UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE CUENCA

FACULTAD DE INGENIERÍAS

Carrera:

Ingeniería de Sistemas.

Tesis previa a la obtención del Título de Ingeniero de Sistemas

Tema:

"Gestor de ambientes virtuales 3D y objetos de información."

Autores:

Nube Katherine Altamirano Calle.
Jhon Javier Jara Matute.

Director:

Ing. Vladimir Robles.

Cuenca, Diciembre del 2011

Nube Altamirano C, Jhon Jara M:
Gestor de ambientes virtuales 3D y objetos de información, Diciembre 2011
Director de Tesis:
Ingeniero Vladimir Robles
CIUDAD:
Cuenca

Acta de Compromiso de Derechos

Los conceptos desarrollados y conclusiones vertidas en este trabajo son de responsabilidad absoluta de los autores.

Jhon Jara M.
0104834254

Nube Altamirano C.
0301913190

Certificado de presentación de tesis.

Ing. Vladimir Robles Tutor de Tesis

CERTIFICA:

Que ha dirigido la tesis diseño de implementación de un gestor de ambientes virtuales 3D y objetos de información , realizada por los estudiantes Nube Katherine Altamirano Calle y Jhon Javier Jara Matute, ha sido revisada en su totalidad y esta lista para ser presentada.

_____

Ing. Vladimir Robles

Agradecimientos Nube Altamirano C.

Agradezco infinitamente a Dios por darme la vida y por sus bendiciones todos los días, a ti Papito gracias por tu apoyo y amor incondicional, por no juzgarme por mis malas acciones, gracias por darme todo lo necesario para poder alcanzar uno de los objetivos en mi vida, a ti Mamita gracias por tus palabras todos los días, gracias por ser mi mejor amiga y darme la confianza que necesite para poder alcanzar este logro, gracias a los dos por ser los mejores padres y por depositar toda la confianza en mí, los quiero mucho, nunca voy a poder pagarles todo lo que hacen por mi.

Gracias a mis hermanos Andrés y Karolina por su apoyo incondiconal y su cariño, gracias a ti Jhon Jara por tu apoyo incondicional, por tu amor, por la paciencia que tienes, y la confianza que depositas cada día en mí.

Gracias a toda mi familia de manera especial a mis tios Carlos, Nelly,Victor e Ines que supieron darme su apoyo en los momentos que mas lo necesite, a mis primos y a todas las personas que de manera directa o indirecta me dieron su apoyo.

Gracias a mi director de Tesis Ing. Vladimir Robles por guiarnos en este proyecto, y estar siempre pendiente del trabajo realizado.

Agradecimientos Jhon Jara M.

Existen muchas personas que supieron ayudarme y apoyarme para cumplir esta meta, pero realmente todo se lo debo especialmente a Dios, a quien agradezco por haberme dado la sabiduría, valentía, esfuerzo, y rodearme de personas que supieron darme sus consejos para salir adelante con este proyecto de Tesis.

Agradezco a mis padres que a pesar de mi comportamiento e inmadurez supieron siempre confiar en mi sabiduría y conocimiento, a quienes su apoyo e influencia ayudo a cumplir la meta de terminar este proyecto de tesis.

Agradezco a mi hermano Pablo Jara por siempre brindarme su apoyo incondicional en todo momento, brindándome la confianza y ejemplo para salir adelante.

Agradezco a un gran amigo Diego Moncayo quien supo aconsejarme siempre y quien ha sido un gran apoyo y ejemplo como profesional para terminar mi carrera.

Agradezco a Nube Altamirano quien supo con cariño y amor darme su apoyo incondicional, quien me ayudo a terminar este proyecto con mucho esfuerzo para lograr culminar esta meta juntos.

Agradezco a nuestro Director de Tesis Vladimir Robles quien nos apoyó siempre, nos ayudó a completar satisfactoriamente nuestro proyecto incentivándonos con su sabiduría y brindándonos su amistad y confianza durante mi esta de estudiante.

Agradezco a la Universidad Politécnica Salesiana institución que me formo y educo para culminar este proyecto, al igual que todos los docentes de la carrera sistemas.

Dedicatoria Nube Altamirano C.

Este proyecto de tesis se la dedico en primer lugar a Dios y la Virgencita, gracias por darme la oportunidad de alcanzar este logro, por darme salud, vida, sabiduria, amor y su bondad. Luego, a mis padres Eduardo y Beatriz por la confianza , el apoyo incondicional y el amor que todos los días me brindan, por velar por mi cada día, por los consejos que me han inculcado para ser mejor, los valores que han sembrado en mi corazón y la motivación que cada día me dan para seguir adelante. A mis hermanos Andrés y Karolina, por apoyarme siempre y darme palabras de aliento para poder alcanzar uno de mis objetivos, por su amor y comprensión. A mi novio y compañero de tesis Jhon Jara M. por apoyarme incondicionalmente y creer en mí, por tus palabras, por la confianza y el amor que siempre me das. A mis abuelitos, Jesús Rodríguez que desde el cielo siempre está conmigo bendiciéndome y que es mi ángel de la guarda, Carlos Calle, Manuel Altamirano y Concepción Cárdenas. A todos mis tíos, por el apoyo moral que siempre me dan y por estar siempre pendientes de mí. A mis amigas y amigos por su amistad y la ayuda que me brindaban cuando los necesitaba.

Dedicatoria Jhon Jara M.

El esfuerzo e ilusión con el cual realizamos este proyecto de tesis es dedicado especialmente a Dios quien supo guiarnos y darnos la sabiduría para cumplir nuestra meta. Se lo dedico con mucho cariño a mis padres quienes supieron siempre apoyarme y aconsejarme dándome siempre el ejemplo de perseverancia y constancia que los caracteriza, y así infundir en mí el valor para salir adelante. A Pablo Jara mi hermano quien supo siempre darme sus consejos, y apoyarme incondicionalmente a pesar de no estar junto a mí, sus palabras y aliento de seguir adelante a más de ser un gran ejemplo en mi vida. A mis familiares quienes supieron aconsejarme y ayudarme directa o indirectamente a realizar este proyecto de tesis. A una gran mujer, amiga, novia y compañera de tesis Nube Altamirano C. quien supo siempre darme buenos consejos, apoyarme incondicionalmente en todo ayudándome a madurar y valorar el esfuerzo de mis padres.

Índice general

Índice de figuras

Capítulo 1

Introducción.

1.1. Justificación.

Con el advenimiento de la educación a distancia, los cursos virtuales, el aprendizaje electrónico (e – learning) y la creación de herramientas informáticas para el soporte del aprendizaje, se ha abierto la posibilidad para que los educandos y maestros puedan compartir un espacio diferente para el intercambio de conocimiento.

En virtud de esto, es fundamental que contemos con aplicaciones que posibiliten realizar interacción virtual con los futuros educandos y maestros, de modo que se pueda acceder a la información que posee nuestra universidad de forma más práctica, intuitiva y apegada a la realidad.

Dada la intención de la Universidad Politécnica Salesiana en mejorar la educación y mantenerse a la vanguardia de la tecnología, se creará un ambiente de interacción virtual con los alumnos, llevando a cabo una primera etapa de un proyecto con grandes expectativas tecnológicas, el cual abarcará la construcción virtual de todo nuestro campus, logrando que los nuevos alumnos puedan conocer y acceder virtualmente a las instalaciones de la universidad. Conforme el proyecto crezca se podrá enriquecer la información virtual que encerrará el modelo final, como por ejemplo: Ubicación georeferenciada, almacenamiento e interacción con base de datos de la universidad, paseo virtual por las instalaciones, información de las instalaciones almacenadas y representadas virtualmente en el modelo final, herramientas que permitan mejorar y ampliar la información del modelo, etc.

En esta fase inicial se presentarán las pautas a seguir, consejos de moldeamiento, facilidad y métodos de interacción con la información a representar. Además se conocerán herramientas que nos pueden facilitar dicho proceso de modelamiento, un análisis final presentará la herramienta recomendada a utilizar en la siguiente fase incluyendo toda su información.

1.2. Descripción de la problemática planteada.

En primera instancia el análisis de cómo llegar al objetivo tecnológico de la universidad para crear una herramienta que permita el modelamiento de ambientes virtuales 3D que puedan obtener objetos tridimensionales de información es el punto clave para el enfoque de la problemática, puesto que la primera fase tiene que satisfacer el asentamiento conceptual de cómo llegar a dicho objetivo. Para lo cual es necesario experimentar con

las herramientas GPL “General Public Licence” de modelamiento virtual que nos ayuden a llegar a nuestro objetivo, dicha experimentación se realizará bajo entornos y escenarios reales, realizando un análisis adecuado del funcionamiento de cada una de las herramientas seleccionadas para el análisis. Dicho análisis y experimentación planteada reflejarán el cómo llegar a realizar la herramienta final, se tendrá que definir la mejor herramienta de modelado 3D cumpliendo con ciertas perspectivas como:

- Escalabilidad de la herramienta seleccionada.
- Personalización y programación de la misma.
- Incorporación en un ambiente de desarrollo.
- Adaptabilidad al objetivo del proyecto.

Al establecer la herramienta adecuada para el desarrollo de la aplicación que cumpla con las características de la problemática planteada se podrá pasar a la siguiente fase que construirá el desarrollo del objetivo específico de visualización de la universidad y los objetos de información.

1.3. Definición de objetivos y alcance del proyecto.

Generales.

- Diseñar un sistema que permita la creación de ambientes virtuales 3D, que puedan contener objetos tridimensionales de información.

Específicos.

- Diseñar un sistema totalmente parametrizable y modular para la creación de ambientes virtuales 3D.
- Utilizar los estándares internacionales para el manejo de entornos virtuales y objetos tridimensionales.
- Emplear herramientas de código abierto para el desarrollo del sistema.
- Establecer líneas de trabajo que permitan llevar a cabo el proceso de investigación e implementación por equipos y tareas independientes.
- Investigar y recopilar información sobre el estado del arte en este tipo de sistemas y tecnologías.
- Establecer un conjunto mínimo de requerimientos de hardware y software para instalar la aplicación de desarrollo de ambientes virtuales.
- Revisar las técnicas de conversión de imágenes bidimensionales a objetos tridimensionales, a fin de establecer si es posible implementar este paradigma para facilitar la creación de los ambientes virtuales.
- Realizar pruebas iniciales de la aplicación, tomando como base parte de la estructura de edificios de la Universidad Politécnica Salesiana.

- Establecer una metodología que permita vincular objetos 3D con las bases de datos o algún protocolo de intercambio de información, a fin de extraer información.

1.4. Descripción de la situación actual de la UPS.

Actualmente la Universidad Politécnica Salesiana, ha iniciado el proyecto de "Gestor de ambientes virtuales 3D y objetos de información" con la investigación y experimentación sobre el ¿cómo?, llegar al objetivo propuesto para mejorar los procesos de educación virtual para los alumnos.

La universidad no cuenta con la información suficiente para el desarrollo de este proyecto por lo que la misma tendrá que ser recolectada. En la fase inicial se realizará el levantamiento de información a partir de fotografías con el afán de realizar pruebas en ambientes reales generando modelos y escenarios 3D representativos de la estructura de la UPS.

Las fotografías obtenidas deberán ser de buena calidad, tratando de cubrir los mínimos detalles para el diseño virtual además de que así permitirán la experimentación con herramientas de conversión de imágenes bidimensionales a imágenes tridimensionales. Una vez recolectada la información necesaria para la experimentación y generación de modelos 3D para la primera fase del proyecto, se cubrirá al 100 % la información requerida para los objetivos planteados. En cuanto a la fase de integración del modelo virtual con objetos de información todo lo necesario se encontrará en la base de datos de la Universidad Politécnica Salesiana.

Capítulo 2

Estado del arte sobre la conversión de imágenes 2D a 3D.

El diseño convencional de imágenes fundamentado en la forma plana, podría compararse con la pintura, donde las formas pueden tener mucha expresividad a través de la luz, el peso, las sombras, combinaciones de colores, etc, pero nunca serán representaciones reales del espacio. El modelado tridimensional es la expresión de la realidad tal cual es en el espacio.

Englobaremos a este proceso todos los recursos necesarios para la creación del proyecto, donde trataremos el modelo tridimensional a partir de representaciones de la realidad en 2 dimensiones.

2.1. Conceptos básicos de modelado 3D.

Una vez que se ha realizado un esquema mental de la representación que se quiere realizar hay que plasmarlo, debemos tener en cuenta varios parámetros que serán detallados a continuación.

2.1.1. Escena. La escena o espacio de trabajo, va ser el entorno que definir toda la información necesaria para identificar y posicionar los modelos, luces y cámaras para su renderización.

La selección de la escena a capturar resultará vital en primera instancia, para poder identificar todo el campo de visión de la zona que interesa documentar. Para ello las zonas deberán ser seleccionadas de manera eficiente, cubriendo 360 grados de la sección que será modelada.

FIGURA 2.1.1. Ejemplo de escena a representar.[**32**]

La representación de los objetos en la escena son creados a partir de fórmulas matemáticas, dividiendo el espacio de trabajo en vistas donde se observan las diferentes proyecciones de los principales elementos que se incluyen en la misma. Los elementos deberán ser editados hasta conseguir el efecto deseado, siendo este el diseño o representación final que exprese el esquema planteado.

2.1.2. Sistemas de coordenadas. Un sistema puede identificarse con las coordenadas en 3 dimensiones del espacio en las cuales tiene lugar la renderización. Este espacio se llama sistema de coordenadas universal, al operar con objetos de la escena, podremos utilizar diferentes sistemas de coordenadas.

Los sistemas de coordenadas utilizados en proyección de imágenes de 3 dimensiones se detallan a continuación.[**29**]

Coordenadas cartesianas 3D.. Se define por dos ejes ortogonales en un sistema bidimensional y tres ejes ortogonales en un sistema tridimensional, que se cortan en el origen O, como podemos ver en la figura 2.1.2.[**15**]

La descripción de escenas tridimensionales requiere usualmente el empleo de un sistema de coordenadas cartesiano.

Los puntos del espacio quedan determinados de forma única por sus tres coordenadas cartesianas (x,y,z) . Las coordenadas 3D quedan exclusivamente para representaciones con más de dos ejes de trabajo simultáneos,[**29**]

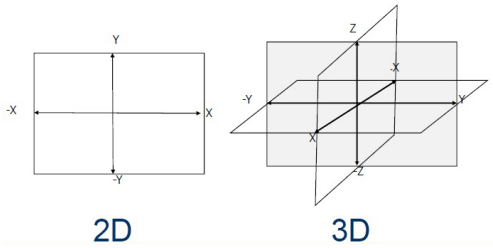

FIGURA 2.1.2. Figura que representan planos en 2D y 3D.[**15**]

Sistemas de coordenadas cilíndricas. El sistema de coordenadas cilíndricas es equivalente en 3 dimensiones a la introdución de coordenadas polares en 2 dimensiones, especifica una coordenada adicional en un eje perpendicular al plano x, y.

Definen puntos mediante la distancia desde el origen en el plano $X-Y$, el angulo desde el eje X en el plano $X-Y$ y el valor Z. Las coordenadas cilíndricas utilizan la siguiente sintaxis.[**15**] , [**29**]

$x<$ [angulo desde el eje X], z.

En la Figura 2.1.3, 5 < 30,6 indica un punto situado a 5 unidades del orígen, 30 grados del eje X en el plano XY y a 6 unidades del eje Z.

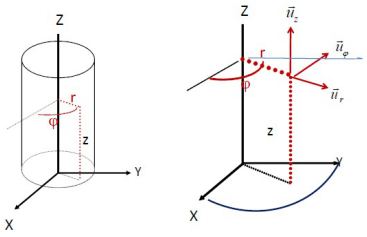

FIGURA 2.1.3. Sistema de coordenadas cilindricas [**15**]

En la Figura 2.1.3 se indica tres vectores unitarios, $\vec{u}_r$ $\vec{u}_\varphi$, $\vec{u}_z$, y perpendiculares entre sí que forman una base ortonormal.

Los sistemas de coordenadas serán utilizados para el análisis del área total del trabajo donde se crearán los objetos.

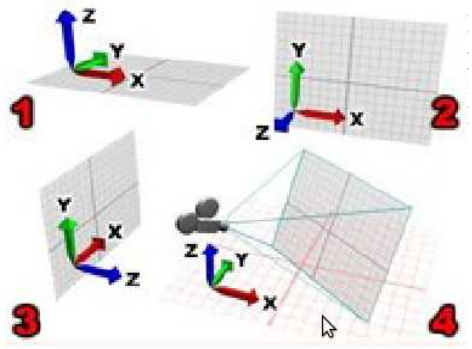

FIGURA 2.1.4. Sistema de coordenadas sobre vista de escenario.

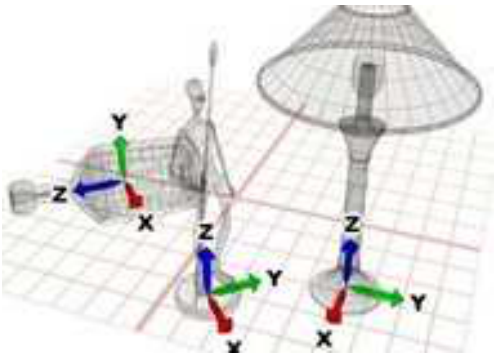

FIGURA 2.1.5. Sistema de coordenadas local a un objeto.

2.1.3. Iluminación de escenas. La iluminación aplicada sobre una escena u objeto específico puede afectar directamente a la percepción de la imagen. La iluminación es una etapa crucial para el éxito de una representación 3D que va a la par con el mapeo de los objetos, porque una buena iluminación puede dar toda su dimensión a la composición. En efecto, la luz es lo que crea el ambiente de autenticidad para las escenas.

Por ejemplo si tenemos un cubo, a cada lado le damos un color diferente esto ayuda al espectador a diferenciar las partes del objeto, al asignar una sombra adecuada a cada lado del cubo podemos obtener no solo la apariencia de un color sólido, sino que también la iluminación mediante la luz, en un ángulo específico como se muestra en la figura 2.1.7, donde la última va un paso mas allá añadiendo una sombra detrás del cubo, dando un efecto de ilusión mucho mas convincente.

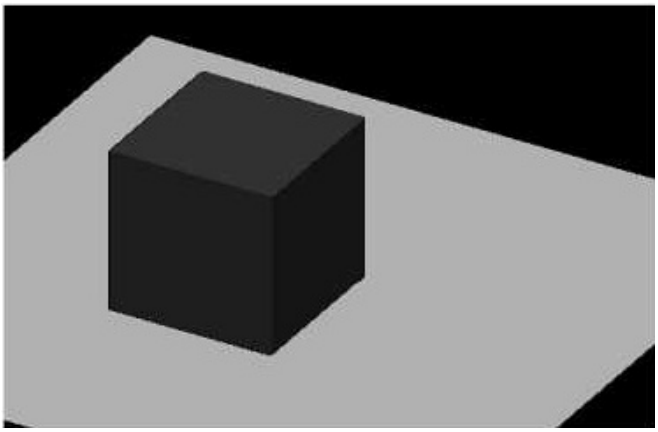

FIGURA 2.1.6. Ilusión de iluminación.

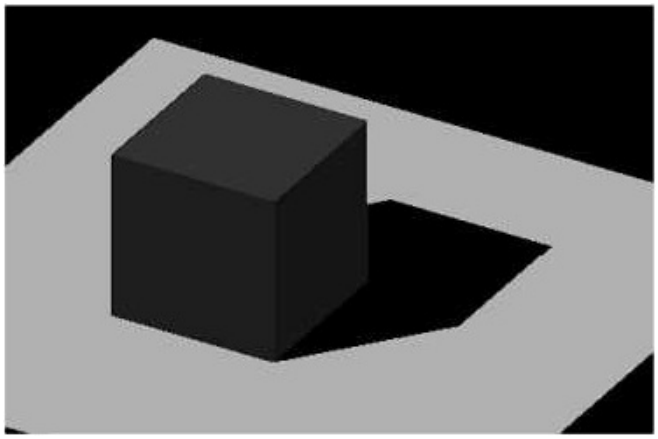

FIGURA 2.1.7. Agregando una sombra para aumentar el realismo.

Tipos de iluminación. A continuación se especifican las principales técnicas de iluminación de manera breve dando a conocer el objetivo principal de cada una.

1. **Omni.-** Son luces que iluminan de igual forma en todas las direcciones desde un punto central, como la función que realiza la bombilla de una casa.
2. **Focal.-**Este tipo de luz es utilizada con bastante frecuencia. Debido a su versatilidad y efectividad este tipo de iluminación permite ajustar el ángulo del foco, aumentándolo o reduciéndolo, para que la luz que se proyecta cubra una parte menor o mayor de la escena.

3. **Ambiental.-** Esta iluminación crea una fuente de luz uniforme y sin dirección, suele utilizarse como luz de relleno en los escenarios que tendrán el efecto de renderización.

Cada uno de los diferentes tipos de iluminación, será utilizado dependiendo de la escena a modelar y las condiciones necesarias para llegar a una representación con mayor realismo.

2.1.4. Materiales y Texturas. Al modelar un objeto, su superficie queda por defecto con un color uniforme y liso. Los materiales y texturas permiten que los objetos adquieran el realismo necesario, es así que primero tendremos que modelar el objeto al que se le aplicará un material o textura.

El diseño que comprende las texturas consiste en aplicar un color o una imagen a un objeto, aunque también las texturas simulan efectos especiales como por ejemplo fuego, explosiones, etc.

Tipos de Texturas. La textura es la utilización de imágenes o texturas pre-diseñadas. Esta definición quiere decir que las texturas y materiales son imágenes tomadas de modelos reales; es decir fotografías, por ejemplo ladrillos, maderas, metales, etc que se aplican a los objetos creados en 3D. Esto permite obtener un realismo basado en fotografías.

Para obtener una buena calidad es conveniente utilizar imágenes de alta resolución a pesar que esto necesite de más memoria en el ordenador.

Las texturas pre-diseñadas también llamadas Shaders que son imágenes generadas por fórmulas matemáticas que darán un aspecto a la superficie bidimensional o tridimensional de los objetos.

La diferencia entre las texturas pre-diseñadas 2D y 3D, es que mientras, que en las de 2D las texturas solo se proyectan en la superficie externa del objeto, en cambio en las de 3D se distribuye por todo el objeto.

Es así que a diferencia de usar materiales de color liso , podemos utilizar materiales y texturas mas representativas de la realidad, esta técnica de la aplicación de una imagen a un objeto, para suministrar detalles adicionales se llama textura geográfica.

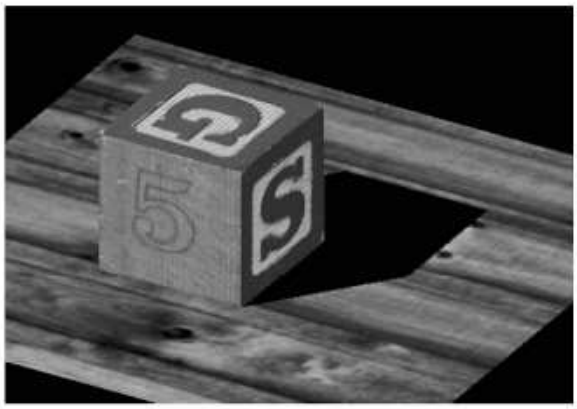

FIGURA 2.1.8. Modelo aplicado texturas.

2.1.5. Render. Es un término usado comúnmente para referirse al proceso de generar una imagen desde un modelo, realizando un procesamiento de polígonos de las imágenes denominado raytracing para mejorar su calidad y realidad.

Al haber completado el diseño y generación de un modelo o escenario 3D, el siguiente paso será la renderización del escenario, para esto se debe establecer las propiedades de los materiales que intervendrán en la escena. Por ejemplo la cantidad de material que refleja el medio ambiente, o a su vez la cantidad de luz que absorbe, estado de la superficie, húmeda, seca, etc. A más de la configuración de la iluminación que tendrá la escena, es decir el tipo de fuente de luz que desea en la escena como el sol, la luna, un foco, etc. Siendo un punto clave para la generación de una imagen realista tomando como base el estudio del medio ambiente que estructura la escena modelada

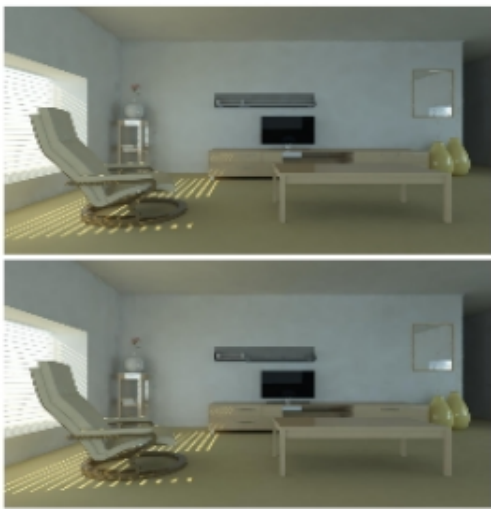

FIGURA 2.1.9. Renderizado de la imagen. [**18**]

En la representacion de la imagen 2.1.9 se aplicaron texturas y procesamiento de renderizado de la imagen obteniendo una mejor calidad de la misma y teniendo resultados muy satisfactorios, cabe recalcar que el mismo tendra un alto costo en recursos.

2.1.6. Geometría en modelaje 3D.. Los objetos desarrollados tienen formas geométricas y quien las define como tal es la geometría de cada objeto creado, realizando reducciones para su definición y representación ya sea de forma amplia o nivel de detalle. Esto permite tener consistencia en la manipulación de los objetos manejados por una Pc. Modelando los objetos consideramos un polígono, el cual puede ser creado de forma cerrada con la coincidencia de inicio y fin de un vértice, que constituyen puntos en común de los objetos creados en 3D. Los programas de creación para modelado 3D ofrecen la manera de manipular formas planas que están establecidas por fragmentos de líneas ya sean rectas, curvas, etc denominadas segmentos.

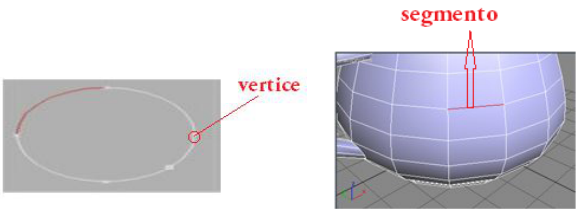

FIGURA 2.1.10. Representación de segmentos en polígonos.

2.2. Estado del arte

Un modelo cargado de grandes volúmenes de información como este caso la construcción de una representación 3D de la Universidad Politécnica Salesiana, requiere de hardware con capacidades de procesamiento gráfico muy alto para su renderización, debido a ser una representación compleja y de gran tamaño. Es así que para esto ha sido necesario tomar en cuenta ciertas limitación en cuanto a no poseer equipos con dichos requerimientos, el proceso de construcción, demostración y visualización del modelo esta limitado en velocidad de procesamiento.

2.3. Técnicas de conversión a objetos 3d.

El modelado 3D se resume siempre a una representación de coordenadas, que conforman estructuras envueltas por una textura. Para una mejor representación podremos figurar, como estructuras de alambre, recubiertas de papel de colores. El objetivo, es realizar la malla de manera simple, para luego crear el material por el cual le daremos sus características tales como, metal, barro, agua, etc. Todo esto para llegar al mismo objetivo que es la representación de una escena virtual, para ello hay técnicas de modelamiento comunes, entre las cuales se encuentran las siguientes:[**50**]

- Estructuras Predefinidas
- Box Modeling
- NURBS Modeling
- Operaciones booleanas
- Extrude and Lathe
- Loft
- Sistema de Partículas
- Modelo por texturas

NURBS se emplea para el modelado de objetos orgánicos, Lathe para objetos iguales en sus lados de un mismo eje, etc. A continuación se expone una breve descripción de cada uno.[**50**]

Estructuras Predefinidas. Se refiere aquellas estructuras ya armadas por el sistema. Existen 3 tipos elementales:

- Primitivas
- Primitivas extendidas
- Librerías[**50**]

Primitivas: Caja, Cono, Esfera, Geo Esfera, Cilindro, Tubo, Anillo, Pirámide, Tetera, y Plano.

Primitivas Extendidas: Hedra, Nudo Toroide, Caja "redondeada", Cilindro "redondeado", Tanque de Aceite, Cápsula, Sprindle, Forma L, Gengon, Forma C, Anillo ondulado, Hose, Prisma.

Librerías: Son formas armadas, disponibles en el sistema como; puertas, ventanas, árboles, escaleras.

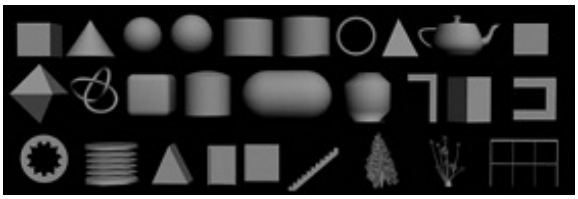

FIGURA 2.3.1. Estructuras para poder modelar objetos o escenas más complejas. [**50**]

Box Modeling. Como su nombre lo indica, es el modelado de figuras complejas a través de una caja, empleando un modificador de mallas, Edith Mesh, para poder extender la caja convirtiéndola en figuras diferentes.

FIGURA 2.3.2. Box modeling. [**50**]

NURBS Modeling. Es una técnica para construir mallas de alta complejidad, de aspecto orgánico o curvado, que emplea como punto de partida splines que hace referencia a figuras 2d para mediante diversos métodos, crear la malla 3d anidando los splines.

FIGURA 2.3.3. NURBS modeling [**50**]

Operaciones Booleanas. Consisten, en tomar dos mallas y aplicarles una de tres operaciones booleanas disponibles: resta, intersección o unión.
Resta: Resta dos figuras $A - B$ o $B - A$, esta operación sirve para quitar de un objeto el volumen de otro.
Intersección: Da como resultado sólo lo que esta "tocándose" de ambas figuras, esta operación permite crear un nuevo sólido a partir del volumen común de dos o más sólidos, esto quiere decir, donde se intersectan los objetos.
Unión: Construye una nueva figura a partir de la combinación de dos o mas solidos. [**2**]

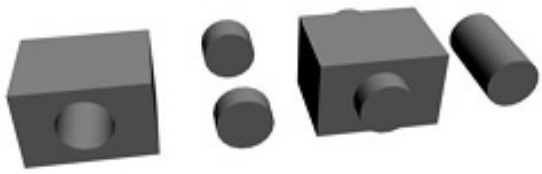

FIGURA 2.3.4. Operaciones booleanas. [**50**]

Extrude and Lathe. Son dos técnicas que a partir de una figura 2d (spline) crea el volumen.
Extrude: Da profundidad a un objeto 2d. Extiende la profundidad.
Lathe: Tomando un spline, lo reproduce por un eje en toda su rotación. Ideal para botellas, copas, y demás objetos sin diferencia en sus costados. Aunque puede combinarse con otra técnica y crear por ejemplo, una taza.

FIGURA 2.3.5. Extrude. [**50**]

FIGURA 2.3.6. Lathe. [**50**]

Loft. Se deben emplear 2 o más splines, para crear una malla 3D continua. El primer spline, funciona como camino mientras que los demás, dan forma, extendiéndose, a través del path. Ideal para crear cables, botellas, etc.

FIGURA 2.3.7. Loft. [**50**]

Sistema de Partículas. Como su nombre lo indica, un sistema de partículas es una proyección de formas geométricas, de forma controlada mediante varios parámetros tales como choque, fricción y demás. Es combinable, con efectos de dinámica y deformadores. Es ideal para crear humo, agua, o cualquier elemento que sean varios objetos y repetitivos.

FIGURA 2.3.8. Sistema de Partículas. [**50**]

Modelo por texturas. Este tipo de modelado en vez de emplear deformadores en la malla, engañan la visualización, con mapas del canal alfa (transparencia) para crear recortes, o engaños directos de relieve por ejemplo, para crear terrenos.

Es un tipo de modelado, usado mucho para abstractos en 3D que no es mas que elementos que tienen forma, color, estructura, etc, es muy fácil de emplear, simplemente se deben manipular los canales para engañar la vista.

FIGURA 2.3.9. Modelo por texturas. [**50**]

2.4. TÉCNICAS BASADAS EN MACHINE LEARNING

Los procesos informáticos basados en machine learning son actualmente utilizados en aplicaciones tales como imágenes médicas, fabricación automatizada y vehículos autónomos. Los métodos para análisis de imágenes de personas han recibido mucha atención y constituyen un área de investigación potencial debido a su gran número de aplicaciones en diferentes campos, tales como, los algoritmos eficaces para la interpretación de los movimientos de las personas que permiten una interacción más natural de la "máquina – humano", seguridad inteligente, sistemas de vigilancia de conducción, etc.

En un principio machine learning se desarrolló como una herramienta de análisis de la bio-mecánica y la investigación médica, sin embargo actualmente se ha convertido en una herramienta cada vez más importante como fuente de datos de animación para el cine y producción de video juegos. Los mejores sistemas están basados en la visión, pero requieren de múltiples cámaras especializadas, cuidadosamente controlando la iluminación, y los trajes especiales con reflectores o marcadores activos conectados a las articulaciones del cuerpo.[**37**]

FIGURA 2.4.1. Multi-cámara óptica estándar (izq). Varias cámaras (der) .[**37**]

En la figura 2.4.1 podemos apreciar una multi-cámara óptica estándar de configuración de captura de movimiento (izquierda) y luego varias cámaras utilizadas para registrar la posición y el movimiento de diferentes articulaciones en el cuerpo (derecha)

2.4.1. Aprendizaje automático y captura de movimiento. Machine Learning utiliza las colecciones de muestras de un proceso eficaz para obtener modelos matemáticos para un proceso. El modelo es a menudo estadístico y por lo tanto útil en procesos difíciles de modelar con exactitud. En la captura de movimiento, la necesidad de modelado preciso y la representación del cuerpo humano puede evitarse aprendiendo a predecir configuraciones 3D, directamente en las imágenes observadas. Para ello es necesario un conjunto de imágenes y sus configuraciones asociadas como datos de entrenamiento.

Una forma de producir tales datos, es el uso de la tecnología existente de captura de movimiento, para grabar los movimientos de los sujetos en la realización de una serie de

actividades diferentes. Estos sistemas pueden proporcionar generalmente la configuración del cuerpo, como un conjunto de ángulos entre los miembros de las principales articulaciones del cuerpo. Las imágenes correspondientes pueden obtenerse mediante el uso de video convencional, sincronizado con las cámaras del sistema de captura de movimiento, o por la representación artificial de imágenes basadas en los datos de captura de movimiento. Es importante el aprendizaje a partir de imágenes para poder representar la forma de su contenido, sin embargo, resulta efectivo el aprendizaje al modelar estadísticas de las imágenes y respuestas básicas sobre la función de ellos, con una colección de imágenes como, la formación de datos para los movimientos reales de los seres humanos significa que captura el conjunto de movimientos típicos que realizan los humanos. [**1**]

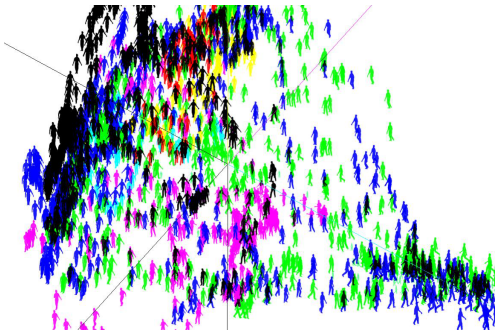

FIGURA 2.4.2. Proyección de siluetas.[**1**]

En la figura 2.4.2 se representa una proyección de la variedad de siluetas humanas en un espacio con características que codifica siluetas con forma de descriptores robustos. Dicha codificación permite representar en 3D los cuerpos y distancias mediante modelos de regresión directa.

Los modelos estadísticos pueden también capturar fácilmente las dependencias temporales y de las correlaciones entre los movimientos de diferentes partes del cuerpo, la tecnología de aprendizaje automático ha comenzado a ser utilizada para sintetizar el movimiento natural en busca de animación humana.

2.4.2. Seguimiento y Regresión. La recuperación del cuerpo humano a partir de imágenes descriptivas es un poderoso método basado en captura de movimiento, dicha regresión es utilizada para obtener estimaciones razonables a partir de la silueta de una imagen.

Sin embargo, al tratar de obtener información fiable a partir de una sola imagen o silueta ocasionalmente se producen errores en la reconstrucción del objeto 3D representativo de la escena.

Por los errores mencionados se reorganizó un método de análisis basado en la captura de imágenes disponibles a partir de un flujo continuo de fotografías o video. Para acelerar

el proceso mencionado en la reconstrucción de escenas 3D y dinámica del movimiento del cuerpo humano, se almacena la probabilidad de representación a cada paso del tiempo, con la finalidad de plantear predicciones dinámicas sobre el modelo.

Modelos de aprendizaje de regresión. La regresión y/o representación de imágenes en modelos 3D basado en la descripción de siluetas, se extiende a la inclusión de modelos dinámicos.[**28**]

Los modelos dinámicos, involucran dos modelos de regresión, un modelo dinámico como tal y un modelo de observación. Los modelos de representación dinámica tienen sus bases alimentadas en modelos matemáticos complejos, que ayudan a la determinación y predicción de resultados, siendo un punto clave para la generación dinámica de resultados en un análisis de escena en tiempo real, estos modelos son los siguientes:

• De predicción dinámica

• De corrección y observación.

Como podemos observar los resultados obtenidos al aplicar los algoritmos de representación dinámica, pueden llegar a ser de gran utilidad al momento de aplicar una generación dinámica de escenas 3D.

2.4.3. Aprendizaje posicionamiento 3D: Regresión en siluetas.

La regresión en siluetas es un método de aprendizaje basado en la recuperación de posturas del cuerpo humano para una representación 3D. El marco de regresión opcional sirve como núcleo en funciones para medir la similitud entre dos imágenes. Para nuestro análisis nos hemos enfocado en el sistema basado en la consideración de siluetas como parámetros de entrada, que se codifican utilizando descriptores de silueta robusta, para obtener la asignación de siluetas para el cuerpo humano la documentación de referencia plantea la distribución y almacenamiento de las propiedades en vectores de regresión, que permiten así un entrenamiento basado en la información almacenada en los mismos, lo que a su vez permite obtener una imagen nueva utilizando solo una fracción de la base de datos de entrenamiento. [**1**].

Representación de las posturas del cuerpo. “Para una recuperación y representación 3D automatizada de la silueta del cuerpo humano, el ambiente de análisis de la escena será estructurada con cámaras en tres ángulos diferentes para cada una de las 18 principales articulaciones del cuerpo permitiendo envolver en 360 grados la escena, tratando de devolver la representación a través de regresiones tomadas a partir de imágenes almacenadas por las cámaras”.[**1**]

Las posturas representadas serán almacenadas en un vector de imágenes, basado en varios parámetros del cuerpo humano donde el sistema podrá aprender a predecir el resultado mediante la información almacenada, a partir de experiencias del sistema que sustancialmente se convertirá en una representación basada en estadísticas de aprendizaje.

La silueta de una persona representada en una imagen podrá ser utilizada para la representación de las posturas basadas en estimación, la ventaja de la estimación de posturas que se basan en siluetas son:

- Pueden extraerse mediante la captura de imágenes fiables y de calidad aceptable para el proceso.
- Son insensibles a agentes externos como la ropa, color y textura.
- Permiten codificar y almacenar gran parte de información útil tanto para la generación de la representación en 3D como para el aprendizaje del sistema. [**28**]

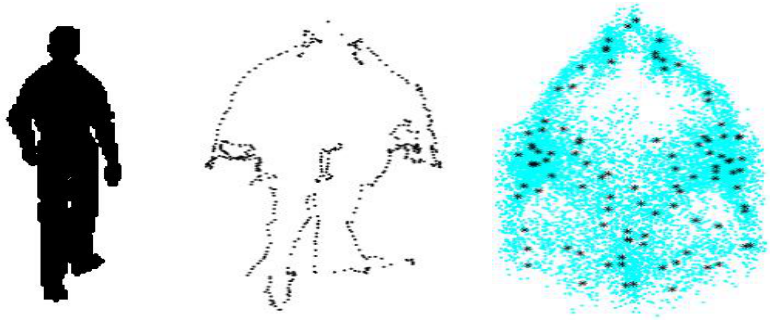

FIGURA 2.4.3. Silueta de codificación. [**1**]

Lo que se representa en la figura 2.4.3 son las siluetas de codificación utilizando los descriptores locales (izquierda), que es codificada como una forma difusa en el espacio (centro) y luego se muestra una proyección sobre los dos primeros componentes principales de la distribución (derecha).
Métodos de regresión. Existen varios métodos de regresión, los cuales se basan en algoritmos matemáticos que optimizan los procesos, no profundizaremos en cada uno de ellos, pero mencionaremos los principales a continuación:[**1**]

- Ridge Regression.
- Relevance Vector Regression Support.
- Vector Regression

2.4.4. Estimación de imágenes basadas en poses.

Al generar la representación de una silueta basada en imágenes, la imagen de fondo puede llegar a representar un grave problema en el momento de tratar de diferenciar la silueta y reconocer en medio de un ambiente cargado. “Esto nos lleva al análisis de imágenes, sin información para la segmentación de la información para llegar a obtener nuestro modelo basado en siluetas para el reconocimiento de posturas, el cual, se convierte en un proceso mucho más complicado aún. El método de regresión basado en el trabajo en general que contiene las imágenes desordenadas exige una codificación adecuada de las características de la imagen.

A diferencia de los métodos top-down que se aplica a una parte de la imagen, que está cubierta por una proyección de un modelo del cuerpo humano, el método bottom-up requiere la capacidad de detectar de forma explícita las partes del cuerpo en una imagen, para una representación adecuada."[**1**]

FIGURA 2.4.4. Estimación de imagenes[**1**]

En la figura 2.4.4 podemos observar la presencia del laboratorio en el fondo de la imagen hace que el problema de estimación de la postura del cuerpo sea significativamente más complicado. Una cuestión importante es diferenciar las partes útiles de la imagen que no sean confundidas con los objetos en el fondo de la misma.

Representación de imágenes. El desarrollo de una efectiva representación de imágenes es un problema conocido en varias ramas de la computación. Aplicaciones sencillas, a menudo hacen uso de las representaciones globales como histogramas de color, pero las tareas más exigentes, como la reconstrucción de posturas o el reconocimiento de objetos, requiere presentaciones más robustas.

Una manera eficaz para garantizar la robustez es mediante la combinación de la información contenida en muchas de las regiones locales de una imagen. El reconocimiento efectivo de imágenes se basa en la capacidad de obtener un subconjunto de las regiones constituyentes de una imagen e identificar el resto de la imagen como irrelevante. [**37**]

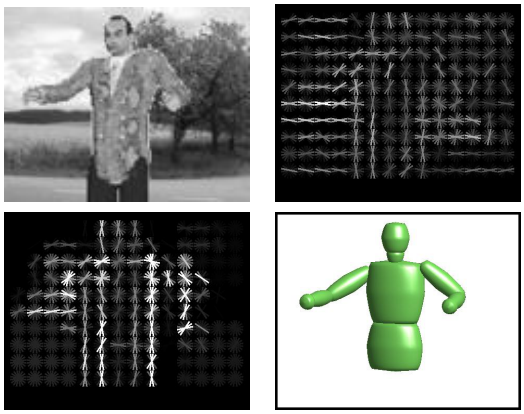

FIGURA 2.4.5. Estimación de posturas a partir de imágenes. [**37**]

2.5. Restricciones de conversión de imágenes.

En este capítulo no únicamente centraremos nuestro análisis en las restricciones de conversión de imágenes, trataremos de involucrarnos en dirección del problema al modelar y reconocer objetos desde fotografías, incluiremos el análisis experimental de los procesos de reconocimiento. La construcción de modelos y escenas 3D están estructuradas en base a una colección de imágenes o fotografías que son capturadas por una cámara, creando una secuencia de fotográficas tomadas preferiblemente cada 20 grados con un máximo de 30 fotografías por colección.

Es necesario tomar en cuenta que el conjunto de imágenes de entrenamiento para un determinado objeto es “escaso” en el sentido de que el cambio entre puntos de vista, es muy amplio, generando complicidad en la adaptación de los objetos, por esto es necesario una colección aceptable de imágenes de buena calidad, teniendo en cuenta, variables importantes como: luz, posición, sombra, etc.

Los enfoques tradicionales geométricos para el reconocimiento de objetos 3D a partir de fotografías originalmente se limitan al reconocimiento de formas simples, como los poliedros, polígonos, etc. que a partir de este análisis de reconocimiento general se han extendido a formas más generales, incluyendo cilindros, superficies algebraicas e incluso superficies de forma libre. Desafortunadamente la complejidad de formación de hipótesis o resultados aceptables se han limitado el éxito de las técnicas de reconocimiento puramente geométricas, en escenas desordenadas. Por el contrario las técnicas basadas en la apariencia, enriquecen su información en descriptores locales de la imagen.

FIGURA 2.5.1. Modelos estimados por el software de reconocimiento.[**20**]

En la figura 2.5.1 podemos observar los resultados de un experimento de reconocimiento. (Izquierda), una imagen de prueba. (Derecha).

Modelamiento. Presenta un enfoque automatizado a la adquisición de modelos 3D, a partir de una colección de fotografías, los modelos constarán de una colección de superficie 3D en forma de paralelogramos. En el siguiente ejemplo se utiliza el oso de peluche representado en la figura 2.5.2 para ilustrar el proceso a seguir en el modelado, mediante el establecimiento de encuentro de patrones de dos en dos lugares de imágenes de entrada.

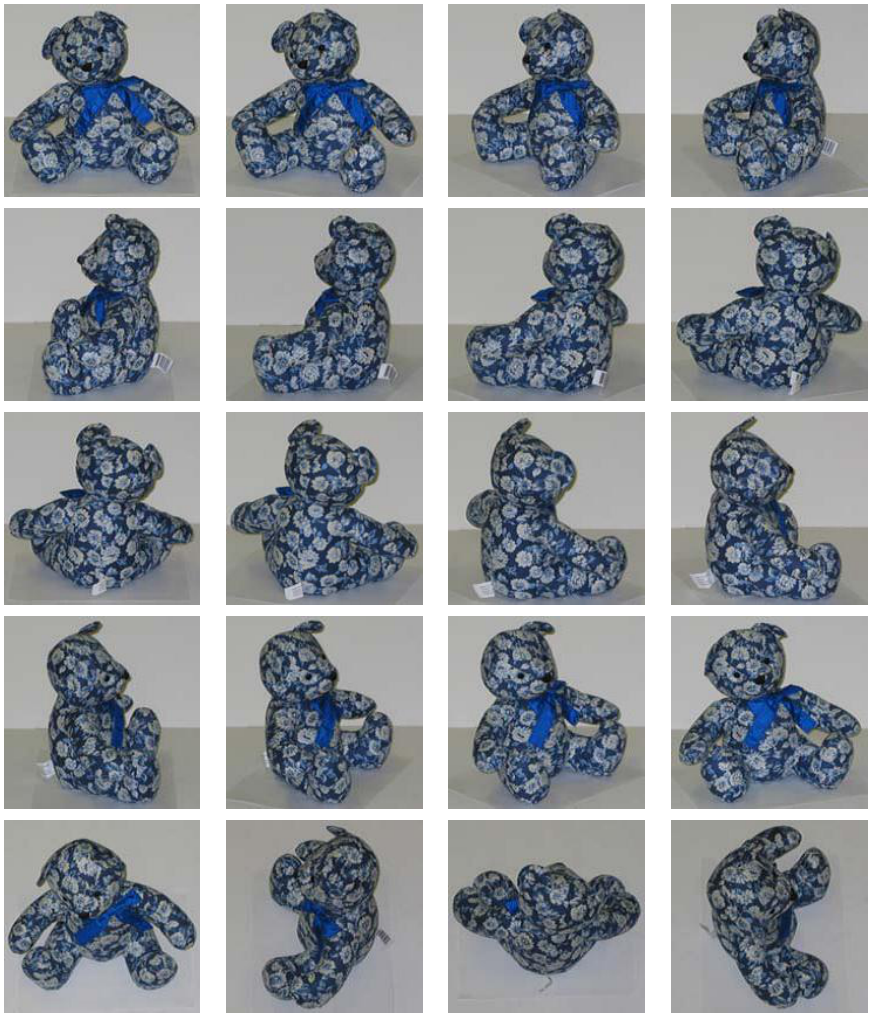

FIGURA 2.5.2. Imagenes utilizadas para construir un modelo basado en fotografias[**19**]

Luego se conectan estos patrones para formar un conjunto global de patrones a través de todas las imágenes. Por lo tanto el proceso constituye modelos a partir de subconjuntos de datos.

En la figura 2.5.2 observamos 20 imágenes que se utilizan para construir un modelo basado en una colección de fotografías de un osito de peluche, existen 16 imágenes tomadas alrededor del anillo ecuatorial, y 4 imágenes de la vista superior. Esta configuración con algunas variaciones en número de imágenes de entrada es típica de los experimentos de nuestro modelo.

2.6. Pruebas de campo.

Pruebas de imágenes basadas en objetos geométricos. En esta sección realizaremos las pruebas de campo respectivas en cuanto a los temas analizados y consultados, para

realizar un test real y poder alcanzar el objetivo planteado, hemos iniciado nuestro trabajo tomando como referencia parte de la infraestructura de la UPS. Para lo cual tomamos de referencia la siguiente escena (figura 2.6.1).

FIGURE 2.6.1. Entrada Universidad Politécnica Salesiana.

Esta representación tomará en cuenta los conceptos planteados en técnicas de conversión de imágenes y métodos de representación de modelos 3D.

Nuestro modelo empezará por la construcción de una escena 3D basada en la fotografía planteada de referencia, la estructura se construirá en base a figuras geométricas debido a la naturaleza de su estructura, que al unirse podrán llegar a representar nuestro objetivo, es así que luego de realizado el proceso de representación del modelo inicial, llegamos a lo que se representa en la figura 2.6.2.

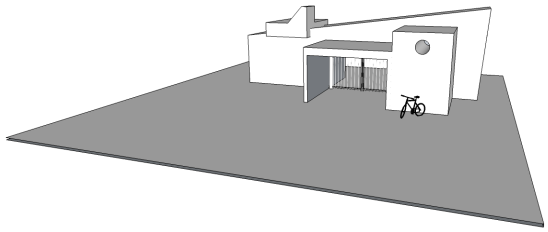

FIGURA 2.6.2. Reconstrucción a partir de figuras geométricas.

Una vez construido nuestro escenario a representar, se aplicarán las texturas para aumentar el realismo de la escena, es así, que tanto colores y texturas de los sectores representativos de la escena serán cargados específicamente. Los colores se han referenciado en una paleta de colores estándar, que mas se asemeje al objetivo que queremos llegar, las texturas de metal y lastre son texturas basadas en fotografías, aplicadas en los sectores necesarios para no sobrecargar la escena con texturas innecesarias.

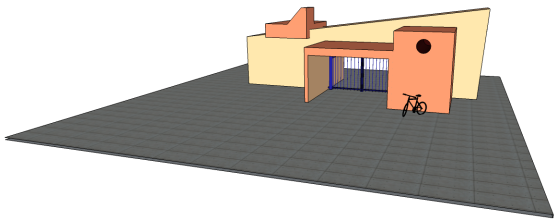

FIGURA 2.6.3. Escenario aplicado textura.

Como podemos observar la representación de nuestra escena ha tomado un enfoque aumentando el realismo de nuestro objetivo, pero aún podemos mejorar dicha escena, tanto las texturas como los colores, los podemos llevar a otro nivel en cuanto a calidad de imagen, aumentado la calidad de pixel y de las texturas, para que el modelo pueda llegar a representar un escenario aceptable.Todo el proceso planteado ha sido realizado con una herramienta de renderización llamada *Índigo Render,* la cual es de libre distribución y puede llegar a construir imágenes con renderización de alta calidad para los modelos creados.

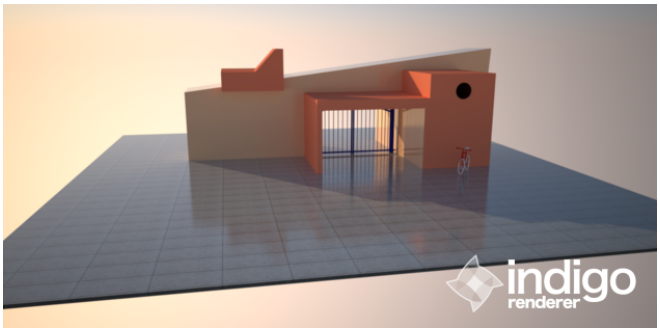

FIGURA 2.6.4. Procesamiento de renderizacion de la imagen.

Pruebas basadas en reconstrucción de objetos 3D a partir de colección de fotografías.

En esta etapa se plantearon pautas importantes para poder llegar a escoger un buen software base que ayude a la experimentación y pruebas. En base a la información descrita en capítulo anterior, el software que cumple experimentalmente con requisitos válidos, en colecciones de imágenes es Strata Foto 3D con una versión libre que permite obtener resultados sorprendentes. Para el análisis nos basamos en un objeto pequeño, del cual se obtuvo una colección de fotos, tomando recomendaciones de los requerimientos antes plateados en el *capítulo 2.5*, es importante recalcar que el software seleccionado se basa en los mismos principios, realizando un análisis y reconstrucción a partir de un grupo de fotografías. Estas fotografías fueron capturadas cada 20 grados, rotando el contorno

céntrico de la figura, además de tomas superiores del objeto, tratando de brindar toda la información necesaria al software para realizar un análisis y reconstrucción exitosa.

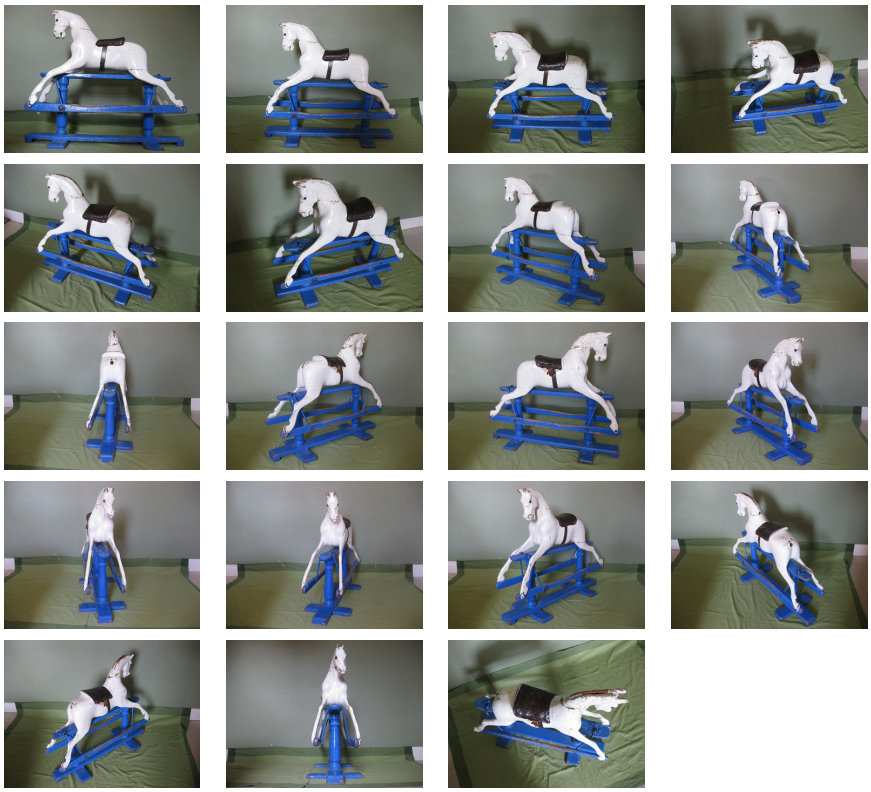

FIGURA 2.6.5. Colección de fotografías capturadas del objeto a análisis de prueba.

El resultado del análisis de la colección de fotos presentada en la figura 2.6.6, demostró una gran utilidad de la herramienta, ya que con este método se puede llegar a representar modelos en 3D con gran rapidez, texturizando directamente en base a la relación de imágenes de referencia, construyendo una representación concreta y con un mínimo de imperfectos aceptable, a continuación se presenta el proceso de representación.

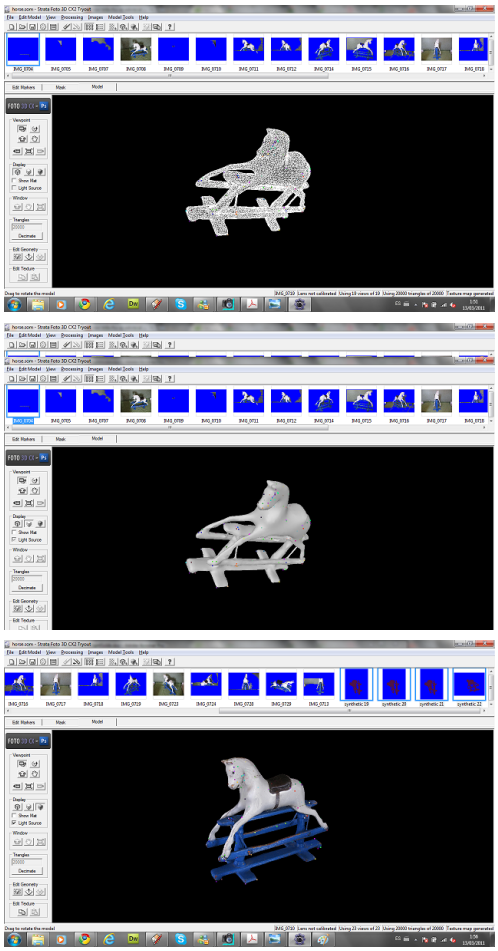

FIGURA 2.6.6. Proceso de representación de escena.

2.7. Conclusiones y síntesis de información obtenida.

El seguimiento de objetos para la reconstrucción 3D es una de las problemáticas más estudiadas, sin embargo es una de las tareas con más enfoques a estudiar.

El estudio del seguimiento y regresión (punto 2.4.2) se puede tratar independientemente del proceso de reconstrucción, ya que en sí los métodos encontrados hacen referencia a cada uno de los procesos, pero no requieren el manejo de un mismo tipo, para efectuar ambos procesos. Con lo cual se puede decir, que el problema real, radica en la conexión entre el módulo de reconstrucción y seguimiento y el consumo de recursos de hardware y software que necesite cada uno [**28**].

Entre los resultados que se encontraron en los diferentes métodos se puede decir, que cada cual se encarga de solucionar una serie de dificultades, dependiendo del problema que se tiene para llegar a solucionar el objetivo planteado.

Los métodos de modelado 3D son varios, pero al mismo tiempo cada uno de los métodos tienen su enfoque al tipo de modelo que se va reconstruir en 3D, es así que basados

en el tipo de estructuras, que conforma el objeto a modelar, será pertinente escoger el tipo de modelado más adecuado, que pueda facilitar la reconstrucción de la escena.

Para acelerar un proceso de creación y representación de escenas 3D es importante considerar que cuando se crean dichas escenas, un factor que afecta considerablemente al tiempo de procesamiento es la cantidad de fotos, y todo el espacio físico del objeto a representar, causando problemas en elementos de gran tamaño. Todos los métodos expuestos anteriormente se basan en los parámetros básicos de estructuración de modelos 3D, iluminación, texturas y renderización para llegar a un efecto mucho más cercano a la realidad.

2.8. Paradigmas obtenidos para la creación del sistema.

Para el desarrollo de la aplicación final se deben tomar en cuenta varios aspectos importantes en cuanto a los parámetros de construcción del sistema. A continuación enumeraremos los elementos críticos a tomar en cuenta en la fase de desarrollo de la aplicación final.

- La aplicación final debe enfocarse al desarrollo de un sistema que contenga la funcionalidad necesaria para la construcción de estructuras geométricas, basadas en la infraestructura neta de los edificios del campus de la Universidad Politécnica Salesiana.
- El sistema debe implementar la funcionalidad necesaria para el manejo de texturas, basadas tanto en colores como texturas con origen en imágenes, además de proporcionar la edición de las mismas, permitiendo cubrir todas las necesidades del usuario final. Sin embargo se debe desplegar la funcionalidad planteada controlando un consumo adecuado de memoria RAM y caché necesaria para que soporte la característica planteada.
- En cuanto al enfoque de tratar de acelerar el proceso mediante el reconocimiento de imágenes, demostró ser de gran utilidad, aun así debemos recordar que el objetivo es utilizar una aplicación que permita virtualizar el campus de la universidad, es así que por condiciones externas al desarrollo del proyecto como el espacio, luz, sombras, y perspectivas necesarias para un respuesta adecuada de la funcionalidad planteada, no es útil el desarrollo o incorporación de esta característica planteada.

Capítulo 3

Lenguajes para modelado de ambientes virtuales 3d.

En esta sección realizaremos un análisis de las diferentes API's especializadas para facilitar los procesos en todas las etapas de la generación de gráficos por computadora. Aquí tendremos en cuenta su nivel de integración con lenguajes de programación de alto nivel además de la descripción general de los mismos, de esta manera seleccionaremos la mejor opción.

3.1. Lenguajes de Programación GPL de ambientes virtuales 3D.

El modelamiento 3D se ha convertido en algo muy popular y útil, especialmente por haber extendido sus campos de aplicabilidad es así que en la actualidad su aplicación ha llegado a formar parte de la publicidad, cinematografía, videojuegos, arquitectura, medicina entre otros, al punto que se han creado API's destinadas a cubrir las expectativas de desarrollo. Estas API's han demostrado ser vitales para los desarrolladores de hardware para gráficos por computadora, ya que proveen un camino al programador para acceder al hardware de manera abstracta, aprovechando las ventajas de la tarjeta gráfica.

Las siguientes API's para gráficos por computadora son particularmente populares:

- OpenGL.
- Direct3D (subconjunto de DirectX para producir gráficos interactivos en 3D).
- XNA.
- RenderMan.
- VRML.

Todos los API's listados son los más utilizados en la actualidad para el desarrollo de aplicaciones basadas en modelamiento 3D, el objetivo principal para la selección del lenguaje más adecuado es que su licencia sea GPL es así que se descartan Direct3D y XNA puesto que están basados en la plataforma Microsoft, que es de pago.

En cuanto a OpenGL, RenderMan y VRML dichos lenguajes tienen licencia GPL, puesto que RenderMan es un lenguaje basado específicamente en la estructura de OpenGL, trataremos los lenguajes base y más eficientes que son OpenGL y VRML. [**48**]

3.2. OPENGL.

OpenGL (Open Graphics Library) es una interfaz de software para gráficos basada en el consumo optimizado del hardware. La interfaz se compone de un conjunto de cientos

de procedimientos y funciones que le permiten a un programador especificar los objetos y operaciones necesarias para la producción de imágenes gráficas de alta calidad, en concreto imágenes de objetos tridimensionales.

Es una especificación estándar que define una API multilenguaje y multiplataforma para escribir aplicaciones que produzcan gráficos 2D y 3D. Fue desarrollada por Silicon Graphics Inc. (SGI) en 1992. Las funciones y comandos de esta API están integrados en las tarjetas de aceleración gráfica 3D, o en su defecto son emulados por el sistema operativo. Es claro que para el propósito específico de modelado 3D no se debe realizar emulación por software sino se debe tener acceso directo al hardware. [**27**]

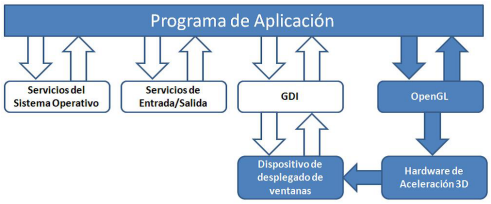

FIGURA 3.2.1. Aplicación utilizando OpenGL y aceleración gráfica por hardware. [**49**]

En la figura 3.2.1 se muestra una aplicación utilizando OpenGL y la aceleración gráfica por hardware; se puede observar que las llamadas a la interfaz de aplicación OpenGL pasan directamente al hardware de aceleración y no pasan a través de la interfaz del dispositivo gráfico (GDI) que sería el caso si se tratara de simulación por software.[**49**]

3.2.1. Características. OpenGL está diseñado de forma completamente independiente del hardware por lo que puede implementarse en plataformas muy diversas (PC, SGI, Digital, Sun, etc) y con diversos sistemas operativos (Linux, Windows, Unix, etc). El precio que se tiene que pagar tras la característica de portabilidad, es que OpenGL no incluye comandos para gestionar el sistema de ventanas o interfaz de usuario, ni para captura de órdenes de usuario, ya sea por mouse o por teclado. En lugar de esto, debemos trabajar con la ayuda de cualquier sistema de interfaz de usuario que se utilice en el sistema operativo sobre la cual estemos desarrollando nuestra aplicación, es así que depende de la integración con un lenguaje de alto nivel.[**41**]

El estándar de gráficos más adoptada.

OpenGL es el primer ambiente para desarrollar aplicaciones interactivas gráficas 2D y 3D. Subsecuentemente a su introducción en 1992, OpenGL se ha vuelto la industria más usada ampliamente 3 y ha brindado aplicaciones gráficas con interfaz de programa(API) 2D y 3D , trayendo a miles de aplicaciones a una variedad de distintas plataformas de computadora. OpenGL adopta el desarrollo veloz e innovador de aplicaciones incorporando un amplio conjunto de herramientas de rendering, el trazado de textura, efectos

especiales, y otras funciones de poderosas de visualización. Diseñadores pueden descubrir el poder de OpenGL a lo largo de todas las populares plataformas de workstations, asegurando un amplio despliegue de aplicaciones.[**33**].

La alta calidad visual y desempeño. Cualquier aplicación de la informática visual que requiere el máximo desempeño en animación 3D en CAD para simulación visual, puede aprovechar las capacidades de calidad superior de OpenGL. Estas capacidades permiten a diseñadores en los diversos mercados tales como televisión, CAD/CAM/CAE, entretenimiento, visualización médica, y realidad virtual para producir y desplegar de manera obligada gráficos 2D y 3D increíbles.

Estable. Las aplicaciones de OpenGL han estado disponibles por más de siete años en una variedad distinta de plataformas. Contribuyentes a la especificación son el buen control, y las actualizaciones propuestas anuncian a tiempo a los diseñadores a adoptar los cambios. Los requisitos de compatibilidad aseguran que estas aplicaciones existentes no se vuelven obsoletas.

Evolucionando. Debido a su completo y futurístico plan, OpenGL permite nuevas innovaciones del hardware para ser accesibles a través del API vía los mecanismos de extensión. De esta manera, las innovaciones aparecen en el API en una moda oportuna, permitiendo a los diseñadores de aplicaciones y a vendedores de hardware incorporar los nuevos rasgos en sus productos lanzados.

Escalable. Las aplicaciones basadas en OpenGL pueden ejecutarse en sistemas que van de PC's convencionales, hasta súper computadores como servidores IBM de alto rendimiento. Como resultado, las aplicaciones pueden escalarse a cualquier clase de máquina que el diseñador escoja.

Fácil usar. OpenGL se estructura bien con un plan intuitivo y comandos lógicos. Las eficientes rutinas de OpenGL típicamente resultan en las aplicaciones con menos líneas de código que aquellas donde se hizo programas usando otras bibliotecas de gráficos o paquetes. Además, los usuarios de OpenGL encapsulan la información sobre el hardware subyacente, liberando al diseñador de la aplicación de tener que diseñar para los rasgos del hardware específicos.

Portabilidad. La biblioteca OpenGL está desarrollada en ANSI C para garantizar la compilación en diferentes plataformas. Además existen una serie de versiones pre compiladas para Windows, Macintosh y ambiente X.

Documentación. Se han publicado numerosos libros sobre OpenGL, y hay mucho código como muestra disponible, haciendo de la información sobre OpenGL fácil obtener y a bajo costo. La documentación e información sobre las versiones comerciales y de dominio público de la biblioteca OpenGL se encuentra en el sitio Web OpenGL Documentacion.

Capacidades de generar gráficos y render. OpenGL permite la descripción matemática de objetos a partir de sus primitivas. Permite la organización de estos objetos en el espacio al tiempo que permite modificar el punto de vista y las propiedades de la cámara.

Permite calcular el color de los píxeles por asignación directa de color, por cálculo de iluminación, por mapeo de texturas o por una combinación de los tres. Permite convertir la descripción matemática de objetos en formas y colores. Permite también la simulación de "efectos atmosféricos" como cubo de profundidad, transparencia, neblina y render volumétrico. [**33**]

Capacidad de desarrollo de interfaz. OpenGL no permite el desarrollo de interfaces gráficas por si solo. [**33**]

3.2.2. Requerimientos. OpenGL no requiere en particular ningún hardware, pero existe en el mercado una gran cantidad de tarjetas aceleradoras graficas que incluyen las rutinas del OpenGL en el hardware, lo que incrementa notablemente la eficiencia en gráficos en aplicaciones de tiempo real. El OpenGL requiere de compiladores C, C++ y de FORTRAN que puedan llamar rutinas compiladas en C. Normalmente se requiere una distribución de OpenGL que incluya el software que lo hace compatible con el ambiente gráfico que se está utilizando, por ejemplo GLX para correr en ambiente X o WinGL para correr en máquinas Win9x o win NT.[**33**]

En cuanto a los requerimientos de software OpenGL para el desarrollo de un aplicativo complejo necesita de varias extensiones que aumentan su capacidad de desarrollo entre los cuales tenemos los siguientes.

Glu. GLU es el acrónimo de OpenGL Utility Library (Librería de utilidad de OpenGL). Esta librería está compuesta por una serie de funciones de dibujo de alto nivel que, a su vez, se basan en las rutinas primitivas de OpenGL. Normalmente se distribuye con esta librería. Las funciones de GLU se reconocen con solo verlas ya que todas comienzan con el prefijo glu, por ejemplo: gluOrtho2D(), que define una matriz de proyección de dos dimensiones.

Glut. GLUT (OpenGL Utility Toolkit) es una sencilla API multiplataforma que provee una reducida funcionalidad para el manejo de ventanas e interacción por medio de teclado y ratón. Su utilidad 6 principal es permitir el desarrollo de aplicaciones con OpenGL sin demasiado esfuerzo, dejando a un lado la complejidad de otros sistemas de ventanas.

Glui. GLUI es una librería de interfaz de usuario escrito en C++. Es el acrónimo de "OpenGL User Interface Library" (Librería de interfaz de usuario de OpenGL). Está basado en GLUT, y proporciona elementos de control tales como botones, cajas de selección y spinners para aplicaciones que usen OpenGL. Es independiente del sistema operativo, sustentándose en GLUT para manejar los elementos dependientes del sistema, como ventanas

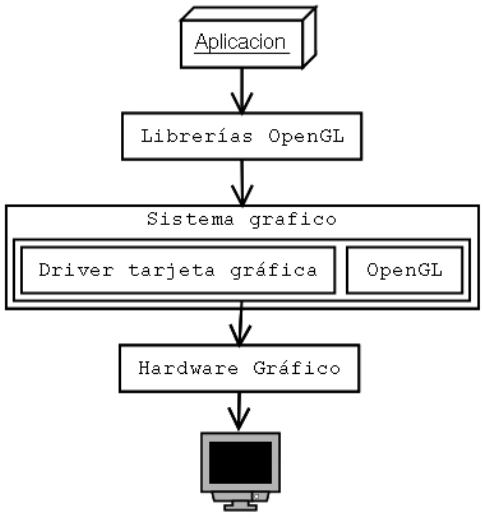

FIGURA 3.2.2. Generación de procesos basados en OpengGL.[**7**]

y el control del ratón. En sí esta librería permite al usuario obtener una interfaz más optimizada y bajo una perspectiva de aplicación común con controles que facilitan su uso. Fue escrito, en un primer momento, por Paúl Rademacher para facilitar su trabajo académico. Ahora mantiene el proyecto Nigel Stewart. [**33**]

3.2.3. Desempeño en aplicaciones de escritorio. Al hablar sobre el desempeño en aplicaciones de escritorio bajo el API de OpenGL, es necesario explicar el funcionamiento de OpenGL, cuáles son sus funciones y como llegarlo a aprovechar de la mejor manera.

Es así que la eficiencia y portabilidad de OpenGL está basada en que su funcionamiento radica como una interfaz entre el programador y el Sistema Operativo. Es decir OpenGL no se encarga de la creación de los gráficos de alta calidad, su trabajo consiste en ser un intermediario enviando órdenes al Sistemas Operativo a partir de una aplicación. Dicho proceso se lo representa en el siguiente gráfico. [**7**]

OpenGL se ha convertido en la base de algunos lenguajes y aplicaciones para el desarrollo de aplicaciones 3D con la capacidad de obtener el mejor desempeño y consumo de recursos del hardware actual, tratando de aprovechar de la mejor manera todos los recursos del computador.

Es así que OpenGL está diseñado para crear ambientes y escenarios 3D dejando el desempeño de lado del hardware mediante el consumo del mismo desde un lenguaje de programación, desde una simple aplicación como la que observamos a continuación. [**7**]

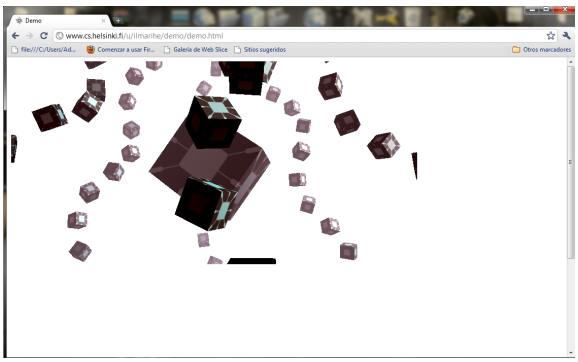

FIGURA 3.2.4. Generación de cubos aleatorios corriendo bajo google chrome 11.0

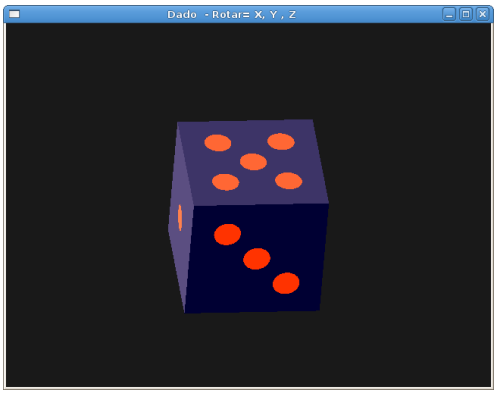

FIGURA 3.2.3. Aplicación básica realizada con OpenGL ES.

OpenGL permite realizar aplicaciones mucho más complejas y avanzadas como programas de virtualización, experimentación e inclusive juegos de video. Las aplicaciones gráficas bajo el API de OpenGL han llegado a formar parte de software muy importantes y reconocidos a nivel mundial como Blender y Google Sketchup, las mismas que son aplicaciones de escritorio con funcionalidades extraordinarias aprovechando al máximo los recursos de nuestro computador.

3.2.4. Desempeño en aplicaciones Web. El desempeño de OpenGL sobre aplicaciones web está definido por su extensión denominada WebGL. WebGL es una representación inmediata del API 3D diseñado para la web. Se deriva de OpenGL ES 2.0, y ofrece similares prestaciónes de funcionalidad, pero en un contexto de HTML. WebGL es diseñado como un marco de representación para el elemento Canvas HTML.

Teniendo en cuenta los casos de uso sobre los gráficos 3D, WebGL elige el enfoque de proporcionar primitivas flexibles que pueden aplicarse a cualquier caso de uso. Las bibliotecas pueden ofrecer una API integración con otros lenguajes a partir de WebGL como base, añadiendo así una conveniencia al desarrollo con WebGL que puede acelerar y simplificar el desarrollo. Sin embargo, debido a su herencia de OpenGL ES 2.0, es

sencillo para los desarrolladores familiarizados con los entornos de escritorio OpenGL o OpenGL ES 2.0.

Al igual que sus predecesores WebGL nos permite obtener la mayor ventaja en cuanto a consumo de recursos de nuestro hardware de desarrollo 3D, logrando una interacción directa desde el cliente (browser) obteniendo una navegación virtual sobre internet. Las aplicaciones realizadas bajo este API podrán extenderse en todos los niveles de complejidad.

WebGL a pesar de ser un complemento de OpenGL relativamente nuevo ha logrado una favorable acogida en este entorno de desarrollo, llegando a formar parte de los principales navegadores que existen en la actualidad como complemento e intérprete de gráficos 3D.

WebGL creció desde los experimentos del canvas 3D comenzados por Mozilla. Mozilla primero demostró un prototipo de Canvas 3D en 2006. A finales de 2007, tanto Mozilla2 como Opera3 habían hecho sus propias implementaciones separadas. A principio de 2009 Mozilla y Khronos comenzaron el WebGL Working Group (Grupo de Trabajo del WebGL). El Grupo de Trabajo del WebGL incluye Apple, Google, Mozilla, y Opera, y WebGL ya está presente en los builds nocturnos de Mozilla Firefox 3.7,4 Mozilla Fennec 1.0 y Google Chrome 4 developer previews, y también en la versión de Safari incorporada en Mac OS X Lion Developer Preview (Safari 5.1). [**16**]

Estas investigaciones empezaron a realizar experimentaciones de alto nivel, tratando de probar el desempeño del API sobre aplicaciones complejas y de gran utilidad es así que Google creo un sitio propio dedicado a dichas investigaciones, el website donde podemos encontrar toda la información esWebGL.

La aplicación de este API puede llegar a mejorar nuestro nivel de educación logrando estar al alcance de todas las personas información interactiva y virtual. Estos experimentos en nuestra investigación nos sorprendieron llegando a demostrar el gran aporte que pueden llegar a dar a la comunidad como se muestra en las imagenes de la figura 3.2.7.

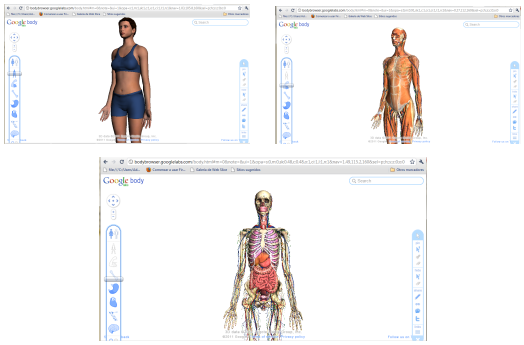

FIGURA 3.2.5. Google Body desarrollado por chrome experiments [**6**]

3.2.5. Pruebas. La etapa de prueba la dividiremos en pruebas sobre aplicaciones de escritorio y aplicaciones web, las mismas están realizadas con los mismos recursos y sobre diferentes sistemas operativos para medir su desempeño sobre diferentes plataformas.

Pruebas OpenGL aplicaciones de escritorio. En esta sección realizaremos un test utilizando las librerías de OpenGL. El análisis de las herramientas que nos provee la librería OpenGL, han demostrado ser de alta calidad, trataremos de mostrar los resultados del consumo de recursos tanto en memoria RAM como en consumo de CPU de las aplicaciones.

Recursos.

- Procesador: 2x Intel(R) Core(TM)2 Duo CPU T5750 @ 2.00GHz.
- Memoria: 3087MB.
- Resolución: 1280x800 pixels.
- OpenGL Renderer: Activado.
- X11 Vendor: The X.Org Foundation.
- VGA Display Controller: Intel Corporation Mobile GM965/GL960 Integrated Graphics Controller.

Bajo el sistema operativo Linux utilizamos herramientas integradas en el mismo para medir el consumo de recursos de la máquina. En este caso utilizamos la aplicación de consola sysstat para verificar el consumo de recursos de nuestra máquina.

Aplicaciones en prueba. Las aplicaciones han sido probadas en diferentes S.O., e incluso diferentes lenguajes de alto nivel. Nuestra primera prueba fue realizar un modelo sencillo en Java con la librería JOGL que permite el consumo de las librerías OpenGL desde Java como lenguaje base. Esta pequeña aplicación contiene la principal funcionalidad como zoom, perspectivas, texturas, rotación, sombras, etc.

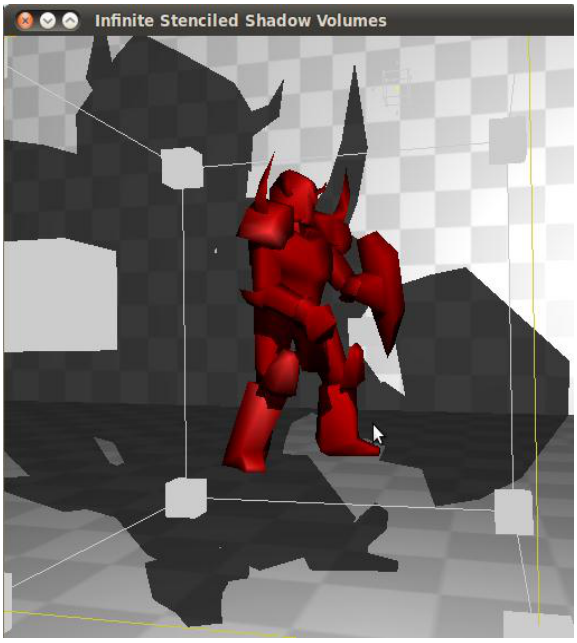

FIGURA 3.2.6. Modelo de Prueba.

Resultados Linux.

- Sistema Operativo Utilizado: Ubuntu 10.04 LTS 32 bits.
- Kernel: Linux 2.6.32-21-generic (i686) .

Consumo de CPU.. El siguiente comando en Linux nos permite observar el consumo de CPU por aplicación.

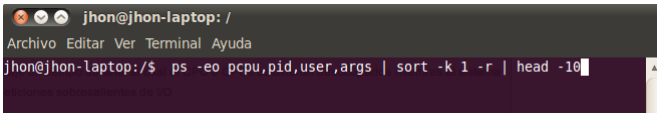

FIGURA 3.2.7. Comando que muestra el consumo de recursos por proceso

El consumo de CPU de la aplicación en porcentaje es el siguiente.
Consumo 2.4 % del CPU.

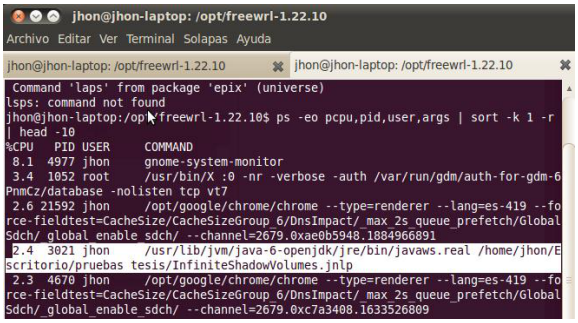

FIGURA 3.2.8. Resultado de pruebas del consumo de CPU del proceso.

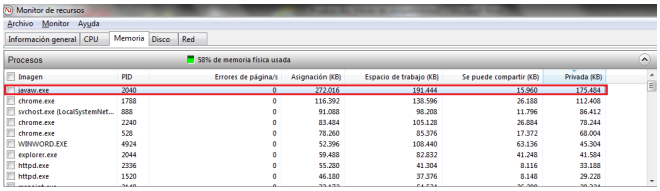

FIGURA 3.2.11. Resultados del consumo de memoria de la aplicación en Windows.

Consumo de Memoria. Este resultado lo obtuvimos del monitor de sistema de Ubuntu.
Consumo de memoria 90.3 MB

FIGURA 3.2.9. Resultados de consumo de memoria.

Resultados Windows.
Recursos.

- Sistema Operativo: Windows 7 Ultimate de 64 bits.

Para el análisis en Windows 7 utilizamos las herramientas que nos provee el sistema operativo en este caso fueron las siguientes:

• Monitor de Recursos -> Para monitorear el consumo de medio de CPU y consumo de memoria RAM de la aplicación.

El resultado del consumo de recursos de la aplicación en Windows fue el siguiente.

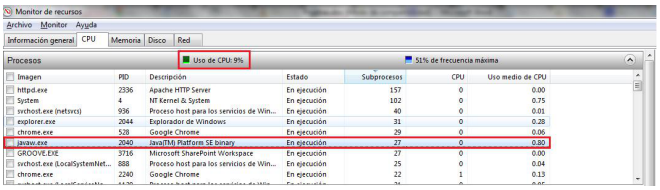

FIGURA 3.2.10. Análisis del consumo de CPU en Windows.

Consumo de CPU 7.2 %
Cálculo. Para calcular el consumo de recursos de un proceso en particular se debe multiplicar el total de consumo que en este caso es 9 % por el uso medio de CPU de la aplicación.

Consumo de Memoria. Este análisis lo realizamos con la misma herramienta en donde los resultados fueron los siguientes.

Consumo de memoria 175.484 KB = 171.83 MB

Los resultados demuestran total superioridad en rendimiento de la aplicación corriendo sobre una plataforma Linux, el consumo de recursos son notablemente mínimos en esta plataforma, estos resultados nos ayudar a seleccionar la mejor plataforma de desarrollo para la aplicación final.

Pruebas OpenGL aplicaciones web con el complemento WebGL.. Las pruebas realizadas en esta ocasión serán con el afán de seleccionar el mejor intérprete del lenguaje WebGL, es así que las pruebas se realizará sobre los navegadores que tengan soporte de interpretación de este lenguaje, ya que actualmente no todos los navegadores tienen esta capacidad, es así que los navegadores seleccionados para esta prueba son los siguientes:

- Google Chrome 11.0.
- Safari 5.0.
- Mozilla Firefox 4.0.

Todos los navegadores están instalados en sus últimas versiones estables disponibles. El modelo a ser probado contiene las características necesarias para llegar a ser un modelo de test aceptable con características como: texturas, zoom, rotación, interacción con el usuario, etc. Al igual que en una aplicación de escritorio WebGL necesita de un lenguaje intérprete en esta ocasión el lenguaje de comunicación será JavaScript por lo cual utilizamos una librería que nos brinda dicha conexión entre Web y el navegador.

La librería utilizada es three.js la cual la podemos encontrar en Three.js. El modelo con el cual realizaremos las pruebas es el siguiente.

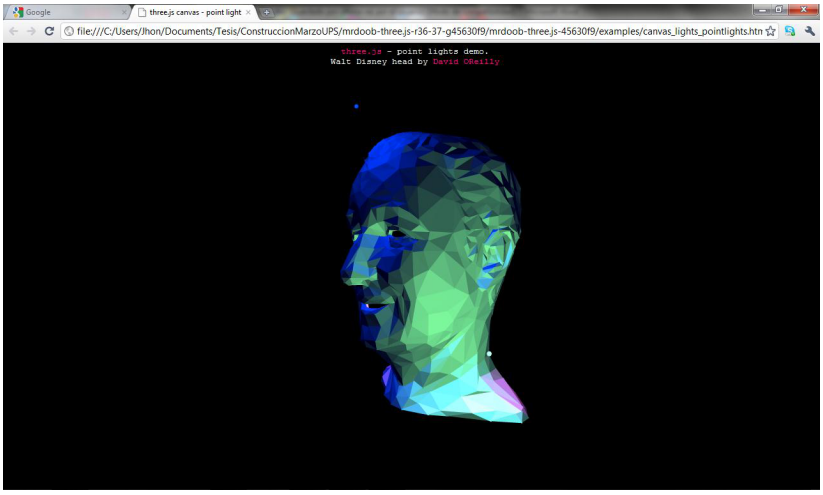

FIGURA 3.2.12. Modelo para pruebas WebGL.

A continuación mostraremos los resultados sobre cada uno de los navegadores propuestos para el análisis de WebGL. Para el análisis lo realizaremos en Windows 7 con los mismos recursos en equipos y herramientas de análisis.

Google Chrome.

Consumo Medio del CPU. El resultado de consumo de recursos fue el siguiente:

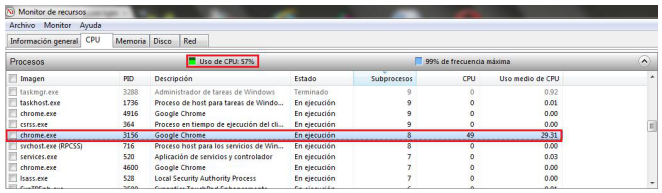

FIGURA 3.2.13. Monitoreo de consumo medio de CPU Google Chrome.

El consumo medio del CPU de la aplicación interpretada en Google Chrome es:

Consumo de CPU 16,7 %.

Al interpretar estos resultados hay que tomar en cuenta que los valores aumentan a un pico al cargar la página con el modelo a interpretar, estos valores luego bajaran hasta establecerse en un mínimo de consumo.

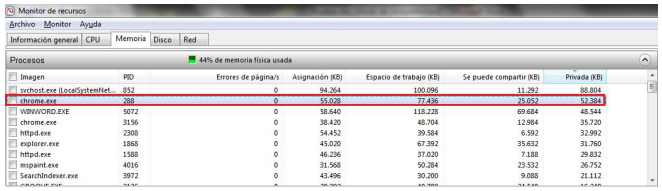

FIGURA 3.2.14. Monitoreo de consumo de memoria Google Chrome.

Consumo de Memoria. Consumo de memoria 53.284 KB = 51,15 MB

Al contrario con del consumo de CPU, el consumo de memoria RAM se mantiene estable.

Safari.

Consumo Medio del CPU. El resultado de consumo de recursos fueron los siguientes.

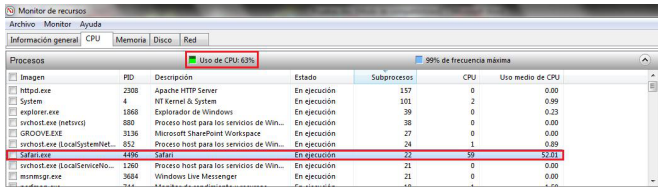

FIGURA 3.2.15. Monitoreo de consumo medio de CPU Safari.

El consumo medio del CPU de la aplicación interpretada en Safari es:

Consumo de CPU -> 32,76 %.

No solo se observa una notable diferencia en consumo de CPU entre Safari y Google Chrome, lo más sobresaliente es que el consumo en Safari no disminuyo en el transcurso de la interpretación puesto que el valor de consumo se mantiene.

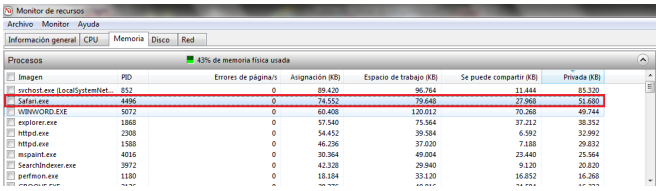

FIGURA 3.2.16. Monitoreo de consumo de memoria Safari.

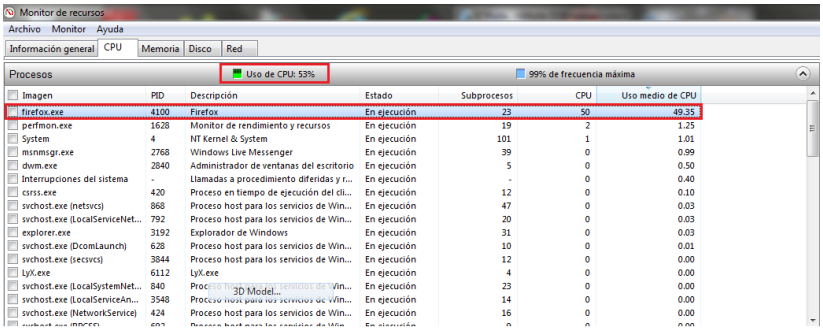

FIGURA 3.2.17. Monitoreo de consumo medio de CPU Mozilla Firefox.

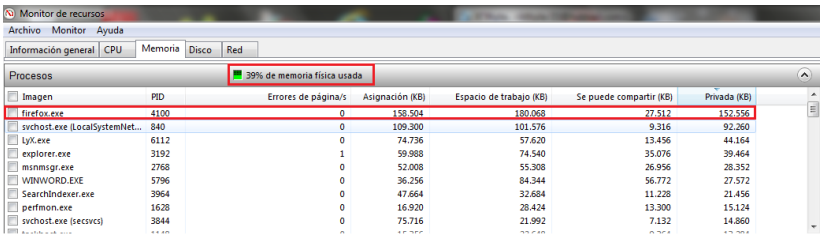

FIGURA 3.2.18. Monitoreo de consumo de memoria Mozilla Firefox.

Consumo de Memoria. Consumo de memoria 51.680 KB = 50,46 MB

Cabe recalcar que el rendimiento de la aplicación en velocidad e interacción con el usuario es mucho menor en Safari.

Mozilla Firefox.

Consumo Medio del CPU. El resultado de consumo de recursos fueron los siguientes.

El consumo medio del CPU de la aplicación interpretada en Firefox es:

Consumo CPU -> 26,155 %

Consumo de Memoria. Consumo de memoria 152.556 KB = 148,98 MB

3.2.6. Conclusiones. Las librerías de código abierto para gráficos no tan solo permiten el diseño e implementación de primitivas o transformaciones convencionales, sino que facilitan al usuario la oportunidad de implementar sistemas gráficos de alta calidad y con gran gestión de procedimientos y funciones. Es así, que muchas aplicaciones como video juegos así como software de desarrollo gráfico basan su implementación en librerías de código abierto como OpenGL.

Sin lugar a dudas, OpenGL es una de las librerías más usadas por los desarrolladores, dado que en su mayoría, los paquetes comerciales requieren licencia, pago, y no se distribuyen con el código abierto. OpenGL por el contrario, están dentro de ese grupo donde el código y la implementación es gratuita, y no requiere licencia (excepto algunos métodos patentados), tiene soporte on-line, documentación, y listas de modificaciones y actualizaciones.

La potencia de la creación de gráficos que provee OpenGL, es impresionante quedando demostrada en las áreas, no solo donde se ve innata la composición de gráficos, como lo serían la animación o el cine, sino en la medicina, meteorología, astronomía, física. Todas estas áreas requieren siempre de simulaciones y gráficos en 3D que permitan orientar a los profesional del cómo sucederán las cosas en casos hipotéticos, siendo un requisito indispensable mostrarlos de manera casi real en un computador.

3.2.7. API's de integración con otros lenguajes. OpenGL es compatible con todos los sistemas operativos más importantes, trabaja con los principales lenguajes de alto nivel, además de ser lo suficientemente flexible en cuanto a su integración con los mismos. Ofrece total independencia de los protocolos de red y sus topologías. Todas las aplicaciones OpenGL utilizada sobre un hardware compatible son capaces de producir resultados consistentes independientemente del sistema operativo y el lenguaje de programación.[**10**]

Sobre cada uno de los lenguajes de programación más robustos se han creado una o más librerías de integración con OpenGL.

C#.

Tao-Framework.: Tao Framework de. NET es una colección de enlaces para facilitar el desarrollo multi-plataforma de juegos relacionados con la utilización de la plataforma. NET basados en la utilidad del API OpenGL.

OpenTK.: Open Toolkit es una avanzada librería de bajo nivel que envuelve la biblioteca OpenGL, OpenAL y OpenCL. Es muy útil para juegos de video, aplicaciones científicas y cualquier otro proyecto que requiera gráficos 3D, audio o funcionalidad de hardware basada en imágenes

Java.

JOGL(JavaOpenGL).: El proyecto JOGL acoge la versión de desarrollo de Java que vincula el API de OpenGL con este lenguaje de programación, y esta diseñado para proporcionar la compatibilidad entre el hardware de desempeño 3D con las aplicaciones escritas en Java. JOGL, proporciona acceso completo a la API OpenGL en todas sus especificaciones así como casi todas las extensiones de la misma.

Java3D.: Es uno de los principales proyectos de java.net, que incluye diferentes librerías como herramientas de creación de elementos 3D basadas en las utilidades que proporciona el API OpenGL.

JGL.: Es una biblioteca de gráficos 3D para el lenguaje de programación Java con un API que es similar a la de OpenGL, y se puede ejecutar en la plataforma Java2 y la versión anterior de Java.

Phyton.

PyOpenGL.: Es un cross-plataform de código abierto en Phyton que integra la funcionalidad de OpenGL sobre gráficos 2-D y gráficos 3-D. PyOpenGL se apoya en las extensiones de OpenGL como GL, GLU y librerías GLUT. Esta librería puede ser utilizada sobre Tkinter, wxPython, FxPy yWin32GUI que son librerías de interfaz de usuario de Phyton.

PyGlut.: Hace que GLUT esté disponible sobre el intérprete de Phyton.

Ruby.

Ruby-Opengl.: Ruby-Opengl se compone de módulos de extensión de Ruby que son los enlaces para las bibliotecas de OpenGL, GLU y GLUT. [**10**]

3.3. VRML

VRML nace como un estándar para visitar mundos virtuales a través de internet. Tiene la virtud de ser un lenguaje que como HTML (Hyper Text Markup Language) están en ficheros de texto plano, es así que VRML (Virtual Reality Modeling Language) se convirtió en un lenguaje representativo de funcionalidades 3D basado en una sintaxis de etiquetas de texto.

En un principio VRML carecía de vida, sus aplicaciones no tenían sonido, ni movimiento alguno, solo permitían visitarlas y desplazarse por ellas. Es por ello que se llegaron a complementar estas soluciones para su versión 2.0.

El lenguaje VRML posibilita la descripción de una escena compuesta por objetos 3D a partir de prototipos basados en formas geométricas básicas o de estructuras en las que se especifican los vértices y las aristas de cada polígono tridimensional y el color de su superficie. VRML permite también definir objetos 3D multimedia, a los cuales se puede asociar un enlace de manera que el usuario pueda acceder a una página web, imágenes, vídeos u otro fichero VRML de Internet cada vez que haga clic en el componente gráfico en cuestión [**45**]. Es así que su funcionamiento está representado en el siguiente gráfico.

El consorcio Web3D actualmente propietarios de VRML actualizaron la funcionalidad de la última versión estable de VRML 2.0 creada en 1997 mediante el desarrollo de X3D un lenguaje informático para gráficos vectoriales definido por una norma ISO que puede emplear tanto una sintaxis similar a la de XML como una del tipo VRML. X3D amplía

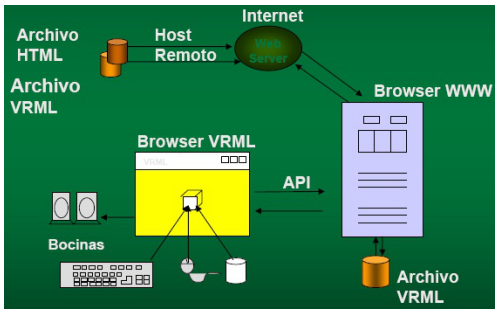

FIGURA 3.3.1. Esquema de funcionamiento VRML.[**26**]

VRML con extensiones de diseño y la posibilidad de emplear XML para modelar escenas completas en tiempo real. [**12**]

3.3.1. Características. VRML es capaz de representar objetos estáticos y animados y puede contener hipervínculos a otros medios de comunicación como el sonido, películas, e imagen. Los intérpretes (navegadores) para VRML están ampliamente disponibles para diferentes plataformas, así como herramientas de desarrollo para la creación de archivos VRML.

VRML se apoya en un modelo de extensibilidad que permite a los nuevos objetos a ser definidos integrarlos a un proceso de registro para que las comunidades desarrollen extensiones compatibles con la norma básica, Es así que posibilita el desarrollo de generadores de aplicaciones y editores, así como para importar datos de otros formatos industriales.

VRML está diseñado para cumplir los siguientes requisitos:

Integridad.: Proporcionar toda la información necesaria para la ejecución y dirección de un conjunto completo de características de gama industria de la aceptación.

Aplicabilidad.: Aplicable en una amplia gama de sistemas.

Multi-usuario.: Permite que las aplicaciones corran sobre una arquitectura de entorno multi-usuario.

Independencia: de estructura. Los elementos de VRML deben ser independientes el uno del otro, o de cualquier dependencia de estructura requiriendo de por si una arquitectura bien definida y ordenada.

Escalabilidad.: Los elementos de VRML están diseñados para llegar a formar parte de composiciones infinitamente grandes.

Estructurado.: Un elemento debe tener una interfaz bien definida y un propósito simple e incondicional.

Compatibilidad.: Proporcionar la capacidad de uso y combinar la dinámica de objetos 3D dentro de un mundo VRML y así permitir la reutilización. (Enfoque orientado a objetos).

Extensibilidad.: Ofrecer la posibilidad de añadir nuevos tipos de objetos que no están explícitamente definidos en VRML, por ejemplo, Sonido.

[**22**]**:**

3.3.2. Requerimientos.

Hardware. Cualquier equipo con 16-bits de resolución gráfica, pero en realidad mientras mejores características en hardware gráfico mejores serán los resultados obtenidos tanto en velocidad como rendimiento. Todos los sistemas operativos compatibles pueden ejecutar VRML. Para obtener los mejores resultados se recomienda una tarjeta gráfica con aceleración 3D.

Software.

- Es independiente de Plataforma.
- Extensible.
- Capacidad de trabajar bajo conexión con un mínimo de ancho de banda.

Los requerimientos tanto de hardware como de software para una aplicación realizada en VRML son relativamente accesibles pero cabe recalcar que para mejorar el rendimiento de nuestros modelos 3D será necesario una tarjeta gráfica con características en desempeño 3D. [**22**]

3.3.3. Desempeño en aplicaciones de escritorio. Las librerías de VRLM no están construidas para realizar aplicación de escritorio, pero su arquitectura permite el desarrollo de aplicaciones que sirvan de intérpretes del lenguaje. Aplicaciones que permiten la interpretación del lenguaje VRML podemos encontrar varias entre las cuales las siguientes:[**51**]

Interprete	**Windows**	**Linux**	**Mac**
BitManagement's BS Contact	SI	SI	SI
Octaga	SI	SI	SI
InstantReality	SI	SI	SI
FreeWRL/FreeX3D	SI	SI	SI
SwirlX3D Free Player	SI	NO	NO
GeoVrml Run-Time	SI	NO	NO

CUADRO 1. Aplicaciones de escritorio como intérpretes de VRML. [**51**]

Nuestro análisis se centrara en las herramientas mas reconocidas por el grupo de desarrolladores web3D los cuales son pioneros en el desarrollo de aplicaciones 3D para internet, las herramientas seleccionas para el proceso son las siguientes:

BitManagement's BS Contact. BS Contact puede ser adoptado por la funcionalidad e interfaz gráfica de usuario, el menú o el tamaño para adaptarse mejor a las necesidades específicas del cliente.

Sus principales características son:

- Espectador estándar solo e integrado en los navegadores web, basados en OpenGL o DirectX.
- Multiplataforma con soporte para Windows, Linux, Mac y Moviles (BS Movil).
- Adecuado para interpretar los formatos vectoriales principales como VRML, X3D, Collada, KMZ.
- Una amplia funcionalidad adecuada para todos los canales de distribución, por ejemplo Web, CD / DVD o PC.

[**5**]

FIGURA 3.3.2. BitManagement's BS Contact.

FreeWRL/FreeX3D.. FreeWRL es una herramienta esencial como intérprete de desarrollo para aplicaciones web3D con tecnologías como VRML y X3D ya que a más de ser una herramienta Open Source supera a otros plugins situándose entre los mejores intérpretes de escritorio del lenguaje. Su poder proviene en la extensibilidad y la facilidad de desarrollo que brinda al interpretar código vectorial de manera estable.[**47**]

Posee un componente geoespacial que nos permite ubicar nuestra modelo o aplicación en cualquier lugar que deseemos.

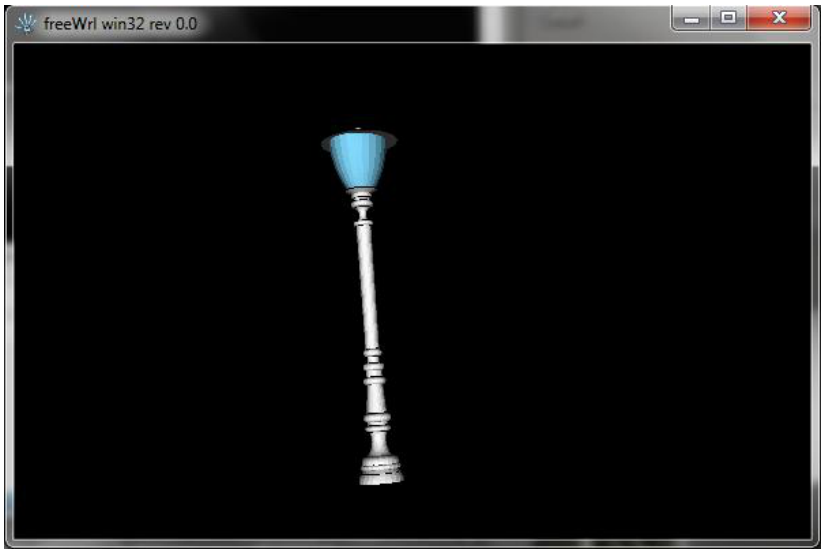

FIGURA 3.3.3. FreeWRL/FreeX3D

A las herramientas seleccionadas en el siguiente punto se les realizara un análisis de consumo de recursos con el afán de seleccionar el mejor intérprete del lenguaje.

3.3.4. Desempeño aplicaciones Web. Virtual Reality Modeling Language (VRML), es un formato para describir y transmitir Objetos en 3D y mundos compuesto de la geometría y multimedia en un entorno de red. Los objetivos de VRML en las aplicaciones web están orientadas a la construcción de productos tales como: diseño asistido por computadora, ingeniería y visualización científica, productos multimedia, ofertas de entretenimiento y educativos, y mundos virtuales compartidos. [**13**]

VRML tiene varias características que lo hacen especialmente atractivo para la creación de mundos virtuales. Estas características incluyen:

- Escenas gráficas.
- Procesamiento de eventos.
- Comportamientos.
- Encapsulación y reutilización.
- Contenido Distribuido.
- Extensibilidad.
- Interactividad.
- Animación.

VRML se ha centrado en el concepto innato de la realidad virtual en internet, para lograr representar un modelo VRML en un browser necesitamos de un plugin que nos ayude a lograr dicho objetivo como Cosmo Player, en el mercado existen varios plugin que nos pueden ayudar a conseguir excelentes resultados con VRML. El análisis de estas herramientas se las realizara en una siguiente etapa además de medir el desempeño de las mismas.

Un buen estilo VRML ligado al performance de aplicaciones web. Hasta ahora hemos aprendido que VRML es un lenguaje de marcas script que nos permite describir formas en 3D además de ambientes interactivos en la WEB. Con las características presentadas hasta el momento se demuestra que se pueden crear paginas en VRML muy interesantes pero se necesitas describir ciertas características que maximizan su potencial como herramienta de desarrollo para ambientes 3D.[**30**]

Información de navegación. VRML permite especificar la información de navegación mediante el nodo NavigationInfo1. En la actualidad la mayoría de los navegadores VRML 2.0 permiten integrar los métodos básicos de navegación permitiendo elegir entre WALK, EXAMEN, FLY o NONE, el siguiente código nos permite especificar el modo de navegación.

NavigationInfo { type "EXAMINE" } #Usa el visor Examine.

Puntos de Vista. Se puede obtener la orientación en una dirección específica permitiendo la rotación y facilidad de navegación para los usuarios.

Información del Mundo Virtual. Contiene la funcionalidad ingresar información dentro los modelos generados, permitiendo a los navegadores mostrar el contenido del campo título interactuando directamente con la información de los objetos 3D como componentes parte de un lenguaje interpretado por el navegador.

Performance. El rendimiento de una aplicación sobre VRML depende de muchos factores manteniendo una estructura adecuada que ayude a mejorar su eficiencia entre las técnicas más importantes son las siguientes:

Agrupación.: Agrupación de objetos en la construcción de la escena.

Nombramiento.: Reutilización de objetos dentro de la escena proporcionándoles un nombre que los identifique.

Eficiencia.: Creación de nodos y estructura adecuada del cuerpo del documento que contiene la codificación del escenario 3D.

[**23, 30**]**:**

3.3.5. Pruebas. En esta sección realizaremos un test utilizando diferentes visores tanto para el Sistema Operativo Windows como para GNU/Linux, las pruebas están enfocadas a determinar el rendimiento de las aplicaciones 3D creadas con VRML como lenguaje base, tratándonos de ayudar a seleccionar el lenguaje para llevar a cabo aplicaciones 3D. En este caso las pruebas realizadas sobre VRML están destinadas a verificar el consumo de recursos de una aplicación real de nuestro proyecto.

Recursos: Las pruebas serán realizadas con los mismos recursos para tratar de tener una mejor interpretación de los resultados.

Procesador: 2x Intel(R) Core(TM)2 Duo CPU T5750 @ 2.00GHz.

Memoria: 3087MB.

Resolución: 1280x800 pixeles.

X11 Vendor: The X.Org Foundation.

VGA Display Controller: Intel Corporation Mobile GM965/GL960 Integrated Graphics Controller.

Aplicaciones en prueba. La aplicación o ejemplo sobre el cual vamos a trabajar en nuestro test, son ejemplos reales de nuestro objetivo final, en esta ocasión utilizaremos como escena la representación de la entrada principal de la Universidad Politécnica Salesiana.

FIGURA 3.3.4. Representación de la entrada de la UPS con VRML.

En nuestro test utilizaremos varios visores, hemos tratado de utilizar los mismos visores tanto para la plataforma Windows como GNU/Linux.

Windows	Linux
BS Contact.	BS Contact.
Freewrl.	InstantReality.

CUADRO 2. Visores VRML para SO Windows y Linux.

Las pruebas nos ayudaran a determinar el mejor lenguaje para desarrollo 3D, tratando de medir el desempeño y consumo de memoria de una aplicación base.

Resultados Linux. Sistema Operativo Utilizado: Ubuntu 10.04 LTS 32 bits.

Kernel: Linux 2.6.32-21-generic (i686).

Pruebas realizadas con el visor BS Contact. **CONSUMO DE CPU.**

Como ya es de nuestro conocimiento las instrucciones utilizadas para visualizar el consumo de CPU de la aplicación es el siguiente:

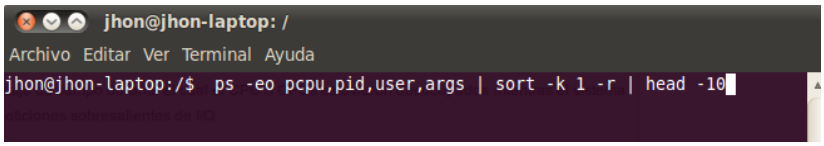

FIGURA 3.3.5. Instrucción para monitorear procesos del usuario.

El consumo de CPU de la aplicación en porcentaje es el siguiente.
Consumo del CPU 82.3 %.

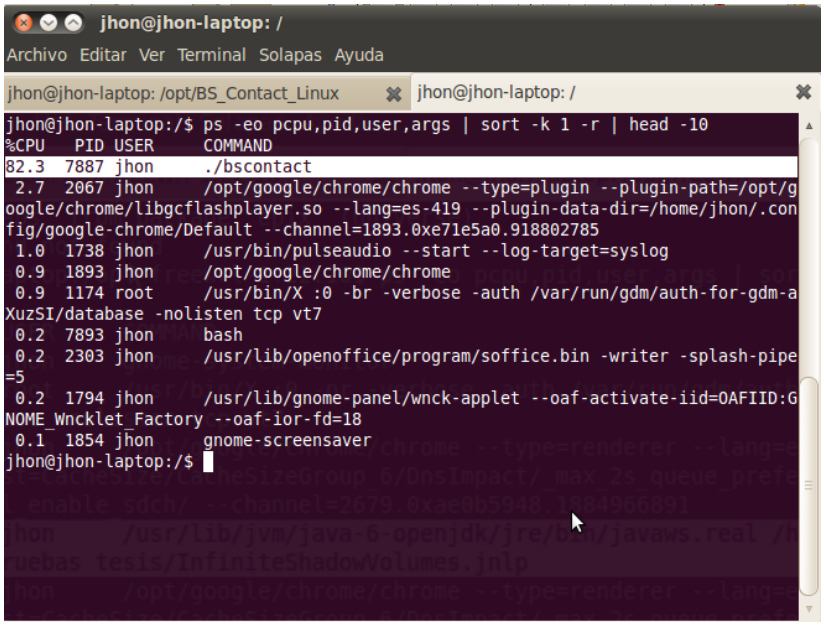

FIGURA 3.3.6. Resultados de consumo de CPU del visor BSContact en Ubuntu.

CONSUMO DE MEMORIA.
Este resultado lo obtuvimos del monitor de sistema de Ubuntu.
Consumo de memoria 48,2 MB

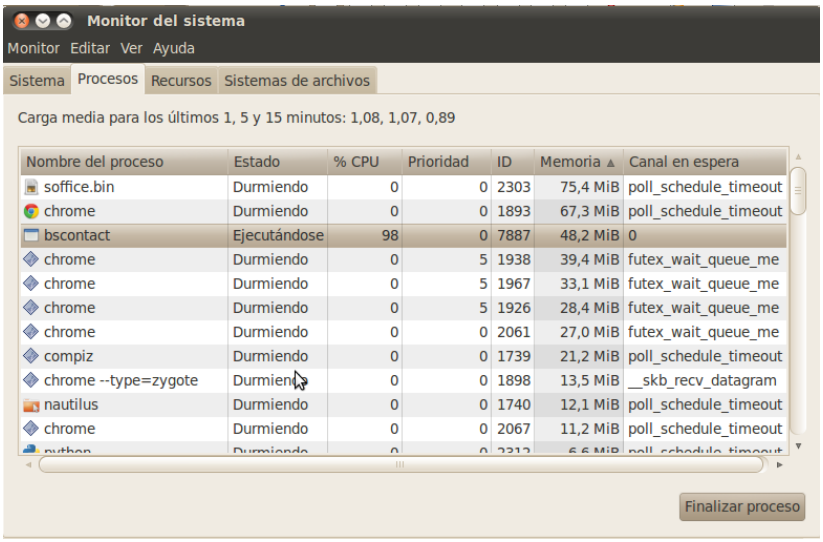

FIGURA 3.3.7. Resultados de consumo de memoria con el visor BS-Contact en Ubuntu.

Pruebas realizadas con instantReality.

CONSUMO DE CPU.. El consumo de CPU de la aplicación en porcentaje es el siguiente.

Consumo del CPU 53.6 %.

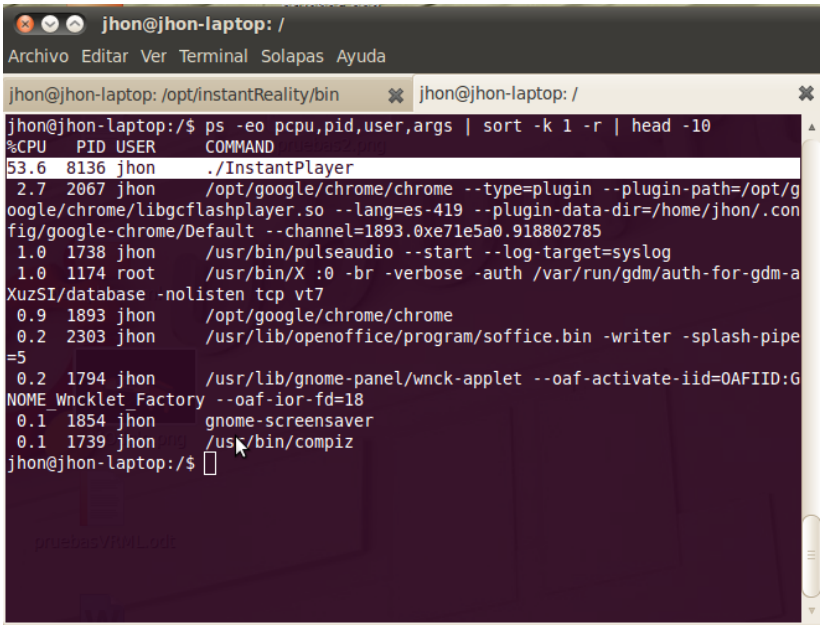

FIGURA 3.3.8. Resultados de consumo de CPU del visor instantReality en Ubuntu.

CONSUMO DE MEMORIA.

Este resultado lo obtuvimos del monitor de sistema de Ubuntu.

Consumo de memoria 232.5 MB

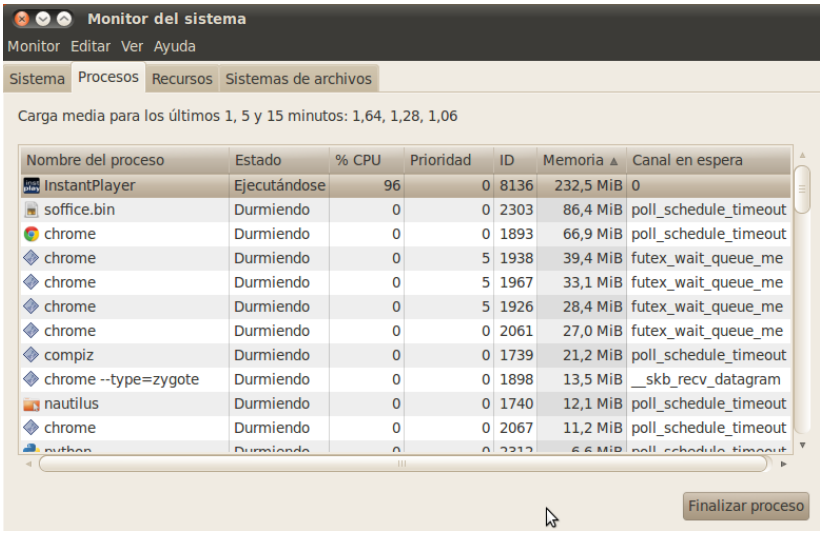

FIGURA 3.3.9. Resultados de consumo de memoria con el visor BS-Contact en Ubuntu.

Resultados Windows. Sistema Operativo: Windows 7 Ultimate de 64 bits.

Para el análisis en Windows 7 utilizamos las herramientas que nos provee el sistema operativo en este caso fueron las siguientes:

Monitor de Recursos -> Para monitorear el consumo de medio de CPU y consumo de memoria RAM de la aplicación.

Pruebas realizadas con FreeWRL. El resultado del consumo de recursos de la aplicación en Windows fue el siguiente.

CONSUMO DE CPU.

Consumo de memoria 38,755 %

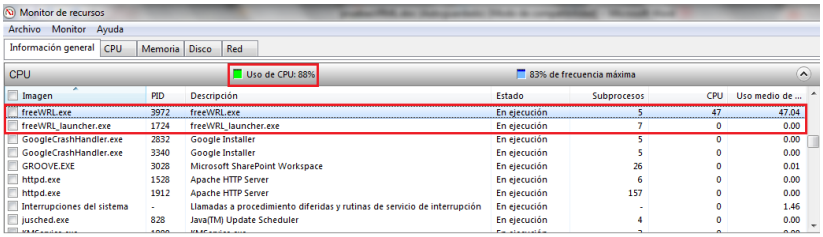

FIGURA 3.3.10. Resultados de consumo de CPU del visor FreeWRL en Windows 7

Calculo.- Para calcular el consumo de recursos de un proceso en particular se debe multiplicar el total de consumo que en este caso es 88 % por el uso medio de CPU de la aplicación.

CONSUMO DE MEMORIA.

Este análisis lo realizamos con la misma herramienta en donde los resultados fueron los siguientes.

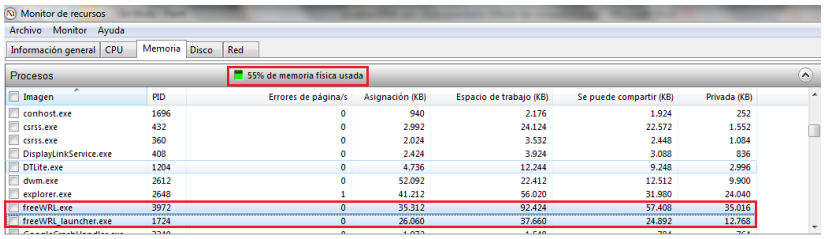

FIGURA 3.3.11. Resultados de consumo de Memoria del visor FreeWRL en Windows 7

Consumo de memoria 61.362 KB = 59.933 MB

Para obtener los resultados de consumo de memoria se debe sumar el consumo de memoria por cada proceso relacionado a la aplicación en este caso los procesos que se ejecutan son:

PROCESO	CONSUMO DE MEMORIA
freeWRL.exe	35.312 KB
freeWRL_launcher.exe	26.060 KB
TOTAL	61.362 KB

CUADRO 3. Procesos relacionados a FreeWRL en Windows 7

Pruebas realizadas con el visor BS Contact. El resultado del consumo de recursos de la aplicación en Windows fue el siguiente.

CONSUMO DE CPU.

Consumo del CPU 20,205 %

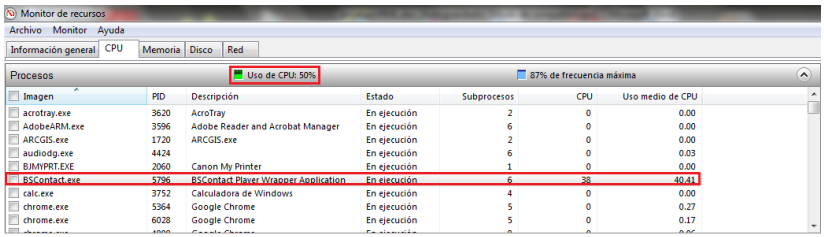

FIGURA 3.3.12. Resultados de consumo de CPU del visor BSContact en Windows 7

Calculo.- Para calcular el consumo de recursos de un proceso en particular se debe multiplicar el total de consumo que en este caso es 50 % por el uso medio de CPU de la aplicación.

CONSUMO DE MEMORIA

Este análisis lo realizamos con la misma herramienta en donde los resultados fueron los siguientes.

Consumo de memoria 337.516 KB = 329,605 MB

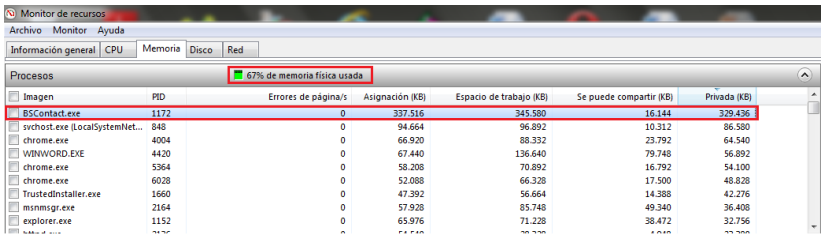

FIGURA 3.3.13. Resultados de consumo de Memoria del visor BSContact en Windows 7

3.3.6. Conclusiones. VRML es un lenguaje script que consume los recursos progresivamente a partir de su interpretación, los visores de escritorio nos ayudaron a representar dicho consumo de recursos, la velocidad o calidad de representación de modelos 3D en el Web son procesos que se forman en el cliente. VRML como lenguaje para desarrollo de aplicaciones 3D es rápido y sencillo con muchas opciones dedicadas a minimizar el tiempo de desarrollo, puesto que es uno de los lenguajes bases en el mundo de la virtualización. Al mismo tiempo la representación del lenguaje es costosa en recursos de máquina, las aplicaciones o visores que ayudaron en las pruebas del lenguaje nos demostraron un consumo muy alto de recursos a pesar de representar un escenario de características simples, las ventajas del lenguaje se basan en minimizar el tiempo de desarrollo es así que luego de la investigación y experimentación demostró deficiencia en el consumo adecuado de recursos de máquina frente a otros lenguajes con el mismo objeto.

3.3.7. API's de integración con otros lenguajes. VRML es un un lenguaje scripting destinado a la interpretación e integración con lenguajes y tecnologías de internet, esta es una particularidad del lenguaje que representa una gran ventaja de desarrollo y adaptabilidad hacia ambientes web.

El lenguaje como tal es extensible y adaptable a diferentes tecnologías que actualmente se usan para el desarrollo de aplicaciones WEB, sus componentes llegan a formar parte del DOM (Document Object Model) siendo completamente adaptable a diferentes componentes de una aplicación web como:

- Arquitectura del sistema de desarrollo.
- HTML y Perfil de Desarrollo.
- Actualizaciones del DOM.
- Eventos HTML.
- Integración CSS.
- Integración con JavaScript.
- Actualmente X3D con WebGL.

[**13**]

A continuación veremos un ejemplo de la integración del lenguaje dentro de una pagina HTML simple.

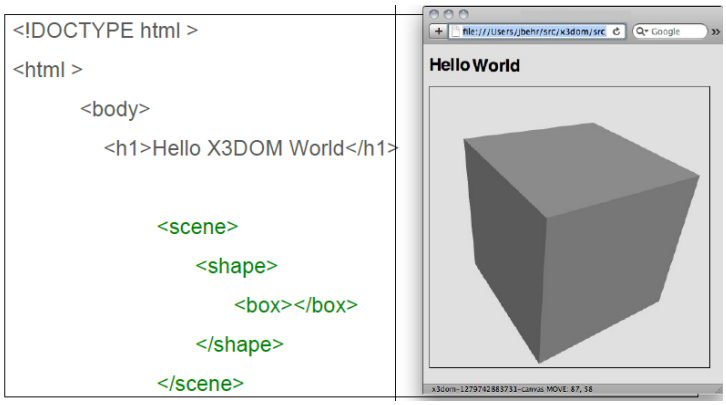

FIGURA 3.3.14. Integración de código VRML dentro de una aplicación HTML[**13**]

La base crucial para la integración con otros lenguajes es la particularidad de fusionar características con JavaScript logrando un control total del DOM en la página, con la unión de estas dos tecnologías, a continuación veremos un pequeño ejemplo de cómo realizar dicha integración de forma muy sencilla.

```
<Scene>
 <Shape>
  <Appearance>
   <Material id='mat' diffuseColor='1 0 0' />
      </Appearance>
      <Box onclick="document.getElementById('mat').diffuseColor='0 1
 0'" />
 </Shape>
</Scene>
```

FIGURA 3.3.15. Código de integración de VRML con eventos JavaScript.[**13**]

La facilidad de integración con el lenguaje JavaScript fácilmente nos representa una integración total con los diferentes de servidores que actualmente se utilizan como:

- Java.
- PHP.
- Ruby.
- Phyton.
- .NET
- Etc.

En realidad esta extensibilidad del lenguaje nos permite concluir esta sección afirmando que es totalmente integrable con cualquier tecnología de servidor.

3.4. Conclusiones y síntesis de la información obtenida.

Los lenguajes de desarrollo de escenarios virtuales son basados construcciones de objetos a partir de términos matemáticos, los algoritmos para desarrollar estas soluciones son la base para minimizar el consumo de recursos.

OpenGL es una librería que se caracteriza por explotar su funcionalidad minimizando el consumo de recursos de máquina y sacarle el mejor provecho a los mismos, es un lenguaje complicado de adaptarse para el desarrollo representando un consumo de tiempo alto en el aprendizaje para los desarrolladores, una de las ventajas mas altas que actualmente nos provee esta librería es la facilidad de integración con aplicaciones web con su nuevo API WEBGL, permitiéndonos una integración completa entre aplicaciones de escritorio y aplicaciones WEB.

El termino WEB3D hace referencia a cualquier lenguaje de programación, protocolo, formato de archivo o tecnología que pueda ser usada para la creación, representación y modelado 3D de universos y escenarios tridimensionales interactivos a través de Internet. Virtual Reality Modelling Lenguaje (VRML) se convirtió en el primer estándar (IETF) para especificar escenas 3D en la WWW es decir versión 3D de HTM con formatos para desarrollo de geometrías 3D conectadas y procesadas y representadas vía WEB. VRML actualmente constituye la base para varios lenguajes y tecnologías con una visión ambiciosa y futurista para la virtualización 3D en internet, tales como X3D que es un lenguaje amplio y fácil de integrar en nuestras aplicaciones, de la misma manera al ser basado en VRML los plugins que existen para los diferentes browser que existen actualmente se pueden procesar los modelos realizados en dicha tecnología. La debilidad de este lenguaje, es que no tiene la posibilidad de desarrollar aplicaciones de escritorio puesto que es un lenguaje nato para aplicaciones WEB.

3.5. Selección del lenguaje más adecuado en construcción de ambientes virtuales.

En realidad la elección del mejor lenguaje o API de desarrollo para aplicaciones 3D es una tarea muy difícil puesto que depende mucho del tipo de aplicación que se quiere llegar a desarrollar, su enfoque de negocio y facilidad para el usuario que se quiere brindar.

En este trabajo se trata de desarrollar una representación de escenario 3D de la Universidad Politécnica Salesiana, y esto representa una posible interpretación de nuestro trabajo para el desarrollo de una aplicación para la construcción de elementos 3D para dichos escenarios. Esta exigencia demuestra la necesidad del desarrollo de una aplicación híbrida con comunicación Web y/o de Escrito, este requerimiento nos ha llevado a la conclusión

de elegir OpenGL como el lenguaje base para el desarrollo de aplicaciones 3D. Cabe recalcar como ya ha sido mencionado con anterioridad la elección de un lenguaje para el desarrollo 3D necesariamente es primordial la finalidad de su construcción.

Capítulo 4

Herramientas de diseño y modelado 3d

Modelar no es simplemente diversión es un medio con un objetivo y llegar a tener la capacidad para aprovechar el modelado en 3D, se dice que es una representación de coordenadas conformadas por estructuras a las que se les puede agregar texturas, imaginemos una casa modelada en 3D con sus respectivos colores y materiales que le caracterizan como la teja, el vidrio, el metal, etc. [**29**]

FIGURA 4.0.1. Modelo 3D realizado con herramientas de diseño GPL. [**29**]

Los objetos modelados tienen que ser evaluados no solo en sus texturas o colores sino en toda su ergonomía, todos y cada uno de los productos tienen ergonomía ya sea perfecta o imperfecta según los detalles que tenga.

Este tipo de modelado tiene muchas ventajas y puede ser aplicado en diferentes áreas en la vida profesional para obtener resultados con detalles de mayor precisión y visualización real.

Aplicado en la Medicina. El modelado en 3D juega un papel importante en el área de la medicina para los diagnósticos de cada paciente y así poder tener una mayor y mejor visualización de los resultados de la estructura de la anatomía obteniendo imágenes o animaciones de los órganos del cuerpo humano.

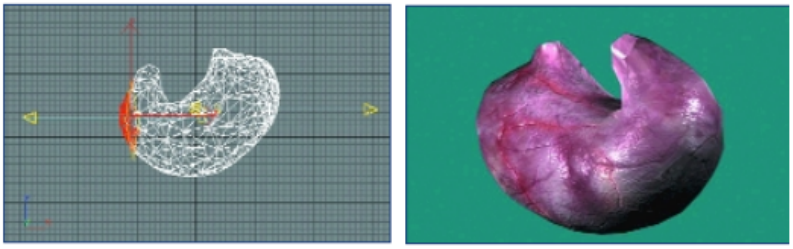

FIGURA 4.0.2. Implementación de diseño 3D en el área de la medicina. [**36**]

Aplicado en Ingenierías y Arquitectura. Las aplicaciones de ingeniera son extensas ya que se pueden modelar gran número de objetos ya sean mecánicos, industriales etc, teniendo como resultado el diseño técnico del modelo y ofreciendo una simulación del funcionamiento del objeto creado antes de que se lance al mercado. Por ejemplo los arquitectos tienen la posibilidad de incrementar su trabajo diseñando sus planos en 3D y generando la documentación técnica respectiva para presentar al cliente las pre visualizaciones de sus proyectos en imágenes foto-realistas y hacer que los usuarios tengan una idea más real del proyecto que se implementará.

Aplicado en Cinematografía. Las imágenes que podemos ver en 3D se logran mediante efectos y combinación de vectores que realizan el nuevo software de desarrollo en 3D, acoplando puntos de emisión de luz y aplicándoles texturas. Actualmente ya hemos vivido esta experiencia gracias a las producciones creadas y con excelentes animaciones como hemos podido ver, avatar, dinosaurios de Jurasic Park, guerra de galaxias, etc.

Aplicado en Publicaciones. El uso de las herramientas de creación 3D brinda la gran posibilidad de crecimiento en el campo de la comercialización, teniendo donde plasmar la creatividad de cada diseñador y dando al usuario una nueva visión revolucionaria permitiendo ver imágenes en 3D sin la necesidad de gafas.

Existen aplicaciones que nos facilitan la creación y modificación de los objetos en 3D, cada una de ellas con diferentes ventajas, facilidades y funcionalidades.

3D Studio Max. Es una aplicación orientada o objetos para el diseño en 3 dimensiones desarrollada por Kinetex, puede tomar objetos a partir de vistas planas o alzadas como punto de inicio, permite la exportación de los modelos a otros formatos distintos del que maneja 3D Studio Max, basado fundamentalmente en el sistema operativo Windows 9x/NT. Este producto permite integrar las escenas creadas en 3D con los personajes u objetos reales, tiene su funcionamiento sobre sistemas que sean multiprocesador y tenga conexión de dispositivos como mouse, teclados, etc., siendo así una de las aplicaciones más utilizadas en el desarrollo de los videojuegos se han creado títulos como Tom Raider, Halo, etc. [**34**]

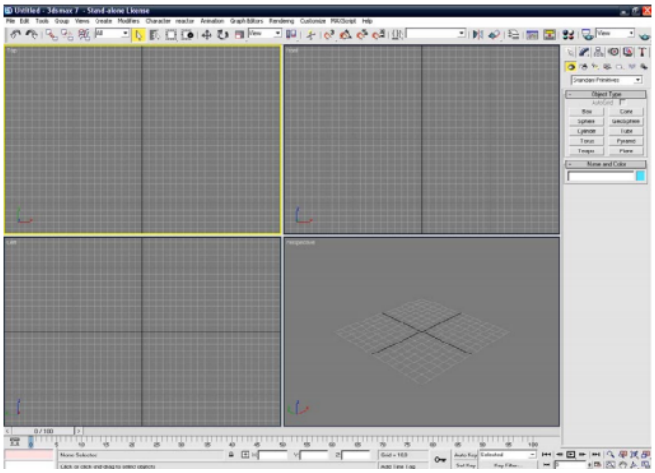

FIGURA 4.0.3. Interfaz de desarrollo 3D studio max. [**34**]

3D Studio Max posee un gran número de Plugins además de ser potente y estable, peor lo que no le hace tan acotable es su alto precio por la aplicación, los requisitos de hardware son muy elevados y no están disponibles para todos los usuarios.
Requerimientos Mínimos.

- Windows NT o superior.
- Procesador mínimo de 150 MHZ compatible con Intel.
- 48Mb de RAM y 200 Mb de espacio libre en disco para el swap.
- Tarjeta de video que soporte para 800 y 600 pixeles y 265 colores.

AutoCAD.. Es un software de diseño asistido por computador desarrollado principalmente para dibujo técnico en 2D pero también para crear modelos en 3D y dando la posibilidad de visualizar las imágenes creadas en diferentes formatos, utilizado en la arquitectura, diseño de interiores e ingenierías. Maneja las imágenes de tipo vectorial también permite la incorporación archivos de tipo fotográfico y posee una organización más viable para los usuarios la cual se da en capas para que el dibujo este en partes independientes y con sus respectivos atributos. [**24**]

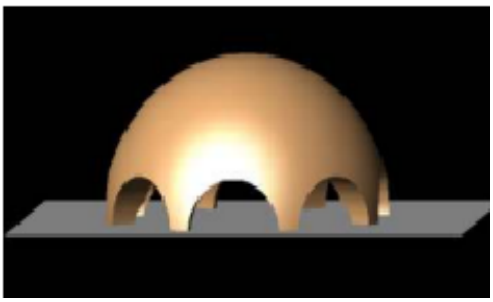

FIGURA 4.0.4. Modelo de una cupula desarrollado en AutoCAD.[**43**]

Requerimientos mínimos.

- Windows XP Vista, 7, 2003
- Memoria 2 GB de RAM.
- Disco duro 32bits,
- 1 Gb para la instalación.
- 1,5 Gb minimo de memoria RAM.

K-3D. Es uno de los programas que crea y modela objetos en 3 dimensiones siendo un software libre y contando con la facilidad de aumentar su procesamiento mediante Plugins para que se haga más versátil, se caracteriza por la creación de polígonos principalmente y posee una interfaz fácil e intuitiva. Al momento de la creación y edición de documentos esta herramienta nos permite hacerlo en tiempo real ya sea mapeado o aplicando texturas.[**14**]

FIGURA 4.0.5. Interfaz de desarrollo K-3D. [**31**]

Requerimientos mínimos.

- Windows XP, 2000, Vista, 7, GNU/Linux, MAC
- Procesador mínimo Pentium III 500 MHz
- Memoria 128 ram
- Disco 200 Mb

TopMod 3D.. Herramienta que se encarga de modelar diseños 3d pero de manera diferente, crea objetos diversos como mallas y objetos sólidos sencillos utilizando plataformas topológicas y teniendo licencia open source con una interfaz muy sencilla y fácil de interactuar disponible para Mac y Windows.[**17**]

Sweet Home 3D.. Es una herramienta desarrollada para las personas que por alguna razón quieren remodelar sus casas interiormente o cambiar sus muebles de un lugar a otro. Sweet Home 3D da la facilidad de tener una visión virtual de la disposición de los muebles aprovechando mejor los espacios con solo dar un clic en la PC's, posee una gran colección

de muebles y materiales para la decoración de la sala, cocina, habitaciones, etc colocadas sobre un plano de 2D.[**11**]

4.1. Herramientas GPL de diseño de ambientes 3D.

BLENDER.. Desarrollado por Ton Roosendaal es una herramienta que fue diseñada para modelar objetos en 3 dimensiones, es multiplataforma, gratuito y bajo la licencia GNU GPL, da la posibilidad de crear, poner texturas , animar, renderizar y editar video en forma no lineal para tener la posibilidad de crear videojuegos en 3D. Blender también ofrece herramientas para simulación de cuerpos dinámicos, este se limita solo en algunas áreas como la ausencia de extra modelado o materiales con textura.

Blender a pesar de que es una herramienta relativamente nueva se ha utilizado en grandes proyectos cinematográficos como la creación de SpiderMan II, tiene gran documentación generadas y aportadas por las comunidades que ayudan a mejorar esta aplicación día a día, ampliando todas las posibilidades ya que utiliza guiones de Phyton. [**35**] [**29**]
Plataformas en las que se puede instalar blender.

- Windows desde: 98 – ME – 2000 – XP – Vista – 7
- Linux: Todas sus distros.
- MacOS: 10.3 en adelande.
- Irix: 2.8 • Solaris: 2.8
- FreeBSD: 6.2

INSIGHT3D. Es una herramienta de diseño 3d que permite crear modelos a partir de fotografías, tomando una serie de fotografías de escenas reales y calculando las posiciones del espacio de cada foto y haciendo que coincidan automáticamente los puntos con los de la escena en 3D .Es una herramienta gratuita bajo la licencia GNU GPL, se dedica a crear modelos poligonales y existe la posibilidad de darles textura a cada uno de los mismos.[**9**]
Plataformas en las que se pueden instalar.

- Windows
- Linux

Google Sketchup. Es una herramientas creada por Google para poder modelar los edificios y monumentos de Google Earth. Al diseñar un objeto en esta herramienta nos ofrece una manera muy similar a como los diseñadores o arquitectos lo hacen a mano, tomando como muestra objetos reales, su interfaz es muy intuitiva ya que es de fácil manejo y mucha familiaridad.

Sketchup tiene una manera diferente al iniciar la creación de objetos, esta se da mediante formas planas o en 2 dimensiones, luego de la manera más sencilla obtenemos la tercera dimensión con solo aplicar la función de empujar / tirar que viene integrado en el mismo programa. Al finalizar con el proyecto podemos cargarlo y situarlo en Google

Earth con sus respectivas coordenadas en caso de que el proyecto sea de una casa, edificio, etc. [**52**]
Requerimientos mínimos para Google Sketchup.
Windows XP, Vista , 7, Linux.

- Procesador: Pentium 4 o AMDk7 , 1 Gb.
- Memoria RAM: 64 Mb, 512 Mb.
- Disco Duro: 300 Mb
- Tarjeta de video 3d con 128 de memoria
- Funciones: Diseños 3D. [**4**]

Mac OS.

- Procesador: 2.1 Gb
- Memoria: 1Gb
- Disco Duro: 300 Mb
- Tarjeta de video 3D con 128 de memoria
- Tarjeta de vídeo versión 1.5 de OpenGL.[**4**]

Linux. Google Sketchup no está disponible para la plataforma de Linux por el momento, pero realizando pruebas y utilizando el virtualizador WINE funciona a la perfección.

J3dWorkBench. Es una herramienta encargada de crear e importar objetos 3d en diferentes formatos, permitiendo también simulaciones de escenas, pueden ser entregados por HTTP a una página de servicio Web mediante JavaWeb Start. J3dWorkbench es una aplicación muy simple creada para escritorio que no necesita saber mucho acerca de programación para poder instalarlo, con solo descomprimir el archivo descargado ".rar" y realizar la ejecución del archivo podemos interactuar con el mismo. Tiene licencia GPL y no necesita de instalación, dos ventajas sobre otras herramientas que se dedican a la misma tarea, pero lo que desencanta de J3d Workbeanch es que no tiene muchas funcionalidades y opciones de edición como otras aplicaciones existentes para modelado 3d, para los primeros usuarios que se integran al mundo en 3 dimensiones es suficiente esta aplicación. [**3**]

4.2. API's de integración con lenguajes de alto nivel.

API'S para la herramienta Blender. Blender, la herramienta de modelado 3D permite crear funciones extras mediante el API Phyton, que impresiona por ser un lenguaje de alto nivel multiparadigma y de código abierto. Brinda muchas posibilidades de añadir nuevas tareas y automatizar las mismas para ganar tiempo en el desarrollo de los modelos y ejecutándolos con Blender-P mediante el modulo global Blender.Run. Blenber actualmente utiliza Phyton 3.x que interpreta todas las funciones como la edición de video, texturas y cambio de la interfaz de usuario.

El API Phyton de Blender es muy importante ya que permite dar más opciones a los objetos creados y poderlos simular desde diferentes perspectivas teniendo una mejor idea del proyecto en creación y tomar decisiones con mayor certeza. Existen 3 tipos de scripts de Phyton para Blender:

Ejecución de manual: Crea tareas en el diseño del proyecto ejecutándolas con Alt+P, además pueden tener un entorno gráfico mediante funciones por evento.

Ejecución automática: Si deseamos tener un objeto con localización dependiendo del tiempo podemos hacerlo aplicando instrucciones al objeto, se da la ejecución automática cuando Blender redibuja el entorno en el que estamos trabajando o deseamos cambiar de Frame.

Logicbrick: Se aplica en el motor de la toma de decisiones en tiempo real. [**38**]

API'S para INSIGHT3D.. No existe API que se integre con INSIGHT3D.

API'S para Google Sketchup. Google nos brinda la herramienta Sketchup de modelado 3D y nos da la posibilidad y libertad de mejorarlo de acuerdo a nuestras necesidades, aplicando su API de RUBY el cual es un lenguaje de programación orientado a objetos teniendo una licencia de software libre que permite que Google Sketchup entienda su interfaz, el usuario puede crear sus propios scripts y desarrollar funciones extras como:
Crear herramientas para dibujos personalizados. Cuando estamos diseñando nuestros modelos en Google Sketchup cada vez se hacen más pesado si le agregamos objetos adicionales o si crece el modelo escogido volviéndose un modelo de gran volumen y dando a veces problemas al cargar el modelo completo, pero si esta es la situación podemos resolverlo creando herramientas avanzadas que aumenten la velocidad mediante el API de Ruby que nos ofrecen para programar y poder resolver problemas personalizados. Para crear una nueva herramienta en Ruby lo primero que se debe hacer es definir una clase nueva para que dé respuesta a ciertos eventos que se deseen. [**44**]

Ejemplo:
class MyTool
def active
puts " Your tool has been activated"
end
end
my_tool = MyTool.new Sketchup.active_model.select_tool my_tool

CUADRO 1. Ejemplo integración del API de Ruby en Google Sketchup.

Tareas automatizadas. Los modelos que se crean en Sketcup pueden de forma manual, también se los puede crear de forma automática mediante líneas de código integradas en Ruby para facilitar las necesidades del lugar donde trabajas o tus propias expectativas, de

esta manera y teniendo esta ventaja a cerca de esta herramienta se ahorra mucho tiempo y dinero. [**44**]
Generar Informes de resultados. Sketchup da la posibilidad de poner texto en los modelos creados en 3D, y no solo tener geometría, el API Ruby permite que obtengamos un modelo con su respectiva información con solo unas líneas de comando.[**44**]
Crear complementos de negocio. Todos los empresarios tienen que comercializar sus productos y lo pueden realizar en la base de usuarios que nos brinda Google permitiendo proteger tú IP mediante el Scrambler de Ruby. Scrambler permite encriptar mensajes, codificando en tiempo real, siempre es el mismo cifrado no existe más de uno.[**44**]

API'S para J3dWorkBench. No existe API's que se integren con J3dWorkBench.

4.3. Pruebas de desempeño en integración.

Las pruebas realizadas en esta sección serán las pruebas de integración de los sistemas definidos en el punto anterior anteriormente, como se puedo observar los no todas las aplicaciones tienen API's de integración es así que únicamente se ara el análisis de las herramientas seleccionadas por la facilidad e integridad de sus API's. Las aplicaciones serán probadas en la integridad de los sistemas y rendimiento sobre los sistemas operativos Windows y Linux. Las pruebas se realizarán sobre el modelo de la figura 4.3.1 para medir la carga del sistema.

FIGURA 4.3.1. Escena de prueba para un test de integración y monitoreo.

Blender.
Pruebas sobre Windows. Las pruebas fueron realizadas para medir la integridad de los sistemas con los diferentes sistemas operativos como es la plataforma Windows. Realizaremos un monitoreo del rendimiento de la aplicación durante la carga de una escena completa sobre cada uno de los software, a continuación mostraremos los resultados del monitereo del consumo de CPU y consumo de Memoria.

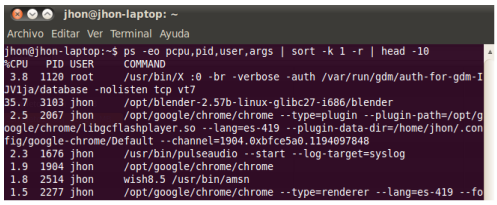

FIGURA 4.3.4. Blender consumo CPU.

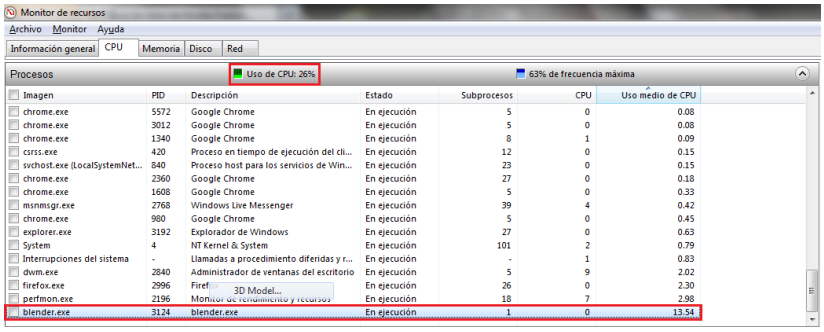

FIGURA 4.3.2. Monitoreo consumo de CPU Blender.

Resultados consumo de CPU.. Los resultados obtenidos fueron los siguientes.

Consumo del CPU 3.52 %.

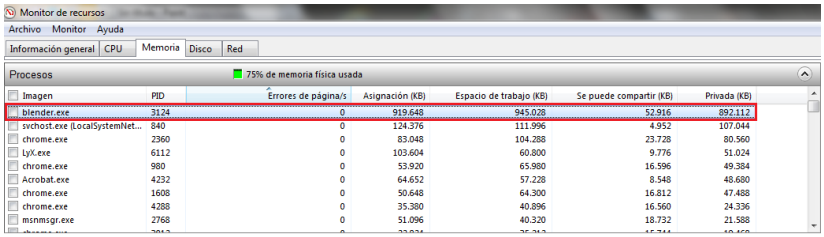

FIGURA 4.3.3. Monitoreo consumo de memoria Blender.

Resultados consumo de Memoria. El resultado de consumo de memoria fue el siguiente.

Consumo de memoria 892.112 KB -> 871,20 MB.

Pruebas sobre Linux. A continuación se presentará las pruebas sobre el sistema operativo Ubuntu 10.10, en el cual se obtuvieron los siguientes resultados tanto para consumo de CPU como consumo de memoria.

Resultados consumo de CPU.. Los resultados obtenidos de consumo fueron los siguientes.

Consumo del CPU 35.7 %.

Resultados consumo de Memoria. El resultado de consumo de memoria fue el siguiente.

Consumo de memoria 2,1 GiB.

Google Sketchup.

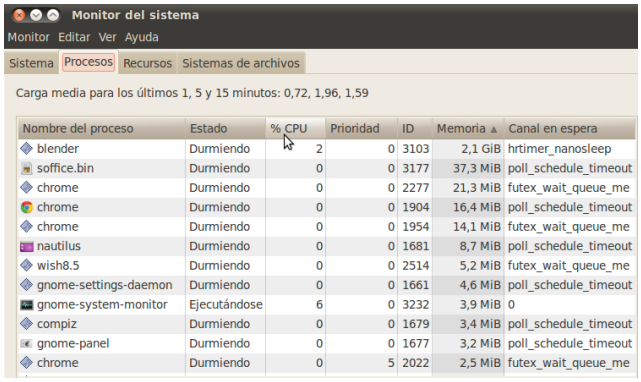

FIGURA 4.3.5. Blender consumo memoria.

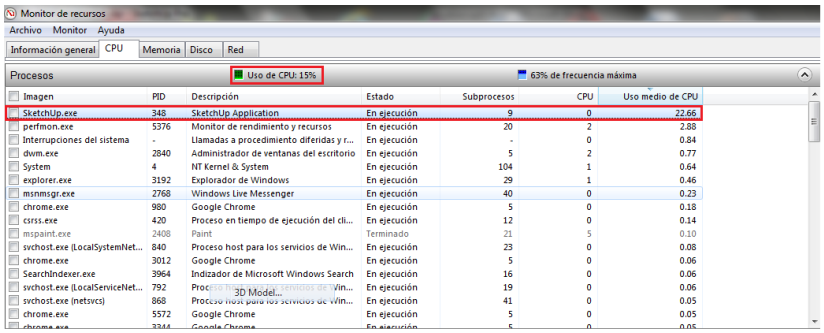

FIGURA 4.3.6. Monitoreo consumo de CPU en Google Sketchup.

Pruebas sobre Windows. Las pruebas se han realizado sobre el mismo equipo y sistema operativo en el cual se han realizado las pruebas anteriores en este caso realizaremos pruebas para medir la integridad de los sistemas con los diferentes sistemas operativos en este caso con bajo la plataforma Windows. Realizaremos un monitoreo del rendimiento de la aplicación durante la carga de una escena completa sobre Google Sketchup, a continuación mostraremos los resultados del monitereo del consumo de CPU y consumo de Memoria.
Resultados consumo de CPU. Los resultados obtenidos fueron los siguientes.

Consumo del CPU 3.39 %.

Resultados consumo de Memoria. El resultado de consumo de memoria fue el siguiente.

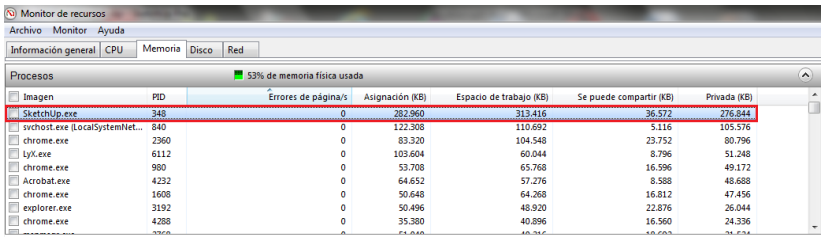

FIGURA 4.3.7. Monitoreo consumo de Memoria en Google Sketchup.

Consumo de memoria 276.844 KB -> 270,355 MB.

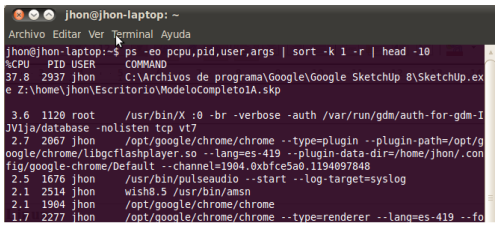

FIGURA 4.3.8. Sketchup consumo de CPU.

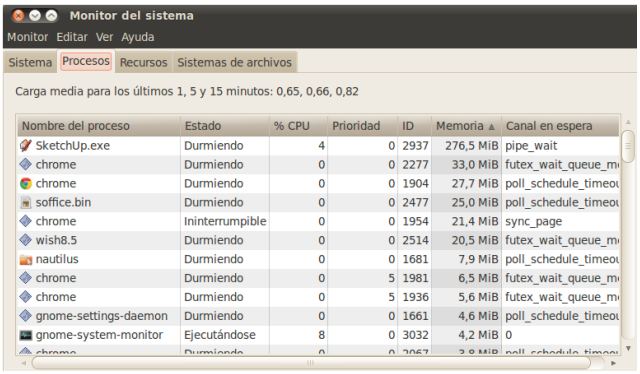

FIGURA 4.3.9. Sketchup consumo de memoria.

Google sketchup demostró ser una herramienta mucho más estable para realizar carga de modelos más pesados, extensión y con mucha información. El consumo de CPU para las dos aplicaciones se mantiene en estable y con una diferencia mínima en consumo de CPU, por el contrario el consumo de memoria demostró una notable diferencia entre las mimas, demostrando google sketchup una mejor administración de recursos.

Pruebas sobre Linux.

Resultados consumo de CPU.. Los resultados obtenidos fueron los siguientes.

Consumo del CPU 37.8 %.

Resultados consumo de Memoria. El resultado de consumo de memoria fue el siguiente.

El resultado de consumo de memoria fue el siguiente.

Consumo de memoria 276,5 MB.

4.4. Conclusión y síntesis de información.

Las herramientas seleccionadas tienen un poderío extraordinario en su área de desarrollo como componentes para representaciones 3D de escenarios, cada una tiene casos de éxito muy valiosos. Blender es una herramienta muy utilizada a nivel de cinematografía muchos films la han utilizado como herramienta principal para el desarrollo de sus escenas 3D, por otro lado en su última versión Blender integró la funcionalidad de aplicar como API de desarrollo al lenguaje Phyton. Con estas características se puede brindar personalización de los utilitarios de la herramienta, brindándonos la funcionalidad de crear

nuestros propios scripts. Google Sketchup, es una herramienta relativamente nueva en el mercado del desarrollo 3D, al igual que Blender brinda de funcionalidades excelentes para el desarrollo de Escenarios y Representaciones 3D, es una herramienta con una línea de aprendizaje muy baja demostrando una integridad sencilla del usuario para usar la aplicación como su herramienta de soporte en modelado 3D. Actualmente Sketchup posee un API de Ruby con más de 500 funciones que nos permiten personalizar el funcionamiento de la aplicación, como la definición de nuevas herramientas de modelado que nos permitan minimizar tiempo de desarrollo de aplicaciones 3D. Esto brinda la ventaja de integrar funcionalidad sobre la aplicación es decir, crear botones en la barra de herramientas, crear nuevas funciones, crear menús, etc.

Capítulo 5

Utilitarios para el desarrollo de ambientes virtuales 3d.

5.1. Google Sketchup.

Es una herramienta informática para diseño y desarrollo 3D publicado por Google que nos permite edificar pequeños o grandes volúmenes de datos dándoles formas arquitectónicas en 3D. Google Sketchup permite solucionar problemas, tener una mejor visualización a cerca de un objeto pudiendo simular los mismos y así poder aprender de manera creativa y tener un diseño de alto nivel. [**39**]

5.1.1. Conceptos básicos de Google Sketchup. Todas las personas por razones desconocidas siempre han tenido curiosidad de conocer el modelado 3d pero no han tenido la oportunidad de entrar en este mundo, pues ahora tenemos esa gran oportunidad con la herramienta Google Sketchup para poder crear nuestros propios modelos.

5.1.2. Características Principales.

- Las imágenes 2D pueden ser situadas delante o detrás de los modelos para obtener un fondo como paisaje.
- Tiene efectos de boceto, al tener un aspecto de dibujo a mano.
- Crea y modifica modelos 3D.
- Da la posibilidad de exportar archivos JPG, TIF, PNG y BMP e importar archivos 3DS, DXF y DWG.
- Comienza el diseño mediante rectángulos, círculos, óvalos, arcos, líneas y dibujos a mano alzada como lo harías en papel.
- Sin que se distorcione la imagen se puede obtener escalas diferentes.
- Admite colocar texturas, colores o materiales como ladrillo, hormigón, madera, vidrio, alfombras, baldosa, cerámica, etc. en los modelos 3D creados de forma individual.
- Posee una interoperabilidad con google Earth.
- Interfaz amigable sobre otros programas que desarrollan de objetos 3D permitiendo la utilización en niños y adolescentes aficionados. [**25**]

5.1.3. Sketchup y OpenGl. Además de disponer de una CPU rápida y de una gran cantidad de RAM, la tarjeta de vídeo y los controladores deben ser totalmente compatibles con OpenGL.

Como ya es de nuestro conocimiento OpenGL es el estándar que utilizan numerosas aplicaciones de software y juegos para dibujar geometrías en 3D. La mayoría de los sistemas operativos de Microsoft Windows y Apple OS X incorporan un controlador de software OpenGL. No obstante, estos controladores se basan en la CPU para realizar los cálculos de procesamiento de OpenGL, una tarea en la que la mayoría de las CPU no son demasiado eficáces.[**46**]

Aceleración por hardware. SketchUp permite beneficiarse de la aceleración por hardware si el equipo dispone de una tarjeta de vídeo totalmente compatible con OpenGL. Lamentablemente, sólo algunas tarjetas de vídeo incluyen controladores 3D totalmente compatibles con OpenGL y que pueden, por tanto, utilizar esta característica (por más que muchos fabricantes afirmen que sus tarjetas sí son totalmente compatibles con OpenGL). La mayoría de controladores 3D están diseñados para juegos y muchas veces no llegan a probarse con otros programas de 3D. En consecuencia, se pueden presentar problemas de incompatibilidad que requieran la corrección de ciertos fallos por parte del fabricante de la tarjeta de vídeo. Se debe desactivar esta opción si existen problemas con el procesamiento en 3D de los modelos, o bien si la tarjeta de vídeo que utilizas no es totalmente compatible con OpenGL y no es compatible con la aceleración por hardware.

Google no puede controlar la calidad del controlador OpenGL instalado en cada equipo, ya que son responsabilidad de los fabricantes. Por ese motivo, Google no puede garantizar que SketchUp funcione bien en todos los equipos con aceleración por hardware. [**46**]

Problemas de compatibilidad. Los problemas con las herramientas, el rendimiento y el procesamiento en SketchUp suelen deberse a que la tarjeta de vídeo no es totalmente compatible con OpenGL, a que el controlador de la tarjeta es antiguo o a la incompatibilidad con el color en 32 bits. Una solución provisional consiste en desactivar la aceleración por hardware en SketchUp hasta que el problema haya podido corregirse. [**46**]

5.1.4. Funcionalidades básicas de la herramienta. Google Sketchup quiere decir “BOSQUEJO” siendo esta la principal función de la herramienta. Permite crear y modificar plantillas para poder trabajar sobre las mismas como un punto inicial, y ofreciéndo una vista con tres ejes de colores x,y,z que representan el ancho, alto y profundidad respectivamente, son coordenadas de Sketcup integradas directamente con Google Earth, las cuales nos facilitan la construcción del modelo seleccionado. Además de exportar como una imagen, como un video o simplemente imprimirla a gusto de cada usuario una vez terminado el proyecto.[**21**]

Configurar Sketchup. Cuando se abre por primera vez la herramienta de Sketchup se presenta una pantalla en la cual elegiremos la plantilla sobre la que se creara nuestro

proyecto, dar clic en Empezar a utilizar Sketchup y se iniciara con la plantilla que se seleccionó. Las plantillas que ofrece Sketchup se listan a continuación:

- Plantilla sencilla-Metros.
- Diseño arquitectónico- pies y pulgadas
- Diseño arquitectónico – milímetros
- Modelos para Google earth- pies y pulgadas
- Modelos para Google Earth – metros
- Ingeniería – pies y pulgadas
- Ingeniería – metros
- Diseño de producto y carpinterías – pies y pulgadas
- Disenso de producto y carpinterías – milímetros
- Vista de plano – pies y pulgadas
- Vista de plano - metros
- Plantilla de aprendizaje básico – pies y pulgadas.
- Plantilla de aprendizaje básico – metros. [**21**]

Crear documento nuevo. Sketchup crea nuevos documentos con los diseños y configuraciones basados de acuerdo a las plantillas escogidas desde Ventana – Preferencias – Plantilla.[**21**]

Abrir un documento. Sketchup no permite abrir más de un documento en Windows si se desea abrir dos documentos al mismo tiempo se tiene que iniciar dos veces el programa para cualquier manipulación, en la plataforma MAC este problema no se presenta, se puede abrir más de un documento en la misma sesión de Sketchup.[**21**]

Entidades de Sketchup.

- Línea: Denominadas también aristas son la base fundamental para la construcción de cada uno de los modelos, Sketchup usa líneas rectas.
- Cara: Se presentan de lado frontal y posterior, el cual Sketchup siempre trata de situar de lado frontal, las caras se crean de forma automática cuando más de tres aristas comparten un plano y forman un blucle.
- Círculo, arco, polígono, curva: Compuestas por varias líneas.
- Trazo a mano alzada: Diseñadas por líneas.
- Grupos: Combinan más de una entidad para realizar operaciones rápidas.
- Componente: Similares a los grupos con la diferencia de que son reutilizables después de ser guardados.
- Línea auxiliar: Es una línea que se utiliza como una guía para poder dibujar.
- Anotación: Indica la longitud de una línea o un radio.
- Superficie: Resultado de combinar varias caras para obtener un modelo con volumen. [**40**]

Sistema de Coordenadas. Los puntos se identifican en un espacio mediante su posición con 3 ejes de dibujo X,Y,Z ya sea por encima o por debajo del suelo para obtener un mayor detalle y visibilidad de los objetos. El eje x se representa mediante una línea continua o punteada de color verde, el eje Y, Z de la misma manera pero en color rojo y azul respectivamente. Los ejes de X, Y es representado como plano del suelo, es importante entender que los sistemas de coordenadas que maneja Google Sketchup ayuda al diseño con mayor precisión.[**40**]

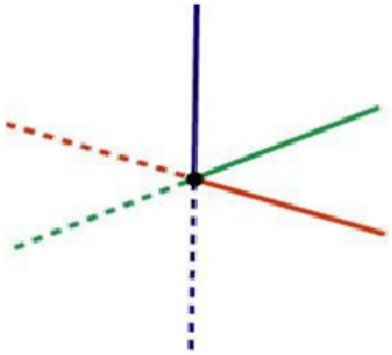

FIGURA 5.1.1. Sistema de Coordenadas en Sketchup [**40**]

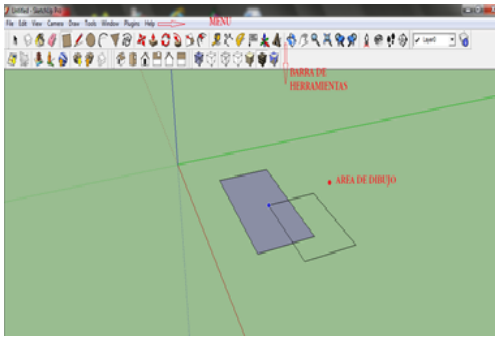

FIGURA 5.1.2. Visualización del modelo.

5.1.4.1. Visualización del modelo. Google Sketchup posee un modo inteligente al visualizar los modelos que se crean gestionando las opciones en los cambios de cada vista que sean necesarios para ir asegurándonos de que se definan bien los puntos e intersecciones.

Con Sketchup tenemos la posibilidad de mirar dentro del modelo creado para obtener una visualización completa ya sea interior como exterior.

Acercar y alejar. Los modelos creados en Sketchup se visualizan de forma sencilla, teniendo las opciones de acercar para mayor detalle de la creación y la opción alejar para tener una visión general.[**21**]

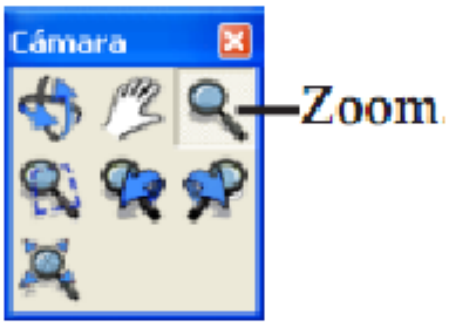

FIGURA 5.1.3. Herramienta zoom

Orbitar un proyecto. Se puede sobrevolar sobre el proyecto desarrollado, dando presentaciones del modelo en diferentes perspectivas, + - 360 grados.[**21**]

FIGURA 5.1.4. Herramienta orbitar

Ampliar áreas determinadas. Se puede ampliar determinadas áreas del modelo en creación.[**21**]

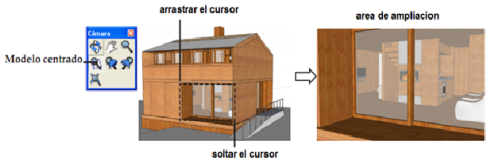

FIGURA 5.1.5. Ampliación de figuras[**21**]

5.1.4.2. Representación de objetos en 3d. Google Sketchup para representar sus modelos introduce el concepto de cámara y la manipulación de un modelo en un espacio 3D, existen dos categorías en las que se pueden representar objetos 3D que son los siguientes:

- Representación por frontera.
- División del espacio.

Representación por frontera. Mediante un conjunto de superficies se describe los objetos 3D los cuales separan la parte interna del objeto de la escena virtual como ejemplo se puede mencionar a las superficies paramétricas e implícitas.

División del espacio. Este tipo de representación se utiliza para poder entender y describir las propiedades internas de los modelos.

5.1.4.3. Herramientas de manipulación básicas.

Motor de Inferencias. Incorpora un motor de inferencias invisible que sirve para dibujar los objetos con mayor precisión e infiere sobre los puntos a partir de otros modelos, como el punto medio de una línea, el centro de un circulo, etc diferenciando los puntos con indicadores de colores.[**52**]

Por ejemplo tenemos cinco inferencias diferentes en la siguiente imagen.

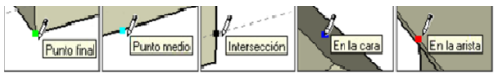

FIGURA 5.1.6. Inferencias

Dibujar con rapidez. Sketchup no solo ofrece una línea para facilitar los modelos que se deseen crear sino, que también tiene incluido círculos y arcos que son un conjunto de combinación de aristas pequeñas.[**52**]

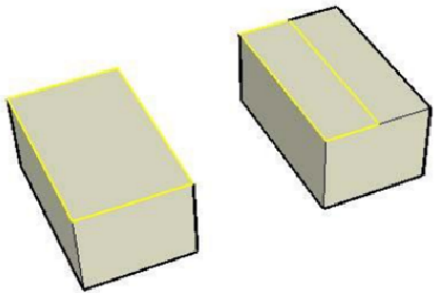

FIGURA 5.1.7. División de un objeto[**52**]

Empujar y Tirar. Esta herramienta nos permite empujar de una parte de las caras del modelo ya sea hacia cualquier dirección con pequeñas limitaciones para cambiar la forma desde un punto inicial hacia el largo de uno de los ejes escogidos. En la imagen podemos apreciar y darnos cuenta que se ha realizado la acción de “Empujar y Tirar” dándonos como resultado 8 caras en el modelo para poder manipularlas. [**52**]

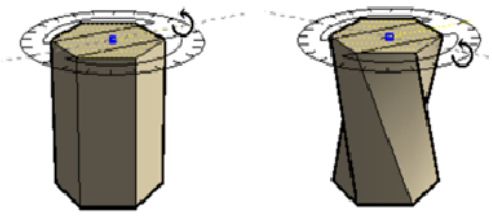

FIGURA 5.1.9. Figuras auto plegadas [**40**]

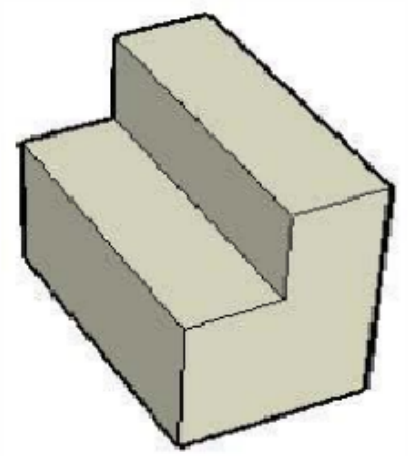

FIGURA 5.1.8. Empujar una cara del modelo.[**52**]

Auto plegado. Sketchup permite también la rotación de una de las caras. Tomamos el ejemplo de un polígono de 6 aristas y en la cara superior aplicamos la herramienta de Girar.[**52**]

5.1.5. Funcionalidades avanzadas de la herramienta. Google Sketchup nos permite realizar funciones avanzadas para poder crear modelos más reales, cada vez se tiene versiones nuevas de Google Sketchup mejorando sus funcionalidades para brindar mejor soporte.

5.1.5.1. Modelado en capas. Skechup trabaja de forma muy diferente a los demás programas que desarrollan objetos 3D, estas no actúan como capas, más bien son una propiedad para la visualización de los modelos creados, cuando hablamos de capas los elementos de Sketchup permanecen conectados unos con otros, pero si estos modelos son parte de una misma entidad geométrica.

En primera instancia Sketchup trabaja con una capa base, que es la capa cero (0) donde se puede colocar las entidades individuales. Si se desea organizar un modelo mediante capas, las entidades tendían que organizarse en grupos o componentes para luego ser asociadas.

Para poder controlar la visibilidad de la geometría de los grupos, las capas tienen que estar asociadas y se podrá ver los modelos organizados, se puede activar o desactivar la visibilidad de varias y cada una de las partes de los modelos creados, las capas no controlan el color de los modelos, se organizan para poder diferenciar las geometrías. [**39**] [**52**]

FIGURA 5.1.10. Modelo creado en capas [**52**]

5.1.5.2. Creación de Componentes propios. Sketchup permite poder crear propios componentes como pueden ser gradas, carros, personas, etc, y reutilizarlos para otros modelos.

Se puede crear componentes que sean configurables para suplantar a los componentes estáticos. Cuando realizamos la creación de un componente se desconectan las geometrías que suelen ser conectadas antes de la selección para adjuntarse a un componente. [**42**]

Crear un nuevo componente. Para la creación de los componentes es importante tener en cuenta la forma en la que será insertada en el modelo desde el explorador de componentes. Debemos tener en cuenta los siguientes pasos para crear un componente.[**42**]

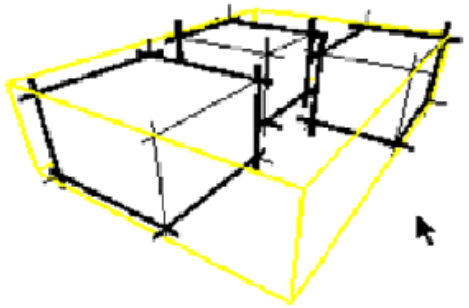

FIGURA 5.1.11. Componente

- Activar la herramienta de selección, el cursor adquiere la forma de una flecha.
- Dar clic y mantener presionado el botón del ratón a una corta distancia de las entidades que se desea seleccionar para iniciar u cuadro de selección de componentes.
- Arrastrar el ratón a la punto puesto de la primera selección.
- Soltar el ratón cuando todos los elementos están incluidos para crear el componente.

- Seleccionar el menú crear componente desde la barra de herramientas o elegir crear componente en el menú de contexto. [**42**]

Definición de componentes y copias. Los componentes al ser creados mediante el cuadro de dialogo "crear componente" se crea una definición y una copia del mismo, la definición de un componente describe el funcionamiento de todos los componentes de un tipo específico y son representadas en imágenes muy pequeñas por el explorador de componentes. En cambio las copias de los componentes son definiciones insertadas en el área de dibujo, todas tienen el mismo funcionamiento y se les puede manipular en el área de dibujo una a la vez sin que afecte a los demás componentes.

Los componentes se insertan en el modelo desde un explorador de componentes, también se tiene la posibilidad de hacer la inserción desde un archivo externo de Sketchup con extensión .skp. [**42**]

- Seleccionar la opción menú Archivo >> Importar >> Modelo 3D.
- En el cuadro de dialogo abrir seleccionar el tipo de archivo .skp.
- Seleccionar abrir. [**42**]

5.1.5.3. Importar un Dibujo CAD.. Sketchup es una herramienta que permite importar modelos CAD, esto se puede realizar mediante la opción "Importar" que se encuentra en el menú Archivo.

Los pasos a seguir de la importación de archivos 3D se aclaran en lo siguiente.

- Seleccionar la opción Importar.
- Seleccionar el tipo de archivo que se desee importar.
- Para modificar las opciones de la importación del archivo, seleccionamos el botón Opciones.
- Seleccionamos OK para importar el archivo. [**52**]

Preparar el archivio CAD para la Importación. Para realizar la importación de archivos en Sketchup no se requiere de mayores conocimientos, pero vale la pena hacer una pequeña preparación para que funcionen los datos de una mejor manera. Sketchup no reconoce todas las entidades de los archivos CAD importado que no tengan relevancia 3D como, textos, patrones, logotipos, etc, pero no elimina las capas que contienen estas entidades, y se puede eliminar en el mismo archivo CAD o en Google Sketchup de forma más sencilla utilizando el comando "Purgar" que elimina las capas que no se utilizan.

Se genera problemas de rendimiento cuando los archivos importados están muy lejos del origen (0,0) de Sketchup, por esta razon se recomienda acercarse al origen antes de importar el archivo.[**52**]

5.1.5.4. Imprimir a escala. Los modelos diseñados en sketchup se pueden imprimir a escala pudiendo tener impresiones de gran tamaño.

Cada campo de escala se usa para poder imprimir el modelo, modificando la escala, el modelo que se exporta tiene una primera medida que corresponde al tamaño de su geometría, la segunda medida en Sketcup es el tamaño del objeto a escala real. Por ejemplo si deseamos imprimir un modelo para una escala de 1/4" =1 solo se debe especificar que 1 pulgada en la impresión equivale a 4 pies en Skepchup.

No existe la posibilidad de imprimir a escala un modelo o imagen en perspectiva, se debe estar seguros de que la perspectiva este desactivada para utilizar las vistas estándares que se encuentra en el menú Cámara >> Estándar y activar la opción de escala.[**39**]

Pasos para imprimir un modelo a escala.

- Paso 1: El modelo se pasa a modo paralelo desactivando el modo de perspectiva, para ello se dirige al menú de herramientas "Cámara" y selecciona la opción "Perspectiva ".
- Paso 2: Se debe elegir una vista escalable que sea estándar que pueden ser "Isometrica, planta, frontal, derecha, posterior o izquierda" seleccionando en el menú cámara la opción "Vistas Estándar".
- Paso 3: Seleccionamos en el menú "Archivo" la opción "Imprimir".
- Paso 4: Desactivar "Ajustar a pagina".

5.1.5.5. Exportar dibujos vectoriales 2D.. En Sketchup se puede realizar la exportación de imágenes 2D, como imágenes raster a los formatos de archivos .JPG, BMP,TGA, PNG de una manera sencilla con la opción "Exportar >> Gráficos 2D".

Las funciones especiales de visualización tales como sombras, texturas o transparencias, no son compatibles con ningún formato de exportación de vectores 2D, los formatos aceptados para la exportación que represente las sombras de los modelos son las imágenes raster basadas en pixeles o también las animaciones. Se utiliza la vista actual para la exportacion.

Pasos para la exportación de dibujos 2D..

- Paso1: Ajustar la visualizacion del modelo que sera exportado.
- Paso 2: En el menú de herramientas seleccionar Archivo > Exportar > Gráfico 2D .
- Paso 3: En el cuadro de dialogo escribir un nombre para el archivo de exportación.
- Paso 4: Seleccionar el tipo de exportación en la lista desplegable, y dar clic en el botón exportar.

5.1.6. Consejos para animación. Sketchup no esta disenada prescisamente para animaciones, pero existen plugin para realizar esta tarea como también la opción "TourGuide" que es similar a los movimientos vectoriales.

Para realizar las animaciones de un proyecto se puede realizar mediante la creación de diferentes páginas de diseño y cada una de las mismas mover el modelo a la posición final que deseemos.

Pasos para realizar una animación.

- Paso 1: Seleccionamos en el menú de herramientas la opción “Ver” y escogemos “Tourguide” y luego añadir página.
- Paso 2: Rotar las escenas hacia la posición final.
- Paso 3: Para verificar los resultados de la animación nos vamos al menú herramienta , opción Ver-Tourguide-Presentación.

5.1.7. API de programación para Ruby. La aplicación de programación Ruby para Google Sketchup permite crear complementos como generadores automáticos de componentes y otras opciones que se podrá incluir en las barra de herramienta de la interfaz de Google Sketchup. La herramienta contiene una consola de Ruby que no es más que un ambiente para poder jugar y experimentar con los comandos de Ruby.

Existen pequeños programas escritos, a los que se les denomina complementos, Plugins o simplemente Scripts de Ruby, los cuales se les puede modificar y crear un propio componente según las necesidades de cada usuario automatizando las tareas y funciones en Sketchup.[**39**]

Se puede implementar API’S para cualquier necesidad como por ejemplo:

- Complemento para encontrar todos los puntos o aristas sueltas de un modelo y que las elimine de forma automática.
- Crear una nueva herramienta “Rectángulo” con funciones diferentes a las que existen.

Estructuras básicas de SketchUp. Casi todos los scripts de SketchUp comienzan accediendo a tres estructuras de datos básicos que son las siguientes.

Sketchup, modelo, y las entidades. [**44**]

El módulo de Sketchup. A diferencia de las clases y los objetos, la primera estructura de datos de Sketchup, es un módulo, que es una colección de métodos. Los métodos en las propiedades de acceso al módulo de Sketchup relacionados con la aplicación SketchUp en su conjunto. Podemos ver las funciones cuando abrimos la consola de Ruby. Ventana> Consola Ruby de SketchUp en el menú principal y ejecutar el siguiente comando:

Sketchup.versión

“Sketchup.versión” muestra el número de versión de la aplicación actual de SketchUp. También puede introducir Sketchup.os_language para determinar el lenguaje común o para acceder a Sketchup.get_locale designador de SketchUp del medio ambiente.

Para ver todos los métodos proporcionados por el módulo de Sketchup, el cual se encuentra en el listado de SketchUp en un apéndice o entrar “Sketchup.methods” en la consola de Ruby.

El método más importante en el módulo de Sketchup es “active_model”, que devuelve el objeto de modelo correspondiente al diseño de SketchUp abierto. El comando siguiente muestra cómo funciona este método:

mod = Sketchup.active_model[**44**]

El modelo de objetos. Al igual que el módulo anterior representa toda la aplicación SketchUp, el modelo de objetos representa un único archivo de SketchUp (skp *.), o más exactamente, la información de diseño que contiene el archivo. Cuando se abre un nuevo archivo, las propiedades del módulo de Sketchup siguen siendo las mismos, pero los datos en el modelo de objetos activo se convierte y es totalmente diferente. Los métodos de la clase Modelo proporcionan información sobre el diseño actual. Por ejemplo, el método “modified?” identifica si el diseño actual se ha modificado desde la última vez que se guardó. Los comandos siguientes muestran cómo se utiliza:

mod = Sketchup.active_model mod.modified?

El método “title” devuelve el título del diseño actual, “description” devuelve su descripción de texto, “path” devuelve la ruta del archivo que contiene el diseño actual. [**44**]

Las entidades de objetos. Todos los objetos geométricos en un diseño de SketchUp está representado por una entidad o una de sus subclases, incluyendo líneas, caras, imágenes, textos, grupos y componentes. Para controlar o modificar objetos de la entidad en un diseño, es necesario acceder primero al diseño de entidades. Este contenedor tiene tres propósitos principales:

1. Agrega nuevos objetos a la entidad para el diseño actual de SketchUp.

2. Mueve, cambia de escala, aplica rotación, y los objetos se borran la entidad en el diseño

3. Tiendas de objetos de la entidad en una matriz que se puede acceder por el índice[**44**]

La clase de entidades contiene muchos métodos que agregar nuevos objetos a la entidad el diseño actual. Además de los métodos más simples (add_X) son las siguientes:

- add_line - crea un objeto de borde a partir de dos puntos.
- add_edges - forma una matriz de objetos de borde de una serie de puntos.
- add_circle - forma una matriz de objetos de borde que se combinan para formar un círculo.
- add_ngon - forma una matriz de objetos de borde que se combinan para formar un polígono.
- add_face - crea un objeto de la cara de los bordes o los puntos.
- add_text - añade una etiqueta con el diseño en un momento dado.[**44**]

Objetos. Para construir un modelo a gran escala, es necesario especificar los valores de muchas características, incluyendo las coordenadas, materiales, texturas y colores. Estos valores son de fácil acceso en el diseño de la ventana, pero en el software, la gestión de esta cantidad de datos es una tarea difícil. Se organiza en características relacionadas con las estructuras de datos jerárquicos. Por ejemplo, si estamos modelando una casa, vamos a crear una estructura de datos global de la casa y las sub-estructuras de sus paredes, puertas y techo. Una sub-estructura puede contener sub-sub-estructuras como el modelo de la puerta, la cerradura estas estructuras de datos se llaman objetos o variables, pero a diferencia de una variable, un objeto contiene varios valores relacionados. Por ejemplo, mientras "door_height" identifica la altura de una puerta, un objeto de puerta pueden contener tipos de valores como la altura, ancho, profundidad, dos objetos del mismo tipo deben tener las mismas características, pero no necesariamente los mismos valores..[**44**]

Clases. Una clase define la estructura de un objeto de la misma manera que un conjunto de planos define la estructura de un edificio. Una clase identifica los datos contenidos en un objeto y los métodos disponibles para operar en los datos del objeto. Entre la biblioteca Ruby y la API de SketchUp, hay cientos de clases disponibles para crear objetos de clases existentes. En Ruby, todo lo que trabajamos es un objeto, por lo tanto, se tiene que trabajar con una clase. El método de la clase muestra el nombre de la clase de un objeto, como se muestra en el siguiente código: [**44**]

5. class Fixnum

3.14159.class Float

"Hello, world".class String

[5, 6, 7].class Array.

Como se muestra:

• 5.class es un objeto de clase Fixnum, es un número de punto fijo.

• 3.14159 es un objeto de clase de acciones en circulación, un número de punto flotante.

• "Hello, world".class, es un objeto de clase String.

• [5, 6, 7]. class es un objeto de la clase Array. [**44**]

El Fixnum, Float, String, matriz y las clases son proporcionados por la biblioteca estándar de Ruby. En la API de SketchUp las clases más importantes son.

- Edge - un objeto creado a partir de la clase Edge representa un segmento de línea en SketchUp.
- Face- un objeto creado a partir de la clase Face representa una superficie de dos dimensiones en SketchUp. [**44**]

Herencia de clases. Puede ser que un arquitecto, requiere el acceso a clases que tengan caracteristicas en comun, como hoteles y hospitales, las dos estructuras son lo suficientemente diferentes como para requerir clases separadas: hotel y hospital. Sin embargo, las

dos clases también contienen características similares, tales como la ubicación, material, y el número de historias. Por esta razón, los siguientes métodos podrían aplicarse igualmente a cualquiera de los hoteles o las clases de Hospital. [**44**]

- num_stories: el número de historias en la estructura.
- Location: ubicación geográfica de la estructura.
- Material: el tipo de material utilizado para construir la estructura.[**44**]

En lugar poner el mismo código para 2 clases, es más eficiente colocarlos en una tercera clase por lo que estará a disposición de los objetos del hotel y el hospital. Esta tercera clase debe incorporar los elementos comunes de las dos clases, ahora se puede definir num_stories, ubicación y métodos de la clase de material en un solo edificio, y si es necesario volver a escribir cualquiera de ellos, sólo tiene que modificar una sola clase.[**44**]

Scripts de Ruby. Se puede escribir un script en cualquier editor de texto, pero se debe tener en cuenta tres puntos importantes.

1. Los comandos deben de ser escritos en el lenguaje de programación Ruby.

2. Sin encriptar los archivos script que tienen nombres que terminan con el sufijo rb. Los nombres de los archivos de comandos cifrada debe terminar con. RBS.

3. Para ejecutar un script, se debe cargar en SketchUp.[**44**]

Uno de los comandos de Ruby más utilizado es la carga, cuando esta se ejecuta en la consola, la carga en SketchUp lee una secuencia de comandos y ejecuta cada uno de sus mandatos en secuencia.

Construcción de una cara. Los objetos de la cara son creados por el método de la clase add_face entidades, el que acepta una serie de aristas, y puede ser proporcionada en una lista separada por una matriz. [**44**]

Ejemplo:

```
# Create the five points of the pentagon
pt1 = [0, 1, 0]
pt2 = [-0.951, 0.309, 0]
pt3 = [-0.588, -0.809, 0]
pt4 = [ 0.588, -0.809, 0]
pt5 = [ 0.951, 0.309, 0][44]
# Draw the face
pent = Sketchup.active_model.entities.add_face pt1, pt2, pt3, pt4, pt5
# Display the locations of the stored vertices
puts "Point 0: " + pent.vertices[0].position.to_s
puts "Point 1: " + pent.vertices[1].position.to_s
puts "Point 2: " + pent.vertices[2].position.to_s
puts "Point 3: " + pent.vertices[3].position.to_s
```

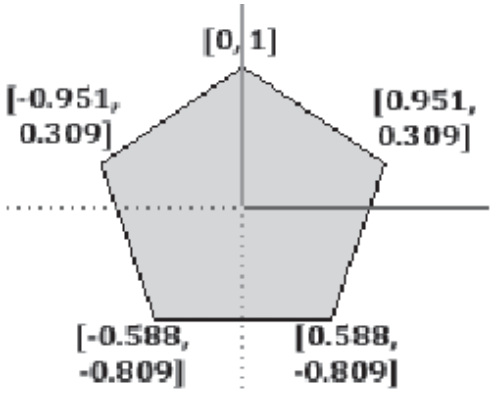

FIGURA 5.1.12. modelo para crear una cara. [**44**]

puts "Point 4: " + pent.vertices[4].position.to_s [**44**]

La figura que se muestra es el resultado del código descrito.

Imágenes. Hay dos formas de crear un gráfico, como una textura o una imagen. Una imagen, es una entidad independiente, como una arista o una cara que se crea con el método de la clase add_image, que acepta tres parámetros: el nombre del archivo de la imagen, el punto donde se debe colocar la imagen, y la anchura deseada de la misma.[**44**]

Para las imágenes, SketchUp acepta *. jpg, *. png, *. bmp, *. tga y *. bmp. de manera predeterminada, la imagen en sus dimensiones tanto altura como el ancho se mantiene constante, pero puede agregar un parámetro opcional que identifica a la altura por separado.[**44**]

Una vez que se crea un objeto como imagen, se puede llamar a los métodos, que devuelven información sobre las dimensiones de la imagen. Por ejemplo, se puede llamar a la altura y profundidad.

5.1.8. API de programación para C++. El SDK de Google Sketchup contiene los siguientes elementos para API C++.

- SkpReader C++ API: Para leer información de un archivo skp y su transformación a otro formato o modelo de objetos.
- SkpWriter C++ API : Para construir un archivo skp de otro formato o modelo de datos de objeto.

El API de C++, SkpReader y de SkpWriter fueron escritos antes de la adquisición de SketchUp de Google y se encuentra en constante desarrollo. [**8**]

SkpReader C++. El SkpReader C + + proporciona acceso de sólo lectura a la información en un modelo de SketchUp y está disponible a través de dos mecanismos: [**8**]

- La aplicación SketchUp en sí mismo, y
- Binario SkpReader independiente.

Con el método SkpReader C++ se puede.[**8**]

- Escribir un exportador plug-in para la aplicación SketchUp.
- Escribir un skp independiente para convertir formatos de .xyz
- Escribir un importador de archivos. Skp para su aplicación.

Las aplicaciones típicas que se construyen con la API de SkpReader incluyen:[**8**]

- Plugin para SketchUp de exportadores.
- Convertidores independientes .Skp a archivo . Xyz.
- Importadores archivo . skp de para otras aplicaciones de diseño.

Nombres de las interfaces.

Las interfaces API SkpReader siempre su prefijo es ISkp.

SkpWriter C++. El SkpWriter C++ API, ofrece la posibilidad de escribir un archivo de SketchUp con formato .skp. La API está disponible como un binario SkpWriter independiente.[**8**]

Con este método se puede.

- Escribir un importador plugin para la aplicación SketchUp.
- Escribir un independiente. Xyz a archivo. Skp convertidor de formato.
- Escribir un exportador de archivo. Skp para su aplicación.

5.1.9. Pruebas de campo con ambientes reales. Para las pruebas de campo con ambientes reales hemos tomado como objetivo principal la infraestructura de la Universidad Politécnica Salesiana, para ello se recolecto un conjunto de fotografías, en el cual nos basamos para realizar dichas pruebas (el conjunto de fotografías estará adjunto como anexo de la documentación de la tesis). A continuación se detallara progresivamente el desempeño de la herramienta, diseñando en base a las fotografías la estructura de la universidad.

En primera instancia utilizamos componentes de la biblioteca de Google Sketchup 3D Warehouse.

En esta ocasión para la creación de un nuevo componente, trabajamos con un componente que ya fue compartido en 3D Warehouse, para lo cual únicamente, lo adaptamos a nuestras necesidades, modificando texturas, colores, tamaño, etc. Nuestro modelo representara las sillas ubicadas en el jardín central de la Universidad, para lo cual nos basamos en la siguiente fotografía.

FIGURA 5.1.13. Modelo real

Luego de los cambios en el modelo compartido el boceto resultante que obtuvimos fue el siguiente.

FIGURA 5.1.14. Modelo modificado.

Representación de los faroles ubicados en el parque de la universidad.

FIGURA 5.1.15. Modelo real.

El modelo creado se representa a continuación.

FIGURA 5.1.16. Representación de un farol .

Representación de la entrada de la Universidad Politécnica Salesiana. Para realizar este trabajo nos basamos en algunas fotografías que podemos observar a continuación.

FIGURA 5.1.17. Modelo real de la entrada a la Universidad Politécnica Salesiana.

A continuación podemos ver el modelo fin de la creación de la entrada de la Universidad Politécnica Salesiana.

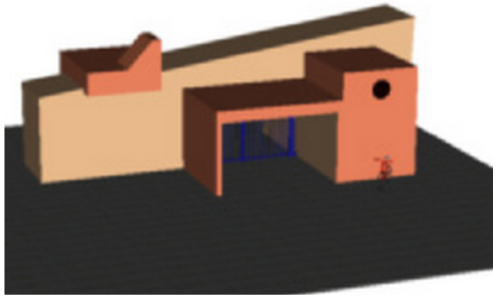

FIGURA 5.1.18. Modelo creado en Sketchup de la entrada de la UPS.

FIGURA 5.1.19. Modelo real.

Representación del edificio Cornelio Merchan. Modelo del edificio Cornelio Merchan realizado en Google Sketchup.

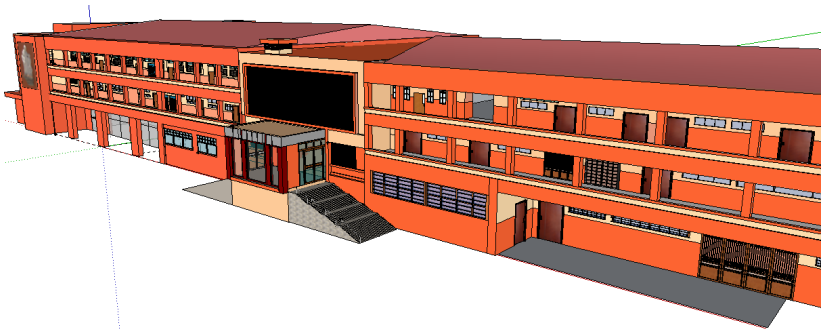

FIGURA 5.1.20. Edificio Cornelio Merchan - Sketchup

Representación del edificio Guillermo Menssi. Modelo del edificio Cornelio Merchan realizado en Google Sketchup.

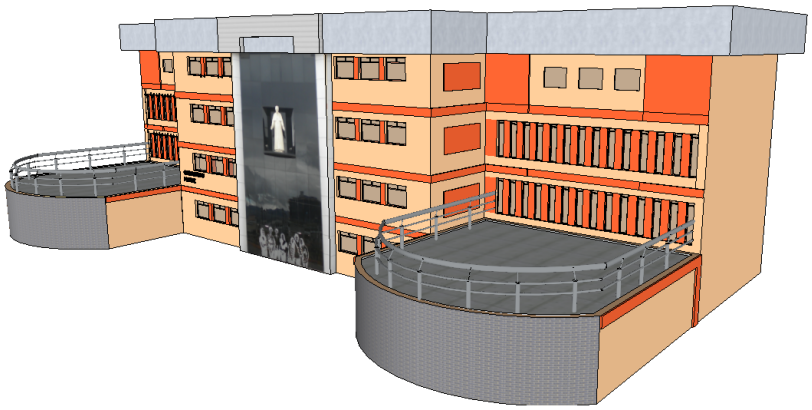

FIGURA 5.1.21. Modelo creado en Google Sketchup

FIGURA 5.1.22. Modelo real

Representación del edificio Mario Rizzini. Modelo del edificio Mario Rizzini realizado en Google Sketchup.

FIGURA 5.1.23. Edificio Mario Rizzini creado en Sketchup

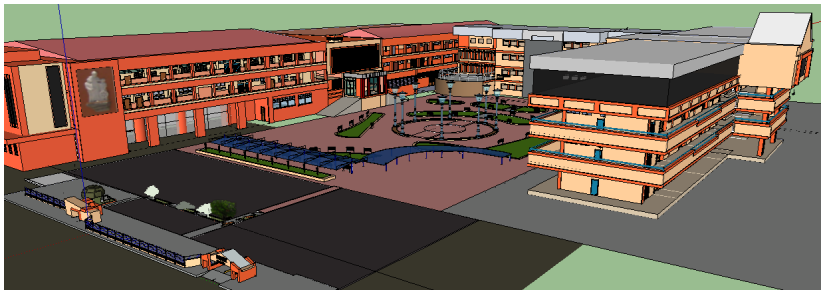

FIGURA 5.1.24. Modelo de Universidad Politécnica Salesiana desarrollado en Sketchup

Representación final del modelo completo de la Universidad Politécnica Salesiana.

5.2. Insigth 3D.

Esta herramienta permite crear modelos 3D a partir de una serie de fotografías de imágenes reales, calculando las posiciones en el espacio de cada una de las fotografías

que han sido tomadas para el diseño, siendo las suficientes para cubrir toda la zona que se desea reconstruir con los respectivos parámetros ópticos de la cámara, está a herramienta ofrece modelos a los cuales se les puede aplicar texturas. [**9**]

5.2.1. Conceptos básicos de Insigth 3D.. Para obtener un modelo que sea correctamente diseñado se deberá tomar en cuenta las siguientes consideraciones de las escenas.

- Todas las fotografías deben ser enfocadas y muy claras.
- No debe existir grandes solapamientos entre fotografías, varias de ellas se ven repetidas pero no lo son.
- El ángulo de las fotografías debe ser de 15 a 20 grados.
- Realizar modelos que tengan escenas con varios detalles que le caracterizan de forma única.
- Se debe evitar escenas que se encuentren en un solo plano en el espacio la herramienta no puede identificar y determinar la distancia focal.
- Las fotografías deben ser intactas sin ninguna manipulación.
- La escena que se quiere reconstruir por lo menos deberá ser divisible para 3. [**9**]

5.2.2. Características principales.

Concordancia y calibración de la cámara. Insight3d es una herramienta diseñada para poder determinar automáticamente todos los parámetros de las cámaras que se utilizan para los modelos que serán diseñados, se trata de buscar los puntos más importantes de las imágenes para dar un seguimiento posterior, calibrando las posiciones de la cámara y calculando a partir de los puntos seguidos, este procedimiento se puede realizar de manera muy sencilla solo haciendo clic “coincidencia de inicio”.[**9**]

Nube de puntos 3D.. Insight3d puede calcular las posiciones de todos los puntos de seguimiento en las fotos de muestra la nube de puntos que resulte debe ser similar a la escena que se quiere reconstruir, para ello se utiliza “Modelado > Triangular” y existirá una nube de puntos en 3D.[**9**]

Texturas. Esta herramienta también permite personalizar los modelos creados añadiendo texturas para dar realismo a los modelos pudiendo poner nuevas texturas o las mismas que se presentan en las fotografías. [**9**]

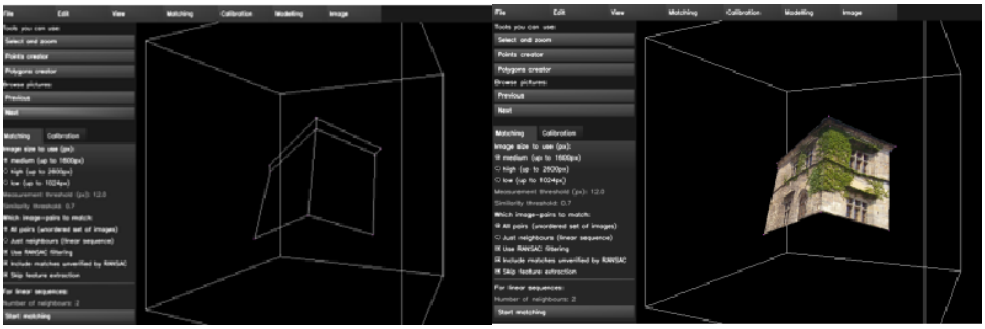

FIGURA 5.2.1. Aplicar texturas a los modelos. [**9**]

Exportar modelos. Los modelos realizados en insight 3D se pueden exportar a varios formatos, pero la que más se ha utilizado es VRML para poder realizar la exportación de los modelos creados en 3D a varias aplicaciones que sean compatibles como por ejemplo Blender [**9**]

5.2.3. Prueba de campo con ambientes reales. En la etapa de pruebas de Insigth3D se intentó realizar un test de la herramienta con ambientes reales la cual no tuvo éxito, puesto que la realizar las pruebas iniciales y más simples de la herramienta con ejemplos que contienen su paquete de instalación, no se pudo obtener ningún resultado favorable sin poder cargar una colección de fotos que permita realizar un proceso de vectorizacion e inflado 3D a partir de un grupo de muestras. El resultado obtenido a cargar dichas imágenes fue el siguiente.

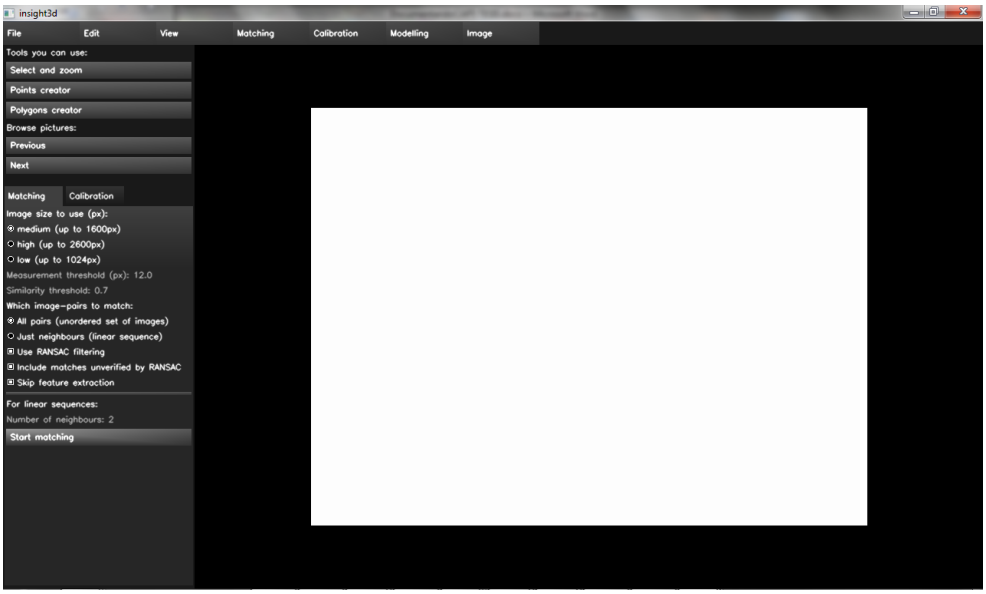

FIGURA 5.2.2. Resultado de pruebas Insigth 3D

Al intentar cargar la colección de fotos del ejemplo incluido, no se obtuvieron resultados favorables como podemos observar en la figura 5.17.

Los resultados obtenidos fue el mismo bajo la plataforma Windows como Linux.

5.3. Conclusión y síntesis de la información obtenida.

Google Sketchup demostró ser una herramientas muy poderosa que utiliza un paradigma de llegar a ser el más accesible para cualquier usuario, tratando de hacer el trabajo de que sus representación en el modelo sean lo más sencillas posibles.

Sketchup es una herramienta que elimina la frustración en la complejidad del aprendizaje de la herramienta, logra desempeñarse como una herramienta sencilla de utilizar e intuitiva minimizando al máximo la línea de aprendizaje. Sketchup como herramienta de diseño 3D aprovecha al máximo las características del hardware de nuestros equipos, además de integrarse completamente a las virtudes de OpenGL, su API de programación RUBY permite automatizar las tareas repetitivas en la creación de un modelo 3D, además de personalizar la funcionalidad de nuestro Google Sketchup con la facilidad de incluir componentes gráficos a la aplicación como menús, barra de herramientas, etc. Además contiene mucha información compartida por toda la comunidad de usuarios de la herramienta, compartiendo un sin número de componentes que permiten disminuir nuestro tiempo de desarrollo de aplicaciones 3D, la funcionalidad de Sketchup no solo permite la integración de desarrollo sobre la aplicación sino es más su funcionalidad extiende a la integración con aplicaciones externas con C++ a partir del SDK para incluirlo dentro de nuestras aplicaciones personales .

Una de las características más sobresalientes de la herramienta es la posibilidad de realizar exportaciones a diferentes formatos como obj, kmz, vrml, etc. Agregando a su funcionalidad la capacidad de integración con otras aplicaciones dedicadas al mismo objetivo.

Por otro lado insight3d demostró ser una aplicación que está en sus primeras etapas con varios bugs, que dejo mucho que desear de su funcionalidad, a pesar de esto es una herramienta muy prometedora y con una visión extraordinaria para minimizar tiempo de desarrollo de aplicaciones 3D a partir de una colección de fotografías, cabe recalcar que sería de muy buen aporte estar a la espera de sus nuevas versiones con el afán de realizar un test de una aplicación futurista y con un éxito prometedor.

5.4. Selección de las herramientas más adecuadas para llevar a cabo la tarea.

En esta ocasión la elección de la herramienta más adecuada, se basara en los resultados obtenidos a partir de todo el análisis de la información recolectada hasta el momento, las pruebas realizadas sobre cada una de las herramientas y la experiencia que logro desarrollarse durante el transcurso de la investigación nos permite seleccionar a Google Sketchup

como la herramienta más adecuada para el desarrollo de aplicaciones 3D, la cual fue seleccionada por varias ventajas frente al resto de herramientas con el mismo objetivo entre las ventajas competitivas para el desarrollo de aplicaciones 3D, tenemos las siguientes:

- Minimiza la línea de tiempo de aprendizaje de la herramienta para los nuevos usuarios.
- Integración de personalización y automatización de tareas dentro de la herramienta.
- Interacción con aplicaciones de desarrollo propio a través del SDK en C++.
- Permite la exportación a diferentes formatos como: obj, dae, kmx, 3ds, dgw, dxf, (Formatos de Autocad), fbx, vrml, etc. Extendiendo la integración con otras aplicaciones 3D.
- Importación de escenarios 3D realizado en otras aplicaciones indiferentemente de la plataforma de desarrollo de dichas aplicaciones. Consumo y desempeño adecuado de la herramienta bajo diferentes sistemas operativos.
- Logra sacar el mayor provecho posible a las características de software de nuestros equipos.

Por cada una de las características mencionadas ***GOOGLE SKETCHUP*** demostró ser la herramienta más adecuada para el desarrollo de aplicaciones 3D.

Conclusiones.

Al concluir con este proyecto de tesis podemos decir que la experiencia obtenida durante la investigación demuestra que el uso de herramientas de modelado y generación de ambientes virtuales 3D se constituye en una metología de soporte para las más variadas áreas, como son el diseño, la publicidad, el área médica, la construcción, etc.

Cuando se emplean herramientas de generación de escenas virtuales 3D a partir de fotografias, un factor que afecta considerablemente al tiempo de procesamiento es la cantidad de fotos y el espacio físico que requiere el objeto a representar. Asimismo, los resultados obtenidos con este tipo de programas no producen objetos 3D con una correcta geometría y estructura.

Otro aspecto importante, son los lenguajes de programación de ambientes virtuales. Entre los más robustos tenemos a OpenGL y a VRML, que generan resultados de muy buena calidad, aspecto que se pudo comprobar en el desarrollo del presente proyecto.

Por otra parte, es posible emplear herramientas gráficas para generar una escena virtual. En este trabajo se experimentó con tres de estas herramientas: Google Sketchup, 3D Studio Max y Blender. Google Sketchup marcó la diferencia al permitir la incorporación de código personalizado, a través de su API desarrollado en Ruby, es así que las pruebas con dicha herramienta demostraron que es la más adecuada para el proceso de levantamiento tridimensional de edificios de la UPS. A continuación listamos algunas de las características más importantes de Google Sketchup:

- Para su empleo requiere un corto periodo de aprendizaje.
- Posee características funcionales que permiten integrar, personalizar y automatizar procesos de renderizado.
- Permite la exportación a diferentes formatos 3D como: obj, dae, kmx, 3ds, dgw, dxf, fbx, vrml, etc.
- Importa escenarios 3D realizados en otras aplicaciones independientemente de la plataforma de desarrollo.
- Permite georefenciar los edicios y ambientes 3D generados dentro de la herramienta (integración directa con Google Earth).

Por las razones antes mencionadas Google Sketchup demostró ser el utilitario más adecuado para el desarrollo de aplicaciones 3D en cuanto se refiere a este proyecto de tesis.

Bibliografía

[1] A. Agarwal. Machine Learning for Image Based Motion Capture. *Institute national politechnique de grenoble*, 2006.

[2] Nebraska Art Whitton, desde Chester. Introducción a 3d. 2008.

[3] Varios autores. Applied integration of software, modeling, e interactive 3d. *Electric3D*.

[4] Varios autores. Ayuda de sketchup.

[5] Varios autores. Bs contact. *Btmanagement*.

[6] Varios autores. Chrome experiments-webgl.

[7] Varios autores. *A Free OpenGL Programming Book*.

[8] Varios autores. Getting started with the google sketchup sdk. *Google Code Labs*.

[9] Varios autores. *Insight3d - quick tutorial*.

[10] Varios autores. The open toolkit manual.

[11] Varios autores. Sweet home 3d.

[12] Varios autores. The virtual reality modeling language. *The VRML Consortium Incorporated*, 1997.

[13] Johannes Behr and Yvonne Jung. A scalable architecture for the html5/x3d integration model x3dom.

[14] Joe Crawford. Animacion 3d bajo software libre mundos virtuales. page 37, 2006.

[15] Universidad Politecnica de Madrid. Sistema de coordenadas y sistemas de referencia. page 26.

[16] Daniel González Losada Eduardo Antuña Díaz, Aitor Díaz Solares. Desarrollo en webgl.

[17] Jianer Chen David Victor Morris Stuart Tosten Tett Ergun Akleman, Vinod Srinivasan. Topmod3d. page 8.

[18] D. Vallejo Fernández, C. González-Morcillo, and L. Jiménez-Linares. Un sistema difuso para la optimización del renderizado.

[19] Cordelia Schmid Jean Ponce Fred Rothganger, Svetlana Lazebnik. 3D object modeling and recognition from photographs and image sequences. *Department of computer science and beckman institute university of Illinois at urbana-champaign, Urbana, IL 61801, USA*.

[20] Ph.D. Computer Science Fredrick H. Rothganger. 3D object modeling and recognition in photographs and video. *University of Illinois at Urbana-Champaign*, 2004.

[21] Joao Gaspar. *Google SketchUp 7 paso a paso en espanol*. 2010.

[22] Chris Marrin Gavin Bell, Rikk Carey. *The Virtual Reality Modeling Language Specification*, volume Version 2.0. 1996.

[23] Chris Marrin Gavin Bell, Rikk Carey. The virtual reality modeling language specification. Version 2.0, 2010.

[24] Jose Luis Cogollor Gomez. *Autocad Basico*. 2010.

[25] Chris Grover. *Google Sketchup The Missing Manual*, volume I. 2009.

[26] Jair Jonathan Monroy Cedillo Gustavo Nóñez Esquer. Mundos virtual 3d.

[27] Evan Hart. New OpenGL Features. *Creating 3D models in Ruby*, 2006.

[28] Yoshihiko Hasegawa Hitoshi Iba, Topon Kumar Paul. *Applied Genetic Programming and Machine Learning*. 2010.

[29] Javier Belanche Alejandro Conty Roberto Dominguez Jose Antonio Fernandez Luis Fernando Ruiz Guillermo Vaya Javier Alonso Albusac Jimenez, Fernando Arroba Rubio. *Tecnologias Libres para Sintesis de Imagen Digital Tridimensional*. 2006.

[30] Mercedes Martínez Cámara Eduardo Sanz Adán Félix Santamaría Peña Jacinto Blanco Fernández Jimenez Macias, Emilio Pérez de la parte. *Escenarios virtuales Web3D: Simulación con VRML, Java3D y X3D*, volume Version 2.0.

[31] Gustavo A. Osorio Juan D. Osorio, Flavio A. Prieto and J. I. R. Sola. José ignacio rojas sola. page 76, 2004.

[32] Javier Gómez Lahoz. *Reconstrucción 3d y realidad virtual en criminología.* 2010.

[33] Carranza Atho Fredy Florian Cruz Laura. Opengl y vtk. page 11, 2006.

[34] Ernest Pujol Leon. Art en 3 dimensions. page 76.

[35] Raquel P. Conde LÂŽopez. Mapeo facial de emociones sinteticas. 2005.

[36] D.-Q. Luis. Modelado 3d de escenarios virtuales realistas para simuladores quirórgicos aplicados a la funduplicatura de nissen. 1.

[37] Mc-Graw Hill Tom Mitchell. Machine learning. 1997.

[38] Tony Mullen. *Mastering Blender*. 2009.

[39] Kelly L. Murdock. *Google SketchUp and SketchUp Pro 7 Bible*. 2009.

[40] Ing.Luis Alberto Parra Pineros. Administracion de redes de computadores. 2010.

[41] Escet-Universidad rey Juan Carlos de Madrid. Introducción a opengl.

[42] Bonnie Roskes. *Google Sketchup Cookbook*. 2009.

[43] Claudia A. Lenti Rubén D. Morelli. Educación basada en competencias para el área de sistemas de representación. 2003.

[44] M. Scarpino. *Automatic sketchUp creating 3D models in Ruby*. 2010.

[45] Daniel K. Schneider and Sylvere Martin-Michiellot. Vrml primer and tutorial. 1998.

[46] Mike Trados. *Real World Google Sketchup 7*, volume 7.1. 2010.

[47] John Stewart Tuomas J. Lukka and others. Freewrl.

[48] Technische universitat wien. Graphics apis. 2008.

[49] Roberto Loaeza Valerio. Desarrollo de software para modelado y fabricación de objetos de madera usando un robot scara. page 31, 2009.

[50] "http://www.cristalab.com/blog/modelado-3d-fundamentos-basicos-c15531l/"Varios Autores. Modelado 3d, fundamentos basicos. 2005.

[51] Web3D Consortium Varios autores. X3d resources.

[52] John Wiley and Inc. Sons. *Google SketchUp for Site Design*. 2009.