Universidad de las Ciencias Informáticas
Vertex, Centro Entornos Interactivos 3D, Facultad 4

Componente para visualización de modelos 3D en la Web

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor: Sergio Antonio Quintana Gómez

Tutores: MSc. Rubén Alcolea Núñez

Ing. Sadiel Lázaro Caballero Ramos

Ing. Julio César Espronceda Pérez

La Habana, 2018

Nunca sabes qué tan fuerte eres hasta que ser fuerte,
es la única opción que tienes.
Robert Nesta Marley

Dedicatoria

Tengo muchísimos seres queridos, familiares, amigos y personas a las que admiro mucho que pudiera mencionar en esta dedicatoria. Sin embargo espero que sean capases de entender que no solo mi tesis, sino toda mi vida y mi que hacer como hombre de bien de nuestra sociedad se lo dedico solo a una persona. Quien ha sabido ser madre, padre y amiga. Quien se sabía todos mis escondites pero aun así, siempre me dejaba tocar la base primero. Con ese tono de voz tan alto y regaños a todas horas. Muchos no saben cómo podemos convivir bajo el mismo techo, pero solo nosotros sabemos lo mucho que nos amamos y las cosas que hemos pasado juntos. Estoy muy agradecido por la bendición que es tenerte hoy aquí a mi lado. Cuando te caíste aquella fatal noche fue mi nombre al primero que acudiste aun sabiendo que yo no estaba. Hoy, es tu nombre el único que cabe en estas líneas porque es a ti: Martha de la Caridad Fuentes Fuentes a quien dedico el fruto de estos 5 años. Te amo abuelita.

Agradecimientos

Quisiera agradecer a todas las personas que hicieron posible que este sueño se hiciera realidad y a todo el que contribuyó a que este este trabajo llegara a su conclusión. Así como a todo aquel que de una forma u otra me impulsó a desear ser ingeniero.

Especialmente a mis padres: *A mi madre por brindarme todo su amor desde el momento en que me tuvo en su vientre, por cuidar de mí y no estar ausente nunca cuando la necesite en los momentos felices y en los de tristeza. Por su esfuerzo en complacer siempre mis caprichos y ayudar a formar los conceptos y principios que hoy llevo conmigo. A mi padre por ser siempre mi modelo a seguir con sus defectos y virtudes. Por todos sus consejos y regaños. Por apoyarme siempre en mis decisiones y ayudar en todo momento a levantarme en momentos difíciles. Por ser el compañero y amigo que siempre necesité en las buenas, las malas y las peores. Por legarme el amor por la ingeniería y el baloncesto.*

A mi abuela: *A mi abuela Martha por brindarme su amor desde siempre, por todos los regaños y castigos que fortalecieron mi carácter, por hacerme el niño mimado de la casa, por la merienda, los zapatos abrochados, la mano al cruzar la calle y la camisa por dentro. Por ayudar a sentar las bases del joven que hoy soy.*

A mis hermanos: *A mi hermano Mayito quien junto a Mandy forjaron mi interés y luego amor por esta casa de altos estudios. Por ser el amigo que siempre necesité, por brindarme siempre su amor a pesar de su forma de pensar cuando nací. Por enseñarme que el respeto se gana con respeto y que no hay distancia que sea capaz de separar a dos hermanos. A Denis por ser un ejemplo de joven universitaria que supo crecerse ante muchas dificultades y forjar en mí ese espíritu de luchadora y triunfadora que ella posee. A Mandy por su cariño brindado a pesar de no llevar mis apellidos, por su preocupación, por sus consejos y tantas cosas compartidas en la vida, por ser parte de esa otra familia a la que amo mucho y contribuir a mi crianza junto a Mayita, Rodo y abuela Angelina.*

A mis amigos, estos hermanos que te regala la vida: *A Ana Isabel, tratando de ser breve le agradezco por enseñarme que el amor sincero y puro entre dos amigos de sexos opuestos existe, y que conservarlo así es uno de los retos más lindos que te impone la vida. Por ser fiel consejera, confidente y brindarme una madre como Deisy. Por mantener vivo nuestro amor a pesar de las distancias. A Andrés por su compañía en las buenas y en las malas, por ser fiel a los principios que forjaron nuestra amistad y saber manejar mi carácter como nadie. Por su apoyo incondicional y regalarme sus consejos y vivencias. Por saber guardar secretos y demostrarme que en estos tiempos difíciles que enfrentamos, los amigos de verdad existen. A Yordany por entrar a mi vida justo cuando necesitaba de alguien con tanta sabiduría, calma, sencillez, bondad y muchas otras virtudes que sería imposible pensar que caben dentro de una sola persona. Por tus consejos y camaradería. A Randel por sus locuras y ocurrencias, quien hizo la vida de toda mi familia un poco más divertida y agitada. Por apoyarme en mis decisiones y ser un fiel compañero.*

A mi novia: *Por dedicarme solo amor durante estos 5 años de carrera y apoyarme siempre. Por no permitir nunca que la distancia deteriorara lo que mucho nos costó tener. Por ser fiel compañera y amiga. Por darme ánimos y fuerzas para continuar después de cada tropiezo.*

A Aragón: *Por haber hecho tanto por mi familia y por mí, por ser como un hermano para mi padre y brindarme los medios posibles para hacer menos complicado el paso por esta universidad tan llena de tecnología.*

A LMT: *Muchas cosas pasaron durante estos 5 años de carrera y a estos hombres y amigos les debo mucho de lo que hoy soy, por lo que quisiera agradecer a Andro por ser mas que un amigo, un hermano con el cual conté sin dudar en esta escuela, a Pablo por ser un fiel amigo y compañero, por tanto apoyo y ayuda brindada, al Loren por ser el mejor de esta fraternidad, ejemplo de humildad y compañerismo, a Raiko por enseñarnos que el esfuerzo siempre trae buenos resultados, a Alvaro por su amistad incondicional y por tanto apoyo, preocupación y no permitirme nunca caer en malas decisiones. Esta fraternidad tiene una versión femenina por lo que quisiera agradecer a Lianet, Lidice, Jessica, Daylilis y Mirdolis por su amistad, por las cosas que juntos vivimos y por brindarme felicidad durante estos 5 años.*

A mis profesores: *Quisiera agradecer a todos los profesores que pusieron su granito de arena en mi preparación como profesional en especial a la profe Zaida por ser siempre como una madre para todos nosotros, por*

brindarnos toda su dedicación, sabiduría y amor. A Rubén y Julio por ser mas que mis tutores, mis padres dentro de esta escuela, nadie mejor que ellos sabe por lo que yo pase en esta carrera, por su apoyo brindado y consejos les agradeceré toda la vida. A Sadiel por comportarse como un hermano y no dejarme caer nunca ante las dificultades que se enfrenta un estudiante en la realización de su tesis.

A mis compañeros de carrera: *Mucho aprendí de mis compañeros durante estos 5 años de carrera y quisiera agradecer a todos los que alguna forma contribuyeron a mi formación como ingeniero y a los momentos felices en esta universidad.*

A mis compañeros de básquet: *A Roberto por su liderazgo, a Alex por su rectitud, al Lima por su dedicación, a Carrión y Daryl por su apoyo, a Yander por ser como es, a Leo, el Chiqui, Guido, Joel, Fernando, Wilvian y Pedro por su rivalidad y exigirme mejorar cada día para estar a la altura del baloncesto de esta universidad.*

A mis familiares: *A todos aquellos familiares que me apoyan desde las lejanas provincias de este país y son parte de este trabajo de diploma.*

Declaración de autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales sobre esta, con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

Sergio Antonio Quintana Gómez
Autor

MSc. Rubén Alcolea Núñez
Tutor

Ing. Sadiel Lázaro Caballero Ramos
Tutor

Ing. Julio César Espronceda Pérez
Tutor

Resumen

El centro Vertex tiene dentro de sus compromisos productivos el desarrollo de videojuegos. Algunos de los videojuegos que se han desarrollado son La Neurona (versiones 1 y 2), Aventuras en la manigua, Caos Numérico y Villa Tesoro. Durante la etapa de diseño de un videojuego, los diseñadores crean los modelos 3D para el videojuego. Con el objetivo de reutilizar estos diseños en futuros videojuegos, en el centro Vertex se está desarrollando un repositorio web para almacenar las mejores versiones de los modelos 3D que se diseñan en el centro. Actualmente, para visualizar un modelo 3D del repositorio, es necesario descargar el modelo del repositorio y abrirlo con un software profesional de diseño. Esto consume tiempo y retrasa el proceso de visualización de información importante de un modelo 3D.

Este trabajo propone un componente para visualizar los modelos 3D del repositorio web del centro. Este incluye funcionalidades para visualizar en modo mallado, controles para las cámaras de la escena, así como funciones para rotar y hacer zoom al modelo. Como resultado, los diseñadores podrán visualizar la mejor versión de un modelo 3D del repositorio y sin necesidad de descargar el modelo o instalar algún software adicional para visualizarlo.

Para la implementación del componente se utilizó la biblioteca **Three.js**. Como resultado, se obtuvo un componente de visualización de modelos 3D que se integró a la versión actual del repositorio web del centro Vertex.

Palabras clave: modelo 3D, repositorio, visualización, web.

Índice general

Índice de figuras

Índice de tablas

Introducción

La aparición y el desarrollo de Internet se consideran un hito trascendental en el desarrollo y la historia más reciente del hombre. Esta evolución constante en los últimos años supera con creces las previsiones de los expertos. Desde su surgimiento, Internet ha transitado a través de diferentes fases. A finales de los años 60, la comunicación entre dos computadoras se realizaba a través de una red de computadoras [1]. Posteriormente, a principios de los años 80, la aparición del protocolo TCP/IP posibilitó conectar un número mayor de computadores, representando un paso de avance significativo. Sin embargo, no es hasta finales de los años 80 y principios de los 90 en el que se inicia el uso comercial de Internet. Con la creación de la World Wide Web (WWW) en 1991, se incrementó aún más el uso de Internet y se estimuló su rápido crecimiento. Años más tarde, la producción de teléfonos más avanzados posibilitó la conexión de dispositivos móviles a Internet, dando inicio a la Internet móvil [2]. Sin embargo, la evolución de Internet y la creación de nuevas tecnologías dio paso a las redes sociales. Estas permiten a los usuarios conectarse a través de Internet para compartir contenidos, hacer amistades, establecer colaboraciones académicas, entre otras tareas colaborativas. Pero sin dudas, el paso culminante del desarrollo de Internet en la actualidad es Internet de las Cosas. Este revolucionario paradigma brinda la posibilidad de conectar a Internet objetos de la vida cotidiana (tales como los electrodomésticos) con el objetivo de comunicarse e intercambiar información a través de la red. Como resultado, es posible interactuar con dispositivos en una forma que hasta hace algunos años, podría parecer de ciencia ficción.

El surgimiento de Internet incluyó nuevos desarrollos tecnológicos así como la fusión de infraestructuras de redes y sistemas de telecomunicaciones existentes. En la década de 1980, tecnologías que se reconocerían más tarde como las bases de la moderna Internet, empezaron a expandirse por todo el mundo [3]. Un ejemplo de esto fue la web creada en 1989 por Tim Berners Lee. Esta consistía en una forma de organizar la información usando como medio físico de comunicación la red de Internet y el protocolo Hypertext Transference Protocol (HTTP, por sus siglas en inglés) utilizado en los navegadores para realizar peticiones a los servidores web y recibir respuestas de estos. Tim Berners Lee unió el Internet (la tecnología) y el protocolo HTTP y creó la web. De esta manera, todo el mundo podría conectarse y compartir información usando Internet [4].

En los años 90 se introdujo la World Wide Web, la que tenía como objetivo principal el desarrollo de estándares tecnológicos disponibles que garantizaran el crecimiento homogéneo de la web. Como consecuencia, la infraestructura de Internet se esparció por el mundo para crear la moderna red mundial de computadoras que existe hoy. Esta red atravesó los países occidentales e intentó una penetración en los

países en desarrollo, creando un acceso mundial a la información y las comunicaciones sin precedentes [3].

La web ha pasado por varias etapas: la Web 1.0 que se caracteriza por el contenido estático, la Web 2.0 caracterizada por el contenido dinámico o interactivo y la Web 3.0 caracterizada por el contenido colaborativo. Entre las tecnologías actuales que se pueden encontrar en la Web 3.0 está la Web 3D. Esta se refiere a la tecnología 3D en Internet y se usa de manera masiva en juegos, en paseos virtuales mundiales, ingeniería geoespacial, investigación de alta tecnología y redes sociales [4].

En la actualidad, los repositorios digitales son ejemplos de aplicaciones web muy utilizadas y de gran aceptación para numerosos usuarios en Internet. Los repositorios digitales contienen archivos digitales de productos científicos y académicos que pueden ser accedidos por los usuarios. Específicamente, los repositorios institucionales son plataformas web de servicios informáticos que conservan los recursos científicos y académicos (físicos o digitales) de las universidades a partir de la enumeración de un conjunto de datos específicos (metadatos). De esta forma, los recursos se pueden recopilar, catalogar, acceder, gestionar, difundir y preservar en el tiempo. La representación de estos recursos se logra mediante el registro persistente del conjunto de datos asociados a ellos. Estos datos sirven como síntesis y reemplazo del objeto "real", lo cual permite distribuir el recurso sin requerir del objeto en sí, sino usando su representación. Las actividades de catalogación, acceso, gestión y difusión de los contenidos son las más consolidadas con el crecimiento de los repositorios, por el contrario, la recopilación de materiales y la preservación todavía se encuentran en sus primeros pasos [5].

Un ejemplo de repositorio digital utilizado para visualizar y compartir contenido 3D es Sketchfab. La compañía responsable de este repositorio se creó en Francia y en la actualidad se encuentra localizada en París y Nueva York. Sketchfab ofrece entre sus servicios, un visualizador de modelos 3D de una alta calidad que se basa en tecnología WebGL. Este visualizador se utiliza en el sitio web de Sketchfab, pero también puede ser embebido en otros sitios web externos, como es el caso de Facebook [6]. Sin embargo, para poder utilizar el visualizador de Sketchfab o acceder a ciertos modelos de este repositorio, es necesario pagar una licencia, debido a que el software es propietario. Sketchfab ofrece además un portal comunitario, donde los visitantes pueden navegar, calificar y descargar algunos modelos 3D que son públicos. El visualizador 3D de Sketchfab utiliza la biblioteca WebGL JavaScript API para mostrar los modelos 3D en los navegadores que no soportan la tecnología WebGL. Con este propósito, el visualizador de Sketchfab emplea una secuencia de imágenes 2D a partir del objeto 3D pre-renderizado [7].

Uno de los objetivos planificados por el gobierno cubano en los Lineamientos de la Política Económica es la informatización de la sociedad cubana. Para cumplir esta meta, se han diseñado diversas estrategias que contribuyen al incremento gradual de la informatización en Cuba. La creación de la Universidad de las Ciencias Informáticas (UCI) en el año 2002 ha sido uno de los aciertos más grandes en aras de informatizar el país. La principal misión de la UCI es formar Ingenieros en Ciencias Informáticas y producir software que contribuyan a informatizar los procesos sustanciales de la sociedad cubana. Actualmente, la UCI posee varios centros de producción de software, en los cuales se desarrollan diferentes tipos de proyectos en diversas áreas de la sociedad. Entre las soluciones que se han desarrollado existen sistemas PACS, sistemas

RP, sistemas operativos basados en Software Libre, entre otros. Específicamente en el Centro de Entornos Interactivos 3D (Vertex) se han desarrollado diferentes proyectos que utilizan diseño 3D, animaciones y videojuegos. Como resultados, se han realizado propuestas de videojuegos, entre las que se pueden mencionar La Neurona (versiones 1 y 2), la Súper Claria, Caos Numérico y Villa Tesoro.

Durante la elaboración de un videojuego, el equipo de diseño genera una gran cantidad de modelos 3D. Por esta razón, en el centro Vertex se trabaja en la creación de un repositorio web para almacenar los modelos 3D de cada diseñador. Este repositorio permitirá a cada diseñador subir sus prototipos al repositorio o descargar otros ya existentes. En la versión actual del repositorio web, la selección de la mejor versión de un modelo 3D es muy lenta, debido a que no es posible visualizar de forma rápida y eficiente el modelo, ni conocer información importante como la cantidad de polígonos, los tipos de polígonos que componen el modelo o las animaciones que este contiene. Como resultado, si un diseñador desea visualizar o conocer información importante de un modelo 3D, tiene que descargarlo del repositorio web y posteriormente abrirlo con un software adicional como 3DStudio Max o algún visor especializado. Este proceso consume tiempo y retrasa la toma de decisiones en proyectos reales.

Teniendo en cuenta la situación problemática antes descrita, se presenta el siguiente **problema de investigación:** ¿Cómo visualizar los modelos 3D del repositorio web del centro Vertex? Se define como **objeto de estudio** la visualización de modelos 3D.

Para dar solución al problema de investigación planteado se define el siguiente **objetivo general**: desarrollar un componente para visualizar modelos 3D del repositorio web, sin tener que descargarlos o utilizar un software adicional. Se define como **campo de acción** la visualización de modelos 3D en la Web. Para dar cumplimiento al objetivo planteado, se definieron las siguientes tareas de investigación:

1. Elaboración del marco teórico a partir del estado del arte sobre las tecnologías para visualización en la web.
2. Selección de la tecnología más propicia para la visualización de modelos 3D en la web.
3. Selección de las bibliotecas necesarias para la carga de escenas gráficas y modelos 3D.
4. Diseño del sistema para tener una guía durante el proceso de desarrollo.
5. Implementación de un componente de software para visualizar en la web modelos 3D así como información importante contenida en estos modelos.
6. Validación de los resultados obtenidos mediante la realización de pruebas.
7. Integración del componente desarrollado en el repositorio web del centro.

Durante el desarrollo de la presente investigación, se utilizaron los siguientes métodos de la investigación científica:

- **Histórico-Lógico:** Método teórico utilizado para analizar la evolución y las tendencias actuales de la visualización de modelos 3D y sus aplicaciones.
- **Analítico-Sintético:** Método teórico utilizado para analizar y extraer información sobre la visualización 3D en la web.

- **Consulta de fuentes de información:** Método empírico utilizado para la consulta de fuentes bibliográficas durante el proceso de investigación.
- **Pruebas:** Método empírico utilizado para validar el componente desarrollado como solución.
- **Observación:** Método empírico utilizado para apreciar la calidad y el rendimiento de la visualización de los modelos 3D en el repositorio web.

La presente investigación se ha estructurado en tres capítulos. A continuación se muestra una breve síntesis de cada uno.

Capítulo 1: Fundamentación teórica

En este capítulo se elaborará el marco teórico de la investigación y se analizarán las principales bibliotecas y algoritmos que se utilizan en la visualización 3D en la web. Además, se explicarán algunos conceptos que pueden ser importantes para una mayor comprensión del objeto de estudio.

Capítulo 2: Propuesta de solución

En este capítulo se explicará el proceso de visualización 3D en la web seleccionado como solución propuesta. Además, se explicarán con mayor profundidad las ventajas del método que se propone y se describirán la metodología de desarrollo de software, así como las herramientas y los lenguajes utilizados.

Capítulo 3: Resultados y validación del sistema

En este capítulo se definirán las pruebas para validar los resultados de la investigación. Para esto se realizará un levantamiento de tareas de ingeniería por Historias de Usuario para cada una de las iteraciones definidas. Finalmente, se ejecutarán las pruebas de aceptación y se mostrarán sus resultados.

CAPÍTULO 1

Fundamentación teórica

En el presente capítulo se exponen los elementos teóricos fundamentales acerca de los modelos 3D, así como las tecnologías utilizadas para su creación y visualización. Además, se realiza un análisis de las principales aplicaciones informáticas para gráficos tridimensionales y tecnologías para la visualización 3D en la web. Se definen los lenguajes de programación y las herramientas a utilizar durante el desarrollo así como la metodología utilizada.

1.1. Repositorio

El Diccionario de la Real Academia (DRAE) define un repositorio como "Lugar donde se guarda algo" por otra parte, el término deriva del latín "repositorium" que significa armario o alacena. Partiendo de estas definiciones, se aplicó al léxico específico de la informática para designar a los repositorios de información digital. Un repositorio de información digital es un sistema de red formado por hardware, software, datos y procedimientos. Este sistema sirve para almacenar, conservar y dar acceso a documentos digitales [8].

Un repositorio es un espacio que se utiliza para almacenar distintas cosas. La idea de un repositorio puede asociarse al concepto de archivo o de depósito. En un repositorio, se guarda algo, que puede ser material (físico) o simbólico. En este sentido, actualmente se suele hacer referencia a las bases de datos digitales y a diversos sistemas informáticos como repositorios [9].

1.1.1. Repositorio Web

Un repositorio digital o web es un medio para gestionar, almacenar, preservar, difundir y facilitar el acceso a los objetos digitales que alberga [10].

- Los repositorios no son una forma de publicación
- La calidad de los contenidos no se evalúan dentro de los repositorios
- Se verifica la calidad de los metadatos
- No infringe las leyes de propiedad intelectual

- Son de acceso abierto
- Utilizan estándares abiertos

A continuación, se muestran ejemplos de algunos repositorios que se encuentran en la web:

- **OER Commons:** materiales y recursos educativos en línea accesibles para toda la comunidad de Internet.
- **Europeana:** permite explorar los recursos digitales de los museos, bibliotecas, archivos y colecciones audiovisuales de Europa.
- **Proyecto Gutemberg:** biblioteca de libros digitales.

1.1.2. Tipos de Repositorio Web

Existen diferentes tipos de Repositorios digitales. Estos pueden ser repositorios institucionales, temáticos o de datos. A continuación se explican con más detalles estos tipos de repositorios.

- Repositorios Institucionales
  Un Repositorio Institucional (RI) es un archivo en línea donde se depositan, en formato digital, materiales derivados de la producción científica o académica de una institución. Los repositorios institucionales se han convertido en la principal forma de publicar, preservar y difundir la información digital de las organizaciones, soportados en su mayoría por software libre. Entre los contenidos de los repositorios institucionales es posible encontrar tesis doctorales, artículos de carácter científico, ponencias o comunicaciones a congresos, revistas electrónicas editadas por la institución, materiales docentes, entre otros [11]. Un ejemplo de este tipo de repositorio, es el repositorio de la UCI. Este contiene los documentos de tesis que desarrollan los estudiantes para optar por el título de Ingeniero en Ciencias Informáticas. En este repositorio también se guardan los documentos de tesis de maestrías y doctorados que discuten los profesores para optar por los títulos antes mencionados. En la Figura 1.1 se muestra una imagen del repositorio institucional de la UCI.

Figura 1.1. Repositorio Institucional de la Universidad de las Ciencias Informáticas.

- Repositorios Disciplinares o Temáticos
  Existe un consenso general respecto a que los repositorios temáticos fueron los primeros repositorios en aparecer. Estos pueden describirse como proveedores de servicios que recolectan datos estructurados relacionados con un tema. De esta forma, el usuario puede buscar a través de muchas fuentes distribuidas en todo el mundo desde un punto de acceso único. Este tipo de repositorios se desarrolló en el ámbito de disciplinas académicas concretas, tales como la física, las ciencias de la información, las ciencias cognitivas, la salud, entre otras [8]. Un ejemplo de repositorio temático es el Internet Archive. Este proyecto proporciona el acceso a colecciones históricas existentes en formato digital, tal y como se muestra en la Figura 1.2.

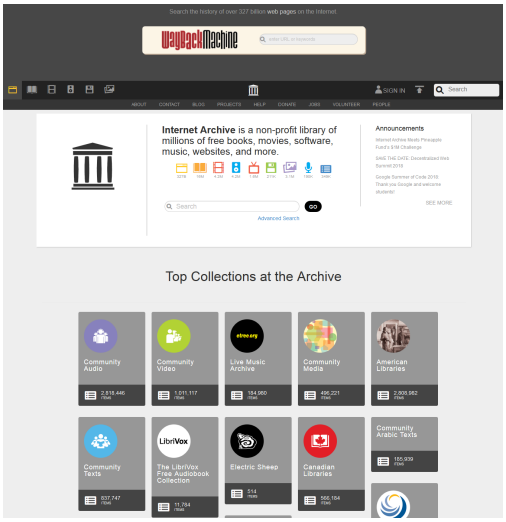


Figura 1.2. Internet Archive.

- Repositorios de Datos
  Los Repositorios de Datos de investigación son útiles para validar resultados de investigación. Estos deben vincularse de alguna manera a las publicaciones científicas donde fueron utilizados esos datos. Por esta razón, algunos de los problemas se pueden abordar de forma conjunta, tanto para los repositorios institucionales como para los repositorios de datos de investigación [12]. Algunos ejemplos de Repositorios de Datos Abiertos son **World Bank data**, **United Nations data** y **Fundación TIC datos**.

1.1.3. Repositorio web del centro Vertex

El repositorio web del centro Vertex que se encuentra en desarrollo, es capaz de almacenar los modelos diseñados por el equipo de desarrollo de videojuegos. De esta forma, cada diseñador del centro tendrá acceso a los modelos desarrollados por sus compañeros de equipo y otros diseñadores del centro. Este repositorio se desarrolló utilizando el framework Yii en su versión 2.0, y HTML5 para la interfaz visual. Una de las funcionalidades más importantes del repositorio es la de subir los modelos para su almacenamiento

y presentación a los usuarios. Esta funcionalidad ofrece a los diseñadores la posibilidad de descargar los modelos creados por otros diseñadores y usuarios con acceso al repositorio. Actualmente, para subir un modelo al repositorio, se debe comprimir en formato **.zip** el modelo 3D y todos los archivos que contienen información asociada a sus propiedades, tales como las texturas, el mapa de normales, el UV y la Oclusión Ambiental. Como resultado, se almacena en la misma dirección del repositorio, todo el contenido asociado a un modelo 3D.

El repositorio soporta en estos momentos dos tipos de modelos 3D: los modelos con extensión **.OBJ** y los **.FBX**. El objetivo final del presente trabajo es lograr visualizar los modelos 3D que se encuentran almacenados en el repositorio, directamente en la web mediante un visor integrado al repositorio. De esta forma, si un usuario desea observar un diseño almacenado en el repositorio pueda hacerlo desde la web y sin necesidad de descargarlo ni utilizar un software especializado en visualización 3D.

1.2. Modelos 3D

El modelado es el proceso de creación pura, ya sea esculpir, construir con bloques, trabajos de construcción, tallar, arquitectura o moldeado. Para crear un modelo 3D se siguen los mismos conceptos, más allá del software de diseño que se utilice. Para obtener el diseño final, se inicia con figuras básicas y se va aumentando la complejidad hasta que la unificación de las mismas conforma un objeto más trabajado y con mejor calidad visual. Las técnicas de modelado a usar dependen en gran medida del propósito del modelo. Una aplicación de los modelos 3D es su uso en videojuegos. Una gran mayoría de los juegos desarrollados hoy en día se hacen con 3DMax. Esto incluye juegos de PC, así como juegos de consola para Xbox, Playstation y Nintendo's Game Cube.

Un modelo 3D se puede obtener de diferentes formas. El modelado se realizar con objetos primitivos como bloques, cubos y esferas o se puede crear uniendo polígonos. Según la experiencia alcanzada, se podrá descubrir objetos que faciliten el modelado usando un método o utilizando otros objetos [13].

Otra etapa importante en el proceso de creación de modelos 3D es la representación o *rendering*. Este se encarga de la generación de una imagen bidimensional o tridimensional a partir de un modelo, mediante programas o aplicaciones. El *rendering* se utiliza principalmente en diseños arquitectónicos, videojuegos, películas animadas, simuladores, efectos especiales de TV y visualización de diseños. Las técnicas y características utilizadas varían según el proyecto. El *rendering* contribuye a aumentar la eficiencia y a reducir los costos en el diseño. Dicho de otra forma, durante el proceso de *rendering*, la computadora interpreta la escena en tres dimensiones y la plasma en una imagen bidimensional.

La eficiencia de diferentes motores de *rendering* se basa en una compensación entre velocidad y calidad. Por ejemplo, el *rendering* utilizado para visualizar objetos en las ventanas gráficas se optimiza para la velocidad, pero el que se utiliza para generar una imagen final, se inclina hacia la calidad. Con este objetivo, es posible establecer configuraciones en los motores para acelerar el proceso o mejorar la calidad de los resultados.

1.3. Visualizadores 3D

1.3.1. Visualización 3D en la Web

Para visualizar un modelo 3D en la web, primeramente se selecciona una biblioteca de visualización. Luego se define la extensión de archivos 3D con las que se va a trabajar, para determinar los *scripts* de carga necesarios. Una vez definidos los aspectos anteriores, se concretan las funcionalidades que se desean brindar en el visor, para así obtener un *script* de funcionalidades o proceder a su desarrollo teniendo en cuenta las funciones que ofrece la biblioteca de visualización seleccionada para las propiedades de los modelos que se van a visualizar.

1.3.2. Herramientas de visualización 3D en la Web

Sketchfab es una plataforma web que permite la gestión de modelos 3D en línea. Esta plataforma incluye un visualizador de modelos 3D basado en tecnología WebGL. Este se utiliza en la página web de Sketchfab, pero también puede ser embebido en otros sitios web externos, entre ellos Facebook. Sin embargo, al ser un producto privativo, es necesario pagar por utilizar el visor o por acceder a algunos modelos del repositorio que no son completamente libres. Sketchfab ofrece además un portal, donde los visitantes pueden navegar, calificar de acuerdo a cuanto gustó un modelo específico y descargar modelos 3D públicos. El visualizador 3D de Sketchfab utiliza WebGL JavaScript para mostrar los modelos 3D, en aquellos navegadores que no soportan la tecnología WebGL.

Otra tecnología analizada para visualizar objetos tridimensionales es Blend4Web. Este es un marco de código abierto para crear y mostrar gráficos interactivos 3D en navegadores web. El marco Blend4Web aprovecha Blender para editar escenas en 3D. La representación del contenido se basa en WebGL, Web Audio, WebVR y otros estándares web, sin el uso de complementos. Blend4Web tiene doble licencia y se distribuye bajo un código abierto GPLv3 y una licencia comercial. Una de las ventajas que tiene es que el código fuente está alojado en GitHub. Las aplicaciones web de Blend4Web se pueden implementar en sitios web de redes sociales como Facebook. La cadena de herramientas Blend4Web incluyen bibliotecas de JavaScript, el complemento de Blender y un conjunto de herramientas para ajustar parámetros de escenas 3D, depurar y optimizar. Blend4Web fue desarrollado en 2010 por la empresa Triumph, con sede en Moscú y fue lanzado al público el 28 de marzo de 2014.

Otra opción para crear modelos 3D con diferentes propósitos son las herramientas de diseño 3D. Estas en su mayoría, exportan a diferentes formatos como STL y OBJ, los prototipos creados en sus diferentes extensiones de acuerdo al software utilizado. Algunos ejemplos de estas herramientas son: Blender, 3DStudio, Ligthwave o Maya por solo mencionar algunas.

1.3.3. Bibliotecas para la carga de escenas gráficas y modelos 3D en la Web

Web Graphics Library (WebGL, por sus siglas en inglés) es una especificación estándar que define una Interfaz de Programación de Aplicaciones (API, por sus siglas en inglés) implementada en JavaScript para

la representación de gráficos en 3D en navegadores web [14]. Esta no precisa del uso de plugins adicionales en las plataformas que soportan OpenGL 2.0 u OpenGL ES 2.0. WebGL se puede integrar en los estándares web del navegador para acelerar el procesamiento de imágenes y efectos como parte del "canvas" de la página web, a través del hardware de la Unidad de Procesamiento Gráfico (GPU, por sus siglas en inglés). Los elementos de WebGL se pueden combinar con otros elementos HTML y otras partes de la página. Los programas WebGL consisten en un código de control escrito en JavaScript que permite usar la implementación nativa de OpenGL ES 2.0, un lenguaje similar a C o C++, que se ejecuta en la GPU. WebGL está diseñado y gestionado por el consorcio de tecnología sin ánimo de lucro Khronos Group.

Por otra parte, la biblioteca **Three.js** abstrae muchas de las características de WebGL, lo que facilita su uso para el programador [15]. Por ejemplo, con WebGL se pueden aprovechar todas las prestaciones de OpenGL Shading Language (GLSL, por sus siglas en inglés), como lenguaje de programación de *shaders*. Three.js es una biblioteca de alto nivel, y esto supone una menor complejidad que al utilizar WebGL sin ninguna capa de abstracción superior.

Three.js es una biblioteca ligera escrita en JavaScript para crear y mostrar gráficos en 3D animados en un navegador Web. Esta puede ser utilizada en combinación con el canvas de HTML5, SVG o WebGL. El código fuente está alojado en un repositorio en GitHub. Su uso se ha incrementado en los últimos años y se ha popularizado como una de las más importantes para la creación de animaciones en WebGL [16]. A continuación se muestran algunas de las ventajas más importantes de Three.js:

- Facilita la realización de gráficos 3D simples con una alta calidad.
- Proporciona un acceso a bajo nivel a las capacidades de renderizado como proyección y animación.
- Soporta controles avanzados de *rendering*, como el postprocesamiento, *rendering* multipaso y representación diferida, proporcionando así la obtención de efectos con un algo nivel de realismo.
- Carga instantánea e integración flexible en el código de un sitio web existente.
- Se integra de forma fácil con otras bibliotecas de JavaScript.
- Cuenta con una amplia colección de ejemplos que cubren casi todos los aspectos relacionados con la visualización de modelos 3D.

Three.js es una biblioteca relativamente joven. En consecuencia, tiene algunas desventajas, entre las que se puedne mencionar:

- Cuenta con un desarrollo joven y tiene poca documentación y referencias informativas.
- El cambio frecuente del núcleo, provoca la desactualización de muchas soluciones.
- Los rendimientos gráficos dependen en gran medida de los navegadores y las potencialidades de los móviles.
- No es compatible con la experiencia de realidad virtual.

Threejs soporta animaciones en tiempo real y gráficos de datos. Esto hace de la biblioteca una opción adecuada para crear efectos visuales interactivos para proyectos compatibles con otras bibliotecas web. Sin

embargo, no está diseñada para proyectos que requieren interacciones complejas o que requieren un alto procesamiento gráfico. Es posible afirmar, que Three.js todavía está en una etapa incipiente, donde la escasa documentación y las actualizaciones pueden causar problemas de compatibilidad en las aplicaciones.

1.4. Formatos de archivos 3D

Los software de diseño exportan los modelos 3D generados en diferentes formatos. Cada formato tiene ciertas especificaciones que definen un estándar para cada formato. Esto propicia que a nivel de la comunidad de diseñadores, exista cierta compatibilidad en cuanto al contenido y la información que se almacena en archivos del mismo formato. La selección de un formato u otro depende del objetivo final para el que fue creado el modelo. En los siguientes epígrafes se abordarán los formatos **OBJ** y **FBX**, que son los formatos que soporta actualmente el repositorio web del centro Vertex.

1.4.1. Formato OBJ

El formato OBJ es un formato de definición de geometría desarrollado por primera vez por la empresa Wavefront Technologies para su paquete de animación Advanced Visualizer. Este formato de archivo es abierto y ha sido adoptado por otros proveedores de aplicaciones de gráficos 3D.

El formato de archivo OBJ es bastante simple, debido a que solo representa la geometría 3D. Entre las propiedades que se guardan en este formato se encuentran la posición de cada vértice, la posición de cada coordenada UV para las texturas, las normales de los vértices y las caras que definen el modelo. Los vértices se almacenan en orden antihorario. De esta forma, no es necesario declarar explícitamente la información de las normales.

1.4.2. Formato FBX

El formato FBX es un formato de archivo propietario que fue desarrollado originalmente por la empresa Kaydara y que fue adquirido por Autodesk en el 2006. Este formato utiliza un modelo basado en objetos, lo que permite el almacenamiento de movimiento, junto con datos en 2D, 3D, audio y video. Este tipo de archivo tiene mayor compatibilidad con otros paquetes de software 3D como Cinema 4D, SoftImage 3D, PowerAnimator, LightWave 3D y 3D Studio MAX. En la actualidad es uno de los principales formatos de intercambio 3D utilizado por muchas herramientas. El formato FBX tiene dos versiones: una basada en texto (ascii) y otra versión binaria.

1.4.3. Selección de las tecnologías para la visualización 3D en la Web

Three.js se basa en Javascript y aprovecha igualmente las capacidades WebGL de los navegadores para visualizar modelos 3D en la web. Una de las ventajas de Three.js es el uso directo de formatos de archivo como OBJ que se utilizan con bastante frecuencia. Por esta razón, no es necesario ninguna preparación

previa del modelo, salvo la simplificación del modelo con el fin de reducir el tamaño de los documentos lo suficiente, para ser cargados en la web en un tiempo razonable.

Three.js es un proyecto de código abierto liberado por Ricardo Cabello en 2010. En la actualidad dicho proyecto posee un fuerte desarrollo y ha sido escalado con otras bibliotecas que aportan funcionalidades necesarias para la visualización. Además, abstrae al programador de utilizar la biblioteca WebGL a bajo nivel, debido a que incorpora una serie de funciones que agilizan el proceso de desarrollo. Por estas razones, se seleccionó Three.js como biblioteca para la visualización de modelos 3D. Esta se encarga de generar el *rendering* para visualizar el modelo directamente en el repositorio web.

1.5. Lenguajes de Programación

El presente trabajo propone un componente para visualizar modelos 3D que serán almacenados en el repositorio web. Un objetivo del presente trabajo es lograr la integración del componente que se desarrollará con la versión actual del repositorio web. Para lograr este objetivo, es necesario garantizar la compatibilidad del componente propuesto como solución con el repositorio web que se encuentra en desarrollo. Por esta razón, una vez analizadas las tecnologías con que se desarrolla actualmente el repositorio, se decide utilizar las herramientas y tecnologías que están definidas para el desarrollo del repositorio. Como lenguaje de programación del lado del servidor se utilizará PHP. Como lenguaje de programación del lado del cliente se utilizará JavaScript. Además se utilizará CSS para modificar la apariencia visual de los elementos gráficos del componente de visualización y HTML para definir el contenido que tendrá el componente de visualización.

1.6. Metodología para el desarrollo de software

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos de software. Estas metodologías se aplican con el objetivo de asegurar la eficiencia en el proceso de producción de software. En la Figura 1.3 se muestra una comparación entre las Metodologías Tradicionales y las Metodologías Ágiles.

Metodologías tradicionales	Metodologías ágiles
Predictivos	Adaptativos
Orientados a procesos	Orientados a personas
Proceso rígido	Proceso flexible
Se concibe como un proyecto	Un proyecto es subdividido en varios proyectos más pequeños
Poca comunicación con el cliente	Comunicación constante con el cliente
Entrega de software al finalizar el desarrollo	Entregas constantes de software
Documentación extensa	Poca documentación

Figura 1.3. Comparación entre metodologías tradicionales y ágiles.

Una vez analizada la comparación anterior, se decide utilizar una metodología ágil para la presente investigación. Existen dos razones principales que justifican esta selección. En primer lugar, la flexibilidad de la metodología, lo que posibilita su adaptación de acuerdo a la realidad de cada equipo y proyecto. En segundo lugar, la poca documentación que genera. Este elemento es fundamental para un equipo de desarrollo de una persona, donde esta debe hacer todo el flujo de trabajo ingenieril, además del desarrollo del componente de visualización propuesto. La mayoría de los equipos ágiles exitosos han adaptado prácticas ágiles de distintas metodologías para generar un proceso de desarrollo propio que se ajusta a sus necesidades.

La metodología ágil XP es la más conocida [17]. Esta fue desarrollada por Kent Beck, con el objetivo de guiar equipos de desarrollo de software pequeños o medianos, entre dos y diez desarrolladores, en ambientes de requerimientos imprecisos o cambiantes. XP tiene como base cinco valores: simplicidad, comunicación, retroalimentación, respeto y coraje. Estos valores, a su vez, son la base para la definición de sus principios. De ellos, los fundamentales son: la retroalimentación rápida, asumir simplicidad, el cambio incremental, la aceptación del cambio y el trabajo de calidad [18].

Como resultado el análisis realizado sobre las metodologías y la profundización sobre XP se decide que para el desarrollo de este proyecto se utilizará una versión reducida de la metodología ágil XP la cual contará de 4 fases o etapas: planificación del proyecto, diseño, codificación y pruebas.

Fase I Planificación del proyecto: El primer paso de cualquier proyecto que utiliza la metodología XP es definir las historias de usuario con el cliente.

Fase II Diseño: La metodología XP sugiere los diseños simples y sencillos. Hay que procurar hacerlo todo lo menos complicado posible para lograr un diseño fácil de entender e implementar. Esto en el futuro puede reducir el esfuerzo y el tiempo del desarrollo.

Fase III Codificación: El cliente es el responsable de describir detalladamente cada historia de usuario antes de comenzar el desarrollo. Además, debe estar presente durante la realización de las pruebas, con el

objetivo de verificar que la historia implementada cumple con la funcionalidad especificada. La codificación del proyecto debe respetar un estándar de codificación. Esto se considera una buena práctica que mantiene el código consistente y facilita su comprensión y escalabilidad.

Fase IV Pruebas: Uno de los pilares de la metodología XP es el uso de pruebas para comprobar el funcionamiento del código a medida que se va implementando.

1.7. Herramientas

En los siguientes epígrafes se exponen las herramientas que se emplearán en el desarrollo del proyecto, así como sus principales características y funciones.

1.7.1. Visual Paradigm:

Visual Paradigm es una herramienta que propicia un conjunto de funcionalidades para el desarrollo de programas informáticos. Con Visual Paradigm es posible generar artefactos ingenieriles que se crean en etapas como la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. Esta herramienta ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo de software a través de la representación de todo tipo de diagramas. Aunque es una herramienta privada disponible en varias ediciones, existe una alternativa libre y gratuita de este software: la versión Visual Paradigm UML 6.4 Community Edition.

1.7.2. PHP Storm:

PhpStorm es un entorno de desarrollo apropiado para trabajar con otros frameworks de trabajo, como pueden ser Symfony, Drupal, WordPress, Zend Framework, Laravel, Joomla!, Yii, entre otros. Como entorno de desarrollo, PHP PhpStorm es ligero y se centra en la productividad del desarrollador. El editor de PhpStorm soporta todas las características del lenguaje PHP para proyectos modernos y legados. Además, aprovecha al máximo las avanzadas tecnologías de front-end, como HTML5, CSS, Sass, Less, Stylus, CoffeeScript, TypeScript, Emmet y JavaScript, con refactorizaciones, depuración y pruebas de unidades disponibles.

1.7.3. XAMPP:

XAMPP es un servidor web de plataforma software libre, que consiste principalmente en el sistema de gestión de bases de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script PHP y Perl. El nombre es en realidad un acrónimo: X (para los diferentes sistemas operativos), Apache, MariaDB/MySQL, PHP, Perl. A partir de la versión 5.6.15, XAMPP cambió la base de datos MySQL por MariaDB, un *fork* de MySQL con licencia GPL. El programa se distribuye con la licencia GNU y actúa como

un servidor web libre, fácil de usar y capaz de interpretar páginas dinámicas. En la actualidad, XAMPP está disponible para Microsoft Windows, GNU/Linux, Solaris y Mac OS X.

Consideraciones parciales del capítulo

Una vez concluido el proceso de investigación, es posible definir aspectos concretos de la solución. Se define como propuesta de solución, la elaboración de un componente de visualización de modelos 3D para integrarse en el repositorio web del centro Vertex. Esto brinda la posibilidad de visualizar los modelos 3D del repositorio sin necesidad de descargarlos o utilizar algún software adicional para su visualización.

Además, se define la plataforma Sketchfab como sistema homólogo para definir los requisitos funcionales del componente. La mayoría de las funcionalidades que están presentes en la plataforma Sketchfab, están acordes a la información y estilos de visualización que se desean incorporar como solución. Sin embargo, esta no puede utilizarse por ser privativa. Por esta razón, se propone la elaboración de un nuevo componente de visualización.

Como biblioteca de visualización 3D se utilizará Three.js. Esta incluye un grupo de funcionalidades que la hacen apropiada para implementar la carga de modelos 3D que soporta actualmente el repositorio web y la visualización de los mismos.

CAPÍTULO 2

Propuesta de solución

En el presente capítulo se expone la propuesta de solución dada al problema de investigación del presente trabajo. Durante el desarrollo de este capítulo se describe la arquitectura del sistema y las historias de usuarios para cada una de las funcionalidades del software. Además, se crea el plan de duración del proyecto y se definen las iteraciones de cada historia de usuario. Finalmente, se explican los formatos de archivos 3D que soporta el visor implementado, el diagrama de clases así como la integración del componente de visualización propuesto con el repositorio web del centro Vertex.

2.1. Solución Técnica

Como solución al problema de investigación planteado, se propone el desarrollo de un componente para visualizar los modelos 3D que se encuentran en el repositorio, directamente desde la web. El usuario debe seleccionar un modelo para poder interactuar con este o visualizar información importante en el visor. Una vez seleccionado el modelo, el usuario puede modificar las opciones de visualización a través de un menú que contiene los siguientes requisitos funcionales:

- Mostrar el modelo en su vista de mallado.
- Mostrar el texturizado.
- Mostrar el mapeado de texturas UV.
- Mostrar la normal de los vértices del modelo.
- Rotar el modelo.
- Hacer zoom al modelo.

Requisitos no funcionales

Los requisitos no funcionales son requisitos que imponen restricciones en el diseño o la implementación como restricciones en el diseño o estándares de calidad. Son propiedades o cualidades que el producto debe tener [19]. En este caso el componente propuesto adopta algunas de las tecnologías y herramientas que propone el repositorio web en desarrollo al que se desea integrar una vez culminada su implementación.

Requerimientos de Software

- **Servidor:** Sistema Operativo Windows 8 o superior, Navegador Web Firefox 57 o superior, servidor web Apache 2, lenguaje de programación PHP 5.3 o superior y gestor de base de datos MySQL 4.0 o superior. Se recomienda la utilización del paquete de programas XAMPP para Sistema Operativo Windows ya que es multiplataforma e incluye Apache, PHP y MySQL por defecto.
- **Cliente:** Sistema Operativo Windows 8 o superior, Navegador Web Firefox 57 o superior.

Requerimientos de Hardware

- **Servidor:** Procesador Intel Core i3 a 3.40GHz, 4Gb de RAM, 1Tb de disco duro.
- **Cliente:** Procesador Intel Core i3 a 3.40GHz, 4Gb de RAM, 100Gb de disco duro.

Requerimientos de Diseño e Implementación

Se utiliza PHPStorm 2017.3.1 como entorno de desarrollo, XAMPP como servidor web, MySQL como sistema gestor de base de datos.

Para una mejor comprensión del funcionamiento del componente se realizó el diseño que se muestra en la Figura 2.1:

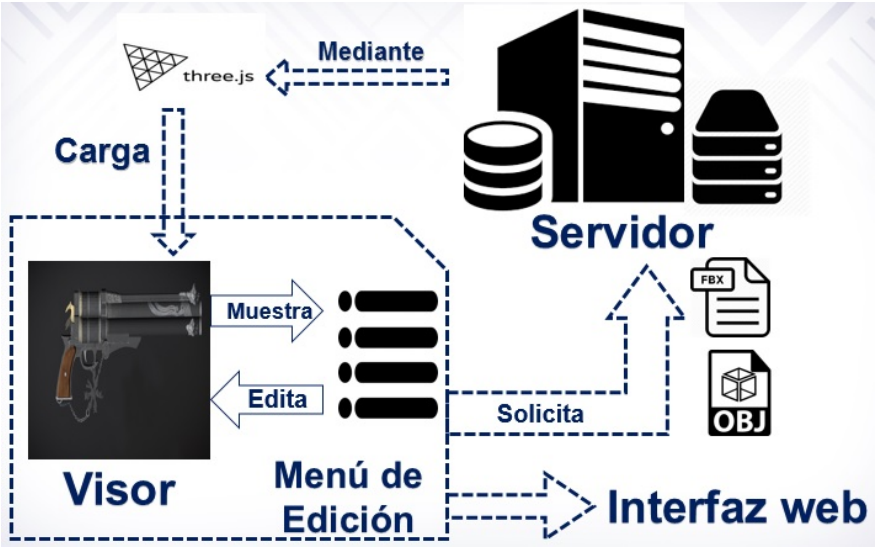


Figura 2.1. Arquitectura del sistema.

Este diseño muestra los elementos que componen el visor. Estos son la interfaz web, el servidor y la biblioteca de visualización en la web Three.js. El componente de visualización funciona de la siguiente forma. La interfaz web incluye dos áreas fundamentales denominadas visor y menú de edición. El visor permite mostrar el modelo 3D seleccionado y el menú de edición posibilita modificar las opciones de visualización del modelo en el visor. Para lograr esto primeramente la interfaz web solicita al servidor el modelo que

se desea mostrar en el visor. El servidor busca en la base de datos de modelos 3D el modelo solicitado de acuerdo a su ID y lo envía a la interfaz web. Finalmente, se visualiza el modelo 3D en el navegador mediante la biblioteca Three.js.

Todos los modelos no se muestran de la misma forma, aunque siempre se utilice la misma biblioteca de visualización. Esto se debe a que estos modelos poseen extensiones distintas de acuerdo con el software de diseño con el que fueron creados o con las necesidades de su diseñador en el momento de exportar su prototipo. Para estos casos entonces existen distintos *scripts* que pertenecen a la propia biblioteca pero tienen un tratamiento diferenciado para cada extensión de archivos. Actualmente, el repositorio soporta modelos de extensión OBJ y FBX por lo que es necesario la utilización de los *scripts* **OBJLoader.js** y **FBXLoader.js** respectivamente. Estos contienen un grupo de características de sus respectivas extensiones como son la geometría, el material, las texturas, los colores, las animaciones así como el modo en el que se cargan archivos de sus respectivas extensiones. Para esto primeramente la interfaz web solicita al servidor el modelo y le envía a su vez la extensión del mismo para que el servidor devuelva el *script* necesario para cargar el modelo en el visor. Para esto existe la función **switch('modeltype')**, la cual devuelve la extensión del modelo que está seleccionado. De esta forma, el servidor sabe cuál es el *script* que debe devolver para cargar el modelo. Una vez seleccionado el modelo y el *script* de carga, el visor muestra el modelo 3D a partir de la biblioteca de visualización 3D Three.js.

El menú de edición también posee funcionalidades propias de acuerdo al tipo de archivo cargado en el visor según las características e información que se desean mostrar. En este caso, el visor selecciona el tipo de menú para cada tipo de archivo, identificándolos de igual forma que los *scripts* mediante la función '**modeltype**'. Si el modelo que se visualiza tiene extensión OBJ, se muestra el menú para archivos OBJ y si es FBX, entonces se muestra el menú diseñado para este nuevo formato.

Como se explicó anteriormente, cada menú tiene funcionalidades distintas. Al seleccionar un modelo OBJ, el menú de edición muestra las secciones **Render**, **Wireframe**,**Material Channels**, **Geometry** y **UV**. Estas secciones se abordan con más detalles a continuación.

La sección **Render** incluye la función **Valores por defecto**:

- **Valores por defecto:** Muestra la visualización final del modelo 3D. Esta función permite restaurar el estado de la visualización del modelo 3D después de modificar parámetros de visualización. De esta forma, el visor es capaz de recuperar la visualización inicial del modelo.

La sección **Wireframe** visualiza la geometría del modelo en modo *wireframe*. El valor predeterminado es falso (polígonos planos). Además se pueden encontrar opciones como:

- **White button:** Permite ver el enmallado del modelo de color blanco.
- **Black button:** Permite ver el enmallado del modelo de color negro.
- **Aleatory:** Permite ver el enmallado en un color aleatorio.
- **Wireframe Sólido:** Muestra el modelo en un sólido de color rojo y con el enmallado de color blanco de forma que se pueda observar bien los polígonos que conforman el modelo sin depender de la textura y así poder obtener una mejor experiencia visual de esta propiedad.

- **WireOnly:** Permite ver el mallado del modelo sin un sólido ni texturas de fondo.
- **Desactivar:** Esta opción permite desactivar cualquier funcionalidad que se encuentre en visualización de la sección Wireframe.

La sección **Material Channels** contiene las opciones para trabajar con el material del modelo:

- **Normal:** Crea una textura con un mapa de normales del modelo 3D. Los valores RGB afectan la superficie normal de cada fragmento de píxel y cambian la forma en que se ilumina el color. Los mapas de normales no cambian la forma real de la superficie, solo la iluminación.
- **Glossy:** Esta función trabaja con el suavizado del modelo de forma tal que lo activa o lo desactiva y de esta forma se puede apreciar la terminación del modelo. Esta función está desactivada por defecto.
- **Ambient Occlusion:** La oclusión ambiental es un gran efecto que proporciona una iluminación más realista a los objetos en una escena en 3D. Funciona al observar la geometría de la escena para ver lo que está cerca: pisos, paredes u otra geometría. En el mundo real, el efecto que se observa cuando los objetos están cerca uno del otro es un sombreado sutil. Este es el efecto que crea la oclusión ambiental de manera muy convincente.

La sección **Geometry** es una alternativa fácil de usar para BufferGeometry. Las geometrías almacenan atributos (posiciones de vértices, caras, colores, etc.) utilizando objetos como **vector** o **color** que son más fáciles de leer y editar, pero menos eficientes que las matrices. La geometría está formada por una colección de vértices y caras que combinan tres vértices en una cara triangular. Con Three.js es posible crear una geometría personalizada definiendo los vértices y las caras, pero también existen formas de acceder y establecer nuevas propiedades personalizadas. En esta sección se encuentran las siguientes opciones:

- **Vertex Normal:** Representa flechas para visualizar los vectores normales de un vértice de un objeto. Esta opción requiere que las normales se hayan especificado en un atributo personalizado o se hayan calculado utilizando la función **computeVertexNormals**. A diferencia de **FaceNormalsHelper**, esta funciona con BufferGeometry.
- **Largo:** Permite agrandar el tamaño de las normales mostradas en la funcionalidad anterior a un valor por defecto.

La sección **UV** contiene la opción **UV Checker**:

- **UV Checker**: La texturización UV permite que los polígonos que componen un objeto 3D se visualicen con color (y otros atributos de superficie) a partir de una imagen ordinaria. Esta imagen se denomina mapa de textura UV. El proceso de mapeo UV implica la asignación de píxeles en la imagen a las asignaciones de superficie en el polígono, generalmente haciendo una copia de una pieza triangular del mapa de imagen y pegándola en un triángulo en el objeto. UV es una alternativa al mapeo de proyección (por ejemplo, usando cualquier par de las coordenadas X, Y, Z del modelo, o cualquier transformación de la posición); solo se asigna un espacio de textura en lugar del espacio geométrico del objeto. Pero el cálculo de representación utiliza las coordenadas de textura UV para determinar cómo pintar la superficie tridimensional.

En caso de seleccionarse un modelo de extensión .FBX, entonces el menú muestra la opción **Skeleton**. Esta opción es muy importante para este tipo de modelos:

- **Skeleton:** Muestra la estructura esquelética del modelo.

Para incorporar funcionalidades necesarias en cualquier visor se utilizan otras bibliotecas que se incluyen en Three.js. Por ejemplo, la biblioteca **TrackballControls.js** permite controlar las cámaras de la escena. Esta permite rotar, mover, acercar o alejar el modelo con el fin de visualizarlo con mayor nivel de detalle. La biblioteca **Detector.js** se utiliza para verificar si el navegador web tiene soporte para webGL.

2.2. Diagramas de clases

La solución propuesta está compuesta por cinco clases. Estas se pueden observar en el diagrama de clases que se muestra en la Figura 2.2. La clase principal es **ModelView**, esta contiene llamadas a las demás clases asociadas a ella y las principales funcionalidades del componente, tales como **Glossy**, **VertexNormal**, **Wireframe** entre otras que se observan en la imagen. La clase **TrackballControls** controla el movimiento de las cámaras de la escena con sus métodos, como por ejemplo **rotateCamera()** para la rotación, **zoomCamera()** para aumentar o disminuir el tamaño de la visualización entre otros. La clase **Detector** valida que el navegador en el que se está trabajando tiene soporte para WebGL, enviando un mensaje de error en caso de no ser así, el cual se puede recibir mediante el método **getWebGLErrorMessage()**. Por otra parte, las clases **OBJLoader** y **FBXLoader** se utilizan para cargar los modelos que se encuentran en el repositorio, teniendo en cuenta su extensión, ya sea OBJ o FBX. Estas clases cuentan con sus métodos principales THREEOBJLoader() y THREEFBXLoader() para la carga de modelos .OBJ y .FBX respectivamente. Además contienen métodos para trabajar con otras propiedades tales como **geometryMergeVertices()** y **parseSkeleton()**, que se muestran en el diagrama de clases.

2.3. Integración del visor con el Repositorio web

Para integrar la biblioteca Three.js con el repositorio web de modelos 3D es necesario primeramente agregar una lista de archivos javascripts a la clase ModelView donde se encuentra el código del visor 3d. Los archivos que se deben adicionar son los siguientes: Three.js, OBJLoader.js, MTLLoader.js, OrbitControls.js, Detector.js, FBXLoader.js y SkeletonHelper.js. Estos ficheros permiten reconocer los modelos 3d en el visor web, utilizar sus propiedades geométricas y modificar sus materiales y texturas para cumplir con las funcionalidades del componente. Para una mejor comprensión, se muestra la Figura 2.3.

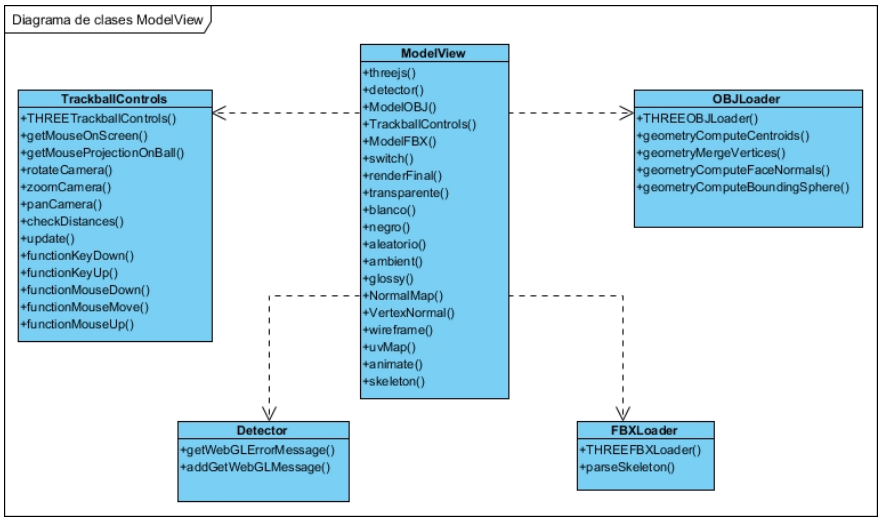


Figura 2.2. Diagrama de clases de la solución propuesta.

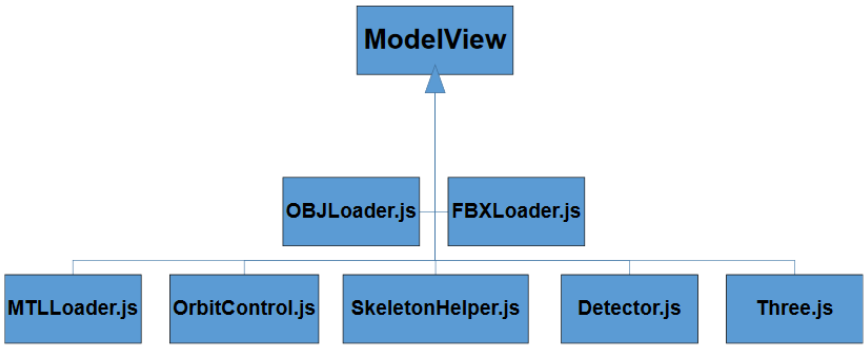


Figura 2.3. Integración del visor con el repositorio web.

2.4. Fase I Planificación del Proyecto

2.4.1. Historias de Usuarios

Las Historias de Usuarios (HU) tienen el mismo propósito de los casos de uso y son similares al empleo de escenarios, con la excepción de que no se limitan a la descripción de la interfaz de usuario. Las HU conducen al proceso de construcción de las pruebas de aceptación (empleados para verificar que las HU se implementaron correctamente). En la etapa de planificación del proyecto se definieron 15 HU, las cuales se muestran a continuación.

Tabla 2.1. Historia de usuario # 1

Historia de usuario	
Número: 1	**Nombre:** Mostrar archivo

Continúa en la próxima página

Tabla 2.1. Continuación de la página anterior

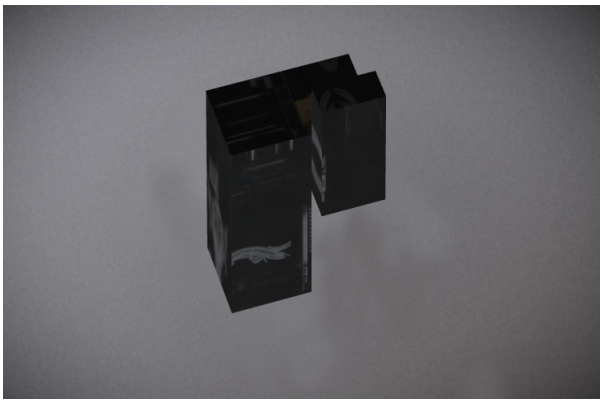
Usuario: Usuario del sistema	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Sergio A Quintana Gómez	
Descripción: Funcionalidad necesaria para poder visualizar los modelos 3D una vez sean seleccionados en el repositorio.	
Observaciones:	
Interfaz: 	

Tabla 2.2. Historia de usuario # 2

Historia de usuario	
Número: 2	**Nombre:** Controlar cámaras de escena
Usuario: Usuario del sistema	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: Sergio A Quintana Gómez	
Descripción: Funcionalidad necesaria para poder rotar, mover y hacer zoom al modelo de forma que se pueda observar desde diferentes ángulos.	
Observaciones:	

Continúa en la próxima página

Tabla 2.2. Continuación de la página anterior

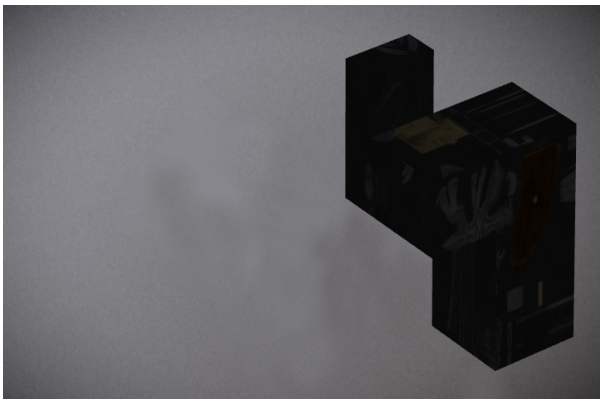
Interfaz: 

Tabla 2.3. Historia de usuario # 3

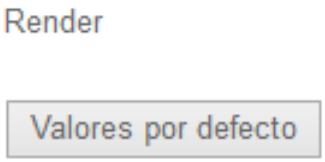
Historia de usuario	
Número: 3	**Nombre:** Seleccionar **Valores por defecto**
Usuario: Usuario del sistema	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1	**Iteración asignada:** 3
Programador responsable: Sergio A Quintana Gómez	
Descripción: Funcionalidad para visualizar la versión original del modelo 3D del diseñador, de forma que deshace cualquier cambio en los parámetros de visualización efectuados en el visor.	
Observaciones: Se mostrará como un botón (Valores por defecto).	
Interfaz: 	

Tabla 2.4. Historia de usuario # 4

Historia de usuario	
Número: 4	**Nombre:** Seleccionar **Desactivar**
Usuario: Usuario del sistema	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta

Continúa en la próxima página

Tabla 2.4. Continuación de la página anterior

Puntos estimados: 1	**Iteración asignada:** 3
Programador responsable: Sergio A Quintana Gómez	
Descripción: Funcionalidad que deshabilita cualquier opción de la sección Wireframe.	
Observaciones: Se mostrará como un botón con el nombre **Desactivar**.	
Interfaz: 	

Tabla 2.5. Historia de usuario # 5

Historia de usuario	
Número: 5	**Nombre:** Seleccionar **White button**
Usuario: Usuario del sistema	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1	**Iteración asignada:** 3
Programador responsable: Sergio A Quintana Gómez	
Descripción: Funcionalidad que activa un enmallado de color blanco sobre el modelo 3D.	
Observaciones: Se mostrará como un botón con el nombre **White button**.	
Interfaz: 	

Tabla 2.6. Historia de usuario # 6

Historia de usuario	
Número: 6	**Nombre:** Seleccionar **Black button**
Usuario: Usuario del sistema	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1	**Iteración asignada:** 3

Continúa en la próxima página

Tabla 2.6. Continuación de la página anterior

Programador responsable: Sergio A Quintana Gómez
Descripción: Funcionalidad que activa un enmallado de color negro sobre el modelo 3D.
Observaciones: Se mostrará como un botón con el nombre **Black button**.
Interfaz: 

Tabla 2.7. Historia de usuario # 7

Historia de usuario	
Número: 7	**Nombre:** Seleccionar **Aleatorio**
Usuario: Usuario del Sistema	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1	**Iteración asignada:** 3
Programador responsable: Sergio A Quintana Gómez	
Descripción: Funcionalidad que activa un enmallado de color aleatorio sobre el modelo 3D.	
Observaciones: Se mostrará como un botón con el nombre **Aleatorio**.	
Interfaz: 	

Tabla 2.8. Historia de usuario # 8

Historia de usuario	
Número: 8	**Nombre:** Seleccionar **Ambient Occlusion**
Usuario: Usuario del sistema	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Media
Puntos estimados: 1	**Iteración asignada:** 4
Programador responsable: Sergio A Quintana Gómez	

Continúa en la próxima página

Tabla 2.8. Continuación de la página anterior

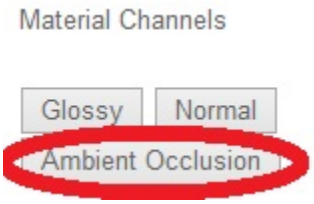
Descripción: Funcionalidad que habilita la oclusión ambiental sobre el modelo 3D.
Observaciones: Se mostrará como un botón con el nombre **Ambient Occlusion**.
Interfaz: 

Tabla 2.9. Historia de usuario # 9

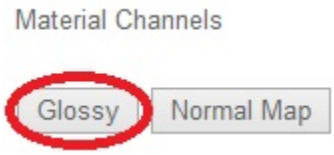
Historia de usuario	
Número: 9	**Nombre:** Seleccionar **Glossy**
Usuario: Usuario del sistema	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1	**Iteración asignada:** 4
Programador responsable: Sergio A Quintana Gómez	
Descripción: Funcionalidad que habilita el suavizado del modelo 3D.	
Observaciones: Se mostrará como un botón con el nombre **Glossy**.	
Interfaz: 	

Tabla 2.10. Historia de usuario # 10

Historia de usuario	
Número: 10	**Nombre:** Seleccionar **Normal**
Usuario: Usuario del sistema	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1	**Iteración asignada:** 4
Programador responsable: Sergio A Quintana Gómez	
Descripción: Funcionalidad que activa el mapa de las normales sobre el modelo 3D.	
Observaciones: Se mostrará como un botón con el nombre **Normal**.	

Continúa en la próxima página

Tabla 2.10. Continuación de la página anterior

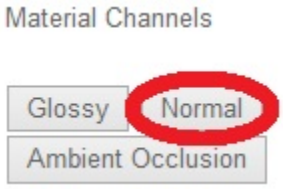
Interfaz: 

Tabla 2.11. Historia de usuario # 11

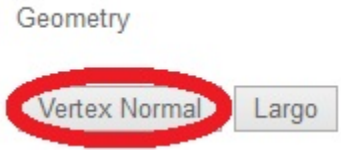
Historia de usuario	
Número: 11	**Nombre:** Seleccionar **Vertex Normals**
Usuario: Usuario del sistema	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1	**Iteración asignada:** 4
Programador responsable: Sergio A Quintana Gómez	
Descripción: Funcionalidad que muestra las normales de los vértices que posee la geometría del modelo 3D.	
Observaciones: Se mostrará como un botón con el nombre **Vertex Normals**.	
Interfaz: 	

Tabla 2.12. Historia de usuario # 12

Historia de usuario	
Número: 12	**Nombre:** Seleccionar **Largo**
Usuario: Usuario del sistema	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1	**Iteración asignada:** 4
Programador responsable: Sergio A Quintana Gómez	
Descripción: Funcionalidad que muestra las normales de los vértices de un mayor tamaño.	
Observaciones: Se mostrará como un botón con el nombre **Largo**.	

Continúa en la próxima página

Tabla 2.12. Continuación de la página anterior

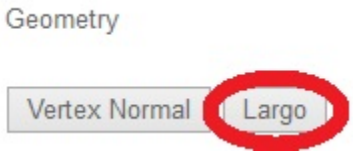
Interfaz: 

Tabla 2.13. Historia de usuario # 13

Historia de usuario	
Número: 13	**Nombre:** Seleccionar **Wireframe Sólido**
Usuario: Usuario del sistema	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1	**Iteración asignada:** 4
Programador responsable: Sergio A Quintana Gómez	
Descripción: Funcionalidad que habilita el enmallado del modelo en un sólido de color rojo y las mallas de color blanco.	
Observaciones: Se mostrará como un botón con el nombre **Wireframe Sólido**.	
Interfaz: 	

Tabla 2.14. Historia de usuario # 14

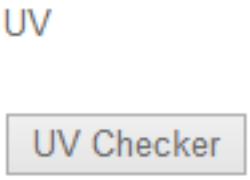
Historia de usuario	
Número: 14	**Nombre:** Seleccionar **WireOnly**
Usuario: Usuario del sistema	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1	**Iteración asignada:** 4
Programador responsable: Sergio A Quintana Gómez	
Descripción: Funcionalidad que habilita el enmallado del modelo sin un sólido o textura de fondo.	
Observaciones: Se mostrará como un botón con el nombre **WireOnly**.	

Continúa en la próxima página

Tabla 2.14. Continuación de la página anterior

Interfaz: 

Tabla 2.15. Historia de usuario # 15

Historia de usuario	
Número: 15	**Nombre:** Seleccionar **UV Checker**
Usuario: Usuario del sistema	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1	**Iteración asignada:** 4
Programador responsable: Sergio A Quintana Gómez	
Descripción: Funcionalidad que muestra la visualización del modelo utilizando una textura sobre el modelo 3D.	
Observaciones: Se mostrará como un botón con el nombre **UV Checker**.	
Interfaz: 	

2.5. Fase II Diseño:

La metodología XP recomienda crear diseños simples y sencillos. Por esta razón, obtener un diseño que sea fácil de entender e implementar es un objetivo deseable para el equipo de desarrollo. Como resultado, en el futuro se reducirán los costos y el esfuerzo de los desarrolladores para mantener el sistema.

En la fase de diseño el cliente prioriza las HU y los programadores realizan una posible estimación del esfuerzo necesario para la programación de cada HU. Para realizar las estimaciones de los esfuerzos asociados a la implementación de las HU se utiliza como medida los puntos de estimación. Estos puntos fueron definidos anteriormente por el cliente al establecer la prioridad de las HU. El valor de las HU es un valor real que generalmente oscila entre 1 y 3 puntos. Esta planificación se puede realizar basada en el

tiempo o el alcance que se estima para cada HU.

2.5.1. Plan de Estimación

A continuación, se muestra mediante una tabla la planificación de las diferentes HU para cada iteración teniendo en cuenta su prioridad.

Tabla 2.16. Estimación de esfuerzo por historia de usuario

Iteración	Historias de usuario		Puntos estimados (semanas)
1	1	Mostrar archivo	1
2	2	Controlar cámaras de escena	1
3	3	Valores por defecto	1
	4	Desactivar	1
	5	White button	1
	6	Black button	1
	7	Aleatorio	1
4	8	Ambient Occlusion	1
	9	Glossy	1
	10	Normal	1
	11	Vertex Normal	1
	12	Largo	1
	13	Wireframe Sólido	1
	14	WireOnly	1
	15	UV Checker	1
Total			**15.0**

2.5.2. Plan de duración de iteraciones

Tabla 2.17. Plan de duración de las iteraciones

Iteración	Historias de usuario		Duración (semanas)
1	1	Mostrar archivo	1.0
2	2	Controlar cámaras de escena	1.0
3	3	Valores por defecto	5.0
	4	Desactivar	
	5	White button	
	6	Black button	
	7	Aleatorio	
4	8	Ambient Occlusion	8.0
	9	Glossy	
	10	Normal	

Continúa en la próxima página

Tabla 2.17. Continuación de la página anterior

	11	Vertex Normal	
	12	Largo	
	13	Wireframe Sólido	
	14	WireOnly	
	15	UV Checker	
Total			**15.0**

2.5.3. Tarjetas CRC

Las tarjetas CRC identifican y organizan las clases bajo el paradigma orientado a objetos (lo que incluye asignación de responsabilidades). Cada tarjeta contiene el nombre de la clase (que representa una o más historias de usuario), una descripción de las responsabilidades o métodos asociados con la clase, así como la lista de las clases con que se relaciona o colaboran con ella. La siguiente tarjeta CRC corresponde al diseño de la solución.

Tabla 2.18. Tarjeta CRC # 1

Tarjeta CRC	
Clase: ModelView	
Responsabilidad	**Colaboración**
• Cargar modelo .OBJ. • Cargar modelo .FBX. • Mover el modelo. • Rotar el modelo. • Hacer zoom al modelo. • Mostrar el render final del modelo. • Mostrar el enmallado del modelo. • Cambiar coloración del enmallado del modelo. • Mostrar la oclusión ambiental del modelo. • Mostrar el suavizado del modelo. • Mostrar el mapa de normales del modelo. • Mostrar las normales de los vértices del modelo. • Mostrar el mapa de texturas del modelo para las coordenadas UV.	

Consideraciones parciales del capítulo

En este capítulo se cumplió con las Fases I y II de la metodología utilizada. Para ello se definieron un total de 15 Historias de Usuarios, divididas en 4 iteraciones y teniendo un plan de duración total de 15 semanas. Con estos artefactos ingenieriles definidos, el proceso de desarrollo está listo para pasar a la próxima etapa, en la que se comenzará con la implementación del sistema.

CAPÍTULO 3

Resultados y validación del sistema

En el presente capítulo se abordan los resultados obtenidos durante el proceso de desarrollo. Para ello se definieron las tareas de ingeniería correspondientes a cada HU por iteraciones. Además se hizo un análisis detallado de los casos de pruebas que se le realizaron al software en cada iteración. De este modo, se puede entregar un producto final probado y listo para entregar al cliente con la mayor aceptación posible, tal y como propone XP.

3.1. Fase III - Codificación

Como se describió anteriormente en el Capítulo 2, el cliente es una parte más del equipo de desarrollo. Su presencia es indispensable en las distintas fases de XP. A la hora de codificar una historia de usuario su presencia es aún más necesaria. Los clientes son los que crean las HU y negocian los tiempos en los que estas serán implementadas. Antes de desarrollar cada HU, el cliente debe precisar las funcionalidades de cada HU y también deberá estar presente cuando se realicen las pruebas que verifican que la HU implementada cumple la funcionalidad requerida.

3.1.1. Tareas de ingeniería para las historias de usuarios

Las tareas de ingeniería son otros artefactos de la metodología XP para la planificación, diseño e implementación del sistema. Con ellas se pretende desglosar en actividades las Historias de Usuario con un nivel técnico asequible a los especialistas que atenderán la misma [20]. A continuación se realizarán las tareas de ingeniería correspondientes a las diferentes iteraciones del proyecto. En este epígrafe solo se mostrarán algunas tareas de ingeniería, las tareas restantes se pueden consultar en el Anexo A.1.

Tareas de ingeniería (Iteración 1) Para la iteración 1 se definieron un total de dos tareas de ingeniería correspondientes a la **HU Mostrar Archivo**, las cuales se muestran a continuación:

HU: Mostrar Archivo

Tabla 3.1. Tarea de ingeniería # 1

Tarea	
Número de tarea: 1	**Número de Historia de usuario:** 1
Nombre de la tarea: Seleccionar el tipo de archivo	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.4
Fecha de inicio: 1 de noviembre de 2017	**Fecha de fin:** 2 de noviembre de 2017
Programador responsable: Sergio A Quintana Gómez	
Descripción: Desarrollar una función que permita obtener la extensión del archivo.	

Tabla 3.2. Tarea de ingeniería # 2

Tarea	
Número de tarea: 2	**Número de Historia de usuario:** 1
Nombre de la tarea: Seleccionar el script de carga	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.6
Fecha de inicio: 3 de noviembre de 2017	**Fecha de fin:** 7 de noviembre de 2017
Programador responsable: Sergio A Quintana Gómez	
Descripción: Desarrollar una función que permita obtener el script necesario para cargar el archivo.	

Tareas de ingeniería (Iteración 2) Para la iteración 2 se definieron un total de tres tareas de ingeniería correspondientes a la **HU Controlar cámaras de la escena**, de las cuales se muestran dos a continuación. Las tareas de ingeniería restantes se pueden consultar en el Anexo A.1.

HU: Controlar cámaras de la escena

Tabla 3.3. Tarea de ingeniería # 3

Tarea	
Número de tarea: 3	**Número de Historia de usuario:** 2
Nombre de la tarea: Mover el modelo	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.4
Fecha de inicio: 8 de noviembre de 2017	**Fecha de fin:** 9 de noviembre de 2017
Programador responsable: Sergio A Quintana Gómez	
Descripción: Desarrollar funciones que permitan mover el modelo.	

Tabla 3.4. Tarea de ingeniería # 4

Tarea	
Número de tarea: 4	**Número de Historia de usuario:** 2
Nombre de la tarea: Rotar el modelo	

Continúa en la próxima página

Tabla 3.4. Continuación de la página anterior

Tipo de tarea: Desarrollo	**Puntos estimados:** 0.4
Fecha de inicio: 10 de noviembre de 2017	**Fecha de fin:** 13 de noviembre de 2017
Programador responsable: Sergio A Quintana Gómez	
Descripción: Desarrollar funciones que permitan rotar el modelo.	

Tareas de ingeniería (Iteración 3) Para la iteración 3 se definieron un total de diez tareas de ingeniería correspondientes a las Historias de Usuario de esta iteración. De las diez tareas, se muestran tres a continuación. Las restantes se pueden consultar en el Anexo A.1.

HU: Seleccionar Valores por defecto

Tabla 3.5. Tarea de ingeniería # 5

Tarea	
Número de tarea: 5	**Número de Historia de usuario:** 3
Nombre de la tarea: Implementar funcionalidad Valores por defecto	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.8
Fecha de inicio: 15 de noviembre de 2017	**Fecha de fin:** 20 de noviembre de 2017
Programador responsable: Sergio A Quintana Gómez	
Descripción: Desarrollar la función que permita reiniciar los cambios relizados al modelo.	

HU: Seleccionar Desactivar

Tabla 3.6. Tarea de ingeniería # 6

Tarea	
Número de tarea: 6	**Número de Historia de usuario:** 4
Nombre de la tarea: Implementar funcionalidad Desactivar	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.8
Fecha de inicio: 21 de noviembre de 2017	**Fecha de fin:** 24 de noviembre de 2017
Programador responsable: Sergio A Quintana Gómez	
Descripción: Desarrollar la función que permita eliminar las opciones seleccionadas en el submenú Wireframe.	

HU: Seleccionar White button

Tabla 3.7. Tarea de ingeniería # 7

Tarea	
Número de tarea: 7	**Número de Historia de usuario:** 5
Nombre de la tarea: Implementar funcionalidad White	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.8
Fecha de inicio: 28 de noviembre de 2017	**Fecha de fin:** 1 de diciembre de 2017

Continúa en la próxima página

Tabla 3.7. Continuación de la página anterior

Programador responsable: Sergio A Quintana Gómez
Descripción: Desarrollar la función que permita establecer blanco como color del enmallado del modelo.

Tareas de ingeniería (Iteración 4) Para la iteración 4 se definieron un total de doce tareas de ingeniería correspondientes a las Historias de Usuario de esta iteración. De las doce tareas, se muestran dos a continuación. Las restantes se pueden consultar en el Anexo A.1.

HU: Seleccionar Ambient Occlusion

Tabla 3.8. Tarea de ingeniería # 8

Tarea	
Número de tarea: 8	**Número de Historia de usuario:** 8
Nombre de la tarea: Implementar funcionalidad Ambient Occlusion	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.8
Fecha de inicio: 9 de enero de 2018	**Fecha de fin:** 12 de enero de 2018
Programador responsable: Sergio A Quintana Gómez	
Descripción: Desarrollar la función que permita aplicar Ambient Occlusion al modelo 3D.	

Tabla 3.9. Tarea de ingeniería # 9

Tarea	
Número de tarea: 9	**Número de Historia de usuario:** 8
Nombre de la tarea: Crear botón Ambient Occlusion	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 15 de diciembre de 2017	**Fecha de fin:** 15 de diciembre de 2017
Programador responsable: Sergio A Quintana Gómez	
Descripción: Desarrollar un botón para la funcionalidad Ambient Occlusion.	

Como propone XP, una vez definidas las tareas de ingeniería por UH, es necesario realizar un conjunto de pruebas a la solución desarrollada para comprobar el nivel de aceptación y calidad de esta para el cliente. Primeramente, se deben diseñar los casos de prueba que se van a utilizar para posteriormente aplicarlos y probar el software. De esta forma, se chequea el cumplimiento por parte de los desarrolladores de las funcionalidades que solicitó el cliente.

3.2. Fase IV - Pruebas

La integración continua, la entrega y la implementación dependen en gran medida de las pruebas automatizadas para determinar la eficacia y la corrección de cada cambio de código. Se necesitan diferentes

tipos de pruebas a lo largo de estos procesos para ganar confianza en una solución dada [21].

Uno de los pilares de la eXtreme Programming es el proceso de pruebas. XP anima a probar constantemente tanto como sea posible. Esto permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. También permite aumentar la seguridad de evitar efectos colaterales no deseados a la hora de realizar modificaciones. XP divide las pruebas del sistema en dos grupos: pruebas unitarias, encargadas de verificar el código y diseñada por los programadores, y pruebas de aceptación o pruebas funcionales destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida por el cliente final [22].

3.2.1. Pruebas de aceptación

Las pruebas de aceptación son pruebas de caja negra definidas por el cliente para cada historia de usuario, y tienen como objetivo asegurar que las funcionalidades del sistema cumplen con lo que se espera de ellas. En efecto, las pruebas de aceptación corresponden a una especie de documento de requerimientos en XP, ya que marcan el camino a seguir en cada iteración, indicándole al equipo de desarrollo la dirección a seguir así como los puntos o funcionalidades en que se debe poner el mayor esfuerzo y atención.

Las pruebas de aceptación tienen una importancia crítica para el éxito de una iteración. Por esta razón, el probador debe diseñar las pruebas de aceptación lo antes posible a partir del comienzo de la iteración. Estas pruebas se presentarán luego al cliente para su aprobación y posteriormente se presentarán al equipo de desarrollo [21]. A continuación se definen un grupo de pruebas para cada iteración. Las pruebas restantes se pueden consultar en el Anexo A.2.

Pruebas de aceptación para la Iteración 1 Para la iteración 1 se definieron un total de dos pruebas de aceptación correspondientes a la HU: Mostrar Archivo las cuales se muestran a continuación:

Tabla 3.10. Prueba de aceptación # 1

Caso de prueba de aceptación	
Código: HU1_P1	**Historia de usuario:** 1
Nombre: Seleccionar el tipo de archivo	
Descripción: Prueba realizada a la funcionalidad: Seleccionar el tipo de archivo.	
Condiciones de ejecución: El usuario debe estar autenticado en el sitio.	
Pasos de ejecución: 1. El usuario debe ubicarse en la pantalla de inicio del sitio. 2. Situar el cursor sobre uno de los modelos que se encuentran en esta pantalla. 3. Seleccionar entre las opciones, el icono de visualización o hacer clic izquierdo sobre el modelo.	
Resultados esperados: Satisfactorio	

Tabla 3.11. Prueba de aceptación # 2

Caso de prueba de aceptación	
Código: HU1_P2	**Historia de usuario:** 1
Nombre: Seleccionar el script de carga	
Descripción: Prueba realizada a la funcionalidad: Seleccionar el script de carga.	
Condiciones de ejecución: El usuario debe estar autenticado en el sitio.	
Pasos de ejecución: 1. El usuario debe hacer clic izquierdo en la opción de visualización del modelo una vez situado sobre él o hacer clic izquierdo sobre el modelo. 2. Como respuesta el software debe seleccionar el script de carga correspondiente al tipo de archivo seleccionado. 3. El usuario será redireccionado a la interfaz web donde se encuentra el visor y el menú de edición con el modelo cargado.	
Resultados esperados: Satisfactorio	

Pruebas de aceptación para la Iteración 2 Para la iteración 2 se definieron un total de tres pruebas de aceptación correspondientes a la HU: Controlar cámaras de la escena las cuales se muestran a continuación.

Tabla 3.12. Prueba de aceptación # 3

Caso de prueba de aceptación	
Código: HU2_P1	**Historia de usuario:** 2
Nombre: Mover el modelo	
Descripción: Prueba realizada a la funcionalidad: Mover el modelo.	
Condiciones de ejecución: 1. El usuario debe estar autenticado en el sitio. 2. El usuario debe haber seleccionado un modelo para visualizar.	
Pasos de ejecución: 1. El usuario debe hacer clic derecho sobre el visor donde se encuentra el modelo cargado. 2. Manteniendo presionado el clic derecho el usuario debe mover el cursor hacia diferentes direcciones dentro del marco del visor. 3. Como respuesta del sistema el modelo deberá moverse en el sentido del cursor.	
Resultados esperados: Satisfactorio	

Tabla 3.13. Prueba de aceptación # 4

Caso de prueba de aceptación	
Código: HU2_P2	**Historia de usuario:** 2

Continúa en la próxima página

Tabla 3.13. Continuación de la página anterior

Nombre: Rotar el modelo
Descripción: Prueba realizada a la funcionalidad: Rotar el modelo.
Condiciones de ejecución: 1. El usuario debe estar autenticado en el sitio. 2. El usuario debe haber seleccionado un modelo para visualizar.
Pasos de ejecución: 1. El usuario debe hacer clic izquierdo sobre el visor donde se encuentra el modelo cargado. 2. Manteniendo presionado el clic izquierdo el usuario debe mover el cursor hacia diferentes direcciones dentro del marco del visor. 3. Como respuesta del sistema, el modelo deberá rotar sobre su eje en el sentido del cursor.
Resultados esperados: Satisfactorio

Tabla 3.14. Prueba de aceptación # 5

Caso de prueba de aceptación	
Código: HU2_P3	**Historia de usuario:** 2
Nombre: Hacer zoom	
Descripción: Prueba realizada a la funcionalidad: Hacer zoom.	
Condiciones de ejecución: 1. El usuario debe estar autenticado en el sitio. 2. El usuario debe haber seleccionado un modelo para visualizar.	
Pasos de ejecución: 1. El usuario debe situar el cursor en el área del visor 2. Hacer rodar el botón central del ratón hacia arriba o abajo. 3. Como respuesta del sistema, el modelo debe aumentar o disminuir su tamaño respectivamente.	
Resultados esperados: Satisfactorio	

Pruebas de aceptación para la iteración 3 Para la iteración 3 se definieron un total de diez pruebas de aceptación correspondientes a las Historias de Usuario de esta iteración. En esta sección solo se muestran dos, las restantes se pueden consultar en el Anexo A.2.

Tabla 3.15. Prueba de aceptación # 6

Caso de prueba de aceptación	
Código: HU3_P1	**Historia de usuario:** 3
Nombre: Crear botón Valores por defecto	

Continúa en la próxima página

Tabla 3.15. Continuación de la página anterior

Descripción: Prueba realizada a la funcionalidad: Crear botón Valores por defecto.
Condiciones de ejecución: 1. El usuario debe estar autenticado en el sitio. 2. El usuario debe haber seleccionado un modelo para visualizar.
Pasos de ejecución: 1. Como respuesta del sistema, el usuario debe poder visualizar en el menú de edición un botón con el nombre Valores por defecto.
Resultados esperados: Satisfactorio

Tabla 3.16. Prueba de aceptación # 7

Caso de prueba de aceptación	
Código: HU3_P2	**Historia de usuario:** 3
Nombre: Implementar funcionalidad Valores por defecto	
Descripción: Prueba realizada a la funcionalidad: Implementar funcionalidad Valores por defecto.	
Condiciones de ejecución: 1. El usuario debe estar autenticado en el sitio. 2. El usuario debe haber seleccionado un modelo para visualizar.	
Pasos de ejecución: 1. El usuario debe hacer clic izquierdo en el botón Valores por defecto. 2. Como respuesta del sistema, el usuario debe poder observar en el visor el modelo en la forma que se diseñó inicialmente.	
Resultados esperados: Satisfactorio	

Pruebas de aceptación para la iteración 4 Para la iteración 4 se definieron un total de doce pruebas de aceptación correspondientes a las Historias de Usuario de esta iteración. De las doce tareas, se muestran dos a continuación. Las restantes se pueden consultar en el Anexo A.2.

Tabla 3.17. Prueba de aceptación # 8

Caso de prueba de aceptación	
Código: HU8_P1	**Historia de usuario:** 8
Nombre: Crear botón Ambient Occlusion	
Descripción: Prueba realizada a la funcionalidad: Crear botón Ambient Occlusion.	
Condiciones de ejecución: 1. El usuario debe estar autenticado en el sitio. 2. El usuario debe haber seleccionado un modelo para visualizar.	

Continúa en la próxima página

Tabla 3.17. Continuación de la página anterior

Pasos de ejecución: 1. Como respuesta del sistema, el usuario debe poder visualizar en el menú de edición un botón con el nombre Ambient Occlusion.
Resultados esperados: Satisfactorio

Tabla 3.18. Prueba de aceptación # 9

Caso de prueba de aceptación	
Código: HU8_P2	**Historia de usuario:** 8
Nombre: Implementar funcionalidad Ambient Occlusion	
Descripción: Prueba realizada a la funcionalidad: Implementar funcionalidad Ambient Occlusion.	
Condiciones de ejecución: 1. El usuario debe estar autenticado en el sitio. 2. El usuario debe haber seleccionado un modelo para visualizar.	
Pasos de ejecución: 1. El usuario debe hacer clic izquierdo en el botón Ambient Occlusion del menú de edición. 2. Como respuesta del sistema, el usuario debe poder observar la propiedad Oclusión Ambiental del modelo 3D.	
Resultados esperados: Satisfactorio	

Resultados de las pruebas de aceptación

En total se realizaron un total de 27 casos de pruebas de aceptación. Estas fueron realizadas pasando por cada iteración, supervisando de esta manera el cumplimiento de cada tarea de ingeniería correspondiente a estas. A continuación se muestra una gráfica con el grado de aceptación obtenido al realizar las pruebas.

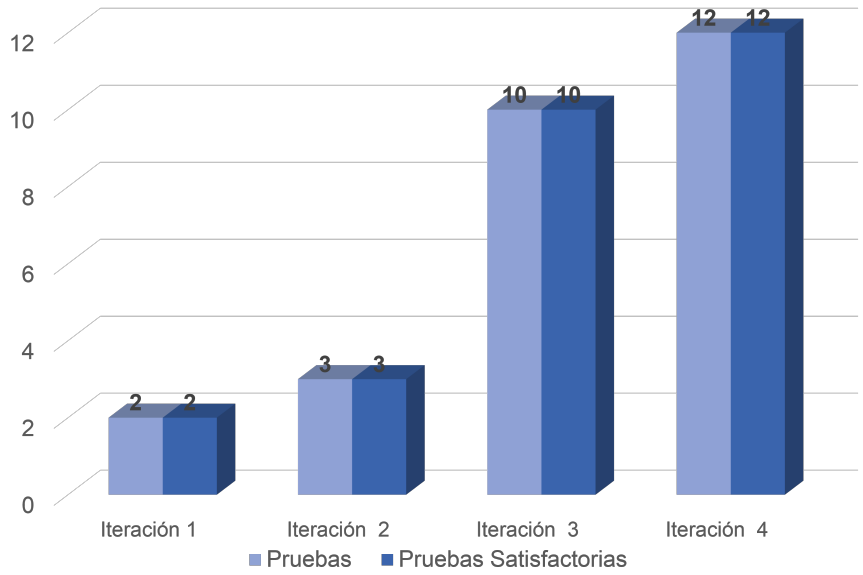


Figura 3.1. Resultados de las pruebas de aceptación.

Al culminar el proceso de pruebas por iteración, se detectaron un grupo de no conformidades que se fueron solucionando antes de comenzar la siguiente etapa, permitiendo que al culminar la última iteración de pruebas el componente tuviera un gran por ciento de aceptación ante el cliente. En la primera iteración se detectaron 2 no conformidades relacionadas con la vía de selección de los modelos. Para la segunda iteración se detectaron un total de 3 no conformidades relacionadas con la configuración del manejo de las cámaras de la escena. En la tercera iteración hubo un total de 4 no conformidades relacionadas con la nomenclatura de los botones y posiciones en el menú. Para la última iteración fueron corregidas 5 no conformidades relacionadas con nomenclatura de botones y posicionado en el menú, quedando un producto probado y con la mayor calidad posible.

3.3. Selección de modelos para pruebas de visualización

Para hacer las pruebas de visualización, se hizo una selección de modelos 3D donde se pudieron probar todas las funcionalidades del componente implementado. Para hacer la selección se definieron algunas características para los modelos. Se tuvieron en cuenta modelos 3D de distintos tamaños (Megabytes), diferentes cantidades de polígonos, modelos en los diferentes formatos que soporta el repositorio, así como la presencia de animaciones en los modelos. La selección final de los modelos se muestra en la siguiente tabla:

Tabla 3.19. Modelos utilizados para las pruebas de visualización

Nombre	Tipo	Tamaño(Mb)	Cantidad de polígonos	Animación	Descripción
Wolf	OBJ	2.21	87	No	Modelo que representa un lobo
Golff	FBX	0.20	-	Si	Lanzamiento en el GOLF
FBX	FBX	2.93	-	Si	Hombre corriendo
Cube	OBJ	0.02	18	No	Diseño a base de cubos

3.4. Pruebas de visualización Etapa I

Las pruebas de visualización consisten en visualizar correctamente un modelo seleccionado utilizando para ello el componente implementado. La visualización se considera correcta si el modelo se visualiza conservando los detalles y proporciones del modelo original. Los pasos a seguir al aplicar estas pruebas será subir al repositorio los modelos 3D en los formatos que este soporta, ya sea FBX u OBJ y comprobar la calidad visual con la que se muestra el modelo. En los resultados de esta prueba se describirá el comportamiento del software y los resultados alcanzados. A continuación, se muestran imágenes correspondientes a los resultados de las pruebas realizadas.

Figura 3.2. Imagen de prueba de visualización para modelo Wolf.

Figura 3.3. Imagen de prueba de visualización para modelo Golff.

Figura 3.4. Imagen de prueba de visualización para modelo FBX.

Figura 3.5. Imagen de prueba de visualización para modelo Cube.

Resultados de las pruebas de visualización Etapa I:

Como resultado de aplicar estas pruebas se obtuvo que el componente visualiza todos estos modelos 3D sin presentar problemas con las extensiones de los archivos ni con la identificación de los menús correspon-

dientes a cada uno de ellos. Por otro lado se identificó que los modelos que contienen un archivo **.mtl** con un solo material no presenta problemas, pero el visor es incapaz de trabajar con archivos **.mtl** que tengan asignado más de un material, por lo que en caso de que algún modelo en el repositorio tenga asociado un archivo de este tipo presentará problemas para visualizar el material del mismo. Tal es el caso del modelo Wolf.

3.5. Pruebas de visualización Etapa II

En esta segunda etapa se pretende probar las funcionalidades del componente a partir de un modelo seleccionado y siguiendo una serie de pasos definidos con el objetivo de abarcar todas las funcionalidades del software. De esta forma se comprobará la completitud del mismo.

Pasos a seguir para la segunda etapa:

1. Seleccionar modelo no usado en pruebas anteriores (Ver Anexo A.1).
2. Ampliar tamaño del modelo (Ver Anexo 3.11).
3. Disminuir tamaño del modelo (Ver Anexo A.2).
4. Rotar modelo en diferentes direcciones (Ver Anexo 3.10).
5. Mover modelo hacia diferentes direcciones (Ver Anexos A.3 y A.4).
6. Mostrar mallado del modelo de color blanco (Ver Anexo A.5).
7. Mostrar mallado del modelo de color negro (Ver Anexo A.6).
8. Mostrar mallado del modelo de un color aleatorio (Ver Anexo A.7).
9. Eliminar opciones del mallado mostrado en el paso anterior (Ver Anexo A.8).
10. Mostrar la oclusión ambiental (Ver Anexo A.9).
11. Mostrar el mapeado normal (Ver Anexo A.10).
12. Mostrar las normales de los vértices (Ver Anexo 3.9).
13. Mostrar el modelo de color rojo y el mallado color blanco (Opción Wireframe Sólido) (Ver Anexo 3.6).
14. Mostrar el mapeado de texturas UV (Ver Anexo 3.8).
15. Mostrar el modelo en su vista inicial (Opción Valores por defecto).
16. Mostrar Texturizado (Ver Anexo 3.7).

A continuación, se mostrarán imágenes de las pruebas realizadas a los requisitos principales del componente, las restantes funcionalidades también fueron probadas y se muestran en los anexos A.3.

Figura 3.6. Prueba realizada a la funcionalidad mostrar mallado (Wireframe Sólido).

Figura 3.7. Prueba realizada a la funcionalidad mostrar texturizado.

Figura 3.8. Prueba realizada a la funcionalidad mostrar mapeado de texturas UV.

Figura 3.9. Prueba realizada a la funcionalidad mostrar la normal de los vértices del modelo.

Figura 3.10. Prueba realizada a la funcionalidad rotar modelo.

Figura 3.11. Prueba realizada a la funcionalidad hacer zoom al modelo.

Consideraciones parciales del capítulo

En este capítulo se concluyeron las Fase III y IV de la metodología utilizada. Como resultado, se definieron 27 tareas de ingeniería para las Historias de Usuarios antes definidas por iteraciones. Una vez culminado este proceso, se hizo el diseño de casos de prueba para cada tarea de ingeniería. Se definieron un total de 27 casos de pruebas de aceptación. Luego de ser aceptadas por el cliente, las pruebas se aplicaron al sistema y se mostraron sus resultados. Como consecuencia, se obtuvo un software probado e integrado a la versión actual del repositorio web del centro Vertex.

Conclusiones

Con la realización del presente trabajo es posible arribar a las siguientes conclusiones:

- El componente de visualización propuesto permite visualizar los modelos 3D sin necesidad de instalar ningún software adicional, lo que facilita el proceso de visualización de los modelos 3D del repositorio.
- El uso de la biblioteca Three.js permitió realizar un visualizador de modelo 3D explotando las ventajas de WebGL en los navegadores Web.

Recomendaciones

Con el objetivo de mejorar el componente de visualización actual y continuar su desarrollo futuro, se proponen las siguientes recomendaciones:

- Implementar un método que sea capaz de leer archivos mtl con texturas divididas en diferentes materiales, ya que en modelos con muchos polígonos es muy común que tengan varios materiales asignados.
- Implementar funcionalidades que permitan trabajar con propiedades como geometría, texturas y materiales de los archivos con extensión FBX.

Acrónimos

API Interfaz de Programación de Aplicaciones. 9

GLSL OpenGL Shading Language. 10
GPU Unidad de Procesamiento Gráfico. 10

HTTP Hypertext Transference Protocol. 1

UCI Universidad de las Ciencias Informáticas. 2, 6

WebGL Web Graphics Library. 9, 11, 12

Referencias bibliográficas

[1] Natalia Olifer y Victor Olifer. *Computer networks: Principles, technologies and protocols for network design*. Wiley Publishing, 2005 (vid. pág. 1).

[2] Charith Perera et al., «Context aware computing for the internet of things: A survey». En: *IEEE communications surveys & tutorials* 16.1 (2014), págs. 414-454 (vid. pág. 1).

[3] Joseph Carl Robnett Licklider. «Historia de Internet». En: *Boston, Estados Unidos* (2002) (vid. págs. 1, 2).

[4] Alicia Ramos Martín y Maria Jesus Ramos Martín. *Aplicaciones Web*. Ediciones Paraninfo, SA, 2014 (vid. págs. 1, 2).

[5] Jose Texier et al., «El Uso de Repositorios y su Importancia para la Educación en Ingeniería». En: *World Engineering Education Forum (WEEF)(Buenos Aires, 2012)*. 2012 (vid. pág. 2).

[6] Alban Denoyel. «Facebook now supports Sketchfab 3D embeds!» En: *Sketchfab* (2015) (vid. pág. 2).

[7] Bart Veldhuizen. «Sketchfab releases Physically Based Rendering». En: *BlenderNation* (2015) (vid. pág. 2).

[8] Maria Vanesa Doria, Claudia Inés Inchaurrondo y Germán Antonio Montejano. «Directrices para la construcción de un repositorio temático». En: *TE & ET* (2013) (vid. págs. 5, 7).

[9] Julián Pérez Porto y María Merino. «Definición de repositorio». En: *Definicion.de* (2016) (vid. pág. 5).

[10] JORGE POLANCO-CORTÉS. «Repositorios digitales». En: *Definición y pautas para su creación. Disponible on line en https://ucrindex. ucr. ac. cr/docs/repositorios-digitales-definicion-y-pautas-para-su-creacion. pdf* (2014) (vid. pág. 5).

[11] Cristina Barrueco José Manuel; García Testar. «Repositorios Institucionales Universitarios: Evolución y perspectivas». En: *Fesabid 2009.* (2009) (vid. pág. 6).

[12] Antonio Hernández Pérez y María Antonia García Moreno. «Datos abiertos y repositorios de datos: nuevo reto para los bibliotecarios». En: (2013) (vid. pág. 7).

[13] Kelly L. Murdock. *3ds Max® 8 Bible*. Wiley Publishing, Inc., 2006 (vid. pág. 8).

[14] WebGL. *WebGL : Sitio oficial*. 2006. URL: https://www.khronos.org/webgl/ (vid. pág. 10).

[15] Threejs. *Three.js : Sitio oficial*. 2010. URL: https://threejs.org/ (vid. pág. 10).

[16] Luz Caballero. «An Introduction to WebGL ? Part 1». En: *Dev.Opera* (2011) (vid. pág. 10).

[17] Danilo Sato et al., «Experiences tracking agile projects: an empirical study». En: *Journal of the Brazilian Computer Society* 12.3 (2006), págs. 45-64 (vid. pág. 13).

[18] Kent Beck. «Extreme Programming Explained: Embrace Change [1ª ed.]» En: *Addison Wesley, Stoughton* (1999) (vid. pág. 13).

[19] I. Sommerville. «Ïngeniería del software"». En: *Pearson* (2005) (vid. pág. 16).

[20] Angel E Agreda Marin, Raquel Diéguez Batista y Yaimara Pizano Yan. «SISTEMA DE GESTIÓN DE INFORMACIÓN PARA LA ADMINISTRACIÓN DE RECURSOS HUMANOS DE LA CONTRALORÍA DEL MUNICIPIO ANGOSTURA, ESTADO BOLIVAR». En: *Universidad&Ciencia* 3.1 (2017), págs. 1-11 (vid. pág. 33).

[21] Justin Ellingwood. «An Introduction to Continuous Integration, Delivery, and Deployment». En: *DigitalOcean* (2017) (vid. pág. 37).

[22] Fernando Cócaro Dayvis Malfará Diego Cukerman. «Testing en eXtreme Programming». En: *Gestión de Software* (2006) (vid. pág. 37).

Generado con LaTeX: 18 de junio de 2018: 12:12 Noon

Apéndices

APÉNDICE A

Anexos

A.1. Tareas de ingeniería

Tareas de ingeniería (Iteración 2)

HU: Controlar cámaras de la escena

Tabla A.1. Tarea de ingeniería # 10

Tarea	
Número de tarea: 10	**Número de Historia de usuario:** 2
Nombre de la tarea: Hacer zoom al modelo	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 14 de noviembre de 2017	**Fecha de fin:** 14 de noviembre de 2017
Programador responsable: Sergio A Quintana Gómez	
Descripción: Desarrollar una función que permita aumentar el tamaño de visualización del modelo.	

HU: Seleccionar Valores por defecto

Tabla A.2. Tarea de ingeniería # 11

Tarea	
Número de tarea: 11	**Número de Historia de usuario:** 3
Nombre de la tarea: Crear botón Valores por defecto	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 14 de noviembre de 2017	**Fecha de fin:** 14 de noviembre de 2017
Programador responsable: Sergio A Quintana Gómez	
Descripción: Desarrollar un botón para la funcionalidad Valores por defecto.	

HU: Seleccionar Desactivar

Tabla A.3. Tarea de ingeniería # 12

Tarea	
Número de tarea: 12	**Número de Historia de usuario:** 4
Nombre de la tarea: Crear botón Desactivar	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 27 de noviembre de 2017	**Fecha de fin:** 27 de noviembre de 2017
Programador responsable: Sergio A Quintana Gómez	
Descripción: Desarrollar un botón para la funcionalidad desactivar.	

HU: Seleccionar White button

Tabla A.4. Tarea de ingeniería # 13

Tarea	
Número de tarea: 13	**Número de Historia de usuario:** 5
Nombre de la tarea: Crear botón White	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 4 de diciembre de 2017	**Fecha de fin:** 4 de diciembre de 2017
Programador responsable: Sergio A Quintana Gómez	
Descripción: Desarrollar un botón para la funcionalidad White.	

HU: Seleccionar Black button

Tabla A.5. Tarea de ingeniería # 14

Tarea	
Número de tarea: 14	**Número de Historia de usuario:** 6
Nombre de la tarea: Implementar funcionalidad Black	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.8
Fecha de inicio: 5 de diciembre de 2017	**Fecha de fin:** 8 de diciembre de 2017
Programador responsable: Sergio A Quintana Gómez	
Descripción: Desarrollar la función que permita establecer negro como color del enmallado del modelo.	

Tabla A.6. Tarea de ingeniería # 15

Tarea	
Número de tarea: 15	**Número de Historia de usuario:** 6
Nombre de la tarea: Crear botón Black	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 11 de diciembre de 2017	**Fecha de fin:** 11 de diciembre de 2017
Programador responsable: Sergio A Quintana Gómez	

Continúa en la próxima página

Tabla A.6. Continuación de la página anterior

Descripción: Desarrollar un botón para la funcionalidad Black.

HU: Seleccionar Aleatorio

Tabla A.7. Tarea de ingeniería # 16

Tarea	
Número de tarea: 16	**Número de Historia de usuario:** 7
Nombre de la tarea: Implementar funcionalidad Aleatorio	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.8
Fecha de inicio: 12 de diciembre de 2017	**Fecha de fin:** 15 de diciembre de 2017
Programador responsable: Sergio A Quintana Gómez	
Descripción: Desarrollar la función que permita establecer Aleatorio como color del enmallado del modelo.	

Tabla A.8. Tarea de ingeniería # 17

Tarea	
Número de tarea: 17	**Número de Historia de usuario:** 7
Nombre de la tarea: Crear botón Aleatorio	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 8 de enero de 2018	**Fecha de fin:** 8 de enero de 2018
Programador responsable: Sergio A Quintana Gómez	
Descripción: Desarrollar un boton para la funcionalidad Aleatorio.	

Tareas de ingeniería (Iteración 4)

HU: Seleccionar Glossy

Tabla A.9. Tarea de ingeniería # 18

Tarea	
Número de tarea: 18	**Número de Historia de usuario:** 9
Nombre de la tarea: Implementar funcionalidad Glossy	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.8
Fecha de inicio: 16 de enero de 2018	**Fecha de fin:** 19 de enero de 2018
Programador responsable: Sergio A Quintana Gómez	
Descripción: Desarrollar la función que permita observar la propiedad de suavisado del modelo.	

Tabla A.10. Tarea de ingeniería # 19

Tarea	
Número de tarea: 19	**Número de Historia de usuario:** 9
Nombre de la tarea: Crear botón Glossy	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 22 de enero de 2018	**Fecha de fin:** 22 de enero de 2018
Programador responsable: Sergio A Quintana Gómez	
Descripción: Desarrollar un boton para la funcionalidad Glossy.	

HU: Seleccionar Normal

Tabla A.11. Tarea de ingeniería # 20

Tarea	
Número de tarea: 20	**Número de Historia de usuario:** 10
Nombre de la tarea: Implementar funcionalidad Normal	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.8
Fecha de inicio: 23 de enero de 2018	**Fecha de fin:** 26 de enero de 2018
Programador responsable: Sergio A Quintana Gómez	
Descripción: Desarrollar la función que permita observar el mapeado normal o (mapa de las normales) del modelo.	

Tabla A.12. Tarea de ingeniería # 21

Tarea	
Número de tarea: 21	**Número de Historia de usuario:** 10
Nombre de la tarea: Crear botón Normal	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 29 de enero de 2018	**Fecha de fin:** 29 de enero de 2018
Programador responsable: Sergio A Quintana Gómez	
Descripción: Desarrollar un boton para la funcionalidad Normal.	

HU: Seleccionar Vertex Normal

Tabla A.13. Tarea de ingeniería # 22

Tarea	
Número de tarea: 22	**Número de Historia de usuario:** 11
Nombre de la tarea: Implementar funcionalidad Vertex Normal	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.8
Fecha de inicio: 30 de enero de 2018	**Fecha de fin:** 2 de febrero de 2018
Programador responsable: Sergio A Quintana Gómez	

Continúa en la próxima página

Tabla A.13. Continuación de la página anterior

Descripción: Desarrollar la función que permita observar la normal de los vertices del modelo.

Tabla A.14. Tarea de ingeniería # 23

Tarea	
Número de tarea: 23	**Número de Historia de usuario:** 11
Nombre de la tarea: Crear botón Vertex Normal	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 5 de febrero de 2018	**Fecha de fin:** 5 de febrero de 2018
Programador responsable: Sergio A Quintana Gómez	
Descripción: Desarrollar un boton para la funcionalidad Vertex Normal.	

HU: Seleccionar Largo

Tabla A.15. Tarea de ingeniería # 24

Tarea	
Número de tarea: 24	**Número de Historia de usuario:** 12
Nombre de la tarea: Implementar funcionalidad Largo	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.8
Fecha de inicio: 6 de febrero de 2018	**Fecha de fin:** 9 de febrero de 2018
Programador responsable: Sergio A Quintana Gómez	
Descripción: Desarrollar la función que permita observar la normal de los vertices del modelo de mayor longitud.	

Tabla A.16. Tarea de ingeniería # 25

Tarea	
Número de tarea: 25	**Número de Historia de usuario:** 12
Nombre de la tarea: Crear botón Largo	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 12 de febrero de 2018	**Fecha de fin:** 12 de febrero de 2018
Programador responsable: Sergio A Quintana Gómez	
Descripción: Desarrollar un boton para la funcionalidad Largo.	

HU: Seleccionar Wirefrafe Sólido

Tabla A.17. Tarea de ingeniería # 26

Tarea	
Número de tarea: 26	**Número de Historia de usuario:** 13

Continúa en la próxima página

Tabla A.17. Continuación de la página anterior

Nombre de la tarea: Implementar funcionalidad Wirefrafe Sólido	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.8
Fecha de inicio: 13 de febrero de 2018	**Fecha de fin:** 16 de febrero de 2018
Programador responsable: Sergio A Quintana Gómez	
Descripción: Desarrollar la función que permita observar el enmallado del modelo.	

Tabla A.18. Tarea de ingeniería # 27

Tarea	
Número de tarea: 27	**Número de Historia de usuario:** 13
Nombre de la tarea: Crear botón Wirefrafe Sólido	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 19 de febrero de 2018	**Fecha de fin:** 19 de febrero de 2018
Programador responsable: Sergio A Quintana Gómez	
Descripción: Desarrollar un boton para la funcionalidad Wirefrafe Sólido.	

HU: Seleccionar UV Checker

Tabla A.19. Tarea de ingeniería # 28

Tarea	
Número de tarea: 28	**Número de Historia de usuario:** 14
Nombre de la tarea: Implementar funcionalidad UV Checker	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.8
Fecha de inicio: 20 de febrero de 2018	**Fecha de fin:** 23 de febrero de 2018
Programador responsable: Sergio A Quintana Gómez	
Descripción: Desarrollar la función que permita observar el mapeado de texturas para las cordenada UV del modelo.	

Tabla A.20. Tarea de ingeniería # 29

Tarea	
Número de tarea: 29	**Número de Historia de usuario:** 14
Nombre de la tarea: Crear botón UV Checker	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 26 de febrero de 2018	**Fecha de fin:** 26 de febrero de 2018
Programador responsable: Sergio A Quintana Gómez	
Descripción: Desarrollar un boton para la funcionalidad UV Checker.	

A.2. Pruebas de aceptación

Pruebas de aceptación para la iteración 3

Tabla A.21. Prueba de aceptación # 10

Caso de prueba de aceptación	
Código: HU4_P1	**Historia de usuario:** 4
Nombre: Crear botón Desactivar	
Descripción: Prueba realizada a la funcionalidad: Crear botón Desactivar.	
Condiciones de ejecución: 1. El usuario debe estar autenticado en el sitio. 2. El usuario debe haber seleccionado un modelo para visualizar.	
Pasos de ejecución: 1. Como respuesta del sistema el usuario debe poder visualizar en el menú de edición un botón con el nombre Desactivar.	
Resultados esperados: Satisfactorio	

Tabla A.22. Prueba de aceptación # 11

Caso de prueba de aceptación	
Código: HU4_P2	**Historia de usuario:** 4
Nombre: Implementar funcionalidad Desactivar	
Descripción: Prueba realizada a la funcionalidad: Implementar funcionalidad Desactivar.	
Condiciones de ejecución: 1. El usuario debe estar autenticado en el sitio. 2. El usuario debe haber seleccionado un modelo para visualizar.	
Pasos de ejecución: 1. El usuario debe hacer clic izquierdo en el botón Desactivar 2. Como respuesta del sistema el usuario debe poder observar que las opciones de Wireframe que se habian ejecutado ya no se visualizan.	
Resultados esperados: Satisfactorio	

Tabla A.23. Prueba de aceptación # 12

Caso de prueba de aceptación	
Código: HU5_P1	**Historia de usuario:** 5
Nombre: Crear botón White	
Descripción: Prueba realizada a la funcionalidad: Crear botón White.	

Continúa en la próxima página

Tabla A.23. Continuación de la página anterior

Condiciones de ejecución: 1. El usuario debe estar autenticado en el sitio. 2. El usuario debe haber seleccionado un modelo para visualizar.
Pasos de ejecución: 1. Como respuesta del sistema el usuario debe poder visualizar en el menú de edición un botón de color blanco.
Resultados esperados: Satisfactorio

Tabla A.24. Prueba de aceptación # 13

Caso de prueba de aceptación	
Código: HU5_P2	**Historia de usuario:** 5
Nombre: Implementar funcionalidad White	
Descripción: Prueba realizada a la funcionalidad: Implementar funcionalidad White.	
Condiciones de ejecución: 1. El usuario debe estar autenticado en el sitio. 2. El usuario debe haber seleccionado un modelo para visualizar.	
Pasos de ejecución: 1. El usuario debe hacer clic izquierdo en el botón de color blanco del submenú Wireframe. 2. Como respuesta del sistema el usuario debe poder observar el enmallao del modelo de color blanco.	
Resultados esperados: Satisfactorio	

Tabla A.25. Prueba de aceptación # 14

Caso de prueba de aceptación	
Código: HU6_P1	**Historia de usuario:** 6
Nombre: Crear botón Black	
Descripción: Prueba realizada a la funcionalidad: Crear botón Black.	
Condiciones de ejecución: 1. El usuario debe estar autenticado en el sitio. 2. El usuario debe haber seleccionado un modelo para visualizar.	
Pasos de ejecución: 1. Como respuesta del sistema el usuario debe poder visualizar en el menú de edición un botón de color negro.	
Resultados esperados: Satisfactorio	

Tabla A.26. Prueba de aceptación # 15

Caso de prueba de aceptación	
Código: HU6_P2	**Historia de usuario:** 6
Nombre: Implementar funcionalidad Black	
Descripción: Prueba realizada a la funcionalidad: Implementar funcionalidad Black.	
Condiciones de ejecución: 1. El usuario debe estar autenticado en el sitio. 2. El usuario debe haber seleccionado un modelo para visualizar.	
Pasos de ejecución: 1. El usuario debe hacer clic izquierdo en el botón de color negro del submenú Wireframe. 2. Como respuesta del sistema el usuario debe poder observar el enmallao del modelo de color negro.	
Resultados esperados: Satisfactorio	

Tabla A.27. Prueba de aceptación # 16

Caso de prueba de aceptación	
Código: HU7_P1	**Historia de usuario:** 7
Nombre: Crear botón Aleatorio	
Descripción: Prueba realizada a la funcionalidad: Crear botón Aleatorio.	
Condiciones de ejecución: 1. El usuario debe estar autenticado en el sitio. 2. El usuario debe haber seleccionado un modelo para visualizar.	
Pasos de ejecución: 1. Como respuesta del sistema el usuario debe poder visualizar en el menú de edición un botón con el nombre Aleatorio.	
Resultados esperados: Satisfactorio	

Tabla A.28. Prueba de aceptación # 17

Caso de prueba de aceptación	
Código: HU7_P2	**Historia de usuario:** 7
Nombre: Implementar funcionalidad Black	
Descripción: Prueba realizada a la funcionalidad: Implementar funcionalidad Aleatorio.	
Condiciones de ejecución: 1. El usuario debe estar autenticado en el sitio. 2. El usuario debe haber seleccionado un modelo para visualizar.	

Continúa en la próxima página

Tabla A.28. Continuación de la página anterior

Pasos de ejecución: 1. El usuario debe hacer clic izquierdo en el botón Aleatorio del submenú Wireframe. 2. Como respuesta del sistema el usuario debe poder observar el enmallao del modelo de color negro.
Resultados esperados: Satisfactorio

Pruebas de aceptación para la iteración 4

Tabla A.29. Prueba de aceptación # 18

Caso de prueba de aceptación	
Código: HU9_P1	**Historia de usuario:** 9
Nombre: Crear botón Glossy	
Descripción: Prueba realizada a la funcionalidad: Crear botón Glossy.	
Condiciones de ejecución: 1. El usuario debe estar autenticado en el sitio. 2. El usuario debe haber seleccionado un modelo para visualizar.	
Pasos de ejecución: 1. Como respuesta del sistema el usuario debe poder visualizar en el menú de edición un botón con el nombre Glossy.	
Resultados esperados: Satisfactorio	

Tabla A.30. Prueba de aceptación # 19

Caso de prueba de aceptación	
Código: HU9_P2	**Historia de usuario:** 9
Nombre: Implementar funcionalidad Glossy	
Descripción: Prueba realizada a la funcionalidad: Implementar funcionalidad Glossy.	
Condiciones de ejecución: 1. El usuario debe estar autenticado en el sitio. 2. El usuario debe haber seleccionado un modelo para visualizar.	
Pasos de ejecución: 1. El usuario debe hacer clic izquierdo en el botón Glossy del menú de edición. 2. Como respuesta del sistema el usuario debe poder observar la propiedad de suavizado del modelo.	
Resultados esperados: Satisfactorio	

Tabla A.31. Prueba de aceptación # 20

Caso de prueba de aceptación	
Código: HU10_P1	**Historia de usuario:** 10
Nombre: Crear botón Normal	
Descripción: Prueba realizada a la funcionalidad: Crear botón Normal.	
Condiciones de ejecución: 1. El usuario debe estar autenticado en el sitio. 2. El usuario debe haber seleccionado un modelo para visualizar.	
Pasos de ejecución: 1. Como respuesta del sistema el usuario debe poder visualizar en el menú de edición un botón con el nombre Normal.	
Resultados esperados: Satisfactorio	

Tabla A.32. Prueba de aceptación # 21

Caso de prueba de aceptación	
Código: HU10_P2	**Historia de usuario:** 10
Nombre: Implementar funcionalidad Normal	
Descripción: Prueba realizada a la funcionalidad: Implementar funcionalidad Normal.	
Condiciones de ejecución: 1. El usuario debe estar autenticado en el sitio. 2. El usuario debe haber seleccionado un modelo para visualizar.	
Pasos de ejecución: 1. El usuario debe hacer clic izquierdo en el botón Normal Map del menú de edición. 2. Como respuesta del sistema el usuario debe poder observar la propiedad de mapeado normal del modelo.	
Resultados esperados: Satisfactorio	

Tabla A.33. Prueba de aceptación # 22

Caso de prueba de aceptación	
Código: HU11_P1	**Historia de usuario:** 11
Nombre: Crear botón Vertex Normal	
Descripción: Prueba realizada a la funcionalidad: Crear botón Vertex Normal.	
Condiciones de ejecución: 1. El usuario debe estar autenticado en el sitio. 2. El usuario debe haber seleccionado un modelo para visualizar.	

Continúa en la próxima página

Tabla A.33. Continuación de la página anterior

Pasos de ejecución: 1. Como respuesta del sistema el usuario debe poder visualizar en el menú de edición un botón con el nombre Vertex Normal.
Resultados esperados: Satisfactorio

Tabla A.34. Prueba de aceptación # 23

Caso de prueba de aceptación	
Código: HU11_P2	**Historia de usuario:** 11
Nombre: Implementar funcionalidad Vertex Normal	
Descripción: Prueba realizada a la funcionalidad: Implementar funcionalidad Vertex Normal.	
Condiciones de ejecución: 1. El usuario debe estar autenticado en el sitio. 2. El usuario debe haber seleccionado un modelo para visualizar.	
Pasos de ejecución: 1. El usuario debe hacer clic izquierdo en el botón Vertex Normal del menú de edición. 2. Como respuesta del sistema el usuario debe poder observar las normales de los vertices del modelo.	
Resultados esperados: Satisfactorio	

Tabla A.35. Prueba de aceptación # 24

Caso de prueba de aceptación	
Código: HU12_P1	**Historia de usuario:** 12
Nombre: Crear botón Wireframe Sólido	
Descripción: Prueba realizada a la funcionalidad: Crear botón Wireframe Sólido.	
Condiciones de ejecución: 1. El usuario debe estar autenticado en el sitio. 2. El usuario debe haber seleccionado un modelo para visualizar.	
Pasos de ejecución: 1. Como respuesta del sistema el usuario debe poder visualizar en el menú de edición un botón con el nombre Wireframe Sólido.	
Resultados esperados: Satisfactorio	

Tabla A.36. Prueba de aceptación # 25

Caso de prueba de aceptación

Continúa en la próxima página

Tabla A.36. Continuación de la página anterior

Código: HU12_P2	**Historia de usuario:** 12
Nombre: Implementar funcionalidad Wireframe Sólido	
Descripción: Prueba realizada a la funcionalidad: Implementar funcionalidad Wireframe Sólido.	
Condiciones de ejecución: 1. El usuario debe estar autenticado en el sitio. 2. El usuario debe haber seleccionado un modelo para visualizar.	
Pasos de ejecución: 1. El usuario debe hacer clic izquierdo en el botón Wireframe Sólido del menú de edición. 2. Como respuesta del sistema el usuario debe poder observar el enmallado del modelo de color blanco y el modelo mostrando una textura de color rojo.	
Resultados esperados: Satisfactorio	

Tabla A.37. Prueba de aceptación # 26

Caso de prueba de aceptación	
Código: HU13_P1	**Historia de usuario:** 13
Nombre: Crear botón UV Checker	
Descripción: Prueba realizada a la funcionalidad: Crear botón UV Checker.	
Condiciones de ejecución: 1. El usuario debe estar autenticado en el sitio. 2. El usuario debe haber seleccionado un modelo para visualizar.	
Pasos de ejecución: 1. Como respuesta del sistema el usuario debe poder visualizar en el menú de edición un botón con el nombre UV Checker.	
Resultados esperados: Satisfactorio	

Tabla A.38. Prueba de aceptación # 27

Caso de prueba de aceptación	
Código: HU13_P2	**Historia de usuario:** 13
Nombre: Implementar funcionalidad UV Checker	
Descripción: Prueba realizada a la funcionalidad: Implementar funcionalidad UV Checker.	
Condiciones de ejecución: 1. El usuario debe estar autenticado en el sitio. 2. El usuario debe haber seleccionado un modelo para visualizar.	

Continúa en la próxima página

Tabla A.38. Continuación de la página anterior

Pasos de ejecución: 1. El usuario debe hacer clic izquierdo en el botón UV Checker del menú de edición. 2. Como respuesta del sistema el usuario debe poder observar el mapeado de texturas del modelo en las condenada UV.
Resultados esperados: Satisfactorio

A.3. Pruebas de visualización Etapa II

Figura A.1. Prueba realizada para el paso 1: Seleccionar un modelo no usado en pruebas anteriores.

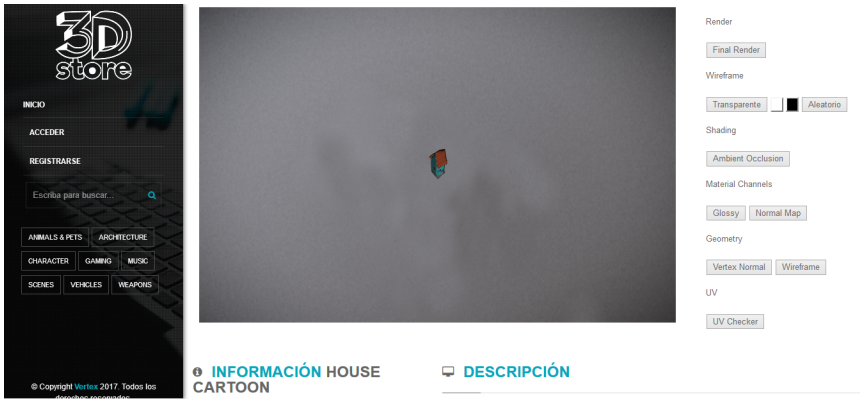


Figura A.2. Prueba realizada para el paso 3: Disminuir tamaño del modelo.

Figura A.3. Prueba realizada para el paso 5: Mover el modelo en diferentes direcciones.

Figura A.4. Prueba realizada para el paso 5: Mover el modelo en diferentes direcciones.

Figura A.5. Prueba realizada para el paso 6: Mostrar mallado del modelo de color blanco.

Figura A.6. Prueba realizada para el paso 7: Mostrar mallado del modelo de color negro.

Figura A.7. Prueba realizada para el paso 7: Mostrar mallado del modelo de color aleatorio.

Figura A.8. Prueba realizada para el paso 9: Eliminar color del mallado mostrado en el paso anterior.

Figura A.9. Prueba realizada para el paso 10: Mostrar Oclusión Ambiental.

Figura A.10. Prueba realizada para el paso 11: Mostrar mapeado normal.

Figura A.11. Prueba realizada para el paso 16: Mostrar el modelo en su vista inicial (Opción Valores por defecto).