in Unity*. (2023, January 6). ? - YouTube. Retrieved January 11, 2023, from

<https://www.youtube.com/watch?v=27h3l32S3s8&list=PLaWi-C2BWT5q9l7EVXQDCI10DEgyFWl9n&index=11>

*Test runner*. (n.d.). GameCI. Retrieved February 9, 2023, from

<https://game.ci/docs/github/test-runner>

*Texture colors are a little bit off*. (2012, January 26). Unity Answers. Retrieved January 26, 2023, from

<https://answers.unity.com/questions/210240/texture-colors-are-a-little-bit-off.html>

*Tiny RPG - Forest | 2D Characters*. (2018, April 4). Unity Asset Store. Retrieved January 11, 2023, from

<https://assetstore.unity.com/packages/2d/characters/tiny-rpg-forest-114685>

*Top 5 Tools for Unity Programmers*. (2023, January 6). ? - YouTube. Retrieved January 11, 2023, from

<https://www.youtube.com/watch?v=a38pRScGk8I&list=PLaWi-C2BWT5q9I7EVXQDCII0DEgyFWI9n&index=4>

Torii. (n.d.). Wikipedia. Retrieved February 3, 2023, from <https://es.wikipedia.org/wiki/Torii>

Tramiel, J. (n.d.). Atari. Wikipedia. Retrieved February 3, 2023, from <https://es.wikipedia.org/wiki/Atari>

Tubb, J. (n.d.). Pong. Wikipedia. Retrieved January 11, 2023, from <https://es.wikipedia.org/wiki/Pong>

*2D Camera in Unity (Cinemachine Tutorial)*. (2018, September 9). YouTube. Retrieved January 11, 2023, from <https://youtu.be/2jTY11AmOlq>

*2D Camera in Unity + Confining to bounds [Cinemachine]*. (2021, March 30). YouTube. Retrieved January 11, 2023, from <https://youtu.be/xxshfe3yzqA>

*2D Character Customization - Full Skin Changing - Unity Tutorial*. (2022, July 9). YouTube. Retrieved January 11, 2023, from [https://www.youtube.com/watch?v=ZgCB4tifQ\\_c](https://www.youtube.com/watch?v=ZgCB4tifQ_c)

*2D CHARACTER CUSTOMIZATION in Unity Tutorial*. (2020, April 6). YouTube. Retrieved January 11, 2023, from <https://www.youtube.com/watch?v=kAPIWJJ6NQI>

*2D Simple UI Pack | 2D Icons*. (2022, April 15). Unity Asset Store. Retrieved January 11, 2023, from <https://assetstore.unity.com/packages/2d/gui/icons/2d-simple-ui-pack-218050>

*Type is an interface or abstract class and cannot be instantiated*. (n.d.). Stack Overflow. Retrieved January 26, 2023, from <https://stackoverflow.com/questions/24644464/type-is-an-interface-or-abstract-class-and-cannot-be-instantiated>

*Uniendo al mundo con Pokémon.* (n.d.). The Pokémon Company International. Retrieved

January 11, 2023, from <https://corporate.pokemon.com/es-es/>

*Unit testing in Unity and C# using Unity Test Runner | Tutorial.* (2018, April 18). YouTube.

Retrieved January 11, 2023, from <https://www.youtube.com/watch?v=vLvA7ZkFGRo>

*Unit Tests in Unity.* (2020, June 7). YouTube. Retrieved January 11, 2023, from

<https://www.youtube.com/watch?v=PDYB32qAsLU>

*Unity.* (n.d.). Unity Real-Time Development Platform | 3D, 2D VR & AR Engine. Retrieved

January 11, 2023, from <https://unity.com/>

*unity3d - How do I serialize ScriptableObjects?* (2018, September 18). Stack Overflow.

Retrieved January 26, 2023, from

<https://stackoverflow.com/questions/52377939/how-do-i-serialize-scriptableobjects>

*Unity Assembly Definition Files and Rider 2018.2 | The .NET Tools Blog.* (2018, September

25). The JetBrains Blog. Retrieved January 11, 2023, from

<https://blog.jetbrains.com/dotnet/2018/09/26/unity-assembly-definition-files-rider-2018-2/>

*Unity Basics - Move towards and follow target.* (2021, July 11). YouTube. Retrieved January

11, 2023, from <https://www.youtube.com/watch?v=wp8m6xyIPtE>

*Unity Coroutines.* (n.d.). Javatpoint. Retrieved January 11, 2023, from

<https://www.javatpoint.com/unity-coroutines>

*Unity in 100 Seconds.* (2022, March 7). YouTube. Retrieved January 11, 2023, from

<https://www.youtube.com/watch?v=iqlH4okiQqg>

*Unity (motor de videojuego).* (n.d.). Wikipedia. Retrieved January 11, 2023, from

[https://es.wikipedia.org/wiki/Unity\\_\(motor\\_de\\_videojuego\)](https://es.wikipedia.org/wiki/Unity_(motor_de_videojuego))

- Unity para retrasados.* (2018, July 25). YouTube. Retrieved January 11, 2023, from <https://www.youtube.com/watch?v=IT7vDqm4xiY>
- Unity para retrasados 2.* (2018, August 1). YouTube. Retrieved January 11, 2023, from <https://www.youtube.com/watch?v=DctCSiWHUlo>
- Unity Scenes 2: Keeping Objects When Changing Scenes (DontDestroyOnLoad).* (2021, October 13). YouTube. Retrieved January 11, 2023, from <https://youtu.be/vGxNLVXYlto>
- Unity Scenes 3: Loading Scenes Additively.* (2021, October 13). YouTube. Retrieved January 11, 2023, from [https://youtu.be/l\\_Y5SmWkOm4](https://youtu.be/l_Y5SmWkOm4)
- Unity - Scripting API: SceneManagement.Scene.buildIndex.* (n.d.). Unity - Manual. Retrieved January 26, 2023, from <https://docs.unity3d.com/ScriptReference/SceneManagement.Scene-buildIndex.html>
- Unity - Scripting API: Vector3.MoveTowards.* (n.d.). Unity - Manual. Retrieved January 11, 2023, from <https://docs.unity3d.com/ScriptReference/Vector3.MoveTowards.html>
- Unity - Test runner · Actions · GitHub Marketplace · GitHub.* (n.d.). GitHub. Retrieved February 9, 2023, from <https://github.com/marketplace/actions/unity-test-runner>
- Unity tutorial: how to fade between scenes.* (n.d.). Gamedevelopertips. Retrieved January 11, 2023, from <https://gamedevelopertips.com/unity-how-fade-between-scenes/>
- Unity UI Tutorial - Create a Health Bar in 90 Seconds.* (2019, September 24). YouTube. Retrieved January 26, 2023, from [https://www.youtube.com/watch?v=mi\\_SPOsippi](https://www.youtube.com/watch?v=mi_SPOsippi)
- Usage limits, billing, and administration.* (n.d.). GitHub Docs. Retrieved February 9, 2023, from

<https://docs.github.com/en/actions/learn-github-actions/usage-limits-billing-and-administration>

*Video Games - South Korea.* (n.d.). Statista. Retrieved February 9, 2023, from

<https://www.statista.com/outlook/dmo/digital-media/video-games/south-korea>

*Videojuego de rol.* (n.d.). Wikipedia. Retrieved January 11, 2023, from

[https://es.wikipedia.org/wiki/Videojuego\\_de\\_rol](https://es.wikipedia.org/wiki/Videojuego_de_rol)

*What Canvas Size Should you use for Pixel Art? (Pixel Art Tutorial).* (2020, November 29).

YouTube. Retrieved January 11, 2023, from <https://youtu.be/Z8earctNBxg>

*White-Faced Heron Egretta - Free photo on Pixabay.* (2022, September 21). Pixabay.

Retrieved February 3, 2023, from

<https://pixabay.com/photos/white-faced-heron-heron-7469267/>