Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

CREACIÓN DE UN VIDEOJUEGO BASADO EN LA HISTORIA Y CULTURA DE GUATEMALA PARA LAS PLATAFORMAS ANDROID, IOS Y WINDOWS

Juan Alberto Barillas Velásquez
Asesorado por el Ing. Luis Fernando Espino Barrios

Guatemala, mayo de 2018

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

CREACIÓN DE UN VIDEOJUEGO BASADO EN LA HISTORIA Y CULTURA DE GUATEMALA PARA LAS PLATAFORMAS ANDROID, IOS Y WINDOWS

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

JUAN ALBERTO BARILLAS VELÁSQUEZ
ASESORADO POR EL ING. LUIS FERNANDO ESPINO BARRIOS

AL CONFERÍRSELE EL TÍTULO DE

INGENIERO EN CIENCIAS Y SISTEMAS

GUATEMALA, MAYO DE 2018

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Pedro Antonio Aguilar Polanco
VOCAL I	Ing. Angel Roberto Sic García
VOCAL II	Ing. Pablo Christian de León Rodríguez
VOCAL III	Ing. José Milton de León Bran
VOCAL IV	Br. Oscar Humberto Galicia Nuñez
VOCAL V	Br. Carlos Enrique Gómez Donis
SECRETARIA	Inga. Lesbia Magalí Herrera López

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Murphy Olympo Paiz Recinos
EXAMINADOR	Ing. César Augusto Fernández Cáceres
EXAMINADOR	Ing. Edgar Estuardo Santos Sutuj
EXAMINADORA	Inga. Mirna Ivonne Aldana
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

CREACIÓN DE UN VIDEOJUEGO BASADO EN LA HISTORIA Y CULTURA DE GUATEMALA PARA LAS PLATAFORMAS ANDROID, IOS Y WINDOWS

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería en Ciencias y Sistemas, con fecha de febrero de 2017.

Juan Alberto Barillas Velásquez

Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas

Guatemala, 10 de mayo de 2017

Ingeniero
Carlos Alfredo Azurdia Morales
Coordinador del Área de Trabajos de Graduación

Respetable Ingeniero Azurdia:

Por este medio le informo que como asesor del trabajo de graduación del estudiante universitario de la carrera de Ingeniería en Ciencias y Sistemas, JUAN ALBERTO BARILLAS VELÁSQUEZ, carné 200312931, he revisado y a mi criterio el mismo cumple con los objetivos propuestos para su desarrollo, según el protocolo del trabajo de graduación titulado: "CREACIÓN DE UN VIDEOJUEGO BASADO EN LA HISTORIA Y CULTURA DE GUATEMALA PARA LAS PLATAFORMAS ANDROID, IOS Y WINDOWS".

Agradeciendo su atención a la presente.

Atentamente,

Ing. Luis Fernando Espino Barrios
Escuela de Ciencias y Sistemas
Asesor de trabajo de graduación
Colegiado: 9145

Universidad San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 24 de Mayo de 2017

Ingeniero
Marlon Antonio Pérez Türk
Director de la Escuela de Ingeniería
En Ciencias y Sistemas

Respetable Ingeniero Pérez:

Por este medio hago de su conocimiento que he revisado el trabajo de graduación del estudiante **JUAN ALBERTO BARILLAS VELÁSQUEZ** con carné **200312931 y CUI 2435 26393 0101,** titulado **"CREACIÓN DE UN VIDEOJUEGO BASADO EN LA HISTORIA Y CULTURA DE GUATEMALA PARA LAS PLATAFORMAS ANDROID, IOS Y WINDOWS",** y a mi criterio el mismo cumple con los objetivos propuestos para su desarrollo, según el protocolo.

Al agradecer su atención a la presente, aprovecho la oportunidad para suscribirme,

Atentamente,

Ing. Carlos Alfredo Azurdia
Coordinador de Privados
y Revisión de Trabajos de Graduación

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
INGENIERIA EN CIENCIAS Y SISTEMAS
COORDINACION TESIS

ESCUELA DE INGENIERÍA EN CIENCIAS Y SISTEMAS

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERÌA
ESCUELA DE INGENIERÍA EN
CIENCIAS Y SISTEMAS
TEL: 24767644

*El Director de la Escuela de Ingeniería en Ciencias y Sistemas de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer el dictamen del asesor con el visto bueno del revisor y del Licenciado en Letras, del trabajo de graduación "**CREACIÓN DE UN VIDEOJUEGO BASADO EN LA HISTORIA Y CULTURA DE GUATEMALA PARA LAS PLATAFORMAS ANDROID, IOS Y WINDOWS",** realizado por el estudiante, JUAN ALBERTO BARILLAS VELÁSQUEZ aprueba el presente trabajo y solicita la autorización del mismo.*

"ID Y ENSEÑAD A TODOS"

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DIRECCION DE INGENIERIA EN CIENCIAS Y SISTEMAS

Ing. Marlon Antonio Pérez Türk
Director
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 28 de mayo de 2018

Universidad de San Carlos
de Guatemala

Facultad de Ingeniería
Decanato

Ref.DTG.D.190.2018

El Decano de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería en Ciencias y Sistemas, al trabajo de graduación titulado: **CREACIÓN DE UN VIDEOJUEGO BASADO EN LA HISTORIA Y CULTURA DE GUATEMALA PARA LAS PLATAFORMAS ANDROID, IOS Y WINDOWS,** presentado por el estudiante universitario: **Juan Alberto Barillas Velásquez,** y después de haber culminado las revisiones previas bajo la responsabilidad de las instancias correspondientes, se autoriza la impresión del mismo.

IMPRÍMASE.

Ing. Pedro Antonio Aguilar Polanco
Decano

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DECANO
FACULTAD DE INGENIERIA

Guatemala, mayo de 2018

/cc

ACTO QUE DEDICO A:

Mis padres	Por todo el apoyo que me dieron durante mi carrera y el que me han dado durante toda la vida.
Mi hermano	Por estar siempre ahí.
Mis abuelitas	Por todo el cariño que me demostraron siempre.
Mis compañeros de universidad	Por acompañarme durante toda la carrera y por el apoyo que siempre me brindaron.

AGRADECIMIENTOS A:

Universidad de San Carlos de Guatemala	Por ser una importante influencia en mi carrera, entre otras cosas.
Facultad de Ingeniería	Por ser una importante influencia en mi carrera, entre otras cosas.
Mi asesor de tesis	Al Ing. Luis Espino, por su apoyo para realizar este trabajo.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES V
LISTA DE SÍMBOLOS XI
GLOSARIO XIII
RESUMEN XV
OBJETIVOS XVII
INTRODUCCIÓN XIX
1. MARCO TEÓRICO 1
1.1. Videojuegos 1
1.1.1. Definición 1
1.1.2. Conceptos relacionados y características de los videojuegos 1
1.1.3. Tipos de videojuegos 3
1.1.4. Breve historia de los videojuegos 7
1.1.5. Antecedentes de aplicación en la educación 11
1.2. Parte técnica 16
1.2.1. Herramientas para programación de videojuegos 16
1.2.1.1. Definición y características 16
1.2.1.2. Alternativas en el mercado 16
1.2.1.3. Selección de las herramientas para el proyecto 22
1.2.2. Sistema operativo Android 24
1.2.2.1. Definición y características 24
1.2.1. Sistema Operativo iOS 27
1.2.1.1. Definición y características 27
1.2.2. Sistema Operativo Windows 30

1.2.2.1. Definición y Características 30
1.2.3. Herramientas adicionales a utilizar 33
2. IMPLEMENTACIÓN DEL VIDEOJUEGO 35
2.1. Creación de la trama 35
2.1.1. Investigación sobre historia y cultura de Guatemala 35
2.1.2. Selección del período histórico, ambiente y personajes 35
2.2. Selección del tipo de videojuego a crear 36
2.3. Metodología de desarrollo 37
2.3.1. Especificación de Requisitos de Software (ERS) 37
2.4. Arte del juego 50
2.5. Modelado gráfico del arte 52
2.5.1. Escenario 52
2.5.2. Objetos 72
2.5.3. Personajes 98
2.6. Programación del videojuego 100
2.6.1. Descripción de la arquitectura general 100
2.6.2. Animaciones 101
2.6.3. Controles 103
2.6.4. Menús 104
2.6.5. Física (colisiones) 115
2.6.6. Inteligencia artificial 117
2.6.7. Sonido 120
2.7. Pruebas y corrección de errores/bugs 123
3. UTILIZACIÓN DEL VIDEOJUEGO COMO HERRAMIENTA DE APRENDIZAJE 129
3.1.1. Hipótesis 129
3.2. Selección de la muestra 130

3.2.1. Realización de la investigación 131
3.3. Resultados obtenidos 135
CONCLUSIONES 137
RECOMENDACIONES 139
BIBLIOGRAFÍA 141

ÍNDICE DE ILUSTRACIONES

FIGURAS

1. Videojuego Pong! 11
2. Juego Big Brain Academy: Wii Degree 12
3. Juego Where in The World is Carmen Sandiego 13
4. Juego The Oregon Trail 14
5. Juego Quest Atlantis 15
6. Juego Team Xtreme: Operation Weather Disaster 15
7. Arquitectura del sistema operativo Android 27
8. Versiones recientes de iOS y Market Share 28
9. Capas de la arquitectura iOS 29
10. Arquitectura de la familia Windows NT que comprende Windows 10 .. 32
11. Esquema menú principal de la aplicación 42
12. Esquema interfaz del juego 43
13. Esquema interfaz menú de pausa 44
14. Dibujo del escenario 51
15. Dibujo de la pirámide 51
16. Terreno básico 53
17. Creación del lago (vista superior) 54
18. Creación del lago (vista de perspectiva) 54
19. Creación de montañas (vista superior) 55
20. Creación de montañas (vista de perspectiva) 56
21. Selección de textura base para el terreno 57
22. Terreno cubierto con textura de grama 58
23. Texturas añadidas a la paleta 59

24. Superficie y alrededores del lago con textura aplicada 60
25. Textura de grama y roca aplicada 61
26. Textura de roca aplicada a la cima de una montaña 62
27. Asset Store en Unity y paquete de árboles descargado 63
28. Selección del tipo de árbol a colocar en el terreno 64
29. Árboles colocados en el terreno, vista superior 65
30. Árboles colocados en el terreno, vista en perspectiva 65
31. Selección de la textura de grama a aplicar 67
32. Textura de grama aplicada al terreno 68
33. Agregando una luz direccional a la escena 69
34. Seleccionando y aplicando el tipo de destello a la luz direccional 70
35. Seleccionando y aplicando el tipo de destello a la luz direccional 71
36. Aplicado el objeto simulando el agua al lago 72
37. Inicio del modelado de la pirámide maya en Blender 73
38. Modificación del tamaño del cubo 74
39. Vista en perspectiva del nuevo objeto 74
40. Inclinación de las paredes exteriores del objeto 75
41. Vista en perspectiva del objeto 76
42. Vista superior del objeto con sus caras externas inclinadas 76
43. Objeto duplicado representando el segundo nivel de la pirámide 77
44. Vista en perspectiva de los dos primeros niveles de la pirámide 78
45. Pirámide con 7 niveles creados 79
46. Pirámide con 7 niveles creados. Vista superior 79
47. Pirámide con 7 niveles creados. Vista en perspectiva 80
48. Cubo base para el templo 81
49. Subdivisiones al cubo base 81
50. Alargado del cubo sobre el eje X 82
51. Selección de vértices 83
52. Resultado de la eliminación de los vértices 83

53. Creación de la apertura frontal 84
54. Refinamiento de la apertura frontal (I) 85
55. Refinamiento de la apertura frontal (II) 85
56. Unión de caras de las paredes del templo (I) 86
57. Unión de caras de las paredes del templo (II) 87
58. Paredes del templo con caras simplificadas 87
59. Inicio de modelado del techo del templo 88
60. Creación de la parte superior del templo 89
61. Creación de la parte superior del templo (II) 89
62. Unión del templo con la pirámide 90
63. Inicio de modelado de las escaleras 91
64. Modelado de las escaleras (I) 91
65. Modelado de las escaleras (II) 92
66. Modelado de las escaleras (III) 92
67. Modelado de las escaleras (IV) 93
68. Escaleras del templo terminadas 94
69. Pirámide I dentro del juego 95
70. Pirámide II dentro del juego 96
71. Modelo de un calendario maya (creado en Blender) 97
72. Modelo de una vasija maya (creado en Blender) 97
73. Modelado del objeto tortuga 99
74. Modelado del objeto esqueleto 99
75. Modelado del objeto quetzal 100
76. Animación del objeto “esqueleto” 102
77. Objeto para implementar controles en primera persona 104
78. Bosquejo del menú principal 105
79. Objetos pertenecientes al escenario agrupados 106
80. Escena duplicada para crear el menú 106
81. Vista del menú y cámara agregada a la escena 107

82. Archivo de logo importado como textura GUI 108
83. Logo agregado a la escena 109
84. Logo agregado a la escena, vista preliminar del menú 109
85. Objeto vacío (EmptyObject) agregado a la escena 110
86. Código de declaración de variables 110
87. Código de la función OnGUI 111
88. Código de creación de botones del menú principal 111
89. Valores de configuración del menú 112
90. Vista del menú finalizado 113
91. Ejemplo de código de acción de los botones del menú 114
92. Código de llamado a las opciones del menú 114
93. Menú “Instrucciones” mostrado en pantalla 115
94. Habilitando la opción Navigation Mesh para el terreno 118
95. Creando el Navigation Mesh 119
96. Vista de la geometría de navegación creada 120
97. Configuración de sonidos en Unity 122
98. Aplicación ejecutándose en dispositivo Android 126
99. Aplicación ejecutándose en el dispositivo iOS 127
100. Inicio del juego 132
101. Información mostrada al encontrar el esqueleto 132
102. Información mostrada al encontrar la vasija maya 133
103. Información mostrada al encontrar la tortuga 133
104. Información encontrada al encontrar el calendario maya 134
105. Información encontrada al encontrar el quetzal 134

TABLAS

I. Versiones de Android y market share 25

II. Market share de las versiones más utilizadas de Windows (abril 2017) ……. ……..31

III. Especificaciones Samsung Galaxy S7 124

IV. Especificaciones Huawei Y3II 124

V. Especificaciones Samsung Galaxy J5 (2016) 125

VI. Especificaciones iPhone 6S 126

VII. Resultados de la prueba antes de la utilización del juego 135

VIII. Resultados de la prueba después de la utilización del juego 136

LISTA DE SÍMBOLOS

Símbolo	Significado
%	Porcentaje

GLOSARIO

Display	Dispositivo que representa información de forma visual.
Framework	Plataforma de software reutilizable utilizada para desarrollar aplicaciones.
GUI (Graphical User Interface)	Interfaz que permite a un usuario interactuar con un dispositivo electrónico por medio de imágenes.
IDE	Siglas de Integrated Development Environment, una aplicación de software que provee un conjunto de herramientas integradas a los programadores para el desarrollo de software.
Inteligencia artificial	Área de las ciencias de la computación que estudia la inteligencia de las máquinas y robots.
JavaScript	Lenguaje de programación interpretado utilizado comúnmente en ambientes web.
Kernel	Componente de un sistema operativo que sirve de puente entre las aplicaciones y el hardware de una computadora.
Linux	Sistema operativo basado en código abierto.

Osciloscopio	Instrumento de medición electrónico que permite la observación de una representación gráfica de variaciones de voltaje en el tiempo.
Renderizar	Proceso de generar imágenes a partir de un modelo.
Software Open-Source	Categoría de software en la cual el código fuente está disponible para el dominio público y se provee el derecho a cualquiera de estudiar, modificar y distribuir este software.

RESUMEN

El presente trabajo de tesis describe el proceso de creación y desarrollo de un videojuego para dispositivos móviles cuyo contenido se basa en la historia y cultura de Guatemala.

En la primera sección del documento se plantean todas las bases teóricas sobre el significado y desarrollo de los videojuegos de tipo educativo, para luego en la siguiente sección del documento realizar una descripción detallada del proceso de creación de la aplicación desde la definición del contenido histórico y cultural incrustado en la trama del juego hasta el detalle de todos los aspectos técnicos que comprende su implementación y publicación en dispositivos móviles y en la sistema operativo Microsoft Windows..

Para finalizar, en el último capítulo se desarrolla una pequeña investigación para determinar si el uso de este tipo de videojuegos con contenido educativo ayuda al proceso de aprendizaje de personas de cualquier edad.

OBJETIVOS

General

Diseñar y desarrollar una demostración o prototipo de un videojuego basado en la historia y cultura de Guatemala para los sistemas operativos Android, iOS y Windows con la finalidad de mostrar al mundo la riqueza cultural del país, de una forma novedosa, y como herramienta educativa para personas que deseen conocer más sobre este país (niños y jóvenes en especial).

Específicos

1. Realizar una investigación corta sobre las herramientas de desarrollo de videojuegos para los sistemas Android, iOS y Windows, en la cual se incluya el análisis de, al menos, 3 herramientas de desarrollo y la exposición de las ventajas y desventajas del uso de cada una de ellas.

2. Definir el tipo de videojuego que se desarrollará (acción, aventura, estrategia, rol, etc.) basándose en las herramientas de desarrollo investigadas.

3. Crear un guión para el videojuego basado en elementos históricos y culturales de Guatemala. El guión se creará solamente para un nivel del videojuego, que constituirá la demostración o prototipo del mismo.

4. Realizar el arte gráfico para el videojuego que incluya un mínimo de 2 personajes, 5 objetos inanimados y el ambiente donde se desarrollará el nivel del juego.

5. Modelar (puede ser 3D o 2D dependiendo del tipo de juego que se desarrollará) los personajes, objetos y ambiente que conformarán el primer nivel del juego.

6. Programar la lógica del videojuego utilizando la herramienta de desarrollo seleccionada.

7. Diseñar y programar la interfaz de usuario y menús para el videojuego.

8. Realizar pruebas de usabilidad en un mínimo de 3 dispositivos Android, un dispositivo iOS y un dispositivo con sistema operativo Microsoft Windows. Estas pruebas se utilizarán para la corrección de errores en la programación del videojuego y la optimización del mismo.

9. Realizar un estudio breve para determinar si el uso del videojuego creado ayudó a que las personas aprendieran sobre la historia y cultura de Guatemala, utilizando un grupo de un mínimo de diez sujetos de prueba.

INTRODUCCIÓN

En la actualidad, los videojuegos se han convertido en una de las formas de entretenimiento más populares en la sociedad. Su campo de acción se ha expandido más allá de su simple utilización para fines de ocio, hasta convertirse en herramientas útiles en muchos campos de la vida humana, entre ellos, el educativo. Mediante la utilización de los videojuegos se ha comprobado que se pueden mejorar los procesos de enseñanza/aprendizaje dotándolos de mayor interactividad e interés.

El propósito del presente trabajo de graduación es ejemplificar el rol positivo que pueden desempeñar los videojuegos en el proceso educativo, mediante la creación de un videojuego con contenido histórico y cultural de Guatemala para la plataformas Android, iOS y Microsoft Windows, utilizando, posteriormente, la aplicación creada para realizar un estudio que demuestre que la utilización de este tipo de herramientas puede servir para enseñar conocimientos a personas de diferentes edades y con independencia de su experiencia con la utilización de videojuegos.

.

1. MARCO TEÓRICO

1.1. Videojuegos

1.1.1. Definición

Un videojuego se define como un juego electrónico en el cual el usuario ingresa comandos a través de un dispositivo de entrada y se retroalimenta por medio de un dispositivo de video. La definición anterior menciona algunos conceptos básicos que componen los videojuegos, los cuales se explican con mayor detalle en el presente documento.

Los videojuegos han cambiado significativa y rápidamente, a lo largo de sus años de existencia. Por ello, muchas veces es difícil encontrar una definición que incluya todos los tipos de videojuegos que existen en la actualidad. Sin embargo, algunas características o conceptos básicos son comunes a todos los tipos de videojuegos, como su origen electrónico y que el usuario puede obtener resultados o *feedback* por medio de un dispositivo de video que es, por lo general, una pantalla de televisión, computadora o de algún dispositivo móvil.

1.1.2. Conceptos relacionados y características de los videojuegos

En la siguiente sección se detallan algunos de los conceptos básicos y características principales de los videojuegos, que son esenciales para comprender cómo trabajan estos dispositivos de entretenimiento.

Interactividad: en el contexto de los sistemas informáticos (y de los videojuegos) puede definirse como la capacidad que tiene un sistema de responder de forma dinámica a la información que recibe del usuario.

Dispositivo de entrada: puede definirse como una interfaz por medio de la cual un usuario puede enviar comandos o ingresar información a un sistema. En el ámbito de los videojuegos, los dispositivos de entrada más comunes son los llamados controles o mandos que se utilizan, sobre todo, en las consolas caseras. En el ámbito de las computadoras personales es muy común la utilización del teclado y el ratón como dispositivo de entrada para jugar videojuegos, y en lo que se refiere a los dispositivos móviles, comúnmente el dispositivo de entrada, es el teclado del teléfono o la pantalla táctil del mismo.

Dispositivo de video: es un dispositivo de salida que presenta la información que recibe de un sistema de forma visual al usuario. Los dispositivos de video más comunes utilizados en el ámbito de los videojuegos son los televisores, los monitores de computadora y las pantallas de los celulares o dispositivos móviles.

Medio de distribución: en el ámbito de los videojuegos, el medio de distribución se refiere a la forma en la que el software de un videojuego es distribuido o entregado a los usuarios finales. Los dos medios de distribución más utilizados son los medios físicos y los digitales. Entre los medios físicos se utilizan o han utilizado los cartuchos, tarjetas de memoria, CD, DVD, Blu-ray discs y otros tipos de medios ópticos. Sin embargo, estos medios han perdido popularidad en los últimos años comparación con los medios digitales, como las descargas.

Arte de concepto: dado que los videojuegos poseen de forma inherente un fuerte componente de representación gráfica, es necesario crear un estilo o diseño visual para representarlo de forma computarizada dentro del video juego. El arte de concepto de un videojuego es básicamente el modelo o estilo visual que tomarán los elementos una vez sean animados dentro del juego. La gran mayoría de las veces, este arte de concepto empieza en forma de bosquejos creados en papel por un artista, los cuales son posteriormente animados y modelados (ya sea en dos o tres dimensiones) por los modeladores o diseñadores gráficos en una computadora.

Motor de videojuegos (*video game engine*): un motor de videojuegos (o *videogame engine* como se le conoce en inglés) es un *framework* que abstrae y en el cual se encuentran implementadas muchas funciones o tareas comunes a muchos videojuegos. Los motores de juegos permiten la reutilización de código y ayudan a los artistas, programadores y diseñadores a enfocarse en implementar y crear las características que hacen a su juego único, sin preocuparse por reimplementar funciones básicas que comparten muchos videojuegos del mismo género. Algunas de las funcionalidades típicamente implementadas en los motores de videojuegos incluyen las funciones de renderizado, física, *input*, detección de colisiones, interfaces gráficas de usuarios, etc.[1]

1.1.3. Tipos de Videojuegos

Existen diversas formas de clasificar los tipos de videojuegos que existen. Algunas de las clasificaciones se basan en el tipo de contenido que brindan los videojuegos, otras en el modo de jugarlos, algunas otras en el tipo de audiencia

[1] WARD, Jeff. *What is a Game Engine?* http://www.gamecareerguide.com/features/529/what_is_a_game_.php. Consulta: mayo de 2017.

a la que van dirigidos, e incluso, hay algunas que lo hacen de acuerdo con el tipo de dispositivo en el cual el juego ha sido implementado. La clasificación más común que se utiliza para categorizar los videojuegos es la clasificación basada en el género (o *videogame genre* como se le conoce en inglés). Esta clasificación básicamente agrupa los videojuegos de acuerdo con su modo de juego, es decir, la forma en la que el videojugador interactúa dentro del mismo. A continuación, se describen brevemente los géneros más comunes y populares de videojuegos:

Juegos de acción*:* en ellos, el jugador debe usar su destreza de coordinación entre sus ojos y sus manos para completar los objetivos del juego.[2] Existe una gran cantidad de subgéneros dentro de la categoría de los juegos de acción, entre ellos:

Juegos de Peleas*:* la característica básica de este tipo de juego es que el jugador controla un personaje el cual combate/pelea con otro personaje (o varios) controlados, ya sea por la aplicación (por medio de algún tipo de inteligencia artificial) o por otro jugador o jugadores.

Juegos de Plataformas*:* tipo de juego en el cual el jugador controla a un personaje que progresa dentro del juego al saltar entre distintas plataformas u obstáculos que están organizados de diversas maneras para dificultar el avance del personaje. Comúnmente, este tipo de juegos se desarrollan en un ambiente en dos dimensiones (2D).

Juegos de disparos (*shooters*): se caracterizan, principalmente, porque el personaje controlado por el jugador usa un arma de largo alcance,

[2] HANDFORD, Forest J. *East Coast Games – Game Development Glossary*. http://www.eastcoastgames.com/design/glossary.html. Consulta: abril de 2017

generalmente, es una pistola. Los juegos en primera persona (conocidos como *first person shooters*) son los más populares de este género. En ellos, el jugador “mira” a través de los ojos del personaje.

Juegos de aventura*:* se diferencian de los juegos de acción en que su objetivo principal no es probar las habilidades de coordinación o reflejos del videojugador, sino contar una historia (en la mayor parte de los casos sobre el personaje controlado por el jugador o héroe de la historia) y permitir al jugador la exploración de un mundo virtual. Los retos que enfrenta el jugador en este tipo de juegos son comúnmente rompecabezas que debe resolver.

Juegos de estrategia: en este tipo de videojuegos los retos presentados al jugador son de naturaleza estratégica y de planificación. La mayoría de veces incluyen tácticas de batalla, de economía y retos de exploración. La clave para la victoria es desarrollar una estrategia/planificación superior a la de los rivales. A diferencia de los juegos de acción, la coordinación física del jugador tiene poca o ninguna importancia en los juegos de estrategia.

Existen dos tipos básicos de juegos de estrategia: los juegos en tiempo real o los basados en turnos. En los juegos basados en turnos cada jugador puede reflexionar su próximo movimiento y tomarse el tiempo para ello, mientras que los demás jugadores esperan su turno para realizar sus acciones. Un ejemplo de este tipo de juego es el ajedrez. A diferencia de los juegos basados en turnos, en los juegos en tiempo real los jugadores no tienen turnos para planear sus movimientos o estrategias, sino que todo ocurre de forma simultánea, lo cual obliga a los jugadores a pensar rápido y a hacer más dinámico el juego.

Juegos de rol*:* los juegos de rol son aquellos en los que el desarrollo del personaje o personajes del juego (sus poderes y habilidades, por ejemplo) son una parte esencial del mismo. El personaje o personajes participan en una serie de aventuras conocidas como "*quests*", por medio de las cuales ganan experiencia para su desarrollo. Los retos presentados en este tipo de juegos son de tipo de combate táctico, exploración, logística o de resolver rompecabezas. Igual que los juegos de estrategia, existen juegos de rol con combate por turnos o en tiempo real.

Juegos de deportes*:* simulan un deporte de la vida real, como el fútbol, basquetbol, tenis, etc., aunque hay juegos basados en algún deporte imaginario. En ellos predominan los retos físicos y estratégicos.

Juegos de simulación de vehículos*:* introduce al videojugador a una experiencia de operación de algún tipo de vehículo, ya sea real o imaginario. La mayoría de estos juegos crean una experiencia lo más cercana posible a la realidad. Los más comunes son los juegos de carreras de autos y los juegos de aviones.

Juegos de construcción y gestión*:* presentan al videojugador desafíos de crecimiento económico, exploración y construcción. Pueden encontrarse como híbridos con elementos de los juegos de estrategia. Promueven una forma de control indirecto del jugador, ya que no puede controlar en su totalidad toda la simulación/proceso planteada en el juego y solo puede controlar ciertas entradas al mismo.

1.1.4. Breve Historia de los videojuegos

A pesar de que los videojuegos se volvieron comerciales y populares en la década de 1980, muchos años antes de esa explosión de popularidad existieron algunos videojuegos que, aunque podrían considerarse un poco primitivos, exponían de muy buena forma los principios básicos que tiempo después adoptarían los videojuegos más recientes.

- **Primeros videojuegos**

A continuación, se presenta una breve reseña de los primeros juegos dispositivos de juego creados por el hombre.

  - **Dispositivo de entretenimiento de tubos de rayos catódicos (*cathode ray tube amusement device*)**

Este dispositivo es posiblemente el primer juego electrónico interactivo producido por el hombre. Fue creado por Thomas T. Goldsmith Jr. y Estle Ray Mann. Se creó alrededor del año 1947 y su funcionamiento se basaba en el uso de un tubo de rayos catódicos.

El objetivo del juego era básicamente derribar aviones controlando un tipo de mira y disparando hacia ellos. La mira aparecía como un punto en la pantalla del juego la cual controlaba el jugador por medio de una manija. Al lograr alinear esta mira con uno de los aviones dibujados sobre la pantalla y presionar un botón de disparo, se creaba un efecto de explosión simulando el derribo del avión, con lo cual se cumplía con el objetivo del juego. [3]

[3] Fuente: COHEN, D.S. *Cathode-Ray Tube Amusement Device – The First Electronic Game.* https://www.lifewire.com/cathode-ray-tube-amusement-device-729579*. Consulta: abril de 2017.*

- **Juego de ajedrez**

El matemático británico Alan Turing diseñó en el año de 1947 un programa para un juego de ajedrez, aunque solo de forma teórica. Tiempo después, Dietrich Prinz lo desarrolló para crear el primer juego de ajedrez para la computadora Ferranti Mark I, la cual es conocida como la primera computadora comercial para uso general producida. Este juego tenía la limitante de que solo podía calcular ciertos movimientos y no podía jugar un juego completo de ajedrez.

- ***Nim***

En 1951, se presentó la primera computadora digital creada específicamente para jugar un videojuego. Esta computadora, llamada NIMROD y creada por la firma británica Ferranti, ejecutaba el juego conocido como Nim, un juego matemático de estrategia.

- ***Tic-Tac-Toe***

En el año 1952, el profesor británico Alexander Douglas creó el primer juego en utilizar un *display* digital. El juego, conocido como OXO, fue creado para la computadora EDSAC, y era básicamente una versión del popular juego Tic-Tac-Toe, tres en raya o “totito", como popularmente se le conoce.[4]

.

[4] Fuente: COHEN, D.S. *OXO aka Noughts and Crosses - The First Video Game*. http://classicgames.about.com/od/computergames/p/OXOProfile.htm. Consulta: abril de 2017.

- **Tenis para dos**

En el año de 1958, el físico americano William Higinbotham creó un juego llamado Tenis para dos (*Tennis for two*) que simulaba un juego de tenis. Este juego fue creado para la computadora analógica Donnel Model 30 y utilizaba una pantalla de osciloscopio para la visualización de las imágenes del juego.[5]

- ***Spacewar!***

En el año de 1961, varios estudiantes del Massachusetts Institute of Technology (MIT) crearon el juego Spacewar! en la mini-computadora DEC PDP-1. Se creó para que intervinieran dos jugadores. Cada uno controlaba una nave especial y debían intentar destruir la nave de su oponente.

- ***Magnavox Odyssey***

La Magnavox Odyssey o *Brown Box* es considerada la primera consola casera de videojuegos. La diseñó Ralph H. Baer y fue la primera consola en utilizar una televisión como dispositivo de visualización de las imágenes de los juegos. Había 12 juegos que podían ser utilizados con ella.

- ***Galaxy Game***

En el año de 1971, Bill Pits y HughTuck desarrollaron el juego Galaxy Game, el primer juego de computadora en utilizar monedas. El juego, en sí, era una versión de Spacewar! y la única máquina creada fue instalada en la Universidad de Sanford en Estados Unidos.

[5] Fuente: *The First Videogame?* https://www.bnl.gov/about/history/firstvideo.php . Consulta: abril de 2017.

- ***Computer Space***

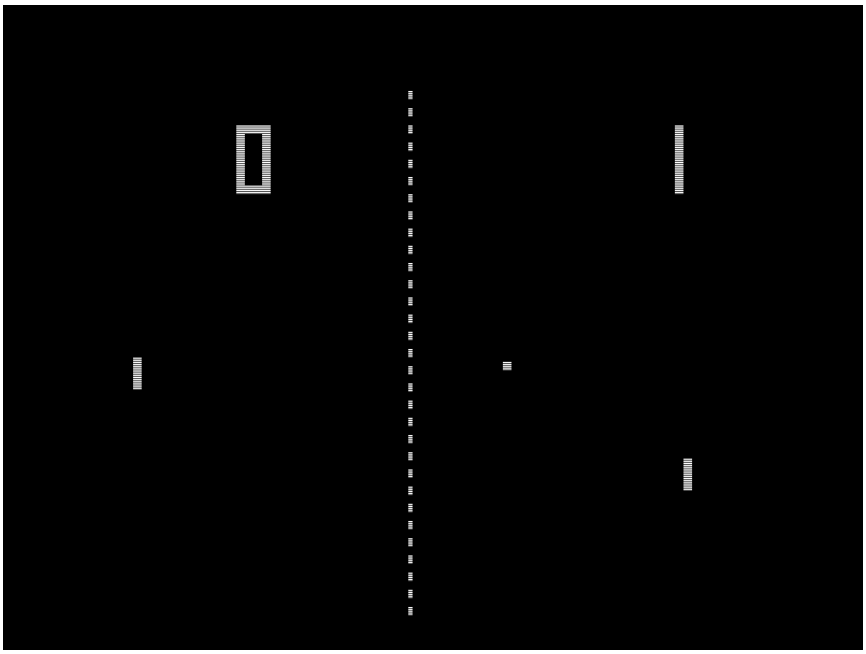
Fue un juego creado por Nolan Bushnell y Ted Dabney, como una variación de Spacewar!. El juego era operado por moneadas y es considerado el primer juego (operado por monedas o no) comercial de la historia.

- ***Pong***

El juego conocido como Pong fue uno de los primeros videojuegos en tener un gran éxito comercial. El concepto del juego no era nada nuevo en el mundo de los videojuegos: consistía en una simulación de un juego de tenis de mesa en el cual participaban dos jugadores. El juego, desarrollado por la empresa Atari, se creó en un inicio como una máquina monedera para su utilización en tabernas y locales de juegos, pero luego evolucionó a una versión de consola casera que tuvo de igual forma un gran éxito comercial, catapultando de esa forma la industria de las consolas caseras de videojuegos que sigue floreciendo aún hasta nuestros días.[6]

[6] Fuente: ALCOM, Alan. *First-Hand:The Development of Pong: Early Days of Atari and the Video Game Industry.* http://www.ieeeghn.org/wiki/index.php/First-Hand:The_Development_of_Pong:_Early_Days_of_Atari_and_the_Video_Game_Industry. Consulta: abril de 2017.

Figura 1. **Videojuego *Pong!***



Fuente: *Pong*. https://en.wikipedia.org/wiki/Pong. Consulta: abril de 2017.

1.1.5. Antecedentes de aplicación en la educación

Un videojuego educativo podría definirse como un juego cuyo objetivo principal es brindar al jugador algún tipo de conocimiento o potenciarlo. En los juegos educativos, la importancia no recae tanto en su mecánica de jugabilidad (o el género al que pertenece), sino en la información que brinda al usuario.

Aunque, como se mencionó anteriormente, el objetivo principal de un juego educativo es el proceso de enseñanza/aprendizaje, esto no significa que carezcan del factor entretenimiento, ya que por él los videojuegos son, muchas veces, una excelente estrategia para enseñar algún contenido a un grupo de personas.

Para tener una mejor idea del concepto detrás de los juegos educativos, se listan a continuación algunos ejemplos:

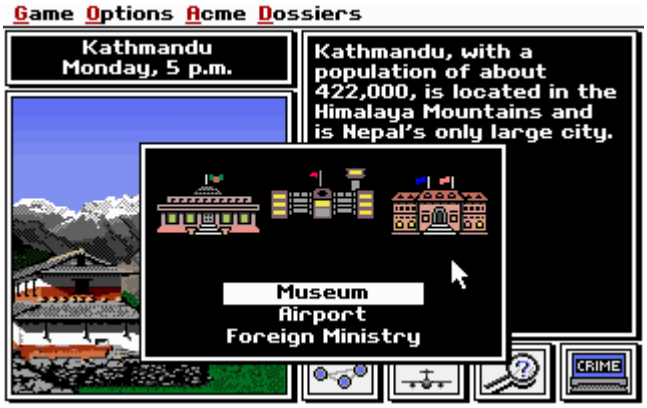
- *Big Brain Academy: Wii Degree*

Figura 2. **Juego *Big Brain Academy: Wii Degree***

Fuente: *HANDS-ON BIG BRAIN ACADEMY: WII DEGREE*. http://www.ign.com/articles/2007/05/25/hands-on-big-brain-academy-wii-degree. Consulta: abril de 2017.

- *Where in The World is Carmen Sandiego*

Figura 3. **Juego *Where in The World is Carmen Sandiego***



Fuente: *Where in the World is Carmen Sandiego*. http://www.abandonia.com/en/games/13/Where+in+the+World+is+Carmen+Sandiego.html. Consulta: abril de 2017.

- *The Oregon Trail*

Figura 4. **Juego *The Oregon Trail***

Fuente: *The Oregon Trail (video game).* https://en.wikipedia.org/wiki/The_Oregon_Trail_(video_game). Consulta: abril de 2017.

- *Quest Atlantis*

Figura 5. **Juego *Quest Atlantis***

Fuente: *Quest Atlantis*. https://sites.google.com/site/learningcontract11448457/learning-objectives/quest-atlantis. Consulta: abril de 2017.

- *Team Xtreme: Operation Weather Disaster*

Figura 6. **Juego *Team Xtreme: Operation Weather Disaster***

Fuente: *Team Xtreme: Operation Weather Disaster* http://serious.gameclassification.com/EN/games/16547-Team-Xtreme-Operation-Weather-Disaster/index.html. Consulta: abril de 2017.

1.2. Parte técnica

1.2.1. Herramientas para programación de videojuegos

Existen diversos tipos de herramientas para programar o crear videojuegos dependiendo del tipo de videojuego que se quiera crear o la plataforma objetivo para el cual se esté programando. A continuación, se describen algunas de las herramientas más comunes que existen en la actualidad.

1.2.1.1. Definición y características

Debido a que el proceso de desarrollo de videojuegos es largo y con varias etapas, hay diversos tipos de herramientas que se utilizan en cada una de ellas.

Las etapas técnicas más importantes en la creación de un videojuego son el modelado de los objetos que formarán parte del videojuego y la programación del comportamiento o la lógica de cómo interactuarán estos objetos dentro del mismo.

1.2.1.2. Alternativas en el mercado

- **Herramientas de modelado en 3D**

Las herramientas de modelado en 3D permiten crear representaciones computarizadas en tres dimensiones de cualquier tipo de objeto ya sea real o imaginario. Algunas de las herramientas de este tipo más populares en el mercado se describen a continuación.

- *Maya*

Es una aplicación propietaria distribuida y desarrollada por la compañía Autodesk, Inc. Se utiliza para crear modelos o recursos en tercera dimensión para usarlos en videojuegos, películas, etc. Esta aplicación funciona para los sistemas operativos Microsoft Windows, Mac OS y Linux.

Algunas de sus características son las herramientas de modelado 3D, como NURBS, *Subdivision Surfaces*, diversos polígonos básicos, modelado por medio de polígonos y subdivisión de mallas (*meshe*s), modelado de superficies, UVs, Normales y Colores por Vértice.

Maya también posee funciones de animación 3D, efectos especiales como fluidos, partículas, fuerzas; y diversas funciones de renderizado.[7]

- *3D Studio Max*

Al igual que *Maya*, *3D Studio Max* es un software desarrollado por Autodesk, Inc., y que se utiliza para modelado 3D, creación de animaciones y renderizado.

Algunas de sus características principales son las siguientes:

Creación de objetos por medio de polígonos, curvas tipo *spline* y NURBS. Aplicación de texturas y tonos/sombreado. Diversas herramientas de animación, como animación procedural. Herramientas para crear personajes, como esqueletos, modificadores de piel y músculos. Efectos de simulación de ropa,

[7] Fuente: *Autodesk Maya Features*. http://usa.autodesk.com/maya/features/. Consulta: abril de 2017.

cabello, pelaje, y diversos efectos basados en partículas. Y, por último, varias capacidades de renderizado en 3D y opciones de interoperabilidad con otros programas como Adobe Photoshop, After Effects, etc.

- *Blender*

Es una suite de creación de contenido 3D *open-source*. Está disponible para los sistemas operativos más utilizados como Windows, Linux, Mac Os y FreeBSD. Algunas de las características o capacidades de este software se listan a continuación:

Interfaz personalizable
Modelado de personajes (*Character Modeling*)
Rigging
Modelado de sólidos
Animación
Renderizado
UV Unwrapping
Renderizado por medio de trazado de rayos (*Raytrace Rendering*)
Física y partículas
Shading
Creación de juegos
Imaging – Compositing
Aplicación extensible
Manejo, importación y exportación de una gran cantidad de formatos de archivos.[8]

[8] Fuente: *Blender Features*. http://www.blender.org/features-gallery/features/. Consullta: abril de 2017.

- *Cheetah 3D*

Cheetah 3D es un software de modelado, animación y renderizado creado para el sistema operativo Mac OS. Algunas de las características que tiene esta aplicación son:

Herramientas de modelado de polígonos, superficies de subdivisión (*subdivision surfaces*) y *splines*.

Herramientas de animación incluyendo *Character Rigging*.
Dinámica – Física
Edición UV
Renderizado
Scripting
Importación y exportación de diversos formatos de animación y modelos.[9]

- *Cinema 4D*

Cinema 4D es una aplicación para modelado, animación y renderizado desarrollada por la compañía *Maxon Computer*. Existen diferentes versiones de la aplicación, pero la versión Cinema 4D Studio es la más robusta y con la mayor cantidad de características. [10]

[9] Fuente: *Sitio web oficial de Cheetah3D*. http://www.cheetah3d.com/index.php. Consulta: abril de 2017.

[10] Fuente: *Sitio web oficial de Maxon*. http://www.maxon.net/products/cinema-4d-studio/overview.html. Consulta: abril de 2017.

- ***Videogame engines***

El motor de juegos o *videogame engine* es, posiblemente, la parte de software más indispensable al desarrollar un videojuego. Una correcta selección de este motor puede determinar en gran parte el éxito o fracaso de un proyecto de este tipo.

La definición de un motor de videojuegos se abordó previamente en este documento, por lo que ahora se presenta una breve reseña de algunos de los motores de juego más populares del mercado en la actualidad.

  - *Unity*

Unity es una herramienta de desarrollo de videojuegos en tres dimensiones (3D). Unity posee capacidades de renderizado, iluminación (sombras en tiempo real, *lightmapping*, entre otras), creación de terrenos (con una herramienta de creación de árboles), texturas de tipo *substances*, física basada en el motor NVIDIA PhysX, *pathfinding,* audio a través de FMOD, capacidades de *scripting* en tres lenguajes: JavaScript, C# y Boo, y soporte para funciones de red en los juegos.

Unity también posee la capacidad de publicar un juego a varias plataformas distintas, como lo son el Unity Web Player, Flash, iOS, Android, PS3, Xbox 360, Wii y PC.[11]

[11] Fuente: *Sitio web oficial de Unity3D*. http://unity3d.com/unity/. Consulta: abril de 2017.

- *Unreal Engine*

Unreal Engine (actualmente en su versión 4) es un framework de desarrollo de videojuegos para las PCs, Xbox 360, iOS y Playstation 3. Algunas de las características o *features* de esta herramienta son las siguientes:

Animación
Inteligencia Artificial (AI)
The Unreal Audio System
El sistema de efectos de partículas Unreal Cascade
Cinematics
Unreal Game Editor
El sistema de *scripting* “Kismet”
Iluminación por medio de *Unreal Lightmass*
Creación y edición de terrenos
Conectividad LAN y “direct IP”
Física por medio de NVIDIA PhysX
Shaders en tiempo real
Renderizado por medio del sistema Gemini
Lenguaje de programación *Unreal Script*
Soporte multi-núcleo por medio del sistema de computación distribuida *Unreal Swarm*.[12]

- *Source Engine*

Source Engine es un motor de juegos creado por la compañía Valve. Ha sido utilizado en varios de los juegos más exitosos de esta empresa como

[12] Fuente: *Unreal Engine Features*. http://www.unrealengine.com/en/features/. Consulta: abril de 2017.

HalfLife 2, Portal y TeamFortress 2. El motor tiene un enfoque modular y tiene funciones de animación, inteligencia artificial avanzada, física, renderizado basado en *shaders*, funciones de red sobre LAN e internet, etc.[13]

- *Blender*

A pesar de ser principalmente una herramienta de modelado en 3D, *Blender* también tiene integrado un motor de juegos que cuenta con diversas características como:

Editor lógico gráfico
Detección de colisiones y física incluyendo el *BulletPhysics Library* y física de vehículos
Scripting en Python
Iluminación basada en OpenGL
Diversos tipos de materiales y texturas
Reproducción de juegos sin compilar ni preprocesamiento
Audio por medio del SDL *toolkit*
Escenas en múltiples capas [14]

1.2.1.3. Selección de las herramientas para el proyecto

El criterio más importante al seleccionar el *game engine* que se utilizará para el desarrollo del juego fue su compatibilidad para exportar el juego diseñado y creado a varias plataformas, especialmente Android, iOS y Windows. De todos los motores examinados, el que mejor cumple con este

[13] Fuente: *Source Engine*. http://source.valvesoftware.com/. Consulta: abril de 2017.
[14] Fuente: *Blender Features*. http://www.blender.org/features-gallery/features/. Consulta: abril de 2017.

requisito es Unity. Como fue mencionado anteriormente, Unity permite la publicación de un mismo proyecto a varias plataformas, entre ellas Android, iOS y Windows, sin tener que crear diferentes versiones del juego con diferentes tipos de programación por parte del desarrollador.

Otro de los criterios que se tomó en cuenta durante la selección de herramientas para trabajar fue su costo. Algunas de las mejores y más completas herramientas de desarrollo de videojuegos y de modelado 3D tienen un alto costo, por lo cual no es viable su uso para proyectos pequeños de desarrollo como el juego que se quiere crear. Este es otro de los puntos en los que Unity resultó ser una herramienta apropiada para el proyecto, ya que se ofrece una versión gratuita con todas las características básicas para crear un juego completo y de calidad.

Una vez seleccionado Unity como *game engine* para el juego, se buscó una aplicación de modelado que tuviera funciones de exportación de los modelos compatibles con Unity. Las aplicaciones encontradas que cumplen con este requisito fueron: Maya, 3D Studio Max, Cheetah 3D, Cinema 4D, Blender, Carrara, Lightware, XSI y Modo.

Se tomó, nuevamente, el criterio del costo para la selección y con base en esto se eligió Blender, debido a su naturaleza *open-source* y gratuita, además de que se cuenta con cierta experiencia previa en esta herramienta.

1.2.2. Sistema operativo Android

1.2.2.1. Definición y características

Android es un sistema operativo desarrollado principalmente por la compañía Google para usarlo en dispositivos móviles, como los smartphones o tablets. Android está basado en el sistema GNU/Linux.

A pesar de que el componente principal de Android es el sistema operativo, éste puede ser más bien considerado como un conjunto de aplicaciones entre las cuales se encuentra el mismo sistema operativo, componentes *middleware* y algunas otras aplicaciones esenciales para el funcionamiento del conjunto.

Algunas características de Android han ido cambiando con las versiones de este sistema, que se ha caracterizado por tener un mercado bastante fragmentado en cuanto a las versiones instaladas en la variedad de dispositivos que lo soportan. Esto significa que existe una gran cantidad de dispositivos ejecutando versiones antiguas de Android y que aún no han dado el salto a la última versión que posee las características más recientes del sistema.

Tabla I. **Versiones de Android y *market share***

No. de Versión	Nombre	Versión del API	Distribución
2.2	Froyo	8	0.1%
2.3.3 – 2.3.7	Gingerbread	10	1.7%
4.0.3 – 4.0.4	Ice Cream Sandwich	15	1.6%
4.1.x	Jelly Bean	16	6.0%
4.2.x		17	8.3%
4.3		18	2.4%
4.4	KitKat	19	29.2%
5.0	Lollipop	21	14.1%
5.1		22	21.4%
6.0	Marshmallow	23	15.2%

Fuente: *Paneles de control*. http://developer.android.com/about/dashboards/index.html. Consulta: abril de 2017.

- Características generales de Android

    - **Gráficos:** Android hace uso de la librería OpenGL ES 2.0 para desplegar gráficos en 2D y 3D.
    - **Almacenamiento:** Android almacena sus datos en la base relacional SQLite.
    - **Conectividad:** soporta tecnologías como GSM/EDGE, CDMA, UMTS, Bluetooth, Wi-Fi, WiMAX, entre otras.
    - **Mensajería:** soporta mensajes tipo SMS, MMS, y Android Google Cloud Messaging (GCM).
    - **Soporte para múltiples lenguajes.**
    - **Navegador Web:** navegador basado en los motores *open-source* WebKit y Chrome V8 Javascript.

- **Soporte para Java:** la mayoría de las aplicaciones para Android están creadas en Java y son ejecutadas en el dispositivo por la máquina virtual Dalvik.
- **Soporte de Media:** Android soporta una gran cantidad de formatos de archivos de multimedia, entre los cuales se encuentran: H.263, H.264, MPEG-4 SP, AMR, AAC, MP3, MIDI, OggVorbis, FLAC, WAV, JPEG, PNG, GIF, BMP, entre otros.

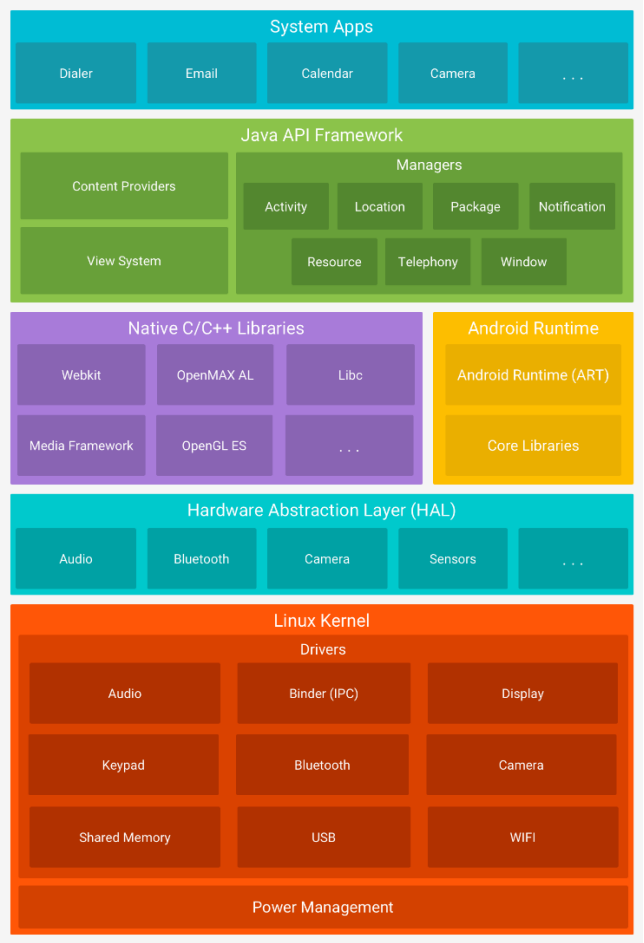
- Arquitectura de Aplicaciones para Android

El desarrollo de aplicaciones para la plataforma Android se realiza utilizando el lenguaje de programación Java. El Android SDK (*Software Development Kit*) provee las librerías y herramientas necesarias para la construcción de aplicaciones para este sistema.

La arquitectura de Android está formada básicamente por cinco componentes, los cuales son: las aplicaciones, el f*ramework* de aplicaciones, librerías, el *runtime* de Android y el kernel de Linux.[15]

[15] Fuente: *App Framework*. https://developer.android.com/guide/platform/index.html. Consulta: mayo de 2017.

Figura 7. **Arquitectura del sistema operativo Android**



Fuente: *Arquitectura de la plataforma*. http://developer.android.com/images/system-architecture.jpg. Consulta: abril de 2017.

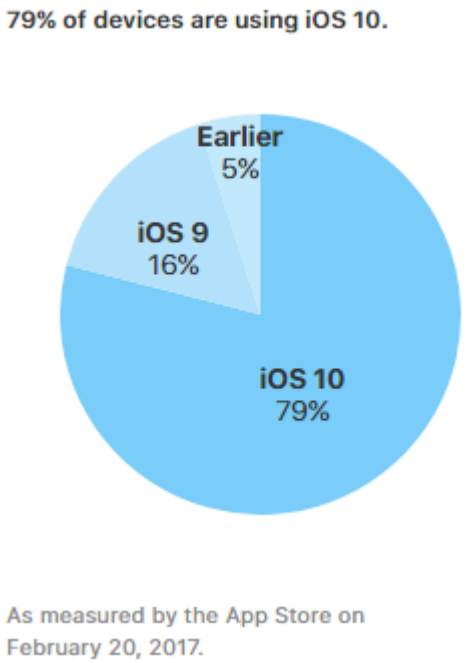
1.2.1. Sistema Operativo iOS

1.2.1.1. Definición y características

El sistema iOS es un sistema operativo móvil creado por la empresa Apple Inc. para sus dispositivos iPhone, iPad y iPod Touch. La funcionalidad principal

del sistema operativo es administrar el hardware de los dispositivos Apple y proveer comunicación entre dicho hardware y las aplicaciones utilizadas por el usuario, aunque también integra varias aplicaciones nativas que proveen funcionalidades básicas al usuario tales como el correo electrónico (aplicación *Mail*[16]) o la navegación en la web (con el navegador Safari[17]), entre otras. La versión más reciente de iOS a la fecha es la versión 10. Actualmente, iOS es el segundo sistema operativo para móvil más utilizado en el mundo, por detrás de Android.

Figura 8. **Versiones recientes de iOS y Market Share**



Fuente: *App Store*. https://developer.apple.com/support/app-store/. Consulta: mayo de 2017.

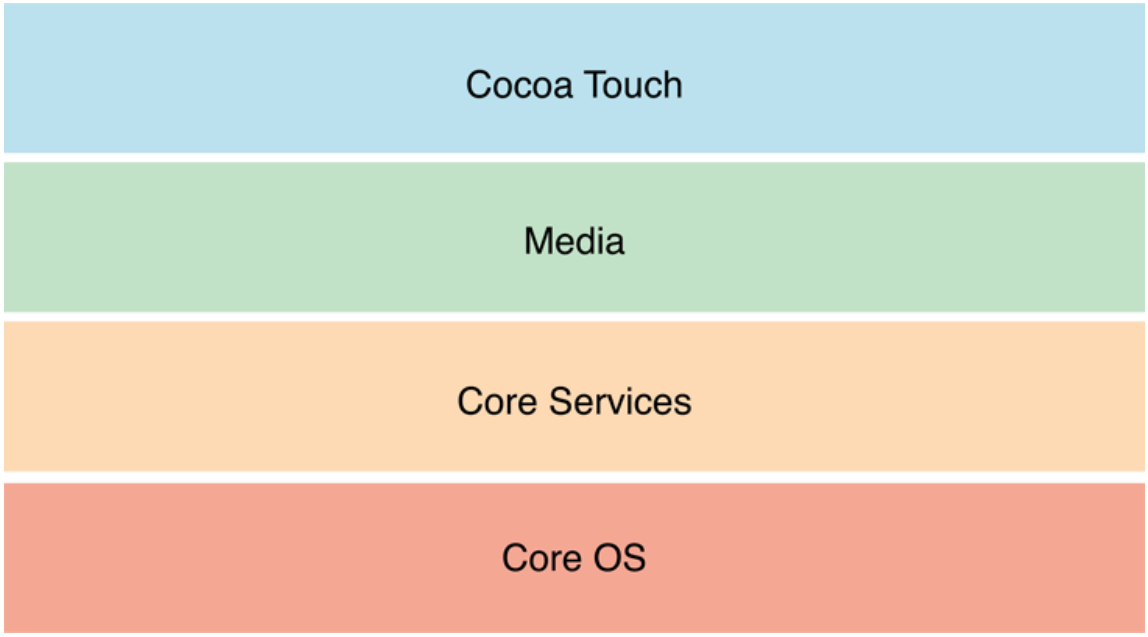
El desarrollo de aplicaciones para iOS se realiza por medio del iOS *Software Development Kit*. El lenguaje de desarrollo utilizado para crear dichas

[16] Fuente: *Mail Support*. https://support.apple.com/mail. Consulta: abril de 2017.
[17] Fuente: Safari. La mejor forma de ver los sitios web. https://www.apple.com/la/safari/ Consulta: abril de 2017.

aplicaciones es Objective-C. El desarrollo de estas aplicaciones se hace utilizando una serie de capas o *layers* que provee el sistema operativo, cada una incluye varios servicios de diferente grado de complejidad.

Las cuatro capas que provee la arquitectura de iOS son: *Cocoa Touch*, *Media*, *Core Services* y *Core OS*.

Figura 9. **Capas de la arquitectura iOS**



Fuente: *About the iOS Technologies*. https://developer.apple.com/library/content/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/Introduction/Introduction.html. Consulta: mayo de 2017.

La capa *Cocoa Touch* provee servicios o *frameworks* para definir la estructura básica de la aplicación y su apariencia. También provee el soporte y servicios para implementar *multitasking,* funcionalidad de entrada táctil, notificaciones y otros servicios de alto nivel.

La capa de nombre Media provee tecnologías y servicios para implementar diferentes tipos de multimedia en las aplicaciones. La capa *Core Services* provee varios servicios básicos para las aplicaciones. Entre estos, se encuentran los *frameworks Core Foundation* y *Foundation*. En estos *frameworks* se definen los tipos básicos de datos que todas las aplicaciones usan. Esta capa también provee soporte para otras características como los servicios de ubicación, iCloud, social media y servicios de red.

Por último, la capa *Core OS* provee varios servicios de bajo nivel que son utilizados por las otras capas de la arquitectura. Los servicios de esta capa pueden ser utilizados directamente por una aplicación en caso de que se requiera alguna funcionalidad especial, como en seguridad o comunicación con un accesorio externo de hardware.

1.2.2. Sistema Operativo Windows

1.2.2.1. Definición y Características

Microsoft Windows es una familia de sistemas operativos desarrollada y comercializada por la empresa Microsoft. En esta familia se encuentran sistemas operativos creados para su uso en servidores, computadoras personales, dispositivos móviles y consolas de videojuegos.

Es el sistema operativo más utilizado en el mundo en computadoras personales (*desktops*, *laptops* y *workstations*), Windows. La versión más reciente en la actualidad es Windows 10, lanzada al mercado en julio del año 2015.

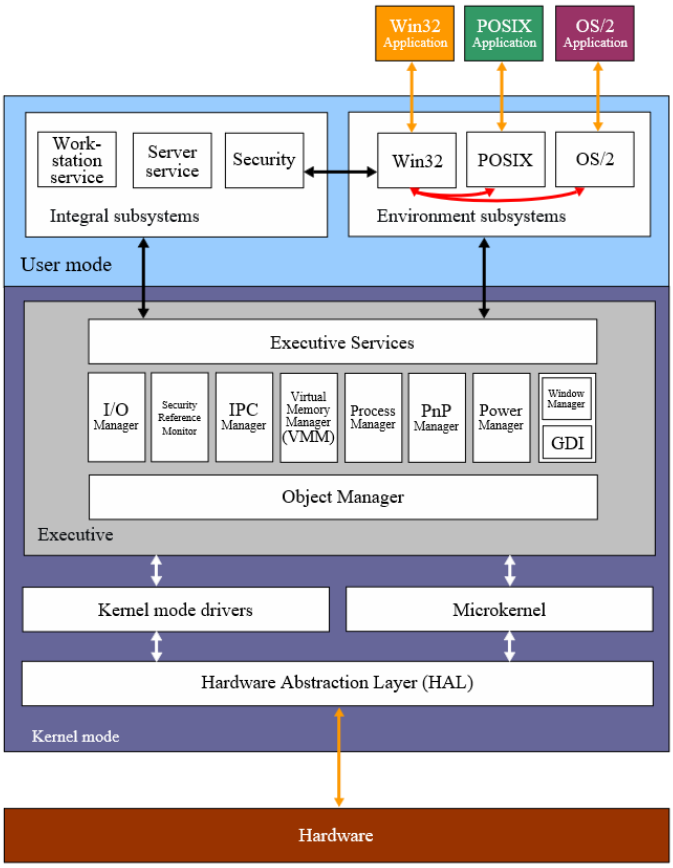
Tabla II. ***Market Share* de las versiones más utilizadas de Windows (abril 2017)**

OS	Versión del API
Windows XP	7.04%
Windows Vista	0.70%
Windows 7	48.50%
Windows 8	1.59%
Windows 8.1	6.96%
Windows 10	26.28%
Total	91.07%

Fuente: *Desktop Operating System Market Share, Abr 2017*. http://www.netmarketshare.com/operating-system-market-share.aspx?qprid=10&qpcustomd=0&qpsp=219&qpnp=1&qptimeframe=M. Consulta: mayo de 2017.

Windows está programado con los lenguajes C, C++ y Assembly, y se trabaja bajo un modelo de código cerrado (*closed source*) y código compartido (*shared source*). En su arquitectura utiliza un modelo de *kernel* híbrido el cual funciona en dos capas: modo usuario y modo *kernel* con varios módulos interactuando entre estas dos capas.

Figura 10. **Arquitectura de la familia Windows NT que comprende Windows 10**



Fuente: *Architecture of Windows NT.*
https://en.wikipedia.org/wiki/Hybrid_kernel#/media/File:Windows_2000_architecture.svg

La característica principal de la arquitectura en la última versión de este sistema operativo (Windows 10) es la implementación de un núcleo o *kernel* (que lleva por nombre Windows Core) el cual es compartido por todas las variantes más recientes de este sistema, independientemente de la plataforma o hardware en la cual se ejecuten. Anteriormente, cada versión de Windows poseía un *kernel* único con algunas características dependientes de la

plataforma, lo cual no permitía una compatibilidad completa entre las diferentes versiones de este sistema y dificultaba el trabajo de desarrollo y mantenimiento de aplicaciones para éste.

Windows Core está formado por una serie de APIs (*Application Programming Interfaces*) de bajo nivel que están organizadas bajo el concepto de contratos. Estos contratos están organizados, a su vez, para formar la *Universal App Plataform*, que constituye un *framework* bajo el cual se desarrollan actualmente las aplicaciones para los sistemas Windows. La utilización de este *framework* asegura que la aplicación que se haya desarrollado se ejecutará sin problemas en cualquier versión y plataforma en la cual se encuentre instalado este sistema operativo[18].

1.2.3. Herramientas adicionales a utilizar

- GIMP

El software GIMP (*GNU Image Manipulation Program*) es una herramienta de manipulación de fotografías multiplataforma gratuita de software libre. Con este software se pueden realizar tareas de manipulación de imágenes, como el retoque de fotografías, la construcción y composición de imágenes, conversión de formatos, renderizado, procesamiento de tareas por medio de scripts, etc. GIMP también posee la característica de ser extensible por medio de la instalación de extensiones y *plug-ins* para obtener funciones añadidas.[19]

[18] Fuente: *Architecture of Windows 10.* https://social.technet.microsoft.com/wiki/contents/articles/31048.architecture-of-windows-10.aspx. Consulta: mayo de 2017.

[19] Fuente: *Documentación Sitio Oficial GIMP.* http://docs.gimp.org/2.8/en/introduction.html. Consulta: mayo de 2017.

2. IMPLEMENTACIÓN DEL VIDEOJUEGO

2.1. Creación de la trama

El objetivo principal de este proyecto es crear un juego basado en la historia y cultura de Guatemala, por lo que el primer paso para crearlo fue establecer la trama que se representará en el nivel de demostración del juego. Para esto, se realizó una breve investigación sobre historia y cultura de Guatemala para determinar el período histórico y los elementos que formarán parte del juego. Este proceso de investigación se describe a continuación.

2.1.1. Investigación sobre historia y cultura de Guatemala

Consistió en la lectura de algunos documentos (libros y sitios web) en los cuales se describían los períodos más importantes de la historia del país.

Los períodos históricos investigados comprendieron la época prehispánica (con énfasis en la civilización maya), la época colonial, el período de independencia, el período del conflicto interno a finales del siglo XX y el período actual del país. Se seleccionaron estos períodos porque se consideran representativos de la historia del país.

2.1.2. Selección del período histórico, ambiente y personajes

Después del análisis de los períodos históricos y su factibilidad para plasmarlos en un videojuego, se seleccionó el período prehispánico debido a su importancia para la historia del país y por ser uno con los que más se identifica a Guatemala, sobre todo, por la cultura maya que floreció durante ese tiempo.

Al realizar la representación de la cultura maya y sus diversos elementos en el juego, se está creando una forma de traer al presente una parte importante de la historia del país, y de recordar la relevancia de una cultura milenaria en la historia no solo de Guatemala sino de todo el mundo.

La trama o historia que se desarrollará en el juego, es decir, los personajes que participarán en él y los objetivos que el jugador deberá cumplir, se creó con base en los personajes e historias del libro maya Popol Vuh, aunque no de una forma estrictamente literal a como aparecen allí. El acercamiento se hizo como en muchos videojuegos basados en personajes y hechos históricos, en los cuales se crea una historia original dentro del juego basada en estos eventos y personajes. Esto permite crear un contenido original y creativo, a la vez que se retienen los elementos educativos.

2.2. Selección del tipo de videojuego a crear

Una vez definida la base histórica para crear el videojuego, se eligió el tipo o género de videojuego que se desarrollará. Los factores de decisión más importantes fueron la adaptabilidad del género a la historia y ambiente que se querían plasmar en el juego y la plataforma de publicación (dispositivos móviles con sistemas operativos Android y iOS, y dispositivos con sistema operativo Windows).

Con base en los elementos históricos de la cultura maya que se querían plasmar en el juego, se pensó, en primera instancia, en desarrollar un juego de estrategia, ya que se han utilizado para crear juegos con elementos históricos; sin embargo, este tipo de juegos, generalmente requieren de un esquema de controles que no es apropiado para una plataforma móvil, lo cual es una limitante.

Luego, se consideró el género de acción-aventura para crear el juego. Bajo este paradigma se podría crear un mundo en tercera dimensión representando diversos elementos de la cultura maya (como pirámides y otras estructuras arquitectónicas propias de esta cultura) y desarrollar objetivos de combate y de exploración, como parte del juego. Para hacer el juego adaptable a las plataformas Android y iOS, se consideró que se podría utilizar la variante en primera persona de este género de juegos para simplificar los controles de la aplicación y hacerla idónea para su utilización en teléfonos móviles con pantallas táctiles. Además de estas ventajas, también se suma la popularidad de este tipo de juegos y su especial atractivo que tienen con los videojugadores.

2.3. Metodología de desarrollo

En la siguiente sección se presenta el documento de Especificación de Requisitos de Software (ERS) perteneciente a la aplicación desarrollada.

2.3.1. Especificación de Requisitos de Software (ERS)

- Introducción
    - Propósito

El propósito de este documento es dar a conocer las especificaciones y detalles sobre las cuales se desarrollará la demostración de un videojuego con base en la historia y cultura de Guatemala. En este documento se especifican los requisitos que debe cumplir la aplicación y se explica su arquitectura general.

- Alcance del proyecto

El objetivo de este proyecto es desarrollar una demostración de un videojuego cuyo ambiente e historia sea basado en la cultura guatemalteca. La demostración consiste en que el desarrollo incluirá solamente un pequeño nivel donde se expondrán las características básicas de funcionamiento y contenido que pudieran implementarse luego en una versión completa del juego.

El videojuego será del género de primera persona y estará disponible para teléfonos móviles de últimas generaciones ejecutando los sistemas operativos Android y iOS, así como para dispositivos que utilicen el sistema Microsoft Windows.

- Descripción General

  - Perspectiva del producto

El producto será una aplicación independiente (*stand-alone*) para los sistemas operativos Android, iOS y Microsoft Windows, en la cual el usuario podrá interactuar con un mundo virtual en tres dimensiones, basado en la cultura maya guatemalteca. El usuario podrá interactuar y movilizarse dentro de este mundo usando como dispositivo de entrada la pantalla táctil de su dispositivo móvil y deberá completar varios objetivos para avanzar en la historia del juego. El producto está orientado a usuarios con una mediana experiencia en el uso de videojuegos.

En la primera versión de esta aplicación se desarrollará un pequeño nivel de demostración en el cual se expondrán todas las ideas básicas que formarán

un videojuego completo en futuras versiones de la aplicación. Se estima desarrollar en esta demo un tiempo de juego de aproximadamente 10 minutos.

- Funcionalidad del producto

En resumen, las funciones que se implementarán en la aplicación son las siguientes. El detalle de cada una de ellas se expondrá en la sección de requisitos de este documento:

Mostrar Menú Principal
Iniciar Juego
Movilizar personaje
Interactuar con el ambiente
Mostrar pantalla de juego completo
Mostrar Instrucciones
Salir del Juego

- Características de los Usuarios

  - Videojugadores experimentados

Este tipo de usuarios tienen varios años de experiencia en el uso de videojuegos de este tipo y conocen su forma de funcionamiento. La curva de aprendizaje para este tipo de usuarios es muy corta. Están por lo regular en el rango de los 20 – 35 años.

- Videojugadores casuales

Este tipo de usuarios utiliza videojuegos ocasionalmente y aunque tiene cierto nivel de experiencia es probable que desconozca el funcionamiento de los juegos en primera persona implementados en un dispositivo móvil. La curva de aprendizaje de este tipo de usuario es de dificultad media. El rango de edad es aproximadamente de 25 – 45 años.

- No videojugadores

Son usuarios que rara vez usan videojuegos en dispositivos móviles o en algún otro tipo de plataforma. Desconocen casi en su totalidad la forma de jugar un videojuego de primera persona. La curva de aprendizaje de este tipo de usuarios puede ser un poco prolongada y con un cierto nivel de dificultad elevado. El rango de edad de este tipo de usuarios es de 35 años en adelante aproximadamente.

- Restricciones

Para la versión del juego que se desarrollará para teléfonos móviles, debido a la naturaleza de su dispositivo de entrada (pantallas táctiles), la lista de controles o comandos que se pueden programar en el juego es bastante limitada, igual que su complejidad. Contrastan con la flexibilidad y variedad de controles que existen en una *PC* o en una consola de videojuegos. Por eso, la lista de acciones que se podrán ejecutar dentro del juego será corta y se adaptarán a los comandos que puede un usuario ejecutar en un teléfono con pantalla táctil. La versión del juego para la plataforma Microsoft Windows tendrá la posibilidad de incorporar controles más complejos basados en el uso del teclado y el *mouse*, pero se tratará de crear una experiencia de juego lo más

uniforme posible para las tres plataformas de desarrollo (Android, iOS y Windows).

Otra restricción relacionada con la utilización de un dispositivo móvil como plataforma es el escaso poder de procesamiento que poseen el *CPU*, memoria y *GPU* en comparación con una plataforma como las *PC* o las consolas de última generación, por lo que la cantidad de contenido que se puede incluir en un nivel del juego y su calidad gráfica es limitada. Por lo anterior, igual que como ocurre con los controles de entrada, la versión del juego para Windows tendrá gráficos superiores a los de la versión para dispositivos móviles, pero se intentará que la experiencia visual del usuario no cambie demasiado entre las distintas plataformas.

- Suposiciones y dependencias

  No aplicable.

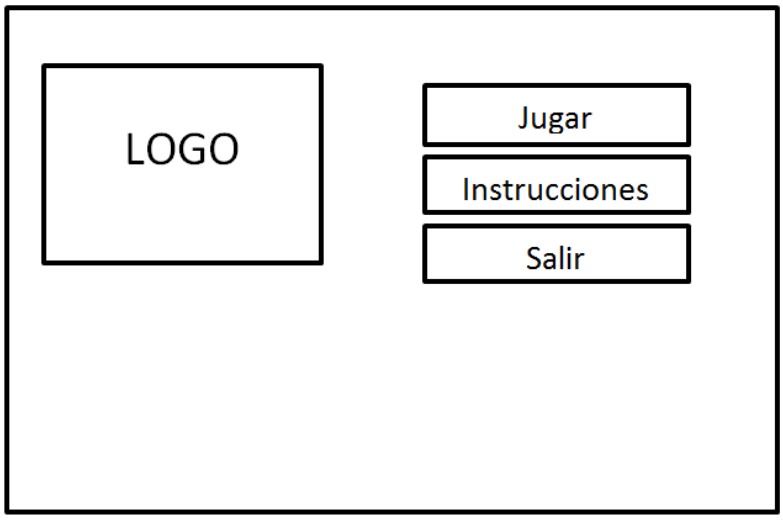
- Evolución previsible del sistema

En un futuro cercano, se espera un desarrollo completo de todos los niveles del videojuego, basándose en las características presentadas en el nivel de demostración que se desarrollará en esta versión de la aplicación. También se espera que el juego pueda ser portado a otras plataformas además de los dispositivos móviles basados en Android y iOS, y el sistema operativo Windows.

- Requisitos específicos
    - Requisitos comunes de las interfaces
        - Interfaces de usuario

Menú principal: la interfaz de usuario del menú principal aparecerá al iniciarse la ejecución de la aplicación. En esta interfaz se mostrará el logo representativo del videojuego, y tres botones interactivos que pueden ser utilizados por el usuario: “Jugar”, “Instrucciones” y “Salir”. Al presionar el usuario el botón “Jugar” entrará al juego en sí, al presionar el botón “Instrucciones” se mostrará en pantalla un texto explicando los controles del juego; por último, al presionar el usuario sobre el botón “Salir”

Figura 11. **Esquema menú principal de la aplicación**

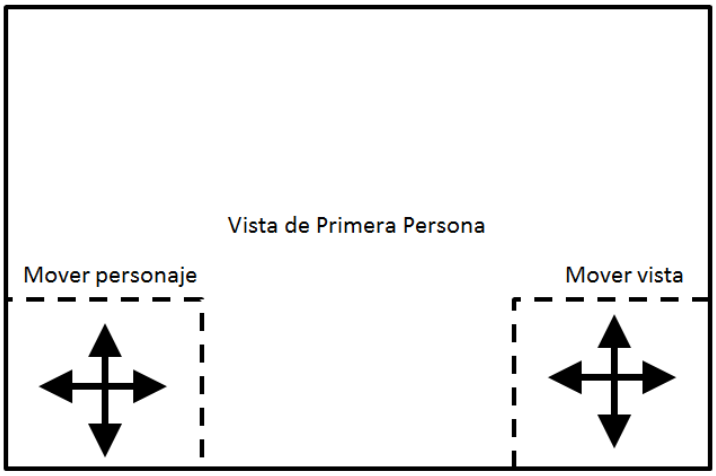


Fuente: elaboración propia, empleando Microsoft Word.

- Interfaz del juego

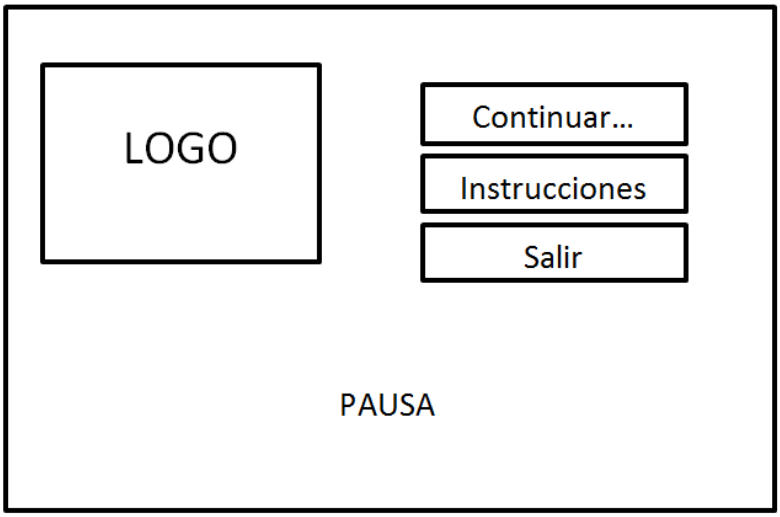
Se presentará el juego en modo de pantalla completa sobre la cual el usuario podrá interactuar utilizando la pantalla táctil del teléfono. El usuario podrá presionar el botón “Regresar” de su teléfono (en el caso de estar utilizando un dispositivo con sistema Android) para mostrar un menú dentro del juego (un menú de pausa) muy similar al menú principal antes descrito. Este menú presentará tres botones con las opciones: “Continuar…”, “Instrucciones” y “Salir”. El botón “Regresar” existe en la gran mayoría de teléfonos que utilizan el sistema operativo Android. En el caso de Windows para ingresar a este menú se utilizará la tecla Esc.

Figura 12. **Esquema interfaz del juego**



Fuente: elaboración propia, empleando Microsoft Word.

Figura 13. **Esquema interfaz menú de pausa**



Fuente: elaboración propia, empleando Microsoft Word.

- Interfaces de *hardware*

No aplica.

- Interfaces de *software*

Esta aplicación no interactuará ni tendrá interfaces con otras aplicaciones. Funcionará de forma independiente.

- Interfaces de comunicación

El juego desarrollado será de un solo jugador y no poseerá funcionalidad en línea, por lo que no se implementarán ningún tipo de interfaces de comunicación.

- Requisitos funcionales

Identificador: R1
Nombre: Movilizar personaje
Entradas: para la versión para móviles (Android y iOS): comandos ingresados por el usuario utilizando gestos de arrastre en el área inferior izquierda de la interfaz de usuario del juego. Para la versión para el S.O. Windows: comandos ingresados por medio del teclado.
Salidas/resultados: movimiento del personaje (cambio de posición) dentro del escenario creado en tres dimensiones. El personaje podrá moverse hacia adelante, atrás, derecha e izquierda dependiendo de los controles ingresados por el usuario sobre la pantalla táctil o teclado en ese momento en específico.

Identificador: R2
Nombre: Rotar vista del personaje
Entradas: para la versión para móviles (Android y iOS): gestos de arrastre realizados por el usuario sobre la pantalla táctil del dispositivo móvil en el área inferior derecha de la interfaz de usuario. Para la versión para el S.O. Windows: comandos ingresados por medio del mouse.
Salidas/resultados: la vista de cámara de primera persona del personaje principal del juego debe moverse de acuerdo con la dirección del gesto de arrastre ingresado por el usuario. El personaje deberá poder alzar la vista, bajar la vista y rotarla hacia la derecha o izquierda.

Identificador: R3
Nombre: Interactuar con el medio
Entradas: para la versión para móviles (Android y iOS): el usuario da un toque o *tap* a la pantalla táctil del teléfono móvil cuando se encuentra dentro del juego.

Para la versión para el S.O. Windows: el usuario utiliza el clic izquierdo del *mouse*.

Salidas/resultados: dependiendo de la vista que tenga el personaje controlado por el usuario en el juego en el momento específico en el que se ejecute el comando de interacción (*tap* o clic izquierdo en el *mouse*) deberán de ocurrir distintos eventos, tales como lanzar algún objeto, recoger un objeto, etc.

Identificador: R4

Nombre: Completar objetivo

Entradas: el personaje controlado por el usuario ejecuta alguna acción definida como un objetivo del nivel.

Salidas/resultados: se despliega en pantalla un texto que indica la obtención del objetivo. Se marca el siguiente objetivo como activo dentro de la aplicación.

Identificador: R5

Nombre: Completar Nivel del Juego.

Entradas: el usuario completa el último objetivo del nivel.

Salidas/resultados: se despliega en pantalla un texto haciendo referencia a que se ha completado el nivel y que el juego ha finalizado. Se despliega luego el menú principal del juego.

Identificador: R6

Nombre: Iniciar el juego.

Entradas: el usuario presiona sobre el botón “Jugar” en el menú principal de la aplicación.

Salidas/resultados: se despliega la vista de primera persona del personaje principal del juego. El usuario puede comenzar a jugar.

Identificador: R7

Nombre: Mostrar Instrucciones.

Entradas: el usuario presiona sobre el botón “Instrucciones” en el menú principal de la aplicación.

Salidas/resultados: se despliega un texto en pantalla explicando los controles del juego. El fondo de pantalla sigue siendo el mismo del menú principal. Al final del texto se despliega un botón que diga “Regresar” el cual al ser presionado despliega nuevamente el menú principal.

Identificador: R8

Nombre: Salir del juego.

Entradas: el usuario presiona sobre el botón “Salir” en el menú principal de la aplicación.

Salidas/resultados: se termina la ejecución de la aplicación.

- Requisitos no funcionales
    - Requisitos de rendimiento

Al tratarse la aplicación de un juego de primera persona en tercera dimensión es importante que tenga un buen desempeño gráfico y que las imágenes sean renderizadas en forma fluida para no degradar la experiencia del videojugador. Por ello, el *framerate* del juego debe estar en todo momento sobre los 30 *frames* por segundo (fps).

Aunque el parámetro anterior dependerá en gran parte de la capacidad de hardware del teléfono o computadora en el que se ejecute la aplicación, se buscará desarrollar el juego con un nivel de requerimiento gráfico no

excesivamente elevado para que sea utilizable en teléfonos móviles y computadoras de media a alta capacidad.

- Requisitos de seguridad

Los requisitos de seguridad para esta aplicación son mínimos ya que en ella no se almacenarán o manejarán datos de importancia. La única preocupación relacionada con la seguridad es la posibilidad de que alguna vulnerabilidad en el software pueda ser aprovechada por un atacante para acceder a otras funciones del sistema operativo. Dado que la aplicación utilizará un motor de juegos desarrollado por una tercera parte, la existencia de vulnerabilidades dependerá de los desarrolladores de dicho motor de juegos. Debido a esto, el desarrollo de este proyecto carece de requisitos de seguridad que se le apliquen.

- Requisitos de fiabilidad

Se espera una aplicación que presente una alta fiabilidad, una vez se ejecute en un sistema de hardware capaz de soportarla (que cumpla con los requerimientos mínimos que se especificarán posteriormente para la aplicación). Se espera que en Android la aplicación funcione de forma aceptable con las especificaciones promedio de hardware encontradas actualmente en los dispositivos que utilizan este sistema operativo. iOS es un sistema exclusivo para dispositivos creados por Apple y cuentan con un hardware de altas prestaciones, se espera que la aplicación funcione con una alta fiabilidad en todos los dispositivos que ofrece el mercado que utilicen este sistema operativo. La versión del juego para la plataforma Windows se espera que funcione de forma correcta para dispositivos que puedan ejecutar sin problemas de la versión Windows 7 en delante de este sistema operativo.

La aplicación debe ser probada intensivamente para buscar *bugs* o errores de programación en el juego antes de publicarla a los usuarios finales. Se deberá probar la existencia de *bugs* siguiendo rutinas de juego que no sean las que seguiría un usuario normal que únicamente está interesado en completar los objetivos previstos durante el juego.

Se espera que en la versión final de la aplicación ocurran errores de funcionamiento en un máximo de una vez por cada cien (100) ejecuciones de la aplicación.

- Requisitos de disponibilidad

No aplicable.

- Requisitos de mantenibilidad

Las rutinas de mantenimiento o de actualización de la aplicación (corrección de errores) se realizarán de acuerdo con los reportes de errores o retroalimentación recibidos por los usuarios. Esto es aplicable para la versión del juego que se desarrollará en este proyecto. En futuras versiones que expandan el nivel de demostración a un juego completo se deberá planificar un nuevo sistema o cronograma de mantenimiento de esta.

- Requisitos de portabilidad

La aplicación debe ser desarrollada para su publicación final en las plataformas Android, iOS y Windows, por lo que debe cumplir los requerimientos de portabilidad para el software desarrollado para cada una de estas plataformas. En el caso de Android, la aplicación deberá ser desarrollada

para ser portable entre las diversas versiones de este sistema más utilizadas hasta su momento de publicación, que actualmente comprenden las versiones 4.1 - 4.3 (Jelly Bean), 4.4 (KitKat), 5.0 – 5.1 (Lollipop) y 6.0 (Marshmallow). Para el caso de iOS, la aplicación deberá funcionar únicamente en la última versión de este S.O. a la fecha: iOS 10. Por último, para Windows, la aplicación deberá poder utilizarse en dispositivos que ejecuten desde la versión Windows 7 hasta la más reciente (Windows 10).

También se requiere que la aplicación sea desarrollada bajo un motor de juegos que permita su publicación en otras plataformas diferentes a Android, iOS y Windows, como Playstation, XBOX o consolas recientes de la empresa Nintendo.

- Otros requerimientos

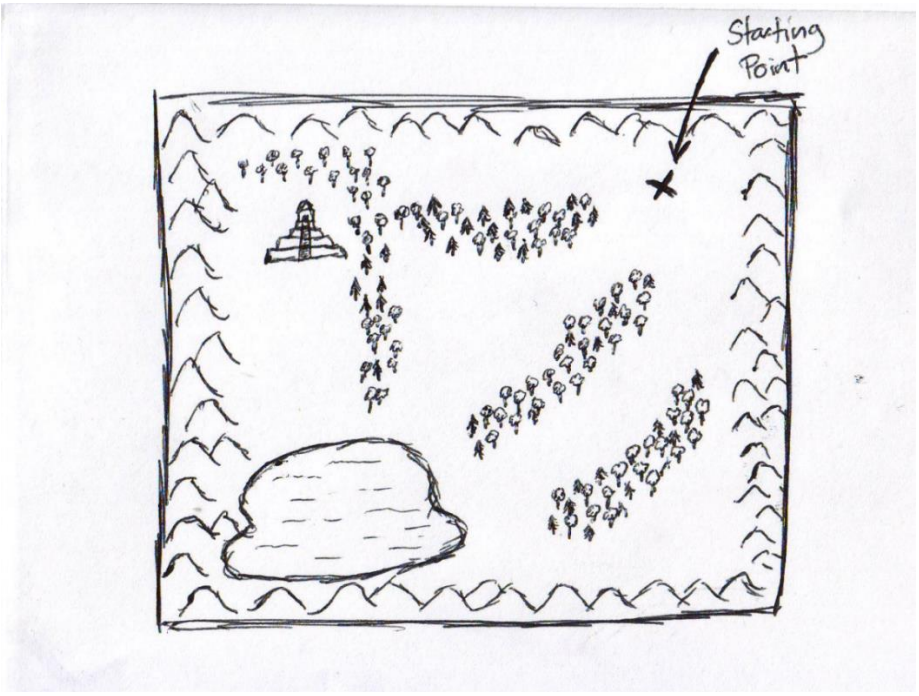
Ninguno.

2.4. Arte del juego

Para iniciar el proceso de desarrollo del arte gráfico del juego, se realizaron algunos bosquejos o dibujos de lo que más tarde se plasmaría y sería modelado en el mundo en tercera dimensión representado dentro del juego.

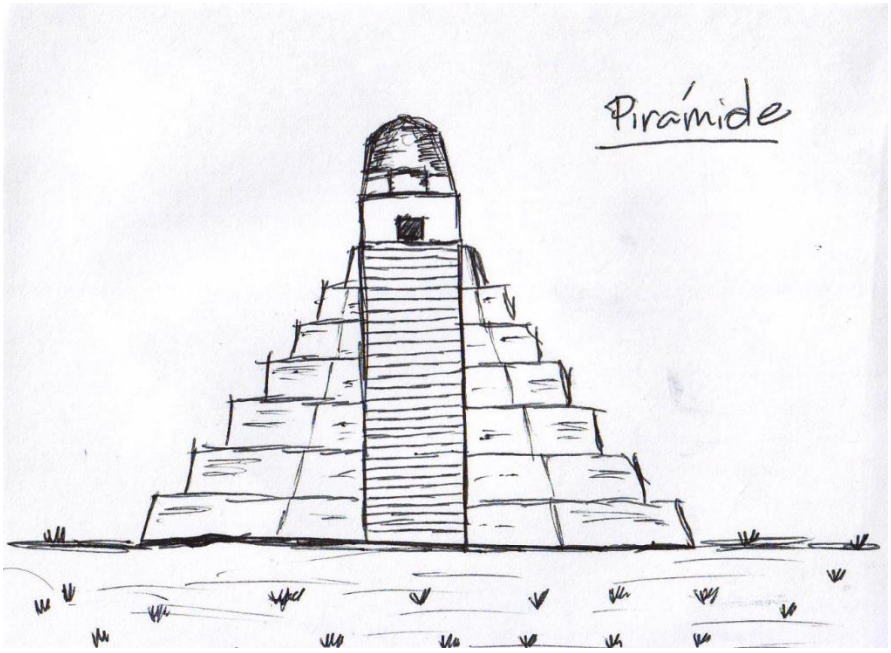
En las figuras se muestran algunos de estos dibujos realizados. Luego, se tomaron como base para programar o modelar los objetos o elementos dentro del juego.

Figura 14. **Dibujo del escenario**



Fuente: elaboración propia.

Figura 15. **Dibujo de la pirámide**



Fuente: elaboración propia.

2.5. Modelado gráfico del arte

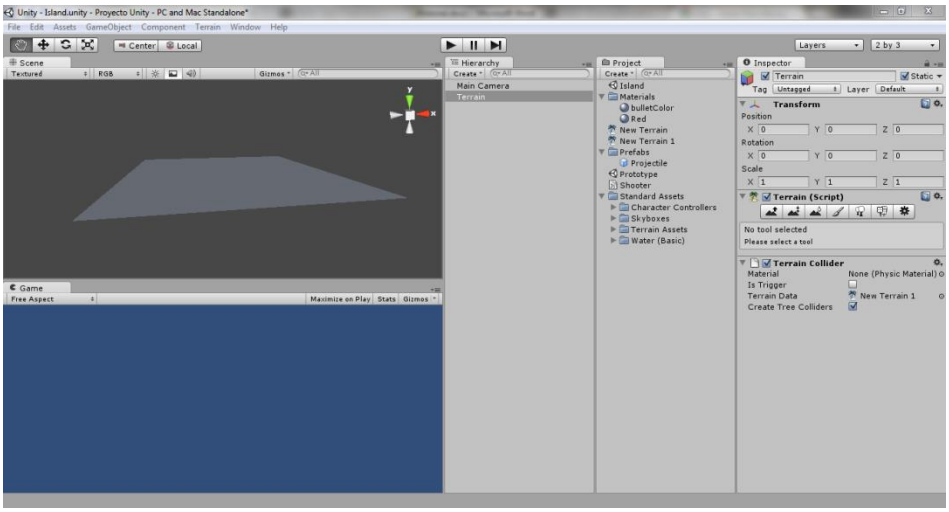
Luego de haberse realizado los esquemas o bosquejos de los elementos u objetos gráficos que forman parte del nivel de demostración del videojuego, se modelaron y construyeron estos elementos en forma de gráficas computarizadas en tres dimensiones. A continuación, se describe a detalle este proceso.

2.5.1. Escenario

Se creó un nuevo proyecto dentro de la herramienta Unity y, dentro de este nuevo proyecto, se creó una nueva escena. El primer elemento del escenario creado fue el terreno. Para ello, se utilizó la herramienta *Create Terrain* ubicada en el menú *Terrain* de la barra de herramientas de Unity.

Luego de haberse creado el terreno básico, es necesario configurar algunos parámetros para determinar la forma que tendrá este terreno dentro del juego. El primer parámetro modificado fue el tamaño del terreno. Para esto se utilizó el comando *Set Resolution* y se ingresaron los valores de 500 para el largo (*length*) y el ancho (*width*). Estos valores representan metros dentro de la herramienta, por lo que el terreno base tendrá, entonces, 500 metros de largo por 500 metros de ancho.

Figura 16. **Terreno básico**

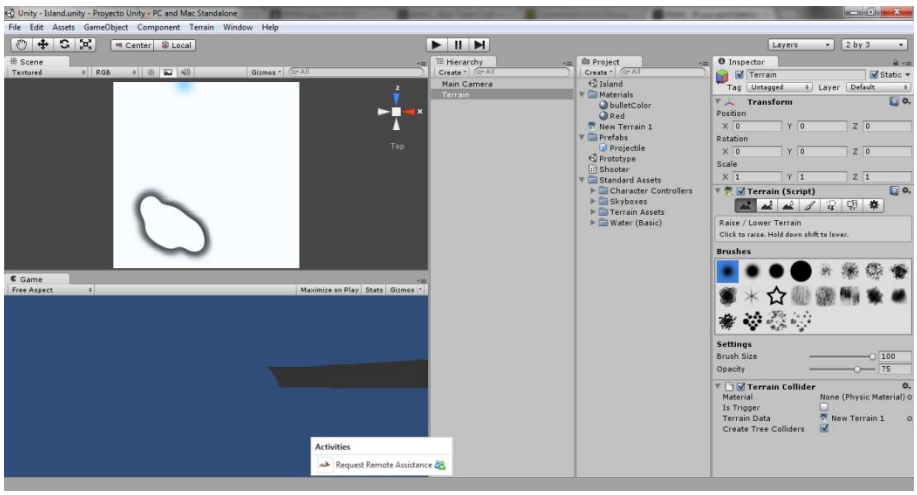


Fuente: elaboración propia, empleando Unity.

Por defecto, los terrenos en Unity tienen una elevación estándar de 0 metros, pero en el caso de este proyecto, se elevó la altura estándar del terreno a los 30 metros para crear hundimientos. Para realizar esto se utilizó la herramienta *Flatten Heightmap* del menú *Terrain* de Unity, ingresando un valor de 30 (metros) en la variable *Height* de esta herramienta.

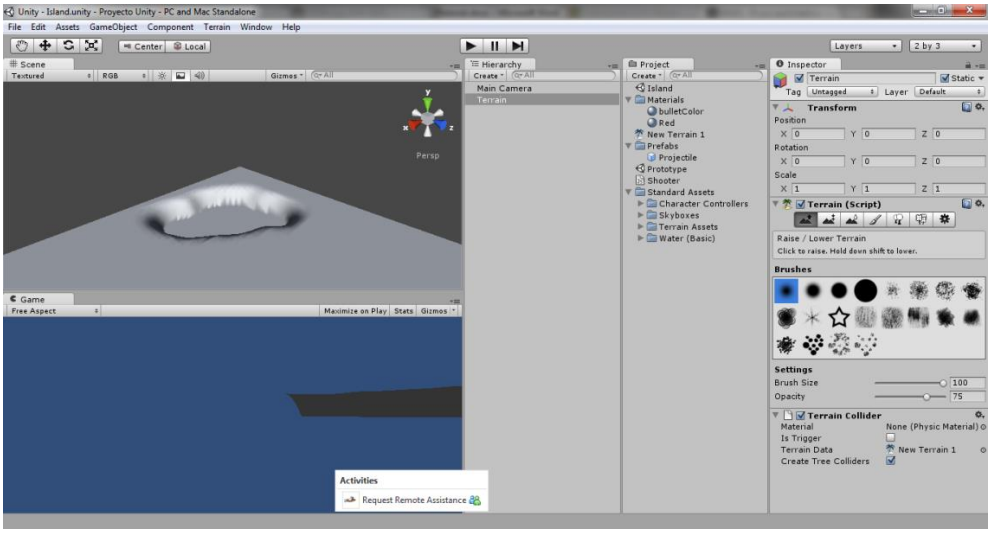
Una vez se ha elevado el terreno, se realiza el contorno del lago que se encuentra en el esquema del escenario (Figura 14). Para llevar a cabo esto, se utilizó la herramienta *RaiseHeight*, con la cual se realizan elevaciones y hundimientos sobre un terreno y posee varias opciones configurables. Estas opciones permiten especificar el estilo de “pincel” para realizar las elevaciones/hundimientos. Para realizar el lago se utilizaron las opciones *Brush Size* = 100, *Opacity* = 75 y se utilizó la opción de inversión de esta herramienta (activada al presionar la tecla *Shift* del teclado mientras se realizan los trazos) para hundir el terreno. El resultado se muestra en las dos siguientes figuras:

Figura 17. **Creación del lago (vista superior)**



Fuente: elaboración propia, empleando Unity.

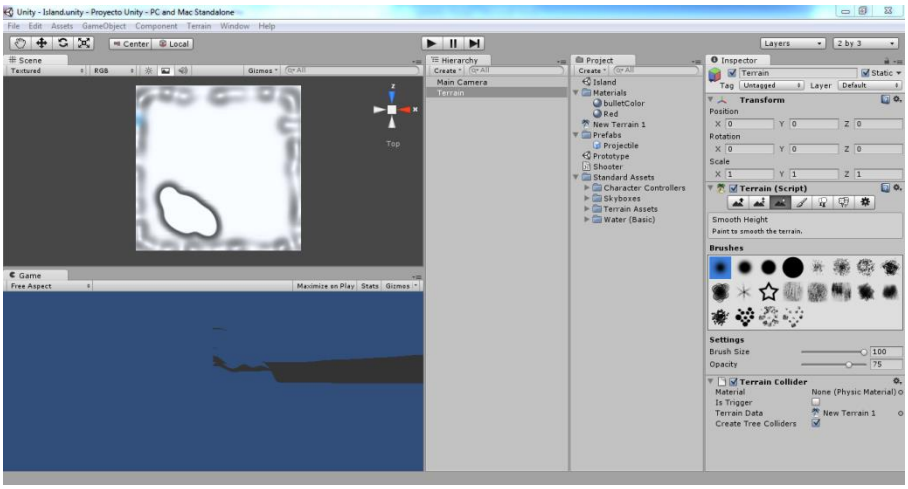
Figura 18. **Creación del lago (vista de perspectiva)**



Fuente: elaboración propia, empleando Unity.

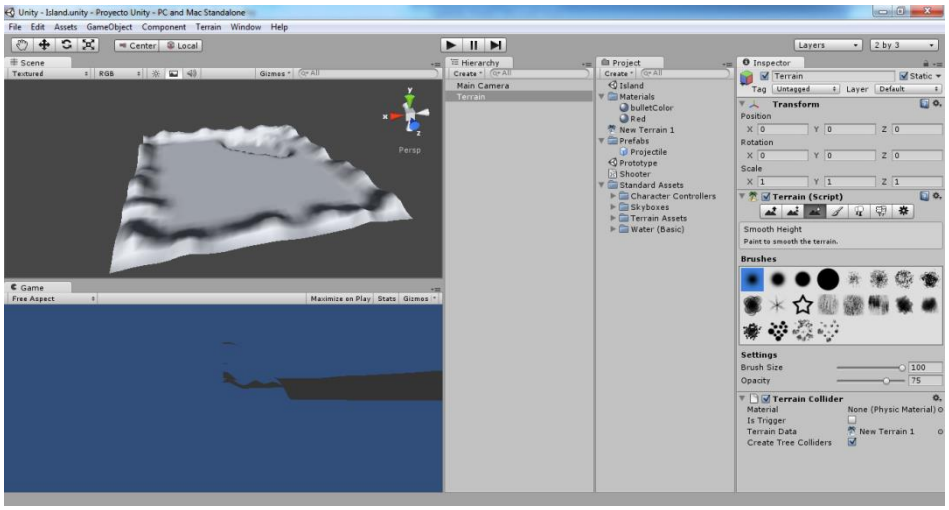
También con la herramienta *Raise Height* se crearon algunas montañas en los bordes del terreno para definir límites dentro de los cuales los personajes del videojuego puedan interactuar. El resultado se muestra en las siguientes figuras:

Figura 19. **Creación de montañas (vista superior)**



Fuente: elaboración propia, empleando Unity.

Figura 20. **Creación de montañas (vista de perspectiva)**



Fuente: elaboración propia, empleando Unity.

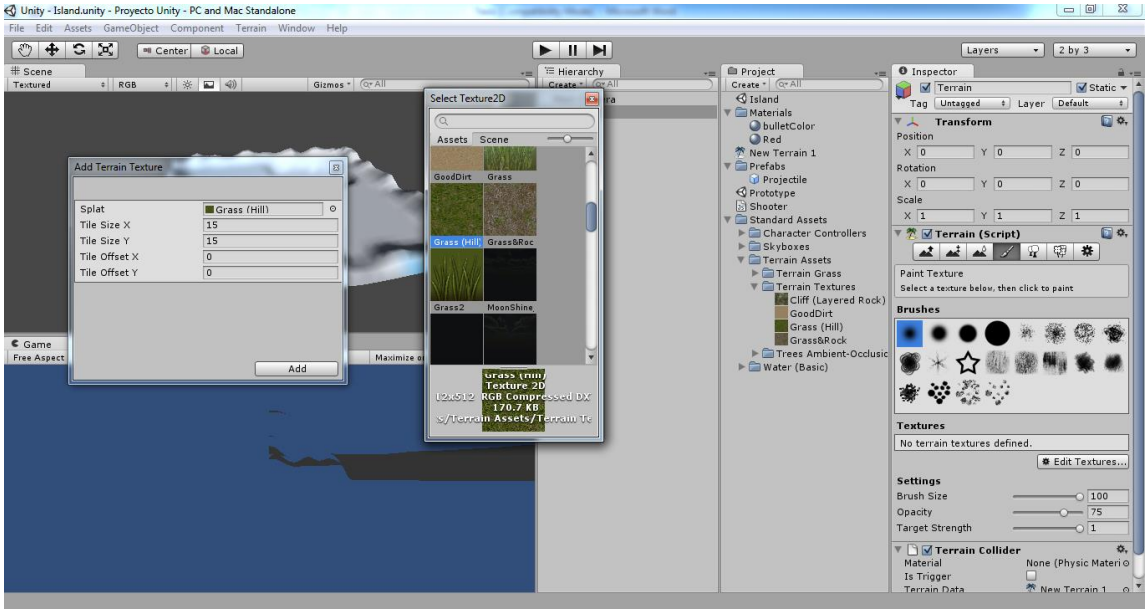
Luego, se utilizó la herramienta *Smooth Height* para suavizar algunos de los bordes de las montañas y orillas del lago.

Con esto se concluyó la creación de la forma básica del terreno. El siguiente paso consistió en añadir texturas al terreno. La primera textura que se añade a un terreno en Unity cubre por defecto toda el área de este, por lo que es necesario cerciorarse de que esta textura dominante represente la mayor parte del tipo de terreno que estamos creando.

Las texturas que se utilizaron para decorar el terreno en este videojuego provienen del paquete estándar de recursos que provee Unity por defecto. Este paquete de recursos fue importado al crear el proyecto anteriormente.

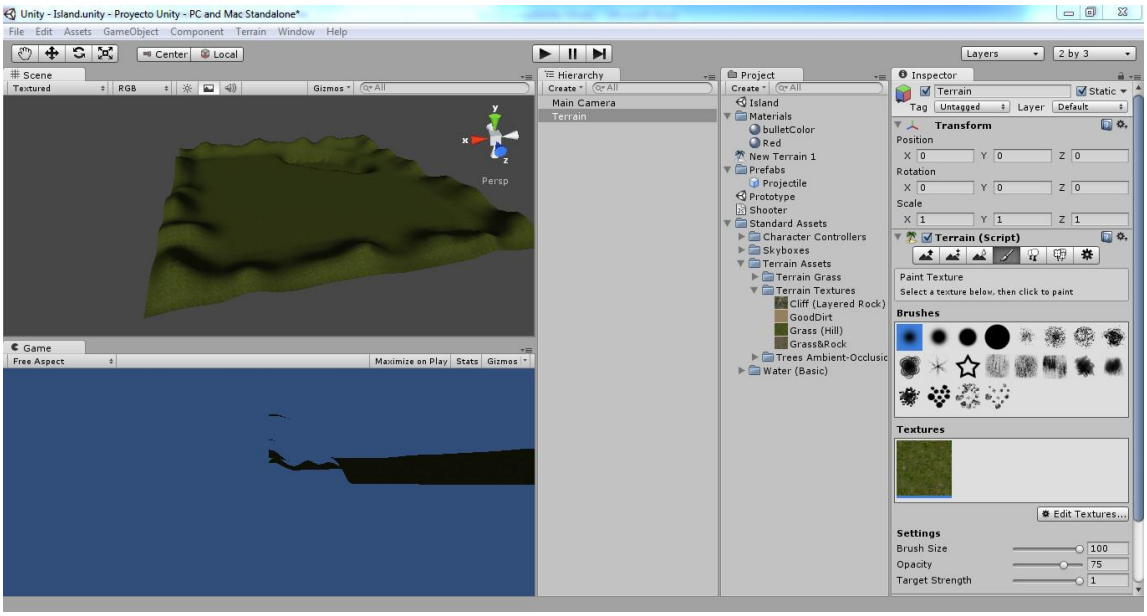
Para añadir la primera textura se utilizó la herramienta *Paint Texture* con la cual se seleccionó la textura *Grass* (*Hill*) del paquete de texturas de Unity para crear un terreno cubierto por grama como terreno base.

Figura 21. **Selección de textura base para el terreno**



Fuente: elaboración propia, empleando Unity.

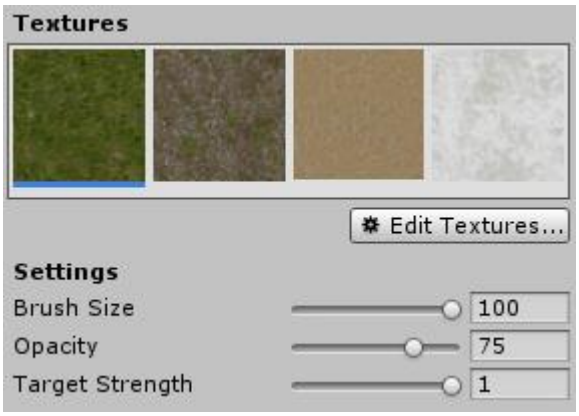
Figura 22. **Terreno cubierto con textura de grama**



Fuente: elaboración propia, empleando Unity.

El siguiente paso consistió en agregar tres texturas más a la paleta de texturas utilizadas para pintar el terreno siguiendo el procedimiento utilizado anteriormente para añadir la textura de grama. Las tres texturas añadidas fueron *Grass&Rock*, *GoodDirt* y *Cliff* (*Layered Rock*). Las propiedades de las texturas *Grass&Rock* y *GoodDirt* fueron configuradas en sus valores por defecto, mientras que los valores de *Tile Size* X y *Tile Size* Y de la textura Cliff fueron cambiados a 70 en ambos casos.

Figura 23. **Texturas añadidas a la paleta**

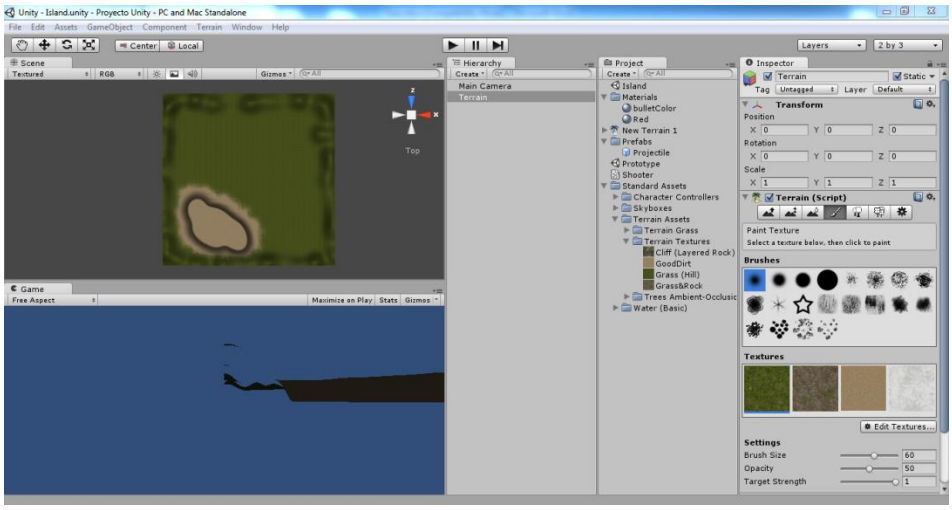


Fuente: elaboración propia, empleando Unity.

Luego, se pintaron algunas partes del terreno con la textura *GoodDirt* para simular partes arenosas en el escenario. Los valores de configuración para pintar la textura fueron configurados con los siguientes valores: *Brush Size* = 60, *Opacity* = 50 y *Target Strength* = 1. El tipo de pincel (o *brush*) utilizado fue el primer pincel de la paleta de Unity, también se usó el quinto pincel para realizar algunos detalles en el contorno.

El área del terreno que fue pintada con esta textura fue la superficie debajo del lago y sus alrededores. El resultado se muestra en la siguiente figura:

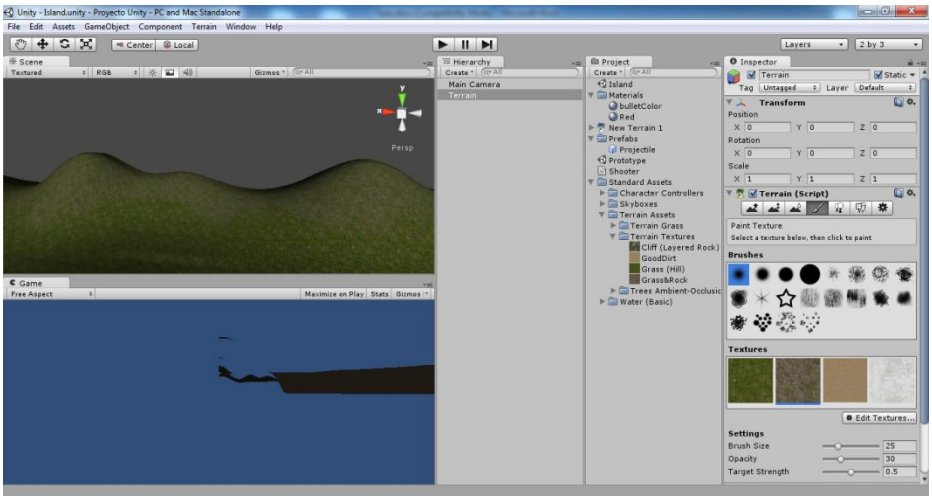
Figura 24. **Superficie y alrededores del lago con textura aplicada**



Fuente: elaboración propia, empleando Unity.

La siguiente textura en ser aplicada al terreno fue *Grass&Rock*. Los valores del pincel para esta textura se configuraron en: *Brush Size* = 25, *Opacity* = 30 y *Target Strength* = 0.5. Esta textura fue aplicada a la cima de las partes montañosas que se encuentran en el contorno del escenario. El resultado se muestra en la siguiente figura:

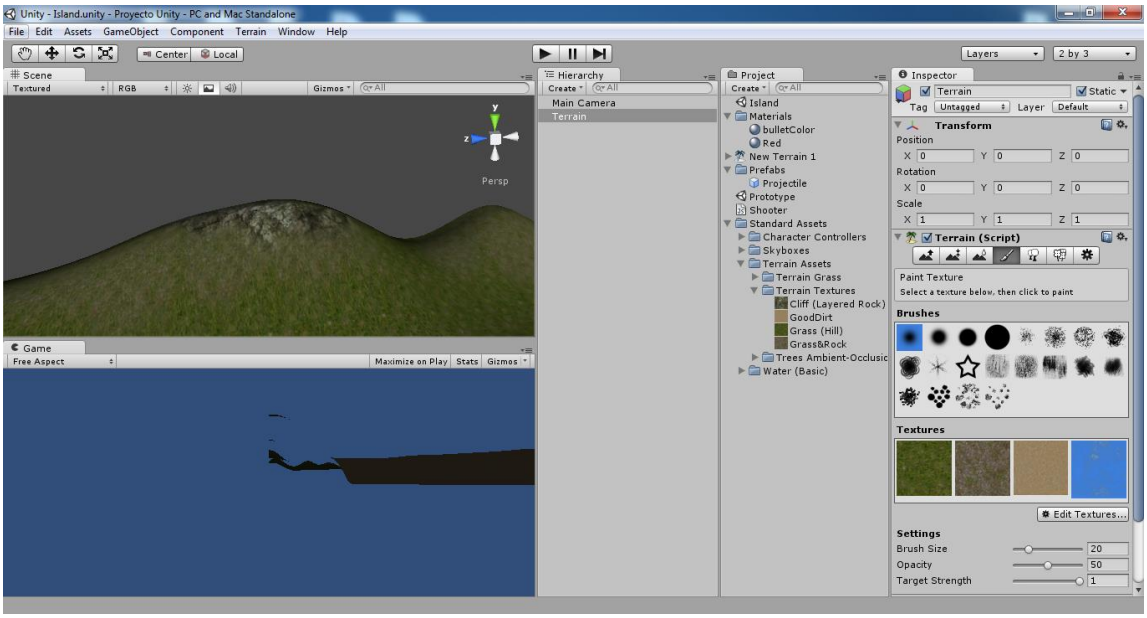
Figura 25. **Textura de grama y roca aplicada**



Fuente: elaboración propia, empleando Unity.

Para finalizar, se aplicó la textura *Cliff* (*Layered Rock*) a algunas de las partes altas de las montañas que rodean el área de juego. Los valores de configuración que se utilizaron para aplicar la textura fueron los siguientes: *Brush Size* = 20, *Opacity* = 50 y *Target Strength* = 1. El resultado se muestra a continuación:

Figura 26. **Textura de roca aplicada a la cima de una montaña**



Fuente: elaboración propia, empleando Unity.

- Incorporación de árboles a la escena

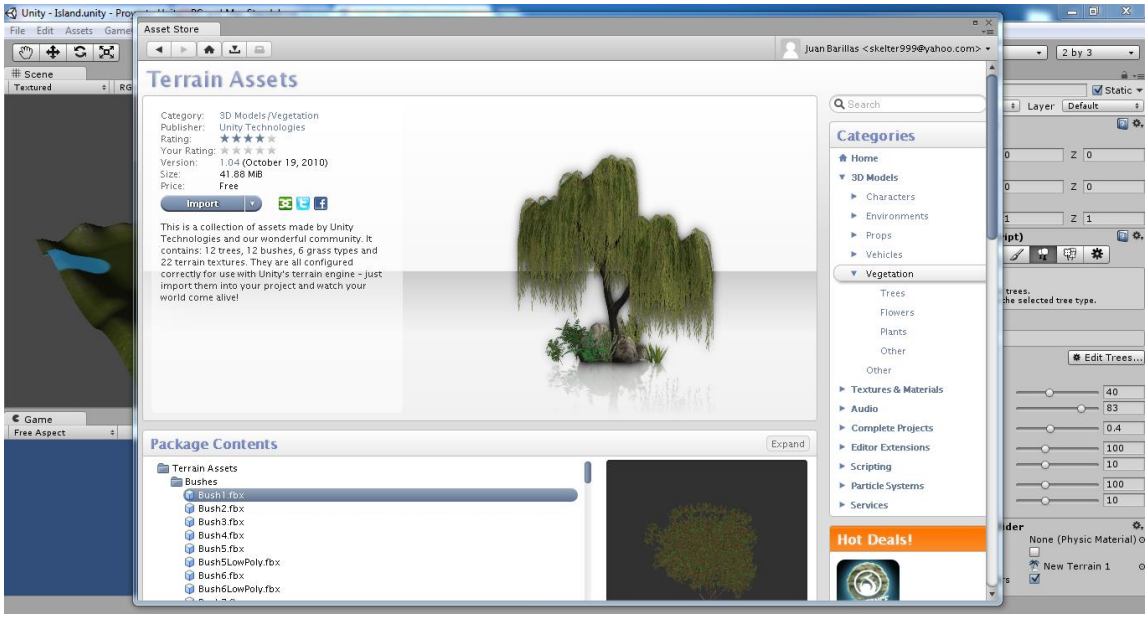
El siguiente paso en la creación del escenario consistió en añadir árboles al terreno. Para esto, se descargó un paquete con varios modelos de árboles y arbustos creado por Unity Technologies y disponible en la tienda de *Assets* de la herramienta. Para ingresar a la tienda, se utiliza la opción *Asset Store*, ubicada en el menú *Window*.

Los recursos que se pueden descargar de esta tienda están organizados en diferentes categorías y algunos de ellos son de pago mientras que otros son libres. El recurso descargado que se utilizó en este proyecto se encuentra en la categoría *3D Models* → *Vegetation* y tiene por nombre *Terrain Assets*. Este

paquete incluye 12 modelos de árboles, 12 modelos de arbustos, 6 tipos de grama y 22 texturas de terreno en él.

Para importar este paquete al proyecto en Unity basta con seleccionar la opción *Import* ubicada en la página del paquete en la *Asset Store* y éste será descargado e incorporado automáticamente al proyecto que se está trabajando en Unity.

Figura 27. ***Asset Store* en Unity y paquete de árboles descargado**

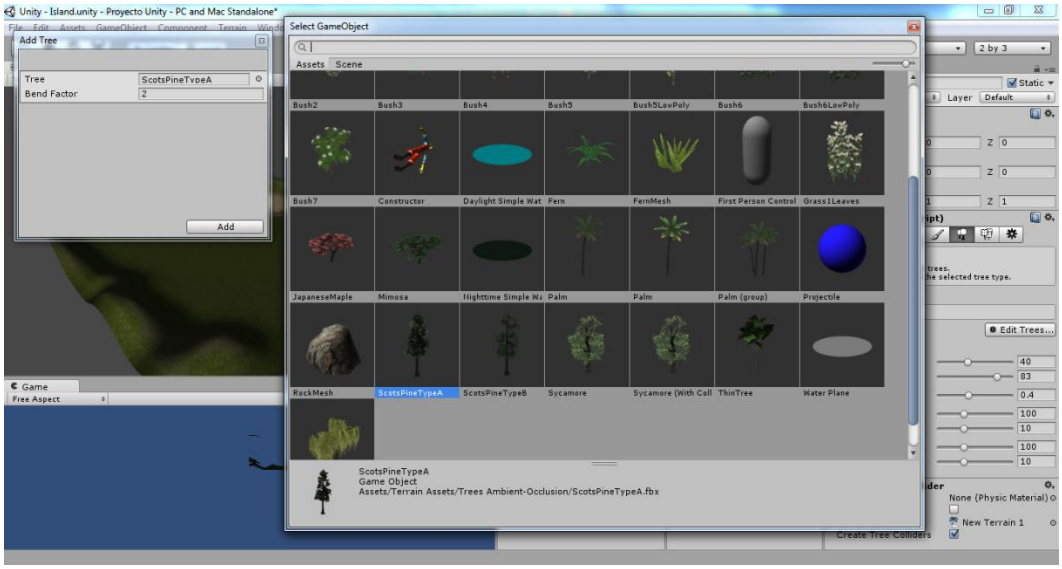


Fuente: elaboración propia, empleando Unity.

Luego, se utilizó la herramienta de edición de terreno *Place Trees* para colocar los árboles sobre la superficie de la escena. Dentro de las opciones de árboles disponibles luego de haber descargado el paquete anterior, se seleccionó el tipo de árbol *ScotsPineTypeA* para colocarlo en el terreno. Los valores de configuración para la herramienta *Place Trees* utilizados fueron los

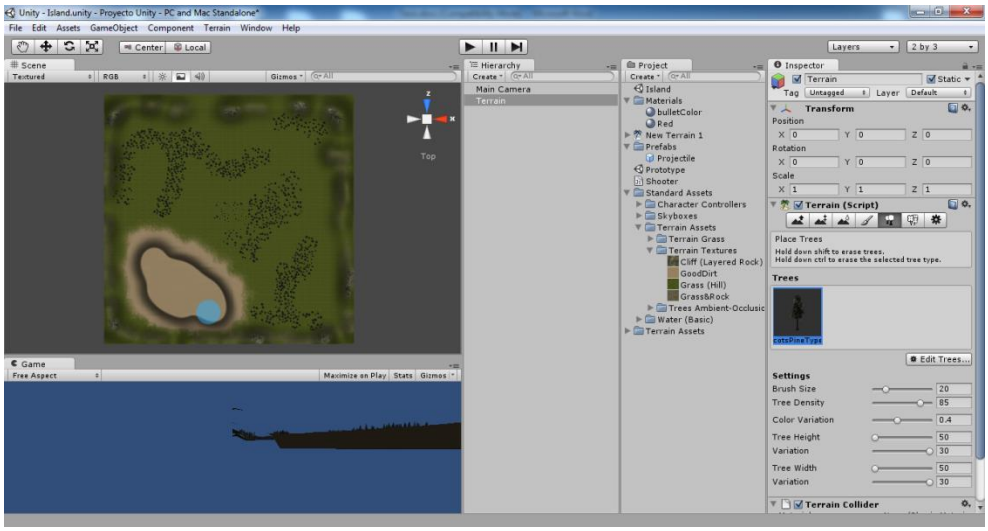
siguientes: *Brush Size* = 20, *Tree Density* = 85, *Color Variation* = 0.4, *Tree Height/Widt*h = 50, *Tree Height/Width Variation* = 30. Los resultados luego de la colocación de los árboles se muestran en las siguientes figuras:

Figura 28. **Selección del tipo de árbol a colocar en el terreno**



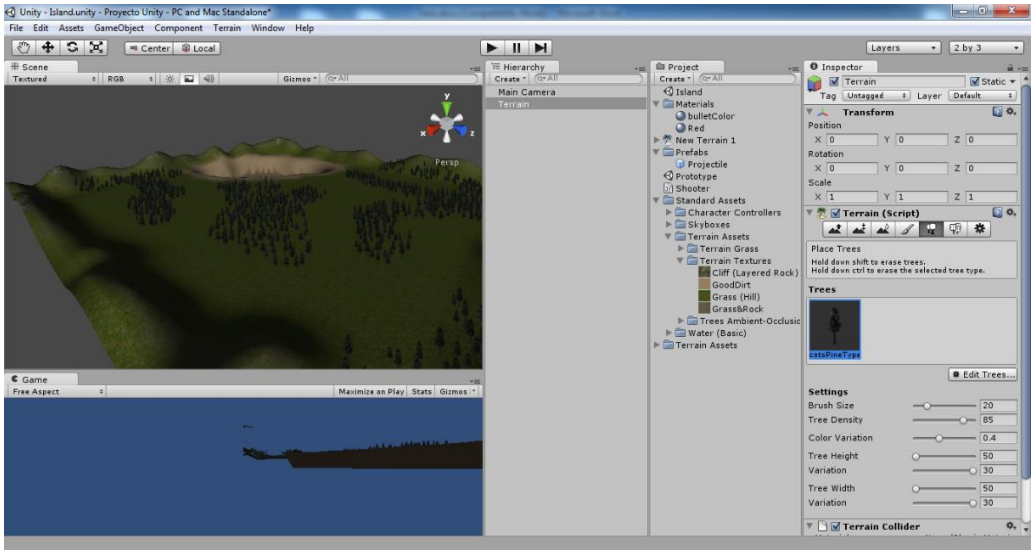
Fuente: elaboración propia, empleando Unity.

Figura 29. **Árboles colocados en el terreno, vista superior**



Fuente: elaboración propia, empleando Unity.

Figura 30. **Árboles colocados en el terreno, vista en perspectiva**



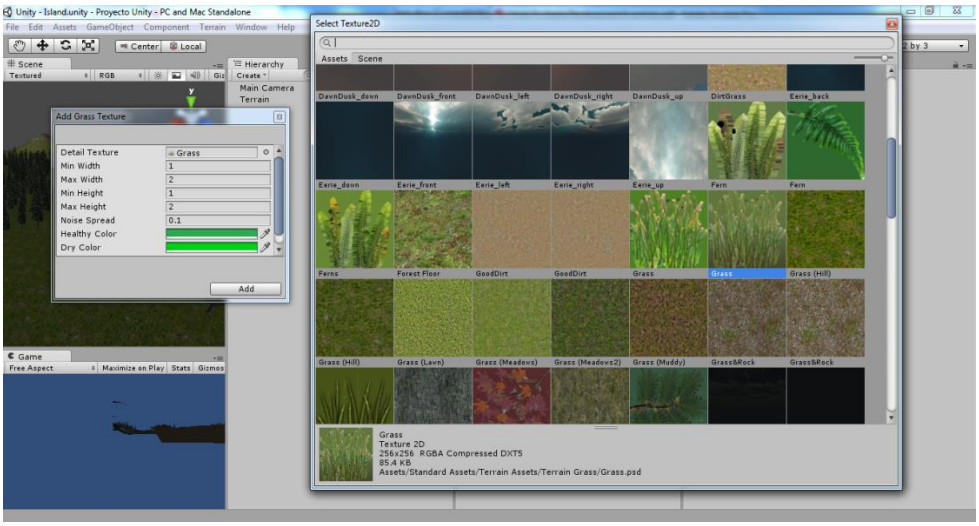
Fuente: elaboración propia, empleando Unity.

Luego de haber colocado los árboles del tipo antes mencionado, se agregó un nuevo tipo de árbol a la paleta para complementar y hacer más densas las zonas boscosas del terreno. El nuevo tipo de árbol utilizado fue el *ScotsPineTypeB*. Los valores de configuración para la colocación de este tipo de árbol del terreno fueron los mismos que los utilizados con el primer tipo de árbol.

- Incorporación de grama al terreno

El siguiente paso en la construcción del terreno es agregarle detalles de grama para incrementar la apariencia de realidad a la textura de grama que fue aplicada al inicio para cubrirlo. Para ello, se utilizó la herramienta *Paint Foliage* incluida en Unity. Dentro de esta herramienta se añadió una textura de grama incluida en los recursos (*assets*) estándar de Unity. Este tipo de texturas incluidas en Unity son de tipo 2D (dos dimensiones) por lo que, al aplicarlas, se utilizó un método conocido como *billboarding* para hacer que la textura rote con relación a la posición de la cámara en el juego y con esto crear un efecto más tridimensional. La opción para aplicar *billboarding* se encuentra en el diálogo de selección de textura en Unity.

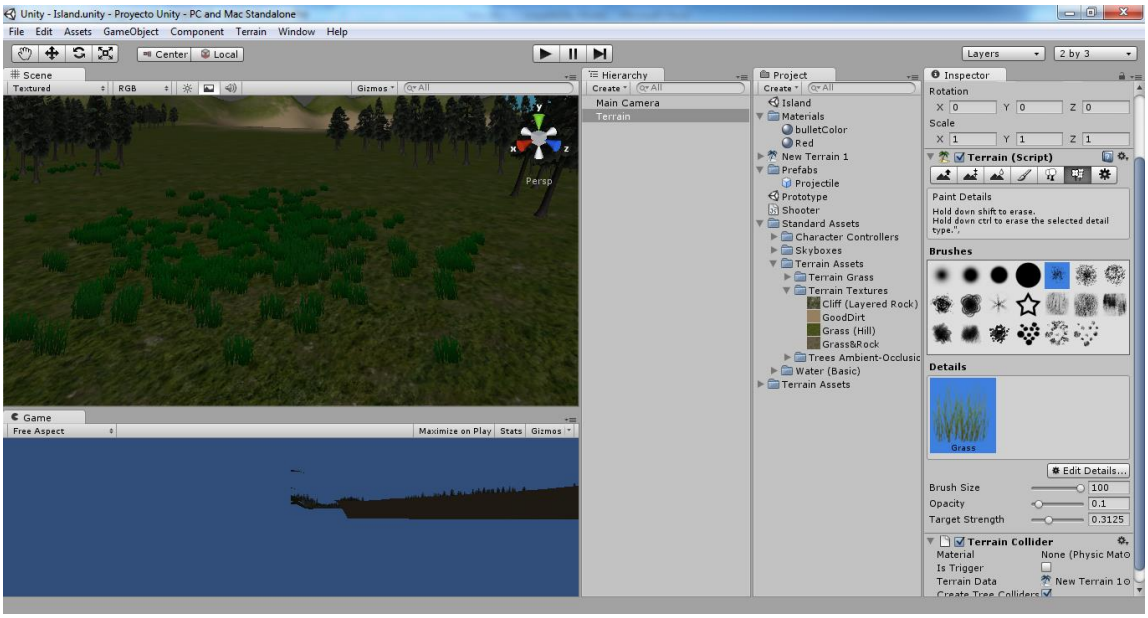
Figura 31. **Selección de la textura de grama a aplicar**



Fuente: elaboración propia, empleando Unity.

Para la aplicación de esta textura de grama se utilizaron los siguientes valores de configuración para el pincel de aplicación: *Brush Size* = 100, *Opacity* = 0.1 y *Target Strength* = 0.3. El tipo de pincel utilizado fue uno de tipo “salpicado” (número 5 en la paleta de pinceles de Unity). Estos valores de configuración hacen que no se aplique demasiada textura al pintar sobre el terreno, debido a que si se utilizada demasiada de este tipo de textura el rendimiento del juego puede verse afectado. La grama aplicada al terreno se muestra en la siguiente figura.

Figura 32. **Textura de grama aplicada al terreno**



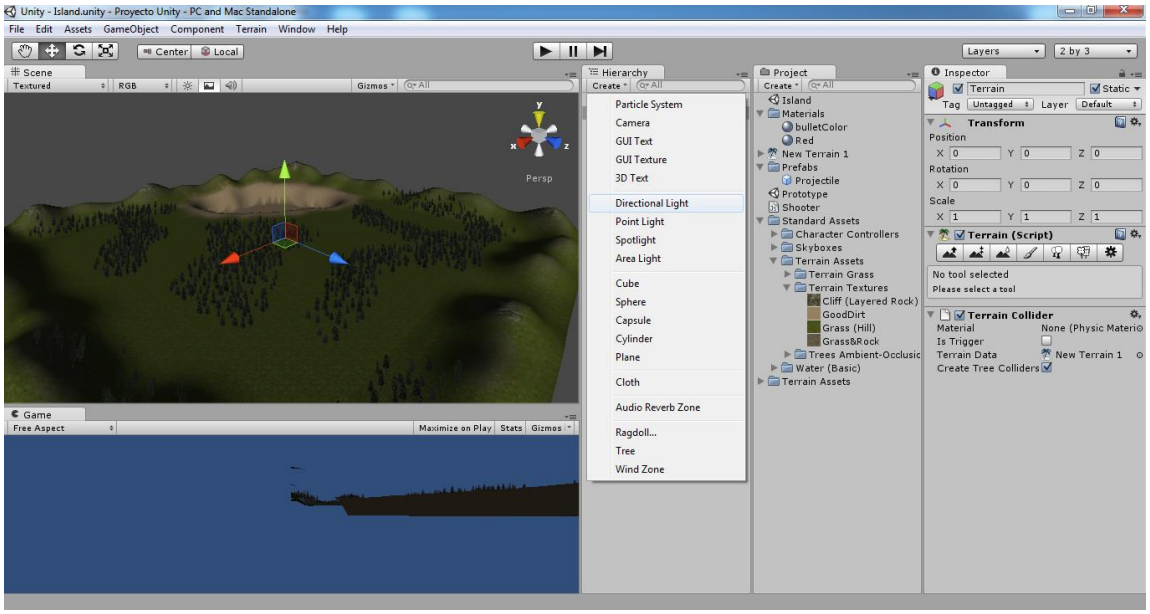
Fuente: elaboración propia, empleando Unity.

- Iluminación

La fuente principal de iluminación utilizada en la escena creada es la luz del sol. Para simular este tipo de luz dentro del juego se utilizó un objeto de tipo *Directional Light*. La característica principal de este tipo de luz en Unity es que no emana de un solo punto, sino que simplemente los rayos simulados viajan en una sola dirección. Este es uno de los tres tipos de luz que se pueden crear dentro de esta herramienta.

Para agregar una luz de este tipo en Unity se utiliza la opción *Create* → *Directional Light* en el panel de jerarquía de objetos dentro de la aplicación.

Figura 33. **Agregando una luz direccional a la escena**

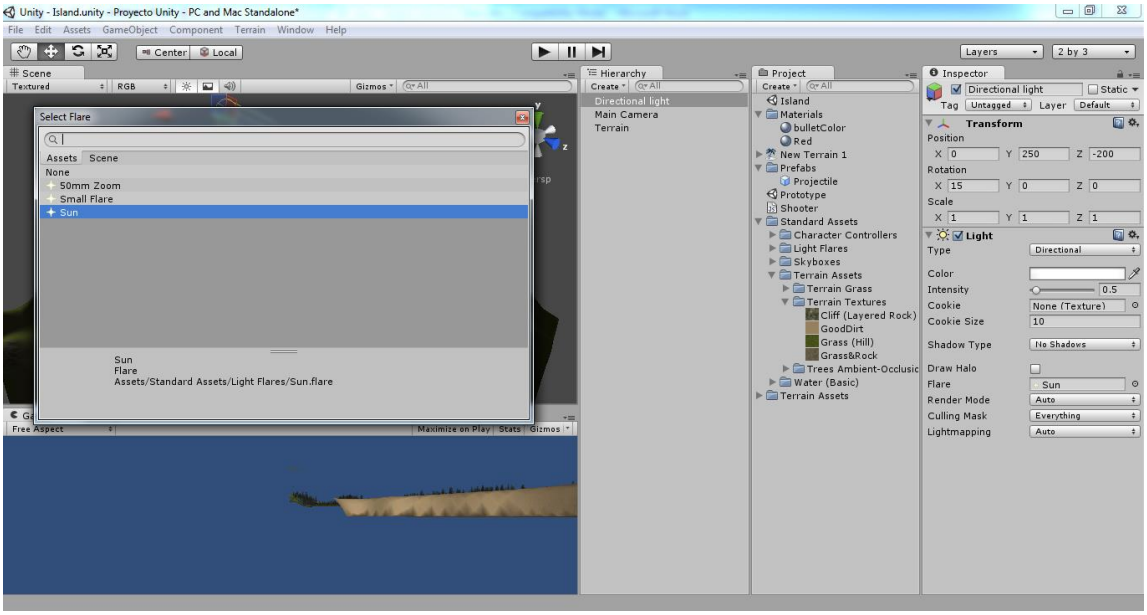


Fuente: elaboración propia, empleando Unity.

A continuación, para que el jugador vea esta luz, se añadió un destello a la misma o *light flare*. Este tipo de efecto para las luces se encuentra en uno de los paquetes de *assets* estándar incluidos en Unity. Para utilizarlo se debe importar el paquete con nombre *Light Flares* al proyecto

Después de importar este paquete, para aplicar el efecto a la luz direccional que se agregó previamente, se seleccionó la luz en la jerarquía de objetos de la escena y se configuró la opción *Flare* de la misma con la selección de nombre *Sun*.

Figura 34. **Seleccionando y aplicando el tipo de destello a la luz direccional**



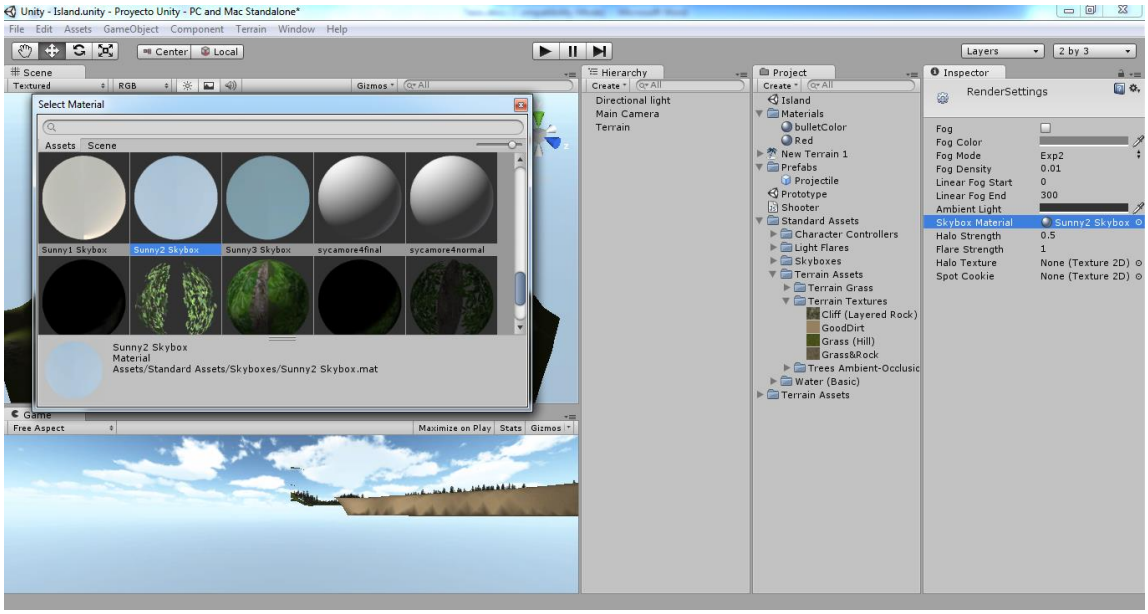
Fuente: elaboración propia, empleando Unity.

- Cielo

Para agregar la textura del cielo a la escena del videojuego se utilizó un objeto de tipo *Skybox*. Este objeto no es más que un cubo que envuelve toda la escena creada y al cual se le aplican texturas por dentro para que representen el horizonte que mira el jugador.

Para agregar un objeto de este tipo a la escena se debe ingresar al menú *Edit* de la interfaz de Unity y luego a la opción *Render Settings*. Dentro de estas opciones de configuración se configuró el material del *Skybox* a la textura con nombre *Sunny2 Skybox* incluida por defecto en Unity. La configuración y el resultado se muestra en la siguiente figura.

Figura 35. **Seleccionando y aplicando el tipo de destello a la luz direccional**



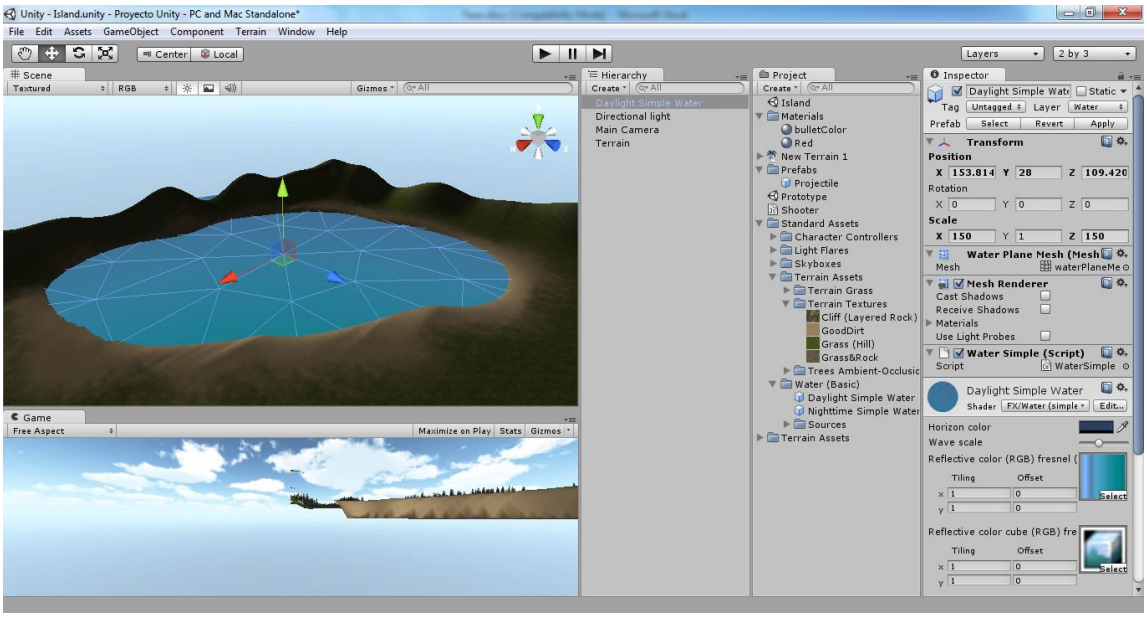
Fuente: elaboración propia, empleando Unity.

- Incorporación de agua al lago

Para añadir agua al lago creado previamente como parte del terreno en el videojuego, se utilizó un material prefabricado llamado *Daylight Simple Water* que es parte del paquete estándar de recursos de Unity.

La forma de este objeto prefabricado es la de un disco plano, el cual fue colocado en el centro del lago a la altura de la orilla y luego expandido incrementando los valores de escala para que cubriera toda la superficie del lago. El resultado se muestra en la siguiente figura.

Figura 36. **Aplicado el objeto simulando el agua al lago**



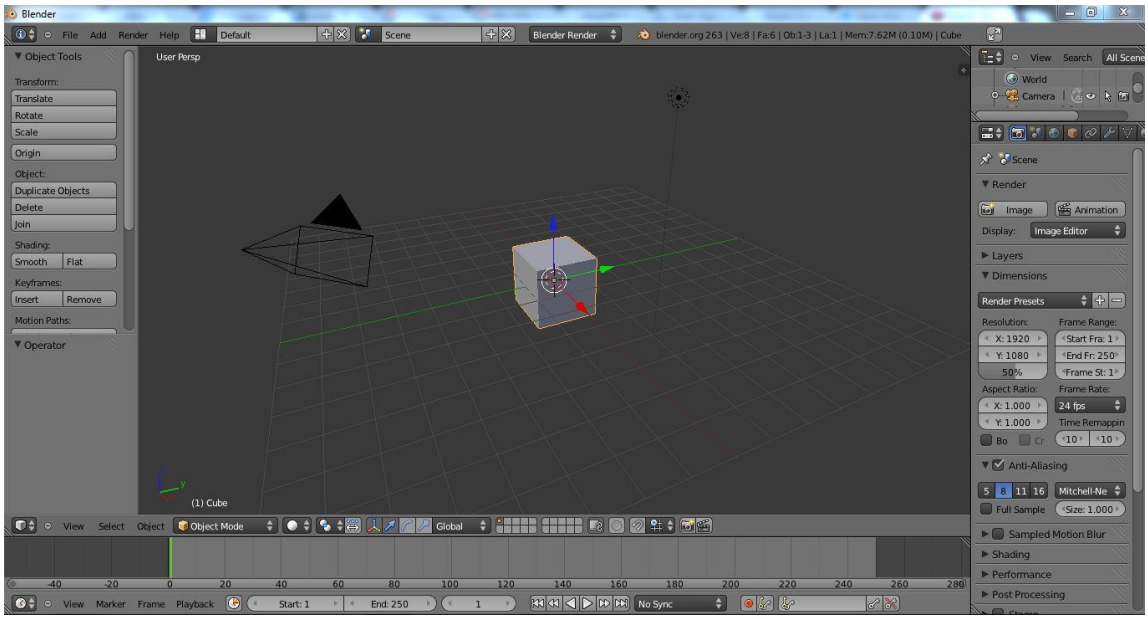
Fuente: elaboración propia, empleando Unity.

2.5.2. Objetos

Como se describió anteriormente, el videojuego estará ambientado en lo que se podría considerar una ciudad maya de la época prehispánica. Posiblemente el elemento arquitectónico más importante y representativo de las ciudades mayas de aquella época sean sus pirámides, por lo que se decidió modelar una pirámide basándose en el diseño general de las pirámides creadas por los mayas en aquella época. El proceso de creación de esta pirámide se describe en las siguientes páginas.

El proceso de elaboración de la pirámide maya en Blender se inició creando los diferentes niveles de la pirámide a partir de la modificación de objetos tipo cubo, como el que se muestra en la siguiente figura.

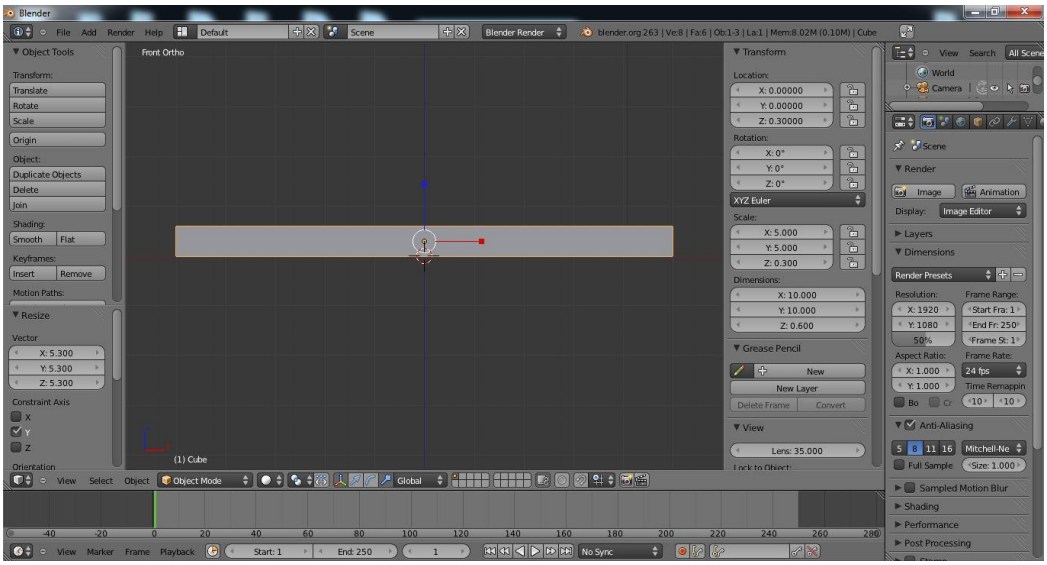
Figura 37. **Inicio del modelado de la pirámide maya en Blender**



Fuente: elaboración propia, empleando Blender.

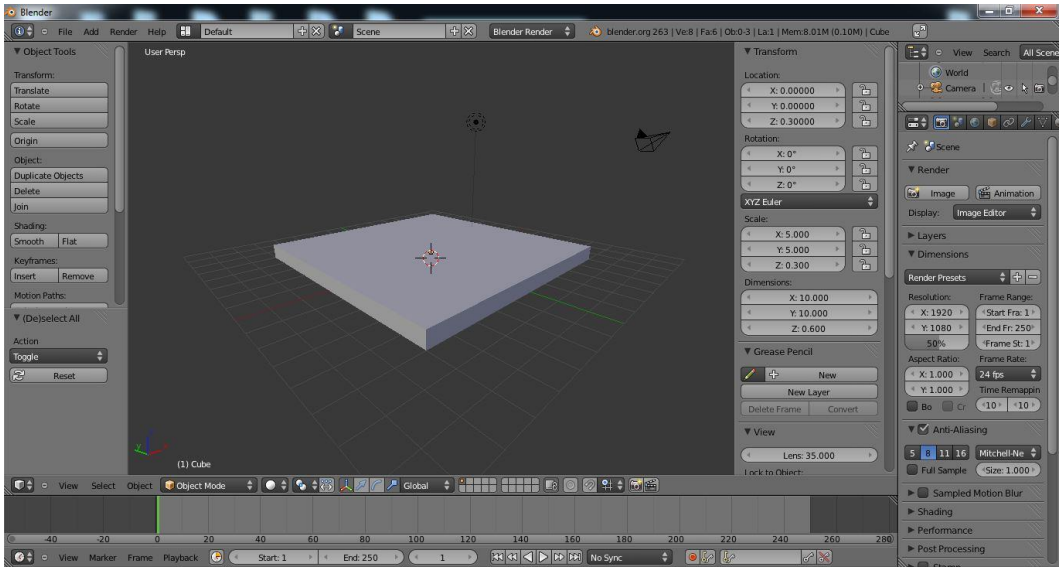
La primera modificación realizada al cubo para crear el primer nivel de la pirámide fue el de modificar su tamaño o escala sobre los ejes X y Y, a modo de que simulara lo que podría llamarse una “plancha” de piedra o concreto.

Figura 38. **Modificación del tamaño del cubo**



Fuente: elaboración propia, empleando Blender.

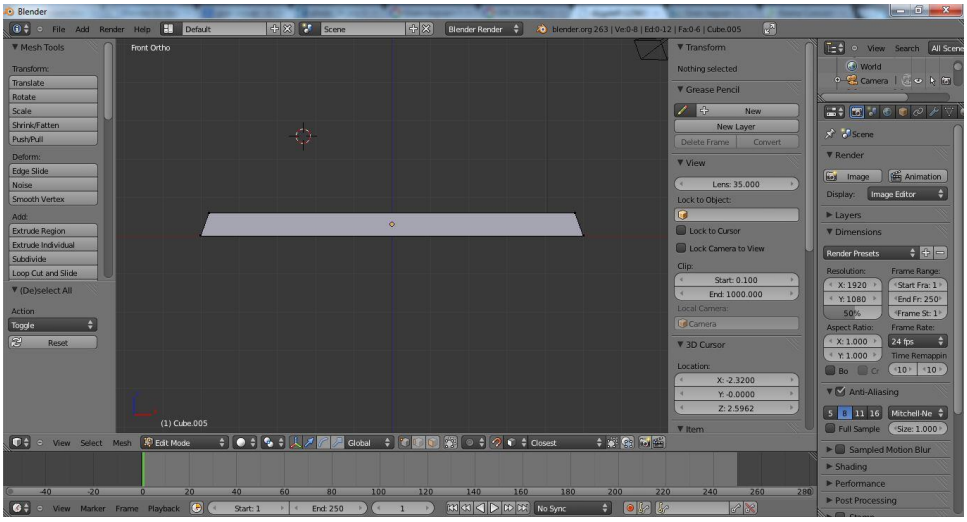
Figura 39. **Vista en perspectiva del nuevo objeto**



Fuente: elaboración propia, empleando Blender.

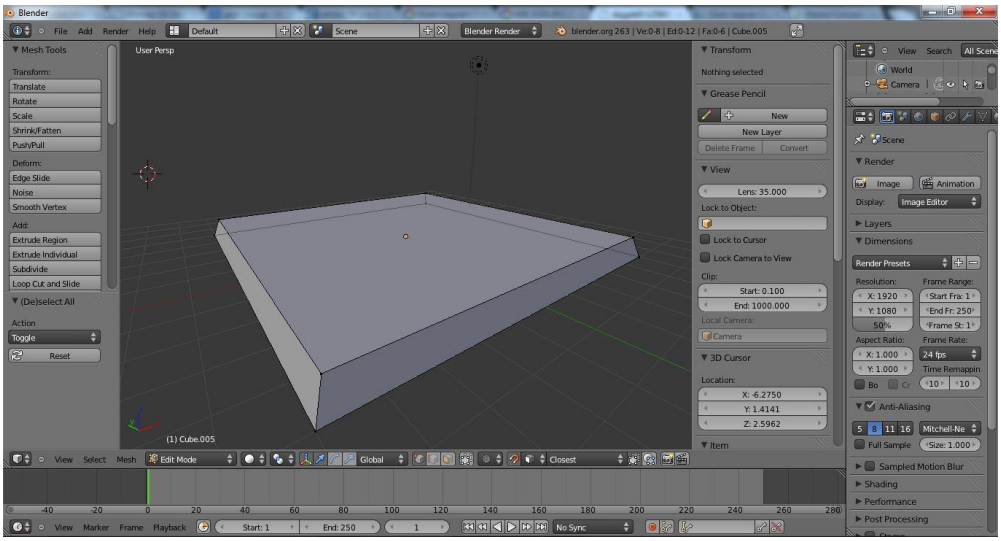
El siguiente paso consistió en realizar una pequeña inclinación de las caras laterales del objeto como se muestra en las siguientes figuras. A partir de este objeto se construyeron los niveles de la pirámide.

Figura 40. **Inclinación de las paredes exteriores del objeto**



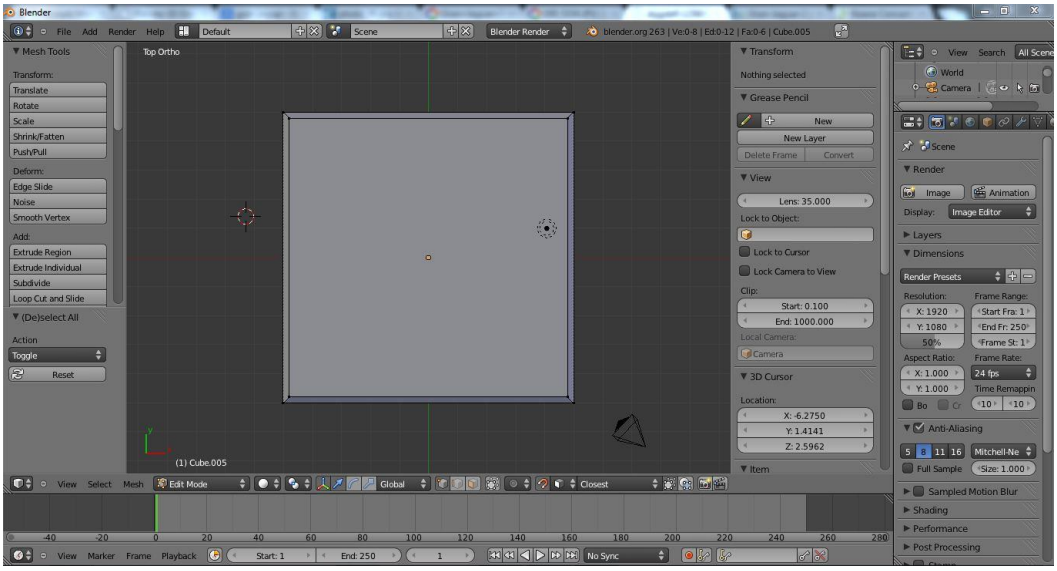
Fuente: elaboración propia, empleando Blender.

Figura 41. **Vista en perspectiva del objeto**



Fuente: elaboración propia, empleando Blender.

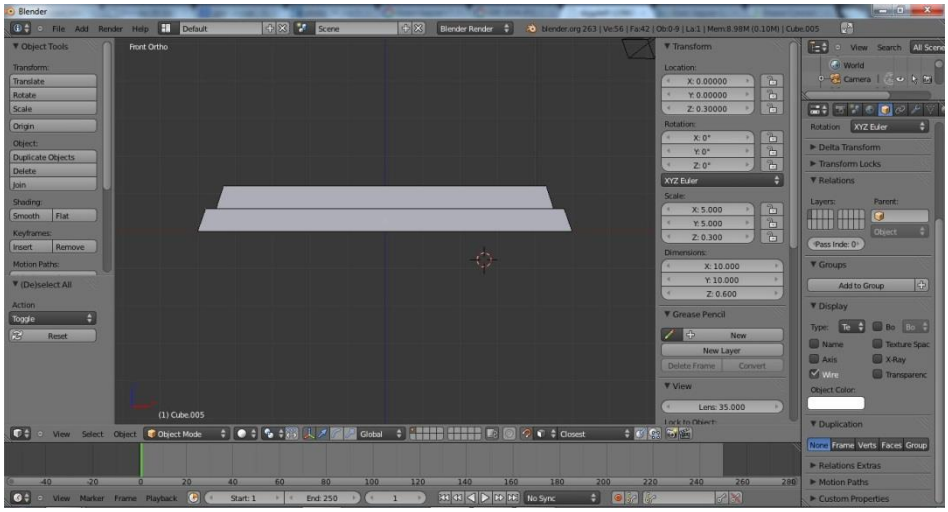
Figura 42. **Vista superior del objeto con sus caras externas inclinadas**



Fuente: elaboración propia, empleando Blender.

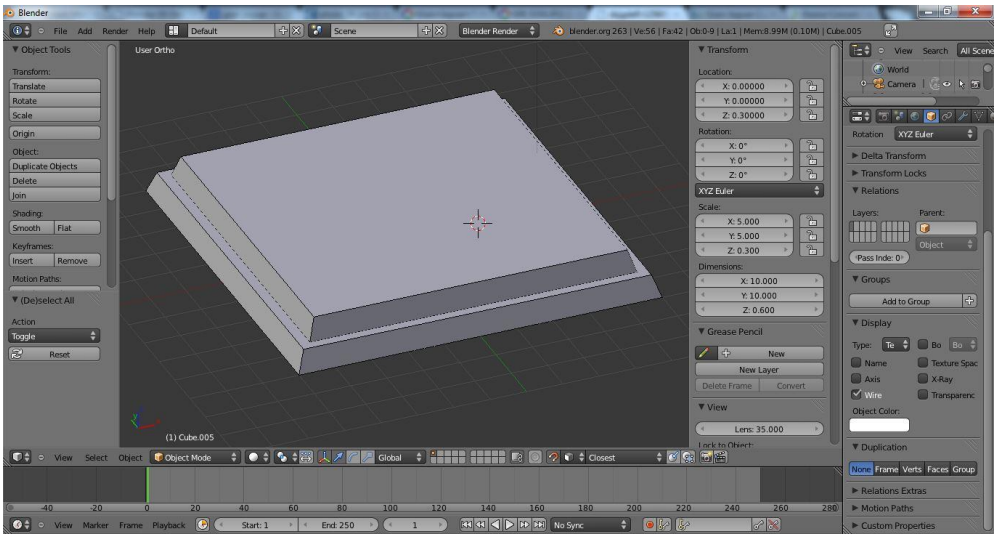
Una vez se conformó el objeto que representa el primer nivel de la pirámide, se realizó un duplicado para crear el segundo nivel. El ancho y largo de este nuevo objeto se redujeron levemente (dejando la altura sin variación) y luego se colocó y alineó a la parte superior del primer objeto, creando los primeros escalones de la pirámide. El resultado se muestra en las siguientes figuras.

Figura 43. **Objeto duplicado representando el segundo nivel de la pirámide**



Fuente: elaboración propia, empleando Blender.

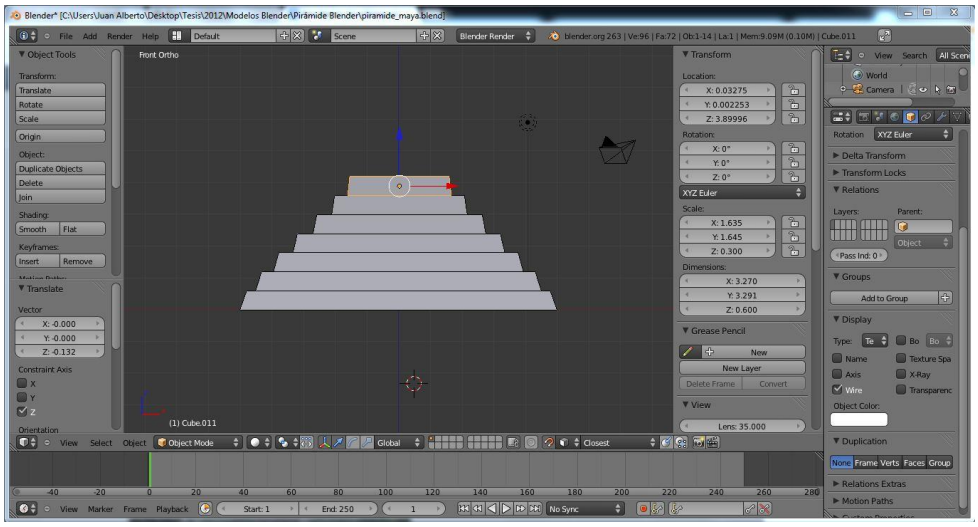
Figura 44. **Vista en perspectiva de los dos primeros niveles de la pirámide**



Fuente: elaboración propia, empleando Blender.

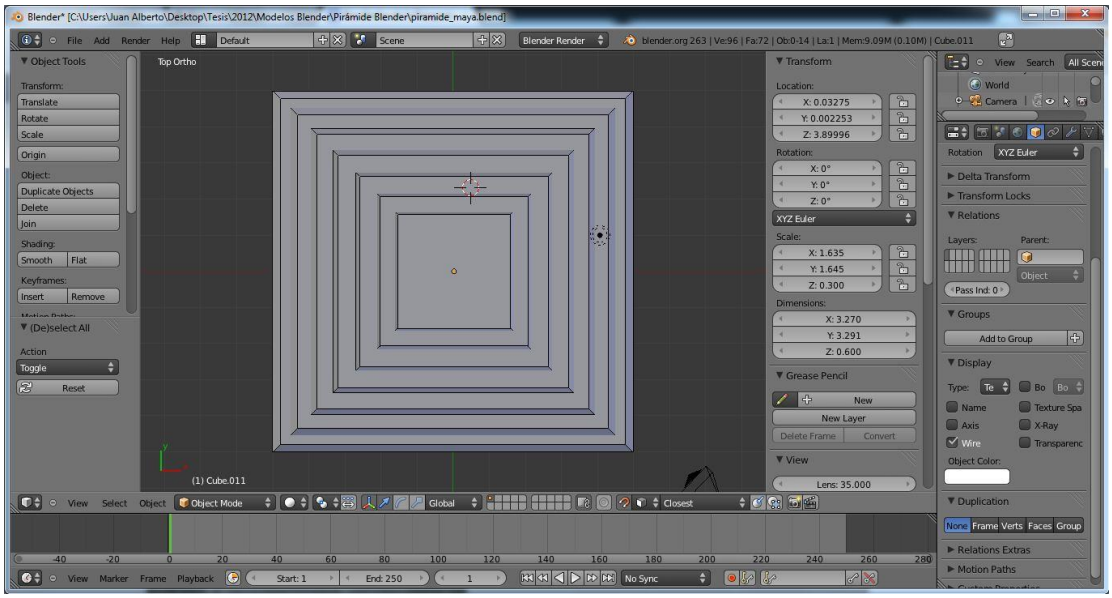
Los pasos descritos anteriormente para crear el segundo nivel de la pirámide fueron repetidos varias veces más para crear los siguientes niveles de esta, hasta tener una estructura de 7 niveles, como se muestra en las siguientes figuras.

Figura 45. **Pirámide con 7 niveles creados**



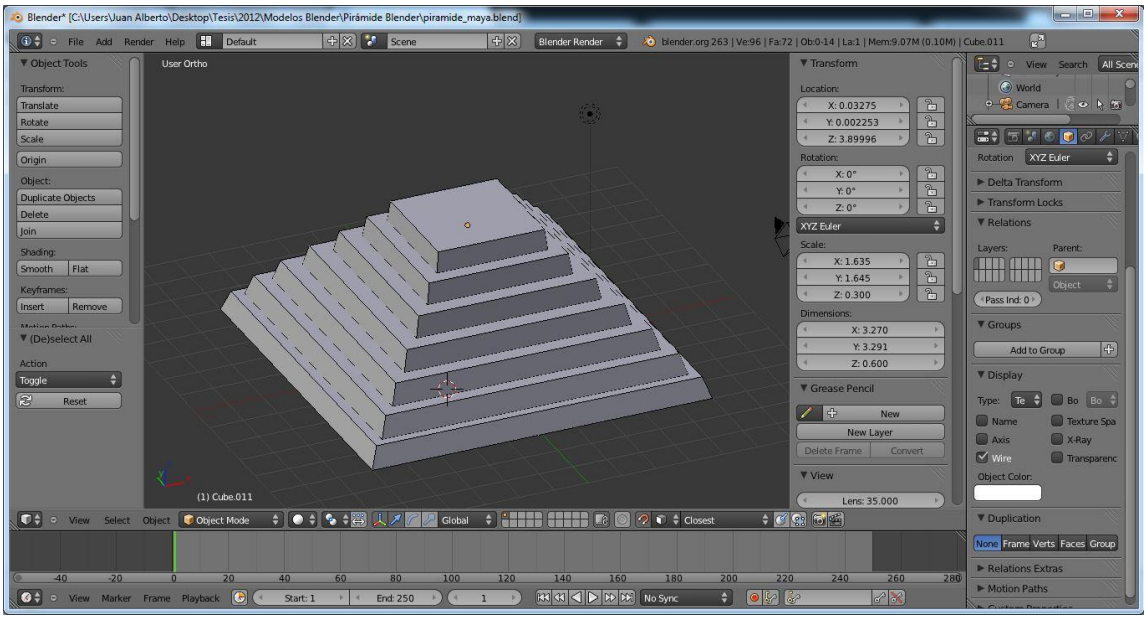
Fuente: elaboración propia, empleando Blender.

Figura 46. **Pirámide con 7 niveles creados. Vista superior**



Fuente: elaboración propia, empleando Blender.

Figura 47. **Pirámide con 7 niveles creados. Vista en perspectiva**

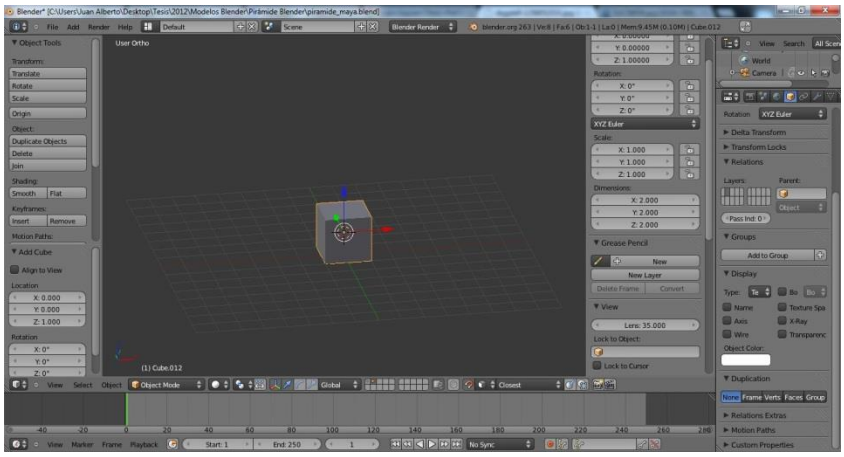


Fuente: elaboración propia, empleando Blender.

- Creación del templo en la cima de la pirámide

Para el modelado del templo en la cima de la pirámide maya, se empezó de nuevo con un cubo en una capa o *layer* nueva en Blender.

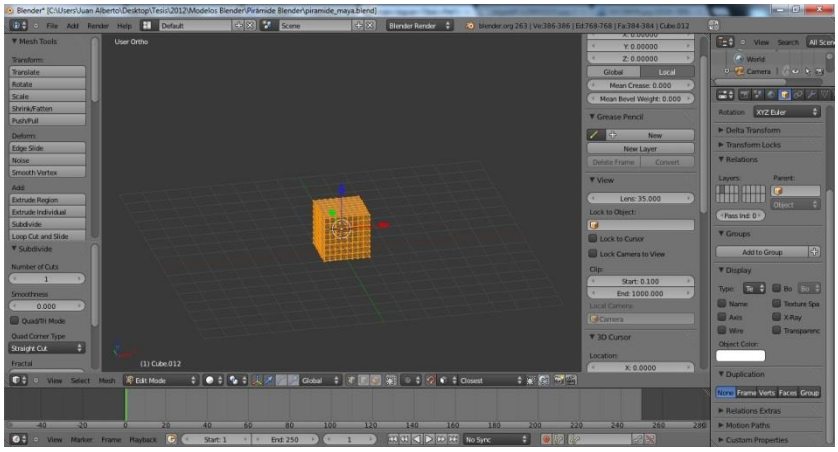
Figura 48. **Cubo base para el templo**



Fuente: elaboración propia, empleando Blender.

A continuación, se realizaron varias subdivisiones consecutivas del cubo para obtener más vértices interiores en el mismo.

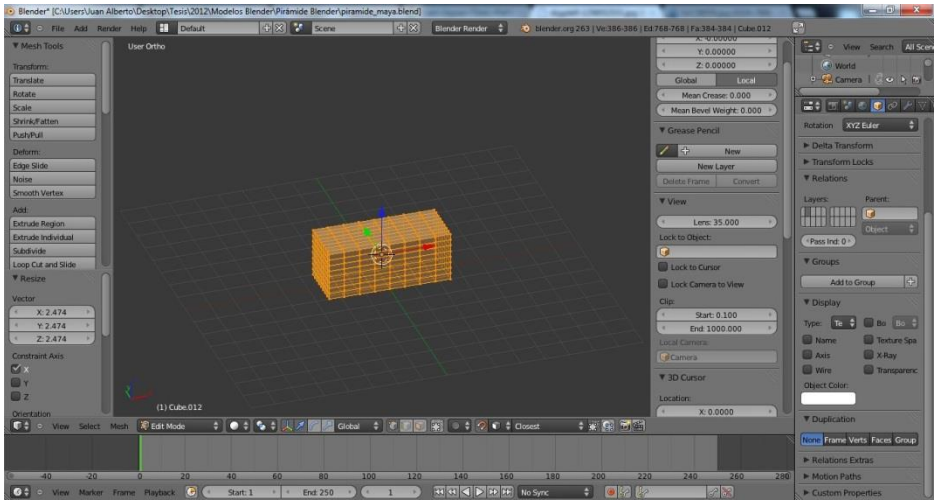
Figura 49. **Subdivisiones al cubo base**



Fuente: elaboración propia, empleando Blender.

A continuación, se agrandó el cubo sobre el eje X para obtener el objeto mostrado en la siguiente figura.

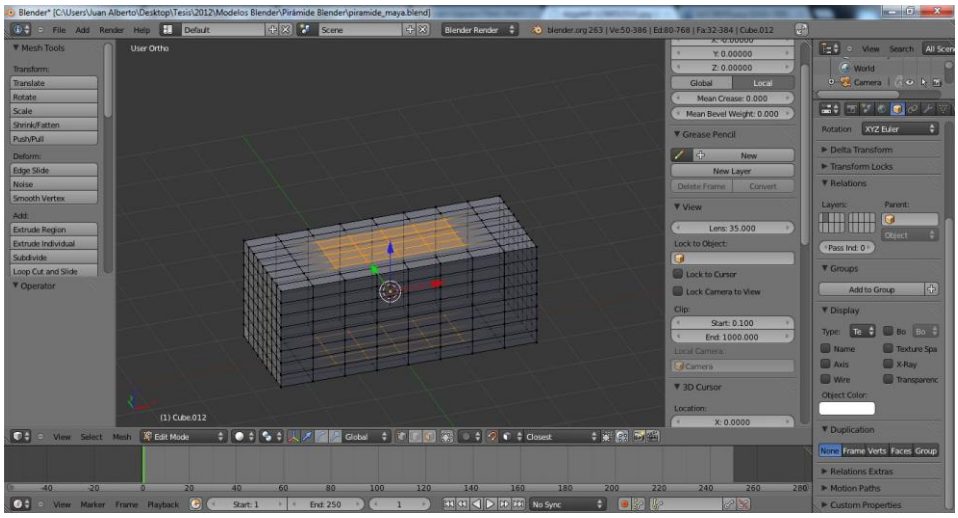
Figura 50. **Alargado del cubo sobre el eje X**



Fuente: elaboración propia, empleando Blender.

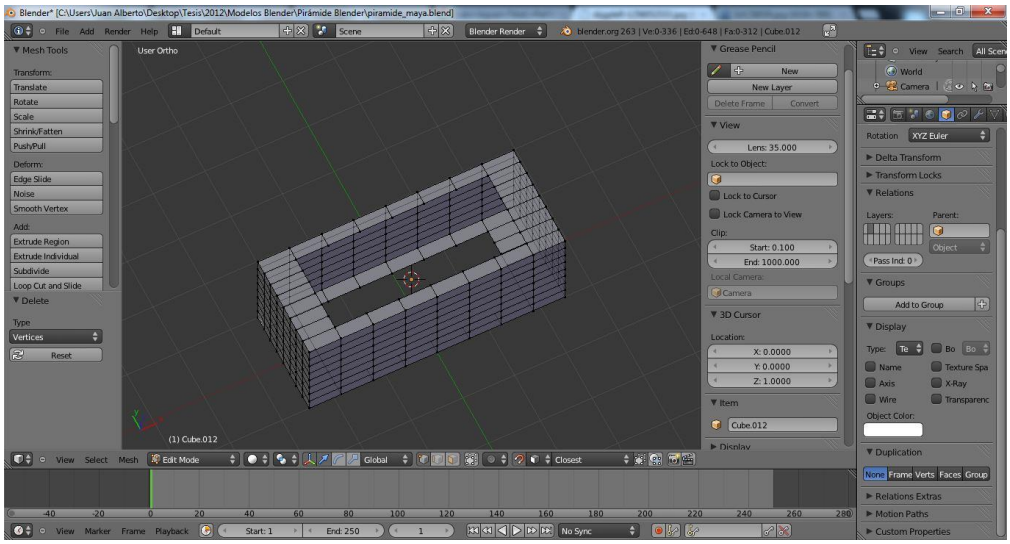
Luego, se seleccionaron los vértices centrales de las caras superior e inferior del cubo y se eliminaron para crear una abertura.

Figura 51. **Selección de vértices**



Fuente: elaboración propia, empleando Blender.

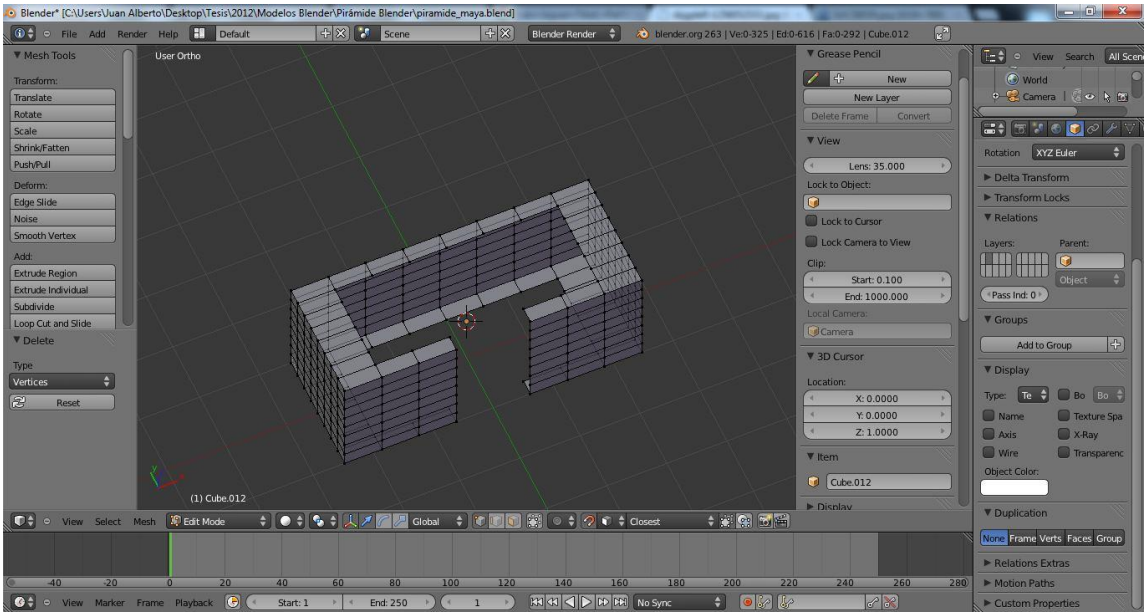
Figura 52. **Resultado de la eliminación de los vértices**



Fuente: elaboración propia, empleando Blender.

Luego, se seleccionó la columna de vértices centrales de la cara frontal del cubo y se eliminó para crear otra apertura como la mostrada en la siguiente figura.

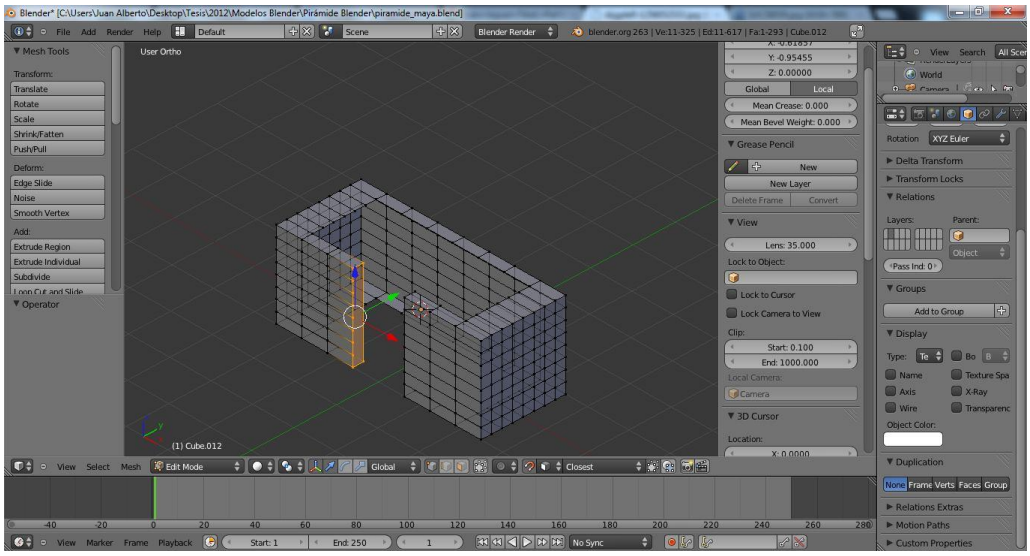
Figura 53. **Creación de la apertura frontal**



Fuente: elaboración propia, empleando Blender.

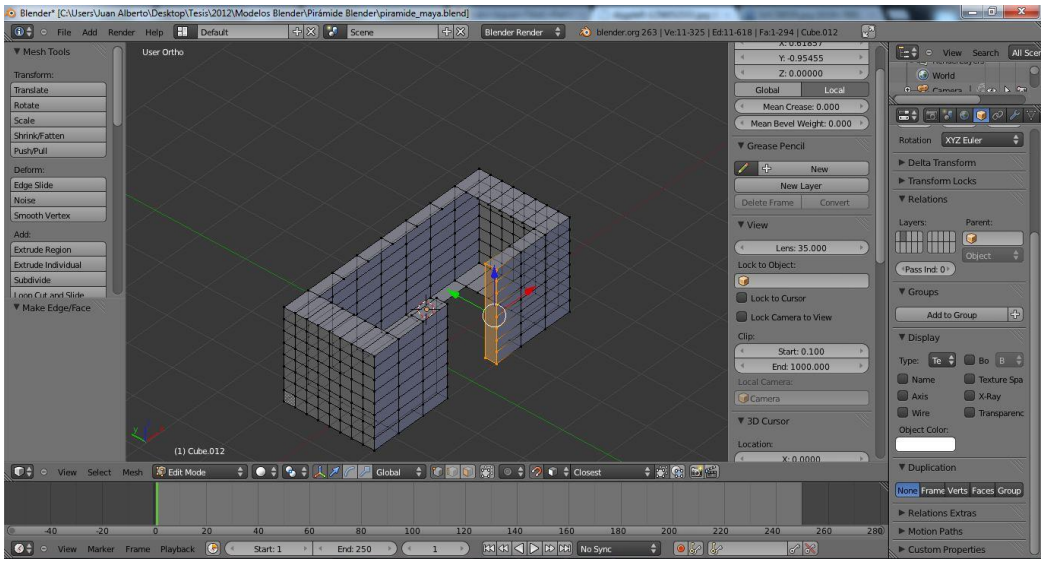
A continuación, se seleccionaron los vértices mostrados en la siguiente figura y se creó una nueva cara o *face* con base en estos. El mismo procedimiento se replicó con los vértices del lado opuesto.

Figura 54. **Refinamiento de la apertura frontal (I)**



Fuente: elaboración propia, empleando Blender.

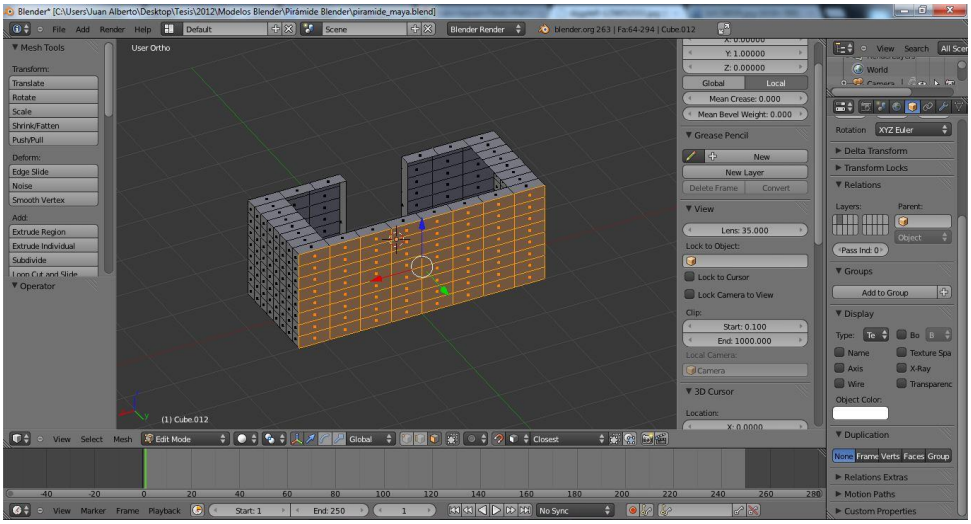
Figura 55. **Refinamiento de la apertura frontal (II)**



Fuente: elaboración propia, empleando Blender.

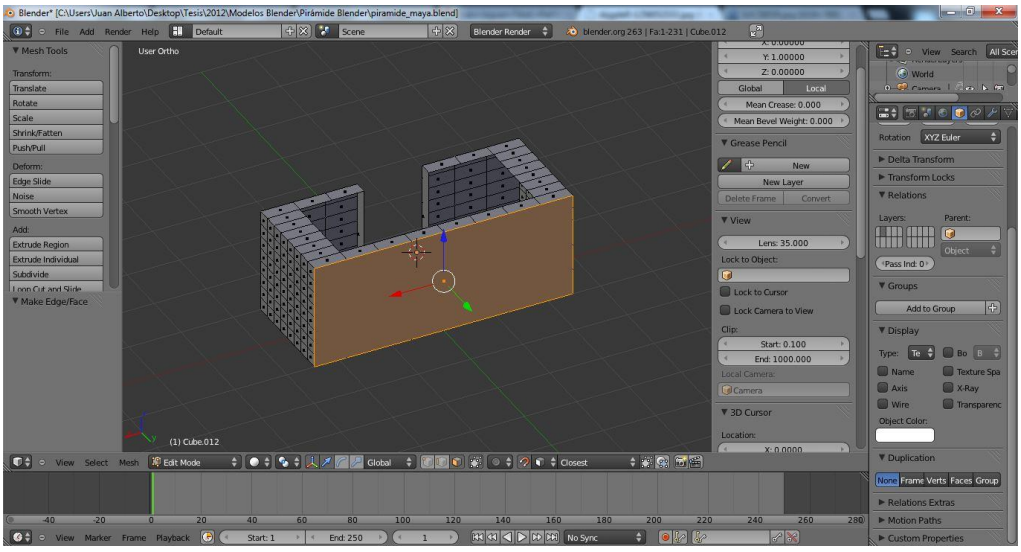
Como se puede observar en las figuras anteriores, el objeto creado tiene, por el momento, un gran número de caras diminutas en cada uno de sus lados. Por lo tanto, se simplificó el objeto al unir varias de estas caras adyacentes en una sola.

Figura 56. **Unión de caras de las paredes del templo (I)**



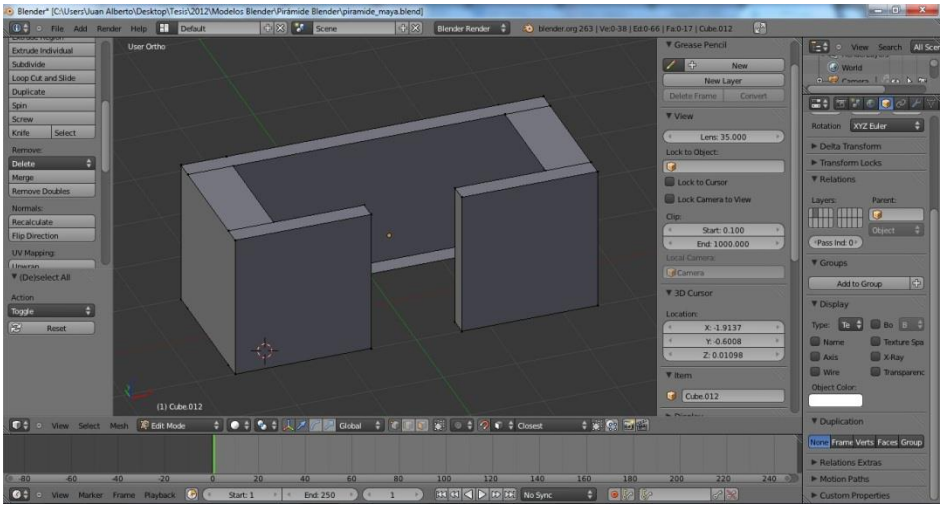
Fuente: elaboración propia, empleando Blender.

Figura 57. **Unión de caras de las paredes del templo (II)**



Fuente: elaboración propia, empleando Blender.

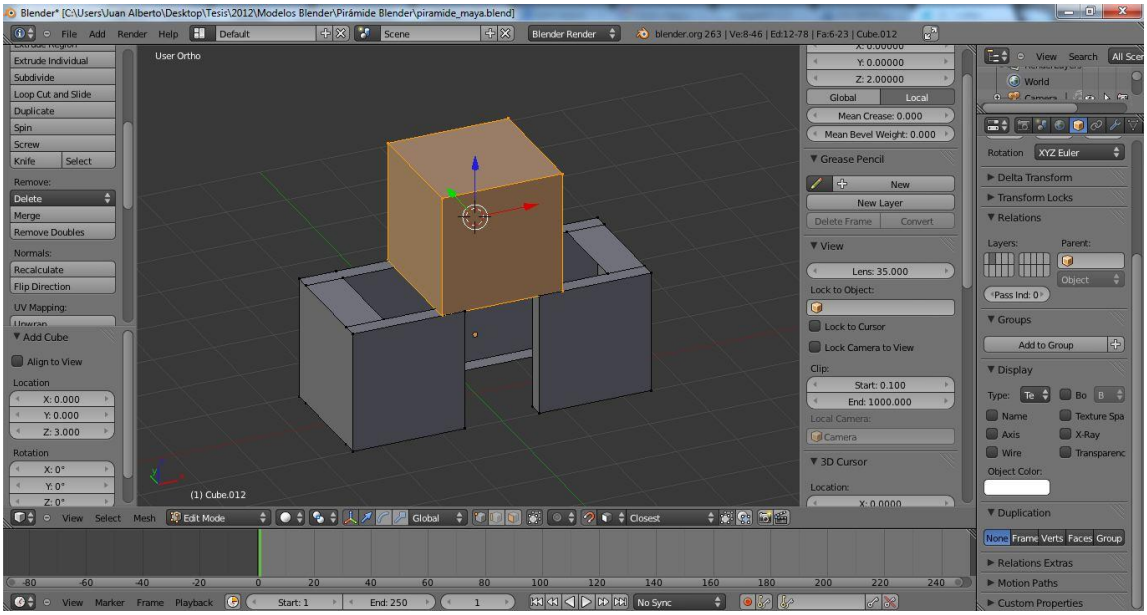
Figura 58. **Paredes del templo con caras simplificadas**



Fuente: elaboración propia, empleando Blender.

Para crear la parte superior del templo, se añadió un cubo en la parte superior del objeto trabajado anteriormente.

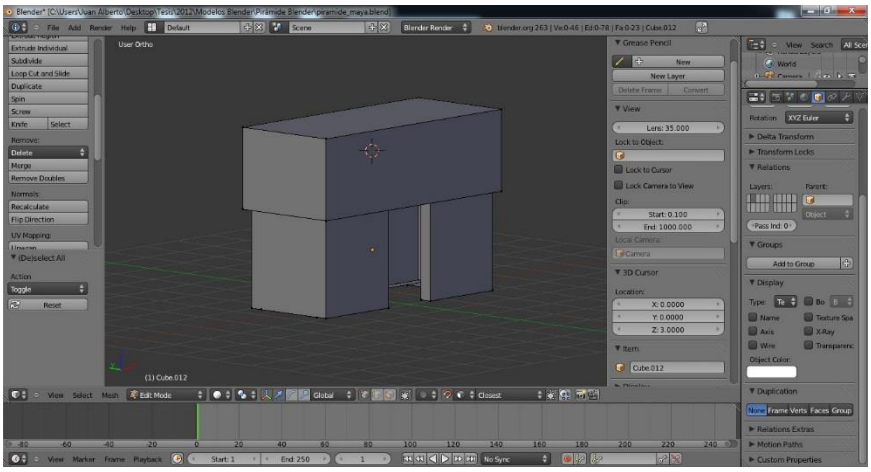
Figura 59. **Inicio de modelado del techo del templo**



Fuente: elaboración propia, empleando Blender.

A continuación, se modificó el tamaño del cubo para crear el techo o parte superior del templo.

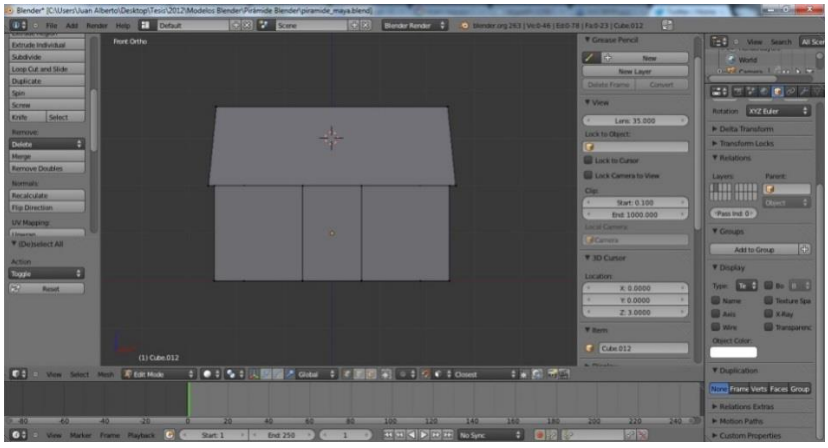
Figura 60. **Creación de la parte superior del templo**



Fuente: elaboración propia, empleando Blender.

Para crear una inclinación en la parte del techo, se seleccionaron los vértices de la cara superior del mismo, y se disminuyó su tamaño sobre los ejes X y Y.

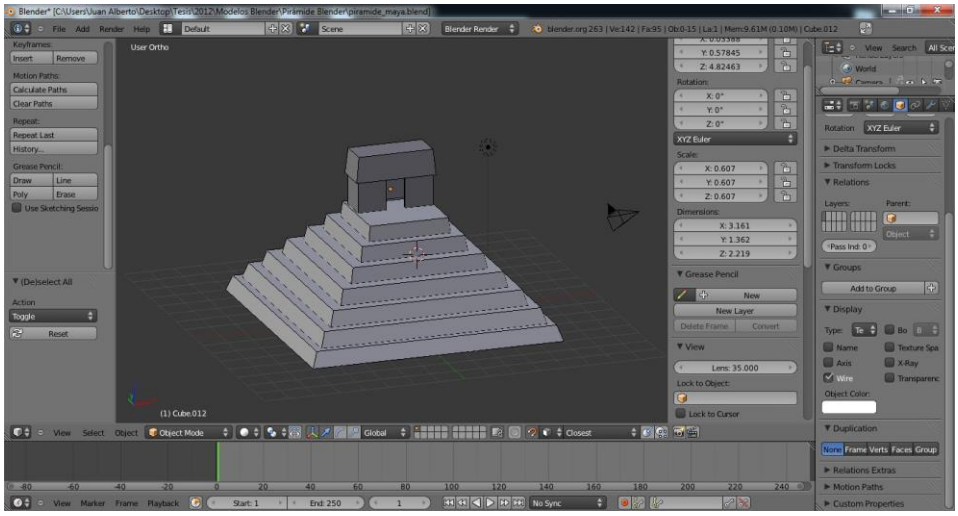
Figura 61. **Creación de la parte superior del templo (II)**



Fuente: elaboración propia, empleando Blender.

A continuación, se trasladó el templo a la misma capa o *layer* de la pirámide para unirlos.

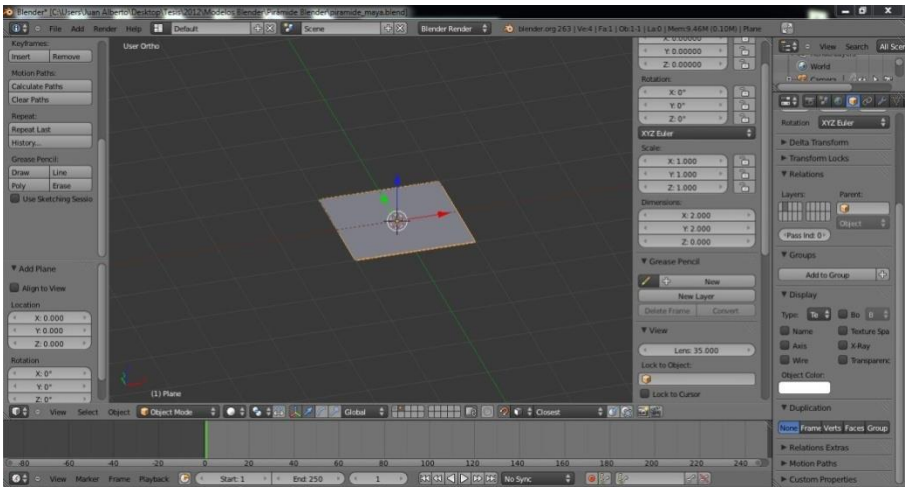
Figura 62. **Unión del templo con la pirámide**



Fuente: elaboración propia, empleando Blender.

Para realizar las escaleras frontales de la pirámide se empezó añadiendo un plano en una nueva capa o *layer* de Blender.

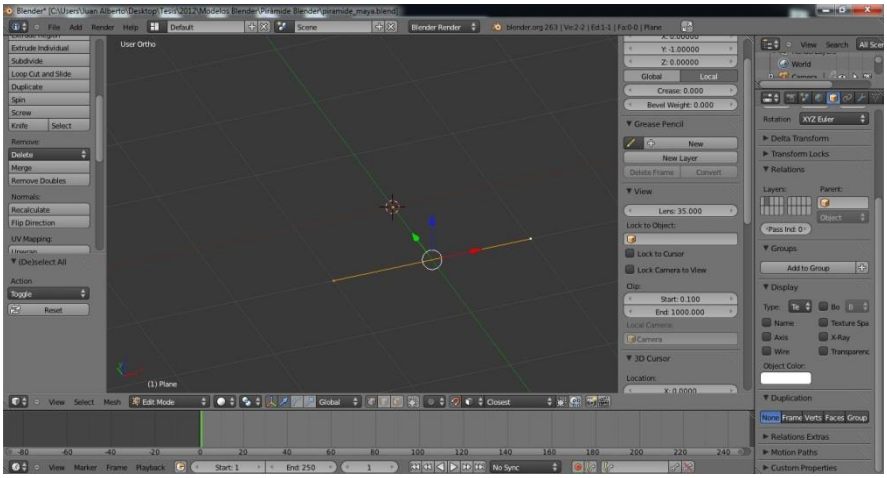
Figura 63. **Inicio de modelado de las escaleras**



Fuente: elaboración propia, empleando Blender.

A continuación, se eliminaron dos de los vértices del plano para que quedara solamente un borde o *edge* unido por dos vértices.

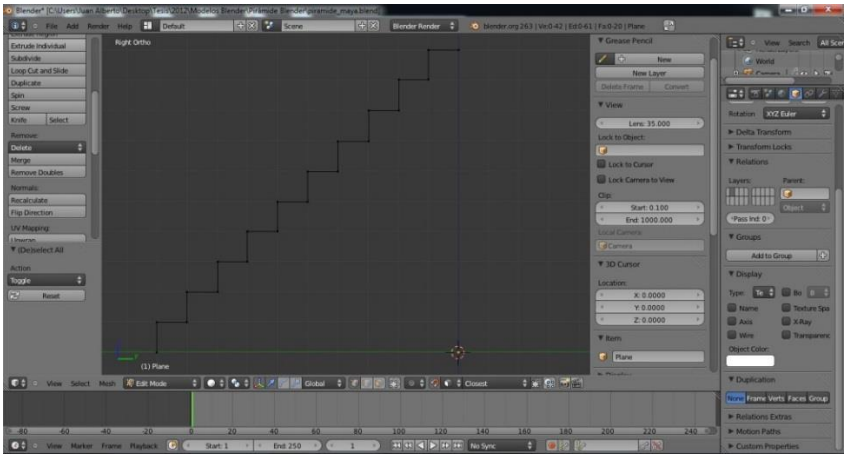
Figura 64. **Modelado de las escaleras (I)**



Fuente: elaboración propia, empleando Blender.

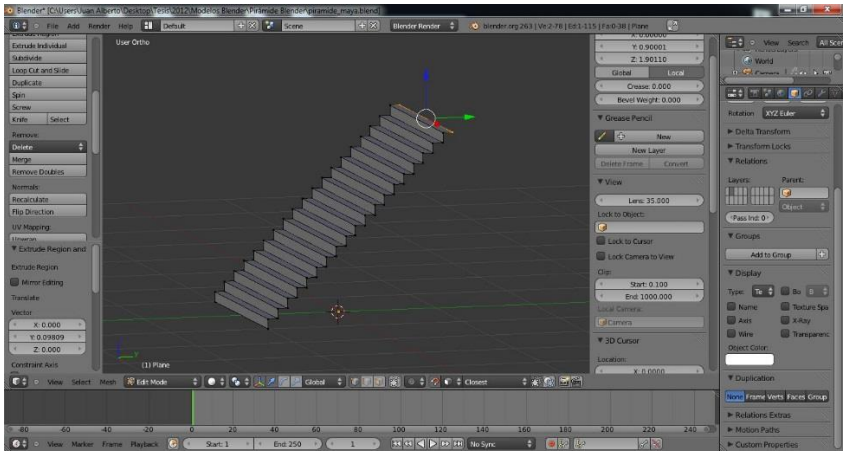
A continuación, utilizando la herramienta *Extrude* de Blender sobre el borde anterior, se crearon los escalones de la pirámide como se muestra en la siguiente figura:

Figura 65. **Modelado de las escaleras (II)**



Fuente: elaboración propia, empleando Blender.

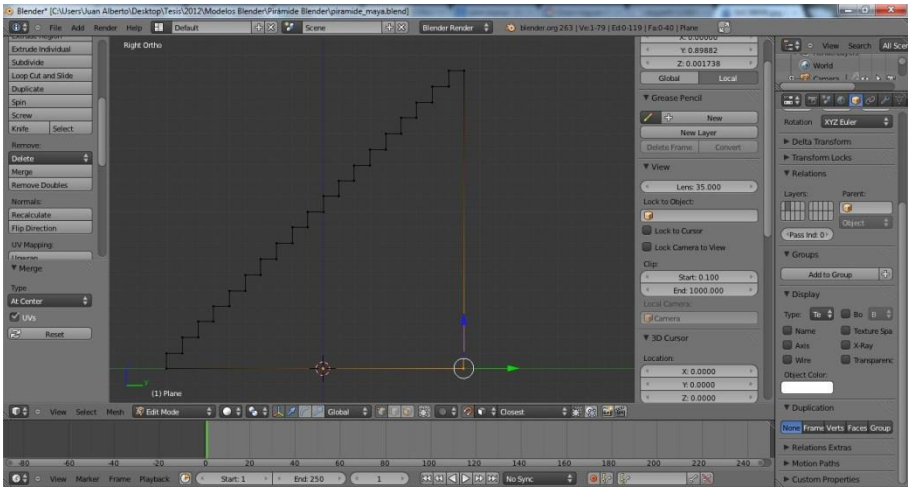
Figura 66. **Modelado de las escaleras (III)**



Fuente: elaboración propia, empleando Blender.

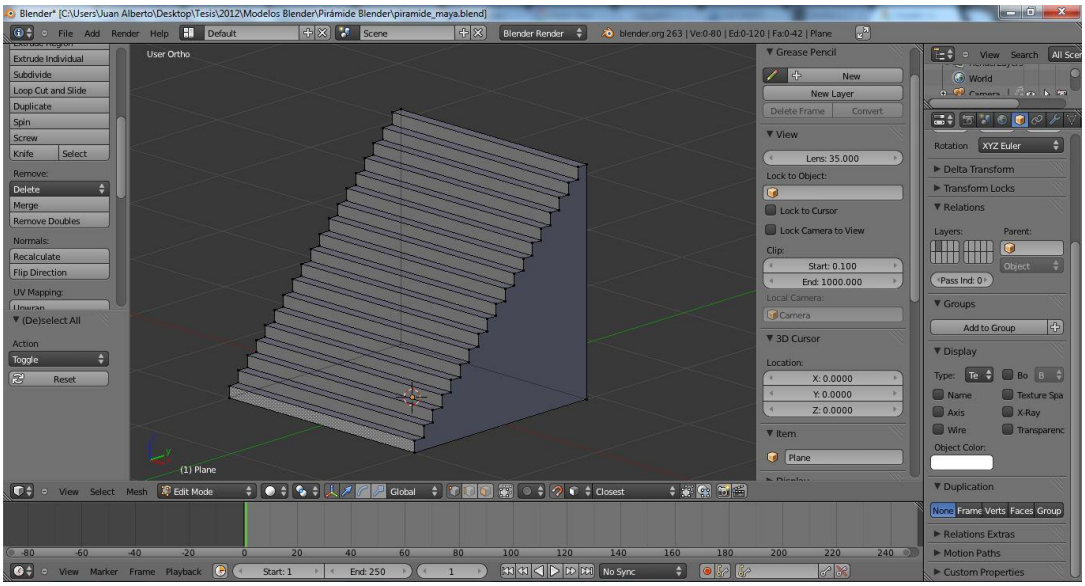
Para crear las dos caras laterales de los escalones se utilizó nuevamente la función *Extrude* en la parte superior e inferior de los mismos y se unieron los nuevos vértices creados utilizando la función *Merge At Centre-Point*. Para finalizar, se crearon las caras utilizando la función *Make Face* de Blender.

Figura 67. **Modelado de las escaleras (IV)**



Fuente: elaboración propia, empleando Blender.

Figura 68. **Escaleras del Templo terminadas**



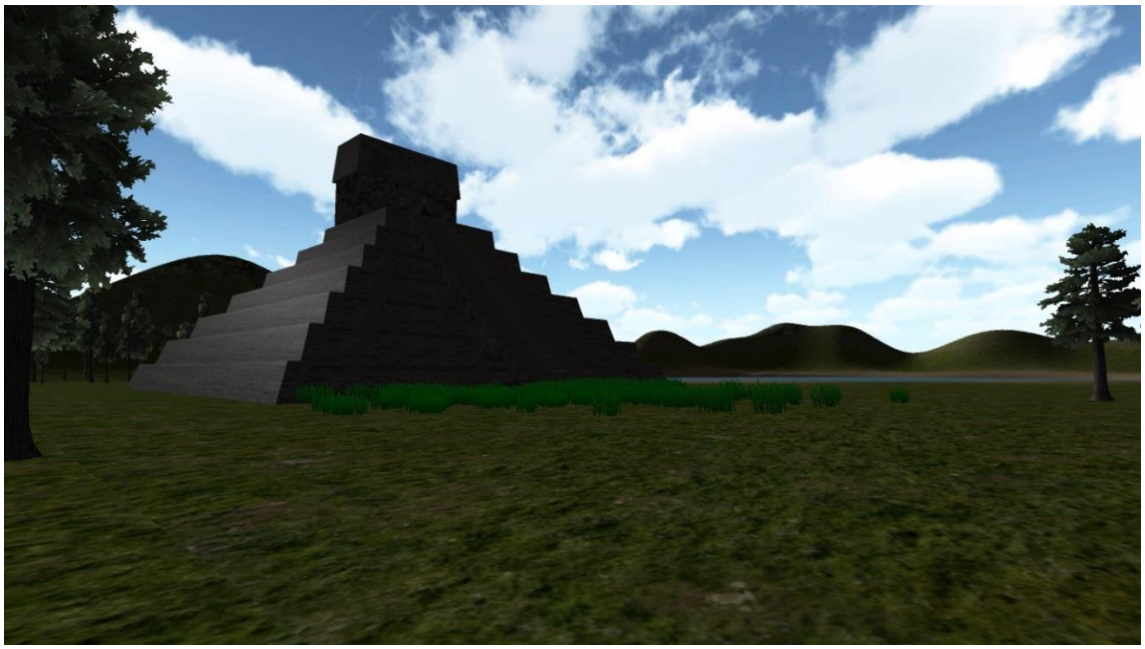
Fuente: elaboración propia, empleando Blender.

Al concluir la creación del modelo de la pirámide en Blender, se crearon dos pirámides dentro del juego desarrollado en la herramienta Unity utilizando este modelo.

Figura 69. **Pirámide I dentro del juego**

Fuente: elaboración propia, empleando Unity.

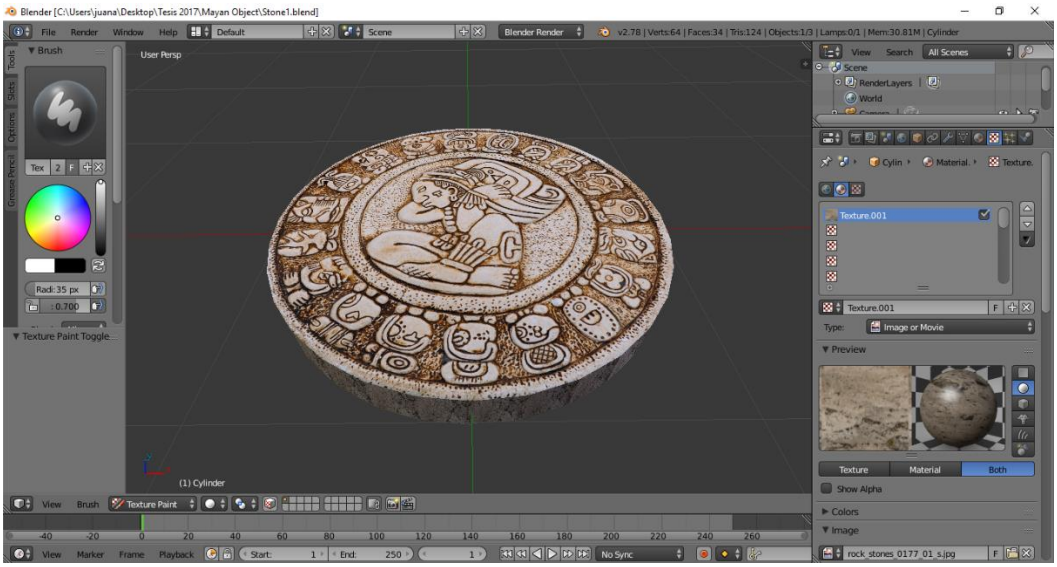
Figura 70. **Pirámide II dentro del juego**



Fuente: elaboración propia, empleando Unity.

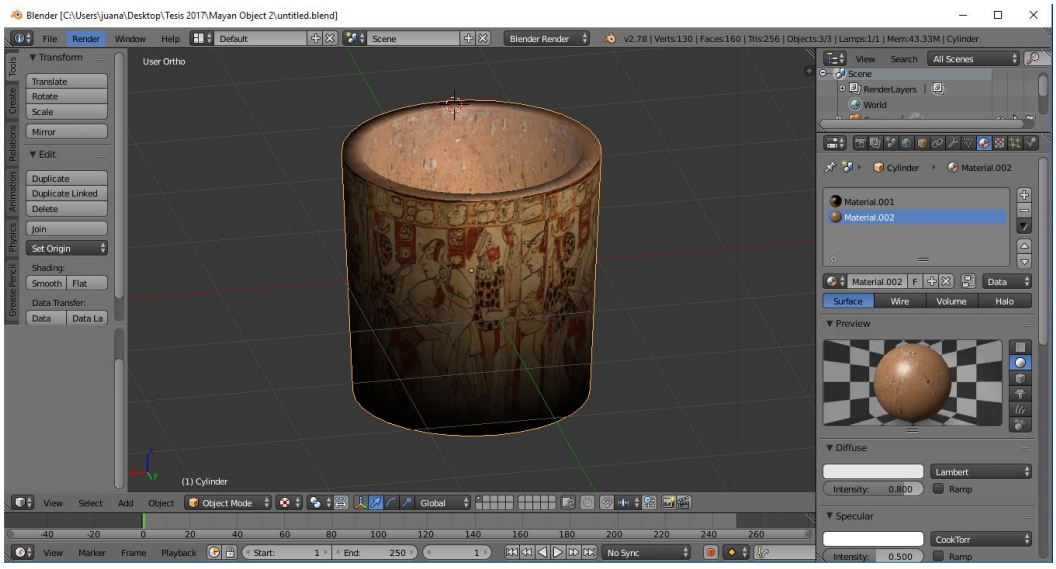
Además del modelo de la pirámide maya, se crearon también modelos de un calendario y de una vasija maya, con la herramienta Blender. Estos objetos fueron incorporados luego al videojuego creado con la herramienta Unity.

Figura 71. **Modelo de un calendario maya (creado en Blender)**



Fuente: elaboración propia, empleando Blender

Figura 72. **Modelo de una vasija maya (creado en Blender)**



Fuente: elaboración propia, empleando Blender.

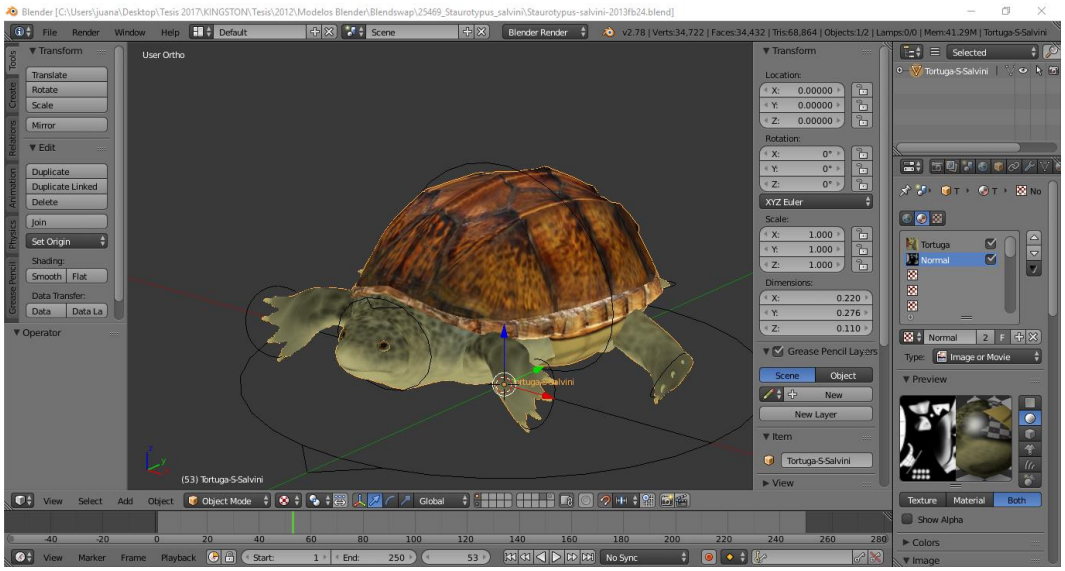
2.5.3. Personajes

Dado que el videojuego creado pertenece al género juego en primera persona, no fue necesario realizar un modelado 3D del personaje principal, debido a que la vista del jugador siempre está fijada desde la perspectiva de los ojos del personaje y se proyecta en todo momento una vista en primera persona de lo que éste está viendo. Esto hace que el jugador nunca observe directamente al personaje por lo que realizar el modelado 3D del mismo sería un gasto de recursos computaciones y puede afectar el rendimiento del juego, ya que representaría un procesamiento de polígonos innecesarios para el motor del juego.

Además del personaje principal, se realizó el modelado de otros personajes o actores que tendrán roles activos dentro del mismo. Igual que otros objetos del juego, el modelado de estos personajes se realizó utilizando la herramienta Blender, tomando como base modelos distribuidos por la comunidad de usuarios de Blender con licencia abierta.

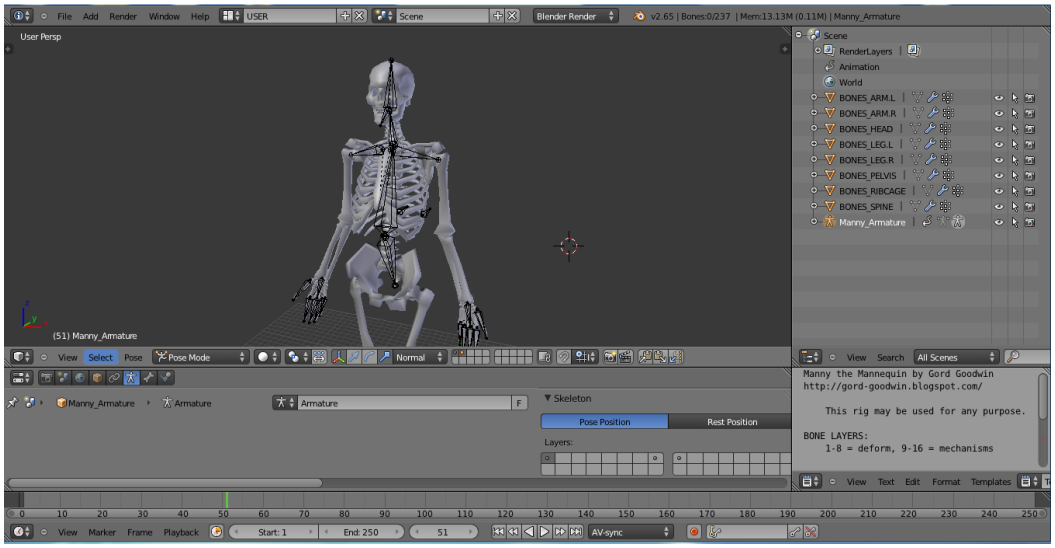
Los tres personajes modelados para formar parte del juego fueron un esqueleto animado, una tortuga y un quetzal. Estos objetos fueron luego instanciados el número de veces necesarias en Unity y animados para formar parte de la lógica de desarrollo del juego.

Figura 73. **Modelado del objeto tortuga**



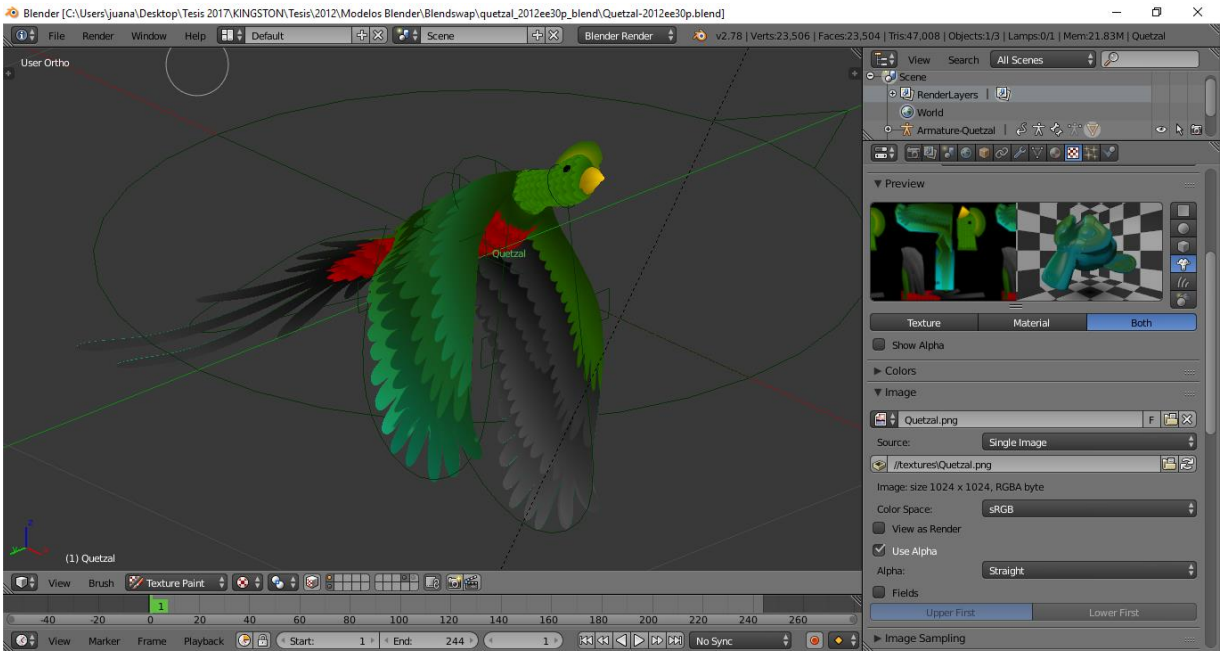
Fuente: elaboración propia, empleando Blender.

Figura 74. **Modelado del objeto esqueleto**



Fuente: elaboración propia, empleando Blender.

Figura 75. **Modelado del objeto quetzal**



Fuente: elaboración propia, empleando Blender.

2.6. Programación del videojuego

2.6.1. Descripción de la arquitectura general

La arquitectura para el juego creado se basó, como se mencionó anteriormente, en el motor de juegos Unity3D. Por medio de este motor se implementó toda la lógica del juego, utilizando las librerías y funciones implementadas en esta herramienta. Para la programación y personalización de esta lógica de la aplicación se utilizaron *scripts* programados en lenguaje Javascript utilizando el *IDE* Mono Develop. Los modelos de los objetos y personajes en tres dimensiones fueron creados utilizando Blender y se importaron estos modelos a Unity.

Una vez concluida la parte de programación en Unity se realizó la publicación del juego hacia las plataformas móviles Android y iOS a través del *plug-in* integrado en Unity para esta función.

En el caso de Android la exportación se realizó con el apoyo del Android *Software Development Kit* (SDK). El paso siguiente consistió en desplegar el archivo APK en un dispositivo móvil para verificar su correcto funcionamiento en el mismo.

Para el caso de iOS, la exportación se realizó con la utilización de la aplicación XCode[20] [21], la cual es un entorno de desarrollo integrado (*IDE*) para el desarrollo de aplicaciones para dispositivos Apple y el uso de un Apple ID (identificador de cuenta de servicios para la misma empresa Apple).

2.6.2. Animaciones

El flujo de trabajo utilizado para las animaciones de los personajes dentro del juego inició mediante la creación del movimiento individual de cada personaje en su fase de modelado en la herramienta Blender.

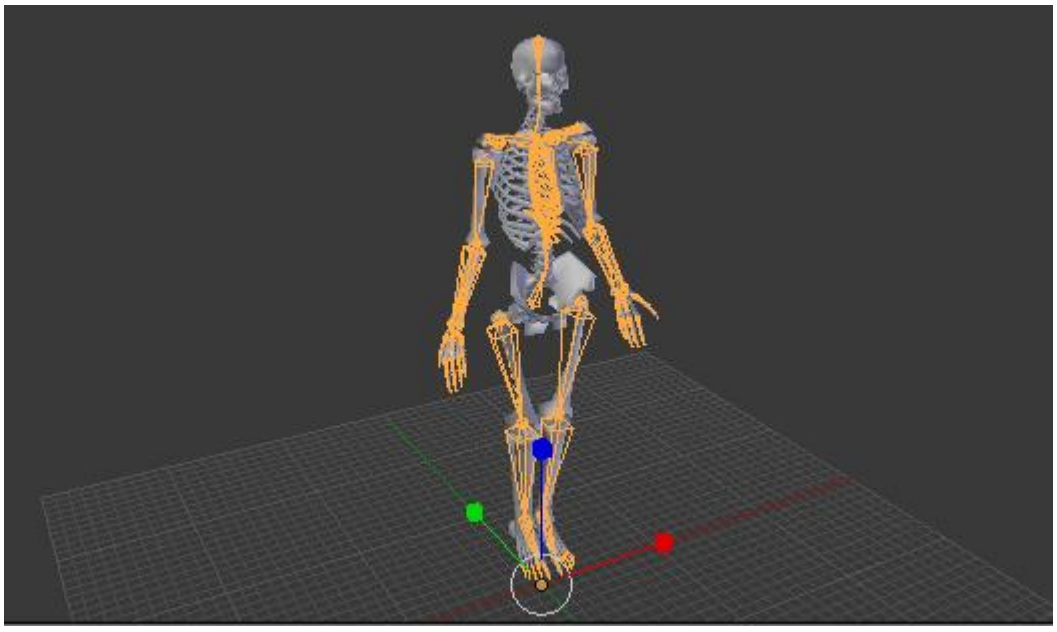
Dentro de Blender, los modelos creados pueden ser animados mediante la utilización de un esqueleto o *armature* que permite añadir movilidad a los modelos tridimensionales. Este proceso es conocido en inglés como *rigging*. Este proceso funciona básicamente mediante la adición de una estructura dentro del modelo a animar. Esta estructura simula los huesos que tienen los

[20] Fuente: *Building your Unity game to an iOS device for testing.* https://unity3d.com/es/learn/tutorials/topics/mobile-touch/building-your-unity-game-ios-device-testing. Consulta: mayo de 2017.
[21] Fuente: *Xcode.* https://developer.apple.com/xcode/. Consulta: mayo de 2017.

seres vivos vertebrados en la realidad. A cada uno de estos huesos se le asignan un grupo de vértices cercanos a él y cada vez que el hueso sea movido este grupo de vértices se deformará para seguir el flujo de movimiento del hueso. Estos movimientos pueden ser almacenados como animaciones junto con el modelo y al momento de realizar la importación de este modelo desde la herramienta Unity. Las animaciones son importadas de igual forma. Estas animaciones creadas en Blender son manejadas en Unity a través de *scripts* que controlan el momento y circunstancias en las cuales las animaciones serán ejecutadas.

En la siguiente figura se muestra el esqueleto (en color naranja) aplicado a uno de los personajes modelados para el juego:

Figura 76. **Animación del objeto "esqueleto"**



Fuente: elaboración propia, empleando Blender.

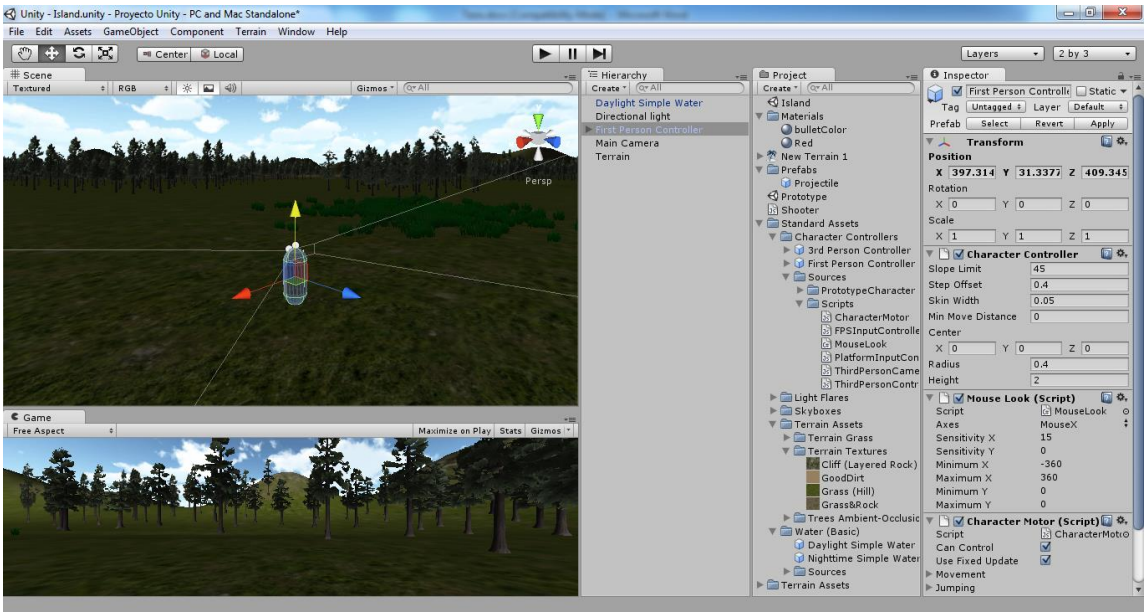
De la misma forma como se utilizó esta técnica para animar este objeto, se utilizó para animar a los modelos de la tortuga y el quetzal que forman parte del juego.

2.6.3. Controles

Los controles definidos para que el usuario pueda ejecutar acciones dentro del juego siguen el estándar que existe en la industria de los videojuegos para los juegos del tipo de primera persona.

Debido a la popularidad de este tipo de juegos, Unity ya trae por defecto un objeto prefabricado que implementa los controles para un jugador en primera persona. Este objeto se encuentra en el paquete de recursos estándar de la aplicación y tiene como nombre *FirstPersonController*. Para utilizarlo, se debe arrastrar el objeto a alguna parte de la escena creada y colocarlo arriba del terreno. El objeto tiene una forma de cápsula como se muestra en la siguiente figura.

Figura 77. **Objeto para implementar controles en primera persona**



Fuente: elaboración propia, empleando Unity.

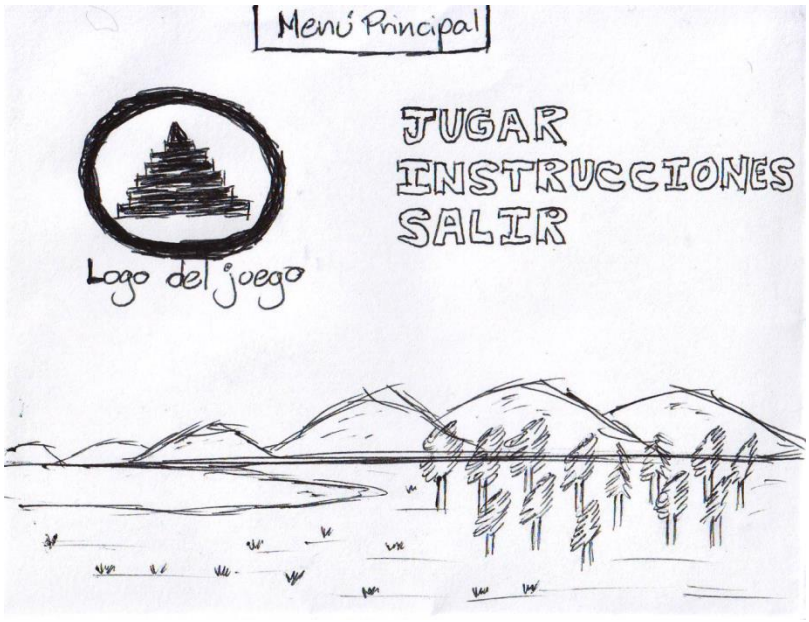
Este objeto, aparte de implementar los controles para que el usuario pueda interactuar con él y con el escenario, funciona como una cámara que representará la vista del jugador dentro del juego.

2.6.4. Menús

Para la interfaz de menús del juego se creó inicialmente un menú de inicio en el cual el jugador puede seleccionar tres opciones: 1. Jugar, 2. Ver las instrucciones del juego y 3. Salir del juego.

Para introducir al jugador a la atmósfera y ambiente del juego antes de que ingrese al juego, se decidió implementar un menú de inicio en el cual el fondo o *background* fuera una toma lejana del escenario creado para el juego.

Figura 78. **Bosquejo del menú principal**



Fuente: elaboración propia.

Para implementar esto, se duplicó la escena del juego para utilizar la copia como fondo del menú principal. Los objetos que se duplicaron en la nueva escena fueron el terreno, el agua del lago y la luz o iluminación (de tipo direccional), previamente añadidos a la escena. Para facilitar el manejo de estos objetos duplicados que forman parte del fondo del menú, se agruparon bajo un solo objeto llamado Escenario. El proceso de agrupación se realizó bajo una relación padre-hijo en Unity.

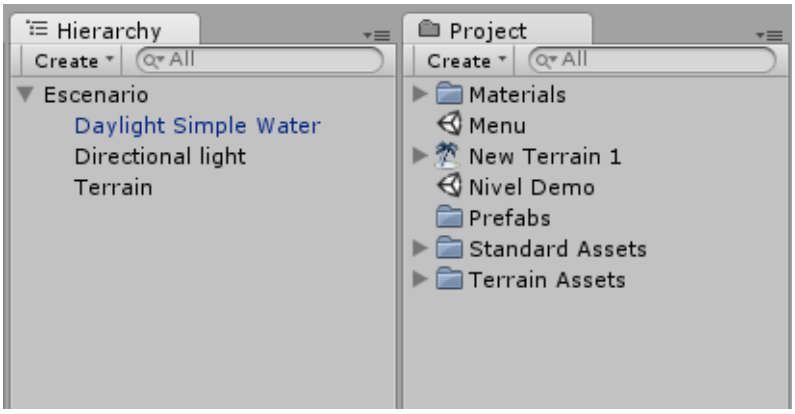
El proceso realizado para crear esta agrupación fue el siguiente:

Figura 79. **Objetos pertenecientes al escenario agrupados**

Fuente: elaboración propia, empleando Unity.

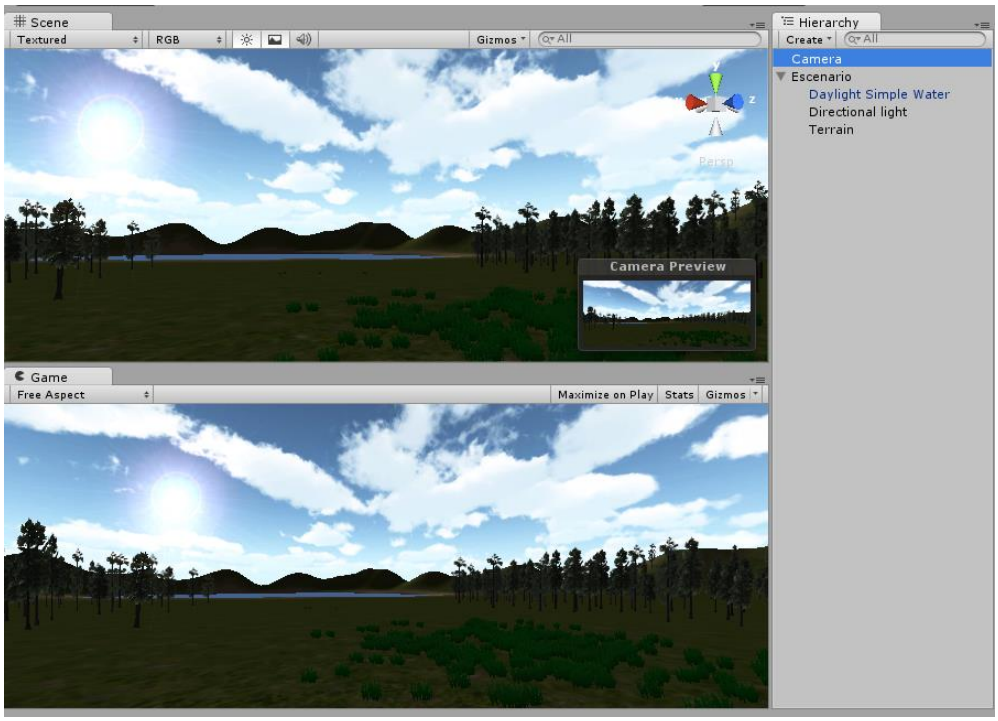
A continuación, se realizó el duplicado de la escena utilizando la función *Duplicate* de Unity y se nombró la nueva escena con el nombre *Menu*. Luego, se removieron los objetos que no forman parte del escenario en esta nueva escena, como los controles de primera persona, etc., borrando todos estos a excepción del objeto Escenario creado anteriormente (que incluye el terreno, el agua y la luz).

Figura 80. **Escena duplicada para crear el menú**



Fuente: elaboración propia, empleando Unity.

Figura 81. **Vista del menú y cámara agregada a la escena**

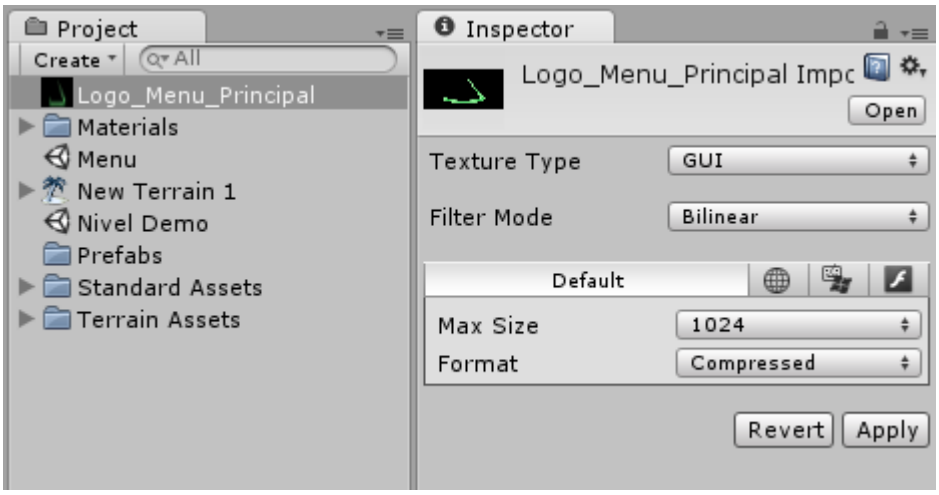


Fuente: elaboración propia, empleando Unity.

Para la creación del logo que se colocó en el menú principal se utilizó la herramienta Gimp, que es un programa de edición de imágenes gratuito muy popular actualmente.

Luego de haber creado la imagen y tenerla en formato PNG, se copió el archivo al folder de *Assets* de Unity para utilizarla dentro del juego. Una vez el archivo estuvo visible en la interfaz de Unity en el panel *Project* se modificaron las propiedades de importación del archivo para que fuera tratado como una textura tipo *GUI*, es decir, una textura 2D que se utiliza para desplegarse en pantalla en vez de aplicarse a objetos 3D.

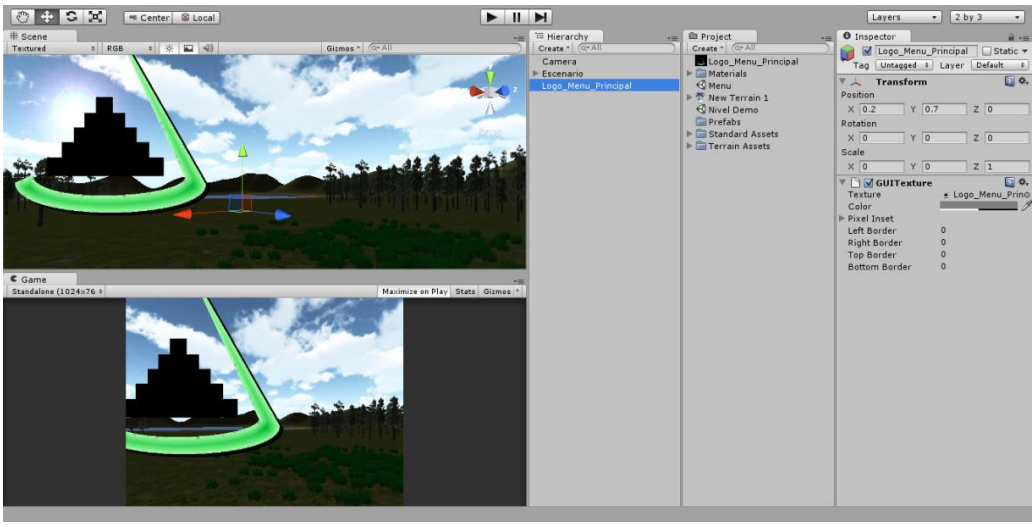
Figura 82. **Archivo de logo importado como textura *GUI***



Fuente: elaboración propia, empleando Unity.

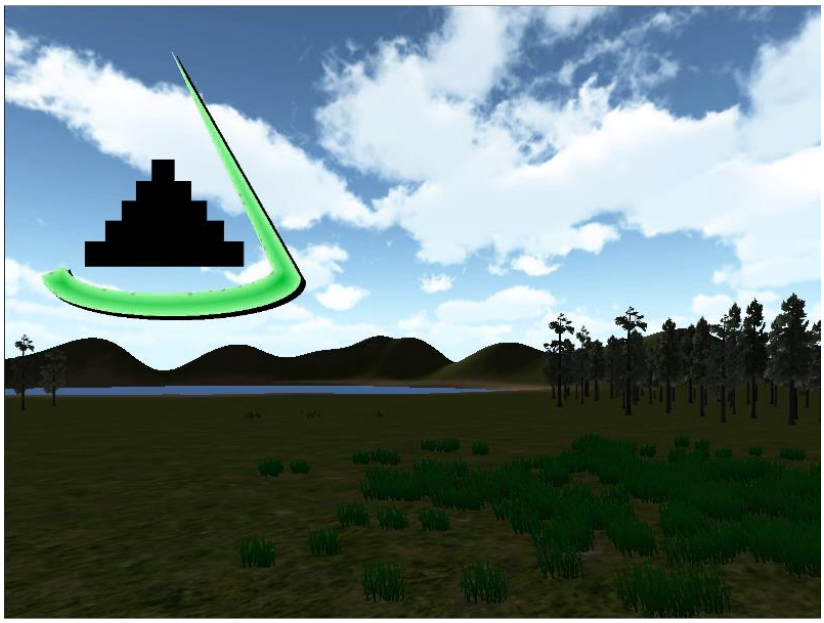
A continuación, se agregó un objeto de tipo *GUI Texture* a la escena para mostrar el logo en pantalla. Luego de haberlo agregado, se modificaron los valores de *Transform* de este nuevo objeto para alinear el logo a la parte superior izquierda de la pantalla. El resultado se muestra en la siguiente figura:

Figura 83. **Logo agregado a la escena**



Fuente: elaboración propia, empleando Unity.

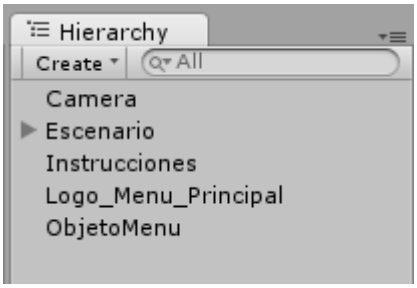
Figura 84. **Logo agregado a la escena, vista preliminar del menú**



Fuente: elaboración propia, empleando Unity.

Para mostrar los botones del menú en la vista creada anteriormente (que contiene el logo del juego), el primer paso fue el de añadir un objeto vacío (*EmptyObject*) a la escena del menú. Este objeto, nombrado ObjetoMenu dentro de la jerarquía de Unity, fue utilizado como el objeto "contenedor" del menú que se programó posteriormente.

Figura 85. **Objeto vacío (*EmptyObject*) agregado a la escena**



Fuente: elaboración propia, empleando Unity.

A continuación, se creó un nuevo JavaScript dentro del proyecto llamado GUIMenuPrincipal. Dentro de este *script,* se declararon 4 variables públicas con nombres skinMenu, areaMenu, botonJugar, botonInstrucciones y botonSalir.

Figura 86. **Código de declaración de variables**

```
var skinMenu : GUISkin;
var areaMenu : Rect;
var botonJugar : Rect;
var botonInstrucciones : Rect;
var botonSalir : Rect;
```

Fuente: elaboración propia, empleando Unity.

Luego se declaró la función *OnGUI()* dentro del *script*. Dentro de esta función se asignó el valor de la variable *skinMenu* a la propiedad *skin* del *GUI* del juego.

Figura 87. **Código de la función *OnGUI***

```
function OnGUI(){
    GUI.skin = skinMenu;

    GUI.BeginGroup(areaMenu);


    GUI.EndGroup();
}
```

Fuente: elaboración propia, empleando Unity.

A continuación, dentro de las dos sentencias *BeginGroup* y *EndGroup* colocadas anteriormente, se insertaron las instrucciones para crear los botones del menú por medio de la función *GUI.Button*. Como parámetros para esta función se ingresaron las variables públicas de tipo Rect creadas anteriormente para los botones y una cadena de texto que aparecerá sobre el botón.

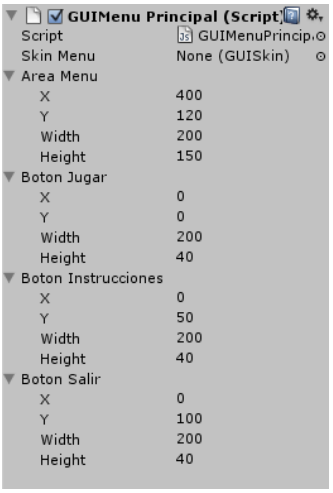
Figura 88. **Código de creación de botones del menú principal**

```
if(GUI.Button(Rect(botonJugar), "Jugar")){
}
if(GUI.Button(Rect(botonInstrucciones), "Instrucciones")){
}
if(GUI.Button(Rect(botonSalir), "Salir")){
}
```

Fuente: elaboración propia, empleando Unity.

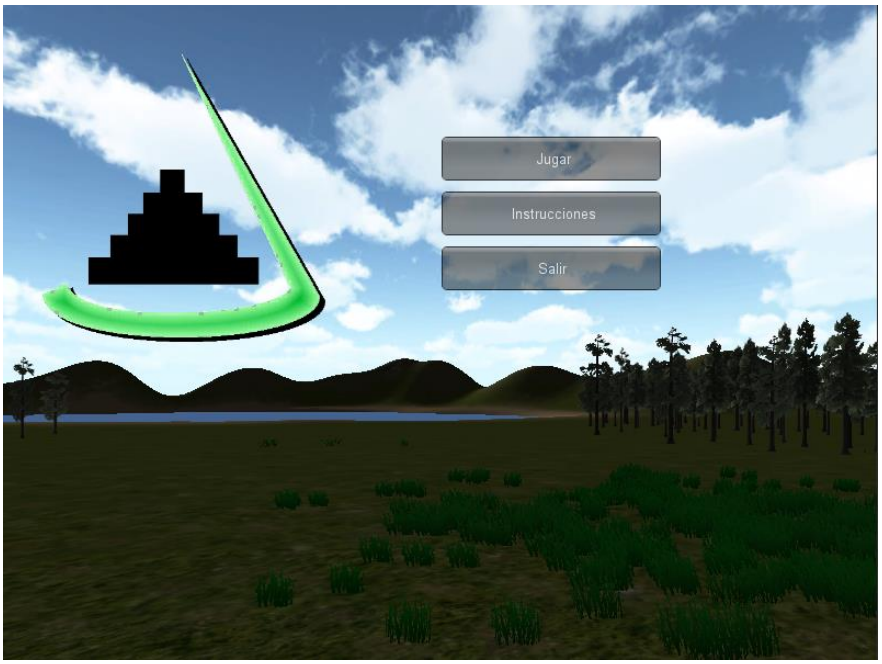
Una vez terminada la programación del *script*, se enlazó este al objeto ObjetoMenu creado anteriormente y añadido a la escena. Como siguiente paso, se le asignaron valores a las variables tipo Rect creadas dentro del *script* utilizando el Inspector de la interfaz de Unity. Los valores asignados se muestran en la siguiente figura.

Figura 89. **Valores de configuración del menú**



Fuente: elaboración propia, empleando Unity.

Figura 90. **Vista del menú finalizado**



Fuente: elaboración propia, empleando Unity.

Para la parte de programación de las acciones de los botones, se agregó una nueva función al script GuiMenuPrincipal. Esta función recibe como parámetro una cadena de texto indicando el nombre del nivel o escena que se debe cargar y en caso de recibir una entrada válida, carga la escena con la función *Application.LoadLevel*. En el caso de recibir como entrada la cadena "salir" se llama a la función *Application.Quit* para salir de la aplicación.

Figura 91. **Ejemplo de código de acción de los botones del menú**

```
function AccionBoton(nombreNivel : String){

    if(nombreNivel != "salir"){
        Application.LoadLevel(nombreNivel);
    }else{
        Application.Quit();
    }
}
```

Fuente: elaboración propia, empleando Unity.

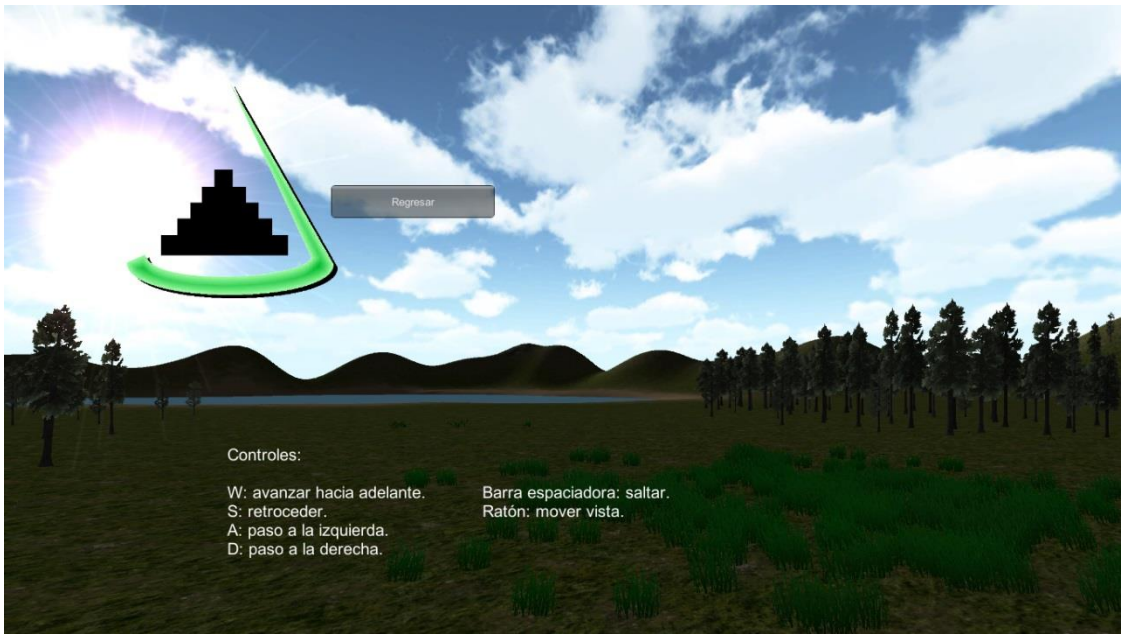
Para usar esta función, es necesario hacer una llamada a la misma cada vez que el usuario presione un botón del menú. Para esto, se introdujo una llamada en cada una de las sentencias *if* creadas dentro de la función *OnGUI* del script y se configuró el parámetro de acuerdo con el botón correspondiente.

Figura 92. **Código de llamado a las opciones del menú**

```
function OnGUI(){
    GUI.skin = skinMenu;

    GUI.BeginGroup(areaMenu);

    if(GUI.Button(Rect(botonJugar), "Jugar")){
        AccionBoton("Nivel Demo");
    }
    if(GUI.Button(Rect(botonInstrucciones), "Instrucciones")){
        AccionBoton("Instrucciones");
    }
    if(GUI.Button(Rect(botonSalir), "Salir")){
        AccionBoton("salir");
    }

    GUI.EndGroup();
}
```

Fuente: elaboración propia, empleando Unity.

Figura 93. **Menú “Instrucciones” mostrado en pantalla**



Fuente: elaboración propia, empleando Unity.

2.6.5. Física (colisiones)

Una parte esencial de la programación de videojuegos es la llamada detección de colisiones. Este término se aplica al acto de detectar cuando dos objetos dentro del juego interactúan físicamente. En Unity, esta detección se hace por medio de un componente llamado *Collider* el cual se agrega a los objetos existentes en el juego y trabaja como una red invisible que rodea estos objetos y sirve para detectar cuando dos o más objetos se intersectan dentro del escenario.

Los componentes de tipo *Collider* primitivos pueden tomar diversas formas geométricas básicas tridimensionales en Unity, como esferas, cajas, cápsulas, etc. Este tipo de *Colliders* simples ayudan a mejorar el rendimiento del juego

cuando se encuentra en su etapa de ejecución, en comparación a utilizar *Colliders* más complejos que se adaptan mejor a la forma de los objetos que envuelven. Por esta razón utilizan formas geométricas más complejas y necesitan más poder computacional para funcionar. En este proyecto, se utilizaron *Colliders* con forma de caja (*BoxCollider*) para envolver a todos los objetos en los que se necesitara computar colisiones.

Para añadir los componentes *BoxCollider* a los objetos de la escena se utilizó la opción *Physics* → *BoxCollider* ubicada en el menú *Component*. Se aplicó una vez seleccionado el objeto al cual se le quiera aplicar el componente de colisión. Al terminar de agregar el *Collider*, aparece un cubo de color verde envolviendo el objeto.

Para concluir la programación de las colisiones se añadió un componente de tipo *Rigidbody* a todos los objetos de la escena a los que se les agregó previamente un *BoxCollider*. Este componente de tipo *Rigidbody* hace que estos objetos estén bajo la influencia del motor de física de la aplicación y permite hacer funcionales las colisiones (es decir, que al detectarse la colisión de dos objetos se modifique su comportamiento). Sin los componentes de tipo *Rigidbody*, las colisiones entre dos o más objetos son detectadas, pero no se modifica el comportamiento de estos, lo cual permitiría, por ejemplo, que el personaje atravesara otros objetos del escenario como árboles o construcciones.

Para agregar el *RigidBody*, se seleccionó el objeto al que anteriormente se le había añadido un componente *Collider* y se utilizó la opción *Physics* → *Rigidbody* dentro del menú *Component*. Este procedimiento de agregar componentes de tipo *Collider* y *Rigidbody* fue realizado con todos los objetos modelados y que fueron agregados a la escena del juego.

2.6.6. Inteligencia Artificial

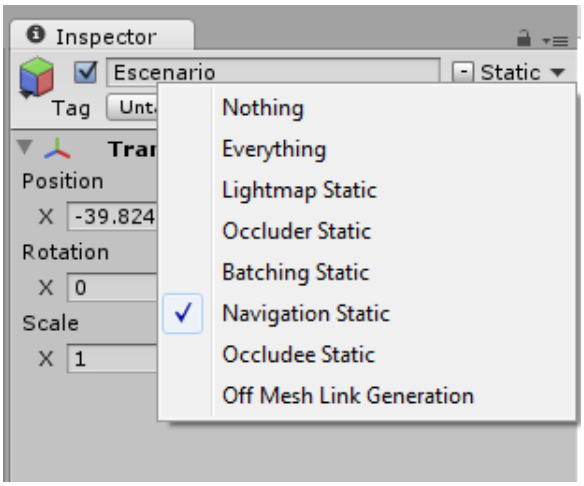
El componente de Inteligencia Artificial en el juego fue implementado utilizando las características Navmesh y *Pathfinding* proveídas por Unity las cuales permiten que los actores dentro del juego puedan moverse dentro del escenario buscando un objetivo o destino y, a la vez esquivar de forma inteligente los obstáculos presentes a través de toda la escena.

El objeto *Navigation Mesh* o Navmesh es básicamente una representación simplificada de la geometría de una escena.[22] Los actores o agentes utilizan esta representación para navegar o moverse a través de la escena, buscando caminos o *paths* para llegar a su destino.

Para crear un *Navigation Mesh* es necesario seleccionar el objeto terreno de la escena y en el panel Inspector cambiar la propiedad *Static* y habilitar la opción *Navigation Mesh*.

[22] *Navmesh and Pathfinding*. http://docs.unity3d.com/Documentation/Manual/NavmeshandPathfinding.html. Consulta: abril de 2017,

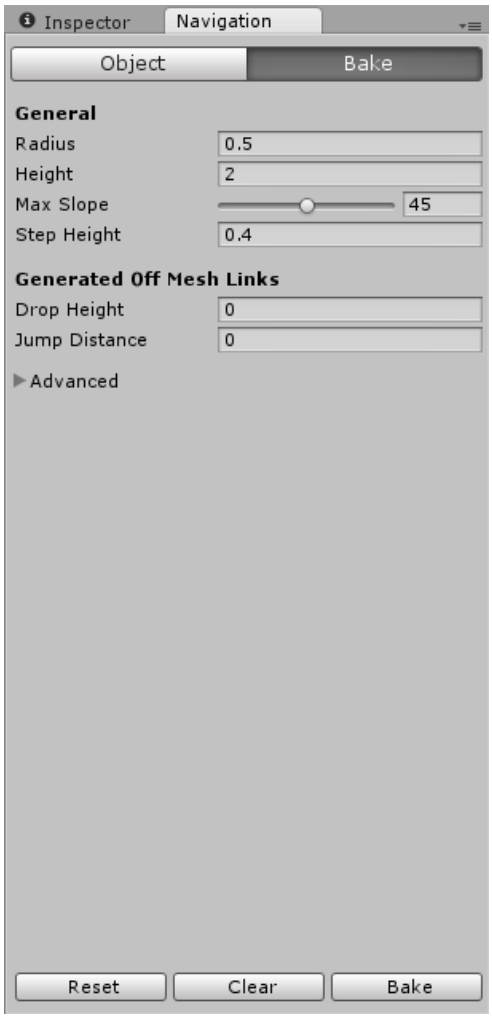
Figura 94. **Habilitando la opción *Navigation Mesh* para el terreno**



Fuente: elaboración propia, empleando Unity.

A continuación, se desplegó la ventana de navegación en el editor de Unity seleccionando la opción *Windows* → *Navigation*. Esto hace que se muestre un nuevo panel de título *Navigation* en la parte derecha de la interfaz de Unity. En este panel se encuentra la opción para generar automáticamente toda la geometría para que los actores naveguen a través de ella. Para esto se utiliza la opción *Bake* que se encuentra en este panel.

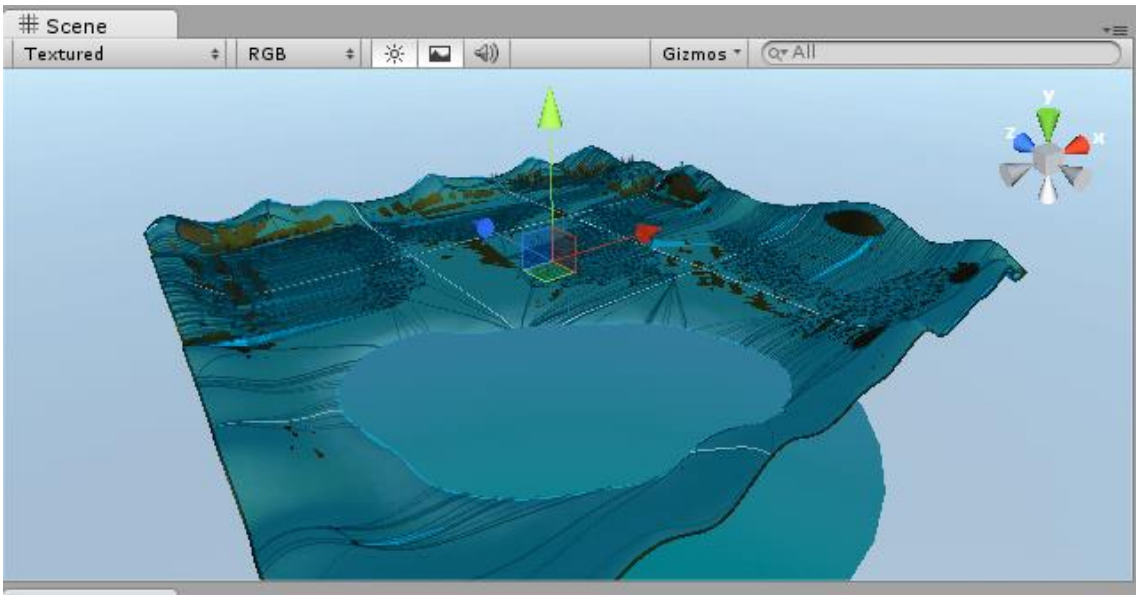
Figura 95. **Creando el *Navigation Mesh***



Fuente: elaboración propia, empleando Unity.

Una vez creada esta geometría, se mostrará de esta forma en la vista de la escena del juego:

Figura 96. **Vista de la geometría de navegación creada**



Fuente: elaboración propia, empleando Unity.

Una vez elaborada esta geometría, el último paso consistió en definir un destino u objetivo para cada uno de los personajes del juego (por medio de la edición de la propiedad *Destination* del personaje) y automáticamente este utilizará la geometría de navegación para calcular rutas y moverse a través de la escena de una forma eficiente.

2.6.7. Sonido

El sonido en Unity se trabaja por medio de dos componentes: la fuente de audio (*Audio Source*) y otro componente conocido como Listener. El Listener viene incorporado por defecto en cualquier objeto de tipo cámara que se agregue a una escena (en el caso del proyecto trabajado hay que recordar que la cámara es parte del objeto que forma el personaje del juego), y se encarga

de “escuchar” o recoger los sonidos producidos por las fuentes de audio y transmitirlos hacia el jugador. En otras palabras, el *Listener* simula los oídos del videojugador.

- Sonidos de ambiente

Existen dos tipos de sonidos que pueden ser configurados en Unity: sonidos 2D y sonidos 3D. Los sonidos 2D mantendrán el mismo volumen de reproducción independientemente de dónde se encuentre el objeto *Listener* en la escena de Unity, mientras que los sonidos en 3D actúan de forma dinámica, sonando más fuerte mientras el objeto *Listener* (el jugador) se encuentre más cerca de ellos y reduciendo su volumen a medida que el jugador se aleja.

En el caso de los sonidos de ambiente que se agregaron al juego, se configuraron como sonidos 2D para que se escucharan de forma constante a través de todo el escenario creado. Estos sonidos agregados consistieron básicamente en sonidos que pueden esperar escucharse en un ambiente de bosque como el que se creó en la escena (sonidos de árboles, pájaros, etc.).

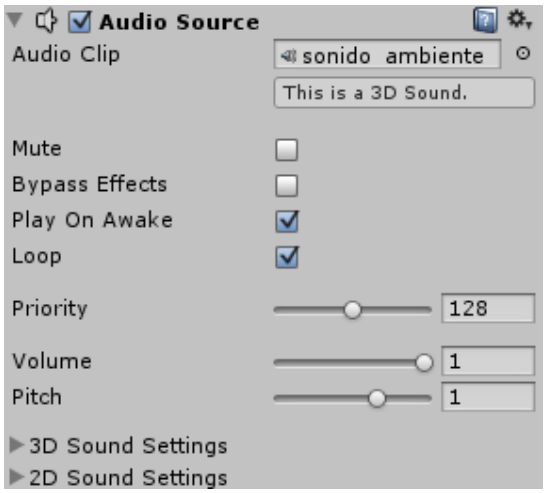
Los archivos fuente para configurar los sonidos ambientales fueron obtenidos del repositorio de sonidos en línea http://www.freesound.org. Este sitio web permite compartir y descargar archivos de sonido con licencia *Creative Commons* 0 para su uso en cualquier proyecto. La licencia de tipo *Creative Commons* 0 permite al creador de un contenido específico renunciar a sus derechos sobre el mismo y permitir que cualquier otra persona pueda usar estos contenidos libremente y sin ningún tipo de restricción.[23]

[23] **Fuente:** *CC0 - “No Rights Reserved”.* https://creativecommons.org/share-your-work/public-domain/cc0**.** Consulta: mayo de 2017.

El proceso para agregar los sonidos al juego dentro de la interfaz de Unity, una vez recolectados los archivos fuente, inició mediante la adherencia de un componente de tipo *Audio Source* al objeto terreno en la escena creada. Para realizar esto se seleccionó el objeto en la jerarquía de objetos de la escena. A continuación, en el menú *Component* de la interfaz de Unity se seleccionó la opción *Audio → Audio Source*.

Luego de agregar este componente, se realizó la configuración del mismo. En la opción *Audio Clip* se seleccionó el archivo fuente del audio y se activaron las opciones *Play On Awake* y *Loop* para iniciar la reproducción del sonido cuando el jugador aparezca en la escena y que el mismo se reproduzca una y otra vez ininterrumpidamente.

Figura 97. **Configuración de sonidos en Unity**



Fuente: elaboración propia, empleando Unity.

2.7. Pruebas y corrección de errores/bugs

- Fase 1: pruebas en PC

Debido a que la aplicación fue desarrollada utilizando la herramienta Unity para el sistema operativo Windows, la mayor parte del proceso de corrección de errores y *bugs* fue realizado en este ambiente. Una vez terminada esta etapa de pruebas y corrección de errores en esa plataforma, se realizaron las pruebas en las plataformas Android y iOS.

La mayor parte de esta primera etapa de pruebas se dedicó a probar las colisiones dentro de la aplicación y los *scripts* de comportamiento programados. También hubo cierto enfoque en verificar la parte gráfica del juego y realizar algunas mejoras a la misma para mejorar su rendimiento en el hardware más limitado que se encuentra en los dispositivos móviles.

- Fase 2: pruebas en dispositivos móviles

    - Android

Una vez finalizado el proceso de corrección de *bugs*/errores de la fase anterior, se seleccionaron tres dispositivos Android distintos para realizar pruebas de usabilidad de la aplicación en los mismos. Se trató de utilizar dispositivos con diferentes niveles de especificación de hardware para asegurarse de que el juego se ejecutara fluidamente tanto en teléfonos con el hardware más reciente como con hardware de un nivel inferior. A continuación, se muestran las especificaciones de los tres dispositivos seleccionados:

- Samsung Galaxy S7:

Tabla III. **Especificaciones Samsung Galaxy S7**

Tamaño de pantalla:	1440 x 2560 pixeles, 5.1 pulgadas.
Multitouch	Sí
Memoria RAM	4 GB RAM
Chipset	Exynos 8890 Octa
CPU	Octa-core (4x2.3 GHz Mongoose & 4x1.6 GHz Cortex-A53)
GPU	Mali-T880 MP12
Sensores	Sensor de huellas, acelerómetro, giroscopio, sensor de proximidad, brújula, barómetro, ritmo cardiaco, SpO2.
Sistema Operativo:	Android 6.0 (Marshmallow)

Fuente: *Samsung Galaxy S7*. http://www.gsmarena.com/samsung_galaxy_s7-7821.php, Consulta: mayo de 2017.

- Huawei Y3:

Tabla IV. **Especificaciones Huawei Y3II**

Tamaño de pantalla:	480 x 854, 4.5 pulgadas
Multitouch	Sí
Memoria RAM	1 GB RAM
Chipset	Mediatek MT6735M
CPU	Quad-core 1.0 GHz Cortex-A53
GPU	Mali-T720MP2
Sensores	Acelerómetro, proximidad.
Sistema Operativo:	Android 5.1 (Lollipop)

Fuente: *Huawei Y3II*. http://www.gsmarena.com/huawei_y3ii-8037.php, Consulta: mayo de 2017.

- Samsung Galaxy J5 (2016)

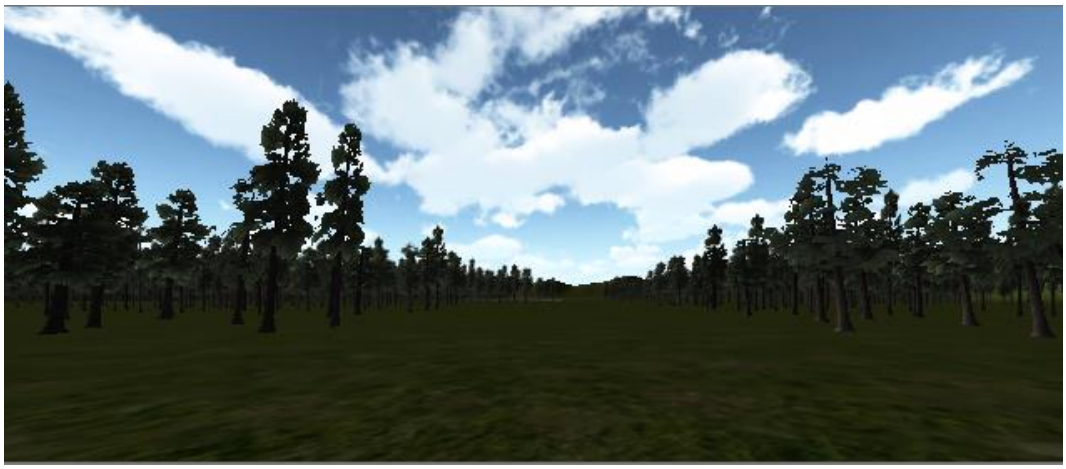
Tabla V. **Especificaciones Samsung Galaxy J5 (2016)**

Tamaño de pantalla:	720 x 1280 pixeles, 5.2 pulgadas.
Multitouch	Sí
Memoria RAM	2 GB RAM
Chipset	Qualcomm MSM8916 Snapdragon 410
CPU	Quad-core 1.2 GHz Cortex-A53
GPU	Adreno 306
Sensores	Acelerómetro, proximidad.
Sistema Operativo:	Android 6.0.1 (Marshmallow)

Fuente: *Samsung Galaxy J5 (2016).* http://www.gsmarena.com/samsung_galaxy_j5_(2016)-7869.php. Consulta: mayo de 2017.

Debido a las mejoras de rendimiento gráfico que fueron realizadas en la primera fase de pruebas en la computadora, la aplicación se ejecutó con fluidez en los tres teléfonos móviles descritos anteriormente, obteniendo un *framerate* aceptable en cada uno de estos dispositivos.

Figura 98. **Aplicación ejecutándose en dispositivo Android**



Fuente: elaboración propia, empleando Unity.

- iOS:

Para realizar las pruebas correspondientes en sistema operativo iOS se utilizó un dispositivo iPhone 6S con las siguientes características:

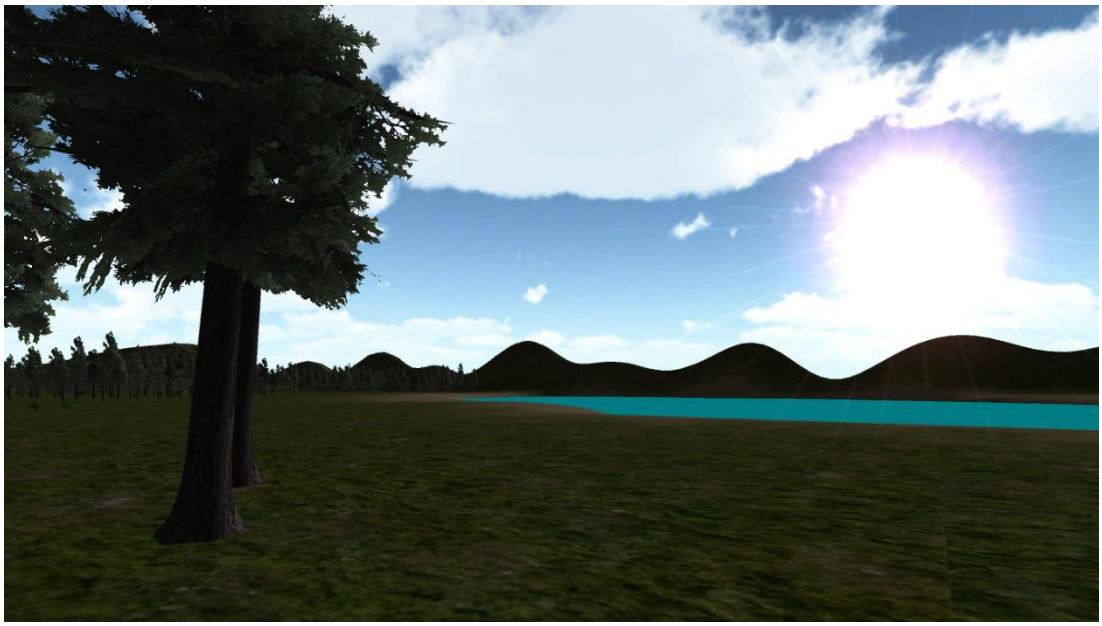
Tabla VI. **Especificaciones iPhone 6S**

Tamaño de pantalla:	750 x 1334 pixeles, 4.7 pulgadas
Multitouch	Sí
Memoria RAM	2 GB RAM
Chipset	Apple A9
CPU	Dual-core 1.84 GHz Twister
GPU	PowerVR GT7600 (six-core graphics)
Sensores	Sensor de huellas, acelerómetro, proximidad, giroscopio, brújula, barómetro.
Sistema Operativo:	iOS 10.3

Fuente: *Apple iPhone 6s.* http://www.gsmarena.com/apple_iphone_6s-7242.php.

Consulta: mayo de 2017.

Figura 99. **Aplicación ejecutándose en el dispositivo iOS**



Fuente: elaboración propia, empleando Unity.

3. UTILIZACIÓN DEL VIDEOJUEGO COMO HERRAMIENTA DE APRENDIZAJE

Una vez concluido el desarrollo de la aplicación de demostración del videojuego diseñado, se utilizó esta aplicación para realizar una investigación sencilla para determinar su utilidad y si cumple con el objetivo de ser utilizada como una herramienta de aprendizaje.

3.1. Descripción de la investigación

Para la realización de esta investigación se utilizó una metodología o proceso bastante simple: se seleccionó una muestra de diez personas siguiendo los criterios que luego se expondrán. A continuación, se hizo que las personas respondieran un pequeño cuestionario con cinco preguntas sobre historia/cultura de Guatemala. Luego, los participantes utilizaron la aplicación desarrollada sin ninguna ayuda o capacitación previa por parte del desarrollador, únicamente guiándose por la ayuda al usuario implementada dentro del videojuego. Por último, luego de haber terminado de utilizar el juego, los usuarios respondieron de nuevo el cuestionario de cinco preguntas sobre los hechos históricos y culturales que fueron implementados dentro de la historia y ambiente del juego, para comprobar, de esta forma, si se había adquirido algún tipo de aprendizaje al utilizar la aplicación reflejado en una mejor puntuación en la prueba.

3.1.1. Hipótesis

La hipótesis que se espera comprobar mediante esta investigación es la siguiente:

Por medio de los videojuegos educativos es posible transmitir conocimiento de forma eficaz a las personas.

3.2. Selección de la muestra

De acuerdo con los objetivos planteados para esta investigación, se seleccionó a diez personas para que probaran el videojuego implementado y de esta forma medir si habían capturado algún tipo de conocimiento de esta experiencia.

El criterio principal para seleccionar la muestra fue obtener un grupo de individuos de diferentes edades, con experiencia previa en el uso de videojuegos o sin ella. Se seleccionaron individuos con estas características para probar el nivel de usabilidad de la aplicación.

Siguiendo estos dos criterios, se seleccionó a diez personas con las siguientes características:

- De acuerdo con la edad:

  De 10 a 20 años: 3 personas
  De 21 a 40 años: 3 personas
  De 41 a 60 años: 2 personas
  De 60 años en adelante: 2 personas

- Experiencia previa con videojuegos:

  Con experiencia: 5 personas
  Sin experiencia: 5 personas

3.2.1. Realización de la investigación

El cuestionario utilizado para la investigación consistió de las siguientes preguntas, todas ellas basadas en el contenido del videojuego desarrollado (el cual a su vez está basado en el libro Popol Vuh como se especificó anteriormente):

¿Con que nombre se le conoce al inframundo en la mitología maya?

¿Quiénes son los dioses gemelos?

¿En qué se convirtieron los dioses gemelos luego de su victoria en el inframundo?

¿Cuál es el nombre de la madre de los dioses gemelos?

¿Quiénes son los dos principales dioses del inframundo maya?

Las respuestas correctas a cada una de estas preguntas se muestran a continuación:

a) Xibalbá

b) Hunapú e Ixbalanqué

c) El Sol y la Luna

d) Ixquic

e) Vucub-Camé y Hun-Camé

La información relacionada con cada una de estas preguntas se le muestra al usuario a lo largo del desarrollo del videojuego, cada vez que se encuentra un objeto relacionado con la cultura maya dentro del mismo.

Figura 100. **Inicio del juego**

Fuente: elaboración propia, empleando Unity.

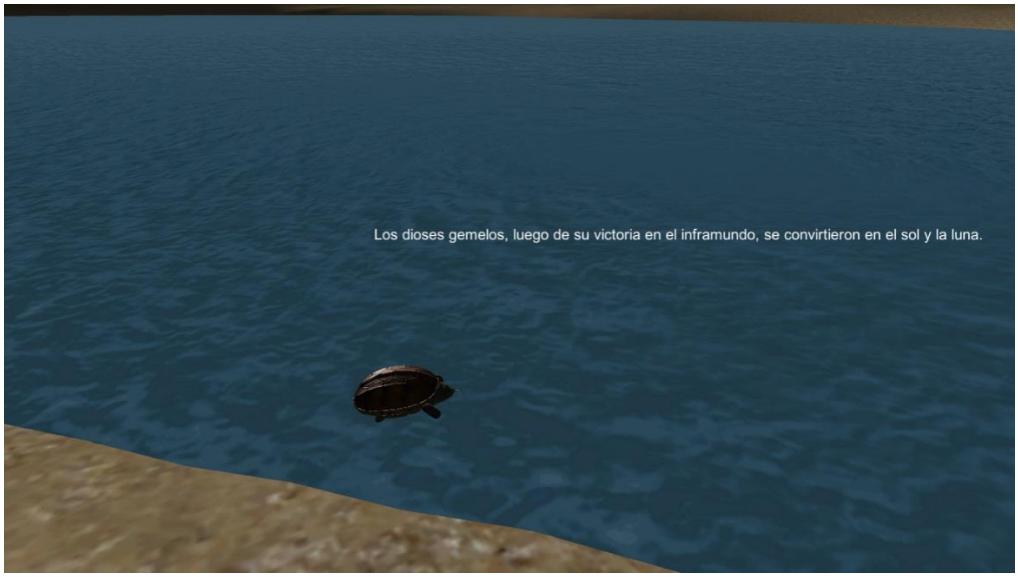
Figura 101. **Información mostrada al encontrar el esqueleto**

Fuente: elaboración propia, empleando Unity.

Figura 102. **Información mostrada al encontrar la vasija maya**

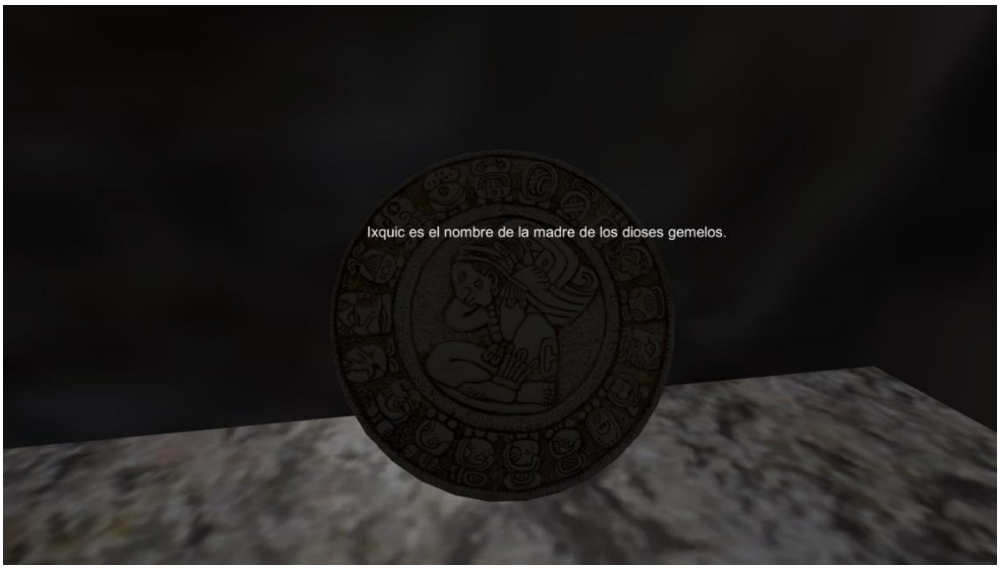
Fuente: elaboración propia, empleando Unity.

Figura 103. **Información mostrada al encontrar la tortuga**



Fuente: elaboración propia, empleando Unity.

Figura 104. **Información encontrada al encontrar el calendario maya**



Fuente: elaboración propia, empleando Unity.

Figura 105. **Información encontrada al encontrar el quetzal**

Fuente: elaboración propia, empleando Unity.

3.3. Resultados obtenidos

A continuación, se muestran los resultados obtenidos por los usuarios en el cuestionario que respondieron sobre historia y cultura de Guatemala antes y después de haber jugado el juego. En la tabla de resultados se excluyen los criterios de edad y experiencia en el uso de los videojuegos ya que estos no afectan en la medición de conocimiento adquirido. Como se mencionó anteriormente, estos criterios fueron utilizados para medir la usabilidad de aplicación.

- Antes de la utilización de la aplicación

Tabla VII. **Resultados de la prueba antes de la utilización del juego**

Pregunta	Porcentaje de Respuestas Correctas
A	50%
B	30%
C	20%
D	10%
E	0%

Fuente: elaboración propia.

- Después de la utilización de la aplicación

Tabla VIII. **Resultados de la prueba después de la utilización del juego**

Pregunta	Porcentaje de Respuestas Correctas
A	80%
B	80%
C	80%
D	50%
E	50%

Fuente: elaboración propia.

Como puede observarse en las tablas de resultados, luego de la utilización del videojuego, el porcentaje de respuestas correctas al responder el cuestionario tuvo un aumento significativo entre los participantes de la investigación, lo cual afirma la hipótesis planteada: el uso de un videojuego educativo funciona de una buena forma para enseñar conocimientos a personas.

De acuerdo con la usabilidad de la aplicación, todos los participantes pudieron completar el nivel de demostración del videojuego utilizando un teléfono móvil de pantalla táctil con sistema Android. Como era de esperarse, se observó una mayor dificultad de uso en las personas de mayor edad y sin experiencia en el uso de este tipo de software, pero esta no alcanzó niveles que impidieran completar todo el juego a este grupo de usuarios, únicamente les tomó más tiempo recorrer la curva de aprendizaje para hacer un uso óptimo de la aplicación.

CONCLUSIONES

1. Como se demostró por medio de las pruebas de conocimiento realizadas a los usuarios que utilizaron el videojuego educativo implementado, este tipo de aplicaciones son una forma efectiva de mejorar los procesos de aprendizaje/enseñanza tradicionales, ya que presentan al usuario el conocimiento de un modo interactivo y diferente, lo cual aumenta el interés del mismo en la información presentada en comparación a otros métodos tradicionales de enseñanza.

2. La utilización de un motor de juegos para el desarrollo de este tipo de aplicaciones simplifica de gran manera el trabajo de programación y modelado que se debe realizar para crear un videojuego de calidad profesional. Además, el uso de un motor multiplataforma permite reducir el costo de desplegar y publicar la aplicación a diferentes sistemas.

3. Guatemala posee una rica historia y cultura que puede ser utilizada perfectamente para desarrollar videojuegos que utilicen estos elementos como base de su contenido. Esto fomentaría el conocimiento entre las personas guatemaltecas sobre su propia cultura e historia y daría a conocer más estos elementos a personas extranjeras.

4. Por medio de plataformas como Android y iOS, y gracias a la rápida evolución del hardware de los dispositivos móviles, en la actualidad, es posible promover y globalizar entre personas de todas las edades y condiciones socioeconómicas videojuegos complejos y de alto rendimiento que antes estaban destinados de forma casi exclusiva al uso

de videojugadores expertos poseedores de hardware altamente especializado para este tipo de aplicaciones (consolas y computadores de alto costo).

RECOMENDACIONES

1. Instituciones educativas y de turismo de Guatemala tienen una oportunidad en el desarrollo de videojuegos educativos para promover la historia y cultura del país entre la juventud nacional e internacional, por lo que se insta a este tipo de instituciones a considerar implementar este tipo de ideas de un corto a mediano plazo.

2. Durante el proceso de desarrollo de un videojuego es importante realizar una cuidadosa selección de las herramientas para su creación, sobre todo en lo referente al motor de juegos ya que los diferentes motores disponibles en el mercado tienen diversas características que se adaptan de mejor o peor forma a distintos géneros de videojuegos.

3. Cuando se desarrolla un videojuego de tipo educativo es necesario no olvidarse que gran parte del éxito de estas herramientas como promotoras de conocimiento radica en el entretenimiento que brinda a los usuarios, además de la información que les exhorta a aprender. Por ello, surge la necesidad de encontrar un balance entre el contenido educativo y de entretenimiento, para no desmotivar al usuario a probar y usar la herramienta o aplicación.

BIBLIOGRAFÍA

1. ADAMS, Ernest. *Fundamentals of Game Design*. 3a Edición. Estados Unidos: New Riders, 2013. 576 p.

2. BLACKMAN, Sue. *Beginning 3D Game Development with Unity 4: All-in-one, multi-platform game development*. 2a Edición. New York, USA: Apress, 2013. 808 p.

3. GOLDSTONE, Will. *Unity 3.x Game Development Essentials – Game development with C# and Javascript*. 2a Edición. Birmingham, UK: Packt Publishing Ltd, 2011. 462 p.

4. ZECHNER, Mario; DIMARZIO, J.F.; GREEN, Robert. *Beginning Android Games*. 3a Edición. New York, USA: Apress, 2016. 605 p.