UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E INDUSTRIAL

CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES E INFORMÁTICOS

TEMA:

DESARROLLO DE UN VIDEOJUEGO 2D PARA LA ENSEÑANZA DE LA CULTURA Y MITOLOGÍA INCA

Trabajo de Graduación Modalidad: Proyecto de Investigación, presentado previo la obtención del título de Ingeniero en Sistemas Computacionales e Informáticos

SUBLÍNEA DE INVESTIGACIÓN:

Orientación a objetos

AUTOR: Carlos José Acuña Aguaguiña
TUTOR: Ing. Franklin Mayorga, Mg

Ambato - Ecuador
Agosto, 2020

APROBACIÓN DEL TUTOR

En calidad de Tutor del Trabajo de Titulación sobre el tema: "DESARROLLO DE UN VIDEOJUEGO 2D PARA LA ENSEÑANZA DE LA CULTURA Y MITOLOGÍA INCA", desarrollado bajo la modalidad de Proyecto de Investigación del señor Carlos José Acuña Aguaguiña, estudiante de la Carrera de Ingeniería en Sistemas Computacionales e Informáticos, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial de la Universidad Técnica de Ambato, me permito indicar que el estudiante ha sido tutorado durante todo el desarrollo del trabajo hasta su conclusión, de acuerdo a lo dispuesto en el Artículo 15 del Reglamento para obtener el Título de Tercer Nivel, de Grado de la Universidad Técnica de Ambato, y el numeral 7.4 del respectivo instructivo.

Ambato, agosto de 2020

Ing. Franklin Mayorga, Mg
EL TUTOR

AUTORÍA

El presente trabajo de investigación titulado: "DESARROLLO DE UN VIDEOJUEGO 2D PARA LA ENSEÑANZA DE LA CULTURA Y MITOLOGÍA INCA" es absolutamente original, auténtico y personal, en tal virtud, el contenido, efectos legales y académicos que se desprenden del mismo son de exclusiva responsabilidad del autor.

Ambato, agosto de 2020

Carlos José Acuña Aguaguiña
CC: 1804413373
AUTOR

DERECHOS DE AUTOR

Autorizo a la Universidad Técnica de Ambato, para que haga uso de este Trabajo de Titulación como un documento disponible para la lectura, consulta y procesos de investigación.
Cedo los derechos de mi Trabajo de Titulación en favor de la Universidad Técnica de Ambato, con fines de difusión pública. Además, autorizo su reproducción total o parcial dentro de las regulaciones de la institución.

Ambato, agosto de 2020

Carlos José Acuña Aguaguiña
CC: 1804413373
AUTOR

APROBACIÓN TRIBUNAL DE GRADO

En calidad de par calificador del Informe Final del Trabajo de Titulación presentado por el señor Carlos José Acuña Aguaguiña, estudiante de la Carrera de Ingeniería en Sistemas Computacionales e Informáticos, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, bajo la Modalidad Proyecto de Investigación, titulado "DESARROLLO DE UN VIDEOJUEGO 2D PARA LA ENSEÑANZA DE LA CULTURA Y MITOLOGÍA INCA", nos permitimos informar que el trabajo ha sido revisado y calificado de acuerdo al Artículo 17 del Reglamento para obtener el Título de Tercer Nivel, de Grado de la Universidad Técnica de Ambato, y al numeral 7.6 del respectivo instructivo. Para cuya constancia suscribimos, conjuntamente con la señora Presidenta del Tribunal.

Ambato, agosto 2020.

Firmado electrónicamente por:
ELSA PILAR URRUTIA

Ing. Pilar Urrutia, Mg.
PRESIDENTA DEL TRIBUNAL

Firmado electrónicamente por:
HERNAN FABRICIO NARANJO AVALOS

Ing. Hernán Naranjo, Mg
PROFESOR CALIFICADOR

Firmado electrónicamente por:
EDWIN HERNANDO BUENANO VALENCIA

Ing. Hernando Buenaño, Mg.
PROFESOR CALIFICADOR

Dedicatoria

Dedicado a mi familia por todo su apoyo y consideración.

A la Facultad de Ingeniería en Sistemas
Electrónica e Industrial.

Gracias totales.

Carlos José Acuña Aguaguiña

Agradecimiento

Agradezco a mi familia por apoyarme en todos los momentos y darme la oportunidad de prepararme y ser una mejor persona en todos los aspectos de la vida.

Carlos José Acuña Aguaguiña

Índice

APROBACIÓN DEL TUTOR ii

AUTORÍA iii

DERECHOS DE AUTOR iv

Aprobación del tribunal de Grado v

Dedicatoria vi

Agradecimiento vii

Resumen Ejecutivo xiv

Abstract xv

Introducción xvi

1 Marco Teórico 1
- 1.1 Tema de investigación . . . 1
- 1.2 Antecedentes investigativos . . . 1
  - 1.2.1 Contextualización del problema . . . 1
  - 1.2.2 Delimitación . . . 2
  - 1.2.3 Justificación . . . 2
- 1.3 Fundamentación teórica . . . 2
- 1.4 Objetivos . . . 4
  - 1.4.1 General . . . 4
  - 1.4.2 Específicos . . . 4

2 Metodología 5
- 2.1 Métodos . . . 5
  - 2.1.1 Modalidad de la investigación . . . 5
  - 2.1.2 Población y muestra . . . 5
  - 2.1.3 Procesamiento y análisis de datos . . . 5

3 Resultados y Discusión 18
- 3.1 Análisis y discusión de los resultados . . . 18
  - 3.1.1 Estudio de Frameworks para el desarrollo del proyecto . . . 18
  - 3.1.2 Comparación de diferentes motores de juegos para la creación de la aplicación . . . 19

3.1.3 Componentes de la interfaz de Unity 20
3.1.4 Selección de metodologías ágiles para el desarrollo de videojuegos 27
3.1.5 Comparación de metodologías ágiles para el desarrollo de videojuegos 36
3.2 Implementación del videojuego educativo en 2D enfocado al aprendizaje de la cultura inca ecuatoriana 37
3.2.1 Fase Pre-juego 37
3.2.2 Fase de juego 48
3.2.3 Fase Post-Juego 81

4 Conclusiones y recomendaciones **89**
4.1 Conclusiones 89
4.2 Recomendaciones 90

Bibliografía **91**

A Tarjetas de tareas **94**

Índice de figuras

2.1 Imperio Inca 6
2.2 Huaynacápac 7
2.3 Atahualpa 8
2.4 Huáscar 9
2.5 Quisquis 10
2.6 Calicuchima 11
2.7 Chapera 12
2.8 Apu Atoc 12
2.9 Batalla de Molleambato 13
2.10 Batalla de Quipaipán 14
2.11 Amaru 15
2.12 Apallimay 16
2.13 Jarjacha 16

3.1 Editor de Unity 21
3.2 Project Window de Unity 22
3.3 Scene View de Unity 22
3.4 Hierarchy Window de Unity 23
3.5 Inspector Window de Unity 23
3.6 Toolbar de Unity 24
3.7 GameObject de Unity 24
3.8 Prefabs de Unity 25
3.9 Input de Unity 25
3.10 Virtual Axes de Unity 26
3.11 Sprites del personaje principal 39
3.12 Sprites NPC (enemigo) 39
3.13 Sprites NPC (bestia mitológica inca) 39
3.14 Sprites NPC (Jefe de nivel) 39
3.15 UI Menú Principal 40
3.16 UI Cinemática 41
3.17 Concepto del juego 44
3.18 Terreno creado con Tilemap 50
3.19 Sprites del personaje principal 51
3.20 Uso de la herramienta Animation para crear animaciones del personaje principal 52
3.21 Uso de la herramienta Animator para ordenar las animaciones del personaje principal 52
3.22 Método “Run” 53
3.23 Método “FlipSprite” 53

3.24 Uso de las opciones de proyecto para modificar la gravedad de los objetos del juego 54
3.25 Método “Jump” 54
3.26 Método “ClimbLadder” 55
3.27 Método “CanShoot” 56
3.28 Método “Shoot” 56
3.29 Ventana de jerarquía donde se encuentran las cámaras virtuales . 57
3.30 Ventana del Inspector de la cámara Idle 57
3.31 Ventana del Inspector de la cámara Run 58
3.32 Previsualización de los fondos de pantalla en pixel art 59
3.33 Resultado de la iteración 1 60
3.34 Método “AddLives” 62
3.35 Método “AddCoins” 62
3.36 Método “Update” para aumentar el número de vidas por conteo de monedas de oro 62
3.37 Método “Start’ con los componentes “PlayerPrefs” para guardar información 63
3.38 Imagen del juego con los datos de vida y monedas de oro 64
3.39 Método “OnTriggerEnter2D” de la clase “ExtraLife.cs” 65
3.40 Método “OnTriggerEnter2D” de la clase “Coin.cs” 65
3.41 Método “OnTriggerEnter2D” de la clase “Player.cs” 66
3.42 Método “OnTriggerEnter2D” de la clase “Checkpoint.cs” 66
3.43 Método “ProcessPlayerDeath” de la clase “LevelManager.cs” . . . 67
3.44 Método “ProcessPlayerRestart” de la clase “LevelManager.cs” . . 67
3.45 Método “Respawn” de la clase “LevelManager.cs” 67
3.46 NPC (Guerrero) 68
3.47 NPC (Apallimay) 68
3.48 NPC (Amaru) 68
3.49 NPC (Jarjacha) 68
3.50 Uso de la herramienta Animation para el NPC (Guerrero) 69
3.51 Uso de la herramienta Animator para el NPC (Guerrero) 69
3.52 Método “IsFacingRight” 69
3.53 Método “Update” para el movimiento del enemigo 70
3.54 Método “TakeDamage” 71
3.55 Método “Die” 71
3.56 Método “OnTriggerEnter2D” de la clase ”Projectile.cs” 72
3.57 Método “Type” 73
3.58 Método “NextSentence” 73
3.59 Método “NextLevel” 73
3.60 Cinemática de la historia dentro del videojuego 74
3.61 Método “LoadNextLevel” 76
3.62 Método “OnTriggerEnter2D” de la clase “LevelExit.cs” 76
3.63 Chapera (Sprites) 77
3.64 Apu Atoc (Sprites) 77
3.65 Huascar (Sprites) 77
3.66 Pantalla de menú principal 78
3.67 Pantalla de controles 78

3.68 Pantalla de créditos 79
3.69 Pantalla de historia 79
3.70 Capturas del juego “Huaraca” 80
3.71 Sección de añadir juego de Gamejolt 82
3.72 Sección de detalle de Gamejolt 83
3.73 Sección de descripción de Gamejolt 84
3.74 Sección de diseño de Gamejolt 84
3.75 Sección de paquetes de Gamejolt 85
3.76 Sección de madurez de Gamejolt 86
3.77 Sección de preferencias de Gamejolt 86
3.78 Reporte general de Gamejolt 87
3.79 Vistas por país del videojuego 87
3.80 Descargas por país del videojuego 88
3.81 Descargas del videojuego por sistema operativo 88

Índice de cuadros

2.1 Personajes del videojuego 17

3.1 Cuadro comparativo de motores de juegos 20
3.2 Historias de usuario de DAV 33
3.3 Prueba de aceptación de DAV 34
3.4 Reserva de producto de DAV 35
3.5 Tarjeta de tarea de DAV 36
3.6 Cuadro comparativo de métodos ágiles 37
3.7 Características de los elementos del videojuego 42
3.8 Requisitos técnicos 43
3.9 Roles 45
3.10 Historias de usuario del juego 46
3.11 Prueba de aceptación PA01 47
3.12 Historias de usuario del juego 48
3.13 Fecha de entrega de cada iteración 48
3.14 Historias de usuario a implementarse en la Iteración 1 49
3.15 Historias de usuario a implementarse en la Iteración 2 61
3.16 Historias de usuario a implementarse en la Iteración 3 75
3.17 Cuadro comparativo de tiendas online 82

Resumen Ejecutivo

En el mundo han existido una gran número de civilizaciones que han moldeado la sociedad, entre ellas se encuentra la cultura Inca, la cual ha heredado costumbres y tradiciones que se han transmitido de generación en generación a través de mitos y leyendas.

Gracias a los videojuegos se puede crear verdaderas obras maestras que no solo entretengan al espectador sino también le den un mensaje, una enseñanza que le sirva para conocer mejor un tema, en este caso una civilización que existió hace mucho tiempo y que tuvo un gran impacto en el desarrollo de la sociedad latinoamericana y ecuatoriana.

A nivel nacional el desarrollo de videojuegos es un campo no muy explorado y por lo tanto no se toma enserio el potencial educativo que tienen, han existido varios intentos de desarrollo pero solo han sido relegados a pequeños proyectos universitarios y trabajos independientes.

El presente videojuego fue desarrollo en Unity3D con el lenguaje de programación C#, y para el diseño se utilizó el programa GIMP.

Se utilizó la metodología ágil conocida como DAV propuesta por por Gabriel Armendáriz estudiante de la facultad de Informática de la EPOCH (Escuela Superior Politécnica de Chimborazo).

Los resultados obtenidos después de la publicación del videojuego en la plataforma Gamejolt fueron favorables debido al número de vistas y descargas realizadas por los usuarios pese a ser un juego independiente con una temática educativa de fondo.

Palabras clave: videojuego, historia, Imperio inca, framework, unity2D

Abstract

In the world there have been a large number of civilizations that have shaped society, among them is the Inca culture, which has inherited customs and traditions that have been transmitted from generation to generation through myths and legends.

Thanks to video games you can create true masterpieces that not only entertain the viewer but also give them a message, a teaching that will help them better understand a subject, in this case a civilization that existed a long time ago and that had a great impact in the development of Latin American and Ecuadorian society.

At the national level the development of video games is a not very explored field and therefore the educational potential they have is not taken seriously, there have been several attempts at development but they have only been relegated to small university projects and independent work.

This video game was developed in Unity3D with the C# programming language, and the GIMP program was used for the design.

The agile methodology known as DAV proposed by Gabriel Armendáriz, student of the Computer Science faculty of EPOCH (Escuela Superior Politécnica de Chimborazo) was used.

The results obtained after the publication of the video game on the Gamejolt platform were favorable due to the number of views and downloads made by users despite being an independent game with a background educational theme.

Keywords: videogame, history, Inca Empire, framework, unity2D

INTRODUCCIÓN

El presente proyecto de investigación denominado "DESARROLLO DE UN VIDEOJUEGO 2D PARA LA ENSEÑANZA DE LA CULTURA Y MITOLOGÍA INCA" se encuentra divido en los siguientes capítulos.

CAPÍTULO I. "MARCO TEÓRICO", plantea un problema el cual se ha de investigar y se busca una justificación para realizar esta investigación además de definir objetivos que guiarán el desarrollo de este proyecto.

CAPÍTULO II. "METODOLOGÍA", reúne todas las técnicas e instrumentos necesarios para obtener, procesar y analizar los datos obtenidos, además, define las etapas para el desarrollo del presente proyecto.

CAPÍTULO IV. "RESULTADOS Y DISCUSIÓN", se indica de manera clara y concisa el desarrollo del proyecto, así como la comparación de las mejores metodologías y frameworks que permitan el desarrollo del videojuego.

CAPÍTULO V. "CONCLUSIONES Y RECOMENDACIONES", en este apartado se señalan algunas conclusiones y recomendaciones que se han encontrado a lo largo del desarrollo del proyecto.

Capítulo 1

Marco Teórico

1.1 Tema de investigación

Desarrollo de un videojuego 2D para la enseñanza de la cultura y mitología inca.

1.2 Antecedentes investigativos

1.2.1 Contextualización del problema

En el mundo han existido un gran número de civilizaciones que han moldeado la sociedad y han heredado sus conocimientos, hallazgos, costumbres y tradiciones para el futuro de la humanidad. Gracias a la compleja conectividad de la globalización con ayuda de las tecnologías de información y comunicación es posible conocer cada rincón del mundo y por lo tanto descubrir muchas de estas culturas que han sido inmortalizadas en el consciente colectivo de la gente mediante obras literarias, películas, series de televisión e incluso videojuegos [1]. Es así como ahora se conoce civilizaciones tan antiguas como la griega y su mitología en torno a dioses como Zeus, Poseidón, Hades entre otros [2], o la cultura vikinga y su mitología nórdica con su propio panteón de ases como Odín, Thor y Loki [3]. Esto ha permitido que los descendientes de estas civilizaciones se sientan orgullosos de sus ancestros y tengan bien definido sus raíces.

Mucho de esta contribución para conocer la historia antigua ha sido en gran medida gracias a la industria de los videojuegos donde se puede destacar varios títulos muy importantes como son GOD OF WAR, un juego que tiene mecánicas de peleas con varios oponentes que se relacionan con la mitología griega y los principales dioses y monstruos de este panteón, además, este juego toma lugar en lugares reales del planeta y donde el jugador puede sumergirse más profundamente en esta cultura [4]. Otro título muy conocido es el juego AGE OF MITOLOGY donde el principal objetivo es ganar a las demás naciones militarmente a través de la evolución de su tecnología, desarrollo de la economía y pidiendo el favor de los dioses. El juego se basa en culturas ricas en cultura y mitología y donde las leyendas de héroes y dioses son contadas de manera ingeniosa y donde el jugador se puede sentir parte de la historia [5].

Sin embargo, esa misma globalización, encabezada por la masiva explosión cultural de Estados Unidos alrededor del mundo está causando que otros países pierdan sus costumbres y tradiciones y se olviden de sus raíces culturales[6]. Además, existen naciones que tratan de rescatar su cultura mediante varios proyectos gubernamentales, así tenemos al gobierno francés que subsidia la producción nacional cinematográfica y obliga a los teatros y cines a estrenar películas de origen

francés cada cierto tiempo [7].
Otra nación que protege su cultura a pesar de la influencia anglosajona es México, donde el estado se preocupa de mantener sus costumbres y tradiciones, a salvo, y hay casos donde su propia gente decide hacer algo al respecto, así tenemos a un joven ilustrador, Pedro Ramírez Lara, que realiza ilustraciones donde los protagonistas son dioses de la época prehispánica de México con el fin de dar a conocer las leyendas, cuentos y mitología de su cultura [8].
Así mismo, un grupo de jóvenes peruanos han creado un cómic sobre las aventuras de los hermanos Ayar, que son parte de las leyendas de mitología inca y una muestra de que la mitología Sudamericana es rica en historias y leyendas [9].
En el Ecuador este tipo de iniciativas no se ha visto apoyada o mencionada en ninguna parte por lo tanto la gente olvida su identidad cultural y prefiere adoptar costumbres extranjeras.

1.2.2 Delimitación

Área Académica: Software
Línea de investigación: Desarrollo de software
Sublíneas de investigación: Orientación a objetos
Delimitación especial: Este proyecto no está delimitada a una área geográfica, ya que busca ser publicada en una tienda virtual, por lo tanto su dimensión es a nivel global
Delimitación temporal: La presente investigación se desarrollará durante los 6 meses posteriores a la aprobación del proyecto por parte del Consejo Directivo de la Facultad que constituye desde el mes de diciembre de 2019 hasta - junio de 2020.

1.2.3 Justificación

La importancia de esta investigación se fundamenta en elevar el interés de las personas por la historia nacional y conocer las raíces de todos los ecuatorianos mediante el uso de herramientas tecnológicas, que en este caso son conocidos como videojuegos, ya que busca rescatar las costumbres y tradiciones de los pueblos autóctonos del país, centrándose en la región andina; siendo un proyecto original ya que busca crear una experiencia diferente para los usuarios. Aquí el beneficiario final son las futuras generaciones del país, especialmente los estudiantes y personas que quieran conocer sobre la maravillosa historia del Ecuador.

1.3 Fundamentación teórica

Como parte inicial se debe conocer el concepto de videojuego, que según el diccionario de la Real Academia Española (RAE) lo define como "Juego electrónico que se visualiza en una pantalla, o un dispositivo electrónico que permite, mediante mandos apropiados, simular juegos en las pantallas de un televisor, una computadora u otro dispositivo electrónico" [10]. Sin embargo, se puede encontrar una definición más profunda a cargo de Ana Laura Rossaro en su artículo "Aprender jugando: Los videojuegos y su potencial educativo" quien dice que "... en general los videojuegos recrean entornos y situaciones en los que el jugador (o

varios jugadores) pueden controlar personajes e interactuar con el entorno para alcanzar un objetivo determinado. . . " [11].
Es así como un videojuego puede tener tantos enfoques como situaciones a explorar, pero en este trabajo se toma particular atención al campo educativo y su impacto en la sociedad ecuatoriana, como resultado se encontraron varios proyectos de este tipo con el fin de entregar un producto que tenga un valor adicional y sirva como ayuda en el aprendizaje de algún tema en particular.
Al realizar una búsqueda de otros proyectos de videojuegos a nivel nacional enfocados en la educación se ha encontrado títulos muy interesantes y que vale la pena mencionar, como son:
El proyecto de Hussein Gabriel Rahman Núñez de la Universidad Técnica de Ambato, que para el 2007, creó un videojuego enfocado en la informática donde los movimientos del personaje se interpretan mediante comandos básicos de lógica de programación lo que permite conocer y entender cómo funcionan la mayoría de los videojuegos y las ciencias computacionales desde un punto de vista distinto al método tradicional de educación teniendo unos resultados favorables al momento de publicar el juego en la plataforma "Game Jolt" con más de 1000 descargas [12].
También se encuentra el programa creado por Alejandro Pardina Bustamante de la Escuela Politécnica Nacional en 2017 donde su enfoque fue mejorar el trabajo en equipo utilizando un videojuego multijugador, comprobando que se mejoró las capacidades colaborativas de las personas que se atrevieron a probar el juego [13].
Con estos ejemplos se puede conocer que los videojuegos pueden ser utilizados como herramientas de aprendizaje haciendo que sus usuarios aprendan algo nuevo mientras resuelven problemas y se entretienen.
En este proyecto se busca crear un videojuego que cumpla con la misma función de enseñar a la gente sobre la cultura y mitología de los pueblos incas en el Ecuador; es lamentable saber que no se ha tratado de realizar un videojuego basado en este tema ya que la historia de los ancestros del país es rica y multidiversa y por lo tanto se puede considerar un primer paso para conocer mejor las costumbres y tradiciones autóctonas del Ecuador.
La presente investigación se enmarca en el paradigma Crítico Propositivo, es crítico porque realiza un Análisis Crítico del problema, y es Propositivo porque busca proponer una solución factible al problema.
Según el plan nacional del Buen Vivir de la constitución de la República del Ecuador en el Art. 293 en el numeral 2.3 se encuentra "Promover el rescate, reconocimiento, investigación y protección del patrimonio cultural, saberes ancestrales, cosmovisiones y dinámicas culturales." De esta manera se cumple con la ley y se beneficia a la nación y a la cultura ecuatoriana.

Entre los términos más utilizados y de los cuales se basa la presente investigación son:

Periodo prehispánico. Dicho de América o de lo relacionado con ella: Anterior a la conquista y colonización españolas [14].
Civilizaciones de América Latina. En la américa precolombina existieron grandes civilizaciones como la civilización olmeca, los mayas, los quiches, culturas del Valle de México, la civilización peruana de Norte Chico y el imperio inca [15].

Imperio inca. Fundadores de uno de los imperios más grandes de la historia de la humanidad, el cual fue conocido como el Tahuantinsuyo ocupaba un territorio de más de 4000 Km y tenía una población de cerca de 12 millones de habitantes reinaron el cono sur de América hasta su declive en el siglo XVI [16].
Cultura y mitología inca ecuatoriana. La cultura inca estaba llena de costumbres, tradiciones y dioses a los cuales sus pobladores adoraban fervientemente para recibir bendiciones, especialmente para sus cosechas. En el panteón Inca se muestra la imagen de Viracocha, dios de la creación, Inti, dios del sol, Mama quilla, diosa de la luna, Pachamama o madre tierra entre otros seres mitológicos y héroes que hicieron hazañas que se cuentan hasta hoy en día [17].
Informática. Ciencia encargada de estudiar y desarrollar máquinas para tratar y transmitir información, así como, los métodos para procesarla. Y también se puede decir que es un conjunto de conocimientos tanto teóricos como prácticos sobre cómo se construyen, funcionan y utilizan las computadoras [18].
Programación. " Es el proceso por medio del cual se diseña, codifica, limpia y protege el código fuente de programas computacionales. A través de la programación se dictan los pasos a seguir para la creación del código fuente de programas informáticos. De acuerdo con ellos el código se escribe, se prueba y se perfecciona " [19].
Videojuegos. El concepto de Johan Huizinga es utilizado tanto en juego reales como virtuales y dice. "El juego es una actividad libre y consciente que ocurre fuera de la vida 'ordinaria' porque se considera que no es seria, aunque a veces absorbe al jugador intensa y completamente. [...] Promueve la formación de grupos sociales que tienden a rodearse a sí mismos de secreto y a acentuar sus diferencias respecto del resto utilizando los medios más variados" [20].
Videojuegos 2D. Es un tipo de videojuego de 2 dimensiones donde se debe tomar en cuenta las colisiones de los cuerpos como en la vida real, este sistema de detección de colisiones es habitualmente un módulo que ayuda a crear acciones para los objetos dentro del videojuego [21].

1.4 Objetivos

1.4.1 General

- Desarrollar un videojuego 2D, enfocado en la enseñanza de las costumbres y mitología inca ecuatoriana.

1.4.2 Específicos

- Investigar las costumbres, tradiciones y mitología del pueblo inca.
- Seleccionar un motor gráfico para el desarrollo de videojuegos.
- Utilizar una metodología ágil que se adapte al desarrollo de videojuegos.
- Implementar el videojuego en 2D enfocado en la enseñanza de las costumbres y mitología inca ecuatoriana.

Capítulo 2

Metodología

2.1 Métodos

2.1.1 Modalidad de la investigación

- **Investigación Bibliográfica**
  La investigación será bibliográfica ya que se apoyará en libros, revistas, tesis del área de informática, documentos técnicos, trabajos de titulación del área de ciencias humanas y arte cuentos, leyendas y leyes existentes para la elaboración del marco teórico que se divide en dos partes fundamentales; la historia, costumbres y tradiciones del pueblo inca en el Ecuador y los videojuegos.

- **Investigación de modalidad especial**
  Al ser una investigación que se centra en un aspecto social se le considera de modalidad especial porque busca, mediante una solución tecnológica, contribuir al desarrollo cultural del país y rescatar costumbres y tradiciones propias de esta tierra.

2.1.2 Población y muestra

El presente proyecto no tiene una población específica ya que su dimensión es global y busca llegar a la mayor cantidad de gente posible.

2.1.3 Procesamiento y análisis de datos

Se debe empezar por recopilar información sobre la cultura y mitología inca, paralelamente a esto, investigar sobre los motores de juegos y metodologías ágiles que permitan el desarrollo del proyecto.

Historia general de la cultura inca

A principios del siglo XIII apareció una tribu indígena de lengua quechua cerca de Cuzco que enseguida adoptó el nombre de su líder: Inca. A pesar de tener dificultades para adaptarse al nuevo territorio tuvo que acoger las costumbres de los pueblos cercanos pero sin abandonar su idioma original. La historia narrada oralmente de los Incas comienza con el primer inca, Manco Cápac.
Se establecieron en el interior de los Andes del Perú Meridional, en la cuenca de Cuzco, cerca de los ríos Huaytanay y Tullumayo, una región ocupada por otras tribus. Después de varios batallas que duraron alrededor de dos siglos el único

pueblo que quedaba era el inca, lo que hizo que se convirtiera en uno de los imperios más grandes que ha conocido la humanidad.
Luego de varios conquistas más allá de su territorio llegó a anexar pueblos formando así el el Tahuantinsuyo, que ocupó una vasta superficie, extendiéndose a lo largo del territorio de las actuales repúblicas del Perú, Ecuador, Bolivia, Argentina, Chile y el sur de Colombia.[22]

Figura 2.1: Imperio Inca
Elaborado por: Carlos Acuña

Personajes principales

Huaynacápac

Huayna Cápac fue el undécimo y antepenúltimo inca del Tahuantinsuyo. Emperador del Cuzco. No está claro el lugar de su nacimiento, se estima nacido en Cuzco, aunque se discute si probablemente nació en Tomebamba, actual ciudad de Cuenca. Hijo del Sapa Inca Túpac Yupanqui y de la Coya Mama Ocllo, nacido durante las expediciones de conquista emprendidas por su padre durante el reinado de Pachacútec. Durante su infancia y juventud fue llamado Titu Cusi Huallpa. Túpac Yupanqui enfermó en Chinchero, eligiendo como su sucesor al menor de sus hijos, hecho que disgustó a algunas panacas cuzqueñas que esperaban que el sucesor fuera Cápac Guari hijo de la concubina Chuqui Ocllo. Gracias a la oportuna intervención de su tío materno Huamán Achachi la conspiración no prosperó y fue nombrado Inca tomando el nombre de Huayna Cápac, Con un inicio tan agitado empezó el gobierno del nuevo Inca, que básicamente tuvo que dedicar todos sus esfuerzos a consolidar los terrenos conquistados por su padre y sofocar las revueltas de provincias levantiscas. Para esto, asumió el control político y religioso del Imperio, desplazando a Apo Chalco Yupanqui, el vigente Villac Umo. Por primera vez en la época imperial se concentraban todos los poderes en una sola persona. Sin embargo, casi al final de su vida nombra un pariente suyo, Cusi Túpac Yupanqui, como nuevo Sumo Sacerdote del Sol

(aparentemente este es el Villac Umo que corona a Manco Inca y que lo secundaría en sus guerras de reconquista como hábil estratega). Antes de su muerte, alrededor del año 1525 dividió su imperio entre sus hijos Atahualpa y Húascar y fue conocido como uno de los mejores líderes de todos los tiempos junto a su padre y abuelo. [23]

Figura 2.2: Huaynacápac
Elaborado por: Carlos Acuña

Atahualpa

Hijo de la princesa shyri Pacha y del rey inca Huaynacápac o (Huayna - Cápac), atahualpa es el fruto de la sangrienta conquista inca en nuestro país. Nació en Quito probablemente en 1497. Durante un tiempo vivió en paz junto a su padre y no tuvo conflictos, al menos bélicos, con su medio hermano Húascar. La división del reino hecha por su padre propició la guerra entre los dos hermanos. Cómo líder militar tomó sabias decisiones. Durante las batallas contra su medio hermano supo aprovechar las victorias y manejar bien las derrotas. Con sus enemigos fue implacable. Cuando llegaron los invasores españoles no quiso aceptar la fé católica ni el bautismo y fue condenado. Pagaron por su rescate, pero fue en vano. El último emperador inca murió el 26 de julio de 1533. Su pueblo afirmó qu en ese momento anocheció a mitad del día. [24]

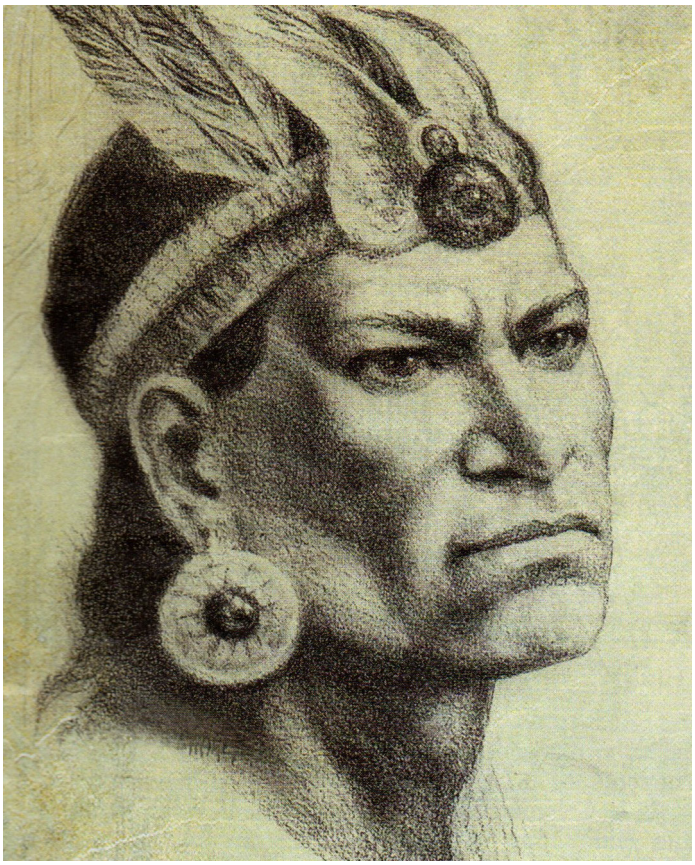

Figura 2.3: Atahualpa
Elaborado por: Carlos Acuña

Huáscar

Primogénito de Huaynacápac o (Huayna - Cápac), y Araua Ocllo. Nació en el Cuzco, alrededor de 1491. Huáscar era de linaje inca puro. Esto era muy importante en el Tahuantinsuyo, pues se creía que los incas de sangre pura descendían directamente de la divinidad. Jamás aceptó que su medio hermano Atahualpa gobierne junto a él, así que, influido por la nobleza cuzqueña, decidió atacarlo. Aunque envió varios contingentes , perdió muchas batallas y no pudo detener la avanzada de los ejércitos del norte quienes por órdenes de Atahualpa decidieron tomarse la capital, el Cuzco. Fue aprisionado Quisquis. Su captura significó la victoria definitiva de Atahualpa. Murió tiempo después de ser capturado y encerrado en la fortaleza de Jauja. [24]

Figura 2.4: Huáscar
Elaborado por: Carlos Acuña

Quisquis

Destacado general inca. Según algunos biógrafos, su nombre significa pequeña ave. Estuvo bajo las órdenes de Huaynacápac; después de su muerte, permaneció al lado de Atahualpa. Tuvo triunfos y derrotas frente a las tropas de Huáscar, pero en Quipaipán elaboró una brillante estrategia para capturar al líder cuzqueño. Después de que los españoles llegaron y atahualpa fuera encarcelado, Quisquis o (Quis-quis) opuso resistencia a los españoles, enfrentándolos y capturando a varios de ellos. Intentó crear una reunificación inca para defenderse de los invasores pero no lo consiguió. Fue asesinado por sus propios capitanes, al no poder convencerlo de rendirse ante los españoles. [24]

Figura 2.5: Quisquis
Elaborado por: Carlos Acuña

Calicuchima

Hijo de Epiclachima y comandante de los ejércitos de Atahualpa. Luego de perder las primeras batallas ante los invasores del Cuzco, preparó junto a Quisquis ofensiva meditada. Aislaron al ejército de Huáscar y lo llevaron al territorio del norte para tenderle una emboscada en Molleambato. Mediante esa estrategia obtienen la primera de varias victorias frente a los guerreros del Sur. Participó en la captura de Huáscar. Fue Calicuchima quien supuestamente entregó el oro de rescate que los invasores españoles pedían por Atahualpa. Fue condenado a muerte debido a que mantenía correspondencia con Quisquís, quien se declaró en rebeldía. [24]

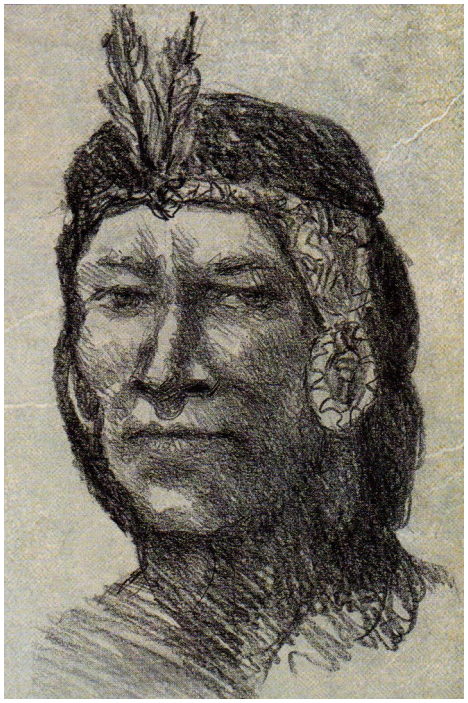

Figura 2.6: Calicuchima
Elaborado por: Carlos Acuña

Chapera

Cacique del reino Cañari. No quiso someterse a Atahualpa y renegó de él para aliarse con Húascar. Avanzó junto a Apu Atoc con el fin de de derrotar a Atahualpa. Fue capturado en la batalla de Molleambato. Posteriormente fue ejecutado en Quito. Su pueblo sufrió la violencia de Atahualpa por haber sido traicionado. [24]

Figura 2.7: Chapera
Elaborado por: Carlos Acuña

Apu Atoc

Capitán al mando del ejército de Huáscar. Aunque sus primeras incursiones por el territorio del norte fueron exitosas, sus contingentes fueron los en sufrir la derrota de los ejércitos de Atahualpa. Fue capturado en la batalla de Molleambato y ejecutado en Quito por las órdenes de Rumiñahui. [24]

Figura 2.8: Apu Atoc
Elaborado por: Carlos Acuña

Batallas principales

Batalla de Molleambato

Antecedentes. El conquistador inca Huaynacápac cumplió con la misión de unificar el imperio inca a través de numerosas batallas. Cuando llegó su vejez decidió heredar el Tahuantinsuyo a sus descendientes. Húascar se quedó con el Cuzco, mientras que Atahualpa hijo de Paccha, princesa de los quitus, heredó el norte. El pueblo de Cuzco , al no estar satisfecho con la división realizada se alió con los cañaris para destronar al heredero del norte. Atoc general de Huáscar, avanzó hasta el dominio cañari para aliarse con ellos. Al llegar su ejército desplazó a los guerreros del norte, quienes ofrecieron poca resistencia. Con el fin de detener la avanzada, Atahualpa, decide viajar a la comarca de Ambatus.[24]
El enfrentamiento. Atoc, quien dirigía la avanzada de Huáscar, decide descansar con su ejército entre Mocha y Ambato, con el fin de esperar refuerzos. Mientras tanto, los hombres de Atahualpa se movilizan a la orilla opuesta del río Ambato y esperan el ataque del adversario, quien tiene que cruzar el río para llegar hasta ellos.
El general Calicuchima, del ejército quiteño, con 5000 hombres, rodea las montañas occidentales y llega a Quisapincha (sur de Ambato) e inicia una ofensiva desde el sur. Las fuerzas de Atahualpa aprovechan la confusión entre las tropas sureñas y vadean el río para atacar desde el norte. Los cuzqueños fueron derrotados. La batalla fue tan sangrienta que el historiador Federico Gonzáles Suárez afirma que años más tarde los conquistadores hallaron el campo blanqueado debido a la cantidad de huesos no sepultados. Los sobrevivientes se refugiaron en Tomebamba. [24]

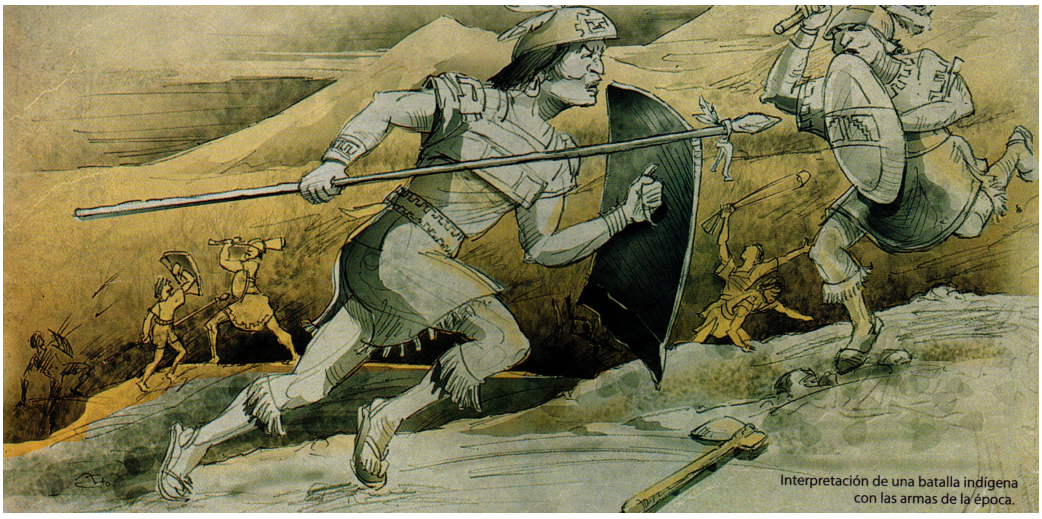

Figura 2.9: Batalla de Molleambato
Elaborado por: Carlos Acuña

Batalla de Quipaipán

Antecedentes. Después de su victoria en Molleambato, Atahualpa se dirigió a Tomebamba, donde fue capturado pero al poco tiempo logró escapar. No tardó en reunir a su ejército y derrotar a los contingentes de Huáscar.
Castigó duramente a los habitantes de Tomebamba que lo habían traicionado y se dirigió hacia los huancavilcas para obtener refuerzos. Tras varias varias batallas tomó ciudades como Cusubamba, Collahuayla, Pombo y Yanamarca y continúo su imparable su avance hacia el sur. Huáscar, contrariado por las derrotas, se movilizó con su ejército hasta Quipaipán, muy cerca de Cuzco para defender la capital.[24]
El enfrentamiento. Quisquis y Calicuchima se asentaron con su ejército cerca del río Cotabamba y enviaron un contingente para armar un ataque. Mientras que Topa Atao, comandante cuzqueño, se movilizó hacia esa zona para abrirle paso a Huáscar.
Calicuchima, por el sur cruzó el río con cinco mil hombres al enterarse que Huáscar descendería por esa zona. Los guerreros del Cuzco recibieron el ataque sin poder defenderse, entonces Calicuchima comunicó a Quisquís que Huáscar seguramente pretendía huir por el norte y lo movilizó hasta allá.
Huáscar llegó desprevenido y al ver muertos a sus hombres intentó huir. Quisquís llegó desde el norte, interceptando al emperador cuzqueño y lo capturó. [24]

Figura 2.10: Batalla de Quipaipán
Elaborado por: Carlos Acuña

Mitología inca

La mitología inca fue transmitida a través de los siglos mediante un lenguaje oral, ya que este pueblo no contaba con un lenguaje escrito, sin embargo sus dioses y su mitología sobrevivió el paso del tiempo y por lo tanto se conoce lo siguiente.

Bestias mitológicas

Amaru

Es una criatura fusión de animales, ya que tiene forma de serpiente, cabeza de camélido, alas de paraje, garras de león y cola de pez. Está relacionada con el agua, pues esta representa la vida que se encuentra en el elemento acuático, esta criatura se encarga de resguardar la vida que se encuentra en los ríos y lagos. [25]

Figura 2.11: Amaru
Elaborado por: Carlos Acuña

Apallimay

Criatura que suele manifestarse en caminos apartados como un inofensivo bebe, el cual pide con llantos que lo carguen, cuanto esto sucede y logra acomodarse en la espalda de su víctima, la criatura comienza a crecer rápidamente hasta convertirse en una pesada carga, con rostro de anciano, una boca con colmillos y fracciones llenas de ira y rencor.
Se dice que para que la victima se libere de esta terrible criatura, necesita únicamente los servicios de un curandero, ya que sino el Apallimay tomará la energía vital de la persona hasta provocarle la muerte.[25]

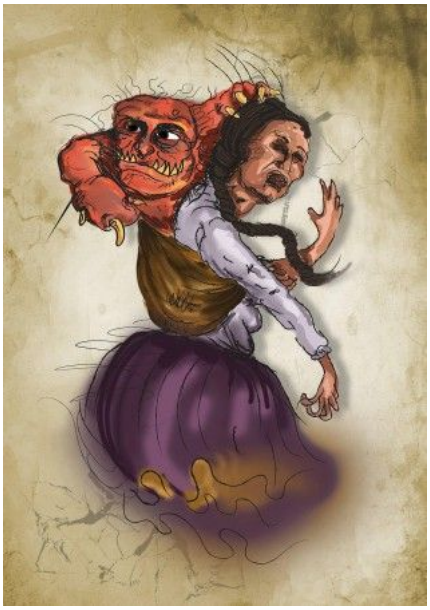

Figura 2.12: Apallimay
Elaborado por: Carlos Acuña

Jarjacha

Según la mitología Inca, cuando una persona comete incesto o infidelidad, su alma puede escapar de su cuerpo y tomar diferentes formas aterradoras. No necesariamente la persona debe estar muerta para que esto pueda suceder, sino que simplemente puede estar dormida.
Estas almas suelen representarse como una llama con una o varias cabezas, e incluso puede aparecer con el cuerpo de llama, pero con cabeza de humano, dependiendo de la frecuencia con que cometieron la falta.
Según las historias, es muy peligroso encontrarse con estas criaturas, ya que tienen el poder de hipnotizar a las personas mirándolas fijamente a los ojos para luego asesinarlas.[25]

Figura 2.13: Jarjacha
Elaborado por: Carlos Acuña

Distribución de personajes históricos y mitológicos para el desarrollo del videojuego

Personaje	**Uso**
Huáscar	Jefe de nivel
Chapera	Jefe de nivel
Apu Atoc	Jefe de nivel
Amaru	Enemigo de nivel
Apallimay	Enemigo de nivel
Jarjacha	Enemigo de nivel

Cuadro 2.1: Personajes del videojuego

Elaborado por: Carlos Acuña

Capítulo 3

Resultados y Discusión

3.1 Análisis y discusión de los resultados

3.1.1 Estudio de Frameworks para el desarrollo del proyecto

Frameworks

En arquitectura de software se usa la palabra framework para describir un conjunto de diseños reusables y código que puede asistir en el desarrollo de software de aplicaciones. Un framework de aplicaciones provee a los desarrollares una estructura y plantilla que pueden usar como base para construir sus aplicaciones. Los frameworks incluyen clases abstractas, clases concretas e interacciones predefinidas sobre las clases. Siendo así que los desarrolladores pueden reducen el esfuerzo de desarrollo a través de la reutilización de código y diseños proporcionados por el framework [26].

Unity3D

Unity es un framework y motor de videojuegos multiplataforma creado por Unity Technologies. Unity está disponible como plataforma de desarrollo para Microsoft Windows, Mac OS, Linux y dispositivos móviles que permite la creación de videojuegos mediante la utilización de objetos que pueden ser importados hacia la herramienta, como son modelos 2D y 3D, archivos de texto, imágenes, sonidos, etc. Todo es hecho mediante varios lenguajes de programación como son BOO, C# y Javascript. MonoDevelop es el IDE con el que trabaja Unity además de usar API's para el uso de rutinas, estructura de datos y clases, estas API's también son conocidas como librerías [27].
Unity es completamente libre para cualquiera o cualquier empresa que gane menos de $ 100000 al año, puede ser descargado sin ningún cargo adicional y sin tarjeta de crédito [28].

Godot Engine

Godot Engine es un software diseñado para crear juegos. Contiene muchas herramientas para crear toda clase de juegos 2D, con varios niveles de complejidad. Tiene un tamaño menor en comparación con otros motores de videojuegos. Godot está disponible bajo licencia MIT, por lo tanto se puede usar de manera libre y redistribuir el juego de cualquier manera.
Godot soporta múltiples plataformas y los videojuegos son codificados en C++,

C#. Godot tiene un editor con auto indentación, resaltado de sintaxis, autocompletado de código y depurador que soporta breakpoints y ejecución paso a paso [29].

Unreal Engine

Unreal Engine fue creado en 1998 con el advenimiento del primer juego de disparos en primera persona Unreal. Esta herramienta soporta DirectX and OpenGL, además de ser viable para otras plataformas como OS X, iOS, Android, Flash, JavaScript y HTML 5, Unreal Engine 4 es la última versión de este motor de videojuegos el cual está completamente programado en C++. [30].
Unreal Engine utiliza el sistema de secuencias de comandos visuales Bluesprints donde el concepto es utilizar una interfaz basado en nodos para crear elementos de juego desde Unreal Editor. Al igual que muchos lenguajes, se utiliza para definir clases u objetos orientados a objetos en el motor. Este sistema es extremadamente flexible y potente, ya que brinda a los diseñadores la capacidad de utilizar las mismas herramientas que utilizan los programadores. [31]

3.1.2 Comparación de diferentes motores de juegos para la creación de la aplicación

En la sección 3.1.1 se da a conocer algunos de los frameworks de videojuegos que se pueden utilizar para la creación del proyecto. Por lo tanto el cuadro 3.1 muestra una comparación entre estas herramientas con el fin de escoger la mejor para el desarrollo del proyecto basado en los siguientes criterios.

- **Requisitos mínimos.** Se refiere a requisitos tanto de software o hardware que un desarrollador debe disponer.
- **Orientación 2D.** Se refiere a la característica del framework que permite realizar videojuegos en 2 dimensiones.
- **Licenciamiento.** Se refiere a las licencias que se otorgan a los desarrolladores, tanto gratuitas como pagadas.
- **Lenguaje de alto nivel.** Se refiere al lenguaje de programación que el desarrollador debe utilizar para trabajar con el framework de creación de videojuegos.
- **Soporte de plataformas.** Se refiere a la posibilidad de correr el juego en varias plataformas.
- **Documentación.** Se refiere a la cantidad de documentación disponible de la herramienta que se desea utilizar junto con la ayuda creada por la comunidad que trabaja con el framework.

	Unity	**Godot Engine**	**Unreal Engine**
Requisitos mínimos	Windows 7 y Windows 10 64-bit	Windows 7	Windows 10 64-bit
Orientación 2D	2D	2D	2D
Licenciamiento	Completamente libre para cualquiera o cualquier empresa que gane menos de $ 100000 al año	MIT License	5% de regalías cuando las ventas del juego superan los $ 3000
Lenguaje de alto nivel	C#, Javascript	GDScript	C++
Soporte de plataformas	Windows, Mac, Linux	Windows, Mac, Linux	Windows, Mac, Linux
Documentación	Existe documentación tanto en la página oficial de Unity como en foros, videos, blogs y varios sitios de Internet	Existe documentación en la página oficial de Godot pero muy poca información en otros recursos externos	Existe documentación en la página oficial de Unreal Engine pero es demasiado básica y en otros recursos la mayoría de la ayuda conlleva el pago de una cantidad de dinero

Cuadro 3.1: Cuadro comparativo de motores de juegos

Elaborado por: Carlos Acuña

En base a los resultados obtenidos se puede determinar que Unity tiene más ventaja sobre los otros dos motores de juego, especialmente en el apartado de documentación ya que al existir una gran cantidad de información hace que la herramienta sea más fácil de entender y se pueda consultar sobre cualquier problema en el transcurso del proyecto.

3.1.3 Componentes de la interfaz de Unity

Interfaz de Unity

Unity está conformado por ventanas que muestran varias partes del proyecto que se está ejecutando, las cuales se puede reorganizar, agrupar, separar y acoplar como se muestra en la figura 3.1

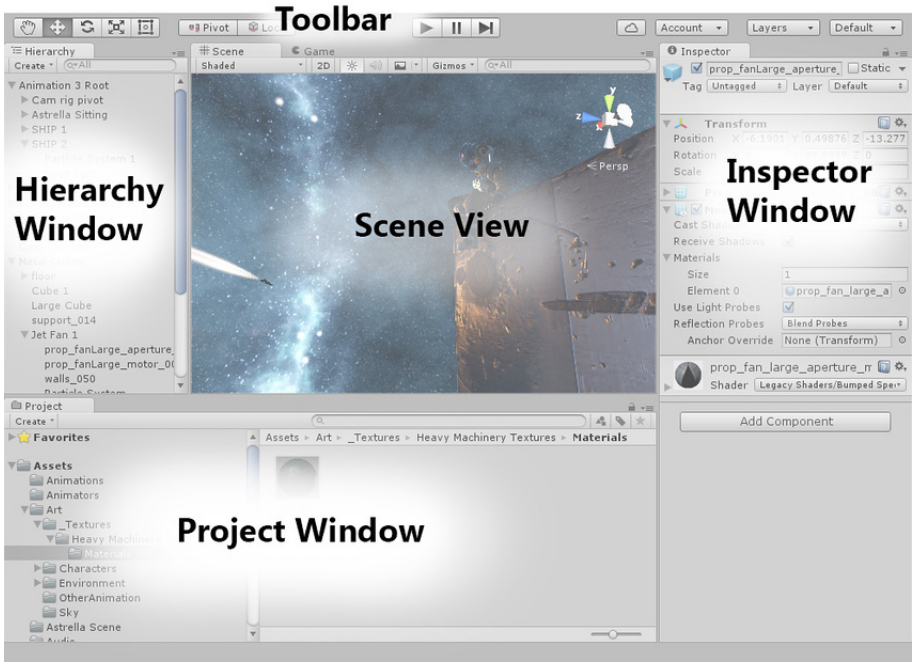

Figura 3.1: Editor de Unity
Elaborado por: Carlos Acuña

- **Project Window.** (ventana del proyecto) muestra los assets de librería que están disponibles para ser usados. Cuando se importe los assets al proyecto, estos aparecen aquí [32].

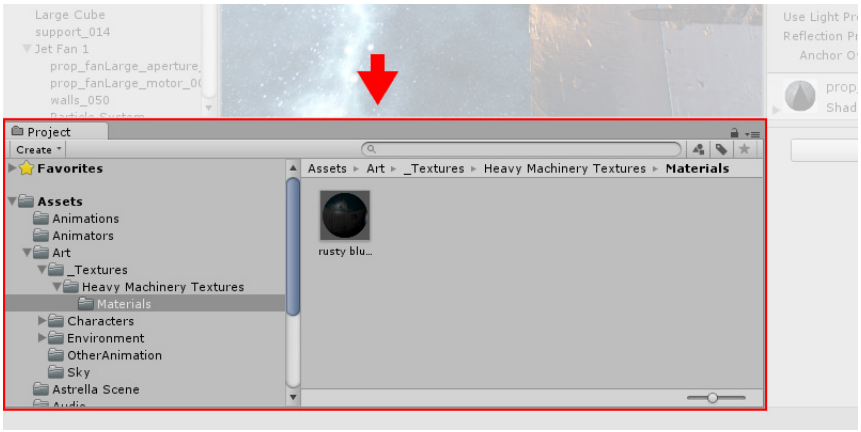

Figura 3.2: Project Window de Unity
Elaborado por: Carlos Acuña

- **Scene View.** (vista de escena) permite la navegación visual y editar la escena. La scene view puede mostrar una perspectiva 2D o 3D dependiendo en el tipo de proyecto en el que la que se está trabajando [32].

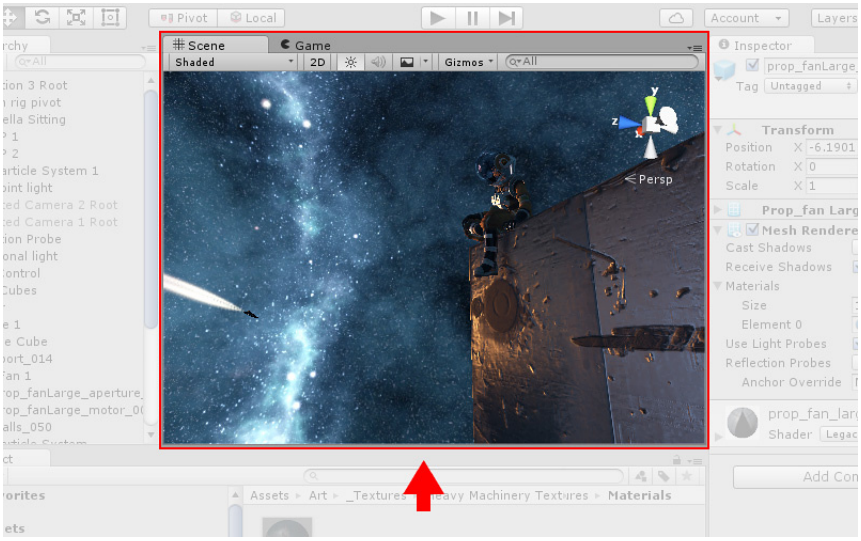

Figura 3.3: Scene View de Unity
Elaborado por: Carlos Acuña

- **Hierarchy Window.** (vista de jerarquía) es una representación de texto jerárquico de cada objeto en la escena. Cada elemento en la escena tiene una entrada en la jerarquía, por lo que las dos ventanas están inherentemente vinculadas. La jerarquía revela la estructura de cómo los objetos están agrupados el uno al otro [32].

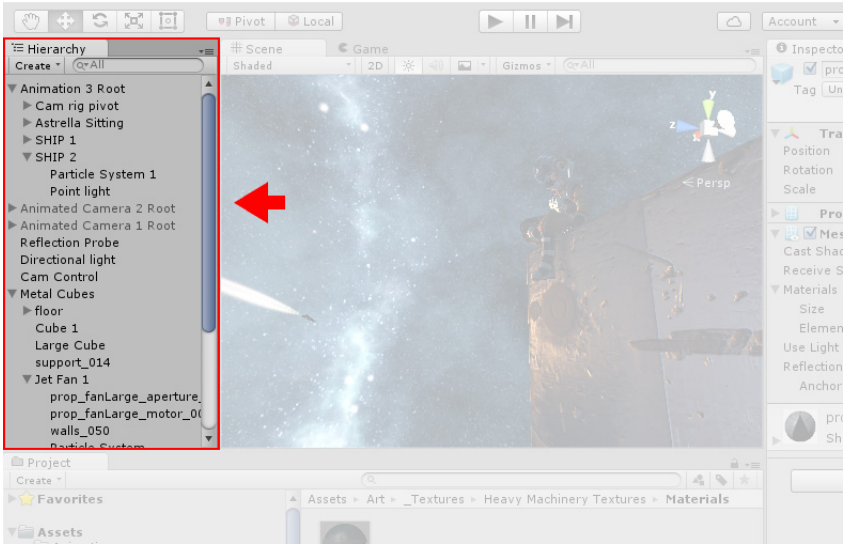

Figura 3.4: Hierarchy Window de Unity
Elaborado por: Carlos Acuña

- **Inspector Window.** (ventana de inspector) permite visualizar y editar todas las propiedades del objeto actualmente seleccionado. Ya que diferentes objetos tienen diferentes propiedades, el layout (diseño) y contenido de la ventana del inspector va a variar [32].

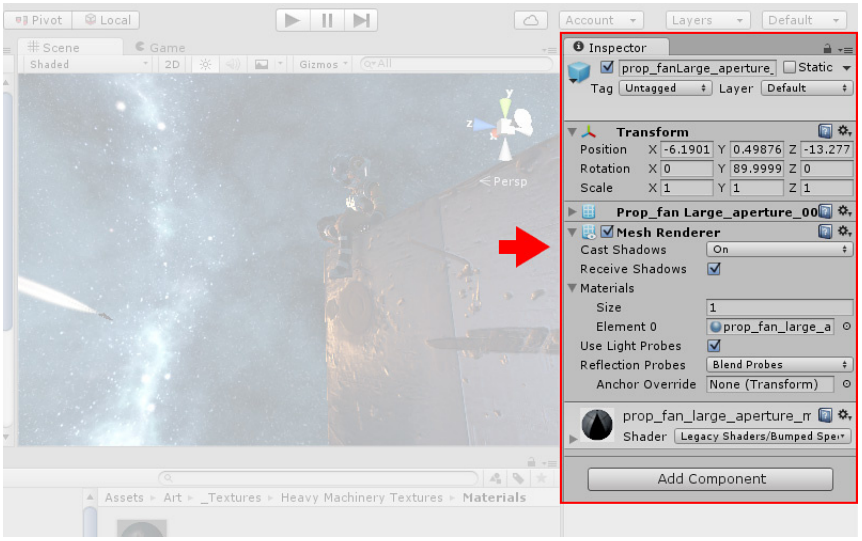

Figura 3.5: Inspector Window de Unity
Elaborado por: Carlos Acuña

- **Toolbar.** (barra de herramientas) proporciona un acceso a las características más esenciales para trabajar. En la izquierda contiene las herramientas básicas para manipular la scene view y los objetos dentro de esta. En el centro están los controles de reproducción, pausa, y pasos. Los botones a la derecha le dan acceso a sus servicios de Unity Cloud y su cuenta de Unity, seguido por un menú de visibilidad de capas, y finalmente el menú del layout del editor (que proporciona algunos diseños alternativos para la ventana del editor, y permite guardar layouts personalizados). La barra de herramienta no es una ventana, y solamente es parte de la interfaz de Unity que usted no puede re-ajustar [32].

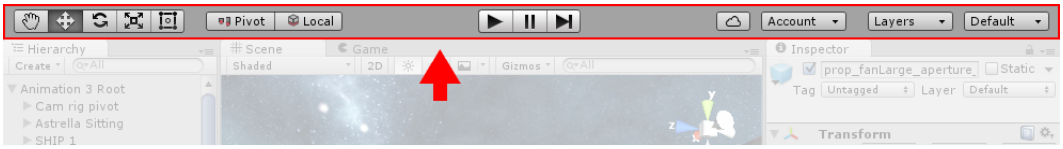

Figura 3.6: Toolbar de Unity
Elaborado por: Carlos Acuña

GameObject

Los GameObjects son objetos fundamentales en Unity que representan personajes, props, y el escenario. Estos no logran nada por sí mismos pero funcionan como contenedoras para Components, que implementan la verdadera funcionalidad.
Un GameObject siempre tiene el componente Transform adjunto (para representar la posición y orientación) y no es posible quitar esto. Los otros componentes que le dan al objeto su funcionalidad pueden ser agregados del menú Component del editor o desde un script. También hay muchos objetos útiles pre-construidos (figuras primitivas, cámaras, etc) [32].

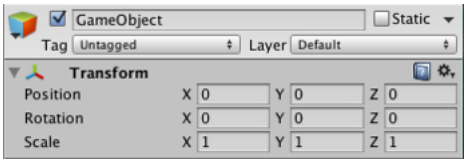

Figura 3.7: GameObject de Unity
Elaborado por: Carlos Acuña

Transform Component

Es imposible crear un GameObject en el Editor sin un Transform Component. Este componente define la posición del GameObject, la rotación y la escala en el mundo del juego y Scene view. El Transform Component también habilita un concepto llamado 'crianza de los hijos', que es una parte crítica del trabajo con GameObjects [32].

Prefabs

En el sistema prefabricado de Unity, los Prefabs actúan como plantillas. Se crean Prefabs en el editor y se guardan como un activo en la ventana del proyecto. Desde Prefabs, se puede crear cualquier número de instancias prefabricadas. Las instancias prefabricadas pueden crearse en el editor y guardarse como parte de sus escenas, o instanciarse en tiempo de ejecución [32].

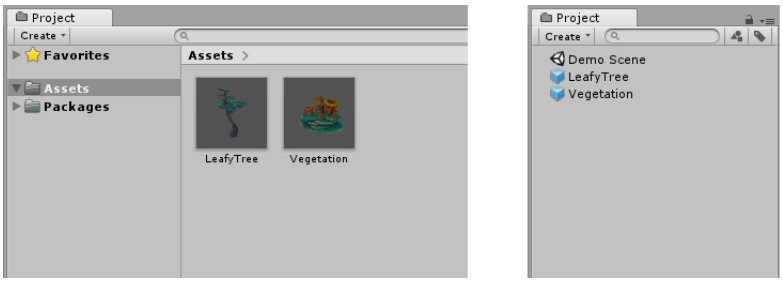

Figura 3.8: Prefabs de Unity
Elaborado por: Carlos Acuña

Input Convencional de Juego

Unity soporta teclado, joystick y gamepad input.
Se pueden crear ejes y botones virtuales en la ventana de entrada, y los usuarios finales pueden configurar la entrada del teclado en un bonito diálogo de configuración de pantalla [32].

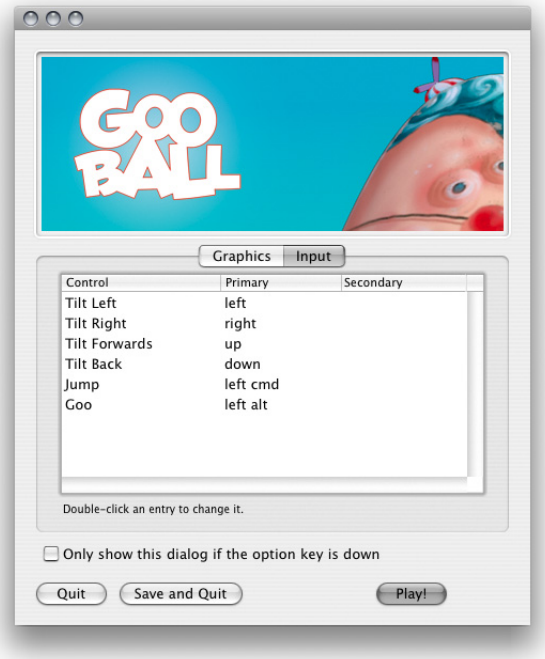

Figura 3.9: Input de Unity
Elaborado por: Carlos Acuña

Virtual Axes (Ejes Virtuales)

Desde los scripts, todos los ejes virtuales son accedidos por su nombre. Cada proyecto tiene el siguiente eje input por defecto cuando es creado:
Horizontal y Vertical son asignados a las teclas w, a, s, d y las teclas de flecha.
Fire1, Fire2, Fire3 están asignadas a las teclas Control, Option (Alt), y Command, respectivamente.
Mouse X y Mouse Y están asignados al movimiento delta del mouse.
Window Shake X y Window Shake Y es asignado al movimiento de la ventana.

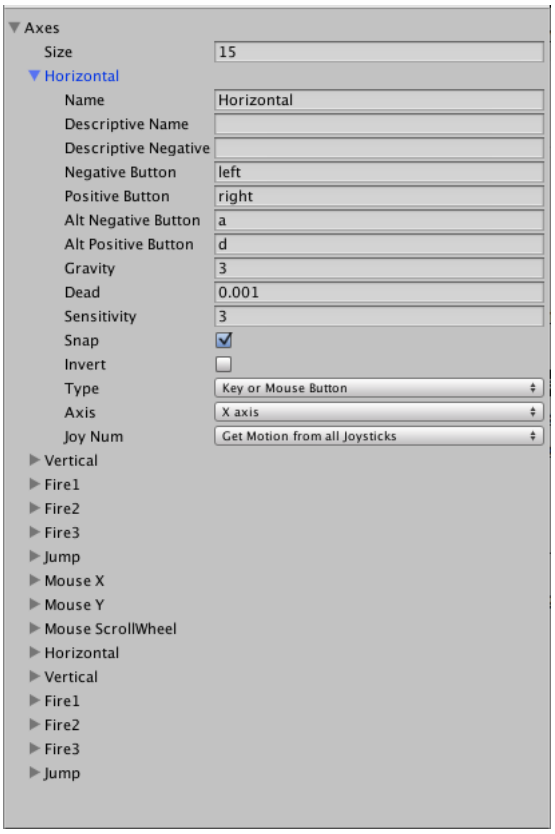

Figura 3.10: Virtual Axes de Unity
Elaborado por: Carlos Acuña

3.1.4 Selección de metodologías ágiles para el desarrollo de videojuegos

Al desarrollar un proyecto de software es prudente utilizar un metodología de desarrollo, preferiblemente ágil, con el fin de desarrollar un proyecto de manera ordenada, que tenga que cumplir metas en un tiempo determinado y que no se desperdicie recursos valiosos durante la creación del software.

Metodologías

SUM

La metodología SUM para videojuegos tiene como objetivo desarrollar videojuegos de calidad en tiempo y costo, así como la mejora continua del proceso para incrementar su eficacia y eficiencia. Pretende obtener resultados predecibles, administrar eficientemente los recursos y riesgos del proyecto, y lograr una alta productividad del equipo de desarrollo. SUM fue concebida para que se adapte a equipos multidisciplinarios pequeños (de tres a siete integrantes que trabajan en un mismo lugar físico o estén distribuidos), y para proyectos cortos (menores a un año de duración) con alto grado de participación del cliente.[33]

Roles

La metodología define cuatro roles: equipo de desarrollo, productor interno, cliente y verificador beta. El productor interno y el cliente se relacionan en forma directa con los roles de SCRUM Master y Product Owner de SCRUM respectivamente.

- **Equipo de desarrollo**. Tiene las características del Scrum team, pero a diferencia de SCRUM se definen subroles dentro del equipo, estos son equivalentes a los usados por la industria local tales como: programador, artista gráfico, artista sonoro y diseñador de juego. Es necesario esta definición ya que se requiere una alta especialización para satisfacer las distintas disciplinas que involucra del desarrollo de videojuegos, aspecto no contemplado en Scrum [33]. El equipo de desarrollo consta de:
  - **Programador.** Define la arquitectura, realiza el dise˜no, implementación y verificación de los componentes de software e integra el contenido audiovisual del videojuego. [33]
  - **Artista gráfico.** Realiza el arte de concepto, el arte 2D, el modelado 3D y la creación de animaciones y texturas. [33]
  - **Artista sonoro.** Se encarga de la creación, grabación, mezcla y edición de los efectos de sonido y música del juego. [33]
  - **Diseñador de juego.** Es el encargado de diseñar el gameplay, la historia, el ambiente, los personajes y todos los elementos que hacen a la experiencia del jugador. Además, diseña los niveles, misiones y los desafíos que enfrenta el jugador. [33]

- **Productor interno.** Responsable de la planificación y ejecución del proyecto, define los objetivos y ayuda a resolver problemas que ocurren en el proyecto. Se comunica con el cliente y mantiene al equipo de desarrollo enfocado en el proyecto. [33]

- **Cliente.** Es el dueño del producto que valida el concepto del juego, aprueba los planes del juego, prioriza las tareas del videojuego y participa en la evaluación del proyecto. [33]

- **Verificador Beta**. Realizan la verificación funcional del videojuego. Dependiendo del tamaño del proyecto pueden participar varios verificadores beta [33]

Ciclo de vida

El ciclo de vida se divide en fases iterativas e incrementales que se ejecutan en forma secuencial con excepción de la fase de gestión de riesgos que se realiza durante todo el proyecto. Las cinco fases secuenciales son: concepto, planificación, elaboración, beta y cierre. [33]

- **Concepto.** Tiene como objetivo principal definir el concepto del videojuego lo que implica definir aspectos de negocio (público objetivo, modelo de negocio), de elementos de juego (principales características, gameplay, personajes e historia entre otros) y técnicos (lenguajes y herramientas para el desarrollo). El concepto del videojuego se construye a partir de ideas y propuestas de cada rol involucrado sobre los aspectos a definir. Las propuestas se refinan a través de reuniones y se analiza su factibilidad con pruebas de concepto. Esta fase analiza cuando se tiene el concepto validado entre todas las partes involucradas. [33]

- **Planificación.** La fase tiene como objetivo principal planificar las restantes fases del proyecto. Para ello es necesario definir el cronograma del proyecto junto con sus principales hitos, conformar el equipo para la fase de elaboración de acuerdo a las necesidades técnicas del proyecto, determinar y tercerizar las tareas que el equipo no pueda cumplir, definir el presupuesto y especificar el videojuego. [33]

- **Elaboración.** El objetivo de esta fase es implementar el videojuego. Para ello se trabaja en forma iterativa e incremental para lograr una versión ejecutable del videojuego al analizar cada iteración. [33]

- **Beta.** La fase tiene como objetivos evaluar y ajustar distintos aspectos del videojuego como por ejemplo gameplay, diversión, curva de aprendizaje y curva de dificultad, además de eliminar la mayor cantidad de errores detectados. Se trabaja en forma iterativa liberando distintas versiones del videojuego para verificar. Para ello primero se distribuye la versión beta del videojuego a verificar y se determinan los aspectos a evaluar y la forma de comunicación. Mientras la versión se verifica, se envían reportes con los errores o evaluaciones realizadas. Estos reportes son analizados para ver la necesidad de realizar ajustes al videojuego. Se puede optar por liberar

una nueva versión del videojuego para verificar una vez que se realizan los ajustes. El ciclo termina cuando se alcanza el criterio de finalización establecido en el plan del proyecto. [33]

- **Cierre.** Esta fase tiene como objetivos entregar la versión final del videojuego al cliente según las formas establecidas y evaluar el desarrollo del proyecto. Para la evaluación se estudian los problemas ocurridos, los éxitos conseguidos, las soluciones halladas, el cumplimiento de objetivos y la certeza de las estimaciones. Con las conclusiones extraídas se registran las lecciones aprendidas y se plantean mejoras a la metodología. En la evaluación es recomendable que participen todas las personas que han estado involucradas en el proyecto. [33]

- **Gestión de riesgos.** Esta fase se realiza durante todo el proyecto con el objetivo de minimizar la ocurrencia y el impacto de problemas. Esto se debe a que distintos riesgos pueden ocurrir en cualquiera de las fases, por lo cual siempre debe existir un seguimiento de los mismos. Para cada uno de los riesgos que se identifican se debe establecer la probabilidad y el impacto de ocurrencia, mecanismos de monitoreo, estrategia de mitigación y plan de contingencia. [33]

DAV

DAV (Desarrollo Ágil de Videojuegos), es una metodología que nace con la unión de las características de SCRUM y XGD, con la finalidad de proporcionar una estructura que permita la creación de videojuegos de forma técnica y sencilla. Esta metodología se compone de los siguientes elementos: valores, prácticas, roles, reuniones, fases, artefactos y herramientas. [34]

Prácticas

DAV utiliza prácticas tanto de SCRUM como de XGD.

- **Equipo completo.** El Equipo DAV debe permanecer en comunicación coherente, ser un todo.[34]

- **Diseño incremental.** Las tareas de un videojuego deben ser de la forma más sencilla posible, que funcionen correctamente.[34]

- **Historias de usuario.** Se trata de una breve descripción de las funcionalidades del videojuego, por lo general descrito por el Dueño del Producto.[34]

- **Ciclos semanales.** El proyecto está organizado y planificado para ser ejecutado en ciclos de corta duración.[34]

- **Integración continua.** En el proyecto existirán trozos de código fuente que continuamente se integrarán en funcionamiento.[34]

- **Código compartido.** Todo el Equipo DAV compartirá el código fuente del proyecto. [34]

- **Reuniones diarias de seguimiento.** Reuniones rápidas del proyecto, con el objetivo de que todo el Equipo DAV esté consciente de la labor que se está realizando en un momento dado. En la mayoría de estas reuniones, gran parte de los participantes simplemente escuchan, sin tener mucho que aportar. [34]

- **Entregas pequeñas.** Producir rápidamente versiones del videojuego que sean jugables, aunque no cuenten con toda la funcionalidad del videojuego. Esta versión ya constituirá un resultado de valor para el negocio. [34]

- **Pruebas.** El videojuego será probado en varios momentos por diferentes personas, dando como resultado una información valiosa que podrá ser tomada como base para decisiones futuras. [34]

- **Refactorización.** Es una actividad constante de reestructuración del código con el objetivo de evitar duplicaciones, mejorar la legibilidad, simplificarlo y hacerlo más flexible para facilitar posteriores cambios. [34]

- **Desarrollo en parejas (Cuando el Equipo Dav sea grande).** Toda la producción del videojuego debe realizarse con trabajo en parejas, pero simplemente los programadores y diseñadores. Esto conlleva ventajas implícitas (menor tasa de errores, mejor diseño y mayor satisfacción del equipo). [34]

- **Contacto directo.** Los clientes se convierten en parte del equipo de desarrollo, de esta manera existe comunicación cara a cara. [34]

- **Planes de riesgo y Mitigación.** Se desarrollan planes de riesgos y mitigación frecuentes por parte del Equipo DAV, la mitigación de riesgos, la monitorización y la gestión de riesgos se lleva a cabo en todas las etapas y con compromiso. [34]

- **Transparencia.** La transparencia en la planificación y desarrollo de módulos, permitirá a cada uno saber quién es responsable de qué y cuándo. [34]

- **Frecuentes reuniones.** Las frecuentes reuniones permite a las personas involucradas en el negocio monitorizar el progreso. [34]

- **Planificación ágil.** La planificación debe ser distribuida a lo largo del proyecto y no se concentrará en una determinada etapa. Las funciones más importantes serán planeadas con más detalle. Se debe evitar la planificación y estimación de las cosas que pueden no ser parte del videojuego. [34]

- **Establecimiento de prioridades.** A lo largo del proyecto se llevan a cabo varias priorizaciones, que permiten al Equipo DAV trabajar siempre en lo que es más importante. [34]

- **Estimación de Póquer.** Es una práctica ágil para reducir las dificultades habituales en las reuniones de trabajo para la planificación por juicio de expertos. En estas reuniones los participantes emplean un juego de cartas para concretar las unidades de esfuerzo que estiman para cada tarea. [34]

Roles

Los roles de DAV se basan únicamente en los de la metodología Scrum, además se incluye en el Equipo DAV el rol de Tester, siendo estos necesarios y suficientes para ser aplicados en equipos pequeños. [34]

- **Dueño del producto.** El Dueño del Producto es la única persona autorizada para decidir sobre cuáles funcionalidades y características funcionales tendrá el videojuego. Es quien representa al cliente y todas aquellas partes interesadas en el videojuego, además es quien maneja y prioriza las funcionalidades a desarrollar y las coloca en la Reserva de Producto. [34]
- **DAV Máster.** El DAV Máster no es un líder típico, sino es un auténtico servidor neutral, que será el encargado de enseñar la metodología DAV a cada integrante implicado en el proyecto, preocupándose de poner la metodología en práctica de modo que se encuentre dentro de la cultura de la organización y así entregue las ventajas previstas, asegurándose de que cada uno siga las reglas y prácticas de DAV. [34]
- **Equipo DAV.** Es un equipo multidisciplinario y auto-organizado, su función principal es construir el videojuego que el Dueño del Producto especifica. [34]
  - **Diseñador.** Realiza el diseño conceptual, define las reglas y mecánicas, escribe el guión y diseña los niveles del videojuego. [34]
  - **Programador.** Implementa la funcionalidad pedida en el diseño, programa la lógica del juego, gráficos, IA y audio y maneja herramientas/motores (infraestructura). [34]
  - **Artista.** Construye los objetos, personajes, niveles y anima a los personajes. [34]
  - **Sonidista.** Crea los sonidos, estilos, efectos y ambientes sonoros para el videojuego. [34]
  - **Tester.** Verifica el contenido y evalúa la jugabilidad. [34]
- **Interesados (Clientes).** Se refiere a la gente que hace posible el proyecto y para quienes el videojuego producirá el beneficio acordado que justifica su producción. Sólo participan directamente durante las revisiones de Iteración. [34]

Fase Pre-juego

En esta fase se realiza la concepción de la idea del juego del Dueño del Producto con ayuda de los miembros del Equipo, es decir, los aspectos fundamentales que conformarán el videojuego. [34]

- **Género.** Se debe especificar el género o géneros al que pertenece el videojuego para establecer las características básicas que tendrá en su posterior diseño. [34]

- **Historia.** Se debe realizar un bosquejo de la trama o historia del videojuego a desarrollar, indicando qué se quiere contar y cómo se quiere contar. [34]
- **Bocetos.** Se crean bocetos o diseños preliminares de los personajes y de dónde trascurrirá la acción del videojuego, ya sean decorados, ambientaciones, ropaje, música, movimientos, etc. [34]
- **Aspecto.** A partir de los bocetos se define el aspecto gráfico y artístico del videojuego, colores, temas dominantes, musicalidad, técnicas de diseño 3D o 2D, posiciones de cámaras, etc. [34]
- **Interfaz de usuario.** Se define la manera de cómo interactuará el jugador con el videojuego y con qué mecanismos contará para ello. [34]
- **Objetivos.** Cuáles son las metas del videojuego, de acuerdo a la historia de éste. [34]
- **Reglas.** Qué cosas podemos hacer y cómo vamos a dejar que se hagan. [34]
- **Características.** Especificación de las principales características de cada personaje del videojuego y de los elementos que intervienen en éste. [34]
- **Jugabilidad.** Es aquí donde se definirá cómo se va a jugar, de qué manera se va a jugar, qué cosas podemos hacer en el juego y cómo va reaccionar el entorno del juego a las acciones del jugador a través del personaje. A su vez se debe establecer cómo será la curva de aprendizaje del jugador. Todo esto sin entrar en detalles gráficos, sonoros o de historia. [34]
- **Diseño de niveles.** Describir qué niveles, según la historia o dificultad, se tendrá, cómo serán éstos, cuántos serán, y qué dificultad y retos se plantearán en cada uno de ellos. [34]
- **Requerimientos técnicos.** Establecer los requerimientos técnicos de equipo que necesitará el videojuego para poder ejecutarse. [34]
- **Marketing.** Parte fundamental, debido a que muchos videojuegos de inversiones millonarias han ido al traste por una mala campaña de publicidad, para evitar esto hay que establecer las líneas de publicidad para el videojuego. [34]

Fase Juego

En esta fase se tiene bien definido el alcance del proyecto, por lo tanto ya existe una idea clara de lo que realmente es el videojuego y lo que se debe hacer. [34]

- **Iteraciones.** Esta fase que suele llamarse Iteración, consta de ciclos que pueden durar de 1 a 4 semanas (por lo general 1-2 semanas), donde se desarrollan las funcionalidades del videojuego, su duración debe ser constante en cada ciclo. Sin embargo, este requisito no es estricto debido a que a veces el tiempo de cada Iteración es diferente. Una iteración cumple los siguientes pasos [34]

- **Elaborar.** Se desarrollan las funcionalidades o requisitos del videojuego que se encuentran en la Reserva de Iteración. [34]
- **Integrar.** Las funcionalidades desarrolladas son continuamente integradas en la nueva versión del videojuego. [34]
- **Revisar.** Se examinan las funcionalidades desarrolladas en la Iteración para corregir posibles errores y probar que su funcionalidad sea correcta. [34]
- **Ajustar.** Se adecua las funcionalidades del videojuego desarrolladas encajándolas correctamente, además se realiza la refactorización del código de las mismas. [34]

Fase post-juego

Esta fase comienza cuando el equipo DAV ha desarrollado con éxito todas las Historias de Usuario y el Dueño del Producto ya no tiene más funcionalidades para implementar. Aquí se realiza el cierre del proyecto, donde se prepara la liberación del videojuego, se verifican las versiones a entregar, se genera la documentación final y se realiza el pre-lanzamiento y el lanzamiento. [34]

Artefactos

En el caso de los artefactos, se han constituido en base a una selección mayoritaria de los artefactos de Scrum complementados con los de XGD, para mantener organizado el proyecto. Estos artefactos, ayudan a planificar y revisar cada uno de las Iteraciones, aportando medios ineludibles para efectuar cada una de las reuniones, los mismos se detallan a continuación [34]:

- **Historia de usuario.** Es la definición de un requisito (normalmente una funcionalidad) que el Dueño del Producto quiere, con total claridad y sin ninguna ambigüedad, la cual el Equipo DAV deberá desarrollar como parte de una Iteración. Son el resultado de escuchar al Dueño del Producto y ayudarle a resumir el requisito en una sola frase. Algo muy importante es que están escritas con el vocabulario del cliente, no con vocabulario técnico. [34]

ID	Como	Quiero.../Quiero que	De modo que.../Para
HU01	Jugador	Visualizar las instrucciones del juego	Saber cómo jugar
HU02	Jugador	La pala se encuentre en la parte inferior y pueda moverse en forma horizontal	La bola no se caiga y revotaría para romper los bloques
HU03	Jugador	Al romper los bloques con la bola se escuche un sonido de "golpe o ruptura"	Sea divertido y le dé realismo

Cuadro 3.2: Historias de usuario de DAV

Elaborado por: Carlos Acuña

Fuente: [34]

- **Prueba de aceptación.** Validan el grado de satisfacción del Dueño del Producto, es decir confirman que una Historia de Usuario ha sido implementada correctamente. Este tipo de pruebas son comúnmente realizadas por el Dueño del Producto y supervisadas por el Tester, informando de todas las deficiencias o errores que se encuentre antes de dar por aprobado el requisito definitivamente.

Prueba de tarea	
Identificador: PA001	**Historia de Usuario (Nro. y Nombre):** HU001 - Visualizar las instrucciones del videojuego
Nombre: Visualizar las instrucciones de usuario	
Descripción: Visualizar en una ventana las instrucciones del videojuego, además mediante un botón de color amarillo debe permitir regresar al menú principal	
Condiciones de ejecución: Debe ejecutarse en el dispositivo móvil, además debe estar terminado el menú inicio	
Entrada/Pasos de ejecución: Escoger la opción "Instrucciones" en el menú principal. En la pantalla que aparece las instrucciones del videojuego, para regresar al menú principal dar clic en el botón regresar.	
Resultado esperado: Mostrar las instrucciones del videojuego en una ventana, además tener un botón que permite salir de la ventana y regresar al menú principal	
Evaluación de la prueba: Correcto	

Cuadro 3.3: Prueba de aceptación de DAV

Elaborado por: Carlos Acuña
Fuente: [34]

  - **Identificador:** Es un identificador único de la Prueba de Aceptación para futuras referencias.
  - **Historia de Usuario(Nro. Y Nombre):** El identificador y nombre de la Historia de Usuario a la cual pertenece la Prueba de Aceptación.
  - **Nombre:** La Prueba de Aceptación debe tener un nombre entendible para cualquier persona, facilitando la comprensión tanto del propósito de la prueba como del campo de aplicación.
  - **Descripción:** Contiene una breve descripción del propósito de la prueba y la funcionalidad que examina.
  - **Condiciones de Ejecución:** Contiene información acerca de hardware o software necesario para poder ejecutar la prueba.
  - **Entrada/Pasos de ejecución:** Pasos a realizar para completar la prueba.

- **Resultado esperado:** Contiene una descripción de lo que el analista debería ver tras haber completado todos los pasos de la prueba.
- **Evaluación de la prueba:** Indica el resultado cualitativo de la ejecución de la prueba, a menudo con un Correcto/Fallido.

- **Reserva del producto.** La Reserva del Producto es un listado de alto nivel, dinámico y públicamente visible para todos los involucrados en el proyecto. En ella, el Dueño de Producto, mantiene una lista actualizada de requerimientos funcionales para el videojuego. Esta lista, representa "qué es lo que se pretende hacer" pero sin mencionar "cómo hacerlo", debido a que esta última, será tarea del Equipo DAV.
  La Reserva del Producto es creada y modificada únicamente por el Dueño de Producto. Durante la reunión de planificación, el Equipo DAV obtendrá los ítems del videojuego, que deberá desarrollar durante la Iteración. En la siguiente figura se muestra el formato de la Reserva del Producto.

P	Nro. HU	Ítem	E	Estado	PA
3	HU01	Como jugador quiero visualizar las instrucciones del juego, para saber cómo jugar	1	Pendiente	PA01
2	HU02	Como jugador quiero visualizar el puntaje de los bloques que voy rompiendo, de modo que pueda conocer cuantos puntos voy acumulando	1	Pendiente	PA02

Cuadro 3.4: Reserva de producto de DAV

Elaborado por: Carlos Acuña
Fuente: [34]

- **El grado de prioridad (PRI).** Los ítems de la Reserva del Producto, deben guardar un orden de prioridad, basado en los beneficios de implementar una funcionalidad y la pérdida o costo que demande posponer la implementación de una funcionalidad.[34]
- **Esfuerzo que demanda.** DAV, propone la estimación de esfuerzo y complejidad que demanda el desarrollo de las funcionalidades, solo para aquellas cuyo orden sea prioritario. Estas estimaciones, no se efectúan sobre ítems poco prioritarios ni tampoco sobre aquellos donde exista un alto grado de incertidumbre. De esta manera, se evita la pérdida de tiempo en estimaciones irrelevantes, postergándolas para el momento en el cual realmente sea necesario comenzar a desarrollarlas.[34]
- **Prueba de aceptación.** Es necesario especificar para cada ítem de la Reserva del Producto, la o las Pruebas de Aceptación que debe superar, para considerar cumplido el requisito.[34]

- **Tarjeta de tarea.** Las Tarjetas de Tarea se usan para describir las tareas que se realizarán en el proyecto y representar las responsabilidades asignadas a cada miembro del Equipo DAV. Estas tareas pueden ser de desarrollo, corrección o mejora, contienen número y nombre de la tarea, Historia de Usuario a desarrollar, fecha de inicio y fin de la tarea, se nombra al miembro del equipo responsable y presenta una descripción de la tarea. [34]

Tarjeta de tarea	
Número de tarea:	**Historia de Usuario (Nro. y Nombre):**
Nombre de Tarea:	
Tipo de tarea:	**Puntos estimados:**
Inicio:	**Fin:**
Miembro responsable:	
Descripción:	

Cuadro 3.5: Tarjeta de tarea de DAV

Elaborado por: Carlos Acuña
Fuente: [34]

3.1.5 Comparación de metodologías ágiles para el desarrollo de videojuegos

En el cuadro 3.6 se muestra que metodología ágil es mejor para la realización de videojuegos.

	SUM	**DAV**
Tolerante a cambios	Se debe crear nuevas tareas para arreglar problemas que ocurran durante iteraciones anteriores	Es susceptible a cambios aún cuando las tareas estén completadas y aprobadas
Tipo de orientación	Orientado a personas	Orientado a personas
Comunicación	La comunicación con el cliente es constante durante cada iteración	La comunicación con el cliente es constante durante cada iteración

Tiempo de entrega	Las entregas son frecuentes ya que consta de iteraciones que permite revisar el producto pero el tiempo es ajustable de acuerdo a la dificultad de cada iteración	Las entregas son frecuentes y tienen un tiempo límite de una a cuatro semanas sin importar la dificultad de la iteración
Documentación	Aunque existe documentación es limitada a teoría y pocos ejemplos	Existe documentación y una gran variedad de ejemplos

Cuadro 3.6: Cuadro comparativo de métodos ágiles

Elaborado por: Carlos Acuña

En base a los resultados obtenidos se puede determinar que la mejor metodología ágil es la metodología DAV debido a que especialmente cuenta con una gran cantidad de documentación con ejemplos y que el tiempo de la iteraciones es más fijo lo que hace que se tenga que realizar cada tarea diligentemente para poder realizar las correcciones de manera temprana sin la necesidad de crear nuevas tareas para corregir los problemas que puedan suceder durante el desarrollo del proyecto.

3.2 Implementación del videojuego educativo en 2D enfocado al aprendizaje de la cultura inca ecuatoriana

De acuerdo a los resultados de la sección 3.1.4 se ha seleccionado a la metodología DAV como la mejor opción para el desarrollo del proyecto de investigación.
A continuación se mostrará el paso a paso del desarrollo de la propuesta de videojuego de acuerdo a la metodología escogida.

3.2.1 Fase Pre-juego

La Fase Pre-juego se presenta como la parte de planificación del videojuego donde se debe dar una completa orientación de lo que va a hacer el juego empezando por dar una descripción, detallando aspectos fundamentales como el género, historia, bocetos, aspectos gráficos e interfaz de usuario. En el mismo apartado explicar las características de los elementos del videojuego así como objetivo, reglas y aspectos técnicos del juego.
Otra parte importante dentro de la fase pre-juego son los roles de equipo que determina que hará cada miembro dentro del proyecto; así también una pieza

fundamental de esta etapa son las historias de usuario donde se muestra el contenido y las funcionalidades que tendrá el videojuego desde el punto de vista del cliente y como punto final también se encuentra las fechas de entrega de cada iteración del juego.

Descripción del juego

El nombre del Juego es "Huaraca", el nombre está inspirado en un tipo de guerrero inca que usaba una honda como arma y fue esencial en las batallas del imperio inca, el modo de juego es un estilo de plataforma muy parecido a juegos como "Mario", "Metroid" y "Broforce". El mundo en cual está ambientado será totalmente 2D con pixel art y cinemáticas en cada nivel, tendrá una vista 2D que seguirá al personaje principal en su aventura.
Huaraca será el personaje principal y único personaje jugable. Existen personajes históricos que serán interpretados como jefes finales de ciertos niveles y personajes NPC(non playable character) que serán representados por guerreros y bestias mitológicas de la mitología inca. Huaraca contará con movimientos predefinidos para que el jugador pueda interactuar (correr, saltar, escalar, disparar).
En este juego no existe niveles de dificultad ya que mientras se siga avanzando por cada uno de los niveles la dificultad irá aumentando ya que se mostrará una mayor cantidad de obstáculos y enemigos que el jugador tendrá que vencer para llegar a la meta.

Aspectos Fundamentales

1. **Género**
   Este juego cae en el género de los juegos de plataforma, donde la coordinación mano ojo es muy importante porque nos permite realizar actividades en las que se utilizan simultáneamente los ojos y las manos para atravesar los diferentes obstáculos presentes en cada nivel.

2. **Historia**
   La historia tiene origen en el conflicto entre Atahualpa y Huascar por el control de los territorios del Tahuantisuyo.
   Se utilizará el recurso literario de la ficción histórica para poder narrar de manera didáctica la historia y que el juego tenga sentido. Así pues el personaje principal contará hechos verídicos y como afectaron a los territorios que hoy pertenecen al Ecuador.

3. **Bocetos**
   Los bocetos son dibujos preliminares de los personajes que van a participar dentro del juego, cabe destacar que estos diseños pueden sufrir cambios en la versión final pero sirven como referencia.

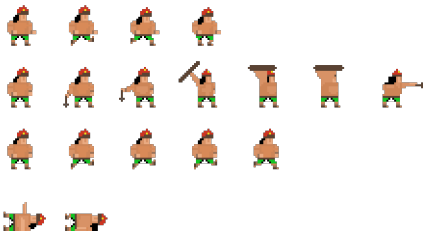

Figura 3.11: Sprites del personaje principal
Elaborado por: Carlos Acuña

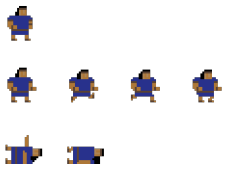

Figura 3.12: Sprites NPC (enemigo)
Elaborado por: Carlos Acuña

Figura 3.13: Sprites NPC (bestia mitológica inca)
Elaborado por: Carlos Acuña

Figura 3.14: Sprites NPC (Jefe de nivel)
Elaborado por: Carlos Acuña

4. **Aspectos gráficos**
   Es un juego totalmente en 2D que cuenta con una sola cámara que sigue al personaje principal, todos los objetos, personajes, terrenos y vistas están hechos en pixel art para dar al juego una imagen diferente y divertida.

5. **Interfaz de usuario**
   En este apartado el jugador podrá realizar las principales acciones que permita realizar el juego, el cual se divide en n interfaces.

   - **UI Logo.** Primera interfaz en mostrarse con el fin de presentar al creador del juego.
   - **UI Menu Principal.** Esta interfaz tendrá cuatro botones que son opciones que el jugador puede escoger como Comenzar Juego, escoger nivel, configuración y créditos.

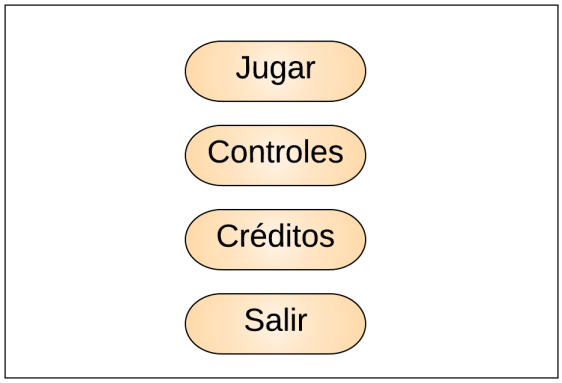

Figura 3.15: UI Menú Principal
Elaborado por: Carlos Acuña

   - **UI Juego.** Aquí se mostrará la cantidad de vidas y puntos acumulados que el jugador tenga en ese nivel, además de ser la pantalla principal donde se ejecutará todas las acciones del jugador.
   - **UI Cinemática.** Esta interfaz se muestra en cada nivel como introducción donde se contará la historia con texto e imágenes para entender el transfondo de lo que está pasando en el juego.

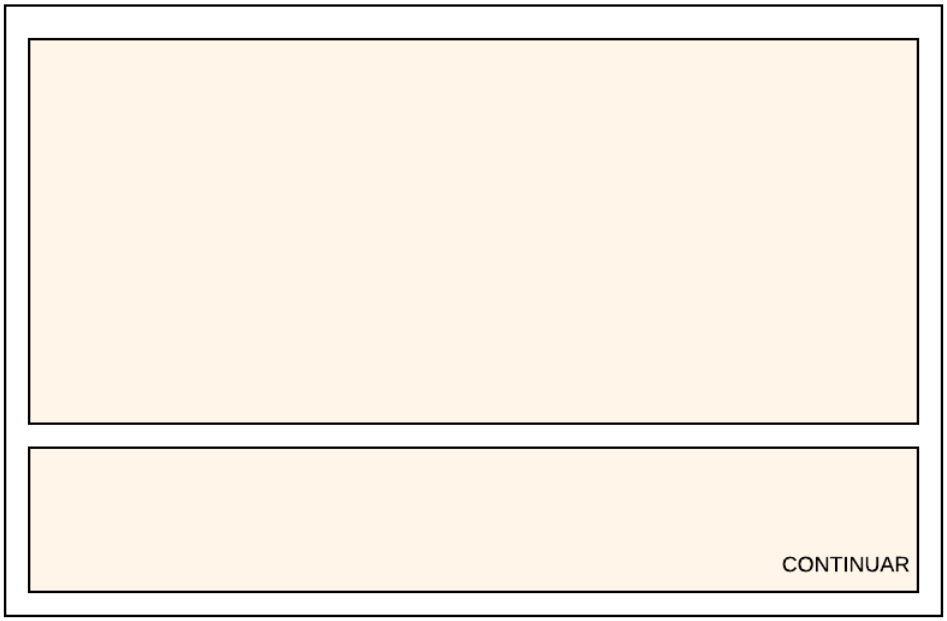

Figura 3.16: UI Cinemática
Elaborado por: Carlos Acuña

6. **Características de los elementos del videojuego**
   En el cuadro 3.7 se describe los elementos principales del juego.

Elementos	**Descripción**
Personaje Principal (Huaraca)	Se muestra al inicio de cada nivel donde realizará acciones de acuerdo a los comandos ejecutados por el jugador con el fin de llegar al final del nivel
NPC (Enemigos)	Se muestran en todos los niveles con el fin de obstaculizar el paso del personajes principal, existen varios enemigos con diferentes acciones y movimientos
NPC (Jefes)	Se muestran al final de ciertos niveles con el fin de obstaculizar el paso del personajes principal, existen varios enemigos con diferentes acciones y movimientos, además de tener una cantidad mayor de puntos de vida
Interactivo (Corazón)	Da al personaje principal una vida adicional
Interactivo (Monedas de oro)	Al acumular una cierta cantidad de monedas de oro se gana una vida adicional
Interactivo (Checkpoint)	Elemento que permite guardar la posición actual del jugador y regresará a este punto en caso de perder una vida.
Interactivo (Choza)	Elemento donde al llegar se da por finalizado el nivel y comienza el siguiente
Peligros (Pinchos)	Son elementos que dañan al personaje principal si cruza con alguno de ellos haciendo perder una vida
Interactivo (Niveles)	Cada nivel creado es más complicado que el anterior y en algunos niveles aparece un jefe para dificultar el avance del jugador

Cuadro 3.7: Características de los elementos del videojuego

Elaborado por: Carlos Acuña

7. **Objetivo del videojuego**

Enseñar sobre la cultura y mitología inca ecuatoriana de una manera divertida y amena a través de un videojuego que reta al jugador a completar niveles atravesando una serie de obstáculos que ponen a prueba sus habilidades y coordinación.
Cada nivel tiene como reto llegar del punto A al punto B con la mayor cantidad de vidas posibles en el menor tiempo posible para continuar con la aventura.

8. **Reglas**

   - Se inicia cada nivel con 1 vida(corazón) y cero monedas.
   - Cada nivel tiene enemigos y peligros que el jugador debe evitar o destruir para llegar al final del juego.
   - Al final de cada nivel el jugador debe llegar a una choza donde al ingresar por una puerta se acabará el nivel actual y comenzará el siguiente.
   - Si el jugador colisiona con un enemigo el jugador de inmediato muere y regresa al punto de partida o punto de chequeo (checkpoint).
   - Si el jugador colisiona con un peligro (pinchos) el jugador de inmediato muere y regresa al punto de partida o punto de chequeo (checkpoint).
   - Si el jugador pierde la misión la repite y todos los objetos obtenidos vuelven a su estado original, una vida (corazón) y 0 monedas.
   - El jugador puede disparar un proyectil que permite destruir a los enemigos.
   - Existen 3 subniveles por nivel y debe completarlos para continuar al siguiente nivel.

Aspectos técnicos

1. **Requerimientos técnicos**

Hardware	**Software**
Intel Core™ Duo o superior; OpenGL 3.0 compliant video card; 256 MB de RAM	Windows Vista o superior

Cuadro 3.8: Requisitos técnicos

Elaborado por: Carlos Acuña

2. **Arquitectura del videojuego**
   La arquitectura que se utiliza es MVC(Modelo Vista Controlador) que es modelo con el que se trabaja en Unity.

3. **Diseño conceptual**

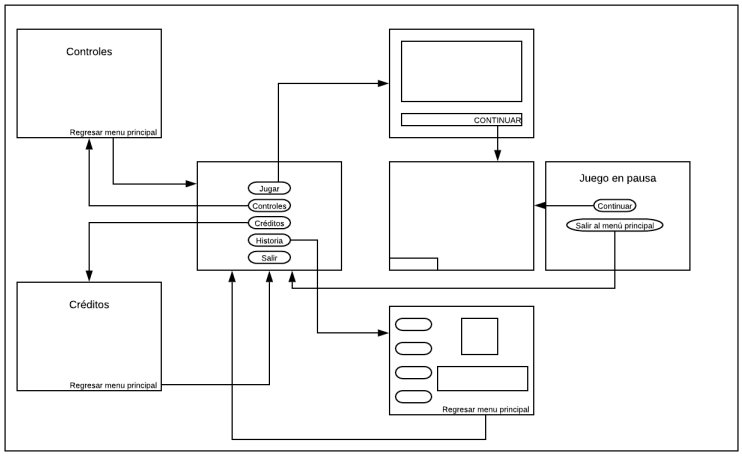

Figura 3.17: Concepto del juego
Elaborado por: Carlos Acuña

Herramientas a utilizarse

- **Diseño gráfico**
  - **GIMP.** GIMP es un acrónimo para (GNU Image Manipulation Program) o programa de manipulación de imágenes. Es un programa distribuido libremente para tareas como retoque de fotos, composición de imágenes y autorización de imágenes. GIMP tiene herramientas que se utilizan para el retoque y edición de imágenes, dibujo de formas libres, cambiar el tamaño, recortar, hacer fotomontajes, convertir a diferentes formatos de imagen, y otras tareas más especializadas. Se pueden también crear imágenes animadas en formato GIF e imágenes animadas en formato MPEG usando un plugin de animación. [35]
- **Desarrollo**
  - **Visual Studio Code.** Visual Studio Code es un editor de código optimizado con soporte para operaciones de desarrollo como depuración, ejecución de tareas y control de versiones. Su objetivo es proporcionar solo las herramientas que un desarrollador necesita para un ciclo rápido de creación de código-depuración y deja flujos de trabajo más complejos para IDE más completos, como Visual Studio IDE. [36]
  - **Unity3D.** Unity es un framework y motor de videojuegos multiplataforma creado por Unity Technologies. Unity está disponible como plataforma de desarrollo para Microsoft Windows, Mac OS, Linux y dispositivos móviles que permite la creación de videojuegos mediante la utilización de objetos que pueden ser importados hacia la herramienta, como son modelos 2D y 3D, archivos de texto, imágenes, sonidos, etc. Todo es hecho mediante varios lenguajes de programación como son BOO, C# y Javascript. MonoDevelop es el IDE con el que trabaja Unity además de usar API's para el uso de rutinas, estructura de datos y clases, estas

API's también son conocidas como librerías [27].
Unity es completamente libre para cualquiera o cualquier empresa que gane menos de $ 100000 al año, puede ser descargado sin ningún cargo adicional y sin tarjeta de crédito [28].

Roles de equipo

Persona	**Contacto**	**Rol**
Carlos Acuña	clsjs94@hotmail.com	DAV Master
		Dueño del producto
		Equipo DAV (Sonidista)
		Equipo DAV (Diseñador)
		Equipo DAV (Programador)

Cuadro 3.9: Roles

Elaborado por: Carlos Acuña

Historias de usuarios

Las historias de usuarios muestran el contenido y las funcionalidades que tendrá el videojuego "Huaraca".

ID	**Como**	**Quiero.../Quiero que**	**De modo que.../Para**
HU01	Jugador	Entrar, salir y ver los créditos del juego	Entrar, salir y ver los créditos del juego
HU02	Jugador	Antes de cada nivel conocer el transfondo histórico y los personajes principales	Motivar a jugar el juego y aprender sobre historia ecuatoriana
HU03	Jugador	Visualizar los controles del juego	Manejar los controles y mecánicas del juego
HU04	Jugador	Visualizar una pantalla de créditos	Conocer a los creadores del videojuego
HU05	Jugador	Visualizar una pantalla que cuente el trasfondo histórico del juego	Conocer la historia del juego
HU06	Jugador	Existan enemigos variados en cada nivel y puedan ser destruidos	Mejorar la experiencia de juego
HU07	Jugador	Al final de cada nivel exista una batalla contra un jefe	Aumentar la dificultad del juego
HU08	Jugador	Los niveles reflejen los paisajes andinos a través de pixel art	Mejorar el aspecto visual del juego
HU09	Jugador	Visualizar las vidas restantes y la puntuación actual	Motivar al jugador a cambiar de estrategia según sus opciones

HU10	Jugador	La cámara siga al jugador en todo momento	Tener una mejor visualización del juego
HU11	Jugador	El personaje principal cambie de animación según su acción	El juego se vea más orgánico y dinámico
HU12	Jugador	Visualizar la meta de cada nivel	Saber que se ha llegado al final de nivel del juego
HU13	Jugador	El personaje pierda el nivel cuando sus vidas lleguen a 0	Mantener la dificultad del juego
HU14	Diseñador	Los niveles tengan un fondo estilo pixel art	Mejorar la calidad visual del juego
HU15	Sonidista	Los efectos de sonido sean acordes a las acciones del jugador	El juego sea más divertido
HU16	Escritor	Usar el recurso literario de la fantasía histórica	Contar una historia entretenida pero basada en hechos reales
HU17	Diseñador	Agregar un story board en cada cambio de nivel	Tener un recurso visual de los eventos que están ocurriendo
HU18	Programador	El personaje realice diferentes acciones	Permitir la interacción y progresión del juego

Cuadro 3.10: Historias de usuario del juego

Elaborado por: Carlos Acuña

Pruebas de aceptación

Las pruebas de aceptación permiten validar las funcionalidades y características del videojuego una vez desarrolladas, de esta forma se garantiza que se cumpla con los requerimientos y pasaran a un estado de completado.

Prueba de tarea	
Identificador: PA01	**Historia de Usuario (Nro. y Nombre):** HU01 - Como jugador quiero entrar, salir y ver los créditos del juego
Nombre: Realizar el menú principal	
Descripción: Crear una interfaz con botones que permita entrar al juego y salir	
Condiciones de ejecución: Debe ejecutarse en cualquier tipo de pc, adaptarse a cualquier tipo de pantalla y funcionar correctamente	
Entrada/Pasos de ejecución: Presentar el menú principal con sus respectivos botones	
Resultado esperado: Acceder a las correspondientes interfaces según el botón seleccionado	

Cuadro 3.11: Prueba de aceptación PA01

Elaborado por: Carlos Acuña

Reserva del producto

En el cuadro 3.12 se observan las historias de usuario con sus respectivas pruebas de aceptación (PA), puntos de prioridad (P) y estimación (E) según el dueño del producto.

P	**Nro. HU**	**Ítem**	**E**	**Estado**	**PA**
3	HU01	Entrar, salir y ver los créditos del juego	1	Pendiente	PA01
2	HU02	Antes de cada nivel conocer el transfondo histórico y los personajes principales	1	Pendiente	PA02
3	HU03	Visualizar los controles del juego	1	Pendiente	PA03
3	HU04	Visualizar una pantalla de créditos	1	Pendiente	PA04
3	HU05	Visualizar una pantalla que cuente el trasfondo histórico del juego	1	Pendiente	PA05
2	HU06	Existan enemigos variados en cada nivel y puedan ser destruidos	1	Pendiente	PA06
3	HU07	Al final de cada nivel exista una batalla contra un jefe	1	Pendiente	PA07
1	HU08	Los niveles reflejen los paisajes andinos a través de pixel art	1	Pendiente	PA08
2	HU09	Visualizar las vidas restantes y la puntuación actual	1	Pendiente	PA09
1	HU10	La cámara siga al jugador en todo momento	1	Pendiente	PA10
1	HU11	El personaje principal cambie de animación según su acción	1	Pendiente	PA11
3	HU12	Visualizar la meta de cada nivel	1	Pendiente	PA12
2	HU13	El personaje pierda el nivel cuando sus vidas lleguen a 0	1	Pendiente	PA13
1	HU14	Los niveles tengan un fondo estilo pixel art	1	Pendiente	PA14

3	HU15	Los efectos de sonido sean acordes a las acciones del jugador	1	Pendiente	PA15
3	HU16	Usar el recurso literario de la fantasía histórica	1	Pendiente	PA16
3	HU17	Agregar un story board en cada cambio de nivel	1	Pendiente	PA17
1	HU18	El personaje realice diferentes acciones	1	Pendiente	PA18

Cuadro 3.12: Historias de usuario del juego

Elaborado por: Carlos Acuña

Determinación de las fechas de entrega

Toda la reserva del producto se ha dividido en 4 iteraciones o entregas, con una duración de 1 mes cada una, las fechas de inicio y fin así como también las fechas de entrega de cada iteración.

Iteración	**Fecha de inicio**	**Fecha de fin**	**Fecha de entrega**
1	18 de diciembre del 2019	17 de enero del 2020	18 de enero del 2020
2	19 de enero del 2020	17 de febrero del 2020	18 de febrero del 2020
3	19 de febrero del 2020	17 de marzo del 2020	18 de marzo del 2020

Cuadro 3.13: Fecha de entrega de cada iteración

Elaborado por: Carlos Acuña

3.2.2 Fase de juego

Una vez que se ha definido el alcance, las funcionalidades y un cronograma de las fechas de entrega de cada iteración del videojuego se empezará el desarrollo de cada una de las iteraciones.
En cada iteración se explica que historias de usuario se ha de implementar y se ha de comentar todos los pasos necesarios para llegar a cumplir los objetivos de esa etapa.
La duración de cada iteración será de 17 días es decir de lunes a sábado, los días lunes se realizarán las reuniones de planificación de iteración y las revisiones de las mismas se efectuarán los días domingos, de esta manera los otros días serán usados para desarrollar las funcionalidades de cada entrega.

Desarrollo de la iteración 1

El objetivo de esta iteración es obtener la primera versión del videojuego, la fecha de revisión de la iteración será el día 18 de enero de 2020, en esta iteración se desarrollará las siguientes historias de usuario.

ID	Descripción
HU08	Los niveles reflejen los paisajes andinos a través de pixel art
HU11	El personaje principal cambie de animación según su acción
HU19	El personaje realice diferentes acciones
HU10	La cámara siga al jugador en todo momento
HU14	Los niveles tengan un fondo estilo pixel art

Cuadro 3.14: Historias de usuario a implementarse en la Iteración 1

Elaborado por: Carlos Acuña

Se han generado tarjetas de tareas en base a las historias de usuarios seleccionadas para la iteración 1, las tareas fueron auto-asignadas al miembro del equipo DAV, el contenido de las tarjetas se encuentran en el Anexo A.
A continuación se resumen las tareas y subtareas realizadas para la iteración 1.

T01: Reflejar paisajes andinos mediante pixel art

Para la creación de los paisajes se debe utilizar sprites, que mediante la herramienta Tilemap de Unity se pueden crear niveles para juegos de plataforma como del presente proyecto. Usando la herramienta Tilemap Pallete se dibuja el terreno, escaleras, peligros y objetos de fondo que se necesite para lograr crear el mapa en el cual el personaje deberá interactuar.
En la figura 3.18 se muestra el resultado de usar el Tilemap para crear el primer nivel del juego.

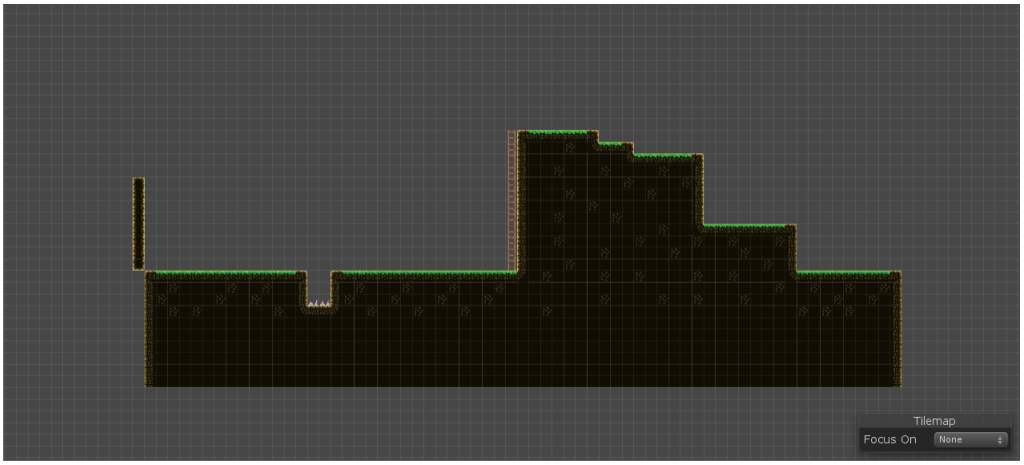

Figura 3.18: Terreno creado con Tilemap
Elaborado por: Carlos Acuña

Cada terreno creado con Tilemap es diferente y depende de la dificultad del nivel la extensión del mismo. Los sprites utilizados fueron creados con el fin de reflejar los terrenos de la cordillera de los Andes y toda la geografía que lo caracteriza pero en un estilo pixel art.

T02: El personaje principal cambie de animación según su acción

Con el uso del programa GIMP se puede dibujar al personajes principal, donde cada imagen es un sprite. Los sprites del personaje principal se unen en una sola imagen conocida como spritesheet la cual contiene todas las posibles acciones que puede ocurrir en el juego como son estar quieto, correr, saltar, disparar y morir. En la figura 3.19 se muestra al personaje principal con sus acciones de movimiento.

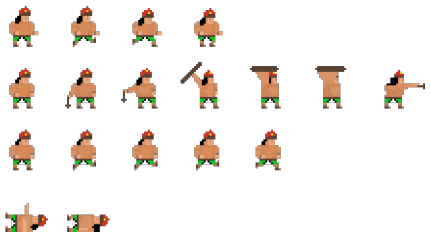

Figura 3.19: Sprites del personaje principal
Elaborado por: Carlos Acuña

Para que el personaje cambie de animación se utiliza la herramienta Animation y Animator de Unity con el fin de crear animaciones basados en los sprites creados. Se ha determinado 5 acciones acciones básicas del personaje principal como son estar quieto, correr, escalar, disparar y morir.
En la figura 3.20 se muestra la creación de las animaciones con los sprites creados y en la figura 3.21 se muestra la interacción de las animaciones de acuerdo al orden en que deben ser ejecutadas por cada acción que se realice.

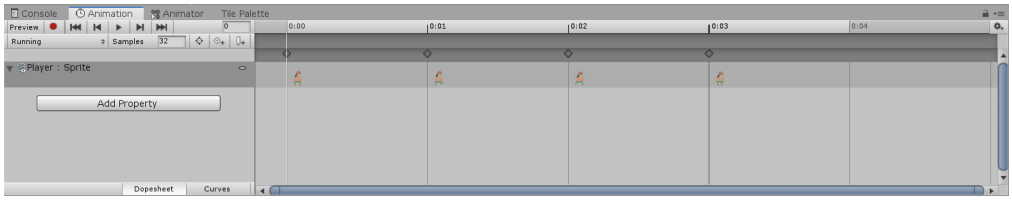

Figura 3.20: Uso de la herramienta Animation para crear animaciones del personaje principal
Elaborado por: Carlos Acuña

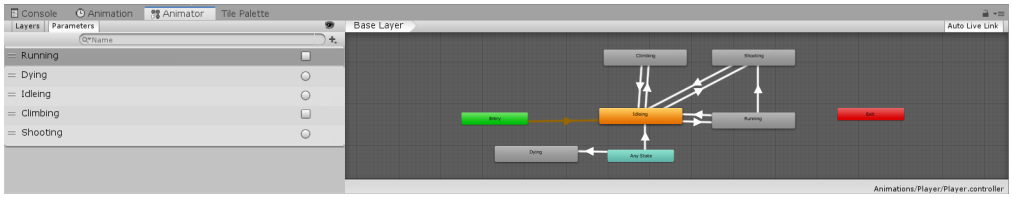

Figura 3.21: Uso de la herramienta Animator para ordenar las animaciones del personaje principal
Elaborado por: Carlos Acuña

T03: El personaje realice diferentes acciones

Dentro del videojuego se necesita que el personaje pueda interactuar con el terreno creado, para lo cual, acciones como correr, saltar, escalar y disparar son indispensables en este tipo de juego y son parte del objetivo de esta tarea. La clase que controlará todas estas acciones es la clase **"Player.cs"** junto con los métodos mostrados a continuación.

- **Correr.** La acción correr permite mover al personaje de izquierda a derecha y viceversa con el fin de que se mueva en todo el terreno hasta llegar a un punto concreto. El método encargado de esta acción se llama **"Run"**, el cual también se encarga de realizar la animación de correr al momento de llamar al método al presionar la tecla izquierda o derecha del teclado.

```
1  private void Run()
2    {
3        float controlThrow = Input.GetAxis("Horizontal");
4        Vector2 playerVelocity = new Vector2(controlThrow * runSpeed
             , myRigidbody.velocity.y);
5        myRigidbody.velocity = playerVelocity;
6
7        bool playerHasHorizontalSpeed = Mathf.Abs(myRigidbody.
             velocity.x) > Mathf.Epsilon;
8        myAnimator.SetBool("Running", playerHasHorizontalSpeed);
9    }
```

Figura 3.22: Método "Run"
Elaborado por: Carlos Acuña

Para que el personaje cambie de vista en el escenario dependiendo de que si ve hacia la derecha o izquierda se utiliza el método **"FlipSprite"** que permite esta acción al momento que el personaje se pone en movimiento dándole una visión más realista a la animación.

```
1  private void FlipSprite()
2    {
3        if (myRigidbody.velocity.x > 0f && !lookRight)
4        {
5            transform.Rotate(0f, 180f, 0f);
6            lookRight = true;
7        }
8        else if (myRigidbody.velocity.x < 0f && lookRight)
9        {
10           transform.Rotate(0f, 180f, 0f);
11           lookRight = false;
12       }
13   }
```

Figura 3.23: Método "FlipSprite"
Elaborado por: Carlos Acuña

- **Saltar.** La acción saltar permite mover al personaje principal de arriba hacia abajo simulando un salto en el cual el personaje detecta el terreno, si hay terreno puede ejecutar el salto caso contrario el jugador no podrá ejecutar más saltos hasta que el personaje toque el suelo nuevamente, para ejecutar esta acción se presiona la tecla arriba del teclado. Cabe destacar que la gravedad es importante para que la simulación del salto sea correcta por lo cual este parámetro se modifica en las opciones de proyectos de Unity. El método encargado de esta acción es el método **"Jump"**

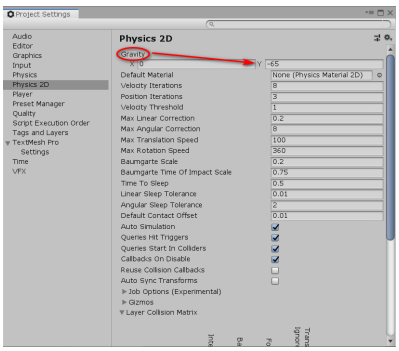

Figura 3.24: Uso de las opciones de proyecto para modificar la gravedad de los objetos del juego
Elaborado por: Carlos Acuña

```
1  private void Jump()
2    {
3        if (!myFeetCollider2D.IsTouchingLayers(LayerMask.GetMask("
             Foreground"))) { return; }
4
5        if (Input.GetButtonDown("Jump"))
6        {
7            Vector2 jumpVelocityToAdd = new Vector2(0f, jumpSpeed);
8            AudioSource.PlayClipAtPoint(jumpSFX, Camera.main.
                 transform.position);
9            myRigidbody.velocity += jumpVelocityToAdd;
10       }
11   }
```

Figura 3.25: Método "Jump"
Elaborado por: Carlos Acuña

- **Escalar.** La acción escalar permite simular la subida del personaje al momento de colisionar con el elemento del terreno conocido como escaleras, esto con el fin de alcanzar grandes alturas que no pueden ser atravesados con la función salto. Se utiliza la tecla arriba del teclado para realizar esta acción y debido a las colisión con el elemento escaleras se evita que el personaje se confunda con la acción de saltar. El método encargado de esta función es **"ClimbLadder"**.

```
1  private void ClimbLadder()
2    {
3        if (!myFeetCollider2D.IsTouchingLayers(LayerMask.GetMask("
             Climbing")))
4        {
5            myAnimator.SetBool("Climbing", false);
6            myRigidbody.gravityScale = gravityScaleAtStart;
7            return;
8        }
9
10       float controlThrow = Input.GetAxis("Vertical");
11       Vector2 climbVelocity = new Vector2(myRigidbody.velocity.x,
             controlThrow * climbSpeed);
12       myRigidbody.velocity = climbVelocity;
13       myRigidbody.gravityScale = 0f;
14
15       bool playerHasVerticalSpeed = Mathf.Abs(myRigidbody.velocity
             .y) > Mathf.Epsilon;
16       myAnimator.SetBool("Climbing", playerHasVerticalSpeed);
17   }
```

Figura 3.26: Método “ClimbLadder”
Elaborado por: Carlos Acuña

- **Disparar.** Como el nombre lo indica, esta acción permite la destrucción de los enemigos que aparezcan en cada nivel, tanto enemigos normales como jefes de nivel, esta acción consta de dos métodos. El primer método **"CanShoot"** que da un tiempo de espera entre cada disparo para que exista una saturación de proyectiles en pantalla al momento de realizar esta acción y el método **"Shoot"** que realiza la acción de disparar, junto con la animación del personaje que está disparando y la instanciación del proyectil que al momento de chocar con un enemigo, este será destruido y desaparecerá de la pantalla. Esta acción se realiza presionando la tecla "x" del teclado.

```
1  private bool CanShoot()
2    {
3        return Time.time > this.waitingShoot;
4    }
```

Figura 3.27: Método "CanShoot"
Elaborado por: Carlos Acuña

```
1  private void Shoot()
2    {
3        float fire = Input.GetAxis("Fire1");
4        if (fire != 0f && CanShoot())
5        {
6            myAnimator.SetTrigger("Shooting");
7            AudioSource.PlayClipAtPoint(throwSFX, Camera.main.
                 transform.position);
8            Instantiate(prefabProjectile, firePoint.position,
                 firePoint.rotation);
9            waitingShoot = Time.time + timingShoot;
10       }
11   }
```

Figura 3.28: Método "Shoot"
Elaborado por: Carlos Acuña

T04: La cámara sigue al jugador en todo momento

Aquí se busca que la pantalla siga los movimientos del personaje principal en todo el videojuego con el fin de que el jugador no se pierda al momento de realizar cualquier tipo de acción y que todo lo que ocurra cerca del personaje se vea de manera clara y concisa siempre centrado en el jugador. Para esto se ha de usar una herramienta de Unity conocida como **"Cinemachine"** que permite la manipulación de la cámara de manera fácil e intuitiva de manejar, adaptándose a las necesidades del juego.
Para crear una cámara de este tipo se necesita añadir esta herramienta al proyecto de Unity, luego se añadirá dos nuevas cámaras virtuales, una para cuando el personaje principal está quieto en la pantalla y otra cuando comience a realizar cualquier movimiento. Estas cámaras tendrán varios elementos que ayudarán a mejorar el seguimiento del jugador y dar una experiencia más pulida al momento de interactuar con el juego.

Figura 3.29: Ventana de jerarquía donde se encuentran las cámaras virtuales
Elaborado por: Carlos Acuña

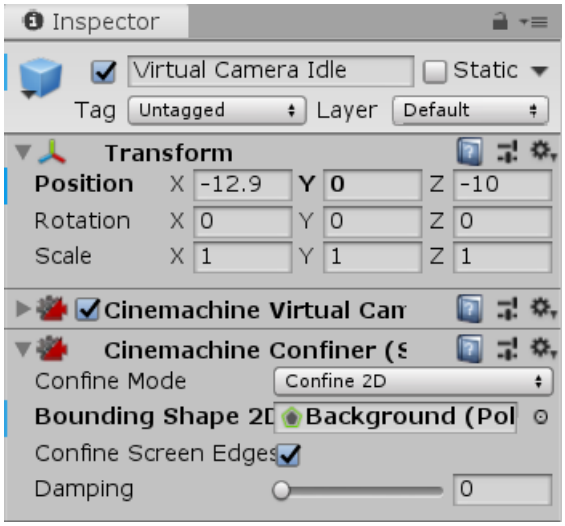

Figura 3.30: Ventana del Inspector de la cámara Idle
Elaborado por: Carlos Acuña

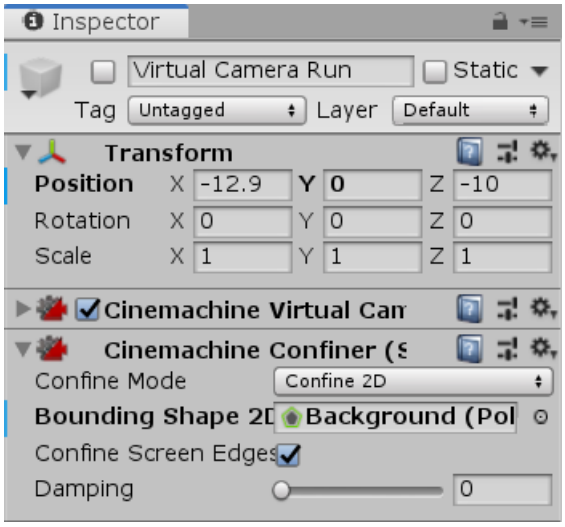

Figura 3.31: Ventana del Inspector de la cámara Run
Elaborado por: Carlos Acuña

T05: Los niveles tengan un fondo estilo pixel art

Para la creación de los fondos se ha recurrido a la búsqueda de escenarios que tengan relación con el paisaje andino, para lo cual se encontró unos fondos de pantalla creados específicamente para videojuegos, los cuales cumplen con las condiciones necesarias para adaptarse al estilo de juego que se está creando. Estos fondos de pantalla se encontraron en la siguiente página web: https://vnitti.itch.io/grassy-mountains-parallax-background de manera gratuita y para su uso libre en cualquier proyecto relacionado con videojuegos.

Figura 3.32: Previsualización de los fondos de pantalla en pixel art
Elaborado por: vnitti

Entrega funcional de la iteracion 1

La primera iteración para el 17 de enero del 2020 fue completada con éxito de acuerdo a la planificación que se había planeado.
Como resultado se obtuvo que el personaje principal está en pantalla sobre el terreno construido en pixel art. Las acciones correr, saltar, disparar y escalar son funcionales y es fondo de pantalla del juego representa lo más fielmente a la geografía andina. La figura 3.33 muestra el resultado final de la iteración 1.

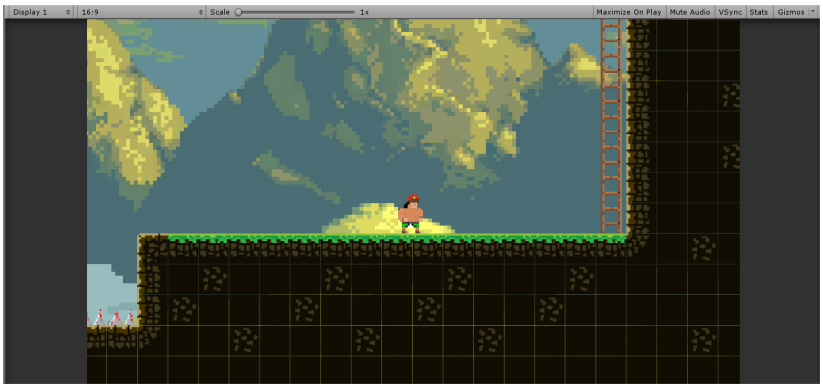

Figura 3.33: Resultado de la iteración 1
Elaborado por: Carlos Acuña

Desarrollo de la iteración 2

El objetivo de esta iteración es obtener la segunda versión del videojuego, la fecha de revisión de la iteración será el día 18 de febrero de 2020, en esta iteración se desarrollará las siguientes historias de usuario.

ID	Descripción
HU09	Visualizar las vidas restantes y la puntuación actual
HU13	El personaje pierda el nivel cuando sus vidas lleguen a 0
HU06	Existan enemigos variados en cada nivel y puedan ser destruidos
HU02	Antes de cada nivel conocer el transfondo histórico y los personajes principales

Cuadro 3.15: Historias de usuario a implementarse en la Iteración 2

Elaborado por: Carlos Acuña

Se han generado tarjetas de tareas en base a las historias de usuarios seleccionadas para la iteración 2, las tareas fueron auto-asignadas al miembro del equipo DAV, el contenido de las tarjetas se encuentran en el Anexo A.
A continuación se resumen las tareas y subtareas realizadas para la iteración 2.

T06: Visualizar las vidas restantes y la puntuación actual

Para visualizar las vidas y puntuación se debe tener en cuenta que se necesitan varias partes para que funcionen de manera correcta, estos elementos son:

- **Creación de la clase "LevelManager.cs".** La clase **"LevelManager.cs"** es la clase encargada de llevar los datos de salud y puntuación, guardar información importante del nivel, añadir las vidas extras y la cantidad de monedas de oro y controlar la acción de muerte del personaje principal.
  Los primeros métodos que se crean en esta clase son **"AddLives"** que permiten añadir una vida adicional al personaje principal y el método **"AddCoins"**que permite que se añada también una vida adicional cada conteo de 100 monedas de oro, este último método funciona conjuntamente con el método **"Update"** de la clase.

```
1  public void AddLives(int livesToAdd)
2  {
3          currentLives += livesToAdd;
4          livesText.text = "x " + currentLives;
5  }
```

Figura 3.34: Método “AddLives”
Elaborado por: Carlos Acuña

```
1  public void AddCoins(int coinToAdd)
2  {
3          coinCount += coinToAdd;
4          coinBonusLifeCount += coinToAdd;
5          coinText.text = "x " + coinCount.ToString();
6  }
```

Figura 3.35: Método “AddCoins”
Elaborado por: Carlos Acuña

```
1  void Update()
2  {
3          if (coinBonusLifeCount >= 100)
4          {
5              currentLives += 1;
6              livesText.text = "x " + currentLives;
7              coinBonusLifeCount -= 100;
8          }
9  }
```

Figura 3.36: Método “Update” para aumentar el número
de vidas por conteo de monedas de oro
Elaborado por: Carlos Acuña

- **Utilización de PlayerPrefs.** Para que se pueda guardar los datos relevantes sobre el juego se necesita usar la herramienta **"PlayerPrefs"** de Unity que permite guardar elementos como números y letras y otro tipo de información, estos datos permiten que funcionen correctamente otras mecánicas del juego. Estos datos se llaman desde el método **"Start"** que no solo funcionan en el nivel actual, sino que estos datos están presenten en todo el videojuego, todo se reinicia al momento de iniciar una nueva partida.

```
1  void Start()
2  {
3          player = FindObjectOfType<Player>();
4
5          if (PlayerPrefs.HasKey("coinCount"))
6          {
7              coinCount = PlayerPrefs.GetInt("coinCount");
8          }
9
10         coinText.text = "x " + coinCount.ToString();
11
12         if (PlayerPrefs.HasKey("playerLives"))
13         {
14             currentLives = PlayerPrefs.GetInt("playerLives");
15         }
16         else
17         {
18             currentLives = startingLives;
19         }
20
21         livesText.text = "x " + currentLives;
22 }
```

Figura 3.37: Método "Start' con los componentes "PlayerPrefs" para guardar información
Elaborado por: Carlos Acuña

Figura 3.38: Imagen del juego con los datos de vida y monedas de oro
Elaborado por: Carlos Acuña

- **Creación de los "GameObject" para vidas y monedas de oro.** Para que el personaje pueda ganar sus vidas y monedas dentro del juego, se necesita la creación de objetos que puedan ser usados para ese fin, es aquí donde se requiere el uso de **"GameObject"**. Estos objetos de juego son elementos que se encontrarán en la mayoría de niveles y ejecutarán eventos una vez que el jugador colisione con estos. Para que estos eventos funcionen se ha creado las clases **"ExtraLife.cs"** y **"Coin.cs"** con su respectivos métodos de colisión, el cual es **"OnTriggerEnter2D"**. Este método permite detectar colisiones del jugador y se ejecute el evento de añadir vidas o añadir monedas que se crearon en la clase **"LevelManager.cs"**.

```
1 private void OnTriggerEnter2D(Collider2D other)
2 {
3         if (other.tag == "Player")
4         {
5             AudioSource.PlayClipAtPoint(healthPickUpSFX, Camera.main
                  .transform.position);
6             levelManager.AddLives(livesToGive);
7             Destroy(gameObject);
8         }
9 }
```

Figura 3.39: Método "OnTriggerEnter2D" de la clase "ExtraLife.cs"
Elaborado por: Carlos Acuña

```
1 private void OnTriggerEnter2D(Collider2D other)
2 {
3
4         if (other.tag == "Player")
5         {
6             AudioSource.PlayClipAtPoint(coinPickUpSFX, Camera.main.
                  transform.position);
7             levelManager.AddCoins(coinValue);
8             Destroy(gameObject);
9         }
10 }
```

Figura 3.40: Método "OnTriggerEnter2D" de la clase "Coin.cs"
Elaborado por: Carlos Acuña

T07: El personaje pierde el nivel cuando sus vidas llegan a 0

En este apartado el objetivo es añadir dificultad al juego mediante el sistema de vidas creada en la tarea T06, al permitir que se pierda el nivel cuando las vidas del personaje principal llegue a 0. Para esto se debe tomar en cuenta que hay varias subtareas que realizar, así tenemos:

- **Creación de un Checkpoint.** Los checkpoints o puntos de control son elementos que permiten en un juego regresar a un estado anterior en un

punto concreto del mapa. Esto se justifica debido a que si pierde una vida, no necesariamente pierde el nivel, ya que el contador de vidas puede mayor a uno y por lo tanto no sería justo para el jugador volver a empezar todo de nuevo pero eso no quita la penalización por perder en esa parte del nivel.
El checkpoint funciona al momento en que el personaje principal choca con el **"GameObject"** conocido como checkpoint y se activa un evento mediante el método **"OnTriggerEnter2D"** de la clase **"Player"**. Este evento guarda la posición actual del personaje y tendrá una animación para confirmar que se ha realizado la acción mediante la clase **"Checkpoint"**.

```
1  private void OnTriggerEnter2D(Collider2D other)
2    {
3        if (other.tag == "Checkpoint")
4        {
5            respawnPosition = other.transform.position;
6        }
7    }
```

Figura 3.41: Método "OnTriggerEnter2D" de la clase "Player.cs"
Elaborado por: Carlos Acuña

```
1  private void OnTriggerEnter2D(Collider2D other)
2    {
3        if (other.tag == "Player")
4        {
5            checkpointActive = true;
6            fire.SetActive(true);
7        }
8    }
```

Figura 3.42: Método "OnTriggerEnter2D" de la clase "Checkpoint.cs"
Elaborado por: Carlos Acuña

- **Acción morir del personaje.** Para que el personaje principal pierda el nivel debe exponerse a varios peligros como al chocar con ciertos elementos del terreno que puedan dañarlo o al toparse con algunos enemigos que impidan su paso.
  Para esto se ha creado subrutinas que permiten la simulación de la acción morir del personaje principal dividiendo en dos partes, la primera se crea el método **"ProcessPlayerDeath"** que es cuando muere pero aún le quedan vidas de sobra y reaparece en el último checkpoint del mapa y el otro método es **"ProcessPlayerRestart"** cuando ya no le queda ninguna vida y se debe reiniciar totalmente el nivel. Estos métodos deben ser llamados en el método **"Respawn"** de la clase **"LevelManager"**.

```
public IEnumerator ProcessPlayerDeath()
  {
      player.GetComponent<Animator>().SetTrigger("Dying");
      player.isAlive = false;
      Instantiate(deathExplosion, player.transform.position,
          player.transform.rotation);
      yield return new WaitForSeconds(waitToRespawn);
      player.isAlive = true;
      coinCount = 0;
      coinText.text = "x " + coinCount.ToString();
      coinBonusLifeCount = 0;
      player.transform.position = player.respawnPosition;
      player.GetComponent<Animator>().SetTrigger("Idleing");
  }
```

Figura 3.43: Método "ProcessPlayerDeath" de la clase "LevelManager.cs"
Elaborado por: Carlos Acuña

```
public IEnumerator ProcessPlayerRestart()
  {
      player.gameObject.SetActive(false);
      gameOverScreen.SetActive(true);
      levelMusic.Stop();
      gameOverMusic.Play();
      yield return new WaitForSeconds(waitToRestart);
      var currentSceneIndex = SceneManager.GetActiveScene().
          buildIndex;
      SceneManager.LoadScene(currentSceneIndex);
  }
```

Figura 3.44: Método "ProcessPlayerRestart" de la clase "LevelManager.cs"
Elaborado por: Carlos Acuña

```
public void Respawn()
  {
      currentLives -= 1;
      livesText.text = "x " + currentLives;
      if (currentLives > 0)
      {
          StartCoroutine("ProcessPlayerDeath");
      }
      else
      {
          StartCoroutine("ProcessPlayerRestart");
      }
  }
```

Figura 3.45: Método "Respawn" de la clase "LevelManager.cs"
Elaborado por: Carlos Acuña

T08: Existan enemigos variados en cada nivel y puedan ser destruidos

En todo juego de plataformas debe existir la figura del enemigo con el fin de obstaculizar el progreso del personaje principal, en este apartado se creado varios tipos de enemigos, ya sean humanos o bestias mitológicas.
Primero se debe diseñar los modelos de los enemigos que serán usados para la animación de cada NPC y su respectiva clase para su funcionamiento.

- **Creación de los enemigos.** Para crear a los enemigos se los diseñó en la herramienta GIMP, basados en guerreros de la época y bestias mitológicas incas, a los cuales se les ha dado un diseño pixel art, así tenemos:

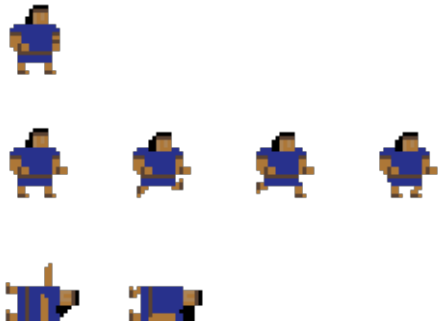

Figura 3.46: NPC (Guerrero)
Elaborado por: Carlos Acuña

Figura 3.47: NPC (Apallimay)
Elaborado por: Carlos Acuña

Figura 3.48: NPC (Amaru)
Elaborado por: Carlos Acuña

Figura 3.49: NPC (Jarjacha)
Elaborado por: Carlos Acuña

- **Animaciones de los enemigos.** Gracias a la herramienta Animator y Animation se puede simular varios movimientos para los enemigos con los sprites creados para este propósito.

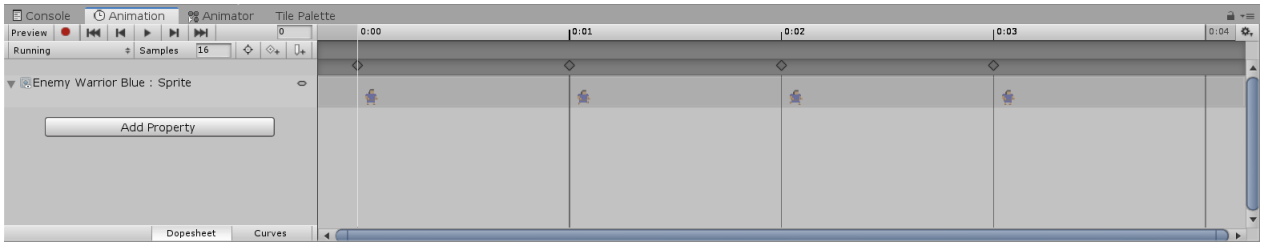

Figura 3.50: Uso de la herramienta Animation para el NPC (Guerrero)
Elaborado por: Carlos Acuña

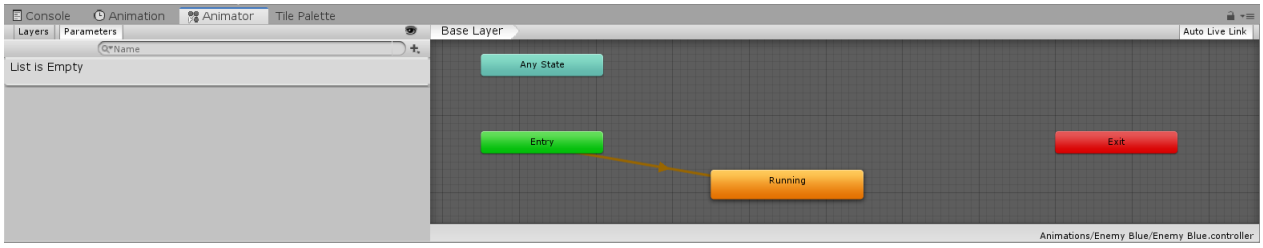

Figura 3.51: Uso de la herramienta Animator para el NPC (Guerrero)
Elaborado por: Carlos Acuña

- **Creación de la clase "Enemy.cs".** Para que cada enemigo tenga un comportamiento dentro del juego, se debe crear un script que cumpla esta función, esta clase se llamará "Enemy.cs", esta clase contendrá varios métodos para el correcto funcionamiento del NPC.

  - **Movimiento del personaje.** Para que el personaje se mueva de izquierda a derecha y viceversa como si estuviera patrullando se ha generado varios métodos que permiten esta acción. Asi tenemos el método **"IsFacingRight"** para mostrar al personaje mirando hacia el lado correcto de la pantalla, donde conjuntamente con el método **"Update"** se permite que realice el movimiento en esa dirección.

```
1 bool IsFacingRight()
2   {
3       return transform.localScale.x > 0;
4   }
```

Figura 3.52: Método "IsFacingRight"
Elaborado por: Carlos Acuña

```
1    void Update()
2    {
3        if (IsFacingRight())
4        {
5            myRigidBody.velocity = new Vector2(moveSpeed, 0f);
6        }
7        else
8        {
9            myRigidBody.velocity = new Vector2(-moveSpeed, 0f);
10       }
11   }
```

Figura 3.53: Método “Update” para el movimiento del enemigo
Elaborado por: Carlos Acuña

– **Vida y muerte del personaje.** Todo NPC enemigo tiene cierta cantidad de vida que cuando llega a 0 desaparece del mapa, para lo cual se ha creado dos métodos que permiten realizar esta acción. Así se obtiene el método **"TakeDamage"** y el método **"Die"** para cumplir esta tarea.

```
1    public void TakeDamage(int damage)
2    {
3        health -= damage;
4        if (health <= 0)
5        {
6            Die();
7        }
8    }
```

Figura 3.54: Método "TakeDamage"
Elaborado por: Carlos Acuña

```
1    private void Die()
2    {
3        AudioSource.PlayClipAtPoint(hurtSFX, Camera.main.transform.
             position);
4        Destroy(gameObject);
5    }
```

Figura 3.55: Método "Die"
Elaborado por: Carlos Acuña

- **Destrucción del enemigo.** Cada enemigo debe poder ser eliminado del juego luego de que el jugador realice la acción disparar y el proyectil choque con el NPC. Para esto se ha de añadir algunas reglas a la clase "Projectile.cs". Dentro del método **"OnTriggerEnter2D"** se creará un evento que detecte la colisión del proyectil con el enemigo y llame al método **"TakeDamage"** de la clase **"Enemy.cs"**.

```
1    private void OnTriggerEnter2D(Collider2D other)
2    {
3        Enemy enemy = other.GetComponent<Enemy>();
4        if (enemy != null)
5        {
6            enemy.TakeDamage(damage);
7            Instantiate(deathExplosion, transform.position,
                 transform.rotation);
8            Destroy(gameObject);
9        }
10   }
```

Figura 3.56: Método "OnTriggerEnter2D" de la clase "Projectile.cs"
Elaborado por: Carlos Acuña

T09: Antes de cada nivel conocer el trasfondo histórico y los personajes principales

Para tener un transfondo histórico se debe crear una cinemática que permita saber lo que está pasando en cada nivel, es así como se utiliza el recurso literario de la fantasía histórica para contar una historia que se basa en hechos reales. Para que esto funcione se debe crear una escena especial que contenga el storyboard y que sea fácil de entender para el jugador.
Para realizar esta actividad se debe crear una escena donde el jugador pueda leer una historieta que se cuenta en primera persona del héroe del juego conocido como Huaraca y que se entienda el trasfondo de la historia que sucede en ella. Para ello se deben realizar las siguientes actividades:

- **Creación del sistema de diálogos.** Este sistema permite simular una pantalla de diálogo donde el personaje principal cuenta su historia. Para esto se creado una clase llamada **"Dialogues.cs"**, donde el método **"Type"** crea un efecto estilizado al momento de mostrar el texto de la historia y además permite cambiar la imagen en cada cambio de frase y el método **"NextSentence"** que alterna entre el botón continuar y el botón que permite ir al siguiente nivel que está bajo el llamado del método **"NextLevel"**.

```
1    IEnumerator Type()
2    {
3        if (textFile != null)
4        {
5            sentences = textFile.text.Split('\n');
6            foreach (char letter in sentences[index].ToCharArray())
7            {
8                textDisplay.text += letter;
9                storyImage.GetComponent<Image>().sprite = sprites[
                     index];
10               myAudioSource.Play();
11               myAudioSource.volume = 0.1f;
12               yield return new WaitForSeconds(typingSpeed);
13           }
14       }
15   }
```

Figura 3.57: Método "Type"
Elaborado por: Carlos Acuña

```
1    public void NextSentence()
2    {
3        continueButton.SetActive(false);
4        if (index < sentences.Length - 1)
5        {
6            index++;
7            textDisplay.text = "";
8            StartCoroutine(Type());
9        }
10       else
11       {
12           textDisplay.text = "";
13           continueButton.SetActive(false);
14           nextLevelButton.SetActive(true);
15       }
16   }
```

Figura 3.58: Método "NextSentence"
Elaborado por: Carlos Acuña

```
1    public void NextLevel()
2    {
3        var currentSceneIndex = SceneManager.GetActiveScene().
             buildIndex;
4        SceneManager.LoadScene(currentSceneIndex + 1);
5    }
```

Figura 3.59: Método "NextLevel"
Elaborado por: Carlos Acuña

Figura 3.60: Cinemática de la historia dentro del videojuego
Elaborado por: Carlos Acuña

Entrega funcional de la iteracion 2

La segunda iteración para el 17 de febrero del 2020 fue completada con éxito de acuerdo a la planificación que se había planeado.
Como resultado se obtuvo que el personaje principal complete su funcionamiento con la acción morir y que las vidas y monedas de oro puedan ser contadas y guardadas en cada nivel. Que cada nivel tenga su contexto histórico y que sirva como medio para presentar el trasfondo del personaje.

Desarrollo de la iteración 3

El objetivo de esta iteración es obtener la tercera y última versión del videojuego, la fecha de revisión de la iteración será el día 18 de marzo de 2020, en esta iteración se desarrollará las siguientes historias de usuario.

ID	Descripción
HU12	Visualizar la meta de cada nivel
HU07	Al final de cada nivel exista una batalla contra un jefe
HU15	Los efectos de sonido sean acordes a las acciones del jugador
HU01	Entrar, salir y ver los créditos del juego
HU03	Visualizar los controles del juego
HU04	Visualizar una pantalla de créditos
HU05	Visualizar una pantalla que cuente el trasfondo histórico del juego
HU17	Agregar un story board en cada cambio de nivel

Cuadro 3.16: Historias de usuario a implementarse en la Iteración 3

Elaborado por: Carlos Acuña

T10: Visualizar la meta de cada nivel

Para visualizar la meta se deben realizar algunos pasos para cumplir con esta tarea:

- **Creación de la meta.** Se debe crear un **"GameObject"** para que pueda interactuar con el jugador y realice un evento dentro del videojuego, la clase que controla estas acciones es la clase **"LevelExit.cs"**.

- **Creación del método "LoadNextLevel".** Este método permite al jugador saber que terminó el nivel y que puede avanzar en su aventura, además de llevar la información importante como lo es la cantidad de vidas que le sobran al jugador y la cantidad de monedas de oro que tiene. Además genera una pequeña animación con música que permite saber al jugador que ha llegado a la meta. Para que se realice esta acción el jugador debe colisionar con el **"GameObject"** de la meta y así poder activar el método **"OnTriggerEnter2D"**.

```
1   IEnumerator LoadNextLevel()
2   {
3       Time.timeScale = LevelExitSlowMoFactor;
4       levelManager.levelMusic.Stop();
5       misionCompletedMusic.Play();
6       yield return new WaitForSecondsRealtime(LevelLoadDelay);
7       Time.timeScale = 1f;
8       PlayerPrefs.SetInt("coinCount", levelManager.coinCount);
9       PlayerPrefs.SetInt("playerLives", levelManager.currentLives)
            ;
10      PlayerPrefs.SetInt(levelToUnlock, 1);
11      var currentSceneIndex = SceneManager.GetActiveScene().
            buildIndex;
12      SceneManager.LoadScene(currentSceneIndex + 1);
13  }
```

Figura 3.61: Método "LoadNextLevel"
Elaborado por: Carlos Acuña

```
1   private void OnTriggerEnter2D(Collider2D other)
2   {
3       StartCoroutine(LoadNextLevel());
4   }
```

Figura 3.62: Método "OnTriggerEnter2D" de la clase "LevelExit.cs"
Elaborado por: Carlos Acuña

T11: Batalla contra jefe de cada nivel

En este apartado se ha añadido una batalla contra jefe final, este personaje es diferente del enemigo común ya que tiene más puntos de vida y que se mueve de manera diferente que un enemigo común. Estos personajes aparecen en algunos niveles y ocupan la misma clase **"Enemy.cs"**, pero se cambian algunos de sus parámetros.

T12: Los efectos de sonido sean acordes de las acciones del juego

Cada vez que se tiene realizar una acción un sonido debe ser utilizado para poder saber que está ocurriendo en el juego, esto se hace mediante un componente de

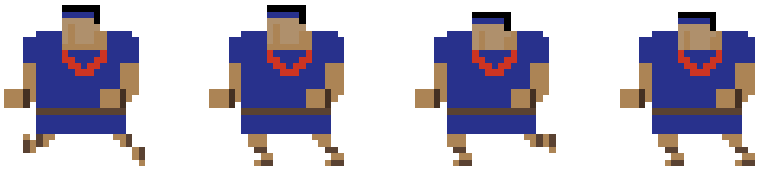

Figura 3.63: Chapera (Sprites)
Elaborado por: Carlos Acuña

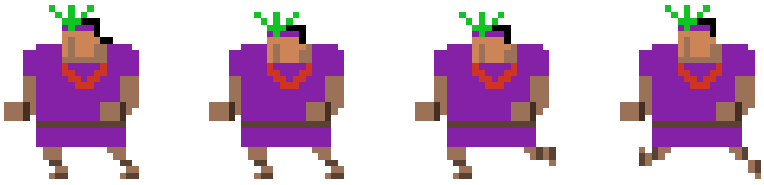

Figura 3.64: Apu Atoc (Sprites)
Elaborado por: Carlos Acuña

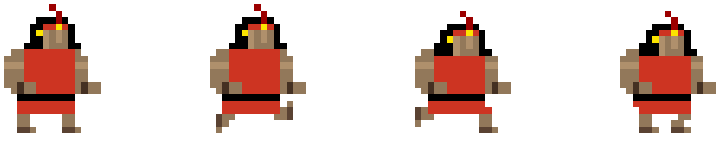

Figura 3.65: Huascar (Sprites)
Elaborado por: Carlos Acuña

Unity que es **"AudioSource"** que permite agregar sonidos a los objetos creados y puede ser activado dependiendo de la acción de cada objeto.

T13: Entrar salir y ver los créditos del juego

En esta actividad se crea el menú principal del juego, que permite que el jugador pueda realizar varias acciones relacionadas con la interfaz del videojuego, como jugar, controles, créditos, historia y salir. Estos botones se ven como la figura 3.66

Figura 3.66: Pantalla de menú principal
Elaborado por: Carlos Acuña

T14: Visualizar los controles del juego

En esta pantalla se puede observar los controles del juego mediante una pantalla que muestra al jugador los botones que debe usar para manejar al personaje principal.

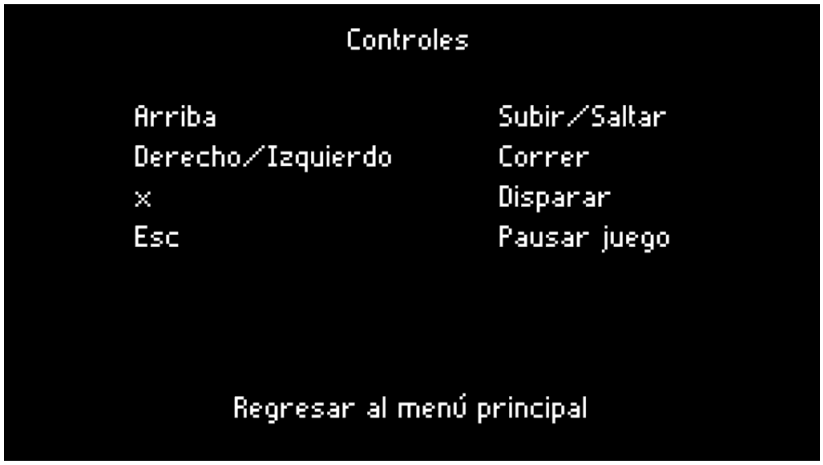

Figura 3.67: Pantalla de controles
Elaborado por: Carlos Acuña

T15: Visualizar los créditos del juego

En esta pantalla se observa los créditos de las personas que han creado el juego.

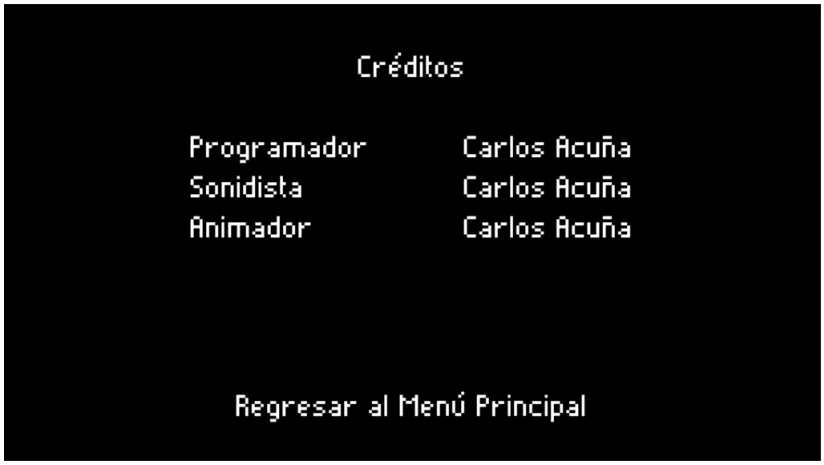

Figura 3.68: Pantalla de créditos
Elaborado por: Carlos Acuña

T16: Visualizar una pantalla que cuente el trasfondo histórico del juego

En esta parte del juego se puede interactuar con una interfaz que permite conocer la historia de cada personaje en la vida real, conocer sus triunfos y derrotas y todo lo relacionado con su vida. En la figura se ve la interfaz donde se muestra una imagen y su respectiva descripción.

Figura 3.69: Pantalla de historia
Elaborado por: Carlos Acuña

T17: Agregar un story board en cada cambio de nivel

El story board solo es las imágenes que se utilizan en la cinemática del videojuego que se puede apreciar en la tarea T09 donde las gráficas son de uso libre y que no exista problemas por derechos de autor.

Entrega funcional de la iteración 3

La primera iteración para el 17 de marzo del 2020 fue completada con éxito de acuerdo a la planificación que se había planeado.
Como resultado el juego está completamente terminado con todas las especificaciones basadas en las historias de usuarios y correcciones necesarias.

Figura 3.70: Capturas del juego “Huaraca”
Elaborado por: Carlos Acuña

3.2.3 Fase Post-Juego

Una vez finalizado con éxito y sin más historias de usuarios que completar se procede a liberar el juego en alguna plataforma virtual con el fin de llegar a una gran cantidad de personas y obtener una retroalimentación como críticas, calificaciones o consejos de todo tipo de personas interesadas en este videojuego.

Evaluación de la tienda online

Existen muchas tiendas onlines que permiten subir videojuegos de manera gratuita que apoya a desarrolladores independientes ha mostrar sus trabajos y distribuirlos en todo el mundo. De las cuales se ha tomado en cuenta dos tiendas online muy conocidas como son **"itch.io"** y **"Gamejolt"**. En el cuadro 3.17 se ha comparado varias características para poder escoger la mejor opción basado en los siguientes criterios.

- **Pago requerido.** Si es necesario pagar una cantidad para poder subir un videojuego a la plataforma.
- **Regalías.** Si la tienda online permite ganar dinero al desarrollador por la venta del videojuego.
- **Multiplataformas.** Permite subir el videojuego para cualquier tipo de plataforma disponible.
- **Estadísticas.** Nivel de estadísticas de la plataforma virtual para el desarrollador sobre cómo se está manejando el videojuego.

	itch.io	**Gamejolt**
Pago requerido	No se requiere ningún pago inicial	No se requiere ningún pago inicial
Regalías	La página permite ganar dinero al desarrollador con un aporte del 10% a la tienda virtual si el juego se vende.	La página permite ganar dinero al desarrollador donde la tienda virtual gana dinero mediante publicidad quedándose con el 70% de esa ganancia. No hay estas características para Ecuador.
Multiplataformas	Soporta una gran variedad de plataformas disponibles	Soporta una gran variedad de plataformas disponibles

Estadísticas	Muestra cantidad de vistas y descargas mensual y anualmente.	Muestra cantidad de vistas y descargas diario, mensual y anualmente, además de en que plataforma y de que país fue descargado o visto el juego.

Cuadro 3.17: Cuadro comparativo de tiendas online

Elaborado por: Carlos Acuña

Con los resultados obtenidos se ha decidido usar la plataforma Gamejolt porque tiene mejores estadísticas que la plataforma itch.io, lo que beneficia para saber mejor la distribución del juego y conocer mejor la opinión de la gente de este juego.

Publicación en la tienda virtual

Para publicar en **"Gamejolt"** se necesita una cuenta con algunas configuraciones para desarrollador y se lo hace de la siguiente manera.

- **Añadir juego.** Para añadir un juego se selecciona cualquiera de estos tipos de configuración dependiendo del avance del juego.

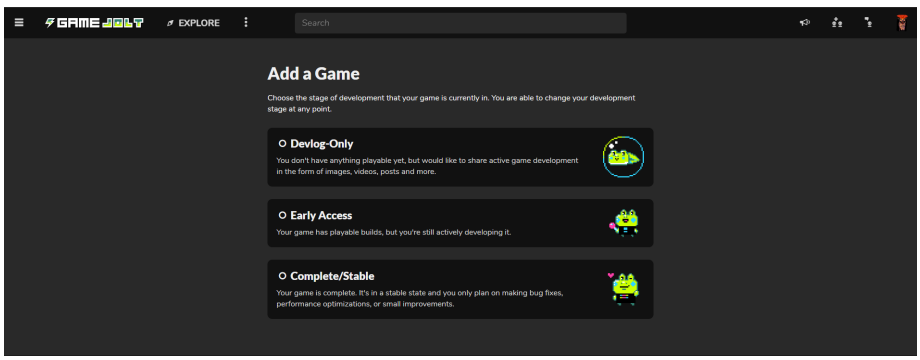

Figura 3.71: Sección de añadir juego de Gamejolt
Elaborado por: Carlos Acuña

- **Detalle.** En esta ventana se agrega datos básicos del juego.

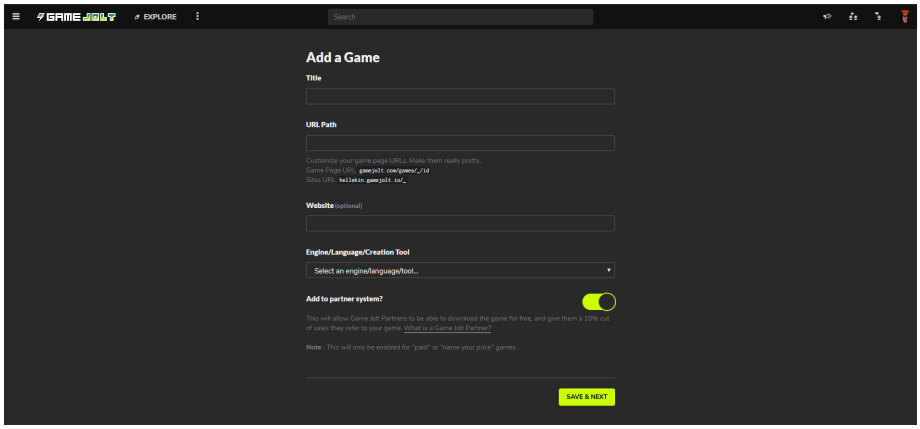

Figura 3.72: Sección de detalle de Gamejolt
Elaborado por: Carlos Acuña

- **Descripción.** En esta sección se agrega una descripción del videojuego.

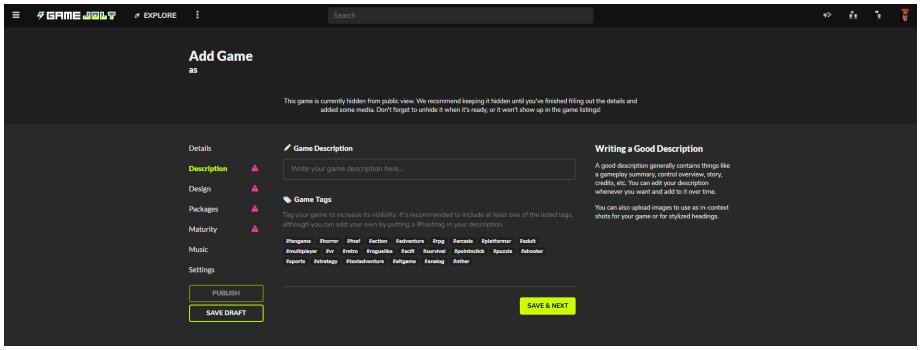

Figura 3.73: Sección de descripción de Gamejolt
Elaborado por: Carlos Acuña

- **Diseño.** En esta sección se agrega varios elementos multimedia para poder mostrar una mejor imagen del videojuego.

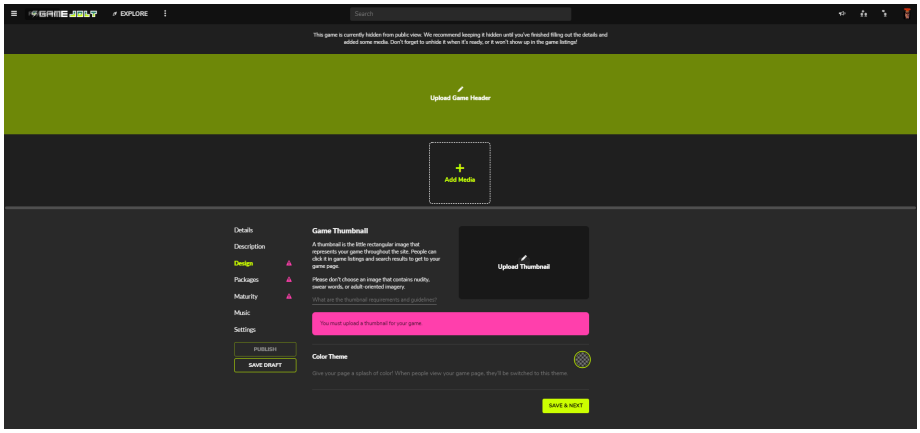

Figura 3.74: Sección de diseño de Gamejolt
Elaborado por: Carlos Acuña

- **Paquetes.** En esta sección se agrega los paquetes necesarios para que se pueda descargar el videojuego.

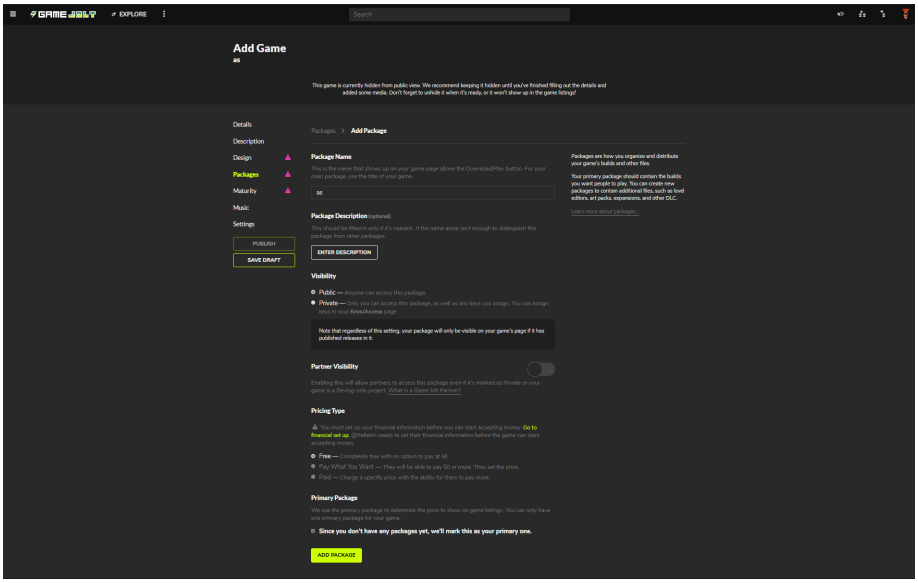

Figura 3.75: Sección de paquetes de Gamejolt
Elaborado por: Carlos Acuña

- **Madurez.** En esta sección se determina el nivel de madurez que debe tener el jugador para poder jugar el videojuego.

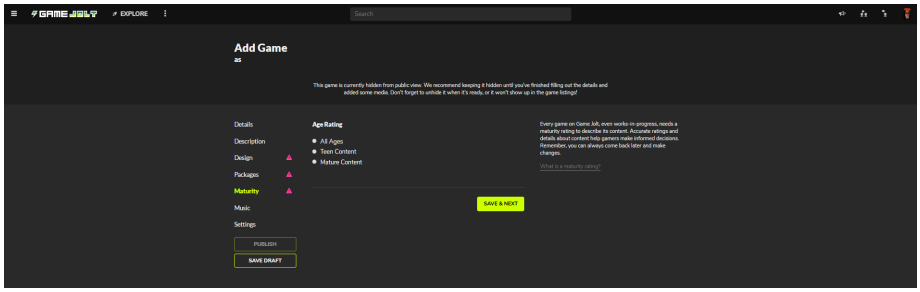

Figura 3.76: Sección de madurez de Gamejolt
Elaborado por: Carlos Acuña

- **Preferencia.** En esta sección se determinan algunas preferencias como permitir anuncios, comentarios y calificaciones.

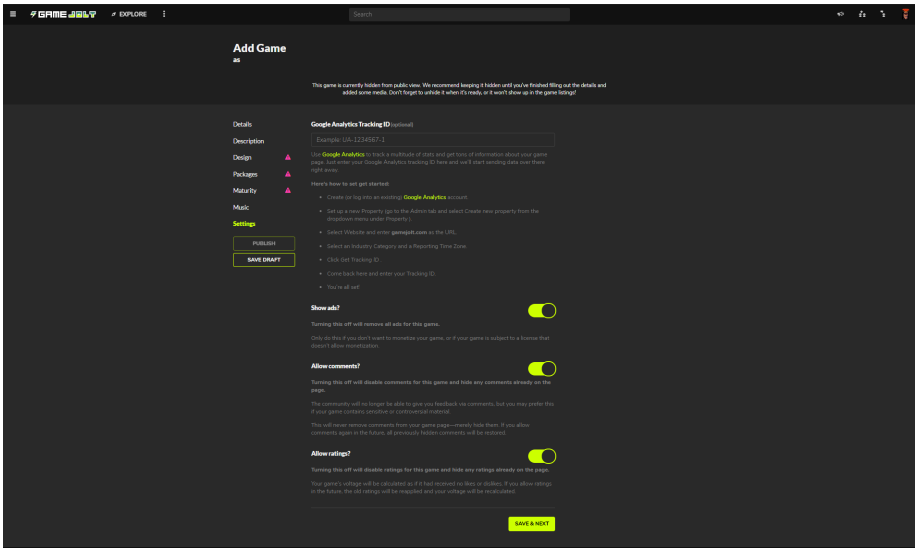

Figura 3.77: Sección de preferencias de Gamejolt
Elaborado por: Carlos Acuña

Al final se guarda todos los cambios y el juego se publica.

Análisis de los resultados de subir el juego a Gamejolt

El juego **"Huaraca"** fue publicado el 20 de marzo de 2020, un mes después se recopila los resultados obtenidos por el videojuego en la página de **"Gamejolt"** de los cuales se puede obtener el siguiente reporte personalizado.
En la figura 3.78 se muestra un reporte del total de vistas del juego, el número de descargas, el número de calificaciones y el total de seguidores que tiene el videojuego

Figura 3.78: Reporte general de Gamejolt
Elaborado por: Carlos Acuña

El total de vistas por país se muestra en la figura 3.79 donde Ecuador destaca con el 95,6% pero además se halló que hay vistas de otros países como Brasil y Egipto.

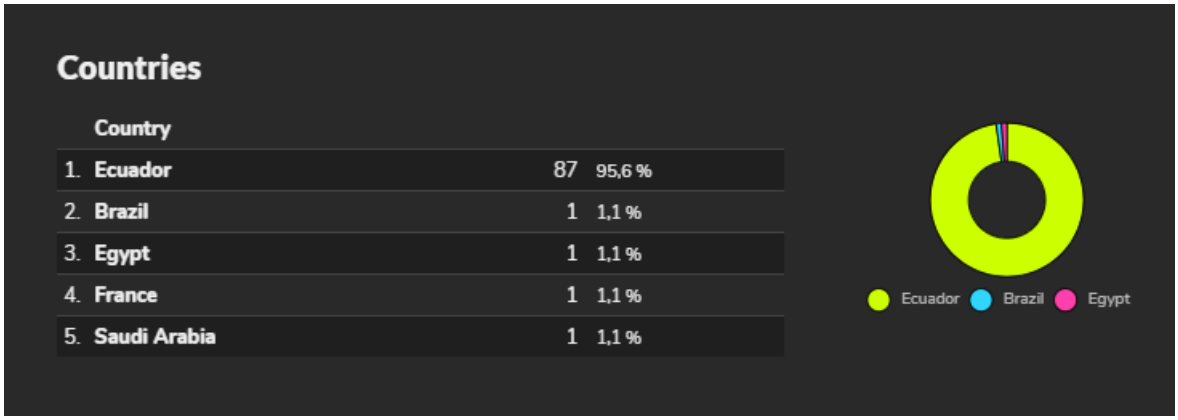

Figura 3.79: Vistas por país del videojuego
Elaborado por: Carlos Acuña

El número de descargas por país que se muestra en la figura 3.80 refleja que el juego tuvo gran aceptación dentro del territorio nacional con un total del 95,45% de las descargas pero además existe un 4,55% de descargar realizadas en otros países como es Chile.

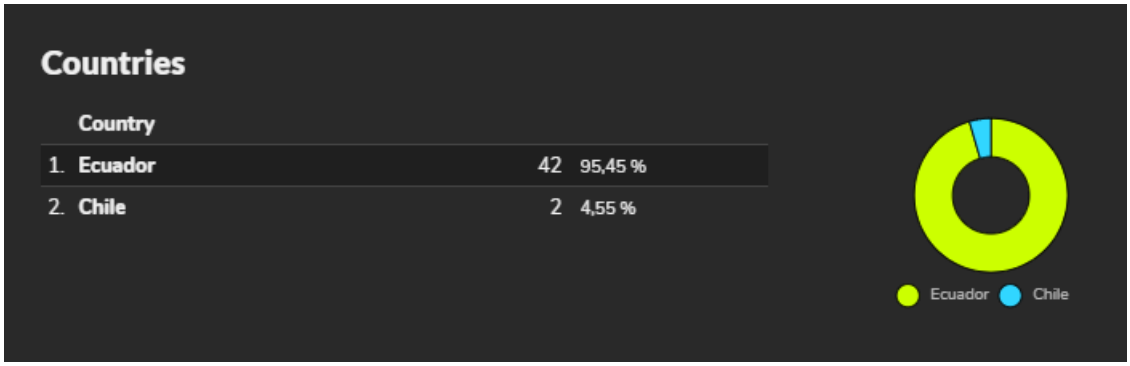

Figura 3.80: Descargas por país del videojuego
Elaborado por: Carlos Acuña

El sistema operativo destacado para la descarga del juego según la figura 3.81 fue other (que puede ser Android) y en segundo lugar fue Windows.

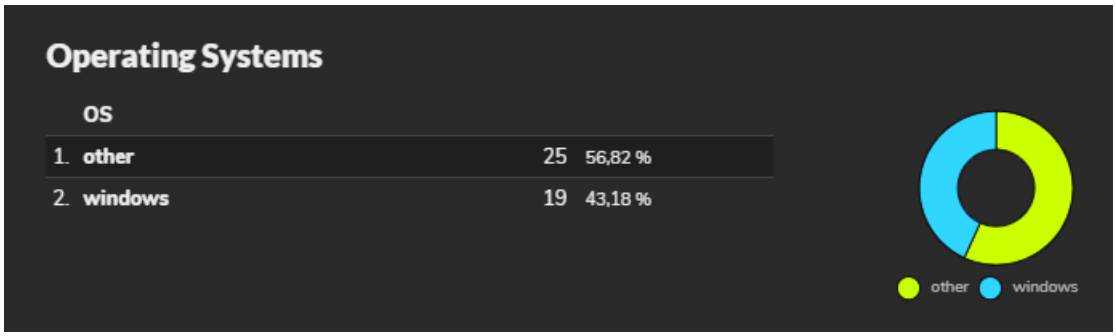

Figura 3.81: Descargas del videojuego por sistema operativo
Elaborado por: Carlos Acuña

Los resultados obtenidos indica una respuesta aceptable pese a ser un juego independiente y sobretodo que el trabajo llegó a un número de personas que viven en el Ecuador, lo que demuestra que a la gente le interesa su cultura y sus tradiciones y que los videojuegos no son ajenos al diario vivir de muchas de estas personas.

Capítulo 4

Conclusiones y recomendaciones

4.1 Conclusiones

- Al utilizar una metodología ágil como DAV, el desarrollo del videojuego tuvo orden y precisión en las tareas que se debe realizar además de agilizar el proceso de creación del proyecto, ya que con todos estos aspectos establecidos se ahorró bastante tiempo y esfuerzo, no hubo necesidad de volver a rehacer el juego desde el principio y tampoco sucedieron errores fatales durante la creación del mismo. Utilizar este tipo de metodologías ayuda a cualquier desarrollador a enfocarse en una tarea específica y completar el trabajo de la manera más óptima posible.

- Encontrar el framework adecuado ayudó a sintetizar las ideas que se tenían sobre la cultura y mitología inca en un videojuego. Al ser una herramienta totalmente nueva, se tiene que dedicar tiempo y esfuerzo en aprender las diferentes técnicas y prácticas de desarrollo correctas para elaborar un producto de calidad.

- Elaborar un videojuego sobre un tema como es la cultura y mitología inca fue un reto debido a toda la información que se debe buscar. Transformarla en una historia que sea fácil de entender, utilizar personajes históricos y momentos importantes de la historia que se convirtieron en niveles del videojuego fue una experiencia enriquecedora y compartir esa experiencia a través de este trabajo fue un objetivo que se cumplió al poder publicar el videojuego en una plataforma que llegó a muchas personas interesados en este tipo de temas.

- Publicar el videojuego en una plataforma como Gamejolt alienta a desarrolladores independientes a mostrar su trabajo y recibir retroalimentación de la gente para mejorar en todos los aspectos posibles dentro del desarrollo de videojuegos. Este videojuego puede inspirar a otros programadores a realizar sus propios proyectos y compartirlos con la comunidad con el fin de expandir este tipo de desarrollo, además de enseñar sobre una de las culturas más importantes de la antigüedad como es la cultura Inca.

4.2 Recomendaciones

- Para entender mejor el transfondo histórico del juego se puede dirigir a la sección historia donde se encuentra pequeñas cápsulas biográficas de todos los personajes y sucesos históricos que ocurrieron en realidad con el fin de entender mejor la historia y mitología ecuatoriana.

- El juego fue optimizado para la mayoría de computadoras que corren con windows pero es importante revisar los requerimientos de hardware y software para evitar problemas al momento de correr el juego.

- Gracias a la página donde fue publicado el videojuego se puede conocer de novedades, actualizaciones y reporte de errores que permiten al desarrollador mejorar el producto para satisfacer las demandas de los seguidores y ofrecer un software de calidad.

- Con más conocimiento y mejores habilidades en el diseño gráfico se puede pensar en una continuación del juego en una segunda versión, con aumento de héroes, habilidades, jefes de nivel entre otras cosas que pueden enriquecer el juego y contar aún más historias sobre la cultura ecuatoriana en general.

Bibliografía

[1] J Tomlinson. *Globalization and Culture*. Wiley, 2013.

[2] P Grimal. *La mitología griega*. Paidós studio. Paidós, 1991.

[3] M P F Álvarez and T M Antón. *Antología de la literatura nórdica antigua (edición bilingüe)*. Acta Salmanticensia. Ediciones Universidad de Salamanca, 2002.

[4] God of War for PlayStation 2 (2005) - MobyGames.

[5] Age of Mythology for Macintosh (2003) - MobyGames.

[6] Globalization and the U.S. — Boundless US History.

[7] Mario Vargas Llosa. Las culturas y la globalización — Edición impresa — EL PAÍS, 2000.

[8] Abril Mulato. Así pasaban sus ratos de ocio los dioses aztecas, según este cómic. jun 2016.

[9] Feria del Libro: jóvenes arequipeños presentan hoy novela gráfica — Noticias — Agencia Peruana de Noticias Andina.

[10] videojuego — Definición de videojuego - Diccionario de la lengua española - Edición del Tricentenario.

[11] Ana Laura Rossaro and Ana Laura Rossaro. Aprender jugando, los videojuegos y su potencial educativo. *Opinión*, 2012.

[12] Hussein Gabriel Rahman Núñez. Videojuego educativo en 3D para dispositivos móviles Android, enfocado al aprendizaje de la Lógica de Programación para usuarios entre los 5 a 18 años de edad. 2017.

[13] Alejandro Pardina Bustamante. Desarrollo de software aplicaciones móviles videojuegos lógica de programación. page 145, 2016.

[14] prehispánico, ca — Definición de prehispánico, ca - «Diccionario de la lengua española» - Edición del Tricentenario.

[15] E Chang-Rodriguez. *Latinoamerica: su civilizacion y su cultura*. Cengage Learning, 2007.

[16] B Baudouin. *Los Incas*. De Vecchi, Editorial, S.A., 2012.

[17] P R Steele, P S Catherine J. Allen, C J Allen, and ABC-Clio Information Services. *Handbook of Inca Mythology*. Handbooks of world mythology. ABC-CLIO, 2004.

[18] C de Pablos Heredero. *Informática y comunicaciones en la empresa*. Biblioteca de economía y finanzas. Esic Editorial, 2004.

[19] Programación Informática - Qué es y Definición 2019.

[20] P Lacasa. *Los videojuegos*. Nuevas tecnologías aplicadas a la educación. Morata, 2011.

[21] F Moya, C Gonzalez, and D Villa. *Desarrollo de Videojuegos: Tecnicas Avanzadas*. Cursos en Español.

[22] B. Baudouin. *Los Incas*. De Vecchi, Editorial, S.A., 2012.

[23] M.R. de Diez Canseco. *Enciclopedia temática del Perú: Primeras civilizaciones*. Enciclopedia temática del Perú. El Comerio, 2006.

[24] Súper Diario Familiar. *Batallas y personajes del Ecuador*. unieditorial, 2010.

[25] H.V. Morel and J.D. Moral. *Diccionario mitológico americano: dioses, razas, leyendas*. Horus (Buenos Aires, Argentina). Editorial Kier, 1987.

[26] X. Chen. *Developing Application Frameworks in .NET*. Books for professionals by professionals. Apress, 2004.

[27] A. Okita. *Learning C# Programming with Unity 3D*. Taylor & Francis, 2014.

[28] J.W. Murray. *C# Game Programming Cookbook for Unity 3D*. Taylor & Francis, 2014.

[29] A. Manzur and G. Marques. *Godot Engine Game Development in 24 Hours, Sams Teach Yourself: The Official Guide to Godot 3.0*. Sams Teach Yourself. Pearson Education, 2018.

[30] A. Sanders. *An Introduction to Unreal Engine 4*. CRC Press, 2016.

[31] Comience con UE4 — Documentación irreal del motor. [Online] https://docs.unrealengine.com/en-US/GettingStarted/index.html.

[32] Manual — Unity. [Online] https://docs.unity3d.com/es/2018.4/Manual/UnityManual.htm

[33] Nicolás Acerenza, Ariel Coppes, Gustavo Mesa, Alejandro Viera, Eduardo Fernández, Tomás Laurenzo, and Diego Vallespir. Una metodología para desarrollo de videojuegos: versión extendida. Technical report, 2009.

[34] A. Barreno, G. Alexander, S. Guaraca and M. Gonzalo. Adaptación de las metodologías ágiles Scrum y Extreme Game Development en una metodología para el desarrollo de videojuegos en Android. Caso práctico: Desarrollo de un videojuego. 2011.

[35] GIMP - About GIMP. [Online] https://www.gimp.org/about/.

[36] Visual Studio Code Frequently Asked Questions. [Online] https://code.visualstudio.com/docs/supporting/faq.

Apéndice A

Tarjetas de tareas

Tarjeta de tarea	
Número de tarea: T01	**Historia de Usuario (Nro. y Nombre):** HU08
Nombre de Tarea: Reflejar paisajes andinos mediante pixel art	
Tipo de tarea: Diseño	**Puntos estimados:** 60 horas
Inicio: 19 de diciembre del 2019	**Fin:** 26 diciembre del 2019
Miembro responsable: Carlos Acuña	
Descripción: Crear niveles en pixel art	

Tarjeta de tarea	
Número de tarea: T02	**Historia de Usuario (Nro. y Nombre):** HU11
Nombre de Tarea: El personaje principal cambie de animación según su acción	
Tipo de tarea: Diseño	**Puntos estimados:** 48 horas
Inicio: 27 de diciembre del 2019	**Fin:** 2 de enero del 2020
Miembro responsable: Carlos Acuña	
Descripción: Crear las animaciones del personaje principal	

Tarjeta de tarea	
Número de tarea: T03	**Historia de Usuario (Nro. y Nombre):** HU19
Nombre de Tarea: Acciones del personaje	
Tipo de tarea: Desarrollo	**Puntos estimados:** 36 horas
Inicio: 3 de enero del 2020	**Fin:** 8 de enero del 2020
Miembro responsable: Carlos Acuña	
Descripción: Crear acciones que permitan la interacción y progresión del juego	

Tarjeta de tarea	
Número de tarea: T04	**Historia de Usuario (Nro. y Nombre):** HU10
Nombre de Tarea: La cámara sigue al jugador en todo momento	
Tipo de tarea: Desarrollo	**Puntos estimados:** 36 horas
Inicio: 9 de enero del 2020	**Fin:** 14 de enero del 2020
Miembro responsable: Carlos Acuña	
Descripción: Crear evento que permite seguir al personaje principal	

Tarjeta de tarea	
Número de tarea: T05	**Historia de Usuario (Nro. y Nombre):** HU14
Nombre de Tarea: Los niveles tengan un fondo estilo pixel art	
Tipo de tarea: Diseño	**Puntos estimados:** 24 horas
Inicio: 15 de enero del 2020	**Fin:** 17 de enero del 2020
Miembro responsable: Carlos Acuña	
Descripción: Encontrar escenarios que sean pixelart	

Tarjeta de tarea	
Número de tarea: T06	**Historia de Usuario (Nro. y Nombre):** HU09
Nombre de Tarea: Visualizar las vidas restantes y la puntuación actual	
Tipo de tarea: Desarrollo	**Puntos estimados:** 48 horas
Inicio: 20 de enero del 2020	**Fin:** 24 de enero del 2020
Miembro responsable: Carlos Acuña	
Descripción: Visualizar las vidas restantes y la puntuación actual del personaje principal	

Tarjeta de tarea	
Número de tarea: T07	**Historia de Usuario (Nro. y Nombre):** HU13
Nombre de Tarea: El personaje pierda el nivel cuando sus vidas lleguen a 0	
Tipo de tarea: Desarrollo	**Puntos estimados:** 48 horas
Inicio: 27 de enero del 2020	**Fin:** 31 de enero del 2020
Miembro responsable: Carlos Acuña	
Descripción: El personaje pierda el nivel cuando sus vidas lleguen a 0 mediante código	

Tarjeta de tarea	
Número de tarea: T08	**Historia de Usuario (Nro. y Nombre):** HU06
Nombre de Tarea: Existan enemigos variados en cada nivel y puedan ser destruidos	
Tipo de tarea: Desarrollo	**Puntos estimados:** 48 horas
Inicio: 3 de febrero del 2020	**Fin:** 7 de febrero del 2020
Miembro responsable: Carlos Acuña	
Descripción: Creación, animación y codificación de varios enemigos en cada nivel	

Tarjeta de tarea	
Número de tarea: T09	**Historia de Usuario (Nro. y Nombre):** HU02
Nombre de Tarea: Antes de cada nivel conocer el transfondo histórico y los personajes principales	
Tipo de tarea: Desarrollo	**Puntos estimados:** 48 horas
Inicio: 10 de febrero del 2020	**Fin:** 14 de febrero del 2020
Miembro responsable: Carlos Acuña	
Descripción: Creación, animación y codificación para presentación del nivel y sus personajes históricos	

Tarjeta de tarea	
Número de tarea: T10	**Historia de Usuario (Nro. y Nombre):** HU12
Nombre de Tarea: Visualizar la meta de cada nivel	
Tipo de tarea: Desarrollo	**Puntos estimados:** 24 horas
Inicio: 17 de febrero del 2020	**Fin:** 19 de febrero del 2020
Miembro responsable: Carlos Acuña	
Descripción: Creación, animación y codificación de la meta de cada nivel	

Tarjeta de tarea	
Número de tarea: T11	**Historia de Usuario (Nro. y Nombre):** HU07
Nombre de Tarea: Al final de cada nivel exista una batalla contra un jefe	
Tipo de tarea: Desarrollo	**Puntos estimados:** 12 horas
Inicio: 20 de febrero del 2020	**Fin:** 21 de febrero del 2020
Miembro responsable: Carlos Acuña	
Descripción: Creación, animación y codificación de la pelea contra jefe de nivel	

Tarjeta de tarea	
Número de tarea: T12	**Historia de Usuario (Nro. y Nombre):** HU15
Nombre de Tarea: Los efectos de sonido sean acordes a las acciones del jugador	
Tipo de tarea: Desarrollo	**Puntos estimados:** 24 horas
Inicio: 24 de febrero del 2020	**Fin:** 26 de febrero del 2020
Miembro responsable: Carlos Acuña	
Descripción: Búsqueda e implementación del sonido y ambientación al juego	

Tarjeta de tarea	
Número de tarea: T13	**Historia de Usuario (Nro. y Nombre):** HU01
Nombre de Tarea: Entrar, salir y ver los créditos del juego	
Tipo de tarea: Desarrollo	**Puntos estimados:** 12 horas
Inicio: 27 de febrero del 2020	**Fin:** 28 de febrero del 2020
Miembro responsable: Carlos Acuña	
Descripción: Creación de las interfaces de usuario	

Tarjeta de tarea	
Número de tarea: T14	**Historia de Usuario (Nro. y Nombre):** HU03
Nombre de Tarea: Visualizar los controles del juego	
Tipo de tarea: Desarrollo	**Puntos estimados:** 24 horas
Inicio: 2 de marzo del 2020	**Fin:** 4 de marzo del 2020
Miembro responsable: Carlos Acuña	
Descripción: Visualizar los controles del juego	

Tarjeta de tarea	
Número de tarea: T15	**Historia de Usuario (Nro. y Nombre):** HU04
Nombre de Tarea: Visualizar los controles del juego	
Tipo de tarea: Desarrollo	**Puntos estimados:** 12 horas
Inicio: 5 de marzo del 2020	**Fin:** 6 de marzo del 2020
Miembro responsable: Carlos Acuña	
Descripción: Visualizar los controles del juego	

Tarjeta de tarea	
Número de tarea: T16	**Historia de Usuario (Nro. y Nombre):** HU05
Nombre de Tarea: Visualizar una pantalla que cuente el trasfondo histórico del juego	
Tipo de tarea: Desarrollo	**Puntos estimados:** 24 horas
Inicio: 9 de marzo del 2020	**Fin:** 11 de marzo del 2020
Miembro responsable: Carlos Acuña	
Descripción: Visualizar una pantalla que cuente el trasfondo histórico del juego	

Tarjeta de tarea	
Número de tarea: T17	**Historia de Usuario (Nro. y Nombre):** HU17
Nombre de Tarea: Agregar un story board en cada cambio de nivel	
Tipo de tarea: Desarrollo	**Puntos estimados:** 24 horas
Inicio: 12 de marzo del 2020	**Fin:** 13 de marzo del 2020
Miembro responsable: Carlos Acuña	
Descripción: Agregar un story board en cada cambio de nivel	