



**UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL
FACULTAD DE INGENIERÍA
CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES**

TÍTULO:

**Diseño e implementación de un simulador de conducción
vehicular utilizando un motor de videojuegos.**

AUTORES:

**Sarabia Lúa, Ginnio Andrés
Guananga Bernabé, Jairo Alejandro**

**Trabajo de Titulación previo a la Obtención del Título de:
INGENIERO EN SISTEMAS COMPUTACIONALES**

TUTOR:

Ing. Salazar Tovar, Cesar Adriano, Mgs

**Guayaquil, Ecuador
22 de septiembre del 2016**



UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL
FACULTAD DE INGENIERÍA
CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES

CERTIFICACIÓN

Certificamos que el presente trabajo fue realizado en su totalidad por nosotros **Sarabia Lúa Ginnio Andrés ; Guananga Bernabé Jairo Alejandro**, como requerimiento parcial para la obtención del Título de **Ingeniero en Sistemas Computacionales**.

TUTOR

Ing. Cesar Salazar, Mgs

DIRECTORA DE CARRERA

Ing. Beatriz Guerrero Yépez, Mgs

Guayaquil, a los 22 días del mes de septiembre del año 2016



UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL
FACULTAD DE INGENIERÍA
CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES

DECLARACIÓN DE RESPONSABILIDAD

Nosotros,

Sarabia Lúa, Ginnio Andrés y Guananga Bernabé, Jairo Alejandro

DECLARAMOS QUE:

El Trabajo de Titulación **Diseño e implementación de un simulador de conducción vehicular utilizando un motor de videojuegos** previo a la obtención del Título de **Ingeniero en Sistemas Computacionales**, ha sido desarrollado en base a una investigación exhaustiva, respetando derechos intelectuales de terceros, cuyas fuentes se incorporan en la bibliografía. Consecuentemente este trabajo es de nuestra total autoría.

En virtud de esta declaración, nos responsabilizamos del contenido, veracidad y alcance científico del Trabajo de Titulación referido.

Guayaquil, a los 22 días del mes de septiembre del año 2016

EL AUTOR

Sarabia Lúa, Ginnio Andrés

EL AUTOR

Guananga Bernabé, Jairo Alejandro



UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL
FACULTAD DE INGENIERÍA
CARRERA DE INGENIERÍA EN SISTEMAS
COMPUTACIONALES

AUTORIZACIÓN

Nosotros,

Sarabia Lúa, Ginnio Andrés y Guananga Bernabé, Jairo Alejandro

Autorizamos a la Universidad Católica de Santiago de Guayaquil, la **publicación** en la biblioteca de la institución del Trabajo de Titulación: **Diseño e implementación de un simulador de conducción vehicular utilizando un motor de video juegos**, cuyo contenido, ideas y criterios son de nuestra exclusiva responsabilidad y total autoría.

Guayaquil, a los 22 días del mes de septiembre del año 2016

EL AUTOR

Sarabia Lúa, Ginnio Andrés

EL AUTOR

Guananga Bernabé, Jairo Alejandro

REPORTE URKUND



Documento [Tesis 2016-08-10 URKUND.docx \(D21351487\)](#)

Presentado 2016-08-09 23:54 (-05:00)

Presentado por Ginnio Andres Sarabia Lua (sarabia.seven@gmail.com)

Recibido cesar.salazar.ucsg@analysis.urkund.com

Mensaje Documento de tesis 2016-08-10 para análisis URKUND [Mostrar el mensaje completo](#)

1% de esta aprox. 24 páginas de documentos largos se componen de texto presente en 1 fuentes.

AGRADECIMIENTO

Agradezco a mis padres Jorge Ginnio Sarabia Valdiviezo y Leida Amada Lúa Morante por apoyarme incondicionalmente en este largo camino, en levantarme cuando yo no veía otra opción más que renunciar.

A mi hermano Jorge Antonio Sarabia Lúa por guiarme siempre con sus consejos y experiencia.

Y finalmente a mi novia Joselyn Isabella Avilez Guerrero, mi compañera que me enseñó que puedo ser mejor, y me acompañó de la mano en este trayecto.

SARABIA LUA, GINNIO ANDRES

DEDICATORIA

El presente trabajo se los dedico a mi abuelos, en especial a mi abuelo paterno: Jorge Enrique Sarabia Álvarez que siempre me inculco estudiar duro, para lograr un título de educación superior.

A mis padres y hermanos por siempre estar ahí conmigo en los buenos momentos y en los no tan buenos.

A mis primos menores los cuales en este momento se encuentran cursando escuela y bachillerato, se los dedico para que nunca se rindan y obtengan un logro académico similar o superior.

Y finalmente se lo dedico a mi novia, Joselyn Avilez Guerrero, eres parte importante de este logro.

SARABIA LUA, GINNIO ANDRES

AGRADECIMIENTO

Mi más sincero agradecimiento a la Universidad Católica de Santiago de Guayaquil, por ser la institución de nivel superior que me acogió para realizar el estudio de la carrera que tanto me gusta y llegar a la ansiada meta de conseguir el título de Ingeniero en Sistemas Computacionales, luego de un largo camino que recorrí junto a docentes y estudiantes, luchando por alcanzar su ideal.

También, mi agradecimiento está dedicado a todas las personas que son parte fundamental de la Facultad de Ingeniería. Docentes, directivos y personal administrativo, que conocieron paso a paso mi trajinar por sus instalaciones y a quienes realicé preguntas, pedí consejos en todas las dudas que daban vueltas en mi mente.

Por último, gracias a todos quienes que de forma directa o indirecta forman parte de la Facultad y que supieron comprender lo que significa la vida de un estudiante universitario.

GUANANGA BERNABE, JAIRO ALEJANDRO

DEDICATORIA

Este trabajo se lo dedico a Dios por permitirme llegar a este momento tan especial en mi vida y por ayudarme a levantar en los momentos difíciles de mi vida.

A mi madre por ser una persona que me ha alentado a continuar con mi camino para llegar a ser un profesional.

A mi padre por apoyarme moralmente y saber guiarme para poder culminar mi carrera.

A mi compañero de tesis porque formamos un buen equipo y a pesar de todo logramos llegar hasta el final del camino.

A mis profesores por la sabiduría que me transmitieron en el desarrollo de mi etapa profesional.

GUANANGA BERNABE, JAIRO ALEJANDRO



**UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL
FACULTAD DE INGENIERÍA
CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES**

TRIBUNAL DE SUSTENTACIÓN

Ing. Cesar Adriano, Salazar Tovar, Mgs
PROFESOR TUTOR

Ing. Beatriz Del Pilar, Guerrero Yépez, Mgs
DIRECTORA DE CARRERA

Ing. Colón Mario, Celleri Mujica, Mgs
COORDINADOR DEL ÁREA

Ing. Roberto Eduardo, Sánchez Calle, Mgs
OPONENTE



**UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL
FACULTAD DE INGENIERÍA
CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES**

CALIFICACIÓN:

Ing. Cesar Adriano, Salazar Tovar, Mgs
PROFESOR TUTOR

Ing. Beatriz Del Pilar, Guerrero Yépez, Mgs
DIRECTORA DE CARRERA

Ing. Colón Mario, Celleri Mujica, Mgs
COORDINADOR DEL ÁREA

Ing. Roberto Eduardo, Sánchez Calle, Mgs
OPONENTE

ÍNDICE GENERAL

RESUMEN	XVIII
ABSTRACT	XIX
INTRODUCCIÓN	20
CAPITULO 1	21
Marco Referencial.....	21
1.1 Fundamentación conceptual	21
1.1.1 Formulación del problema	21
1.1.2 Justificación	22
1.1.3 Delimitación del tema	22
1.1.4 Objetivo General.....	23
1.1.5 Objetivos Específicos	23
1.2 Marco teórico	23
1.2.1 Motor de videojuego	23
1.2.1.1 Unity.....	25
1.2.1.2 Unreal Engine	32
1.2.1.3 CryEngine	36
1.2.2 Simulación	40
1.2.2.1 Simulador.....	41
1.2.2.2 Videojuegos de simulación	42
1.2.3 Modelado 3D	43
1.3 Marco legal.....	44
CAPITULO 2.....	49
Metodología	49
2.1 Tipo de investigación	49
2.1.1 Metodología del ciclo de vida.....	51
2.1.1.1 Fase de Análisis.....	51
2.1.1.2 Fase de Diseño	51
2.1.1.3 Fase de Desarrollo.....	51
2.1.1.4 Fase de Pruebas.....	52

2.2	Diseño de la investigación	52
2.2.1	Población y muestra	52
2.2.2	Técnicas e instrumentos para obtención de información.....	53
2.2.2.1	Entrevista	53
2.2.2.2	Observación	55
2.3	Análisis de la información obtenida.....	57
2.3.1	Entrevista.....	57
2.3.1.1	Requerimientos funcionales.....	57
2.3.1.1.1	Lenguaje de programación.....	57
2.3.1.1.2	Curva de aprendizaje	58
2.3.1.1.3	Recursos de hardware	58
2.3.1.2	Requerimientos no funcionales.....	58
2.3.1.2.1	Aceptación de videojuegos.....	58
2.3.1.2.2	Simulador multiplataforma.....	58
2.3.1.3	Conclusión final de la entrevista	59
2.3.2	Observación	60
2.3.2.1	Requerimientos funcionales.....	60
2.3.2.1.1	Tipo de señalética	60
2.3.2.1.2	Vehículo	60
2.3.2.1.3	Velocidad.....	60
2.3.2.2	Requerimientos no funcionales.....	61
2.3.2.2.2	Tipo de calzada	61
2.3.2.2.3	Obstáculos	61
2.3.2.3	Conclusión final de la observación.....	61
CAPITULO 3.....		61
3.1	Instalación y configuración del entorno de desarrollo	62
3.1.1	Software	62
3.1.1.1	Requisitos de software de Unity.....	63
3.1.1.2	Fuentes de información de Unity.....	63
3.1.2	Hardware	64
3.1.3	Instalación de Unity 5.3.5	64
3.1.4	Creando un proyecto nuevo	68

3.1.5	Interfaz.....	69
3.2	Implementación del simulador del conducción vehicular	70
3.2.1	Grafico	70
3.2.1.1	Iluminación.....	70
3.2.1.2	Motor del terreno.....	70
3.2.1.2.1	Terreno.....	70
3.2.1.2.2	Zonas de viento	71
3.2.1.3	Cámaras	71
3.2.1.4	Materiales y texturas.....	72
3.2.2	Modelado 3D	73
3.2.2.1	Interfaz.....	73
3.2.2.2	Modelado del vehículo	75
3.2.2.3	Modelado objetos de transito	76
3.2.3	Física	80
3.2.3.1	Rigidbodyes	80
3.2.3.2	Colliders	81
3.2.4	Scripting.....	83
3.2.4.1	Programación del vehículo.....	83
3.2.4.1.1	ControlUsuarioVehiculo	84
3.2.4.1.2	ControladorVehiculo	85
3.2.4.1.3	Función Start	86
3.2.4.1.4	Función Update	86
3.2.4.1.5	Función CambioDeMarcha.....	88
3.2.4.1.6	Función Tacómetro.....	89
3.2.4.1.7	Función mover.....	90
3.2.4.1.8	Función TipoVelocidad	92
3.2.4.1.9	Función OnGui	93
3.2.4.2	Programación señales de transito.....	93
3.2.4.2.1	Señal de transito Semáforo	94
3.2.4.2.2	Función OnTriggerEnter	95
3.2.4.2.3	Función Update	95
3.2.4.2.4	Función OnGui	96
3.2.4.2.5	Función OnTriggerExit.....	96

3.2.4.2.6	Script Control.....	97
3.2.5	Hardware.....	97

ÍNDICE DE TABLAS

Tabla 1:	Gestores esenciales y no esenciales.....	24
Tabla 2:	Requerimientos del sistema para Unity.....	28
Tabla 3:	Diferencia de licencias en Unity.....	28
Tabla 4:	Juegos destacados creados con Unity.....	29
Tabla 5:	Requisitos del sistema para Unreal Engine.....	33
Tabla 6:	Juegos destacados realizados con Unreal Engine.....	34
Tabla 7:	Socios de Unreal Engine.....	35
Tabla 8:	Requisitos del sistema CryEngine.....	37
Tabla 9:	Juegos destacados desarrollado en CryEngine.....	38
Tabla 10:	Socios de CryEngine.....	39
Tabla 11:	Docentes que serán entrevistados.....	53
Tabla 12:	Lenguajes de programación utilizados en las materias de programación.....	57
Tabla 13:	Comparación de los motores de videojuegos.....	59
Tabla 14:	Software utilizado en el desarrollo del simulador.....	62
Tabla 15:	Recursos de Unity.....	63
Tabla 16:	Hardware utilizado en el desarrollo del simulador.....	64

ÍNDICE DE GRÁFICOS

Gráfico 1: Desarrolladores de Unity registrados 2012 - 2015	27
Gráfico 2: Socios de Unity	31
Gráfico 3: Objetivos de la simulación.....	41
Gráfico 4: Ejemplo de modelado 3D	43
Gráfico 5: Paradigmas de investigación.....	50
Gráfico 6: MODELO DE ENTREVISTA PARA DOCENTES.....	54
Gráfico 7: MODELO DE FICHA DE OBSERVACIÓN	55
Gráfico 8: Página oficial de Unity	65
Gráfico 9: Página oficial de Unity	65
Gráfico 10: Versiones de Unity	66
Gráfico 11: Creación de cuenta con el software de Unity	67
Gráfico 12: Selección de versión de Unity	68
Gráfico 13: Creación de nuevo proyecto en Unity.....	68
Gráfico 14: Interfaz del motor de videojuegos	69
Gráfico 15: DirectionalLights (Luz direccionada).....	70
Gráfico 16: Propiedades del terreno	71
Gráfico 17: Propiedades del viento	71
Gráfico 18: Fotografía de señal de límite de velocidad	72
Gráfico 19: Propiedades del material.....	72
Gráfico 20: Interfaz de 3D Max	73
Gráfico 21: Icono para mover un objeto.....	74
Gráfico 22: Icono para rotar un objeto	74
Gráfico 23: Icono del panel de creación.....	75
Gráfico 24: Icono del panel de creación.....	75
Gráfico 25: Modelo del vehiculo en 3D Max	75
Gráfico 26: Modelo 3D del vehículo	76
Gráfico 27: Modelo de base de señal de transito.....	77
Gráfico 28: Panel de modificación de objeto.....	77
Gráfico 29: Configuración Chamfer.....	78
Gráfico 30: Modelo de cuerpo de la señal de transito.....	78
Gráfico 31: Función CompoundObjects	79
Gráfico 32: Modelo de señal de tránsito	79
Gráfico 33: Modelo 3D de señales de transito en Unity	80
Gráfico 34: Componente de rigidbody	81
Gráfico 35: Propiedades del Box Collider	82
Gráfico 36: Propiedades de Wheel Collider	83
Gráfico 37: Plano cartesiano.....	84
Gráfico 38 Script ControladorVehiculo.....	85
Gráfico 39 Script función Start	86
Gráfico 40 Script función Update	87
Gráfico 41 Script para direccionales del vehiculo	88
Gráfico 42 Función cambio de marcha	89

Gráfico 43 Script tacómetro	90
Gráfico 44: Código fuente para el movimiento del vehículo.....	91
Gráfico 45: Código fuente para formatear la velocidad del vehículo.....	93
Gráfico 46: Asignación de tag “Player” al objeto Auto.....	93
Gráfico 47 Señal de transito semáforo.....	94
Gráfico 48: Variables declaradas para el semáforo	94
Gráfico 49: Código de la función OnTriggerEnter	95
Gráfico 50: Código de la función Update	95
Gráfico 51: Código del envío de mensajes por pantalla.....	96
Gráfico 52: Código de la función OnTriggerExit.....	96
Gráfico 53: Propiedades del eje horizontal	97
Gráfico 54: Propiedades del eje vertical	98
Gráfico 55: Pantalla de configuración de JoyStick	98

RESUMEN

La presente tesis tiene como objetivo el diseño e implementación de un simulador de conducción vehicular utilizando un motor de videojuegos. El tema fue propuesto debido a que en la facultad de ingeniería, carrera de Ingeniería en sistemas computacionales no existen hasta el momento proyectos de esta índole. El proceso de elaborar videojuegos requiere diversos conocimientos como diseño gráficos (Elaboración de texturas y modelos 3D) y programación, los motores de videojuegos son herramientas de programación utilizadas para la elaboración de proyectos de videojuegos. Con el desarrollo propuesto se espera que los docentes de la carrera sistemas computacionales del área de programación, utilicen el simulador como base para trabajos de tutoría.

El simulador de conducción vehicular cuenta con señales de tránsito las cuales son validadas y penalizar al usuario en caso de cometer infracciones según las leyes ecuatorianas.

PALABRAS CLAVES: SIMULADOR; MOTOR DE VIDEO JUEGOS; VIDEOJUEGO; SOFTWARE

ABSTRACT

This thesis aims to design and implement a vehicle driving simulator using a game engine. The theme was proposed because in the Faculty of Engineering, Engineering in computer systems do not exist so far such projects. The process of developing video games requires different skills such as graphic design (Development of textures and 3D models) and programming, game engines are programming tools used for the development of game projects. The proposed development is expected to teachers of computer systems programming career area, use the simulator as a basis for tutoring work.

The driving simulator has vehicular traffic signs which are validated and penalize the user in case of infringement by Ecuadorian laws.

KEYWORDS: SIMULATOR; GAME ENGINE; GAME; SOFTWARE

INTRODUCCIÓN

En la carrera de sistemas computacionales de la Universidad Católica de Santiago de Guayaquil no se ha indagado en profundidad en los temas de elaboración de videojuegos.

El siguiente proyecto tecnológico va enfocado hacia los docentes del área de programación de la carrera de sistemas computacionales en el cual se busca implementar un simulador de conducción vehicular utilizando un motor de videojuegos, los docentes podrán utilizar este proyecto como base y punto de referencia para enviar proyectos de tutoría a cerca de videojuegos a sus estudiantes.

Se pretende aportar a la carrera con bases para futuros proyectos de videojuegos, es decir que los estudiantes que deseen incursionar en temas de videojuegos no deban empezar desde cero para elaborar un proyecto.

Se elaboró una investigación descriptiva utilizando como técnicas de investigación la entrevista a docentes del área de programación y la observación de parques viales para determinar los requerimientos funcionales y no funcionales del simulador como del motor de videojuegos a utilizar.

CAPITULO 1

Marco Referencial

1.1 Fundamentación conceptual:

1.1.1 Formulación del problema

El desarrollo de un simulador vehicular utilizando un motor de videojuegos ayudará a los estudiantes de la Carrera de Ingeniería en Sistemas de la Universidad Católica de Santiago de Guayaquil a desarrollar habilidades básicas para construir un videojuego.

También quienes deseen utilizar como base el simulador para trabajos de tutoría o investigaciones, con el fin que los estudiantes puedan practicar los conocimientos impartidos en clases.

Las materias que podrían utilizar el simulador como base para proyectos o tutorías son:

- Fundamentos de programación
- Programación orientada a objetos
- Programación en capas
- Simulación

El hecho que la Carrera cuente con un simulador vehicular utilizando un motor de videojuegos también permitirá a los estudiantes aprender temas nuevos tales como:

- Modelado y animación 3D
- Edición de Audio

Lo cual abre la posibilidad de realizar un proyecto o tesis de manera conjunta con otras carreras como:

- Ingeniería en producción y dirección de artes multimedia
- Ingeniería en producción y dirección de artes audiovisuales

1.1.2 Justificación

Se decidió implementar una simulación de conducción vehicular debido a que es un elemento común en la vida de las personas, las cuales han interactuado con un vehículo tanto en la faceta de conductor como de peatón.

Se decidió simular un vehículo debido a la variedad de temas de Física que abarca tales como:

- Movimiento rectilíneo uniforme
- Movimiento rectilíneo uniformemente acelerado
- Movimiento curvilíneo (o circular) uniforme
- Movimiento curvilíneo (o circular) uniformemente acelerado.

Además de incluir dentro de la simulación señales de tránsito para que el usuario final conozca las infracciones y las penalizaciones por el cometimiento de las mismas, esto es debido a la reciente reforma a la Ley Orgánica de Transporte Terrestre y Seguridad Vial realizada en el año 2014.

1.1.3 Delimitación del tema

El desarrollo del proyecto cubrirá los siguientes puntos:

- El simulador constará con pedales de aceleración y frenado como dispositivos de hardware.
- El auto será desarrollado con transmisión automática.
- El escenario de la simulación estará constituido por 4 cuadras de sur a norte y 3 cuadras de este a oeste.
- El escenario contendrá 5 señales de tránsito las cuales las mismas que serán verificadas en todo momento durante la simulación.

- El simulador constará con un modo en el cual el usuario se podrá movilizar de manera libre por todo el escenario.
- Al final de la simulación se emitirá un resumen de las infracciones cometidas, el valor monetario en multas y la cantidad de puntos reducidos en la licencia de conducir.

1.1.4 Objetivo General

- Diseñar e implementar un simulador de conducción en un motor de videojuegos con un escenario controlado para los estudiantes de la carrera de sistemas en la facultad de ingeniería.

1.1.5 Objetivos Específicos

- Identificar los requerimientos funcionales y no funcionales para el desarrollo del simulador.
- Seleccionar la plataforma de hardware y software a utilizar.
- Implementar el simulador de conducción vehicular.

1.2 Marco teórico:

1.2.1 Motor de videojuego

El término motor de videojuegos, según lo que manifiesta Díaz (2014) “hace referencia a una serie de rutinas de programación que permiten el diseño, la creación, el desarrollo y la representación gráfica de un videojuego”.

Un motor de videojuego esta dividió por diferentes componentes llamados gestores. Cada gestor es responsable de brindar diferentes características esenciales para la elaboración de un videojuego. Según Thorn(2011) “los gestores se clasifican en dos grupos: esenciales y los no esenciales”. Los esenciales son los gestores mínimos que debe tener un motor de videojuego, los no esenciales son características que ayuden a que el motor tenga un mejor rendimiento.

Tabla 1: Gestores esenciales y no esenciales

Gestores esenciales	Gestores no esenciales
Gestor de renderizado	Gestor de audio
Gestor de recursos	Gestor de física
Gestor de escena	Gestor de script
Gestor de entrada	
Gestor de errores	

Fuente: Alan Thorn(2011)

Elaborado por: los autores

Según Alan Thorn(2011) los gestores principales son:

- Gestor de recursos: También llamado “assets”, en la industria de videojuegos se refiere a todo el material digital que se utiliza en los videojuegos. El material digital se clasifica en dos tipos: Datos multimedia y datos de comportamiento. Los datos multimedia se refieren a sonidos, imágenes, animaciones, objetos 3D entre otros. Los datos de comportamiento son los climas, tamaño de resolución, que se maneja del videojuego.
- Gestor de renderizado: Permite importar a la escena objetos 3D o 2D, procesando datos como la luz y textura de los objetos para obtener un acabo realista.
- Gestor de entrada (Input): Permite interactuar con dispositivos de entrada, crea una comunicación entre el periférico y el videojuego. Los videojuegos reciben datos de periféricos de entrada vía teclado, mouse, joystick.
- Gestor de audio: Es el responsable de la comunicación del audio dentro del juego con el hardware de audio. Permite manipular los niveles de audio, agregar efectos de sonido.
- Gestor de errores: En el momento de la depuración ayuda a visualizar los problemas que tengan los scripts o la mala configuración de un objeto dentro del videojuego.

- Gestor de escena: El propósito de la escena es de sintetizar y coordinar los recursos del motor de acuerdo a la lógica del videojuego, donde trabaja en conjunto los diferentes gestores que tiene el motor. La tarea principal del gestor de escena es crear la comunicación entre los diferentes gestores como: gestor de física, renderización, scripts.
- Gestor de física: Es un componente del motor dedicado aplicar leyes de física a los objetos del videojuego, como por ejemplo aplicar fuerzas, torque, gravedad a los objetos de la escena.
- Gestor de script: Permite usar un lenguaje como C++ o C# para construir el motor de videojuego en una máquina de código con el que se pueden manipular objetos de la escena, configuración del videojuego, cámaras.

En el mercado existen diversas compañías en el desarrollo de motores de videojuegos, las más destacadas según el sitio web GamemediAx(2015) son:

- Unreal Engine 4
- Unity 5
- CryEngine
- Source Engine
- Blender Game Engine
- Leadwerks Engine
- Esenthel Engine
- S2 Engine
- NeoAxis
- GameGuru

1.2.1.1 Unity

Historia

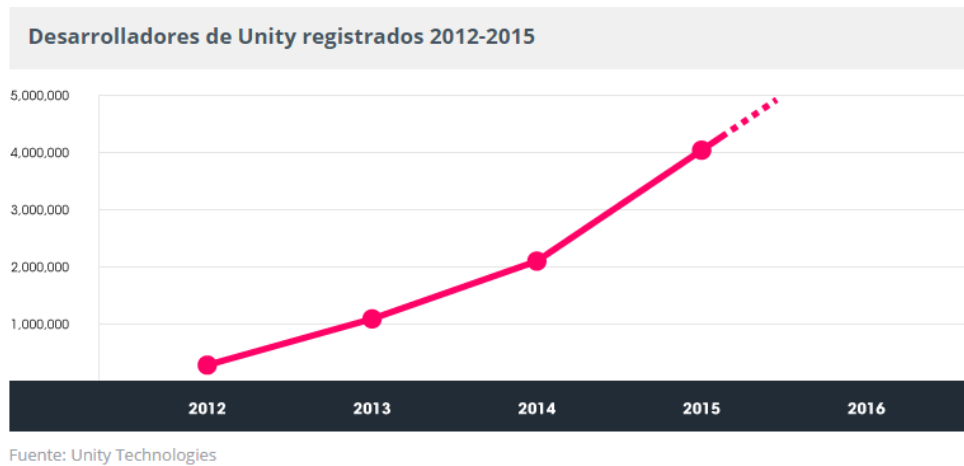
Los inicios de Unity Technologies según lo manifiesta Díaz (2014) “fue fundada en 2004 por David Helgason, Nicholas Francis y Joachim Ante con la idea de democratizar el desarrollo de juegos “. En sus primeras versiones solo era compatible con Mac Os y estaba disponible en dos versiones Indie y Profesional.

La versión Indie fue creada especialmente para los desarrolladores independientes y pequeñas estudios que estaban dando sus primeros pasos en desarrollo de video juegos. El costo de la licencia estaba a un precio al público de \$300 para la versión Indie y la versión Profesional con un precio de \$1500. En el año 2009 UnityTechnologies decidió que Indie desapareciera y se haría totalmente gratuita.

Características

- Unity tiene un editor en el cual se podrá observar, manipular de manera fácil las características principales de lo que se maneje en la escena. Cuenta con un motor de física llamado PhysX 3.3 cuenta con muchas mejoras con el rendimiento de física, nuevos elementos como wheel colliders que dan más realismo en la simulación de un vehículo ya que ofrece mejora para lo que es la suspensión.
- Audio Mixer mejora de una manera considerable el audio dentro del juego, ofreciendo detalles como flujo de trabajos, capturando mezclas de audio en diferentes lugares del proyecto.
- Motor de iluminación Real-time Global Illumination, con más opciones de iluminación y soportado para dispositivos móviles y escritorio.
- Motor de animación cuenta con State Machine Behaviours que permite tener mayor precisión en los scripts que manejen animaciones.
- Según Díaz (2014) Unity está cuenta con una “comunidad de usuarios muy grande, lo que permite encontrar multitud de tutoriales, ayuda, solución de problemas”.

Gráfico 1: Desarrolladores de Unity registrados 2012 - 2015



Fuente: Sitio web de Unity (2016)

Elaborado por: Unity

- Los scripts se los puede programar en dos lenguajes de programación uno es c# el otro java.
- Documentación posee ejemplos, explicación del código concretos de diferentes temas con el fin de los desarrolladores entiendan mejor la herramienta. Además que cuenta con proyectos completos gratuitos con el que puede apreciar la aplicación de los scripts, modelos 3D, iluminación, física.
- Permite la importación de objetos 3D de diferente software de modelado 3D como 3D Max, Blender, objetos que tengan extensión .obj, .fbx.
- Cuenta con una tienda llamada Assets Store en la que se encuentra proyectos, objetos, audio, texturas, etc. algunas de forma gratuita otras pagadas.
- Permite utilizar el mismo código para ejecutarlo en diferentes plataformas como: iOS, Android, Windows phone, Tizen, Windows, Mac, Linux/Steam, WebGL, PlayStation 3, PlayStation 4, PlayStation Vita, Xbox 360, Xbox One, Wii, Android TV, Samsung SMART, Oculus Rift, Gear Vr, PlayStation VR.

Tabla 2: Requerimientos del sistema para Unity

Requisito de	Recomendado
Sistema operativo	Windows 7 SP1+, 8, 10; Mac OS X 10.8+.
Procesador	Compatible con el conjunto de instrucciones SSE2.
Video	tarjeta de vídeo con DX9 (modelo de shader 2.0)
Desarrollo	Visual Studio 2013 Update 2+.

Fuente: Sitio web de Unity (2016)

Elaborado por: los autores

Licencias

Para desarrolladores Indies y Estudios. La versión Personal Edition es totalmente gratuita en cambio la Professional Edition tiene un costo de \$75 mensuales.

Tabla 3: Diferencia de licencias en Unity

Características	Personal Edition	Professional Edition
Motor con todas las prestaciones	SI	SI
Sin regalía	SI	SI
Todas las plataformas	SI	SI

Acceso beta	SI	SI
Pantalla de inicio personalizable	NO	SI
Unity Cloud Build	NO	SI
Unity Analytics Pro	NO	SI
Team License	NO	SI
Prioridad en el reporte de bugs	NO	SI
Game Performance Reporting	NO	SI

Elaborado por: los autores

Fuente: Sitio web de Unity(2016)

Oficinas

La sede principal es en San Francisco, California, en Estados Unidos. También cuenta con oficinas en Reino unido, Ucrania, Suecia, Singapur. Lituana, Corea, Alemania, Finlandia, Dinamarca, Colombia, China, Canadá.

Juegos destacados

Unity cuenta con una diversidad de juegos desarrollados en su plataforma. VER TABLA 4.

Tabla 4: Juegos destacados creados con Unity

Titulo	Desarrollador	Genero	Plataforma
GALAK-Z	17-bit	Arcade, Shooter	PS4,Windows
SUPER DUNGEON BROS	React Games	Action, RPG	Linux, Mac, Windows
KERBAL SPACE	Squad	Simulación	Linux, Mac,

PROGRAM			Windows
CITIES SKYLINES	Colossal Order	Simulación, Estrategia	Linux, Mac, Windows
CUPHEAD	StudioMHR	Platformer, Shooter	Linux, Mac, Windows, XboX 360
LARA CROFT:RELIC RUN	Simutronics	Action, Endless Runner	Android, IOS, Windows Phone
REPUBLIQUE	Cmouflaj	Action, Stealth	Mac, Windows
MEVIUS FINAL FANTASY	Square Enix	RPG	Android, IOS
UNKILLED	Madfinger Games	Action, Shooter	Android, IOS

Fuente: Sitio web de Unity(2016)

Elaborado por: los autores

Educación

Unity cuenta con certificaciones para desarrolladores las cuales se pueden realizar en los países autorizados, existen 3 tipos de certificaciones:

- Professional Programmer.
- Professional Artist.
- Expert.

Además brinda un sistema de licencias educativas para educadores como para instituciones. Para las instituciones que deseen agregar Unity como plan de estudio obtendrán descuentos en sus licencias.







Socios

Sus socios son:

Gráfico 2: Socios de Unity

Microsoft



Sony	
Qualcomm	
Samsung	
Nintendo	
Oculus	
Intel	

Fuente: Sitio web de Unity(2016)

Elaborado por: los autores

1.2.1.2 UnrealEngine

Historia

Creado por EpicGames en el año de 1998, La primera versión presento las siguientes características: Renderizado, motor de física (colisiones), red. Su creación dio cavidad a la famosa serie de Unreal, que soporta plataformas como Windows, Mac, Linux. Con la segunda versión de UnrealEngine se creó Unrealtournament la continuación de la serie Unreal, el cual presento característica más realista en sus diseños.

Características

UnrealEngine 4 posee las siguientes características en su motor gráfico:

- Soporte avanzados para DirectX 11 y 12 brinda un avanzado sistema de luces y sombras maximizando la calidad visual de las imágenes y un sistema de manipular luces dinámicamente entre escenas.
- Texturas casi reales que brindan mayor detalle en objetos, para crear cabello más realista, prendas de vestir.
- Matinee Cinematics brinda al programador beneficios como director de película con facilidades de manejar escenas tramo a tramos, maniobrar las cámaras de manera secuencial, para producir filmes impresionantes.
- Un motor de renderizado que emula un mundo real con material de precisión. Es 100% compactible con hardware de realidad virtual como son los Oculus Rift. Con el código fuente C++ se puede personalizar las herramientas como lo que es las animaciones, renderizado, iluminación, hasta la misma interfaz del usuario. Además que se puede manejar el flujo de los script de los diferentes objetos en tiempo real con una interfaz gráfica en tiempo real y para modificar el código se necesita de visual Studio versiones de 2013 en adelante.
- La herramienta de animación brinda facilidades al momento de darle movimiento a un personaje ya que permite editar el esqueleto de las personas, mallas del esqueleto, teniendo una vista previa de la animación en tiempo real, adicional se puede modificar la física de los objetos con el editor de física.

- Soporte para las siguientes plataformas: Windows, Linux, Mac, Android, IOS, Steam, Oculus, PlayStation 4, Xbox One.
- Tiene integrado tecnologías de compañías líderes en el mercado como: NVIDIA PhysX, Autodesk Gameware, iluminar, Umbra, Oculus VR.
- Requisitos del sistema son:

Tabla 5: Requisitos del sistema para Unreal Engine

Requisito de	Recomendado
Sistema operativo	Windows 7/8 64-bit
Procesador	Quad-core Intel or AMD, 2.5 GHz or faster
Memoria	8 GB RAM
Video	NVIDIA GeForce 470 GTX o AMD Radeon 6870 HD
DirectX	Version 11
Disco duro	8 GB disponible
Desarrollo	Visual Studio 2015 Professional o Visual Studio 2015 Community

Elaborado por: los autores

Fuente: Sitio web de Unreal (2016)

Licencia

El motor es totalmente gratuito pero contiene las siguientes políticas de royalty: El producto creado supera un ingreso mayor a \$3.000 por trimestre EpicGames

obtendrá el 5% de las ganancias. No paga royalty proyecto que sean de tipo cinematográfico.

Oficinas

La sede principal se encuentra en carolina del norte. También tiene oficinas en Utah, Washington, Inglaterra, Japón, Corea del Sur y China.

Juegos destacados

Los juegos que más se destacan son los siguientes utilizando el motor gráfico de UnrealEngine.

Tabla 6: Juegos destacados realizados con Unreal Engine

AÑO	TITULO	COMPAÑÍA
1998	Unreal	Epic Games
1999	Unreal Tournament	Epic Games
2004	Spider-Man 2	Fizz Factor
2005	Brothers In Arms: Road to Hill 30	Gearbox Software
2006	Tom Clancy's Ghost Recon 2	Ubisoft
2006	Gears of War	Epic Games
2007	BioShock	2K Boston/2K Australia








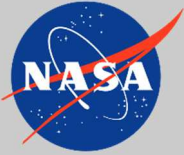


2009	Batman: Arkham Asylum	Rocksteady Studios
2015	Mortal Kombat X	NetherRealm Studios







Elaborado por: los autores
Fuente: Sitio web de Unreal(2016)

Socios

Los socios que tiene UnrealEngine son:

Tabla 7: Socios de Unreal Engine

Warner Bros.		Sega	
Ubisoft		Sony	
Oculus		Nintendo	
Nvidia		NASA	
Intel		ActiVision	

Apple		Microsoft	
Konami		Gameloft	
Google		Capcom	

Elaborado por: los autores

Fuente: Sitio web de Unreal(2016)

1.2.1.3 CryEngine

Historia

Desarrollado por Crytek, fue Fundada por los hermanos Avni, Cevat y FarukYerli en 1999.Su primero producto desarrollado completamente en el motor de CryEngine 1 fue FarCry.

En 2006 la empresa de videojuegos Ubisoft adquirió todos los derechos de CryEngine. Posee un motor gráfico para elaborar escenas con un alto poder en calidad de imágenes y textura permitiendo crear escenas ambientales muy parecidas a la realidad, los script se los realiza en el lenguaje de programación C++

Características

En 2016 se presentó CryEngine 4 el cual contempla las siguientes características:

- DirectX 12 el nuevo Api de Microsoft para obtener mayor rendimiento visual.
Un motor de renderizado capaz de crear superficies realistas con un potente

sistema de materiales e iluminación, simula el comportamiento de la luz con el mundo real obteniendo efectos ambientales reales.

- Permite simular el comportamiento del agua al punto de interactuar con objetos que colisionan con ella. Brinda la opción de refinamiento de la malla de los objetos haciendo que los polígonos obtengan un mejor detalle y una malla de aspecto realista. Se puede editar en tiempo real permitiendo al artista modificar sus diseños.
- Motor de Anti-Aliasing, mediante múltiples técnicas de diseño permite reducir el rastro de pixeles produciendo imágenes más estables. Desenfoque al momento de moverse por la escena y la profundidad de la escena ya no es un problema, se implementa una calidad alta para este efecto a un bajo costo de recursos del hardware.
- Ahora cuenta con un framework para poder desarrollar los script en el lenguaje de programación C#.
- Utiliza técnicas para simular una vegetación realista la cual puede interactuar con cualquier objeto o factores ambientales como: lluvia y viento. Nuevo sistema de partículas el cual permite crear un sistema de partículas con un alto rendimiento y calidad a un bajo costo de la GPU.
- Soporte para las siguientes plataformas: Xbox One, Xbox 360, PlayStation 3, PlayStation 4, Wii U.
- Requisitos del sistema:

Tabla 8: Requisitos del sistema CryEngine

Requisito de	Mínimo	Recomendado
Sistema operativo	Windows Vista SP1, Windows 7, 8.1, 10 (64-bit)	Windows 7, 8.1, 10 (64-bit)
Procesador	Intel Dual-Core 2GHz or AMD Dual-Core 2GHz	Intel Quad-Core (i5 2300) or AMD Octo-Core (FX 8150)
Memoria	4 GB RAM disponible	8 GB RAM disponible
Video	NVIDIA GeForce 400 series or AMD Radeon HD 6000 series	NVIDIA GeForce 660Ti or higher, AMD Radeon HD

		7950 or higher
DirectX	Version 11	Version 11
Disco duro	8 GB	8 GB
Tarjeta de sonido	Tarjeta de sonido compactible con DirectX	Tarjeta de sonido compactible con DirectX

Fuente: Sitio web de CryEngine (2016)

Elaborado por: los autores

Licencia

La licencia es totalmente gratuita libre de royalty, adicional el usuario tiene la opción de cancelar el valor que el desee por el servicio de CryEngine, Del 100% del monto el 70% será donado para un fondo de desarrollo independiente y el otro 30% para Crytek, también posee un paquete de membrecías con valores de \$50 y \$150 que ofrecen servicios como:

- Seminarios web regulares por expertos CryEngine
- Servicios de consultoría
- Descuentos en los paquetes de soporte

Oficinas

Sede principal es en Alemania y cuenta con oficinas en Ucrania, Hungría, Bulgaria, Corea del Sur, China, Turquía.

Juegos destacados

Tabla 9: Juegos destacados desarrollado en CryEngine

Titulo	Motor	Plataforma	Año	Publicador
Far Cry	CryEngine 1	Windows	2004	Ubisoft
Crysis	CryEngine 2	Windows	2007	Electronic Arts
Crysis Warhead	CryEngine 2	Windows	2008	Electronic Arts

Fuente: Sitio web de CryEngine(2016)

Elaborado por: los autores

Socios

Los socios son:

Tabla 10: Socios de CryEngine

Socios financieros	Publicadores de juegos	Socios de tecnología	Socios de desarrollo	Asociaciones
				
				



Fuente: Sitio web de CryEngine(2016)

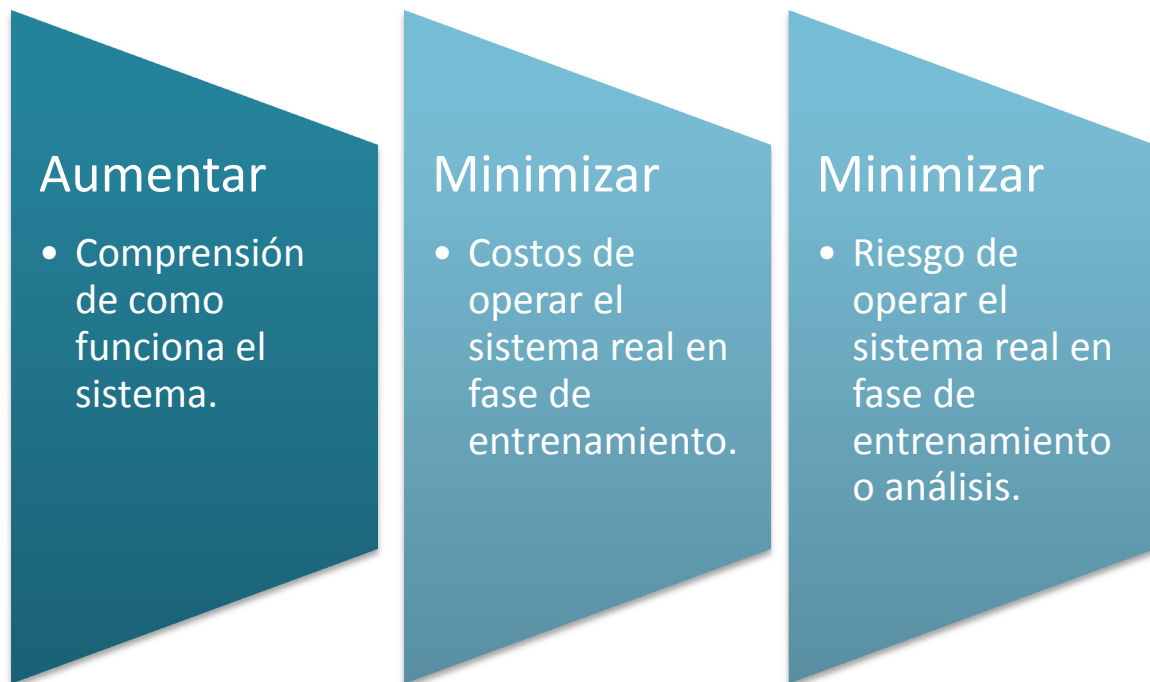
Elaborado por: los autores

1.2.2 Simulación

Se puede definir a la simulación como el proceso de imitar ciertos aspectos de la realidad en un entorno virtual, lo cual permite analizar diversos escenarios a través de variables controladas definidas por el encargado de diseñar la simulación.

Una de los principales usos de la simulación es comprobar que acciones ejecutan los actores (personas, sistemas, bolsa de valores) ante diversos escenarios que intenta apegarse a la realidad, por tal motivo uno de los principales uso de las primeras simulaciones el año 1910 fue el desarrollo de simuladores de vuelo, el hecho de practicar directamente en una aeronave sin previa practica es una actividad de alto riesgo, el simulador de vuelo permite obtener la experiencia necesaria en un ambiente controlado y sin riesgos.

Gráfico 3: Objetivos de la simulación



Elaborado por: los autores

Fuente: María González (2013)

1.2.2.1 Simulador

Según el Diccionario de la Real Academia (2014), en su segunda acepción, simulador es definido como: “Aparato que reproduce el comportamiento de un sistema en determinadas condiciones, aplicado generalmente para el entrenamiento de quienes deben manejar dicho sistema.”

González (2014) por su parte define como simulador a: “Reproducciones muy sofisticadas de aparatos o actividades complejas como, por ejemplo, los simuladores de vuelo, de conducción de vehículos o de realización de deportes concretos.”

Por ende se puede definir a un simulador como una herramienta que permite reproducir un evento asemejándose a la realidad, el mismo está compuesto por un

software específico y puede utilizar en algunos casos hardware específico para mejorar el desarrollo de la simulación como periféricos de control (Volante, Pedales, etc.) así como componentes de ambiente (iluminación, temperatura, efectos de viento, vibración, etc.)

1.2.2.2 Videojuegos de simulación

Los videojuegos de simulación según define Soledad González (2014) en su tesis de pregrado, “son un tipo de videojuegos que intentan recrear situaciones de la vida real.” Los videojuegos de simulación intentan “reproducir tanto las sensaciones físicas como velocidad, aceleración, percepción del entorno y una de sus funciones es dar una experiencia real de algo que no está sucediendo”.

Otra definición por parte de YanegaVenega(2016) indica que: “Son juegos que se emplean para adquirir o ejercitar distintas habilidades o para enseñar comportamientos eficaces en el contexto de situaciones o condiciones simuladas”.

Por lo anterior se puede definir como videojuegos de simulación a aquellos en los que el jugador se sumerge en un mundo virtual el cual simula aspectos de la vida real, los videojuegos de simulación son usados generalmente para aprender destrezas o habilidades en un ambiente controlado y eliminado el riesgo de sufrir siniestros por falta de habilidades o experiencia, como por ejemplo un simulador de vuelo.

Se clasifican según Soledad González (2014) de la siguiente manera:

- **Simuladores instrumentales:** El jugador conduce un vehículo (automóvil, moto, avión, etc.) por un escenario lleno de obstáculos los mismos que deben ser superados y llegar a una meta u objetivo, fueron desarrollados en un inicio para temas militares como son los simuladores de vuelo.
- **Simuladores constructores:** Se ambienta en un tema específico (Medicina, Construcción, etc.), el jugador tiene como papel concreto la creación,

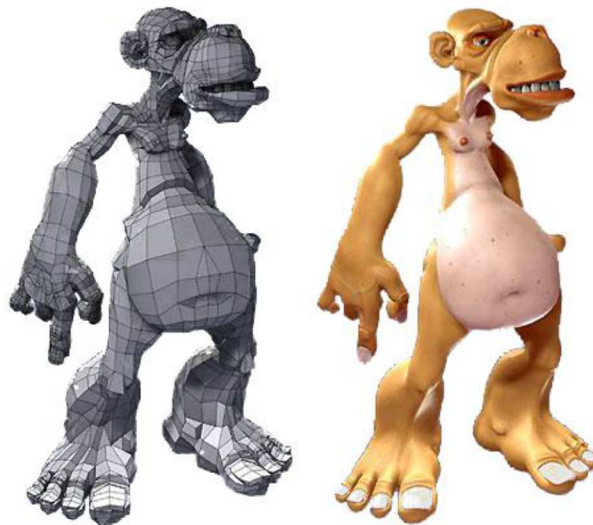
expansión o manejo de proyectos en base a la toma de decisiones y la utilización de recursos para cumplir un objetivo.

- **Simuladores deportivos:** Es un videojuego que simula el desenvolvimiento de un deporte como: futbol, básquet, futbol americano, entre muchos más.

1.2.3 Modelado 3D

El modelo 3D es una representación de coordenadas utilizando 3 ejes (x, y, z), que conforman estructuras conocidas como MESH (Mallas) y que su superficie se encuentra cubierta por una capa de imagen conocida como textura, el producto final de un modelado 3D es una imagen en 3 dimensiones.

Gráfico 4: Ejemplo de modelado 3D



Fuente:(Ramos, Aquino, & Lara, 2013)

Existen aplicaciones conocidas como modeladores, Un modelador es un software especializado en la creación y modificación de un modelo 3D en el mercado existe una gran variedad de software para este fin como: 3DMax, Zbrush, Zmodeler entre otros.

Para la elaboración de un modelo 3D existen diferentes tipos de técnicas para llegar el mismo objetivo como son:

- **Box Modeling:** Consiste en realizar un modelo 3D a través de una forma primitiva (caja, cilindro, etc.) se utiliza para hacer la forma base de un modelo final.
- **NURBS Modeling:** Es una forma de modelar utilizando un modelo matemático, comúnmente usado para la creación de curvas y superficies.
- **Operaciones booleanas:** Consiste en realizar operaciones booleanas en 2 o más estructuras para obtener una estructura o modelo final. Las operaciones que se puede realizar son: Resta, intersección y unión.
- **Extrude | Lathe:** Son técnicas de modelado que realizan modelos utilizando como base imágenes en 2D. Extrude extiende la profundidad de un objeto 2D para convertirlo en un objeto 3D, en cuanto a Lathe toma como base un objeto 2D y lo reproduce sobre un eje, Lathe es ideal para realizar jarrones, tazas, vasos, etc.
- **Loft:** es una técnica de modelado de superficies, esta característica crea una forma haciendo transiciones entre múltiples perfiles y curvas guiadas. (Ramos, Aquino, & Lara, 2013)

1.3 Marco legal

En la Ley Orgánica de Transporte Terrestre, Tránsito y Seguridad Vial desde el artículo 106 al 146 se contemplaba hasta febrero del año 2014 todas las normativas relacionadas a las infracciones de tránsito y las sanciones a las mismas, dichos artículos fueron derogados por Disposición Derogatoria

Décimo Octava de Ley No. 00, publicada en Registro Oficial Suplemento 180 de 10 de Febrero de 2014, y desde entonces pasan a encontrarse en el código orgánico integral penal que desde el artículo 371 al 392 describe las contravenciones de tránsito y sus sanciones las cuales clasifica de la siguiente manera:

Art. 386.- Contravenciones de primera clase: Son sancionadas con pena privativa de libertad de tres días, multa de un salario básico unificado del trabajador en general y reducción de diez puntos en su licencia de conducir.

Ejemplos de contravenciones de primera clase:

- La persona que conduzca sin haber obtenido licencia.
- El conductor que falte de obra a la autoridad o agente de tránsito.
- El conductor que exceda los límites de velocidad fuera del rango moderado.

Art. 387 Contravenciones de segunda clase: Son sancionadas con multa del cincuenta por ciento de un salario básico unificado del trabajador en general y reducción de nueve puntos en el registro de su licencia de conducir.

Ejemplos de contravenciones de segunda clase:

- El conductor que ocasione un accidente de tránsito que resulten danos materiales, cuyos costos sean inferiores a dos salarios básicos unificados del trabajador en general.
- La persona que conduzca con licencia caducada, anulada, revocada o suspendida, la misma que deberá ser retirada inmediatamente por el agente de tránsito.

Art. 388 Contravenciones de Tercera clase: Son sancionadas con multa equivalente al cuarenta por ciento de un salario básico unificado del trabajador en general y reducción de siete punto cinco puntos en su licencia de conducir.

Ejemplos de contravenciones de tercera clase:

- La o el conductor que derrame en la vía pública sustancias o materiales deslizantes, inflamables o contaminantes, salvo caso fortuito o fuerza mayor debidamente comprobados.
- La o el conductor que transporte material inflamable, explosivo o peligroso en vehículos no acondicionados para el efecto o sin el permiso de la autoridad competente y las o los conductores no profesionales que realizaren esta actividad con un vehículo calificado para el efecto.
- La persona que construya o mande a construir reductores de velocidad sobre la calzada de las vías, sin previa autorización o inobservando las disposiciones de los respectivos reglamentos.

Art. 389.- Contravenciones de Cuarta clase: Son sancionadas con multa equivalente al treinta por ciento de un salario básico unificado del trabajador en general, y reducción de seis puntos en su licencia de conducir:

Ejemplos de contravenciones de cuarta clase:

- La o el conductor que desobedezca las órdenes de los agentes de tránsito, o que no respete las señales manuales de dichos agentes, en general toda señalización colocada en las vías públicas, tales como: semáforos, pare, ceda el paso, cruce o preferencia de vías.
- La o el conductor que derrame en la vía pública sustancias o materiales deslizantes, inflamables o contaminantes, salvo caso fortuito o fuerza mayor debidamente comprobados.
- La o el conductor que falte de palabra a la autoridad o agente de tránsito.
- La o el conductor que con un vehículo automotor exceda dentro de un rango moderado los límites de velocidad permitidos, de conformidad con los reglamentos de tránsito correspondientes.

Art. 390.- Contravenciones de Quinta: Son sancionadas con una multa equivalente al quince por ciento de un salario básico unificado del trabajador en general y reducción de cuatro punto cinco puntos en su licencia de conducir.

Ejemplos de contravenciones de quinta clase:

- La o el conductor que conduzca un vehículo en sentido contrario a la vía normal de circulación, siempre que la respectiva señalización esté clara y visible.
- La o el conductor de un vehículo a diésel cuyo tubo de escape no esté instalado de conformidad con los reglamentos de tránsito.
- La o el propietario o conductor de un vehículo automotor que, en caso de emergencia o calamidad pública, luego de ser requeridos, se niegue a prestar la ayuda solicitada.
- La o el conductor de vehículos a motor que, ante las señales de alarma o toque de sirena de un vehículo de emergencia, no deje la vía libre.
- La o el conductor que detenga o estacione un vehículo automotor en lugares no permitidos, para dejar o recoger pasajeros o carga, o por cualquier otro motivo.
- La o el conductor que estacione un vehículo automotor en cualquier tipo de vías, sin tomar las precauciones reglamentariamente previstas para evitar un accidente de tránsito o lo deje abandonado en la vía pública.

Art. 391.- Contravenciones de Sexta clase: Son sancionadas con multa equivalente al diez por ciento de un salario básico unificado del trabajador general y reducción de tres puntos en su licencia de conducir:

Ejemplos de contravenciones de sexta clase:

- La o el conductor de un vehículo automotor que circule contraviniendo las normas previstas en los reglamentos de tránsito y demás disposiciones aplicables, relacionadas con la emanación de gases.
- La persona que no conduzca su vehículo por la derecha en las vías de doble dirección.
- La o el conductor que invada con su vehículo las vías exclusivas asignadas a los buses de transporte rápido.

Art. 392.- Contravenciones de Séptima clase: Son sancionadas con multa equivalente al cinco por ciento de un salario básico unificado del trabajador general y reducción de uno punto cinco puntos en su licencia de conducir.

Ejemplos de contravenciones de séptima clase:

- La o el conductor que use inadecuada y reiteradamente la bocina u otros dispositivos sonoros contraviniendo las normas previstas en los reglamentos de tránsito y demás normas aplicables, referente a la emisión de ruidos.
- La o el conductor de transporte público de servicio masivo de personas y comercial cuyo vehículo circule sin los distintivos e identificación reglamentarios, sobre el tipo de servicio que presta la unidad que conduce.
- La persona con discapacidad que conduzca un vehículo adaptado a su discapacidad sin la identificación o distintivo correspondiente.

CAPITULO 2

Metodología

2.1 Tipo de investigación:

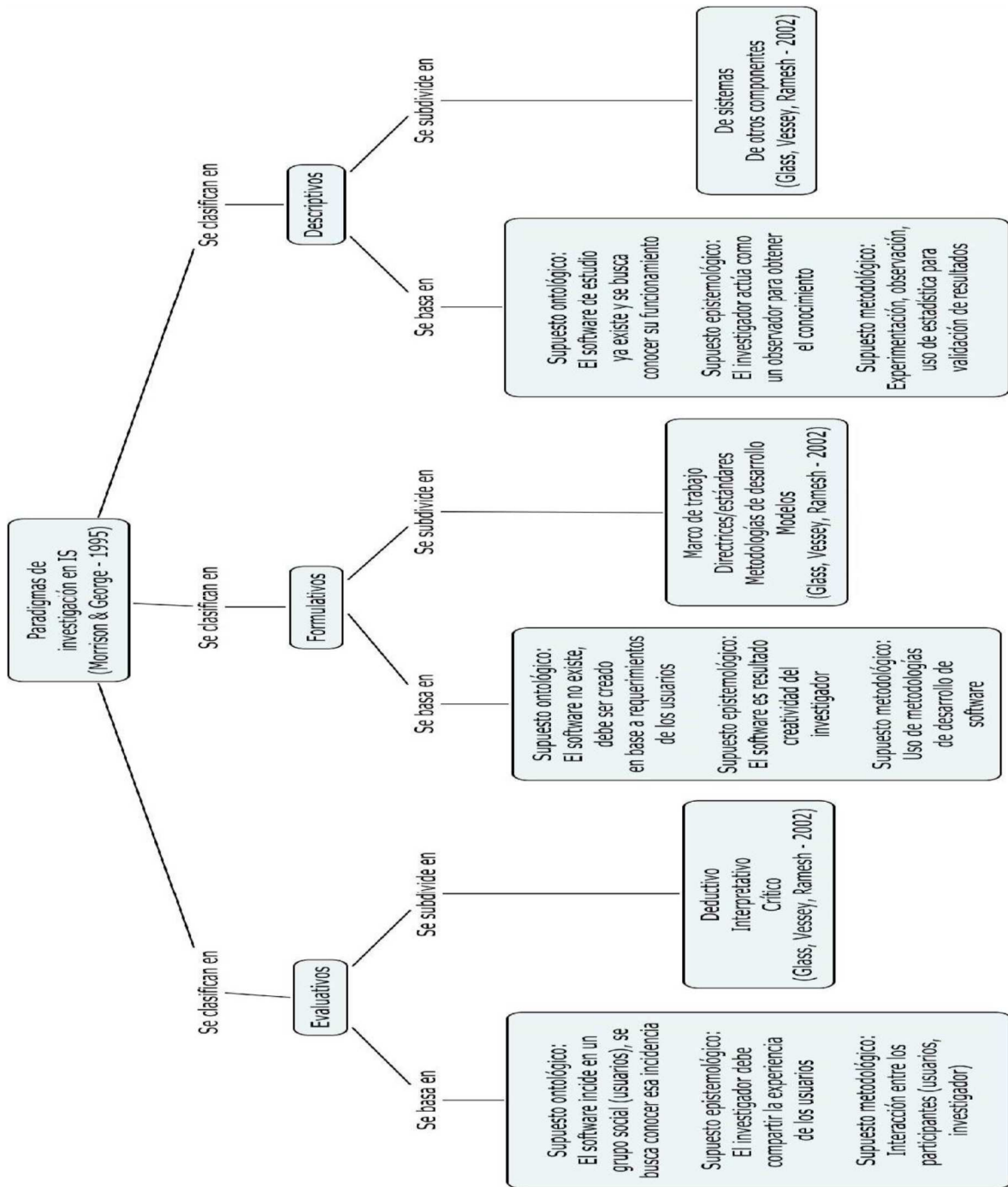
El presente trabajo se plantea desde los paradigmas de investigación de la Ingeniería del Software, específicamente bajo el paradigma formulativo, debido a que los supuestos ontológico, epistemológico y metodológico, se acoplan al desarrollo del simulador de conducción vehicular.

Pressman define: "La ingeniería del software surge a partir de las ingenierías de sistemas y de hardware, y considera tres elementos clave: que son los métodos, las herramientas y los procedimientos que facilitan el control del proceso de desarrollo de software y brinda a los desarrolladores las bases de la calidad de una forma productiva".

Zuma Cataldi define : "El software o producto, en su desarrollo pasa por una serie de etapas que se denominan ciclo de vida, siendo necesario, definir en todas las etapas del ciclo de vida del producto, los procesos, las actividades y las tareas a desarrollar."

Sigwart define: "se denomina ciclo de vida a toda la vida del software, comenzando con su concepción y finalizando en el momento de la desinstalación del mismo".

Gráfico 5: Paradigmas de investigación



Elaborado por: Ing. Cesar Salazar

El simulador de conducción vehicular es un software innovador, realizado con la creatividad y buenas prácticas del investigador en base a los requerimientos funcionales, no funcionales y metodologías del desarrollo del software.

2.1.1 Metodología del ciclo de vida

La metodología seleccionada para el presente trabajo es la metodología del ciclo de vida, que comprende las siguientes etapas:

- a) Fase de Análisis
- b) Fase de Diseño
- c) Fase de Desarrollo
- d) Fase de Pruebas.
- e) Fase de Implementación

2.1.1.1 Fase de Análisis

Se clasifica los requerimientos funcionales y no funcionales para poder seleccionar las herramientas a utilizar para implementar el simulador de conducción vehicular.

2.1.1.2 Fase de Diseño

En esta fase después de analizar los requerimientos se procede a crear los modelos 3D, texturas, menú de opciones. Todo lo necesario que valla a utilizar el simulador. Los modelos 3D se crean en software especializado en modelado 3D.

2.1.1.3 Fase de Desarrollo

Una vez que los objetos 3D y las texturas este correctamente ubicadas en el escenario se procede a crear los scripts.

2.1.1.4 Fase de Pruebas

Se realiza pruebas para comprobar que la funcionalidad simulador este al 100%, que funcione correctamente las físicas, no las texturas no tengan ningún problema, que las señales de tránsito cumplan con lo establecido en el reglamento, caso contrario se procederá a corregir el problema.

2.1.1.5 Fase de Implementación

Después de realizar las pruebas y comprobar que no exista ningún problema con los temas mencionado antes se procede a publicar el simulador.

2.2 Diseño de la investigación:

La investigación se acopla al diseño No experimental Transaccional Descriptivo debido que los datos serán recolectados en un parque vial donde las variables están lo más cercano a la realidad.

2.2.1 Población y muestra

Los datos se van a recolectar por medio de los profesores de las materias de: fundamentos de programación, programación orientada a objetos, programación en capas y programación distribuida de la Carrera de Ingeniería en Sistemas de la Universidad Católica de Santiago de Guayaquil.

Para el diseño del simulador se plantea trabajar con un parque vial público y uno privado debido a que todos los parques viales están regulados y cumplen con la normativa de la Agencia Nacional de Tránsito según lo indica el artículo 188 de la ley orgánica de transporte terrestre tránsito y seguridad vial.

2.2.2 Técnicas e instrumentos para obtención de información

Se define utilizar 2 técnicas de obtención de información para obtener los requerimientos funcionales y no funcionales para el desarrollo del simulador de conducción vehicular, las técnicas son entrevista semi-estructurada y la observación sistemática directa.

2.2.2.1 Entrevista

Por medio de una entrevista semi-estructurada, la cual consiste en que el investigador ejecuta una serie de preguntas preparadas y espontaneas al entrevistado.


Será dirigida a los docentes de las siguientes materias de la Carrera de Ingeniería en Sistemas de la Universidad Católica de Santiago de Guayaquil que durante el semestre A2016 son:

Tabla 11: Docentes que serán entrevistados

Docente	Materia
Ing. Tanya Janeth Armijos Ramos	Fundamentos de programación
Ing. Galo Enrique Cornejo Gómez	Programación orientada a objetos
Ing. Franklin Xavier González Soriano	Programación en capas
Ing. Galo Enrique Cornejo Gómez	Programación distribuida

Elaborado por: los autores

Gráfico 6: MODELO DE ENTREVISTA PARA DOCENTES

<p style="text-align: center;">MODELO DE ENTREVISTA PARA DOCENTES</p> <p style="text-align: center;"></p> <p style="text-align: center;">Universidad Católica Santiago de Guayaquil Entrevista a docentes del área de programación</p> <p>Nombre: Materia: Fecha:</p> <ol style="list-style-type: none">1. ¿Cree usted que los estudiantes se verían atraídos por temas de desarrollo de videojuegos?2. ¿Qué opina acerca de desarrollar un simulador de conducción vehicular utilizando un motor de videojuegos?3. ¿Le sería útil este simulador para utilizarlo como base para futuros proyectos de tutoría en su materia?4. ¿Qué lenguaje de programación utiliza en su materia?5. ¿Qué lenguaje de programación considera que debería utilizar el motor de videojuegos?6. ¿Qué tan importante es para usted que el motor de videojuegos tenga una gran comunidad de desarrolladores?7. ¿Considera importante que el motor de videojuegos genere software multiplataforma (Windows, Mac, Android, Ps4, etc.)?

Elaborado por: los autores

2.2.2.2 Observación

La observación será sistemática debido a que en el investigador utilizara una ficha de observación para registrar la información y será directa porque será realizada en el parque vial de la CTE ubicado en Av. Juan Tanca Marengo el día jueves 14 de Julio de 2016 a las 14:00 y en la escuela de conducción CONDUESPOL ubicada en el Km 30,5 de la vía Perimetral "Campus Gustavo Galindo" Espol el día viernes 15 de Julio de 2016 a las 11:00.

Gráfico 7: MODELO DE FICHA DE OBSERVACIÓN

Fecha:
Hora:
Lugar:

FICHA DE OBSERVACION DEL PARQUE VIAL

1. VEHICULO

1.1 Marca:

1.2 Color:

1.3 Motor: HP

1.4 Cilindraje: CC

1.5 Modelo:

1.6 Tipo de transmisión: Manual Automática

2. EJECUCION DE LA PRUEBA

2.1 Duración de la prueba:

--

Minutos

2.2 Velocidad permitida:

--

Km/hora

3. OBSTACULOS

3.1 Conos:

SI	NO

3.2 Lomas :

--	--

3.3 Reductores de velocidad:

--	--

3.4 Llantas

--	--

3.5 Peatones:

--	--

3.6 Cantidad de obstáculos:

--

5. CALLE (CALZADA)

5.1 ¿Se encuentra señalizada?

5.2 Material de la calzada:

6. SENALES DE TRANSITO

Nombre de la señal de transito

Cantidad

6.1

6.2

6.3

Elaborado por: los autores

Se observará y registrará información relevante del parque vial como características del vehículo, de ejecución de la prueba, de los obstáculos en el circuito y de la señales de tránsito encontrados en el mismo.

2.2.3 Análisis de la información obtenida:

2.2.4 Entrevista

Estos resultados servirán para determinar los requerimientos funcionales y no funcionales que deba cumplir el motor de videojuegos para realizar el simulador de conducción vehicular.

2.2.4.1 Requerimientos funcionales

2.2.4.1.1 Lenguaje de programación

Los docentes indican que lenguaje de programación en su materia y que por tanto sería importante que el motor de videojuegos utilizara dicho lenguaje.

Tabla 12: Lenguajes de programación utilizados en las materias de programación.

Materia	Lenguaje de Programación
Fundamentos de programación	C++
Programación orientada a objetos	Java
Programación en capas	C#
Programación distribuida	Java

Elaborado por: los autores

2.2.4.1.2 Curva de aprendizaje

Los docentes concuerdan en que desearían que la curva de aprendizaje sea rápida, pero son conscientes de que una nueva herramienta generalmente tiene una curva lenta de aprendizaje.

En todo caso prefieren un motor de videojuegos básico con una curva de aprendizaje rápida en lugar de un motor potente con una curva de aprendizaje lenta.

2.2.4.1.3 Recursos de hardware

Los docentes mencionan que los recursos de hardware que cuentan son los equipos de cómputo que se encuentran en el laboratorio donde dictan su materia los cuales cuentan con un procesador Intel Core i5, 8 GB de memoria RAM y 1000 GB de disco duro.

2.2.4.2 Requerimientos no funcionales

2.2.4.2.1 Aceptación de videojuegos

La totalidad de los docentes entrevistados indica que creen que los estudiantes mostraran interés a desarrollar videojuegos, pues es un tema que les gusta generalmente a los estudiantes.

Los docentes de la materia fundamentos de programación, programación orientada a objetos y programación distribuida mencionan que ya han utilizada el tema de videojuegos en sus proyectos de tutoría y que sus estudiantes se muestran interesados en realizar dichos proyectos.

2.2.4.2.2 Simulador multiplataforma

Los entrevistados mencionan que es interesante y en cierta medida útil que el motor de videojuegos genere software a diferentes plataformas, pero no lo considera como una característica o función indispensable que debe tener el mismo.

2.2.4.3 Conclusión final de la entrevista

En base a los resultados obtenidos en la entrevista, se realiza el siguiente cuadro comparativo entre los motores de videojuegos, para determinar cuál es la herramienta idónea para implementar el simulador de conducción vehicular.

Tabla 13: Comparación de los motores de videojuegos.

Requerimientos	Motores de videojuego		
	CryEngine	Unity	Unreal
Lenguaje de programación C++	✓		✓
Lenguaje de programación C#	✓	✓	
Lenguaje de programación JAVA			
Curva de aprendizaje rápida		✓	
Requisitos de hardware que se adapten a los recursos de los laboratorios de programación.	✓	✓	✓
Posibilidad de generar ejecutables en diversas plataformas con el mismo código.	✓	✓	✓
Total	4	4	3

Elaborado por: los autores

El cuadro comparativo indica que los motores de videojuegos Unity y CryEngine cumplen con mayor cantidad de requerimientos, pero el hecho de que Unity cuente con una curva de aprendizaje rápida y una gran comunidad de soporte permite determinar a Unity como motor de videojuegos para el desarrollo del simulador de conducción vehicular.

2.2.5 Observación

La información registrada a través de la observación en los parques viales servirá para determinar los requerimientos funcionales y no funcionales que deberá contar el simulador de conducción vehicular.

2.2.5.1 Requerimientos funcionales

2.2.5.1.1 Tipo de señalética

Se observó que ambos parques viales se cuenta con señalética vertical en los cuales se destaca la presencia de las siguientes:

- 3 Señal de "PARE"
- 4 Señal de "UNA VÍA"
- 5 Señal de "DOBLE VÍA"
- 6 Señal de "LIMITE DE VELOCIDAD"
- 7 Señal de "PROHIBIDO GIRAR A LA IZQUIERDA"

2.3.2.1.2 Vehículo

Se observó que el vehículo utilizado para realizar la prueba es diferente en ambos parques viales, en el parque vial de la CTE utilizan un Chevrolet Aveo tipo Sedan con un motor de 1600 cc, por otra parte en la escuela CONDUESPOL utilizan un Chevrolet Aveo tipo hatchback de 1400 cc, en ambas instituciones el vehículo contaba con transmisión manual.

2.3.2.1.3 Velocidad

La velocidad permitida de la prueba según la señalética que se observaba era de 30 kilómetros por hora.

2.3.2.2 Requerimientos no funcionales

2.3.2.2.2 Tipo de calzada

En ambos parques viales se observó que la calzada era de asfalto y en ambos contaba con señalización horizontal como:

- Doble línea amarilla
- Línea blanca longitudinal continua
- Línea blanca longitudinal discontinua
- Senda peatonal

2.3.2.2.3 Obstáculos

Ambos parque viales no cuentan con una gran cantidad de obstáculos, se destaca que en el parque de la CTE se observó una pendiente y la utilización de conos para presumiblemente evitar choques con las paredes del recinto.

2.3.2.3 Conclusión final de la observación

De la observación realizada se concluye que el simulador de conducción debe contar con un vehículo tipo Sedan o Hatchback, en el escenario se debe encontrar la siguiente señalización vertical:

- Señal de “PARE”
- Señal de “UNA VÍA”
- Señal de “DOBLE VÍA”
- Señal de “LIMITE DE VELOCIDAD”
- Señal de “PROHIBIDO GIRAR A LA IZQUIERDA”

En cuanto a la calzada esta debe ser de asfalto y se debe encontrar señalizada con al menos la siguiente señalética vertical:

- Señal de “DOBLE LINEA AMARILLA”
- Señal de “LINEA BLANCA LONGITUDINAL CONTINUA”
- Señal de “LINEA BLANCA LONGITUDINAL DISCONTINUA”
- Señal de “SENDA PEATONAL”
-

CAPITULO 3

Diseño e implementación del simulador

3.1 Instalación y configuración del entorno de desarrollo:

Para la implementación del simulador se necesitó los siguientes recursos de hardware y software:

3.1.1 Software

El simulador se desarrolló en el Sistema operativo de Windows, los scripts fueron programados con C# como lenguaje de programación. Para los modelos 3D que requerían mayor detalle se utilizó 3D Max 2017 y para la edición y creación de texturas se usó Photoshop Cs6, ciertos objetos y texturas fueron editadas con Unity. Debido a que el software de modelado 3D tiene un consumo de tarjeta de video mayor al de Unity se adquirió una tarjeta de video.

Tabla 14: Software utilizado en el desarrollo del simulador

Sistema operativo	Microsoft Windows 7 Professional
Versión	Service Pack 1
Tipo de sistema	basado en x64
Motor videojuego	Unity
Modelado 3D	3D Max 2017
Texturas	Photoshop Cs6

Elaborado por: los autores

3.1.1.1 Requisitos de software de Unity

Para la correcta instalación de Unity se debe tener instalado previamente lo siguiente:

- Microsoft. NET Framework 4.5, el cual puede ser descargado gratuitamente de la página web de Microsoft a través del siguiente link:<https://www.microsoft.com/es-es/download/details.aspx?id=30653>
- GTK# for .Net 2.12.26, el cual puede ser encontrado en la página oficial del Proyecto GTK la cual es:
<http://www.gtk.org/download/>

3.1.1.2 Fuentes de información de Unity

El motor de videojuegos Unity dispone de una gran documentación acerca de su utilización y configuración, además de contar con una tienda de recursos de modelos 3D y una comunidad de respuestas y dudas a usuarios.

Estos recursos pueden se encuentran disponibles en internet, véase la tabla 15.

Tabla 15: Fuentes de información de Unity

Recurso	Enlace del recurso
Manual de Unity Versión 5.3	http://docs.unity3d.com/es/current/Manual/index.html
Tienda de Unity	https://store.unity3d.com/account/users
Foro de Unity	https://unity3d.com/es/community
Foro de Unity (Español)	http://www.unityspain.com/

Elaborado por: Los autores

3.1.2 Hardware

Los requisitos mínimos de hardware que utiliza Unity es una computadora con al menos 2 GB de memoria ram, un procesador compatible con el conjunto de instrucciones SSE2 y una tarjeta de video con DX9. Para el desarrollo del simulador se utilizó un hardware superior debido a que ese era el hardware que se contaba, véase la tabla 15.

Tabla 16: Hardware utilizado en el desarrollo del simulador

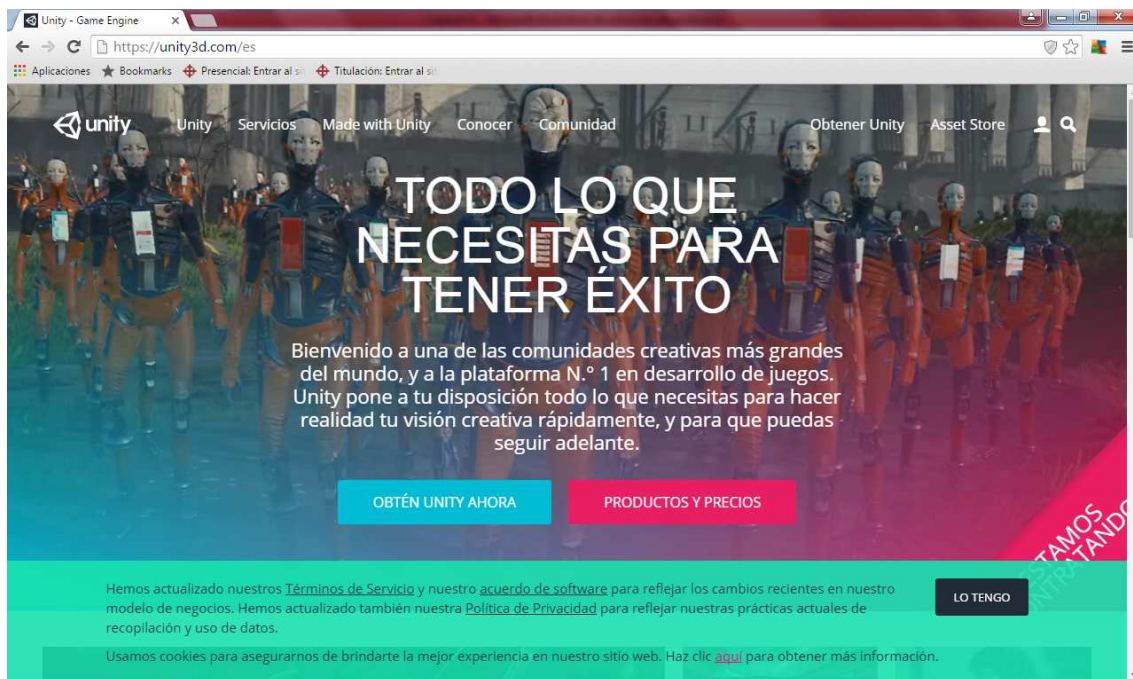
Procesador	Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz, 3401 Mhz
Memoria	8GB
Placa madre	Intel DH67BL
Tarjeta video	MSI GEFORCE GTX 960 2GB
Disco duro	1 TB

Elaborado por: los autores

3.1.3 Instalación de Unity 5.3.5

Para instalar el motor de Unity se tiene que acceder a la página oficial de Unity en el siguiente enlace: <https://unity3d.com/es>

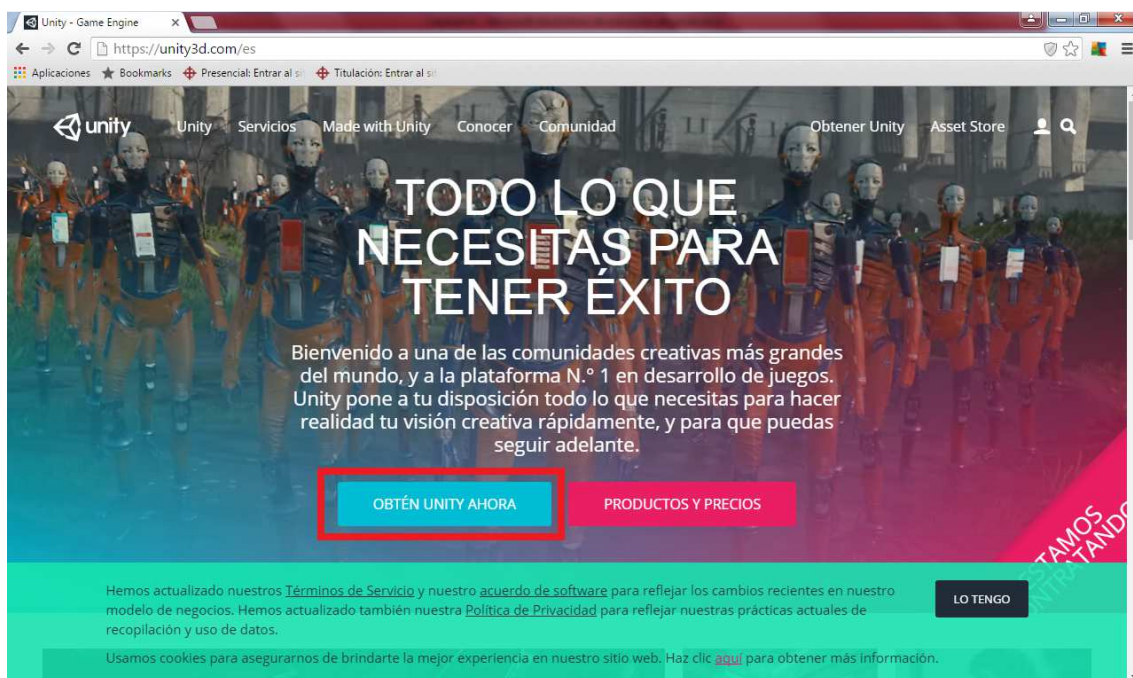
Gráfico 8: Página oficial de Unity



Fuente: Sitio web de Unity (2016)

Para acceder a las diferentes versiones que Unity brinda se debe hacer clic en el botón “**OBTEN UNITY AHORA**”.

Gráfico 9: Página oficial de Unity



Fuente: Sitio web de Unity (2016)

Para el desarrollo del simulador se utilizó la versión Personal, debido a que totalmente gratuita y ofrece todas las funciones de su motor de videojuegos.

Gráfico 10: Versiones de Unity



Fuente: Sitio web de Unity (2016)

Después de haber hecho clic en la versión Personal se empezará a descargar el instalador.

Cuando la descarga del instalador termine se debe seguir los siguientes pasos para completar el proceso de instalación:

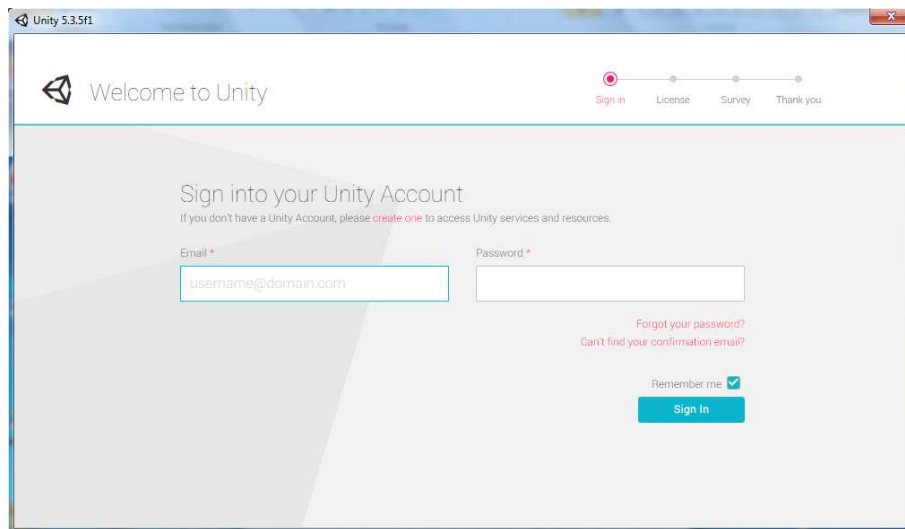
Se debe ejecutar el instalador como administrador.

1. Aparecerá la página del asistente de descarga y se debe clicar en el botón siguiente.
2. Se aceptarán los términos y condiciones y luego se debe hacer clic en el botón siguiente.
3. Se selecciona los componentes que se desea instalar y se presionar siguiente.

4. Se determina la ubicación donde se realizará la instalación de Unity por defecto la ruta es: C:\Program Files\Unity, luego se debe hacer clic en el botón siguiente.
5. Una vez finalizado el proceso de instalación, presionar Finalizar.

Cuando se ejecuta Unity por primera vez el programa solicita crear una cuenta en Unity. Una vez creado el usuario, se escriben los datos y se presiona el botón **“Sign in”**.

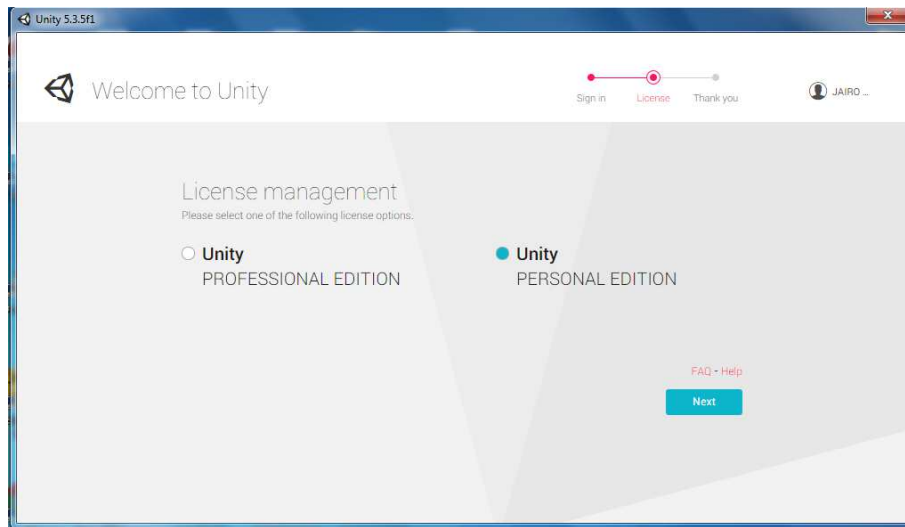
Gráfico 11: Creación de cuenta con el software de Unity



Elaborado por: los autores

En la siguiente ventana el programa solicita que se indique con cual tipo de licencia será usado, se debe elegir la opción de Unity PERSONAL EDITION.

Gráfico 12: Selección de versión de Unity

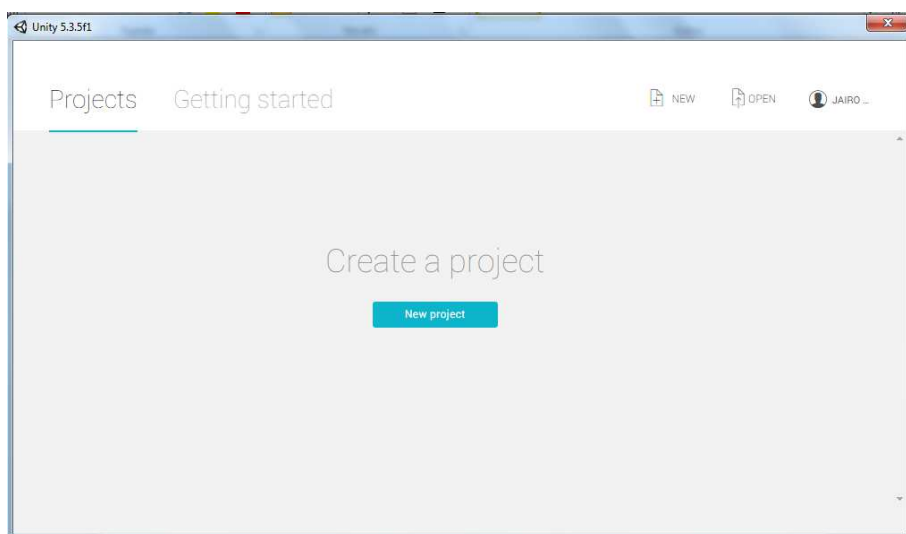


Elaborado por: los autores

3.1.4 Creando un proyecto nuevo

Para empezar a usar Unity se debe crear un nuevo proyecto, Unity permite crear proyectos para soluciones en 3D y 2D. Al momento de crear un nuevo proyecto se define que paquetes predefinidos de Unity se va a importar como es el Estándar Assets u otros paquetes que han sido descargados de la tienda de Unity.

Gráfico 13: Creación de nuevo proyecto en Unity



Elaborado por: los autores

3.1.5 Interfaz

La interfaz de la aplicación Unity se encuentra conformada de la siguiente manera:

Gráfico 14: Interfaz del motor de videojuegos



Elaborado por: los autores

1. Vista de escena: Se visualiza todos los objetos 3D o 2D que se agreguen al proyecto.
2. Vista del juego: Se pre visualiza el juego.
3. Vista de jerarquía: Contiene todos los objetos que ha sido agregados a la escena.
4. Vista del proyecto: Se visualiza la carpeta assets que es el lugar donde se encuentran las texturas, scripts, objetos 3D, sonidos, etc. En esta sección han sido importados todos los recursos necesarios para el proyecto.
5. Vista del inspector: Se visualiza y edita las propiedades de los objetos del objeto seleccionado.

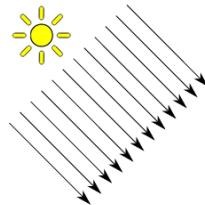
3.2 Implementación del simulador del conducción vehicular:

3.2.1 Grafico

3.2.1.1 Iluminación

El motor de Unity ofrece efectos ambientales como la iluminación y viento. Para la iluminación se utilizó *DirectionalLights*, que cuenta con propiedades para el color de la luz y la intensidad de la misma.

Gráfico 15: DirectionalLights (Luz direccionada)



Fuente: Sitio web de Unity (2016)

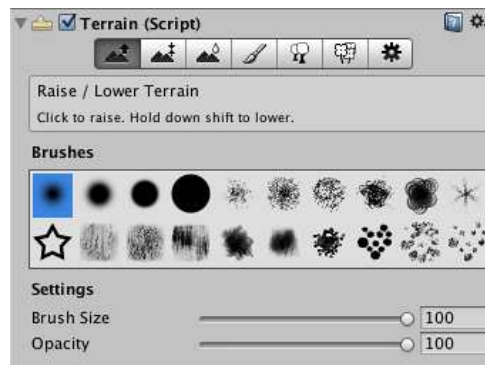
3.2.1.2 Motor del terreno

3.2.1.2.1 Terreno

El primer paso para crear el simulador fue crear el terreno ya que es el lugar donde se encuentran ubicados todos los objetos 3d, para agregar un terreno vamos a la opción *GameObject > 3D Object > Terrain*. El terreno ofrece diferentes opciones para ser editado como:

- Elevar el terreno que nos ayuda a crear montañas.
- Hundir el terreno nos ayuda a crear ríos.
- Agregar texturas al terreno.
- Agregar árboles, césped.

Gráfico 16: Propiedades del terreno

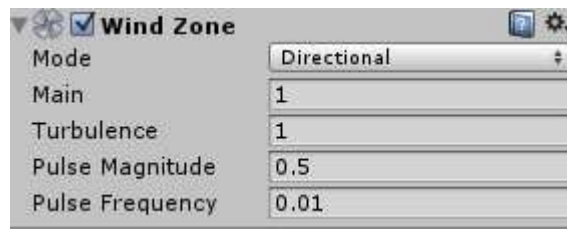


Elaborado por: los autores

3.2.1.2.2 Zonas de viento

Para los efectos del viento se utilizó *WindZone*, que permite graduar la velocidad y la dirección del viento. Se utilizó este efecto ambiental para que los arboles obtengan movimiento en el transcurso de la simulación, también para la velocidad del vehículo ya que se ve afectada por la resistencia al viento.

Gráfico 17: Propiedades del viento



Elaborado por: los autores

3.2.1.3 Cámaras

El simulador está compuesto por 3 cámaras:

- Interior del vehículo.
- Parte trasera del vehículo.
- Vista aérea frontal.

3.2.1.4 Materiales y texturas

Las texturas de las señales de tránsito fueron creadas en Photoshop Cs6, ciertas texturas como límite de velocidad se tomó la foto a la señal de tránsito y luego se la editó para ser usada en el modelo 3D.

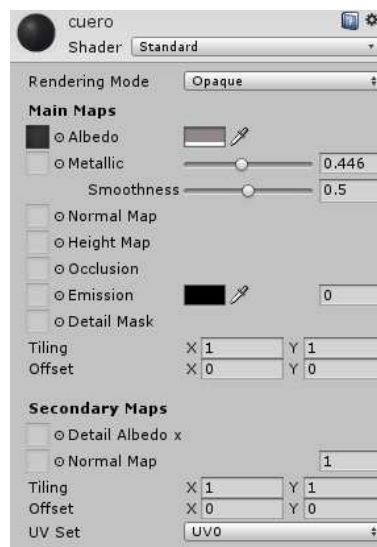
Gráfico 18: Fotografía de señal de límite de velocidad



Elaborado por: los autores

Texturas como césped, cielo fueron descargadas de la tienda de UnityAsset Store. Para agregar una textura a un objeto en Unity primero se crea el material Assets->Create->Material, en las configuraciones del material en la opción MainMaps dentro de ella tenemos Albedo en la misma se le agrega la textura que va a utilizar el material.

Gráfico 19: Propiedades del material



Elaborado por: los autores

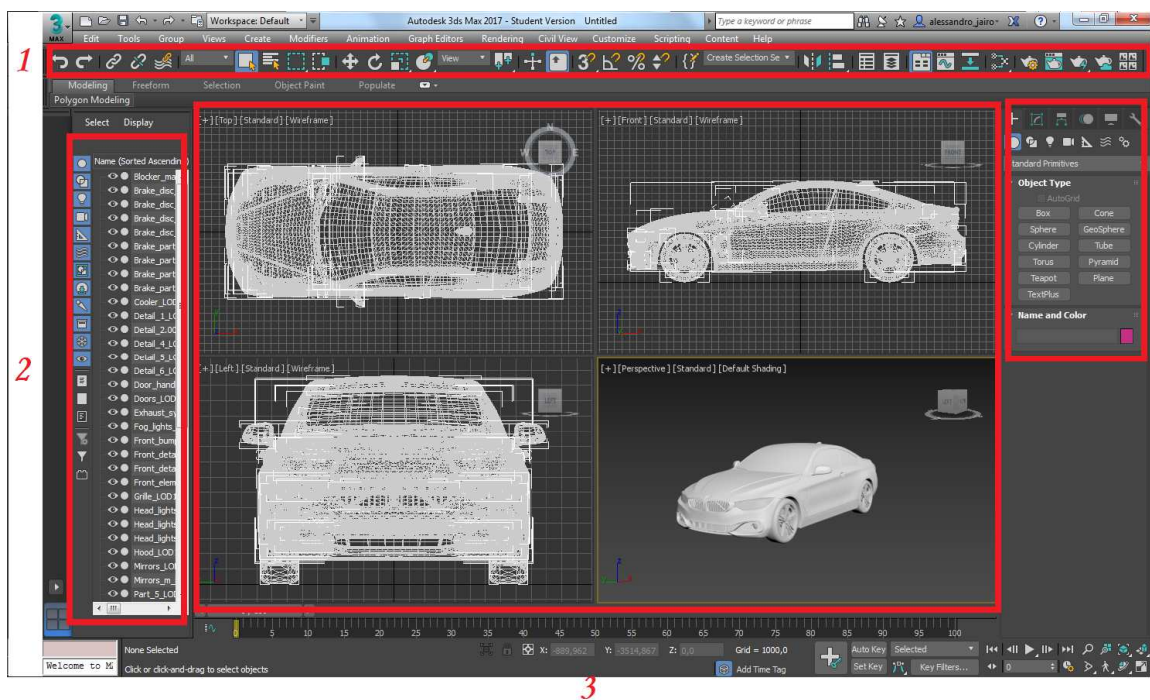
3.2.2 Modelado 3D

Para la elaboración y modificación de los objetos 3D se utilizó el software de modelo 3D Max.

3.2.2.1 Interfaz

3D Max cuenta con diferentes tipos de opciones para la edición de objetos 3D, a continuación se describe las herramientas más utilizadas en el desarrollo del simulador de conducción vehicular:

Gráfico 20: Interfaz de 3D Max



Elaborado por: los autores

1. **Barra de herramientas:** Se encuentran opciones para renderizado, seleccionar y mover el objeto, rotar el objeto, duplicar el objeto, entre otras. Las opciones frecuente el proyecto fueron:

Gráfico 21: Icono para mover un objeto



Elaborado por: los autores

Gráfico 22: Icono para rotar un objeto



Elaborado por: los autores

2. **Escenario:** Se encuentran todos los objetos que se utilizaron para la creación del modelo como cubos, cilindros. Además se crea jerarquía en los objetos.
3. **Ventana:** Está conformada por 4 ventanas en las que se tiene una visión del objeto de diferentes perspectivas como:
 - Visión frontal
 - Visión lateral
 - Visión aérea
 - Visión libre

Para ampliar la ventana en la que se va a trabajar se la selecciona y se presiona la combinación de teclas Alt+W.

4. **Panel de comandos:** Está compuesto por 6 componentes, las que se utilizaron en el proyecto son:
 - **Panel de creación:** La opción permite crear objetos como pirámide, cubos, círculos, cilindros, planos, cámara y luces.

Gráfico 23: Icono del panel de creación



Elaborado por: los autores

- **Panel de modificación:** Permite editar los objetos mediante los vértices, bordes, polígono.

Gráfico 24: Icono del panel de creación

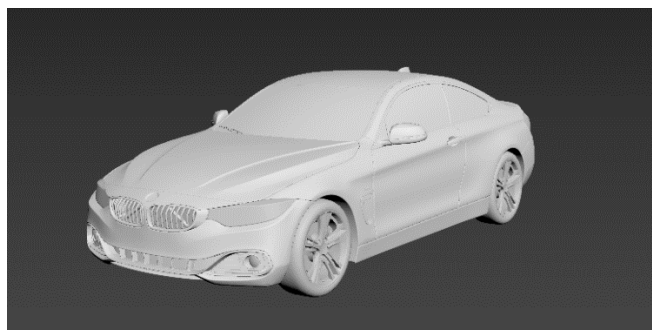


Elaborado por: los autores

3.2.2.2 Modelado del vehículo

El modelo del vehículo fue descargado de la tienda de Unity, el interior del vehículo fue creado en 3D Max, además se editó objetos del modelo original como los neumáticos entre otros.

Gráfico 25: Modelo del vehiculo en 3D Max

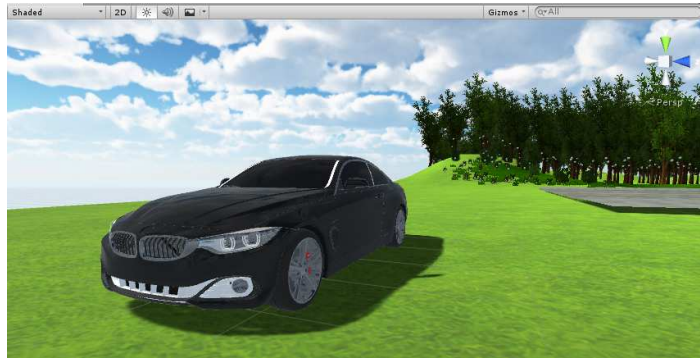


Elaborado por: los autores

Fue elegido para el proyecto debido a que todos sus componentes son objetos 3D individuales como: llantas, puertas, retrovisor, faros, etc. Esto resulta útil debido a

que para el movimiento de los neumáticos Unity necesita que sean objetos independientes. En Unity es posible agregarle las texturas al modelo 3D y y como resultado tenemos:

Gráfico 26: Modelo 3D del vehículo

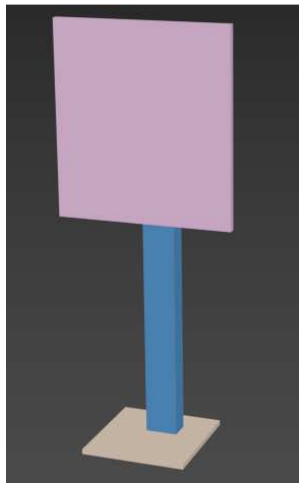


Elaborado por: los autores

3.2.2.3 Modelado objetos de tránsito

Las señales de tránsito, semáforos y calles fueron creadas en 3D Max, para crear los objetos primero se debe crear la base del modelo, en el caso de las señales de tránsito tenemos cubo para la base y el letrero, cilindro o un cubo para cuerpo que va a sostener la señal de tránsito.

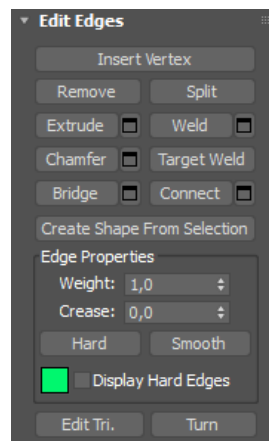
Gráfico 27: Modelo de base de señal de tránsito



Elaborado por: los autores

Para que el letrero tenga los vértices curvados primero se selecciona el objeto y se da clic derecho y se elige la opción *Convert To – Editable Poly*. Esta opción nos permite modificar el objeto por medio del panel de modificación de objeto.

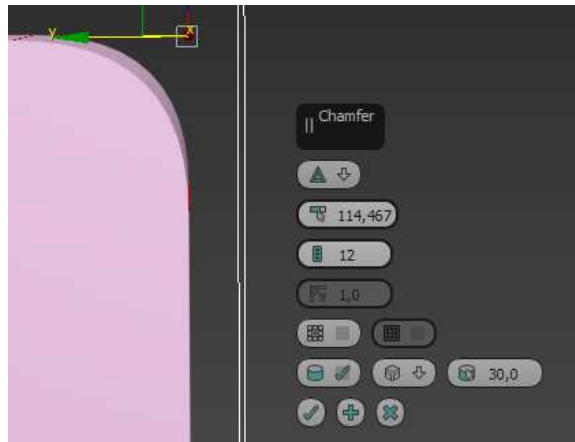
Gráfico 28: Panel de modificación de objeto



Elaborado por: los autores

En este caso se va a modificar el vértice con la opción Chamfer.

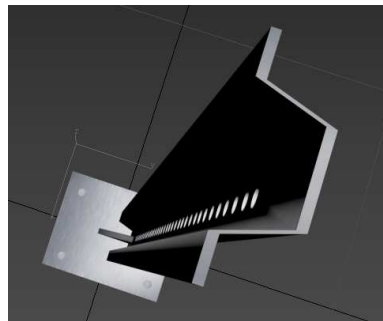
Gráfico 29: Configuración Chamfer



Elaborado por: los autores

Esta función permite agregar un grado de curva al vértice, entre más vértices tenga la curva tendrá un aspecto más realista. Para el cuerpo se modificó los polígonos cambiándolos de posición.

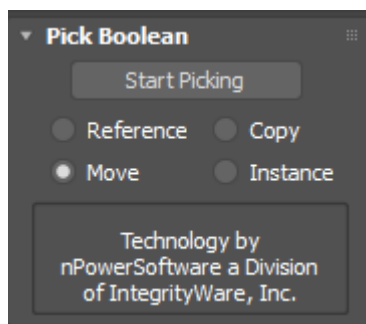
Gráfico 30: Modelo de cuerpo de la señal de tránsito



Elaborado por: los autores

Para realizar los orificios en el cuerpo del objeto se utilizó la opción CompoundObjects. Esta función permite quitar un fragmento de objeto que se encuentre dentro de otro.

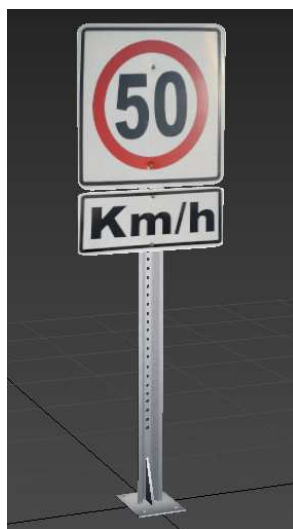
Gráfico 31: Función CompoundObjects



Elaborado por: los autores

Para las texturas se agrega un componente llamado UVW Map, ayuda a darle efectos a la textura como brillo, contraste.

Gráfico 32: Modelo de señal de tránsito



Elaborado por: los autores

Para finalizar se importan los modelos 3D elaborados con 3D Max al motor gráfico de videojuegos Unity.

Gráfico 33: Modelo 3D de señales de tránsito en Unity



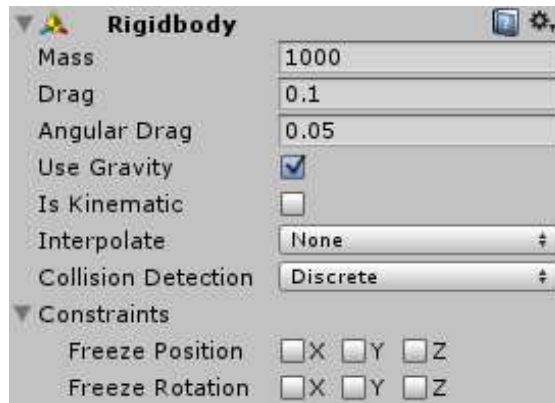
Elaborado por: los autores

3.2.3 Física

3.2.3.1 Rigidbodies

Este componente permite agregar al objeto un comportamiento físico, con el rigidbody el objeto responde a temas como la gravedad. Este componente se agrega al vehículo, semáforo, señales de tránsito, primero se debe tener seleccionado el objeto, luego en la vista del inspector en la opción *AddComponent* se elige *Physics* luego *Rigidbody*. Este complemento permite cambiar el valor de la masa (Kilogramos) del objeto, Si la opción *Use Gravity* se encuentra seleccionada el objeto se verá afectado con gravedad.

Gráfico 34: Componente de rigidbody



Elaborado por: los autores

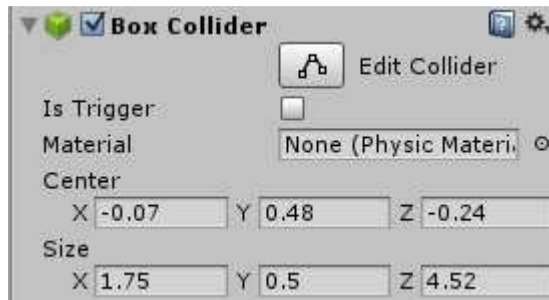
3.2.3.2 Colliders

Este componente se debe agregar al rigidbody, para que las colisiones y física entre dos objetos ocurran los dos objetos deben tener activada la opción collider caso contrario el motor de física no trabaja, en el caso que un objeto tenga rigidbody y no tenga un collider el objeto pasara a través del otro objeto sin ocurrir efectos de física. Unity brinda diferentes tipos de colliders:

- Box Collider
- Sphere Collider
- Capsule Collider
- MeshCollider
- Wheel Collider
- TerrainCollider

En el simulador se utiliza Box Collider, Capsule Collider, Wheel Collider, para agregar el collider al objeto se seleccionad el objeto luego en la vista del inspector en la opción *AddComponent* se elige *Physics* luego Box Collider.

Gráfico 35: Propiedades del Box Collider



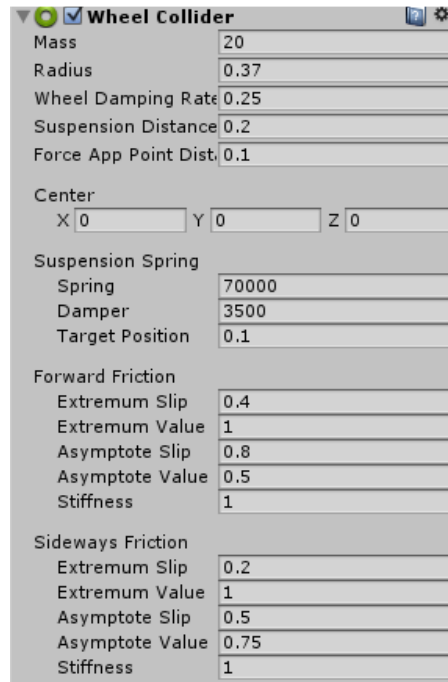
Elaborado por: los autores

Collider brinda diferentes funciones, la función *IsTrigger* permite usar el collider para eventos y desactiva el motor de física en el objeto. Esta función ha sido utilizada en el simulador para la detección de las diferentes infracciones de tránsito que cuenta el simulador. Box Collider han sido aplicado en el vehiculo y en los semáforos, postes de luz, señales de tránsito. SphereCollider se lo aplicó en las señales de tránsito en la la base metálica de las mismas ya que es de forma cilíndrica.

El vehículo además de usar Box Collider también utiliza Wheel Collider este tipo de collider es usado por vehículos terrestres que cuenten con ruedas. Este collider brinda las siguientes propiedades:

- Detección de colisión.
- Física de ruedas.
- Deslizamiento basado en la fricción.

Gráfico 36: Propiedades de Wheel Collider



Elaborado por: los autores

El vehículo es controlado por medio del script con propiedades que brinda Wheel Collider como:

- MotorTorque: Velocidad a la cual vana girar la rueda.
- BrakeTorque: Tiempo de frenado.
- SteerAngle: El ángulo de giro de la llanta.

3.2.4 Scripting

3.2.4.1 Programación del vehículo

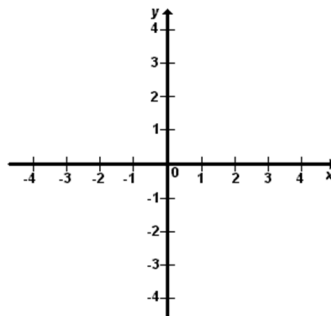
La clase input de Unity permite leer los ejes establecidos en el "Input Manager", también utilizado para leer los datos del acelerómetro de los teléfonos móviles. Los ejes por defecto son:

- Horizontal
- Vertical
- Fire1
- Fire2

- MouseX
- MouseY
- Jump

Los ejes Horizontal, Vertical, MouseX y MouseY hacen referencia al plano cartesiano tal como se muestra en el gráfico 37.

Gráfico 37: Plano cartesiano



Elaborado por: los autores

Eje Horizontal, MouseX hace referencia al eje X en el plano cartesiano, eje Vertical, MouseY pertenecen al eje Y del plano cartesiano. Fire1 hace referencia al botón izquierdo del mouse y Fire2 al botón derecho.

3.2.4.1.1 ControlUsuarioVehiculo

Se encarga de recibir el Input del volante que pertenece al eje Horizontal y el de los pedales que pertenecen al eje Vertical. Se declara una variable de tipo ControladorVehiculo ya que este script es el que encarga de movimiento del vehículo con la función Mover(), en la función Awake() se le asigna a la variable m_Car el script del vehículo mediante la clase GetComponent permite acceder a otros componentes del objeto como audio, scripts, collider, etc. En la función FixedUpdate() se declara 2 variables de tipo float, a la variable h se le asigna los datos del volante y a la variable Y se le asigna los datos del frenado y aceleración. Luego se manda como parámetro los datos de entrada a la función Mover() del script ControladorVehiculo mediante el objeto m_Car. Véase el gráfico 38.

Gráfico 38 Script ControladorVehiculo

```
public class ControlUsuarioVehiculo : MonoBehaviour
{
    private ControladorVehiculo m_Car;
    public float h;
    public float v;
    private void Awake()
    {
        m_Car = GetComponent<ControladorVehiculo>();
    }

    private void FixedUpdate()
    {
        h = CrossPlatformInputManager.GetAxis("Horizontal");
        v = CrossPlatformInputManager.GetAxis("Vertical");

        float frenoMano = CrossPlatformInputManager.GetAxis("Jump");
        m_Car.Mover(h, v, v, frenoMano);
    }
}
```

Elaborado por: los autores

3.2.4.1.2 ControladorVehiculo

Este script es el más importante de todos porque brinda las funciones del movimiento del vehículo, caja de cambio, centro de masa, ajuste de torque, etc. El script está compuesto por las siguientes funciones:

- Start
- Update

Tenemos las siguientes variables declaradas:

1. WheelCollider[] m_WheelColliders: Arreglo de variables de tipo WheelCollider ya que el auto tiene 4 ruedas las cuales se deben asignar a m_WheelColliders.
2. GameObject[] m_WheelMeshes: Arreglo de variables de tipo GameObject, aquí se asigna la rueda(objeto).

3. Rigidbody m_Rigidbody.
4. Vector3 m_CentroDeMasa
5. float m_ControlDeTraccion;

3.2.4.1.3 Función Start

Agrega el centro de masa del vehículo a las ruedas para que cuando el vehículo tenga un volcamiento mantenga su posición original caso contrario dará vueltas de campanas hasta que disminuya su velocidad. La variable m_Rigidbody se le agregar el componente Rigidbody del vehículo, m_TorqueActual esta variable tendrá como valor el torque ajustado con el control de tracción ya que la tracción previene la perdida de adherencia de las ruedas. Véase el gráfico 39.

Gráfico 39 Script función Start

```
private void Start()
{

    m_WheelColliders[0].attachedRigidbody.centerOfMass = m_CentroDeMasa;

    m_Rigidbody = GetComponent<Rigidbody>();
    m_TorqueActual = m_FullTorqueTodasLasLlantas - (m_ControlDeTraccion*m_FullTorqueTodasLasLlantas);
}
```

Elaborado por: los autores

3.2.4.1.4 Función Update

Esta función controla las luces direccionales al momento de aplastar el botón correspondiente a cada una de ellas ya se la luz izquierda, derecha o ambas. Una vez activada se invoca a la función StartCoroutine (parpadeodirecionderch ()), la

función StartCorutine () permite ejecutar una función durante un intervalo de tiempo hasta que la misma termine. Véase el gráfico 40.

Gráfico 40 Script función Update

```
private void Update()
{
    if (Input.GetButtonUp("DireccionDerecha"))
    {
        if (estado_izquierda == false )
        {
            estaParpadeando = true;
            estado_derecha = true;
            band = band + 1;
            if (band > 1) { estaParpadeando = false; band = 0; estado_derecha = false; }
            StartCoroutine(parpadeodireecionderch());
        }
    }
}
```

Elaborado por: los autores

La función parpadeodireecionderch () es la que enciende y apaga la direccional durante un intervalo de 0.5 segundos utilizando la clase WaitForSeconds (). Véase el gráfico 41.

Gráfico 41 Script para direccionales del vehiculo

```
public IEnumerator parpadeodireccionderch()
{
    //el parpadeo sera infinito ion
    while (estaParpadeando)
    {
        derecha.enabled = true;
        visible = true;

        //mostramos la direccional por 0.5 segundos
        yield return new WaitForSeconds(.5f);

        derecha.enabled = false;
        visible = false;
        yield return new WaitForSeconds(.5f);
    }
}
```

Elaborado por: los autores

3.2.4.1.5 Función CambioDeMarcha

Esta función es la que controla la marcha en la que se encuentra el vehículo, primero se declara una variable de tipo float f, que tendrá como valor la velocidad actual dividido para velocidad máxima. Límite superior e inferior nos indica en qué momento se debe hacer el cambio de marcha ya sea subir o bajar la marcha. Véase el gráfico 42.

Gráfico 42 Función cambio de marcha

```
private void CambioDeMarcha()
{
    float f = Mathf.Abs(VelocidadActual/VelocidadMax);
    float limitesuperior = (1/(float) TotalCajaCambio)*(m_NumeroCambio + 1);
    float limiteinferior = (1/(float) TotalCajaCambio)*m_NumeroCambio;

    if (m_NumeroCambio > 0 && f < limiteinferior)
    {
        m_NumeroCambio--;
    }

    if (f > limitesuperior && (m_NumeroCambio < (TotalCajaCambio - 1)))
    {
        m_NumeroCambio++;
    }
}
```

Elaborado por: los autores

Ejemplo: si se tiene una velocidad de 4 y velocidad máxima de 160 el valor de f será 0.025, límite superior será en el caso que la caja de cambio tenga 7 velocidades 0.14 y límite inferior será 0, entonces cuando mi velocidad llegue a 23km/h el valor de f será 0.14 entonces se hace el cambio de marcha.

3.2.4.1.6 Función Tacómetro

Esta función permite que el puntero llegue a su máxima revolución por minuto de la marcha luego el puntero se regrese a 0 para empezar a contar las revoluciones por minuto de nuevo. La función `Math.InverseLerp()` devuelve un valor que se encuentra entre 0 y 1, tiene como parámetro el intervalo inferior y superior de la función `CambioDeMarcha` creando un intervalo, al momento que el valor de velocidad dividido por la velocidad máxima cruce dicho parámetro se regrese el valor a 0 para seguir. Véase el gráfico 43.

Gráfico 43 Script tacómetro

```
private void Tacometro()
{
    float f = (1/(float) TotalCajaCambio);
    var FactorTransmision = Mathf.InverseLerp(f*m_NumeroCambio, f*(m_NumeroCambio + 1), Mathf.Abs(VelocidadActual/VelocidadMax));
}
}
```

Elaborado por: los autores

Ejemplo cuando la velocidad actual sea 7 km/h la función retornará un valor 0.32 en la función anterior CambioDeMarcha para que suba la marcha de primera a segunda la velocidad debe ser 23 km/h entonces cuando la velocidad llegue a 22 km/h la función devolverá un valor de 0.91 cuando llegue a 1 que es el momento en que se realiza el cambio de la marcha se reinicia a para volver a calcular el otro intervalo entre la marcha dos y la tres.

3.2.4.1.7 Función mover

Para que el vehículo se mueva se necesita configurar el Input de entrada del simulador, mediante la función Mathf.Clamp para asignar un rango de tipo flotante a la variable.

- **Movimiento Horizontal:** Se inicializa la variable MoVolante asignándole un rango mediante la función Mathf.Clamp ,Los datos se encuentran en un rango de -1 a 1, el cual se define que los valores de -1 a 0 son para el movimiento izquierdo y 0 a 1 para el movimiento derecho de los datos de entrada del volante.
- **Movimiento Vertical:** Se inicializa la variable aceleración asignándole un rango mediante la función Mathf.Clamp, con el rango de 0 a 1 y la variable frenado con el rango de -1 a 0.

Gráfico 44: Código fuente para el movimiento del vehículo

```
public void Mover(float MoVolante, float aceleracion, float freno, float frenoMano)
{
    MoVolante = Mathf.Clamp(steering, -1, 1);
    aceleracion = Mathf.Clamp(aceleeracion, 0, 1);
    freno = -1*Mathf.Clamp(freno, -1, 0);
    frenoMano = Mathf.Clamp(frenoMano, 0, 1);
    // se agregar el máximo giro de las llantas delanteras
    AnguloGiro = MoVolante * MaximoAnguloGiro;
    m_WheelColliders[0].steerAngle = AnguloGiro;
    m_WheelColliders[1].steerAngle = AnguloGiro;

    float EmpujeTorque = aceleracion * (FullTorque / 4f);
    for (int i = 0; i < 4; i++)
    {
        m_WheelColliders[i].motorTorque = EmpujeTorque; }

    for (int i = 0; i < 4; i++)
    {
        m_WheelColliders[i].brakeTorque = FuerzaFrenado * freno; }

    if (frenoMano > 0f)
    {
        m_WheelColliders[2].brakeTorque = frenoMano * FuerzaFrenoMano;
        m_WheelColliders[3].brakeTorque = frenoMano * FuerzaFrenoMano;
    }
}
```

Elaborado por: los autores

Torque del motor es la fuerza emitida por el motor para mover los neumáticos. La fórmula del torque es:

$$\text{Torque} = \text{Distancia} \times \text{Fuerza}$$

Generalmente el torque se mide en la unidad Newton por metro. El torque del motor influye directamente en la aceleración del vehículo. Unity nos brinda el torque de motor con todas sus características mediando la función propia de WheelCollider – motorTorque, como parámetro recibe la aceleración multiplicada por el torque del vehículo. Con esto el vehículo empezara a moverse, para aplicar frenado se utilizó una función propia de WheelCollider-brakeTorque que recibe como parámetro el freno y la fuerza de frenado, la fuerza de frenado entre más alta sea más fuerza se le aplica al freno por consecuencia frenara de manera rápida.

Ya que esta función hace el trabajo de aceleración y frenado del vehículo aquí se invoca las funciones nombradas anteriormente como Tacómetro, CambioDeMarcha, tipoVelocidad.

3.2.4.1.8 Función TipoVelocidad

Para determinar un límite de velocidad del vehículo se calcula la velocidad del objeto, se tiene una variable *VelocidadMaxima* pública la cual puede ser cambiada desde el mismo objeto.

Para obtener la velocidad del vehículo se utilizó la función `Rigidbody.velocity.magnitude` nos devuelve la velocidad del vehículo.

Debido a la que la función `RigidbodyVehiculo.velocity.magnitude`; devuelve un valor en m/s (metros sobre segundo) se realiza el siguiente cálculo para cambiar la velocidad a otra unidad:

- Para Kilómetros/Hora tenemos:

$$\frac{1m}{s} \times \frac{1km}{1000m} \times \frac{3600s}{1h} = 3.6 km/h$$

$$1 \frac{m}{s} = 3.6 Km/h$$

- Para Millas/Hora por hora tenemos:

$$\frac{1m}{s} \times \frac{1M}{1609.34m} \times \frac{3600s}{1h} = 2,2369 MPH$$

$$1 m/s = 2,2369 MPH$$

El valor que devuelve la función es multiplicado dependiendo de qué medida se este utilizando ya sea KPH o MPH. Véase el gráfico 45.

Gráfico 45: Código fuente para formatear la velocidad del vehículo

```
switch (TipoVelocidad)
{
    case TipoVelocidad.MPH:
        velocidad *= 2.23693629f;
        if (velocidad > VelocidadMaxima)
            RigidbodyVehiculo.velocity = (VelocidadMaxima / 2.23693629f) * RigidbodyVehiculo.velocity.normalized;
        break;
    case TipoVelocidad.KPH:
        velocidad *= 3.6f;
        if (velocidad > VelocidadMaxima)
            RigidbodyVehiculo.velocity = (VelocidadMaxima / 3.6f) * RigidbodyVehiculo.velocity.normalized;
        break;
}
```

Elaborado por: los autores

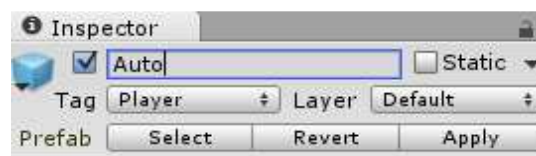
3.2.4.1.9 Función OnGui

Por medio de esta función se muestra el diseño del tacómetro, número de cambio y la velocidad. Diseño del tacómetro fue creado en Photoshop al igual que el puntero, los datos como número de cambio y la velocidad de las funciones habladas anteriormente.

3.2.4.2 Programación señales de tránsito

La propiedad tag es una etiqueta que se la vincula a uno o más objetos, el vehículo tiene como tag "Player", para identificarlo al momento que su collider colisione con otro collider. Véase el gráfico 46.

Gráfico 46: Asignación de tag "Player" al objeto Auto



Elaborado por: los autores

Cada señal de tránsito tiene diferente código pero se utilizan las mismas funciones.

3.2.4.2.1 Señal de tránsito Semáforo

Esta señal de tránsito está compuesta por el semáforo y 2 sensores uno para censar si el auto se pasó la luz roja y el otro para ver si invadió línea cebra. Véase el gráfico 47.

Gráfico 47 Señal de tránsito semáforo



Elaborado por: los autores

Para la señal de tránsito del semáforo se declaró una variable de tipo Light la cual viene directamente del semáforo, es la que indica si la luz del semáforo estaba en rojo. La variable trigger indica si el vehículo colisionó con el sensor. La variable CantidadLuzRoja indica cuantas veces el vehículo realizó esta infracción y finalmente la variable Mensaje que al momento de activarse muestra el mensaje de alerta durante 2 segundos por medio de la función OnGui ().

Gráfico 48: Variables declaradas para el semáforo

```
public Light rojo;
private bool trigger;
public bool PasarLuzroja=false;
public int CantidadLuzroja = 0;
bool Mensaje = false;
// mensaje
private Rect windowRect = new Rect((Screen.width - 800) / 2, (Screen.height - 400) / 2, 800, 50);
```

Elaborado por: los autores

3.2.4.2.2 Función OnTriggerEnter

Dispara una alerta al momento que el tag "Player" (vehículo) colisione cambiando el estado de la variable trigger a true. Véase el gráfico 49.

Gráfico 49: Código de la función OnTriggerEnter

```
void OnTriggerEnter(Collider otro)
{
    if (otro.gameObject.tag == "Player")
    {
        trigger = true;
    }
}
```

Elaborado por: los autores

3.2.4.2.3 Función Update

La función se pregunta si el trigger es positivo y si la luz roja está encendida entonces el conductor no respetó la señal de tránsito y se muestra el mensaje de alerta. El mensaje se lo activa a través de la función esperar (), debido que es una función que se ejecuta en un determinado periodo de tiempo se la invoca con la función StartCoroutine. Véase el gráfico 50.

Gráfico 50: Código de la función Update

```
// Update is called once per frame
void Update () {
    if (trigger && rojo.enabled == true)
    {
        PasarLuzroja = true;
        StartCoroutine(esperar());
    }
}
```

Elaborado por: los autores

3.2.4.2.4 Función OnGui

Permite enviar el mensaje por pantalla de la infracción cometida, con el mensaje “Te pasaste la roja”. Véase el gráfico 51.

Gráfico 51: Código del envío de mensajes por pantalla

```
void DialogWindow(int windowID)
{
    GUI.color = Color.red;
    GUI.Label(new Rect(5, 20, windowRect.width, 20), "<color=#ffffff><size=15><b><i>Te Pasaste la roja</i></b></size></color>");
}
void OnGUI()
{
    if (Mensaje == true)
    {
        GUI.color = Color.red;
        windowRect = GUI.Window(0, windowRect, DialogWindow, "Alerta");
    }
}
```

Elaborado por: los autores

3.2.4.2.5 Función OnTriggerExit

La función se invoca cuando el vehículo abandona el Collider del sensor, se pregunta si la variable PasarLuzRoja está encendida entonces la variable CantidadLuzRoja suma 1 por cada infracción cometida, la cual es utilizada al presentar los resultados de la simulación para indicar cuántas veces ha sido cometida la misma infracción. Véase el gráfico 52.

Gráfico 52: Código de la función OnTriggerExit

```
void OnTriggerExit(Collider otro)
{
    if (otro.gameObject.tag == "Player")
    {
        if (PasarLuzroja == true)
            CantidadLuzroja = CantidadLuzroja + 1;
        PasarLuzroja = false;
        trigger = false;
    }
}
```

Elaborado por: los autores

3.2.4.2.6 Script Control

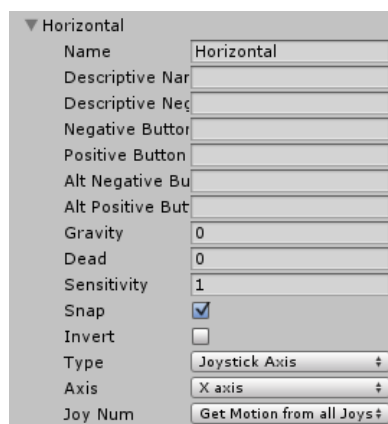
Se encarga de revisar las variables que llevan el control de la señal de tránsito por ejemplo en el caso de semáforo es CantidadLuzroja, una vez que se revisó todas las variables se genera un informe que presenta información como:

- Cantidad de veces que se hizo la infracción de la señal de tránsito.
- Puntos que se disminuyen en la licencia de conducir.
- Dinero que representa cada una de las infracciones.

3.2.5 Hardware

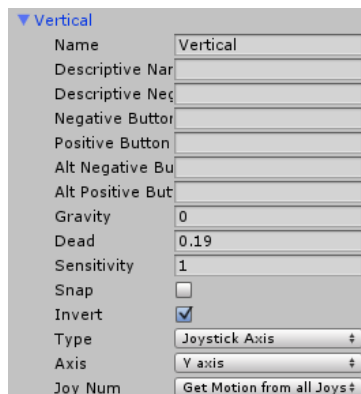
Para el proyecto el hardware necesario debe poder manejar dos tipos de entrada al simulador la cuales son el eje Horizontal y el eje Vertical.

Gráfico 53: Propiedades del eje horizontal



Elaborado por: los autores

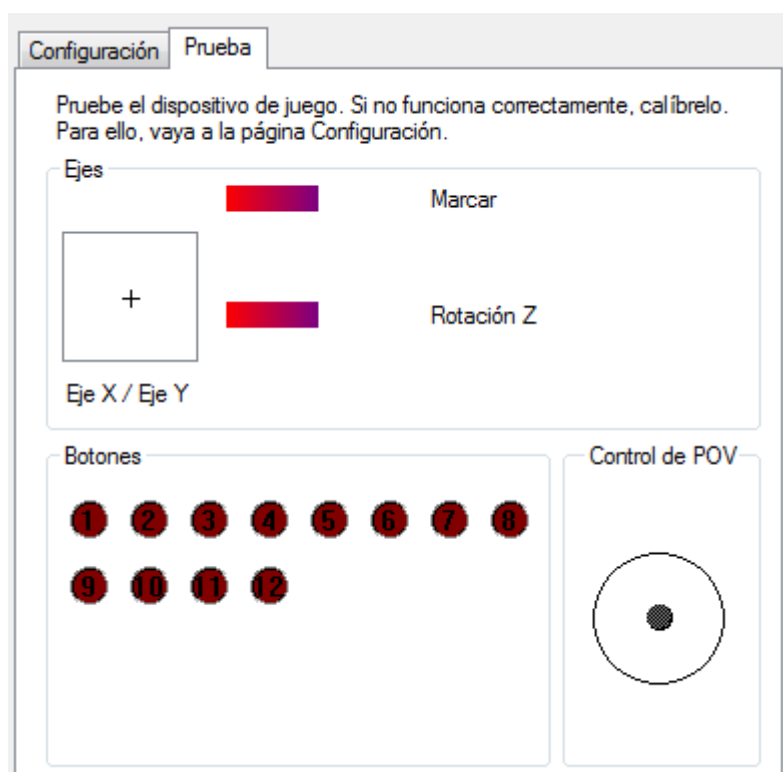
Gráfico 54: Propiedades del eje vertical



Elaborado por: los autores

Para saber si el hardware cumple con las características se realiza una verificación de la siguiente manera: Se ingresa a la opción configuración del dispositivo del juego -> propiedades.

Gráfico 55: Pantalla de configuración de JoyStick



Elaborado por el autor

Se mueve el volante hacia la derecha y se debe mover en el eje de la X en la parte positiva y al moverlo en sentido contrario debe estar en la parte negativa del eje de la X.

Para los pedales de aceleración y frenado estos se manejan en el eje Y, al presionar el acelerador el puntero se mueve en la parte positiva y al presionar el freno se mueve a la parte negativa.

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

Al finalizar este proyecto se concluye que el simulador de conducción vehicular cumple con los objetivos que fueron determinados:

- Se identificaron los requerimientos funcionales y no funcionales para el desarrollo del simulador a través de una entrevista a docentes la cual nos ayudó a determinar los requerimientos del motor de videojuegos a utilizar y la realización de una observación directa en parques viales de la ciudad la cual nos permitió determinar los requerimientos que debe cumplir el simulador de conducción vehicular.
- Se seleccionó la plataforma de software a utilizar a través de un análisis comparativo de diversas herramientas para la elaboración de videojuegos, utilizando los requerimientos funcionales y no funcionales obtenidos en el objetivo anterior.
- Se implementó el simulador de conducción vehicular cumpliendo los requerimientos funcionales y no funcionales determinados a través de las técnicas de información y se utilizó la plataforma de hardware y software seleccionada.

Recomendaciones

- Implementar nuevas señales de tránsito, en el simulador de conducción vehicular.
- Utilizar las características de Unity para exportar el software a diferentes plataformas como sistemas operativos móviles lo cual puede ser puesto en práctica en la materia de programación móvil.
- Adicionar a la implementación del simulador modelos 3D de edificios a través del software CityEngine.
- La persona que desee adicionar o modificar el simulador de conducción vehicular debe los enlaces externos proporcionados en la sección de recursos de conocimientos.

BIBLIOGRAFIA

(20 de 03 de 2015). Obtenido de GamemediAx: <http://gamemediAx.com/10-motores-graficos-con-los-que-hacer-tu-juego/>

(16 de 06 de 2016). Obtenido de Unity: <https://unity3d.com/es>

(16 de 06 de 2016). Obtenido de Unreal Engine: www.unrealengine.com

(16 de 06 de 2016). Obtenido de CryEngine: www.cryengine.com

Gonzalez, M. (2013). *Creacion de un dispositivo de simulacion que permita un analisis de flujo de caja de la empresa BJ Services de Venezuela C.C.P.A.*

Gonzalez, S. (2014). *Diseño y desarrollo de un videojuego educativo con tecnicas de inteligencia artificial para la plataforma Android aplicando la metodología OOHDM. Caso de estudio: Laberinto en 3D.* Sangolquí.

Ramos, A., Aquino, F., & Lara, F. (2013). *Aplicación de software de Modelado 3D y Edición de video, para la elaboracion de identidad corporativa, en pymes del sector servicio automotriz en el municipio de cuscatancingo.* Cuscatancingo.

Real academia española. (2014). *Diccionario de la lengua española (24.a ed.)*
Recuperado de: <http://dle.rae.es/?id=Xw14yph>.

Sudarmilah, E. (2013). *Tech Review: Game Platform for Upgrading Counting Ability on Preschool Children.*

Thorn, A. (2011). *Game Engine Design and Implementation.* Jones & Bartlett Learning.

Venega, Y. (2016). *Videojuegos serios como herramientas para la enseñanza.* La Habana, Cuba.

Videojuego de tipo Arcade para Dispositivos Móviles sobre motor Unity 3D (Tesis de grado, Universidad de Cantabria)2014



**Presidencia
de la República
del Ecuador**



**Plan Nacional
de Ciencia, Tecnología,
Innovación y Saberes**



SENESCYT
Secretaría Nacional de Educación Superior,
Ciencia, Tecnología e Innovación

DECLARACIÓN Y AUTORIZACIÓN

Nosotros, **Sarabia Lúa, Ginnio Andrés, con C.C: # 1204433344 y Guananga Bernabé, Jairo Alejandro con C.C: #0930975438**, autores del trabajo de titulación: **Diseño e implementación de un simulador de conducción vehicular utilizando un motor de videojuegos** previo a la obtención del título de **INGENIERO EN SISTEMAS COMPUTACIONALES** en la Universidad Católica de Santiago de Guayaquil.

1.- Declaramos tener pleno conocimiento de la obligación que tienen las instituciones de educación superior, de conformidad con el Artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del referido trabajo de titulación para que sea integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.

2.- Autorizamos a la SENESCYT a tener una copia del referido trabajo de titulación, con el propósito de generar un repositorio que democratice la información, respetando las políticas de propiedad intelectual vigentes.

Guayaquil, **22** de Septiembre de 2016

f. 

Sarabia Lúa, Ginnio Andrés

C.C: 1204433344

f. 

Guananga Bernabé, Jairo Alejandro

C.C: 0930975438

REPOSITORIO NACIONAL EN CIENCIA Y TECNOLOGÍA

FICHA DE REGISTRO DE TESIS/TRABAJO DE TITULACIÓN

TÍTULO Y SUBTÍTULO:	Diseño e implementación de un simulador de conducción vehicular utilizando un motor de videojuegos		
AUTOR(ES)	Sarabia Lúa, Ginnio Andrés ; Guananga Bernabé, Jairo Alejandro		
REVISOR(ES)/TUTOR(ES)	Ing. Salazar Tovar, Cesar Adriano, Mgs		
INSTITUCIÓN:	Universidad Católica de Santiago de Guayaquil		
FACULTAD:	Facultad de Ingeniería		
CARRERA:	Carrera de Ingeniería en Sistemas Computacionales		
TÍTULO OBTENIDO:	Ingeniero(a) en Sistemas Computacionales		
FECHA DE PUBLICACIÓN:		No. DE PÁGINAS:	103
ÁREAS TEMÁTICAS:	Hardware, Software, Redes y Comunicaciones		
PALABRAS CLAVES/ KEYWORDS:	Simulador, motor de videojuegos, videojuego, software		
RESUMEN/ABSTRACT (150-250 palabras):			
<p>La presente tesis tiene como objetivo el diseño e implementación de un simulador de conducción vehicular utilizando un motor de videojuegos. El tema fue propuesto debido a que en la facultad de ingeniería, carrera de Ingeniería en sistemas computacionales no existen hasta el momento proyectos de esta índole. El proceso de elaborar videojuegos requiere diversos conocimientos como diseño gráficos (Elaboración de texturas y modelos 3D) y programación, los motores de videojuegos son herramientas de programación utilizadas para la elaboración de proyectos de videojuegos. Con el desarrollo propuesto se espera que los docentes de la carrera sistemas computacionales del área de programación, utilicen el simulador como base para trabajos de tutoría.</p> <p>El simulador de conducción vehicular cuenta con señales de tránsito las cuales son validadas y penalizar al usuario en caso de cometer infracciones según las leyes ecuatorianas.</p>			
ADJUNTO PDF:	<input checked="" type="checkbox"/> SI	<input type="checkbox"/> NO	
CONTACTO CON AUTOR/ES:	Teléfono: +593-4-5126437 / 0991988711	E-mail: ginnio.sarabia@cu.ucsg.edu.ec / Jairo.guananga@cu.ucsg.edu.ec	
CONTACTO CON LA INSTITUCIÓN (COORDINADOR DEL PROCESO UTE)::	Nombre: Valencia Macias, Lorgia del Pilar		
	Teléfono: +593-4-2206950 ext 1020		
	E-mail: lorgia.valencia@cu.ucsg.edu.ec		
SECCIÓN PARA USO DE BIBLIOTECA			
Nº. DE REGISTRO (en base a datos):			
Nº. DE CLASIFICACIÓN:			
DIRECCIÓN URL (tesis en la web):			