UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

INGENIERÍA EN COMPUTACIÓN

"REALIDAD O PASEO VIRTUAL:
TEORÍA, ANÁLISIS, DISEÑO Y APLICACIÓN.
CASO PRÁCTICO."

Que para obtener el Título de Ingeniero en Computación

Presenta:

Erick Miranda Márquez

Director:

Ing. Rafael Flores García

Ciudad Universitaria, 2011

AGRADECIMIENTOS

A Dios, pues sin su bendición no hubiese tenido la fuerza suficiente para llegar a este momento.

A mi alma máter, que me abrigó y fue forjando durante todos estos años.

A mi familia, que siempre ha estado al pendiente de mis pasos para que alcanzara el final de un largo camino.

A mi asesor, el Ingeniero Rafael Flores García, por su esmerada y cuidadosa guía en la elaboración del presente trabajo, su experiencia me permitió ir modelando la investigación con mejor acierto.

A mis sinodales, por el tiempo dedicado a la lectura y corrección de mi tesis, siempre con el objetivo de aportar nuevos elementos a mi desarrollo como profesionista.

A mis profesores, que con su tiempo y dedicación inculcaron sus enseñanzas y sembraron en mí la semilla del conocimiento que ha empezado a germinar para dar frutos abundantes a la sociedad.

A mis amigos, que siempre se han hecho presentes para darme el apoyo necesario.

CONTENIDO

INTRODUCCIÓN

INTRODUCCIÓN

La realidad virtual ha tomado un auge impresionante en los últimos años; ya no sólo está enfocada en áreas particulares como la milicia o la ciencia, por el contrario, dicha herramienta ha ampliado su horizonte de posibilidades hacia diferentes industrias, cada vez más cercanas al público, como películas cinematográficas, educación, videojuegos, representaciones tridimensionales, incluso, recorridos virtuales.

En nuestros días existen diversos desarrollos que permiten elaborar simulaciones a nivel usuario (computadora, *'standalone'*) o niveles más grandes como la Simulación Distribuida Interactiva *('Distributed Interactive Simulation, DIS'*), cada uno con su complejidad respectiva. Así como tenemos una enorme variedad en desarrollos, también contamos con una diversidad de herramientas que nos permiten crear entornos virtuales.

Los usos de la realidad virtual son amplios, esto permite que no se le ubique en un área específica, sino que sea una herramienta común en los avances tecnológicos, desde lo más elementales hasta los más complejos, involucrando múltiples campos de especialización como: ingeniería, arquitectura, medicina, biología, historia, diseño gráfico e industrial, sólo por mencionar algunos.

Todo esto la convierte en un campo muy consolidado, pero bajo un panorama experimental; aunque esta aseveración puede parecer contradictoria, a lo largo de este trabajo iremos aclarando algunos puntos al respecto.

En este momento en América Latina empiezan los desarrollos de realidad virtual, pero se consideran fuera del alcance y la mayoría de las veces son presentados a grandes compañías debido al poco auge que se le ha dado a esta disciplina en la región.

En Europa es común ver este tipo de proyectos en lugares como museos, centros comerciales, zonas arqueológicas e incluso templos religiosos. Por mencionar algunos casos existe el espacio *Gaudí de la Pedrera* que tiene una reproducción en línea exacta del edificio original; la catedral de Tuy y la iglesia de Saint-Laurentius,[1] o el caso del Kilmartin House Museum junto con la Universidad de Warwick (Escocia) que han ideado un sistema de reconstrucción digital de monumentos antiguos y lugares arqueológicos.[2]

El desarrollo de realidad virtual que ofrecemos se desenvuelve en dos vertientes, la comercial y la histórico-religiosa. Así encontramos, en un solo producto, dos formas de comercialización: la primera nos permite encaminar nuestro desarrollo hacia sectores públicos o privados con fines informativos; la segunda se orienta hacia la rama científica, al proponer y reconstruir de manera fiel un conjunto arquitectónico, ya sea para fines históricos o inventaríales, el cual estaría encaminado a resolver uno de tantos problemas que existen actualmente en México dentro de los inmuebles religiosos como es el robo de arte sacro:

> *La falta de un inventario de arte sacro, el desinterés del gobierno no sólo para proteger su patrimonio, sino para enlistarlo y evitar su deterioro, porque el acervo es propiedad del Estado, en custodia de la Iglesia…* [3]

---

[1] http://www.ename974.org/Eng/pagina/kerk_tijdsvenster2.html (ví: 5 de junio de 2011)

[2] http://www.kilmartin.org/ (ví: 5 de junio de 2011)

[3] José de Jesús Aguilar Valdés, director del departamento de Arte Sacro del Episcopado Mexicano. Toluca, Méx., a 15 de Agosto de 2007. http://www.diocesistoluca.org.mx/noticias/index.php?option=com_content&task=view&id=1382&Itemid=2 (ví: 10 de Septiembre de 2009)

Al mismo tiempo, pretendemos brindar una herramienta eficiente y rápida para catalogar los objetos que están en custodia de la Iglesia.

Debemos considerar que el producto no pretende ser una copia de versiones ya existentes e incluso no se cuenta con el equipo humano y tecnológico para desarrollarlo, utilizaremos herramientas que no precisamente están enfocadas al desarrollo de realidad virtual, pero que son útiles y no requieren un alto grado de estudio para poder manejarlas. De esta manera nos proponemos crear un producto destinado a cubrir las necesidades que se consideran importantes en el marco ya establecido, las cuales son: crear un producto con fines de venta en el rubro empresarial o, en su defecto, realizar un estudio enfocado a crear una reconstrucción de un conjunto arquitectónico con fines histórico-culturales.

Para lograr nuestro cometido, planteamos la creación de un recorrido virtual a través del templo de Lourdes de la Ciudad de México, localizado en el Centro Histórico, así como la recreación del entorno tanto de los alrededores como del mismo templo lo más fielmente posible, de igual manera produciremos modelos 3D de objetos y artefactos existentes en el interior del inmueble.

Una de las ventajas de las reconstrucciones virtuales es que colocan a la persona dentro del contexto en que el desarrollo es recreado. Así, las representaciones se convierten en material histórico que brinda información de acuerdo a su carácter de fidelidad y compromiso por ser lo más apegado a la realidad.

El objetivo es plantear una solución para los clientes del Club de Banqueros[4] (asociación encargada de realizar banquetes y eventos), quienes no tienen la posibilidad de visitar personalmente las instalaciones del templo, así como entregar una aplicación con una serie de funciones adicionales como el tomar notas, personalizar el decorado del mismo, guardar imágenes del escenario, por mencionar algunas, para lo cual se ideó la forma de entregar una aplicación que

[4] Club de Banqueros, http://www.clubbanqueros.com.mx/ (ví: 21 de Octubre de 2011)

permitiera recrear el ambiente por medio de un recorrido virtual.

El presente trabajo fortalece además las bases técnicas para romper con el mito de que esta tecnología (la realidad virtual) es inalcanzable para países en desarrollo; fomentar e impulsar nuestros propios proyectos de realidad virtual y demostrar que desarrollos como el que se muestra a continuación son viables y necesarios para nuestra nación. De igual manera, acercar dicha tecnología a campos poco recurrentes y motivar que la realidad virtual no sólo pretende ser un campo aplicable a la acción científica, sino evidenciar que también puede ser una herramienta para uso común en sectores específicos de la sociedad.

Esta forma de utilizar la realidad virtual es vital para arrojar una luz nueva en el rubro arquitectónico, histórico e incluso religioso (en el aspecto de acercar a las dependencias religiosas a una nueva forma de inventariar y recrear sus construcciones) y sobre todo, en el campo de la ingeniería en computación, mostrando que hoy en día la ingeniería no está enfocada únicamente en aspectos técnicos, sino que también amplía su espectro de conocimiento para incluirse en desarrollos enfocados a la preservación de monumentos y así resolver carencias que actualmente se tienen en nuestro país, por ejemplo el robo de arte sacro.

Erick Miranda Márquez
Ciudad de México, 2011

Capítulo

1

Visión General

VISIÓN GENERAL

1.1 REALIDAD VIRTUAL

El concepto de Realidad Virtual ya no está alejado de nuestra vida cotidiana, incluso lo usamos con frecuencia aunque no tengamos una idea clara de su significado. Pero ¿podemos definir el concepto de *realidad virtual* (RV, o comúnmente encontrada como VR *'virtual reality')*? La respuesta no es sencilla, pues al interpretar los distintos elementos que conforman el término nos encontramos con diferentes significados de acuerdo al área en la que pretendemos encontrar su definición. Encontramos un sentido casi por cada investigación hecha al respecto, esto se debe a que la evolución de la realidad virtual ha sido tan rápida que no ha permitido establecer parámetros concretos y, aún en estos momentos, está en constante cambio debido a los avances tecnológicos, por tal razón es seguro decir que la realidad virtual va de la mano con los avances en la computación y la tecnología.

Considerando lo anterior no es difícil entender por qué la realidad virtual y su definición es relativa para cada área de estudio, inclusive para cada persona que haya tenido contacto con ella. Con base en esto, más que optar por un solo significado, enunciaremos un conjunto de definiciones, donde cada una entrega una idea de lo que se entiende por realidad virtual. (Figura 1.1)[5]

[5]Juan Carlos Parra Márquez, et. al., *Introducción Práctica a La Realidad Virtual*. Concepción: Ediciones U. Bío-Bío, 2001. p. 3

Definiciones de Realidad Virtual

Experiencia de telepresencia, donde telepresencia es la sensación de presencia utilizando un medio de comunicación.	Es una manera mediante la cual, manipulamos e interactuamos con computadoras y datos extremadamente complejos.
Es un paso más allá de lo que sería la simulación por computadora, tratándose más bien de una simulación interactiva, dinámica y en tiempo real de un sistema.	Consiste en simulaciones tridimensionales interactivas que reproducen ambientes y situaciones reales.
Entorno de tres dimensiones sintetizado por computadora, en el que participantes acoplados de forma adecuada pueden manipular elementos físicos simulados en el entorno y, de alguna manera, relacionarse con las representaciones de otras personas pasadas, presentes o ficticias, o con criaturas inventadas.	Ambiente altamente interactivo donde el usuario participa a través del uso de una computadora en un mundo virtualmente real. Es una simulación tridimensional por computadora durante la cual el usuario resulta inmerso tan completamente que esta realidad, de origen artificial, aparenta ser real.
Es una simulación por computadora, dinámica y tridimensional, con alto contenido gráfico, acústico y táctil, orientada a la visualización de situaciones y variables complejas, durante la cual el usuario ingresa, a través del uso de sofisticados dispositivos de entrada, a mundos que aparentan ser reales, resultando inmerso en ambientes altamente participativos, de origen artificial. Una nueva y sorprendente forma de navegar información.	Simulación tridimensional interactiva por computadora en la que el usuario se siente introducido en un ambiente artificial, y que lo percibe como real, basado en estímulos a los órganos sensoriales.

Es *el medio que proporciona una visualización participativa en tres dimensiones y la simulación de mundos virtuales, siendo dichos mundos el elemento fundamental de un sistema de realidad virtual. La realidad virtual es un entorno generado por computadora en el que los participantes pueden entrar físicamente e interactuar con él, desplazándose por su interior o modificándolo de cualquier manera. En su forma más simple, un mundo virtual podría estar compuesto por un edificio tridimensional por el que podríamos desplazarnos, aunque sin modificar nada. Sin embargo, con el equipamiento adecuado, los usuarios podrían ver, desplazarse e interactuar a través de estos entornos gráficos generados por computadora.*

Realidad Virtual *o mundo virtual es una base de datos gráficos interactivos, explorables y visualizables en tiempo real en forma de imágenes tridimensionales de síntesis capaces de provocar una sensación de inmersión en la imagen. En sus formas más complejas, el entorno virtual es un verdadero "espacio de síntesis", en el que uno tiene la sensación de moverse "físicamente". Esta sensación de "movimiento físico" puede conseguirse de diferentes formas, la más frecuente consiste en la combinación de dos estímulos sensoriales, uno basado en una visión estereoscópica total y el otro en una sensación de correlación muscular, llamada "propioceptiva", entre los movimientos reales del cuerpo y las modificaciones aparentes del espacio artificial en que está inmerso.*

Figura 1.1

Definiciones de Realidad Virtual

Pongamos especial atención en las dos últimas definiciones, pues además de ser las más completas hacen hincapié sobre la generación de mundos en tres dimensiones, la participación del individuo en el entorno gráfico y la sensación de pertenencia en el ambiente, aspectos importantes al momento de identificar desarrollos de Realidad Virtual.

Como hemos leído, la definición de Realidad Virtual se ha estado modificando, nombrándola también como "Realidad Sintética", "Mundos Virtuales o Ficticios". Recientemente en los círculos científicos se ha adoptado el término **"Ambientes Virtuales"** (*'Virtual Environments'*) y más particularmente el término **"Presencia"**, que se define como la experiencia subjetiva de estar en un lugar o ambiente, cuando se está físicamente situado en otro. Aunque el término de presencia simplifica el concepto de realidad virtual no esclarece completamente un significado de lo que es la realidad virtual, lo importante es que se empiezan a crear distinciones que nos permitirán, más adelante, clasificarla de manera más objetiva de acuerdo a su nivel de desarrollo.[6]

LA REALIDAD VIRTUAL ES UN ENTORNO RECREADO POR COMPUTADORA, QUE PERMITE AL USUARIO MOVERSE, EXPLORAR, INTERACTUAR Y SIMULAR LA SENSACIÓN DE PERTENENCIA EN TIEMPO REAL A LO LARGO DE UN AMBIENTE TRIDIMENSIONAL, A TRAVÉS DE CANALES SENSORIALES.

Como ya hemos visto a medida que avanza la tecnología, la realidad virtual adopta nuevos 'aires' y es muy probable que a medida que siga avanzando, también su significado vaya adquiriendo nuevas definiciones, por lo que es necesario que el lector a partir de este punto tenga una idea del uso frecuente de este término en nuestros días.

[6]Ibid., p. 4

Normalmente la RV en sus casos 'clásicos' requiere procesos de gráficos en tiempo real, una pantalla estereoscópica, la cual es necesaria para producir la ilusión 3D, y un sistema de 'seguimiento' que se encargará de obtener las señales: ya sea de la cabeza (como puede ser un casco de RV) o señales que provengan del movimiento de las manos (como el caso de los guantes de RV). Estas tecnologías son conocidas como *'head-mounted displays'* (HMD) y se componen de visores o lentes estereoscópicos que, unidos a un *'data globe'* (guante de datos que es usado para rastrear el movimiento de la mano), permiten de manera completa la inmersión o presencia del usuario en ambiente virtual.[7]

Cabe mencionar que existe hardware más complejo como:

- Rastreadores de movimiento de cabeza:
  (Head trackers).
- Rastreadores de movimiento:
  (Motion trackers).
- Domos de RV:
  (VR Domes).
- Simuladores de RV:
  (VR Simulators).

Los elementos anteriores de hardware para RV son complejos y la mayoría de las veces son dispositivos especializados que no comúnmente pueden estar al alcance de los usuarios habituales. Sin embargo, existe hardware mas cercano a nuestro usuario final y combinándolos de acuerdo a las necesidades del cliente y la naturaleza del recorrido, permiten una inmersión completa del usuario en diferentes niveles.

A continuación se muestran elementos comunes para desarrollos de RV.

---

[7]Mario A. Gutiérrez A., et. al.. *Stepping into Virtual Reality*. London: Springer, 2008. p. 1

Hardware utilizado en RV

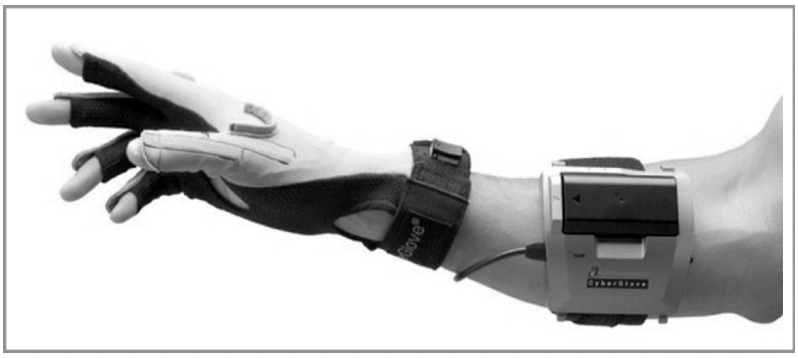


Figura 1.2.a

Figura 1.2.b

Figura 1.2.c

Figura 1.2.a - El *CyberGlobe II,* es un guante de datos inalámbrico que transforma el movimiento de la mano y de los dedos en datos digitales.

Figura 1.2.b - El *5DT HMD 800 Series,* ofrece al usuario una salida de imagen en alta resolución y sonido ambiental.

Figura 1.2.c - El*VisBox-X2,* es una pantalla gigante para inmersión en alta resolución, pudiendo ofrecer en algunos modelos sonido ambiental.

Figura 1.2
Hardware utilizado en RV

1.2 CONCEPTOS GENERALES

1.2.1 RECONOCIENDO LA REALIDAD VIRTUAL

Una de las preguntas que salta a la mente es ¿cómo distinguir un desarrollo de Realidad Virtual con respecto a otros productos como paseos virtuales o realidad aumentada?, también es una pregunta difícil de responder, pero se han establecido marcos que nos pueden dar una respuesta aceptable.

Como se ha mencionado, hay muchas maneras para definir la RV, y por la misma razón encontrar un parámetro que distinga un desarrollo RV también puede resultar complejo, existen varias condiciones que nos pueden ayudar a diferenciar dichos proyectos. Enumeraremos algunas condiciones destacables:[8]

SIMULACIÓN:

Capacidad de representar un sistema. Esta representación debe contener aspectos suficientes de la realidad para convencer al usuario que se encuentra en una situación paralela. Pero hay que tener en consideración que dicha representación puede o no contener modelos físicos de la realidad.

INTERACCIÓN:

Tener control del sistema creado. El hecho de tener el control del sistema hace referencia a que un desarrollo pasa de ser una película o un recorrido previamente programado a un ambiente en el que uno puede desenvolverse de forma autónoma. Para lograr esta interacción existen diversas técnicas e interfaces hombre-máquina, que van desde el teclado, el mouse y la pantalla de una

[8]Márquez Parra, op. cit., p.6

computadora hasta guantes, visores o trajes sensoriales. La interacción con el mundo virtual presupone que al menos el usuario puede trasladarse a voluntad dentro del ambiente virtual y mover objetos, dichas acciones tienen como objetivo producir cambios en ese entorno artificial.

PERCEPCIÓN:

La percepción es un aspecto sumamente importante, ya que la mayoría de los sistemas de RV, para recrear este factor, concentran sus esfuerzos en otorgar sensaciones lo más reales posibles por medio de percepciones sensoriales (visuales, táctiles o auditivas) y por medio de los elementos externos mencionados con anterioridad. Aunque ya existen desarrollos en los que se intenta llegar directamente al cerebro, evitando así las interfaces sensoriales externas, hay modelos más simples en los que recurren a toda la fuerza imaginativa del hombre para que pueda experimentar una realidad parcial. Todo esto con la finalidad de producir en el usuario una 'inmersión' vivencial en un ambiente digital.

Los aspectos anteriores son los principales puntos que debemos considerar al identificar un sistema RV, existen más elementos que no se mencionan pero deben tomarse en cuenta como es la profundidad, aunque esto es obvio ya que en un sistema tridimensional los objetos del mundo virtual deben tener un formato que indique su profundidad y dimensiones.

LOS PRIMEROS TRES ASPECTOS (SIMULACIÓN. INTERACCIÓN Y PERCEPCIÓN) SON IMPRESCINDIBLES EN CUALQUIER SISTEMA RV. ALGUNOS INVESTIGADORES PLANTEAN TRES BASES DE LA REALIDAD VIRTUAL QUE SON LA *INTERACCIÓN, INMERSIÓN E IMAGINACIÓN,* CONOCIDAS TAMBIÉN COMO LAS "3I" (FIGURA 1.3).

LAS TRES BASES DE LA REALIDAD VIRTUAL

Los cinco componentes clásicos de un sistema RV.
Inspirado para esta investigación en Grigore C. Burdea, Philippe Coiffet.
Virtual Reality Technology. Second ed: John Wiley & Sons, Inc., 2003.

Figura 1.3
Las Tres Bases de la Realidad Virtual

1.2.2 FACTORES IMPORTANTES DE LA REALIDAD VIRTUAL

La meta de la Realidad Virtual es crear en el usuario la ilusión de estar en un ambiente que puede ser percibido como real, con la suficiente interactividad para ejecutar tareas específicas de una manera fácil y agradable para quien se encuentra inmerso en el entorno tridimensional.

Retomando algunos conceptos, hay dos factores que pueden describir la experiencia RV desde el aspecto físico y psicológico: *inmersión y presencia.*[9]

INMERSIÓN:

Está relacionado con la configuración física de la interface de usuario de la aplicación RV. Los sistemas RV también pueden ser clasificados como *completamente inmersivo* (aplicable a los HMD), *semi-inmersivo* (aplicable a las pantallas gigantes para inmersión) o *no-inmersivo* (aplicable a las aplicaciones de escritorio basadas en Realidad Virtual). Esta clasificación depende mucho de cómo el usuario puede percibir (ver, escuchar, 'tocar') el mundo durante la simulación. Los sistemas modernos de Realidad Virtual completamente-inmersiva tienen diferentes niveles de inmersión, creando diversos tipos de ilusiones alternativas de la realidad y varios de ellos de uso complejo.

Los sistemas semi-inmersivos utilizan sonido 3D, gráficas en alta resolución y normalmente son usados en sistemas donde más de un usuario comparte el mismo ambiente o simulación, esto ha abierto enormes posibilidades en ambientes de trabajo cooperativo.

Los sistemas no-inmersivos, debido a que son menos costosos, son de fácil instalación y resultan más amigables al usuario final, han venido ganando popularidad. En ocasiones conocidos como *'desktop-based VR system'* que no son más que computadoras basadas en tecnologías o aplicaciones RV; uno de los casos más representativos son los videojuegos, pero no refiriéndonos exclusivamente al puro videojuego, también necesitamos el equipo que lo pueda reproducir, es decir, los sistemas de ejecución o virtualización, en este caso serían las consolas de videojuegos o inclusive algunas computadoras con el suficiente hardware para su ejecución.

[9]Mario A. Gutiérrez, op. cit., p. 2

Los videojuegos son el mejor ejemplo de la combinación de interactividad, fácil manejo, imágenes con un buen nivel de detalle (alta resolución o *'high definition', HD*), un sonido estéreo y en algunos casos sonido 3D, todo lo anterior produce en el jugador un alto interés y envolvimiento en la simulación. Hablando un poco de la parte psicológica, pocos desarrollos de Realidad Virtual pueden competir con un buen videojuego a un nivel de aislamiento psicológico, es decir, apartar al usuario del mundo que lo rodea (ambiente) y producir una respuesta emocional inmersiva casi completa del participante en el desarrollo (videojuego).

Los aspectos psicológicos de la experiencia RV aún son un área de investigación, no se conocen con claridad los factores en la simulación que pueden producir reacciones específicas en términos de respuestas emocionales, qué es lo que 'hipnotiza' y le produce al usuario un grado de interés. Se considera la idea de generalizar áreas de interés, pero eso no termina de convencer completamente. Pero uno de los conceptos importantes que nos puede ayudar a entender la psicología que hay de tras de la experiencia RV es la *'sensación de presencia'*.

PRESENCIA:

Aunque ya mencionamos el término, es necesario comprender un poco más la definición de presencia. La presencia es un término subjetivo, asociado con la psicología del usuario, de ahí deriva la necesidad de comprender más este aspecto de la Realidad Virtual. Según Slater y Wilbur, la *"Presencia es un estado de conciencia en donde existe la sensación (psicológica) de pertenecer al ambiente virtual".*[10]

Presencia es cuando simulaciones multimodales (imágenes, sonidos, retroalimentación, etc.) son procesadas por el cerebro y entendidas como un entorno coherente donde nosotros podemos interactuar y ejecutar algunas

[10]Slater M., Wilbur S. *A Framework for Immersive Virtual Environments (Five): Speculations on the Role of Presence in Virtual Environments*, 1997.

actividades. Esta presencia se consigue cuando el usuario es consciente, deliberadamente o no, de pertenecer a un ambiente virtual (virtual environment, VE). Por ejemplo, retomando el tema de los videojuegos, una persona sabe que el mundo en el videojuego no es real, pero decide situarse como si viviera una situación real. Una señal de presencia es cuando una persona se comporta en un ambiente virtual (VE) de una manera similar a la que se comportaría en una situación parecida en la vida real.

La presencia, en primer lugar, puede involucrar reacciones emocionales del usuario. Una vez que el cerebro integra las imágenes 3D, los sonidos y otros tipos de retroalimentación en la forma de un ambiente coherente, diferentes reacciones pueden surgir, esto ocurre de tal manera que en verdad nos sintamos profundamente involucrados en la experiencia de la virtualización.[11]

El término de presencia se extiende aún más, pero con lo anterior nos podemos confirmar que este punto es sumamente importante para nuestros proyectos RV.

LA PRESENCIA ES UN FACTOR IMPORTANTE PARA NUESTROS DESARROLLOS DE REALIDAD VIRTUAL, YA QUE NOS DARÁ LA PAUTA PARA ENTENDER EL PERFIL DE NUESTRO CLIENTE Y LOGRAR QUE NUESTROS PRODUCTOS TENGAN EL EFECTO QUE NOSOTROS DESEAMOS.

1.2.3 ¿REALIDAD ARTIFICIAL?

¿Y la realidad artificial?, ¿no es lo mismo? De primera impresión podría parecer que sí, y esto nos puede conducir a conceptos erróneos, al menos para el

[11]Gutiérrez A, Mario A, op. cit., pp. 3 - 4

tema que estamos tratando. No vamos hacer un estudio exhaustivo sobre el tema de realidad artificial, pero sí trataremos de esclarecer un poco qué hay detrás de éste concepto.

Cabe aclarar que, mientras más avanzamos más términos escuchamos y esos términos cada vez se van pareciendo entre sí, por lo que hay que cuidar con suma delicadeza el evitar confusiones. La realidad artificial tuvo sus principios por los años noventa con Myron W. Krueger con su libro "Artificial Reality 2", donde acuñó dicho término como la experiencia inmersiva en un ambiente interactivo. Claro, un par de palabras no pueden definir todo un concepto como el de realidad artificial pero nos pueden ayudar un poco.[12]

Básicamente la definición de Myron engloba en pocas palabras lo que es la realidad artificial, si nos damos cuenta, menciona la experiencia inmersiva, también hablada en la Realidad Virtual y un ambiente interactivo, donde la interacción en la Realidad Virtual es fundamental, entonces ¿cuál es la diferencia?

Si empezamos a poner un poco de más atención, en la Realidad Virtual la inmersión tiene diferentes niveles y depende del tipo de desarrollo, ya sea inmersivo o no-inmersivo, básicamente la inmersión se produce de acuerdo a la predisposición del usuario, en la Realidad Virtual por más total que sea la inmersión, siempre habrá algún medio tratando de hacer lo más real posible esa inmersión. En cambio, la inmersión en la realidad artificial se da de forma automática, porque ya está ahí, es parte de nosotros. Por ejemplo, se puede crear una RV en donde una ama de casa aprenda a moverse en una cocina del futuro, suponiendo que hacemos una RV completamente inmersiva, en donde tengamos una habitación del mismo tamaño que la cocina del futuro, le coloquemos un rastreador de movimiento de cabeza (Head trackers, HMD), tal vez para el movimiento del cuerpo un rastreador de movimiento, un guante de datos para el

[12] Myron W. Krueger, "Artificial Reality", p.6

movimiento de la mano y aditamentos extras que realmente hagan sentir a la ama de casa en un ambiente verdadero, pero lo anterior no deja de ser eso, una **recreación** de una realidad. La mujer aprenderá a controlar diversas cosas, como encender la estufa con la mano a distancia mientras el horno indica que la comida ya está lista, todo lo anterior mientras con un movimiento de su otra mano abre el refrigerador. Todo es meramente una simulación. En cambio, en la realidad artificial todo eso será real, probablemente el ama de casa con el movimiento de sus manos a distancia controle varias cosas de su cocina mientras habla con el horno preguntando cuánto tiempo falta para la cocción completa del platillo.

Con lo anterior entendemos la parte de ambiente interactivo, en la Realidad Virtual hablamos de un ambiente (no real) interactivo, mientras que en la realidad artificial es un ambiente real e interactivo. Además de que en esa realidad artificial no existe ningún aditamento asociado en alguna parte del cuerpo para hacer inmersiva la realidad, porque esa realidad es la realidad cotidiana.

Actualmente se están dando enormes frutos en este tipo realidades, aunque los primeros desarrollos no están comprometidos a nivel cotidiano, sino más bien al área de videojuegos, se espera que pronto también se abra ese campo a ambientes más hogareños.

Como ejemplo de estos primeros pasos en realidad artificial tenemos el proyecto NATAL que ahora lo conocemos como 'Kinect' de Microsoft[13] para la Xbox 360, donde lo que alguna vez habíamos escuchado acerca de que no necesitaríamos controles para jugar videojuegos, ni nuevas consolas, comienza a ser algo tangible, al menos en el aspecto de decir "adiós" a los controles manuales.

Expliquemos un poco cómo es que nuestra realidad se empieza a convertir

[13] http://www.xbox.com:80/es-ES/Kinect (ví: 6 de junio de 2011)

en artificial. El sistema 'Kinect' se basa en una barra que incorpora una cámara y un micrófono que se encargan de recoger todos los datos necesarios para la interacción con la consola (en este caso el *Xbox*) como son la voz y los movimientos, ya todo ello sin algún hardware pegado al cuerpo del usuario.

Los aditamentos están compuestos por una cámara (la que se encarga del reconocimiento de rostros), un sensor de profundidad, micrófono multidireccional y software elaborado por el equipo de Microsoft. A continuación menciono algunas características:

SENSOR DE MOVIMIENTO:

Kinect utiliza un sensor de movimiento que controla todo el cuerpo. Así que cuando jugamos, no solo se detecta el movimiento de manos y muñecas, sino el de todo el cuerpo: brazos, piernas, rodillas, cintura, cadera, etcétera.

SEGUIMIENTO ESQUEMÁTICO:

Mientras jugamos, Kinect crea un esquema digital de nuestro cuerpo basándose en datos de profundidad. De esta forma, cuando el usuario se mueva hacia la derecha, izquierda o hacia arriba, el sensor capturará el movimiento y lo trasladará al juego.

RECONOCIMIENTO FACIAL:

Kinect ID recuerda al usuario recopilando datos físicos que se almacenan en el perfil de cada jugador. Así que cuando éste quiere volver a jugar, Kinect sabrá quién es y podrá empezar a jugar fácilmente en cualquier momento.[14]

Existen otras propuestas las de Sony con su PlayStation Move basado en un control sencillo y fácil de usar que permite la captura de una amplia gama de

[14]http://www.xbox.com:80/es-ES/Kinect/GetStarted (ví: 6 de junio de 2011)

movimientos, lo que brinda un nivel de control en el juego.[15] Así como el sistema Wii de Nintendo el cual también intenta acercarse a esta realidad con su variedad de mandos inalámbricos y sensibles al movimiento.[16] Pero hay que destacar que aun están un paso atrás ya que el sistema 'Kinect' permite toda esta interacción sin ningún control pegado al cuerpo.

El reconocimiento de movimientos, de imágenes e incluso el del tono de voz ya es una realidad, pasos que nos acercan más a la realidad artificial.

La realidad artificial, a diferencia de la Realidad Virtual, está más cerca de nosotros y se basa en elementos comunes que ya existen en nuestro entorno para que nuestra interacción sea natural. Pero no nos quedemos con esta idea, en un par de años esta tecnología ha dado grandes pasos. Hay que tener en cuenta que gracias a la alta definición en video, el HD 3D hologramizado es una realidad también.[17]

1.2.4 ARQUITECTURA DE LOS SISTEMAS DE REALIDAD VIRTUAL

Es importante tener una idea de los componentes de un sistema RV, esto nos permitirá tener una mayor visión de qué es lo que necesitamos, de qué medios nos podemos valer y cómo se desenvuelve el sistema.

Para empezar necesitaremos elementos de hardware que permitan la adquisición de datos. Por regla general, siempre necesitaremos de algo que permita la adquisición de datos, sin ellos la interacción no sucedería. A través de los dispositivos de "entrada – salida" el usuario se puede comunicar con el entorno de nuestro desarrollo. Por medio de los elementos de entrada (input) el usuario

[15]http://mx.playstation.com/ps3/playstation-move/index.htm (ví: 6 de junio de 2011)
[16]http://www.nintendo.es/NOE/es_ES/wii_54.html (ví: 6 de junio de 2011)
[17] http://www.holomex.com.mx/esp/holograma s/3d.html (ví: 6 de junio de 2011)

trasmite sus órdenes al sistema de RV, indicándole que desea desplazarse, cambiar el punto de vista actual o interactuar con algún objeto del mundo virtual. Mientras que los dispositivos de salida cumplirán su función de crear el ambiente inmersivo lo más real posible en el usuario.

En sistemas completamente inmersivos, los elementos de adquisición de datos son muy complejos, por tal razón también los elementos de hardware resultan ser robustos. Para esos sistemas hay que disponer de rastreadores de posición los cuales enviarían la información al sistema de la posición del sujeto, hacia dónde está mirando y la ubicación relativa dentro del mundo virtual. Al mismo tiempo el sistema estaría recibiendo información de un guante de datos y procesando esa información para mostrar la representación de una mano dentro del ambiente e interactuando con el mismo.

En nuestro caso nos enfocaremos en los elementos de sistemas RV no-inmersivos, ya que nos serán más útiles para el tipo de sistema que desarrollaremos. En los sistemas RV no inmersivos, estos dispositivos de entrada se reducen a elementos comunes como puede ser el mouse, el teclado, inclusive algunos elementos de mando como pueden ser las palancas de mando o volantes.

Como observamos en la figura 1.4, contamos con cinco elementos básicos en arquitecturas RV. Los dos elementos dentro del cuadro rayado son los que conforman en sí un sistema RV, estamos hablando de un motor RV (software) y los elementos de hardware que se encargarán de la obtención y transmisión de datos. Los demás elementos no pertenecen directamente al sistema RV, pero son necesarios dentro de un ambiente RV.

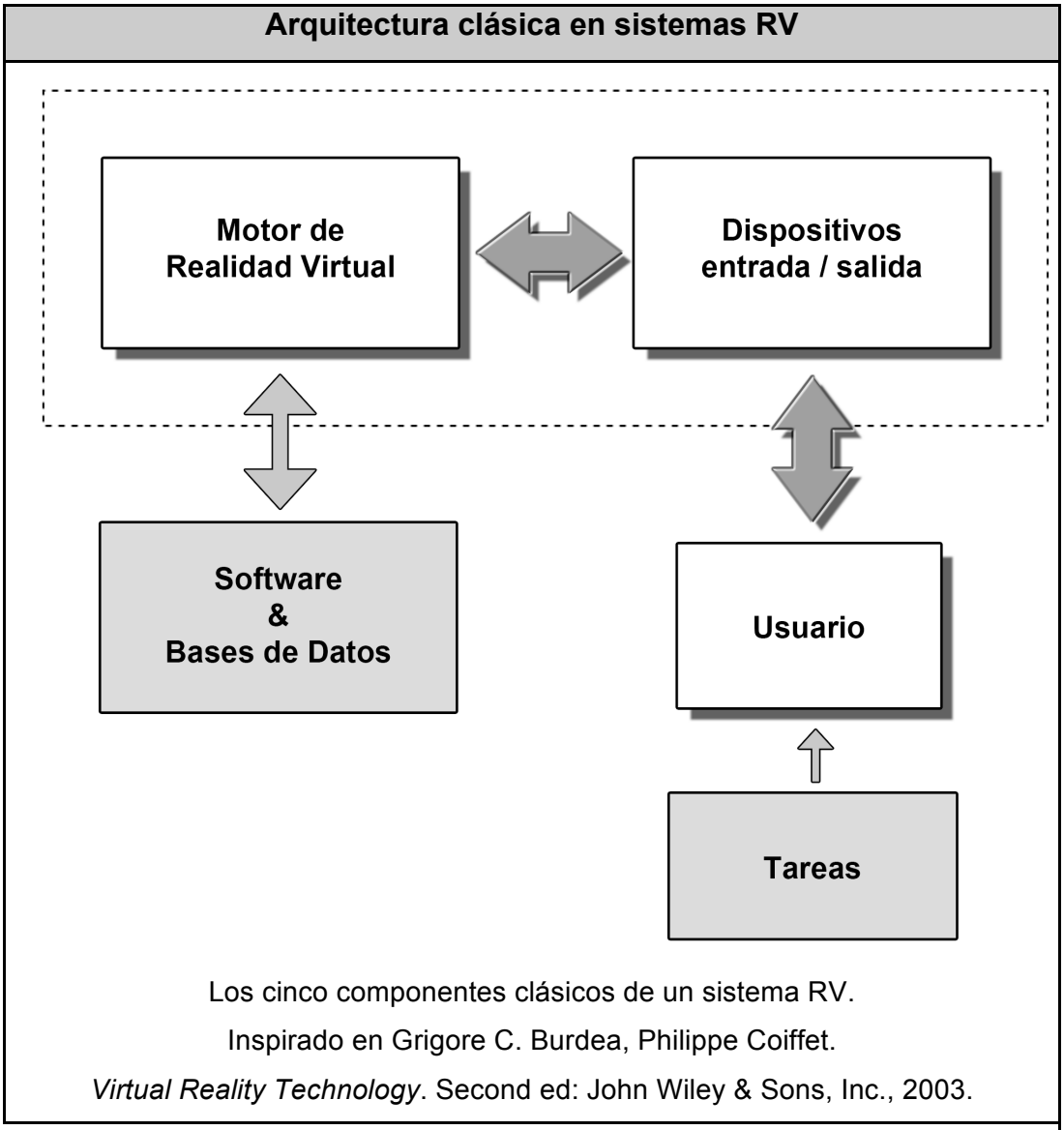


Figura 1.4

Arquitectura Clásica en Sistemas RV

Ya hablamos de los **dispositivos de entrada** y realmente no hay mucho que indagar en ellos, son elementos comunes para nosotros (refiriéndonos a sistemas no-inmersivos) y que no requieren un estudio mayor.

Continuando con los demás elementos de una arquitectura RV, tenemos un **motor de Realidad Virtua**l (*3D VR Engine*), que es una herramienta de programación orientada a objetos para desarrollos de Realidad Virtual. Estos

motores cuentan con características especiales como niveles de detalle para imagen (*'Level of Detail'*, LOD), simulaciones visuales, aplicación de texturas, programación para interactuar con otras plataformas de programación, descripción del lenguaje del motor, algoritmos gráficos, y un sin fin de características esenciales para desarrollos RV.

Podemos encontrar software para programar desarrollos de Realidad Virtual, pero su programación y sus prestaciones muchas veces no son las deseadas, además que en la mayoría de las ocasiones la compra del software no significa que también nos otorguen acceso al código fuente para manipularlo a nuestras necesidades, ni tampoco tenemos el tiempo para crear nuestro propio motor de Realidad Virtual.

Existen lenguajes de programación como es el VRML (Virtual Reality Modeling Language) que representa un formato de representación vectorial en 3D (tres dimensiones), diseñado particularmente para la web. En estos días Java se ha convertido en una plataforma de desarrollo muy demandada de manera que también contamos con Java3D, que junto con VRML, se ha convertido en una herramienta común para la programación de mundos virtuales.

Para compensar las tecnologías que tenemos para crear entornos virtuales nos enfocaremos en otro tipo de motores, que no específicamente son para desarrollos RV pero que básicamente cubren nuestra necesidad de crear un proyecto RV no-inmersivo, son los motores para videojuegos, conocidos como *Game Engine.* Muchos desarrollos han obtenido experiencia de la realidad virtual para la programación de videojuegos. Ya no es necesario tener paquetes de desarrollo profesional RV, ya que los motores de videojuegos nos ofrecen casi las mismas prestaciones de desarrollo que los lenguajes de programación RV.

Un **motor de videojuego** (*Game Engine*), es un software diseñado para la creación y desarrollo de videojuegos. Tenemos una gran variedad de motores

para videojuegos y su estructura depende del tipo de plataforma en la que se quiere programar, ya sea para consolas o equipos de cómputo, donde un factor importante es saber el sistema operativo en donde se realizará la instalación, por ejemplo sistemas como Mac OS, Windows o Linux. El Core de estos motores incluye funciones típicas como: procesamiento gráfico para 2D o 3D, física de juego, detección de colisiones tanto para impacto como para respuesta, sonido, lenguaje de programación, animación, inteligencia artificial, procesamiento para redes, manejo de memoria por mencionar algunas características. Trabajar con este tipo de software resulta, por lo regular, muy barato, además de utilizar modularidad para la programación, esta cualidad nos ayuda para la creación de diferentes proyectos. En el capítulo 2 (Herramientas de Desarrollo), abordaremos con mayor detalle los motores gráficos 3D.

Continuando con la explicación de los elementos que engloban un sistema RV, hablemos ahora del **software**. Si observamos bien, este elemento ya no se considera dentro del sistema RV (Figura 1.4), pero sí forma parte de la arquitectura, esta aclaración es necesaria para saber distinguir entre los elementos directamente relacionados con el sistema RV y los elementos que sirven de apoyo para los sistemas RV. Los demás elementos que mencionaremos se pueden considerar como elementos de apoyo para los sistemas RV, pero no significa que se consideren fuera de la arquitectura RV clásica.

El software está directamente relacionado con el hardware donde se planea instalar el sistema RV, ya que el software debe entablar de alguna manera una comunicación con el motor RV, porque el motor RV se encarga exclusivamente de los ambientes y simulaciones gráficas, así como de la recolección de datos, pero necesita una plataforma o un soporte donde el motor RV pueda ejercer una interacción entre el usuario y la computadora, consola o dispositivo donde esta instalado.

De la misma forma sucede con la parte de **bases de datos**, muchos sistemas que son independientes o conocidos como '*standalone'*, necesitan comunicarse a alguna base de datos que contenga información relacionada ya sea al mismo entorno virtual o alguna clave de acceso para el usuario. Por esta razón también se mencionan bases de datos dentro de este elemento.

Podemos definir este elemento de la arquitectura como *software de apoyo*, generalizándolo como todo el software necesario en donde el motor RV se sustenta para correr apropiadamente la simulación virtual.

El siguiente elemento a considerar es el **usuario**. Podría pasar desapercibido, pero no debe ser así, ya que es un elemento sumamente importante que aporta casi la totalidad del perfil a la simulación. Del usuario depende el éxito o fracaso de nuestro desarrollo RV. Es cierto que debemos tener un panorama general bien establecido para nuestra simulación, pero es igual de necesario e importante contar con un conocimiento pleno de las personas a quienes estaría dirigido nuestro proyecto. Es imprescindible conocer las características de nuestros usuarios finales porque sin esa información nuestro desarrollo puede terminar siendo, más que un ambiente virtual de interacción con el usuario, en un simple paseo monótono y aburrido.

En resumen, la inmersión y la presencia en un desarrollo RV son fundamentales, y estas características están directamente involucradas con el usuario.

Por último tenemos a las **tareas.** En ciertas ocasiones nuestro usuario precisa hacer ciertas tareas, que no dependen directamente de él, pero que requieren de una simulación y una interacción con el entorno.

Un ejemplo es el caso de ambientes virtuales en los que se capacitan personas para que realicen ciertas actividades, pero antes de hacerlas en la vida

real, necesitan un ensayo o entrenamiento previo. En ese caso los usuarios han recibido ciertas tareas que deben ejecutar en el ambiente virtual.

Me parece que no es sencillo encontrar una clasificación determinada para este elemento, porque no necesariamente una tarea debe pasar por el usuario. También una tarea puede pasar directamente a los dispositivos de entrada, sin involucrar al usuario, provocando una situación en donde el usuario debería responder al sistema. Casos como éste son los simuladores de vuelo, en que no solamente el piloto practica horas de vuelo en el simulador, sino también se le envían datos de vuelo al simulador para que reproduzca en el ambiente virtual una situación de vuelo, a la cual el piloto debería estar preparado para responder.

Con el elemento de tareas finalizamos el estudio básico de la arquitectura de un sistema RV, la estructura es un ejemplo de cómo se conforma generalmente, pero no necesariamente necesitan estar los cinco elementos en nuestro desarrollo.

ES IMPORTANTE CONOCER LOS CINCO ELEMENTOS CLÁSICOS DE LOS SISTEMÁS RV, PERO NO IMPLICA QUE LOS MISMOS CONFORMEN NUESTRO DESARROLLO DE REALIDAD VIRTUAL, LOS ELEMENTOS QUE SI SON IRREMPLAZABLES ES EL MOTOR RV Y LOS DISPOSITIVOS DE ENTRADA / SALIDA.

En este capítulo hemos revisado algunos términos que se le dan a la realidad virtual, elementos de hardware importantes para la adquisición de datos, la manera en que podemos identificarla y algunos aspectos sobre la psicología del usuario. Información indispensable al momento de ir definiendo nuestro desarrollo.

En el siguiente capítulo analizaremos algunas herramientas que emplearemos para nuestro proyecto de Realidad Virtual, algunos editores de texto para programación, software de modelado 3D, programas de creación y

manipulación de imágenes y también para edición de video y audio, finalizando con una lista de motores que hay en el mercado para la creación de videojuegos.

Capítulo

2

Herramientas de Desarrollo

HERRAMIENTAS DE DESARROLLO

2.1 EDITORES DE TEXTO PARA PROGRAMACIÓN

Las herramientas siempre han sido necesarias para el hombre. Las encontramos desde tiempos muy antiguos, acompañando al ser humano para que realice sus actividades de una manera más cómoda y sencilla.

De igual manera que con las herramientas para las labores cotidianas, también contamos con herramientas de software que igualmente ayudan a programadores, diseñadores, personas de diferentes rubros, a aminorar la carga de trabajo o brindarle mejoras al software que permitan realizar sus labores.

En este caso, mencionaremos herramientas de software enfocadas a programadores. Nos referimos a programas o módulos encargados de realizar funciones específicas, dependiendo de la naturaleza de su desarrollo, éstos pueden contar a su vez con elementos adicionales, ya sean más programas o módulos que aligeran tareas o procesos mecánicos. En sí, las herramientas de software se diseñan para cubrir vacíos en ciertas áreas que no cuentan con el soporte adecuado.

Muchos lenguajes de programación poseen sus propios editores de texto de programación, pero hay otros lenguajes de programación que no cuentan con este soporte y es necesario usar herramientas externas para facilitarnos la tarea de

programar. Podemos usar, por ejemplo, el bloc de notas de Windows o Linux, el TextEdit de Mac Os o bien programas similares para escribir nuestras líneas de código, lo importante es tener algún editor de texto sin formato para poder implementar nuestro código al momento de compilarlo.

Pero no siempre es cómodo programar de ésta manera, ya que tener texto plano, sin formato y sin ninguna ayuda para ver o corregir nuestros errores, resulta sumamente difícil, incómodo y poco productivo. Para ello se han creado herramientas para programadores como son los *editores de texto para programación.*

2.1.1 CARACTERÍSTICAS DE LOS EDITORES DE PROGRAMACIÓN

Un editor de programación es un software que permite la edición de texto plano. Pero entonces, ¿cuál es la diferencia entre usar un editor de programación y un procesador de texto?

En un editor de programación podemos visualizar todos los caracteres que están presentes en el documento. Existen caracteres especiales llamados *'mark-up'* que son controles para coordinar la escritura de texto, por ejemplo: símbolos para representar un salto de línea, un salto de página o algún espacio de tabulador. En un procesador de texto encontramos lo que se llama 'formato enriquecido' (fileformat-specific), o 'control de caracteres' (control characters) el cual es definido en el set de caracteres. Estos elementos de formato (como puede ser el estilo de letras en 'negrita' o itálicas) contienen información adicional que no se ve, pero que sí puede ser leída por algún editor de texto plano. Esa información acarrea errores al momento de compilar nuestras líneas de código.

Para comprender un poco mejor este tema, en las siguientes tablas se muestran varios ejemplos. En la primera tenemos un ejemplo de líneas escritas

desde un procesador de texto convencional, las cuales cuentan con texto enriquecido, es decir, es un texto con formato y ha sido guardado con extensión '.txt' y en la segunda parte de la misma tabla tenemos el mismo archivo abierto desde un editor de programación, aún guardado con extensión '.txt'. (Figura 2.1)

La tabla siguiente muestra el mismo contenido pero escrito sin formato, utilizando un editor de texto como el TextEdit o el bloc de notas (aclarando que no se usa ningún estilo de formato en sus líneas), también guardado en '.txt' y en la segunda parte de la misma tabla se muestra el archivo abierto desde un editor de programación. (Figura 2.2)

Logramos observar que en la tabla 2.1, el mismo archivo abierto desde diferentes editores de texto muestra cierta información de formato que no aparece en el archivo original pero que es necesaria para los procesadores de texto, esta información extra es traducida por el procesador para saber qué tipo de estilo aplicar a las líneas escritas. Aún teniendo un estilo normal, aparece información acerca del estilo que se está empleando para la línea.

En cambio, en la siguiente tabla 2.2, podemos ver el mismo archivo pero escrito sin formato, empleando un editor básico de texto. Nos percatamos que al abrir el archivo, tanto en el editor de programación como en el procesador de palabras, el texto aparece con un formato plano, es decir, sin información añadida en las líneas, solo aparece que lo que nosotros hemos escrito.

prueba.txt (Texto Enriquecido). ***Desde un procesador de texto***

Texto de Prueba

Texto en Negrita

Texto en cursivas

Texto normal

prueba.txt (Texto Enriquecido). ***Desde un editor de programación***

{\txt\ansi\ansicpg1252\cocoartf1038\cocoasubrtf110
{\fonttbl\f0\fswiss\fcharset0 Helvetica;}
{\colortbl;\red255\green255\blue255;}
\margl1440\margr1440\vieww9000\viewh8400\viewkind0
\pard\tx566\tx1133\tx1700\tx2267\tx2834\tx3401\tx3968\tx4535\tx5102\tx5669\tx6236\tx6803\ql\qnatural\pardirnatural

\f0\fs24 \cf0 Texto de Prueba\
\

\b Texto en Negrita\
\

\i\b0 Texto en cursivas\
\
\pard\tx560\tx1120\tx1680\tx2240\tx2800\tx3360\tx3920\tx4480\tx5040\tx5600\tx6160\tx6720\ql\qnatural\pardirnatural

Se muestra el contenido de un archivo '.txt' con formato, abierto desde un procesador de texto (primera parte), y abierto desde un editor de programación (segunda parte).

Figura 2.1
Ejemplo de Texto Enriquecido

prueba.txt (Sin Formato). ***Desde un procesador de texto***

Texto de Prueba

Texto en Negrita

Texto en cursivas

Texto normal

prueba.txt (Sin Formato). ***Desde un editor de programación***

Texto de Prueba

Texto en Negrita

Texto en cursivas

Texto normal

Se muestra el contenido de un archivo '.txt' sin formato, abierto desde un procesador de texto (primera parte), y abierto desde un editor de programación (segunda parte).

Figura 2.2

Ejemplo de Texto sin Formato

Lo anterior es importante para nuestra investigación, porque algunas veces no nos damos cuenta y escribimos nuestro código en algún procesador de texto que lleva de antemano cierta información y justamente cuando queremos compilar nos aparecen una serie de errores que no tenemos idea de donde salieron y más cuando estamos completamente seguros que hemos escrito correctamente nuestro código, aún guardándolos según nosotros, sin formato enriquecido.

Otra diferencia es, que en los editores de programación podemos manejar lo que se conoce como *codificación de caracteres.* La codificación de caracteres es el método que se emplea para convertir caracteres que comúnmente se utilizan en ciertos idiomas pero que no tienen soporte o no existen otros lenguajes, es decir, letras del alfabeto o algunos símbolos ordinarios que existen en un sistema y que son necesarios ser representados en otro sistema.

Dentro del sistema de codificación de caracteres tenemos las *normas de codificación,* que son las que nos definen la forma en que se deben codificar los caracteres de acuerdo al sistema en el que se quieren representar.

Normalmente los editores de programación vienen en código UNICODE, el cual representó la solución a los problemas que acarreaban tanto el código ASCII como el código ASCII extendido. Se acordó internacionalmente utilizar el código UNICODE que es una tabla que asigna un código a cada uno de los símbolos de todos los alfabetos europeos, ideogramas chinos, japoneses, coreanos y muchas otras formas de escritura, incluyendo símbolos especiales.

Se cuenta también con el formato de codificación UTF-8 (8-bit *Unicode Transformation Format*), que es una implementación del UNICODE que contiene caracteres de longitud variable. Este tipo de codificación usa un grupo de bytes para representar la mayoría de los lenguajes. Algo importante es que también podemos escribir caracteres con acentos con éste sistema de codificación.

En la siguiente tabla se muestra un ejemplo de diferentes tipos de codificación y cómo son interpretadas las letras de acuerdo a cada tipo de codificación empleado.

Codificación UTF-8	ISO Latín 1	Mac OS Roman
a	a	a
e	e	e
i	i	i
o	o	o
u	u	u
á	Ã¡	√°
é	Ã©	√©
í	Ã-	√≠
ó	Ã³	√≥
ú	Ãº	√∫
ñ	Ã±	√±

Ejemplos de cómo se traducen los caracteres de acuerdo a la codificación empleada

Figura 2.3

Ejemplos de Codificación

Con lo anterior podemos entender lo importante que es escribir con determinado tipo de codificación, ya que si no ponemos especial atención en ello, puede que al final de nuestro desarrollo lo que mostremos en pantalla no tenga el sentido que deseamos y termine siendo una serie de símbolos que no pueda entender nuestro usuario. Para el idioma español se recomienda utilizar la codificación UTF-8, ya que con ella podemos escribir caracteres especiales como la 'ñ' y letras con acentos.

Otra característica que resulta de vital importancia es que algunos editores de programación nos permiten insertar nuestro propio estilo de programación. Esta es una característica excepcional ya que nos brindará la oportunidad de crear estilos que no son comunes o que no tiene soporte, ya sea por medio de algún documento o en forma de 'plugin'.

Los editores de programación nos proveen de ciertas herramientas como son las búsquedas y reemplazo de palabras, cortar, copiar, pegar y en ocasiones son comandos con instrucciones avanzadas, como ya mencionamos, también proveen de formato de texto, importación de archivos, tipo de codificación, filtros, resaltado de texto de acuerdo a la sintaxis, conocido como *syntax highlighting,* así como autocompletar mientras escribe; normalmente son menús emergentes que aparecen con una lista de posibles palabras para seleccionar, por si alguna de ellas es el comando que estamos por escribir. Con lo anterior resulta obvio que los editores de programación son herramientas sumamente importantes y de fácil acceso, que aminoran sustancialmente la carga de trabajo al momento de programar y estar editando nuestras líneas de código. Es necesario realizar un buen estudio de los editores de programación que existen en el mercado para seleccionar el que más se adecúe a nuestras necesidades y permitirnos así un mejor desempeño al momento de crear nuestros archivos para compilar.

LOS EDITORES DE PROGRAMACIÓN SON HERRAMIENTAS ESENCIALES QUE NOS AYUDARÁN AL MOMENTO DE ESCRIBIR NUESTROS CÓDIGOS CUANDO EL LENGUAJE DE PROGRAMACIÓN NO CUENTE CON ALGÚN SOPORTE DE EDICIÓN. LO IMPORTANTE ES ESCOGER EL EDITOR CORRECTO QUE SE ADECUE A NUESTRAS NECESIDADES.

2.1.2 EJEMPLOS DE EDITORES DE PROGRAMACIÓN

Existen muchos editores de programación en el mercado, hay unos que están especializados para ciertas áreas, como la programación web por ejemplo, hay otros que cuentan en su biblioteca de sintaxis con varios tipos de lenguajes. A continuación explicaremos de una manera general algunos de los editores de programación más comunes, sus características básicas, si son gratuitos o de pago y para qué sistema operativo están diseñados, incluyendo algunas propiedades que lo pueden distinguir de los demás.

El orden de los editores no depende de ninguna característica en especial.

EDITPLUS TEXT EDITOR

EditPlus es un editor de texto, enfocado para edición HTML, edición PHP y edición Java. Entre sus características tenemos:

- Resaltado de sintaxis para HTML, PHP, Java, C/C++, CSS, ASP, Perl, JavaScript y VBScript. También puede extenderse para otros lenguajes de programación basados en sintaxis de archivos personalizados.
- Barra de herramientas, número de línea, reglas, autocompletado, selección de columnas, motor de búsqueda y remplazo, múltiple copiado y pegado, revisor ortográfico y más.
- Editor de pago.
- Funciona únicamente para sistemas Windows.[18]

NOTEPAD++

Es un editor gratuito de código fuente que soporta varios lenguajes de programación y se ejecuta en sistemas Windows.

Algunas de sus características son:

- Sintaxis coloreada y envoltura de sintaxis, los lenguajes que soporta son C, C++, Java, C#, XML, HTML, PHP, CSS, makefile, ASCII art (.nfo), doxygen, ini file, batch file, JavaScript, ASP, VB/VBS, SQL, Objective-C, RC resource file, Pascal, Perl, Python, Lua, TeX, TCL, Assembler, Ruby, Lisp, Scheme, Properties, Diff, Smalltalk, Postscript, VHDL, Ada, Caml, AutoIt, KiXtart, Matlab, Verilog, Haskell, InnoSetup, CMake, YAML.

[18]EditPlus, http://www.editplus.com (ví: 21 de Septiembre de 2009)

- Autocompletado para la mayoría de los lenguajes soportados, aunque también el usuario puede hacer su propia lista de API (Application programming interface).
- Edición de varios documentos al mismo tiempo, también cuenta con multivista que es la posibilidad de tener dos vistas diferentes.
- Detección del estado del documento.
- Puntos de marca.
- Resaltado de paréntesis y sangría
- Grabación de marcos.
- Funciona únicamente para sistemas Windows.[19]

ULTRAEDIT TEXT EDITOR

Podría considerarse uno de los mejores editores de programación junto con jEdit. UltraEdit. Es un editor de programación y un editor hex que es usado para editar HTML, PHP, JavaScript, Perl, C/C++ y demás lenguajes de programación. Puede editar y manejar archivos que excedan de los 4 gigabytes. Soporta plataformas Windows de 32 y 64bits.

Entre sus características cabe destacar:

- Manejo de archivos mayores a 4 gigas.
- Formato de texto y edición Drag & Drop.
- Revertir cambios archivos activos.
- Soporte FTP.
- Integración a la plataforma Windows.
- Edición de macros.
- Creación de tu propia sintaxis.
- Funciona para sistemas Windows.[20]

[19]Notepad, http://notepad-plus.sourceforge.net/es/site.htm (ví: 21 de Septiembre de 2009)
[20]IDM Computer Solutions, Inc. http://www.ultraedit.com (ví: 21 de Septiembre de 2009)

TOTALEDIT

Es un editor freeware, esto significa que podemos contar con versiones gratuitas o con versiones comerciales. Para ver a detalle sus diferencias hay que revisar la página del proveedor. Pero entre sus características tenemos:

- Sintaxis de color personalizada de acuerdo al lenguaje que se este usando (únicamente para los lenguajes de su biblioteca).
- Comparación de archivos.
- Editor binario (hex editor).
- Búsqueda de frases en la computadora dentro de los archivos creados con TotalEdit Pro.
- Personalización de las barras de herramientas.
- Edición de los colores de sintaxis para los lenguajes de programación.
- Puede usarse en dispositivos USB, únicamente en Windows.
- Funciona únicamente en sistemas Windows.[21]

SLICKEDIT

Es un editor de programación multiplataforma, entre sus características encontramos:

- Nos permite escribir nuestras propias referencias de código y encontrarlas rápidamente.
- Expansión y completado de líneas a través de plantillas.
- Lista y explicación de funciones con el formato Javadoc, XMLdoc o Doxygen.
- Control de versiones.
- Edición de archivos arriba de 2 gigas.
- Grabación de macros.
- Editor de pago.

[21]CoderTools, http://www.codertools.com (ví: 21 de Septiembre de 2009)

- Corre bajo plataformas Windows, Mac OS, Linux, Solaris, AIX, HP-UX.[22]

CODA

Es uno de los editores de programación más fáciles y amigables que podemos encontrar, una deficiencia es que no permite la creación de nuestras propias sintaxis de lenguaje de programación. Fuera de eso, es uno de los mejores editores de programación para sistemas Mac Os. Algunas características son:

- Interfaz completamente integrada al sistema Mac OS.
- Editor de texto plano.
- Soporte FTP.
- Editor CSS
- Terminal.
- Libros sobre diferentes lenguajes de programación.
- Sintaxis de color para los principales lenguajes de programación, además de ser personalizable.
- Edición simultánea de código en equipos de trabajo.
- Consola JavaScript
- Trabaja con Subversion.
- Editor de pago.
- Funciona únicamente en sistemas Mac OS.[23]

ESPRESSO

Es uno de los editores de programación por excelencia en ambientes Mac OS, aunque está enfocado para programación web, su versatilidad en ambientes de desarrollo no sólo lo enmarca en desarrollos web. Su facilidad para escalar el entorno lo hace un potente editor de programación, y marca un enorme potencial a futuro.

---

[22]SlickEdit, http://www.slickedit.com (ví: 21 de Septiembre de 2009)
[23]Panic, http://www.panic.com (ví: 21 de Septiembre de 2009)

Entre sus muchas características tenemos:

- Entorno completamente integrado al sistema Mac OS.
- Editor HTML y CSS.
- Explorador de código.
- Cliente FTP.
- Escalable.
- Motor de sintaxis.
- Sugerencia contextual de código.
- Vista previa.
- Arquitectura basada en plugins.
- Creación de nuevos lenguajes gracias a sus extensiones (Sugar).
- Editor de pago.
- Funciona únicamente en sistemas Mac OS.[24]

A pesar que la plataforma en que se ejecuta es Mac OS, es una buena opción a considerar si se tiene el equipo humano, la solvencia económica y la necesidad de usar lenguajes de programación no comunes o personalizados.

JEDIT

Es un editor de programación que lleva bastante tiempo en el desarrollo del sistema (sin contar el tiempo que se ha empleado en el desarrollo de sus plugins). Su interface y su potencial lo hacen uno de los mejores editores de programación en el mercado. Cabe resaltar que es gratuito.

Por mencionar algunas características tenemos:

- Escrito completamente en Java, por lo que corre en sistemas Windows, Mac OS, Unix, VMS, inclusive en PenDrives (USB portables).

[24]MacRabbit, http://macrabbit.com/espresso (ví: 21 de Septiembre de 2009)

- Extensiones a partir de plugins.
- Se puede programar los plugins o descargarse del repositorio de jEdit.
- Autocompletado y sintaxis de código de más de 130 lenguajes.
- Es configurable y personalizable.
- Funciones de hacer y deshacer ilimitado.
- Copiar y pegar con un número ilimitado de portapapeles.
- Auto completado de instrucciones y funciones ya creadas.[25]

Si hago énfasis en la gratuidad del editor, es porque resulta sorprendente que un software de esta calidad carezca de costo, y se debe básicamente a la colaboración de todas las personas que han usado y creado herramientas para este software, donde cada granito de arena se convierte después en un cimiento impresionante.

MICROSOFT VISUAL STUDIO

Microsoft Visual Studio es un entorno de desarrollo integrado ('IDE') para sistemas operativos Windows. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros.

Entre sus características contamos con:

- Entorno de desarrollo integrado.
- Compatibilidad con la plataforma de desarrollo.
- Team Foundation Server.
- Suscripción a MSDN.
- Herramientas de prueba.
- Desarrollo de bases de datos.
- Depuración y diagnóstico.

[25]jEdit, http://www.jedit.org (ví: 21 de Septiembre de 2009)

- Administración del ciclo de vida de las aplicaciones.
- Arquitectura y modelado.[26]

XCODE

Xcode es el entorno de desarrollo integrado ('IDE') de Apple Inc. y se suministra gratuitamente junto con Mac OS X. Xcode trabaja conjuntamente con Interface Builder, una herencia de NeXT, una herramienta gráfica para la creación de interfaces de usuario.

Xcode incluye la colección de compiladores del proyecto GNU (GCC), y puede compilar código C, C++, Objective-C, Objective-C++, Java y AppleScript mediante una amplia gama de modelos de programación, incluyendo, pero no limitado a Cocoa, Carbón y Java. Otras compañías han añadido soporte para GNU Pascal, 1 Free Pascal, 2 Ada y Perl.3

Entre las características más apreciadas de Xcode está la tecnología para distribuir el proceso de construcción a partir de código fuente entre varios ordenadores. También cuenta con:

- Asistente de edición.
- Editor de código.
- Sistemas integrados de construcción.
- Compilador para varios lenguajes.
- Depurador gráfico.
- Análisis de código en tiempo real.
- Organización de código visual.
- Refactorización.[27]

---

[26] Microsoft, http://www.microsoft.com/spain/visualstudio/products/2010-editions/ultimate/features (ví: 6 de junio de 2011)

[27] Apple Inc. http://developer.apple.com/technologies/tools/features.html (ví: 6 de junio de 2011)

Cabe mencionar que existen muchos más editores de programación, inclusive puede ser que se haya omitido alguno de igual o mejor calidad, además en un par de años puede aparecer otro con mejores prestaciones de los que se han mencionado. Pero la finalidad era dar un panorama de las características de los editores para conocer sus alcances, saber qué es lo que necesitamos y clarificar con un poco de más conocimiento a qué editor recurrir.

2.2 Programas de Modelado 3D

En este apartado haremos un breve estudio de aplicaciones 3D, ya que no todos los programas de modelado 3D nos servirán para crear los modelos que integraremos a nuestro proyecto. Sólo mencionaremos los programas que pueden servir directamente al motor que vamos a usar.

Existe una gran variedad de programas populares para animación y modelado 3D: LightWave, 3ds Max, Alias, Maya, por mencionar algunos.[28]

Los programas de animación y modelado nos ayudan a crear los entornos virtuales como son las construcciones, los objetos, los vehículos, los personajes, es decir, todos aquellos elementos que darán forma a nuestro mundo virtual. Cada programa cuenta con sus propias características de modelado y animación, aunque todos están enfocados a la creación de modelos 3D, cada uno está orientado en realizar ciertas tareas de forma más rápida. Por ejemplo, Maya está enfocado a la animación mientras que 3ds Max hacia la construcción de complejos arquitectónicos y modelado 3D.

Los objetos que construyamos en nuestro programa 3D obviamente necesitan ser exportados a nuestro entorno virtual, existen pocos motores de videojuegos que soportan de forma nativa las extensiones de los modelos que fabriquemos. Para ello mencionaremos los motores más populares que cuentan con soporte para exportar nuestros diseños.

---

[28] Avgerakis, George. *Digital Animation Bible: Creating Professional Animation with 3ds Max, Lightwave, and Maya*: McGraw-Hill, 2003. p. 2

3DS MAX

3ds Max es un potente, fiable y accesible editor de gráficos 3D y paquete de animación. También es favorito entre los animadores y artistas de videojuegos. 3ds Max ofrece:

- Una lista de elementos basada en plantillas de carácter manipulación del sistema.
- Un conjunto de herramientas de modelado poligonal eficiente y texturización UV para flujos de trabajo.
- Extensa gama de plug-ins.
- Una vasta población de usuarios con experiencia para apoyar el desarrollo de juegos.[29]

Este programa podría considerarse entre uno de los que cuenta con mayor soporte para exportar nuestros modelos a nuestro ambiente virtual. En el capítulo 4 estudiaremos más a detalle la forma de importar nuestros elementos a nuestro desarrollo.

MAYA

Herramienta integrada completamente al modelado 3D, animación y renderizado. Es una solución basada en una arquitectura abierta, el software Autodesk Maya es una herramienta de elección para aquellos que requieren un alto nivel de control sobre sus flujos de trabajo de arte del juego y los oleoductos. Maya ofrece:

- Una amplia gama de fotogramas clave no lineal y de animación avanzadas, así como herramientas de edición de carácter.
- Un amplio conjunto de polígonos avanzados, NURBS y herramientas de superficie y subdivisión de modelado y texturizado.
- Interfaz de usuario unificada para representación y flujo de trabajo.

[29]Autodesk, http://usa.autodesk.com/ (ví: 28 de Septiembre de 2009)

- Las secuencias de comandos Python®, Maya Embedded Scripting Language (MEL), y una completa API permite a los desarrolladores personalizar la escala de su aplicación.[30]

BLENDER 3D

Blender es el programa de código abierto por excelencia para la creación de contenido 3D y está disponible para todos los sistemas operativos más importantes bajo la GNU General Public License. Entre sus características podemos encontrar:

- Una gama de tipos de objetos 3D incluyendo mallas poligonales, superficies NURBS, bezier y curvas B-spline, metaballs, fuentes vectoriales (TrueType, PostScript, OpenType).
- El modelado de malla basada en el vértice, EDGE y / o selección de la cara.
- Creación rápida esqueletos.
- Capas de hueso y grupos de colores para una mejor organización de la figura.[31]

LIGHWAVE 3D

En estos días, con el proceso creativo, los artistas necesitan avanzar a la velocidad máxima para hacer frente a los plazos increíbles, exigiendo trabajo y altas expectativas. El uso de LightWave ®, hace posible que un individuo, un equipo pequeño o de una empresa importante pueda ver sus sueños y sus ideas plasmadas dentro de los estándares de calidad, entregándolas a tiempo y acorde a su presupuesto. Artistas de todo el mundo dicen que la velocidad, flexibilidad, valor y el control que ganan con LightWave en todo el proceso creativo les permite hacerlo.

---

[30] Autodesk, http://usa.autodesk.com/ (ví: 28 de Septiembre de 2009)
[31] Blender, http://www.blender.org/ (ví: 28 de Septiembre de 2009)

LightWave ® es un completo programa de modelado, renderizado (proceso de generar una imagen a partir de un modelo, en este caso modelos 3D) y sistema de animación. Utilizado ampliamente en la producción de televisión, efectos visuales, desarrollo de juegos de vídeo, gráficos de impresión y visualización.

- Permite al usuario controlar la base de cálculo del CI. Estas opciones proporcionan un mejor rendimiento para ajustar la solución deseada para cada movimiento de la animación.
- Mejora de la precisión de los bezier cúbicos utilizada en el nodo de la curva.[32]

MILKSHAPE 3D

MilkShape 3D es un *low-polygon modeler* (modelador de objetos en baja generación de polígonos), que fue inicialmente diseñado para Half-Life. De esa antigua versión a la actual se han añadido muchas mejoras.

MilkShape 3D cuenta con todas las operaciones básicas como seleccionar, mover, rotar, escalar, extrusión, subdividir, entre más características. MilkShape 3D también permite la edición de bajo nivel con vértices y herramienta para la modelación de caras. Contiene herramientas de creación de figuras estándar y primitivas extendidas como esferas, cajas, botellas, etc. De igual manera cuenta con animadores y construcciones de esqueletos. Esto permite exportaciones como los formatos tipo Quake o exportar animaciones esqueléticas como Half-Life, Genesis3d, Unreal, etc.[33]

DELED

DeleD es un modelador 3D que se centra principalmente en el desarrollo de juegos 3D. También permite crear y distribuir modelpacks, texturepacks y plugins

[32]NewTek, http://www.newtek.com/lightwave/ (ví: 28 de Septiembre de 2009)
[33]Chumbalum, http://chumbalum.swissquake.ch/ (ví: 28 de Septiembre de 2009)

de tipo DeleD. DeleD se centra en cinco áreas que son:

1. **Edición de Geometría:** tiene muchas herramientas para modificar objetos en 3D, incluyendo Suavizado y operadores booleanos.
2. **Animación:** proporciona un sistema de animación básica que le permite crear objetos animados.
3. **UV Mapping:** permite mapeados para agregar texturas a los modelos a detalle.
4. **Lightmapping:** permite funciones avanzadas de iluminación (lightmapping).
5. **Raytracing:** puede crear imágenes impresionantes, con la incorporación del trazador de rayos.[34]

Algo interesante es que no se requiere de una avanzada técnica de aprendizaje, su estructura permite aprender rápidamente su uso para la creación de modelos 3D.

Existen más programas de modelado y animación 3D, incluso pueden contar también con el soporte para nuestro entorno virtual, pero solo hemos hecho mención de aquellos que cuentan con un mayor soporte para poder exportar nuestros modelos y diseños arquitectónicos. Vale la pena que cada lector, de acuerdo con sus necesidades, explore las ventajas y desventajas que cada software le puede proporcionar para su proyecto.

LOS PROGRAMAS DE MODELADO Y ANIMACIÓN 3D PROPORCIONAN HERRAMIENTAS QUE NOS PERMITEN REALIZAR NUESTRAS TAREAS RÁPIDAMENTE. DEPENDE DE NOSOTROS ESCOGER EL PROGRAMA ADECUADO ASÍ COMO TENER EN CUENTA NUESTRO NIVEL DE MODELADO PARA ELABORAR NUESTRAS FIGURAS.

[34]Delgine, http://www.delgine.com/ (ví: 28 de Septiembre de 2009)

2.3 DISEÑO GRÁFICO

No vamos hablar mucho de las herramientas necesarias para elaborar este tipo de productos, existen en el mercado bastantes programas que nos ayudan a realizarlos. Aunque si trataremos conceptos necesarios para poder entender más a fondo este tema.

2.3.1 DISEÑO GRÁFICO

¿Por qué tenemos que hablar del diseño gráfico en la construcción de un producto de realidad virtual? Porque es una forma de comunicación que utiliza textos y/o imágenes para presentar información. Conocer los elementos básicos del diseño gráfico es la base para cualquier disciplina en donde se vea envuelta la imagen, algunos de estos elementos son las figuras, las texturas, las líneas, los colores, términos que componen el vocabulario en el diseño visual.[35]

Hablemos de algunos términos básicos que nos ayudarán a expresar correctamente nuestras ideas cuando las tengamos que explicar a nuestros artistas o al departamento de diseño.

De acuerdo con la teoría clásica del diseño, su labor consiste en generar una emoción visual, es decir, tiene como objetivo la composición de los elementos de diseño crean un estilo, un mensaje, una expresión, hasta un estado de ánimo.[36] Si fusionamos estos elementos en una correcta armonía podemos crear en el usuario un verdadero sentido de inmersión en nuestro mundo.

---

[35] Edward Rodríguez, *Computer Graphic Artist*. Delhi: Global Media, 2007. p. 4
[36] Ibid. p. 5

Las computadoras dentro de la rama del diseño gráfico se han convertido en una herramienta indispensable, pero no hay que perder el suelo, aún existen muchos diseñadores que trabajan a mano como Milton Glaser. Obviamente, el uso del software de diseño gráfico no hará nuestro trabajo impresionante, ayuda, cierto, pero dependerá de nuestros diseñadores de arte y de la creatividad con la que ellos cuentan, así como de los estímulos que podamos brindarles.

Comentemos algunos conceptos que serán comunes a lo largo de nuestro proyecto, específicamente en el área de diseño gráfico. Empecemos hablando de la **tipografía,** que es el arte y la técnica de establecer un tema escrito utilizando una combinación de fuentes, tamaños de las mismas, el espacio entre palabras así como el espaciado de línea. Una **fuente** es el tipo de letra que estamos empleando, por ejemplo Arial, Times, Helvetica.

Graphic image development o simplemente conocido como **desarrollo de imagen (image development)** es un término empleado en la producción de gráficos (principalmente en producciones de gráficos por computadora). Los **gráficos** son representaciones visuales, ya sean imaginarias o que muestran algún entorno u objeto real. Este término es usado también para distinguir el proceso de **preparación de elementos,** los cuales son usados en medios de comunicación como ilustraciones, fotografías, etcétera, del proceso de **composición de elementos,** por ejemplo, la creación de impresiones, desarrollos web, películas, etc.[37]

La **imagen digital** es una representación en dos dimensiones con un finito grupo de valores digitales, llamados **pixels.** Muchas veces se confunde la idea de imagen con la de gráfico, debido a un mal empleo de concepto. Las imágenes digitales suelen contener demasiada información, misma que se ve reflejada en el número de bytes que ocupan dichas ilustraciones, las cuales en términos de

[37]Ibid. p. 30

almacenamiento no precisamente son aceptables, para ello se utilizan diferentes tipos de comprensión para los datos de imagen que nos ayudan a reducir el tamaño, como son el PNG (*Portable Network Graphics*) cuyo algoritmo de compresión resulta ser muy bueno ya que nos permite comprimir la imagen con una pérdida mínima de datos como el contraste, colores, efectos entre otros.

También se cuenta con el famoso jpeg, aunque existen debates en torno a si es un método de compresión o un formato de imagen; este tipo de compresión, como todos los que se manejan, siempre tendrá como resultado cierta pérdida de datos, ello significa que después de una compresión de datos no obtendremos la misma calidad de imagen que teníamos antes de que se le comprimiera. Lo interesante de los JPEG (*Joint Photographic Experts Group*) es que nos permite manejar la calidad de compresión que queremos en nuestra imagen. Si la compresión de nuestra imagen resultante es muy alta, la pérdida de información será también muy alta, obteniendo como resultado una imagen con una calidad muy pobre, pero en beneficio de un menor tamaño en el archivo. Contrario al ejemplo anterior, si la compresión de datos es muy pequeña, la calidad de la imagen no se verá afectada en gran medida, pero el tamaño del archivo será muy similar al del original. Entonces tenemos que el tamaño de compresión de la imagen es inversamente proporcional al tamaño del archivo (en bytes), mientras que la calidad de la imagen de igual manera es inversamente proporcional al grado de compresión que usemos.

Cabe aclarar que en cada compresión hay pérdida de datos, ya sea en mayor o menor medida, dichos datos no se vuelven a recuperar, lo que significa que en cada compresión, más allá del tipo que decidamos utilizar, no recuperaremos nuevamente la calidad de la imagen que se tenía antes de la compresión.

Existen otros tantos tipos de compresión de imágenes, pero no haremos mención de todos ellos, sólo hemos mencionado con los que trabajaremos.

EL TIPO DE COMPRESIÓN QUE UTILICEMOS AFECTARÁ DIRECTAMENTE LA CALIDAD DE NUESTRA IMAGEN. DEPENDE DE NOSOTROS ESCOGER EL TIPO DE COMPRESIÓN QUE MÁS SE AJUSTE A NUESTRAS NECESIDADES. LOS DATOS QUE SE HAN PERDIDO EN LA COMPRESIÓN NO SON RECUPERABLES.

2.3.2 GRÁFICOS POR COMPUTADORA

Podría parecer que en el apartado anterior ya hemos hablado un poco de este aspecto, pero más bien dimos los pormenores para entender los conceptos que nos encontraremos regularmente en la creación y manipulación de gráficos por computadora.

Gráficos por computadora o mejor conocido como **Computer Graphics (CG)** es un campo de la computación visual. La computación visual se desenvuelve en las áreas de la computación gráfica y sistemas multimedia. Retomando el concepto de CG, este campo utiliza las computadoras para generar y manipular gráficos, logrando integrar o alterar el campo y la información espacial que ha sido obtenida del mundo real, recreando un espacio verdadero o un concepto imaginario.[38]

La industria de los gráficos por computadora abarca muchas áreas, el enfoque que utilizaremos se basa en la creación y diseño de videojuegos. Ya se había comentado que las prestaciones de los motores de videojuegos brindan un amplio grado de comodidad y sencillez al crear un producto de realidad virtual no inmersivo, por ello, hablaremos un poco de la construcción de gráficos por computadora a partir del diseño de videojuegos.

---

[38] Wayne, Carlson. *A Critical History of Computer Graphics and Animation*: The Ohio State University, 2003.

Entender el concepto de gráficos por computadora es la llave para poder crear elementos visuales acordes con las necesidades de nuestros usuarios, como ya hemos mencionado la parte visual es sin duda uno de los eslabones fundamentales de la cadena que comprende nuestro proyecto de realidad virtual, esa parte gráfica puede ser realística, no realística o en ciertos casos hasta artística, el fin de la parte visual es ubicar al usuario dentro de un entorno gráfico en el cual se pueda sentir inmerso e identificado.

Retomando nuevamente los conceptos sobre el área de videojuegos, en el diseño se necesita consolidar varios aspectos, como es la parte gráfica, la programación, ya sea del motor o del mismo lenguaje del videojuego, la inteligencia artificial, la programación de red, el audio, la respuesta y detección de eventos externos como internos, física real o interpretada de acuerdo a la naturaleza del mundo virtual y sobre todo la respuesta en tiempo real. Pero no necesitaremos de todo esto para nuestro proyecto, sólo utilizaremos lo que más se adapte a nuestro desarrollo y adecuaremos las herramientas a nuestras necesidades.[39]

Hemos hecho hincapié en que el diseño de videojuegos no se aparta mucho de los proyectos no inmersivos de realidad virtual, incluso mencionamos que uno de los mejores ejemplos de realidad virtual no inmersiva son los videojuegos, por lo que usaremos esas herramientas para crear nuestro propio desarrollo. Más adelante se irá sustentando poco a poco el por qué de esta decisión.

El desarrollo gráfico en los videojuegos no necesariamente tiene que ser realista, tenemos algunos ejemplos en donde el mundo virtual no es nada parecido a la realidad, pero sí logra en el jugador un gran efecto visual que consigue envolverlo en el ambiente. Sin embargo en la realidad virtual (aunque no es un punto sumamente necesario el presentar un aspecto realístico, casi idéntico de

[39]Allen Sherrod. *Game Graphics Programming*: Thomson, 2009. p. 5

forma gráfica a la realidad) es innegable su importancia, misma que lo convierte en un elemento imprescindible. La realidad virtual trata de representar eso, un medio real a partir de entornos gráficos creados por computadora. Si estamos desarrollando un proyecto en donde nuestra meta es representar la realidad que nos rodea, no nos podemos dar el lujo de perder detalle en el aspecto gráfico de nuestro mundo virtual.

A partir de los años 90's, los gráficos 3D crearon una revolución en varios sectores de la sociedad, no sólo en la que se refiere a realidad virtual, sino también en los videojuegos, en la carrera aeronáutica y espacial, en la medicina y así otros tantos ejemplos más; todo ello ha sido gracias a los avances en la computación tanto en el hardware como en el software.

Los elementos 3D han ofrecido niveles de realismo más marcados que hace un par de años, las simulaciones de iluminación y sombras que se producen en los ambientes tridimensionales asemejan un entorno verdadero, más cercano a la realidad, la animación de los objetos y su complejidad con la que se presentan al usuario tanto en su física como en su forma, permiten que la persona interactúe con ellos, todo esto en conjunto, simula una realidad afín a nosotros.[40]

La información gráfica que se presenta en la computadora se maneja de dos maneras, puede representarse como mapas de bits (imágenes digitales) o como geometría vectorial. Las imágenes digitales son una trama de puntos de color (pixels) que en su conjunto representan la imagen, es muy común utilizar éste tipo de gráficos ya que su manipulación es fácil, pero están limitadas en calidad y precisión, además de ser planas, es decir, bi-dimensionales (figura 2.4ª).

Los gráficos compuestos por geometría vectorial guardan la información del gráfico a través de un sistema de coordenadas cartesianas (X,Y). Esta es la mejor

[40]Ibid. p. 8

manera de guardar gráficos, ya que no tenemos pérdida de información, porque cada figura se identifica por su geometría, de esta manera podemos escalarla sin ningún problema y no tenemos pérdida de información en la calidad de imagen. Naturalmente trabajar con imágenes vectoriales es más complejo, pero la bondad de las propiedades de éstas imágenes nos permiten trabajarlas de manera exactas sin pérdida alguna de información. Incluso no es difícil saltar de dos dimensiones (X, Y) a 3D, basta con agregar un tercer eje (Z) y ya tendremos así una figura tridimensional. Para la creación de este tipo de gráficos se utilizan programas denominados CAD (*Computer-Aided Design* o diseños asistidos por computadora), Figura 2.4b.

La información gráfica de estas imágenes es guardada en una base de datos numéricos, estos datos definen forma, tamaño, color, volumen y datos adicionales que sean necesarios para la representación de la figura, de tal forma que al momento de ser interpretada esta información, la computadora extrae esta información de la base y genera la imagen en la pantalla. La geometría generada por la computadora no es tridimensional, es en dos dimensiones, pero esta representación para figuras 3D reúne varias caras de las formas del gráfico (en una isométrica o perspectiva) dando la impresión de volumen en la figura. Lo interesante de estos gráficos es que la información de los gráficos permanece coherente en el sentido de que, si una parte de la geometría se altera, el conjunto de puntos que componen a la geometría también se modifican de manera proporcional al cambio manteniendo la relación correcta del gráfico sin pérdida alguna de información.[41]

[41] Juan Carlos Parra Márquez, *op. cit.,* p. 12

Representación de gráficos

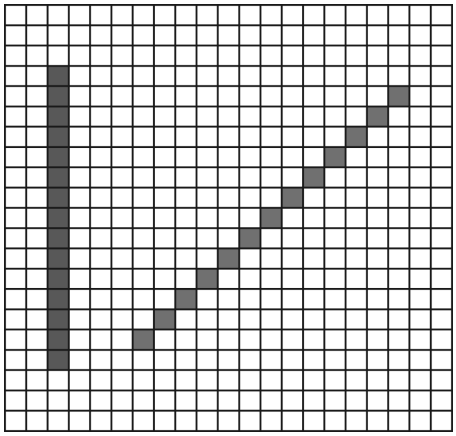


Figura 2.4.a: Mapa de bits

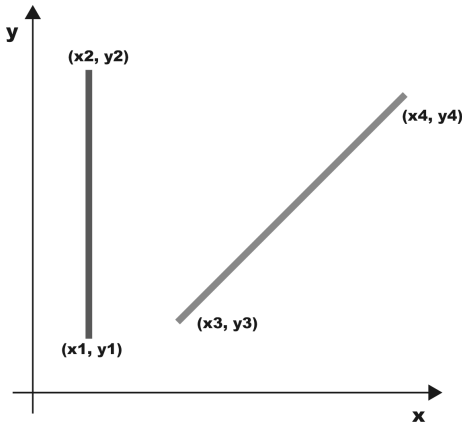


Figura 2.4.b: Imagen vectorial

En los mapas de bits contamos con una imagen que ya está conformada por puntos en una rejilla plana, mientras que en las imágenes vectoriales la información se representa por fórmulas o atributos matemáticos encargados de crear la imagen.

Figura 2.4
Representación de Gráficos

Aclaremos tres ideas importantes que nos ayudarán a obtener la ambientación de nuestros mundos virtuales en los desarrollos gráficos por computadora. Estos conceptos son maneras de representar la realidad por medio de gráficos creados por computadora que van en los rangos de 2D, 2½D y 3D. Esto apoyará de manera significativa en los procesos de edición de imagen y creación de nuestros modelos.

REPRESENTACIONES 2D

Son aquellas imágenes que no muestran algún tipo de profundidad, a partir de trazos básicos generan alguna figura en su concepción más simple, también se les suele llamar figuras primitivas.

La manera de entender la representación de las imágenes 2D no es la misma con la que podemos concebir nuestro mundo real, ya que las imágenes 2D son

representaciones planas, sin profundidad, pero que nos hacen referencia a ciertos objetos o entornos.

Las imágenes 2D son representaciones de entornos u objetos que necesitan un mínimo de detalle, pero que son necesarias para conocer ciertos elementos de los mismos, ya sea localización, dirección, distancias, etcétera, de manera que la persona que las está observando pueda asimilar de forma rápida lo que nosotros deseamos transmitirle. Por ejemplo, algunos mapas son representaciones planas que nos indican de manera concisa atributos de zonas geográficas o algunos dibujos de objetos que nos muestran de forma inmediata características de los mismos.[42]

REPRESENTACIONES 2½D

Las imágenes 2½D son representaciones tridimensionales, ya sean de modelos o entornos, proyectadas en un plano de 2D. Estas representaciones incluyen trazos a perspectiva, sistemas de proyecciones como las proyecciones isométricas y axonométricas, las cuales hacen referencia a las proyecciones ya sea de la misma figura o con respecto a los ejes respectivamente, mientras que en la perspectiva los objetos o las escenas son dibujadas a partir de un punto de fuga, que es el lugar donde convergen todas las rectas paralelas en una dirección, este punto se sitúa en el infinito imaginario del plano. Podemos tener tantos puntos de fuga como deseemos.[43]

---

[42] Jennifer, Whyte, *Virtual Reality and the Built Environment*: Architectural Press, 2002. p. 35
[43] Ibid., p. 37

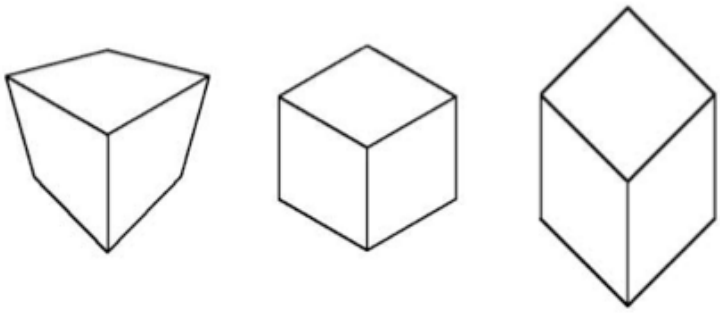
Conjunto de representaciones

Las figuras muestran tres representaciones, en perspectiva, axonométrica e isométrica respectivamente. **Jennifer Whyte, p. 37**

Figura 2.5

Conjunto de representaciones en profundidad

REPRESENTACIONES 3D

Las representaciones tridimensionales son modelos que nos permiten ver aspectos espaciales de entornos o ambientes que existen en nuestra realidad o mundos ficticios. Estos aspectos han formado una herramienta imprescindible, sobre todo en áreas como la arquitectura, diseño, ingeniería, dándole nueva forma de ver la realidad bajo un aspecto digital.

Las modelos 3D son creados a partir de zonas espaciales de los programas 3D llamadas escenas, estos objetos constan con una estructura de volumen generada por cálculos matemáticos y geometrías básicas como los cubos, esferas, conos, etcétera.

A pesar de todo, aun no contamos con la tecnología suficiente para poder representar nuestros modelos en 3D bajo un aspecto tridimensional real, esto significa que las características de nuestras pantallas o proyectores muestran nuestros modelos 3D en planos 2D a través de representaciones 2½D. Aunque nuestros modelos sean creados con una estructura tridimensional nosotros los

veremos en 2½D. Estas representaciones son mostradas a partir de proyecciones axonométricas o isométricas.

Aunque la RV se describe como interactiva, espacial, representaciones en tiempo real, todo eso es visto por el usuario como representaciones 2½D, pongamos un ejemplo: cuando nosotros creamos un modelo en algún programa de modelado 3D lo estamos haciendo a partir de la realidad, es decir, lo hacemos con un ancho, un alto, un largo, con un volumen que se asemeja al real (consideremos que el programa de modelado es para el diseñador 3D, como el torno para un alfarero). Pero al momento de presentar el modelo, el hardware no lo mostrará como una representación luminosa que esté flotando en el aire, sino que se basará en la construcción de planos con sistemas de proyecciones adecuados al objeto que se está proyectando, a partir del punto de vista con el que se está observando, es decir no pasa de ser una representación 2½D. Ya es un hecho que los proyectores holográficos están en nuestro mundo, [44] pero aún es difícil poner esa tecnología a la mano de la gente, por lo que deberemos esperar un tiempo para que nuestros modelos 3D puedan ser representados 'flotando en el aire'.

Existen muchas maneras de categorizar los modelos tridimensionales, depende el área en la que se está trabajando, en nuestro caso, las podemos categorizar como ***estáticas, dinámicas e interactivas.*** Las estáticas son modelos físicos hechos normalmente en programas CAD, dichos modelos se pueden observar desde cualquier punto espacial. Los modelos dinámicos cambian con respecto al tiempo, estos modelos los podemos visualizar en animaciones computarizadas. Los modelos interactivos son estructuras generadas por computadora con ciertas características a considerar como: poder interactuar con

[44] ICT, http://gl.ict.usc.edu/Research/3DDisplay/ (ví: 12 de Octubre de 2009)

ellas, son generadas en tiempo real y son representaciones espaciales, objetos descritos como parte en la RV.[45]

2.3.3 DÁNDOLE VIDA A LAS LÍNEAS

Hasta el momento hemos dado un vistazo rápido a algunos conceptos básicos del diseño gráfico por computadora, la intención no es tratar a profundidad ni hacer un tratado completo sobre el tema del diseño gráfico o de los gráficos por computadora, pero sí pretendemos dar una pequeña semblanza de algunos conceptos que nos estarán acompañando a lo largo de nuestro proyecto, más adelante iremos consolidando los conceptos a medida que los vayamos utilizando, basta por el momento tenerlos en consideración.

Cuando presentamos imágenes a través de medios digitales, ya sean pantallas, proyectores, monitores, LCD's, algo importante que debemos de considerar es la forma en que se presenta el color. En los gráficos por computadora hay una necesidad de especificar colores de tal manera que sean compatibles con el hardware que estamos usando a la vez que sean comprendidos por nuestro usuario.[46] Para tener un ejemplo de lo anterior, juguemos un poco con el monitor de nuestra computadora, en la parte de configuración de monitor, la mayoría de los controladores de nuestras pantallas nos muestran una lista de posibles configuraciones que puede utilizar nuestro dispositivo, a medida que los vayamos variando podemos observar cómo los colores van cambiando, pero no es la misma configuración para una pantalla LCD que para un monitor que utiliza una configuración RGB.

A continuación discutiremos un poco sobre los modelos de color que se

---

[45]Ibid., p. 39

[46], Haim, Levkowitz. *Color Theory and Modeling for Computer Graphics, Visualization, and Multimedia Applications*: Kluwer Academic, 1997. p. 46

utilizan con frecuencia en los gráficos por computadora.

COLOR

El color es algo importante y no siempre se muestra ante nuestros ojos como quisiéramos mostrarlo, hay una frase que realmente engloba todo el estudio del color y me parece muy acertada.

> "*El color es el elemento que produce e incorpora una dimensión mágica en la comunicación visual. Refleja el mundo cotidiano y la experiencia humana, dándole al diseñador un lenguaje común con el cual expresa humor, emociones y significados"* [47]

En primera instancia los colores primarios originalmente son los que nos ofrece el espectro: rojo, naranja, amarillo, verde, azul, índigo y violeta. En la teoría clásica del color (pintura, diseño gráfico, superficies de color, etcétera) usualmente se refiere a los tres colores, rojo, azul y amarillo (conocidos como colores fundamentales o elementales). A partir de ellos, en teoría, podemos obtener los demás colores y combinando los tres colores elementales obtenemos el negro. Con respecto a las imágenes digitales o los dispositivo que despliegan colores (por ejemplo los monitores) los colores primarios son el rojo, verde y azul (en realidad no es azul como tal, es un azul violáceo), esta gama se denomina como RGB y cuando mezclamos estos colores obtenemos el blanco. En cuestiones de impresiones, esa gama primaria cambia al cian, magenta y amarillo denominada como CMYK. Un ejemplo práctico es observar las tintas o los tóners de nuestras impresoras.

Sin embargo, el tema de cuál configuración es mejor para el tipo de dispositivo que estamos empleando ya no es un punto a considerar, La pregunta que salta es ¿por qué ya no?, esto se debe a que de todas las computadoras en

---

[47] Bob Gordon y Maggie Gordon. *The Complete Guide to Digital Graphic Design*: Thames & Hudson, 2005. p. 54

donde correremos nuestras aplicaciones, un gran porcentaje de ellas (por no decir todas), ya contarán con la configuración más adecuada, y esta ha sido otorgada por el controlador del dispositivo o configurada personalmente por el usuario, nosotros únicamente adoptaremos esa configuración, de manera que ya no es necesario reestructurarla.

Como hemos visto anteriormente, para la aplicación de color en nuestros modelos utilizaremos la gamma RGB que es la más adecuada para ser representada por dispositivos digitales.

¿Sólo nos importa el color de nuestra aplicación?

Depende a qué nos estemos refiriendo con esta pregunta, si únicamente vamos a analizar la cuestión del desarrollo como tal, es decir, una aplicación o un software que se ejecutará únicamente en una computadora y se mostrará por medio de un hardware de video ya sea un monitor, una pantalla o algo similar, con lo que hemos analizado bastaría. Pero si también vamos a darle una presentación física, entra en juego otros elementos como la mercadotecnia, la publicidad y de igual manera el color, este último visto desde otro ángulo.

Existe una tendencia llamada “pronóstico del color”, utilizada sobre todo por los modistas y gente de la industria textil. Esta metodología está enfocada para esas áreas, pero tomar provecho de los estudios existentes también nos ayuda en la parte de la presentación del producto final, obviamente no la usaremos dentro de un recorrido donde tenemos que acercarnos lo más que se pueda a la realidad, pero sí lo podemos proyectar a la presentación de nuestro producto al público.[48]

El hecho de comprender un poco el pronóstico del color nos ayuda para

[48]Tracy Diane, Tom Cassidy. *Colour Forecasting*: Blackwell Publishing Ltd, 2005.

negociar y anticipar la tendencia del color en nuestros productos, ya sea la demanda, las preferencias del consumidor, incluso poder crear un color característico que nos represente. La mercadotecnia, entendida como una herramienta y no como un instrumento de venta, es la que influye en la decisión del consumidor para comprar. Sin embargo, una correcta elección del color, basada en las preferencias del cliente, influirá de igual manera en el consumidor.

EL COLOR CAUSA EN EL CLIENTE EMOCIONES QUE PUEDEN IDENTIFICAR NUESTRO PRODUCTO INMEDIATAMENTE, INCLUSO, ACERCAR A PERSONAS QUE NO ESTÉN INTERESADAS EN NUESTRO PRODUCTO. PRONOSTICAR UN COLOR PERFECTO ES LA BASE PARA ATRAER LA MIRADA DE NUESTROS CLIENTES Y COMPRADORES.

TEXTURAS E ILUMINACIÓN

Las texturas juegan un papel importante en el diseño de los mundos virtuales. La texturización y la iluminación combinadas adecuadamente brindan en la escena ambientes realistas hasta con un cierto nivel, si lo deseamos así, de dramatismo en la misma.[49]

Crear excelentes texturas nos ayudan enormemente al momento de diseñar los elementos del mundo virtual, ya sean objetos, los terrenos, nuestras escenografías, etcétera. Por ejemplo, al aplicar correctamente una textura de madera a un cubo, podemos obtener una caja rota con una estructura ya sea de metal o de madera, y en la misma textura se pueden dibujar detalles como los clavos o tornillos, evitándonos la necesidad de tener que modelar también estos elementos, optimizando tanto la estructura de nuestro modelo como el tiempo que empleemos al realizar nuestra figura.

---

[49] Jean-Marc Gauthier, *Building Interactive Worlds in 3d: Virtual Sets and Pre-arVisualization for Games, Film, and the Web.*: Elsevier, 2005. p. 83

Cuando encontramos en una imagen texturizada características que tienen una mínima variación o atributos periódicos en toda la imagen, se denominan ***patrones.*** Los patrones son de gran utilidad para texturizar grandes áreas o en su defecto, figuras que básicamente no muestran demasiados cambios en su composición. Algunas texturas también son creadas a partir de unir varios *subpatrones*, los cuales deben de coincidir tanto en estructura como en dimensiones para que al ser aplicadas, dichos patrones repetitivos no muestren diferencias al ser colocados en nuestros modelos.

Conjunto de representaciones

Textura para un barril.

Textura aplicada a un cilindro

Una textura aplicada correctamente a una forma básica nos ahorrará tiempo y sobre todo, eficiencia en la carga de elementos en nuestro recorrido.

Figura 2.6

Textura

Los atributos y utilidades de las texturas se pueden resumir de la siguiente manera:

- Las texturas son patrones repetitivos, que caracteriza a la superficie de varias clases de objetos. La clasificación e identificación de estos objetos resulta sumamente fácil si la textura es lo más semejante posible al objeto de la realidad.
- Las texturas proveen información vital acerca del orden o disposición de los objetos en nuestro entorno virtual.
- Los atributos con los que cuenta la textura pueden describir en términos cualitativos al objeto, como puede ser la rugosidad, la homogeneidad con algún otro elemento el entorno, su orientación, su estructura, etcétera.[50]

Algo que debemos tener en consideración es que entre más creíbles sean nuestras texturas, nuestro desarrollo de realidad virtual se asemejará más a nuestro mundo cotidiano. La mayoría de los objetos que nos encontramos en la vida real, no tienen una sola textura básica, sino que lo conforman un conjunto de materiales; encontrar el punto en que podamos representar ese objeto con nuestra textura es la meta que perseguimos, de ello depende que nuestro objeto pase de ser una simple figura sin sentido, perdida en la inmensidad de nuestro entorno virtual, y se convierta en un objeto que pertenezca y se identifique con esa realidad que estamos representando.

Otro factor importante es la ***iluminación,*** crear objetos únicamente con texturas no es suficiente. La texturización le da forma y realismo a nuestro modelo, pero la luz le da vida al mismo. A donde observemos, de alguna u otra forma siempre hay un haz de luz iluminando nuestro entorno.

La iluminación es un tema importante, pero no abordaremos completamente el tema en este punto, solo mencionaremos los conceptos básicos y más adelante volveremos a retomar el tema pero directamente en el desarrollo de nuestro

[50] Tinku Acharya, *Image Processing Principles and Applications*: Wiley-Intersciencie, 2005. p. 182

proyecto, ya que en nuestro caso, quien va a proveer éste elemento será nuestro motor.

La iluminación ambiental es indispensable, crea una atmósfera más real que la que simplemente podemos obtener con las texturas, manejar la iluminación es algo muy complejo, se necesita de sensibilidad y, sobre todo, de capacidad de observación. El colocar luces por todos lados implica que podamos perder el realismo de nuestro ambiente y convertirlo en una masa de luz, o por el contrario, al no colocar correctamente nuestra iluminación, podemos perder la realidad del entorno convirtiéndolo en un lugar que nada tenga que ver con el lugar que pretendemos representar.

La iluminación ambiental proviene de la iluminación o la luz que se encuentra en el medio ambiente real. Una pista es saber cuánta luz está reflejando cierto material, lo cual es indicativo de qué tan iluminado está el ambiente en ese momento. Tampoco hay que perder de vista que las luces tienen atributos como es el color, la intensidad, rango, y tipo.[51]

Podemos encontrar varios tipos de luz en escenas virtuales, los tipos más comunes de iluminación que tenemos son:[52]

- **Luz direccional (*Directional lights*):** la fuente de esta luz se asume como infinita, en este tipo de iluminación no importa qué tan lejos se encuentre la fuente, de manera que los rayos de luz mantienen una forma paralela de uno con respecto al otro. El sol es el ejemplo clásico de luz direccional.[53]

- **Luz puntual (*Point light*):** Es similar a la luz direccional, con la diferencia

---

[51]David H Eberly, *3d Game Engine Architecture*: Morgan Kaufmann, 2006. p. 223

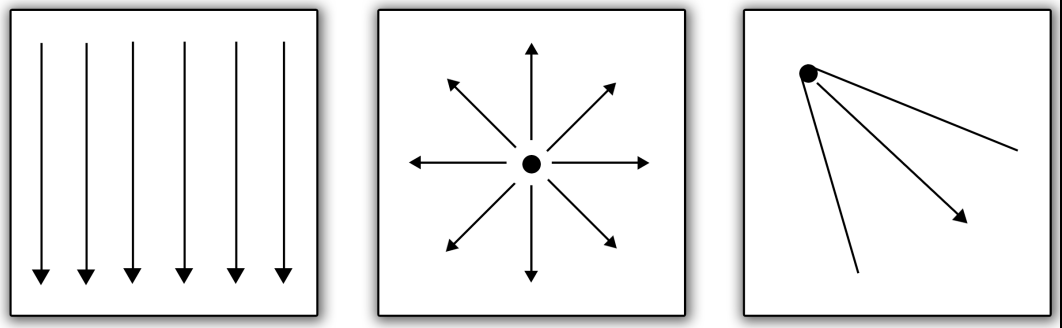
[52]Allen Sherrod, op. cit., p. 346

[53]David H Eberly, *3D Game Engine Design A practical approach to Real-Time computer graphics*: Morgan Kaufmann, 2001. pp. 100 - 101

que la luz se va desvaneciendo conforme se va alejando la fuente con respecto a la zona que estamos iluminando. Esta se conoce como ***distancia de atenuación***, y en la computación gráfica es muy común usar este tipo de luz para crear efectos en la escena. La luz puntual tiene una posición en el espacio la cual dependerá directamente sobre la magnitud de la distancia de atenuación. Cabe mencionar que este tipo de luz es irradiada en todas direcciones.

- **Luz spot (spot light):** es parecida a la luz puntual, con la diferencia que sólo alumbra hacia una dirección específica semejante al volumen de un cono.

Tipos más comunes de Luz



Luz Direccional **Luz Puntual** **Luz Spot**

Visualizar el tipo de luz correcta de acuerdo a la escena, dará mas vida a nuestro recorrido.

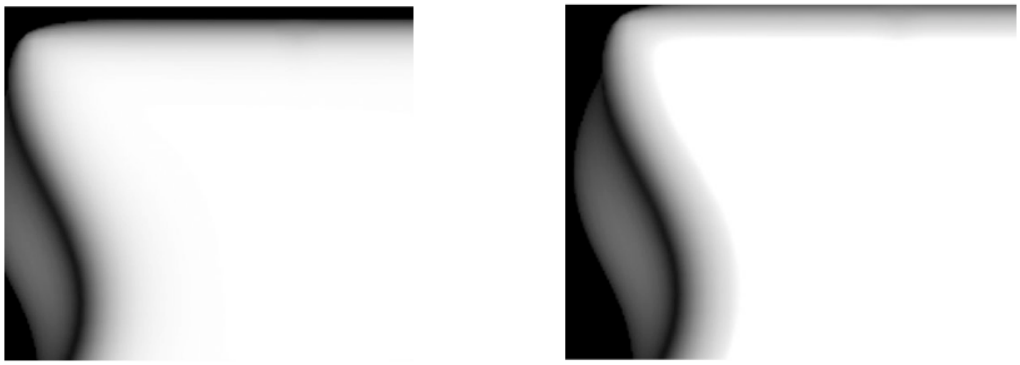
Figura 2.7

Tipos mas comunes de Luz

Ya sólo no falta hablar un poco de los términos que se emplean en los motores para ciertas características en la iluminación, estas propiedades son las que nos permitirán manipular el aspecto de nuestra luz para crear mejores ambientes. Los términos más comunes son:

- **Emisión:** indica cuánta luz es irradiada sobre una superficie.
- **Ambiente:** representa la luz que es 'rebotada' completamente alrededor de la escena y en toda la superficie.
- **Difuso:** es la luz que ha sido reflejada, es decir, después que la luz toca la superficie de algún objeto ésta es diseminada en muchas direcciones.
- **Especular:** la luz especular es semejante a la luz difusa, solo que ésta es reflejada en dirección opuesta a la fuente de luz.

Términos más comunes en la creación de Luz



Difuso **Especular**

Sherrod, Allen, op. cit., pp. 349 - 351

Figura 2.8
Términos en la creación de Luz

LAS TEXTURAS PROVEEN EL MAYOR REALISMO POSIBLE A NUESTROS MODELOS, LOS CUALES DEBEN ESTAR CONSTRUIDOS CON EL MÍNIMO NIVEL DE DETALLE POSIBLE, MIENTRAS QUE LA ILUMINACIÓN AYUDA A RECREAR LA ATMÓSFERA DEL ESPACIO QUE DESEAMOS ERIGIR.

2.4 AUDIO Y VIDEO

¿Qué termina por complementar un entorno?, efectivamente es el audio. Un efectivo tema de audio aumenta el nivel de inmersión, agregando los últimos efectos anímicos que nos harían falta por presentarle al usuario. Ya hemos mencionado los efectos visuales y su relevancia en un ambiente virtual, aunado a ello, el sonido complementa las emociones que generamos con la vista. La ventaja es que a medida que avanzamos en el desarrollo de nuevas tecnologías aumenta la capacidad de reproducción en audio de los equipos, incrementando el realismo del sonido que desplegamos en el ambiente virtual.

Así como el diseño gráfico es toda una especialidad que no concierne directamente con nosotros, de igual manera, los creadores de audio y compositores tienen detrás de ellos una gran escuela que ha durado muchos años, no se espera que al finalizar el tema de audio ya seamos capaces de componer obras monumentales para nuestro desarrollo de realidad virtual, pero sí de entender el audio de una manera básica.

AUDIO

Cualquier forma de creación de sonido puede considerarse como una pequeña composición musical. Antes que nada, tratamos de descubrir el sonido correcto que enmarca ciertos objetos o entornos dentro de nuestro marco virtual, seleccionamos el audio que más se acerque al que deseamos, ya sea que lo produzcamos de forma digital o lo grabemos de nuestro medio ambiente, la finalidad, ante todo, es obtener y guardar ese sonido.

Pero no siempre es suficiente contar solo con sonidos ambientales, también entra en juego el sonido vocal, conocido como '*voice-overs'.* Esencialmente no es más que el sonido de la voz de una persona que hemos obtenido mediante un micrófono y una grabadora de audio. Ya dependerá de nosotros encontrar el matiz

correcto de la voz que necesitemos para nuestro recorrido, para ello, no hay mejor forma de encontrarla que con una audición.

El audio es la combinación de todos los efectos de sonido, voces y música que se reproducen a lo largo del producto,[54] ya sea un videojuego o un recorrido virtual. Probablemente ni nuestra experiencia, ni los métodos teóricos con los que contemos, tampoco el hardware y software y mucho menos el equipo humano ni el capital económico nos permitan crear un verdadero trabajo de audio en nuestro desarrollo, pero lo que sí podemos hacer es recrear, lo más fielmente posible, una experiencia real para nuestro usuario a partir de los sonidos que tengamos a nuestro alcance: me refiero a los sonidos que podemos obtener de la naturaleza o del medio en donde estemos recreando nuestro recorrido virtual.

Empecemos por lo esencial, nosotros no somos programadores multimedia, de manera que no vamos a programar librerías para poder reproducir el sonido que estemos obteniendo a partir de diferentes medios. Lo que haremos es utilizar librerías que ya han sido creadas para manejar ese tipo de archivos digitales. Afortunadamente existen varios sistemas de reconocimiento de audio disponibles en el mercado, ya sea para ambientes de entretenimiento multimedia o aplicaciones de videojuegos. Esos sistemas son incorporados mediante librerías las cuales podemos enlazar o utilizar para nuestros productos, evidentemente esto es más fácil para un programador que no tiene conocimientos en el área de sonido, de esta manera estamos asegurando el soporte para nuestro audio.

Antes de continuar vale la pena hacer una equivalencia entre los términos que se utilizan en la creación de audio y los conceptos que nosotros (área científica) manejamos.[55]

---

[54]Les Pardew, Alpine Studios. *Game design for teens*: Thomson, 2004. p. 13
[55]Ibid., p. 142

Similitudes en la creación y manejo del Sonido	
Área Científica	**Músicos / Diseño de Audio**
Periodo	Longitud de onda
Amplitud	Volumen
Frecuencia	Tono
Hertz	Frecuencia de muestreo

Figura 2.9

Similitudes en la creación de Sonido

Existen muchos tipos de librerías de audio y su fuente se debe a: los fabricantes de sistemas operativos, compañías de audio, fabricantes de tarjetas de sonido, comunidades de programación de audio en código abierto y compañías que desarrollan software para audio. Pero el factor más importante que debemos considerar es: ¿en qué sistema operativo necesito el soporte para mi audio? La respuesta a esta pregunta es la que nos guiará para seleccionar correctamente las librerías que mejor se ajusten a nuestro proyecto.[56]

BEATNIK AUDIO ENGINE

Es un motor de audio escalable para PC's, asistentes personales (PDA's), teléfonos móviles y dispositivos digitales similares. Provee sintetizador MIDI (Musical Instrument Digital Interface) así como un motor de sonido digital pregrabado. Soporta sonidos simultáneos a partir de múltiples archivos.

FMOD

Firelight Technologies nos ofrece un motor de audio que se ejecuta en varias plataformas como Windows, Linux, Macintosh y recientemente en consolas como

[56]Martin Wilde, *Audio programming for interactive games*: Elsevier, 2004. p. 9

el GameCube, PlayStation2 y la XBox. Soporta la API de audio DirectX en PC, pero no es necesaria. Funciona en cualquier computadora con una tarjeta de sonido de 16 bits.

DIRECTX AUDIO

Una de las distribuciones más extensas de librerías de audio es DirectX Audio Application Programming Interface (API) de la empresa Microsoft. Es parte de la gran familia DirectX y está disponible exclusivamente en productos de la familia Windows o en las consolas XBox.

GAMECODA, MILES SOUND SYSTEM AND MUSYX

El sistema de GameCODS corre en PC's, PlayStation y XBox mientras que Miles Sound System únicamente se ejecuta en sistemas Windows y Macintosh. MusyX es el sistema oficial de audio y sonido de la familia Nintendo. El único problema de estos sistemas es que no los podemos descargar de forma gratuita y sólo están disponibles para desarrolladores con licencia.

MAC OS X CORE AUDIO

El Carbon Sound Manager en los sistemas OS X, es una solución de audio eficiente y flexible tanto para manejar audio digital como para el procesamiento de señales, provee de forma nativa audio multicanal permitiendo escalarlo a n-canales. La API provee un extenso soporte para MIDI además de contar con un administrador de configuración para ajustar múltiples puertos.

SOLUCIONES UNIX

En plataformas UNIX, tradicionalmente los mismos vendedores de productos ofrecen sus propias API's para el procesamiento de audio digital. Esto significa que cada aplicación cuenta con su propia API de audio, con frecuencia re-escriben el código para implementar el correcto soporte de audio, aunque no siempre la calidad es buena, y es más seguro contar con pérdida de calidad en el sonido.

Tecnologías que no precisamente se encuentran desarrolladas para estos sistemas pero si cuentan con soporte para estas plataformas, procuran mejores soluciones como puede ser JAVA. El Open Sound System (OSS) es un grupo de drivers que provee un API uniforme para la mayoría de las arquitecturas UNIX. También contamos con ALSA (Advanced Linux Sound Architecture), que suministra soporte tanto para audio como para archivos MIDI en plataformas Linux. ALSA soporta todos los tipos de interfaces de audio, como puede ser de tarjetas de sonido proyectadas a los consumidores comunes así como a interfaces profesionales de audio multicanal.

OPENAL

La Open Audio Library, es un API tridimensional de audio disponible para sistemas Windows y Macintosh. Su objetivo principal es permitir a un programador distribuir correctamente las fuentes de audio en un espacio tridimensional en torno a un oyente. El SDK (*Software Development Kit*) de OpenAL está disponible en el sitio para desarrolladores de Creative Labs y fue co-desarrollado por Creative Labs y Loki Entertainment Software.

Así como también existe compresión de imagen, también contamos con compresores de audio, conocidos como ***códec de audio,*** y su función principal es la de reducir cuantitativamente datos digitales en una señal auditiva. Tenemos estándares para el campo de la videoconferencia y estándares que provienen del campo del entretenimiento y multimedia. Entre los estándares más usados para el entretenimiento y multimedia contamos con el MP3, Ogg Vorbis y AC3, los anteriores códec de audio se consideran con pérdida en su compresión.

Terminaremos esta sección ofreciendo algunas pequeñas pistas para saber en qué momentos podemos aplicar audio en nuestro entorno virtual:

- En movimientos marcados o transiciones de niveles.
- Efectos de audio para el entorno o medio ambiente.
- Piezas musicales para acentuar puntos dramáticos en el recorrido.

- Diálogos.
- Interfaces de audio, imagen o video.
- Efectos especiales.

VIDEO

Introducciones, preámbulos, prólogos, comienzos, intervalos, cargas de niveles, finales, créditos, cualquier espacio en el que consideremos que sea necesario un video para no perder la atención del usuario mientras suceden procesos en segundo plano es el lugar adecuado para insertar un video.

El video se ha convertido en una herramienta informativa y de entretenimiento primordial para los procesos de multimedia. Como hemos leído con anterioridad, la compresión es una herramienta importante para reducir el número de bits que su usan para representar cada pixel en una imagen. Pero ¿por qué hablamos de imagen cuando ya estamos en la parte de video?

> *"Un video clip es una secuencia de fotografías o cuadros. Cada fotografía puede ser procesada de manera aislada, al igual que una imagen fija. Un ejemplo típico es el estándar JPEG."*[57]

En sí, la industria del video y animación no precisamente continúa utilizando fotografías, también utilizan imágenes, dibujos, representaciones animadas de algún objeto o personaje ya sea tridimensional o en dos dimensiones; a lo que vamos es que, un video clip en conjunto es una secuencia de imágenes que proyectadas a una cierta velocidad da la sensación de movimiento, pero al final no dejan de ser imágenes fijas.

Se cuentan con varios estándares para la industria del video, pero hay dos principales el NTSC (*National Television System Committee*). Este sistema es

---

[57] David Austerberry, *The technology of video and audio streaming*: Elseiver, 2005. p. 79

utilizado en Norte América, Asia y algunas partes de América Latina. Mientras que el oriente y occidente de Europa utilizan el sistema llamada PAL (*Phase Alternate Line*).[58]

No detallaremos los sistemas NTSC y PAL, basta mencionar una de las características principales que diferencian estos sistemas. Como ya sabemos, la ilusión de movimiento es creada al proyectar un determinado número de cuadros o imágenes por segundo. En este caso, para un sistema NTSC se hace pasar 30 cuadros por segundo (FPS, frames per second). Mientras que los sistemas PAL pasan 25 cuadros por segundo. Una respuesta a la pregunta del por qué la diferencia, es que el sistema PAL fue desarrollado después de la introducción del televisor de color y por consecuencia no requiere cambios adicionales mientras que en el sistema NTSC la tasa de cuadros varía al añadir color.[59]

Hemos llegado también a la parte de compresión, que en estos casos es necesaria, trabajar con archivos de medios, demanda una gran cantidad tanto en espacio de disco como en RAM, y normalmente al terminar de crear y editar un video, el tamaño del mismo en bytes es enorme. De manera que se hace sumamente necesaria la compresión, y al igual que en la imagen al comprimir hay pérdida de datos, la compresión en el video es lo mismo, permitirnos reducir el tamaño de nuestro archivo a partir de eliminar información redundante. No explicaremos como se realiza esta compresión de datos ya que escapa de los límites de este trabajo.

Hay dos tipos de compresión sin pérdidas y con pérdidas, la primera no existe cambio alguno en el video, aún si hay cambio de formato, mientras que en la segunda el archivo resultante es una aproximación del original. Tenemos para ello tres tipos de códecs de compresión para video:

---

[58]Marcus Weise and Diana Weynand, *How video works*: Elseiver, 2007. pp. 15 -16
[59]Loc. cit.

- Estándares internacionales.
- Propietarios.
- Estándares abiertos.

Las normas internacionales suelen utilizar la tecnología patentada. Los códecs propietarios son creados a partir de una variedad de métodos. Los estándares abiertos son por lo general creados dentro de comunidades de código abierto y son libres para todo el mundo.[60]

Únicamente mencionaremos una lista parcial de los códecs de compresión de video, ya dependerá del lector investigar qué códec le conviene utilizar para su desarrollo.

- H.261.
- MPEG - 1.
- H.263.
- MPEG - 2.
- MPEG - 4
- AVC (H.264)
- WMV
- AVI
- VP6 y VP7
- libTheora
- RealVideo
- Indeo

Sobre programas de edición de video, de forma gratuita para sistemas Windows se cuenta con el *Movie Maker* y para algunas versiones de sistemas OS X se cuenta con *iMovie*. Existen otros programas de edición de video pero algunos

[60]David Austerberry, op. cit., p.87

representan costos muy elevados, de igual manera será cuestión del lector buscar algún software adecuado a sus necesidades para la creación y edición de video.

2.5 SOFTWARE PARA DESARROLLO DE APLICACIONES DE REALIDAD VIRTUAL Y LA AFINIDAD CON GAME ENGINES

Existen muchos programas en el mercado que nos permiten crear entornos virtuales. Cada producto tiene ciertas particularidades que de acuerdo a la naturaleza de nuestro producto nos proveen herramientas de diseño que nos facilitan en mayor grado el poder realizarlas. Por ejemplo, algunos cuentan con mejores herramientas en la parte de luz y color, otras pueden tener mayores prestaciones con respecto al audio o sonido, otras con respecto a la importación de objetos 3D, etcétera.

2.5.1 VRML Y SOFTWARE PARA RV

Antes de continuar, vale la pena dar una pequeña explicación de qué es **VRML** (*Virtual Reality Modeling Language*), el cual es un estándar para ambientes virtuales en internet. Es necesario mencionarlo ya que es el estándar por excelencia para aplicaciones en web. Aunque nuestro desarrollo no está directamente involucrado con este estándar es necesario tenerlo presente.

Los diseños VRML se basan en la interpretación y despliegue mediante programas denominados '*Browser*', los cuales son visualizadores que interpretan el código y a partir de esto presentan el ambiente renderizando las imágenes correspondientes. VRML se puede definir como un lenguaje cuyo texto básico está en código ASCII y sus ambientes generalmente se reconocen por la extensión WRL.[61]

[61] Juan Carlos Parra Márquez,, op. cit., p 66

Mencionaremos algunos de los productos que existen en el mercado, aclarando que pueden existir mas, esta lista es únicamente informativa para aprender básicamente la estructura de los programas para desarrollos RV.[62]

RAYDREAM STUDIO

Es un conjunto de aplicaciones de la empresa *Metacreations.* No es un software complicado y el precio es accesible. Se recomienda para laboratorios escolares, puesto que tiene versiones para Windows y Macintosh. La interfaz está diseñada en modo gráfico y un elemento clave es la funcionalidad ya que incorpora herramientas para elegir, diseñar e incorporar objetos a partir de su librería y además podemos realizar modificaciones en la iluminación, deformaciones, efectos de renderizado, animaciones, etcétera.

3D OPEN SYSTEM

Es una herramienta para desarrollar mundos virtuales dinámicos 3D. Una de las características importantes es que también puede ejecutarse en varias plataformas. Permite crear mundos 3D con visualización en tiempo real e incluye interiores y exteriores 3D, objetos estáticos y móviles, imágenes JPG, luces, color dinámico 3D, efectos, videos AVI, sonidos (MP3, WAV, MID), soporte para documentos HTML. Se pueden crear mundos ilimitados en tamaño y anidar mundos en otros.

V-REALM

Es un paquete que permite crear objetos 3D y mundos para ser vistos mediante el browser *V-Realm* u otros compatibles con el estándar VRML.

Sus características incluyen múltiples editores, texturas, luces, cámaras y editores tanto para objetos individuales como en grupo, soporta formatos JPEG, GIF y RGB. Posee herramientas para manipular objetos 3D tales como

[62]Juan Carlos Parra Márquez, op. cit., pp. 30 -37

transformaciones, re-modelado, despliegues simultáneos de escenas y otros. Algo interesante es que incluye un programa que permite importar archivos desde 3D *Studio, Autocad, TrueType Font* y otros.

INTERNET SPACE BUILDER

Es un editor 3D para diseñadores Web, permite la creación de mundos virtuales y publicarlos en internet. Puede diseñar y editar escenas u objetos 3D usando operaciones booleanas (suma y resta) sobre figuras primitivas. También puede exportar escenas en formato VRML. Pose una galería decente de formas, texturas, pinturas, objetos y vídeos. Permite crear puntos de vista adicionales (cámaras) y posee características avanzadas de mapeo de texturas, color y transparencia. Procesa archivos BMP, GIF, JPG y PNG, además de importar archivos 3DS, DXF y otros.

3 DEM

Permite principalmente crear escenas terrestres en tres dimensiones de baja resolución, animación de vuelo en tiempo real y mundos VRML de variados tamaños y diversas fuentes de datos. 3 DEM utiliza librerías de SGI/Microsoft OpenGL para producir modelos 3D de alta velocidad. Puede renderizar imágenes en 16 o 24 bits de color. La renderización de imágenes puede ser azul-rojo (requeridos para visualización estereoscópica con anáglifos) o proyecciones de color 3D requeridos para gafas de obturación (LCD).

MULTIGEN

Aunque se ha dicho que existen muchos productos para RV y que muchos de los modelos son importados desde sistemas CAD (por ejemplo DXF), hay un número considerable de sistemas de modelamiento usados para crear sus propias formas 3D. Entre ellos podemos destacar a Multigen, que proviene de una de las compañías de mayor reconocimiento y que produce software de modelamiento desde 1986.

Estos productos son usados ampliamente en diversas aplicaciones de simulación en tiempo real para entrenamiento industrial u otros fines. En diversas plataformas tales como SGI, Sony, Nintendo, Sega, Macintosh, PC y otros equipos especiales para 3D en tiempo real.

SUPERSCAPE VRT

Es un sistema profesional para el diseño de mundos 3D interactivo sobre plataforma PC. Estos mundos pueden ser publicados en VRLM en Internet y usar su navegador Viscape, como también navegar con el browser propio Visualizer. De igual manera se pueden crear mundos mediante su versión gratuita 3D-WebMaster. Superscape está compuesto de siete editores que permiten la creación de sus aplicaciones. En estos se pueden desarrollar mundos a partir de objetos 3D, texturas y sonido utilizando editores de formas, imágenes y sonidos, entre otros. A los objetos se les puede asociar propiedades tales como: dinámica, velocidad angular, gravedad y otros, además de incorporar comportamientos mediante el uso del lenguaje interno de Superscape llamado SCL (*Superscape Control Language*). Posee librerías que contienen texturas, objetos y sonidos, los cuales pueden ser utilizados libremente por el diseñador.

2.5.2 ¿POR QUÉ GAME ENGINES?

Como hemos visto, los programas para creación de mundos virtuales ofrecen ciertas características que resultan muy útiles al momento de crear entornos tridimensionales, pero básicamente están desarrollados para trabajar bajo estándares VRML, en cambio los motores para videojuegos, mejor conocidos como *Game Engine* ofrecen las mismas herramientas de desarrollo pero son más accesibles de ser instalados en las plataformas existentes más comunes en el mercado, además de ofrecer portabilidad, escalabilidad, nivel de desarrollo y aplicaciones mejor logradas que con la mayoría de los programas para realidad virtual. ¿Qué significa esto?, un *game engine* es mucho más fácil de usar, están

en continuo desarrollo sobre todo aquellos de pago, en algunos de ellos tenemos libre acceso al código, lo que significa que podemos crear nuestras propias funciones o modificar las ya existentes para nuestra conveniencia, además van a la par también con los avances tanto en tecnología de hardware como los programas externos que utilizan como herramientas.

En el capítulo uno hablamos de manera general de algunas características de los motores RV, a continuación las mencionaremos de manera formal.[63]

1. **Importación de modelos:** capacidad de importar formas 3D de otros programas de modelado.
2. **Librerías: b**ibliotecas de formas geométricas, texturas y en ocasiones figuras compuestas. Lo importante es que se pueden re-utilizar las formas para ambientar el entorno.
3. **Operaciones geométricas:** manipulación de objetos en la escena.
4. **Nivel de detalle:** conocida como LOD (*Level of Detail*) permite la optimización de la visualización de la escena, al intercambiar los niveles de detalle dentro del ambiente.
5. **Animación:** asignación de movimiento a algún objeto dentro del mundo virtual.
6. **Articulación:** permite que los objetos puedan ser organizados en jerarquías.
7. **Detección de colisiones:** identifica cuándo un objeto toca o intercepta a otro.
8. **Propiedades físicas:** asignación de componentes físicos a los objetos como masa, gravedad, fricción, velocidad, etc.
9. **Color y texturización:** nos permite asignar texturas y colores a los elementos y entornos ambientales de la escena 3D.

[63] Juan Carlos Parra Márquez, op. cit., pp. 23 - 25

10. **Fuentes de luz:** definir la luz del entorno, tanto ambiental como elementos individuales.
11. **Incorporación de audio:** importar formatos de sonido y asociar el mismo a objetos o eventos en el mundo virtual.
12. **Lenguajes de programación:** el software de desarrollo dispone de comandos que dicten comportamientos de objetos y permitan leer tanto las entradas como salidas de información.
13. **Manipulación de eventos:** permite activar o desactivar eventos ya sea de comportamientos o interacciones en el entorno virtual.
14. **Configuración de dispositivos múltiples:** permite incorporar hardware externo.
15. **Mundos paralelos:** generación de ambientes virtuales constituidos por sub-mundos.
16. **Conectividad en red: p**ermite que el mundo virtual pueda ser utilizado en una red.
17. **Exportación en VRML:** permite exportar los modelos para ser visualizados en entornos web.

Con respecto a la última característica, no necesariamente nuestros modelos deben ser exportados en VRML para que puedan ser visualizados en internet, se consideraba obligatorio para conservar el estándar, pero ahora contamos con herramientas eficaces que pueden hacer que nuestro entorno hecho en algún motor de videojuegos pueda ser visualizado en internet sin ningún problema, incluso los mismos motores de videojuegos proveen dichas herramientas por si fuera necesaria esa exportación.

Puede leerse un poco aventurado, pero los motores para videojuego, exceptuando la última, cuentan con todas las características antes descritas, incluso con las mismas o mayores prestaciones, además que en algunos casos, dependiendo la naturaleza del motor, ofrece más características.

Como leímos en el primer capítulo, podemos escoger el motor que más se adecúe a las necesidades de nuestro proyecto, si lo queremos enfocado en una o múltiples plataformas, si será un recorrido en primera o tercera persona,cuál es la envergadura de procesamiento, si necesitamos modelos con un gran detalle o basta una programación y texturización básica.

Desde 1998 se ha creado un concurso anual llamado, *Front Line Awards,* que está enfocado a la excelencia e innovación en herramientas para desarrollo de videojuegos. ¿Por qué hacemos referencia a *Front Line Awards*? Hoy en día es muy difícil hacer una lista completa de todos los motores de videojuegos existentes en el mercado, hay gratuitos y bajo licencia, hay para una o varias plataformas, se pueden enfocar en ciertos aspectos como iluminación o contar con herramientas que cubren todos los ángulos. Además de que no existe algún estudio exhaustivo que valore todos los motores.

Front Line Awards ha sido reconocido por sus estudios de mercado sobre herramientas destinadas a la creación y apoyo para desarrollos de videojuegos. Tiene más de una década (el primer premio fue en 1998) recopilando y haciendo mención de los mejores programas durante el año. Incluso sus premios han sido reconocidos por los mismos desarrolladores, tanto de los programadores de los game engines como de los mismos creadores de videojuegos.

> Por más de una década *Game Developer* ha señalado a lo mejor en herramientas de desarrollo de juegos con su evento anual *Front Line Awards*. Aquí tomamos un momento para rendir honor a los productos que son los más efectivos para ayudar a los desarrolladores a realizar un gran trabajo. Ya sea en las trincheras o al borde de la línea, éstas son las herramientas en que confían artistas, diseñadores e ingenieros. Aunque el negocio de la creación de estas herramientas puede ser muy competitivo con el próximo gran producto siempre rondando a la vuelta de la esquina, nosotros queremos rendir un homenaje especial con nuestro *Hall of Fame* Award a los

productos que han hecho un impacto duradero año tras año en la industria de los videojuegos (y que además opta por ganar en su categoría específica). [64]

Por lo anterior la mejor recopilación que se puede hacer es de los últimos tres años, 2008, 2009 y 2010 del Front Line Awards, mencionando los productos finalistas y el ganador en la categoría de game engine.[65]

Ganadores del Front Line Awards (2008 al 2010)		
2008 Finalistas	**2009 Finalistas**	**2010 Finalistas**
Unity 2.1 Unity Technologies CryEngine 2 Crytek Gamebryo 2.5 Emergent Game Tech. Source Protocol 14 Valve Corporation	CryEngine 3 Crytek Gamebryo LightSpeed Emergent Game Tech. Source 1.6.1.6 Valve Unity 2.5.1 Unity Technologies	CryEngine 3 Crytek Gamebryo LightSpeed Emergent Game Tech. Unity 3 Unity Technologies Vision Engine 8 Valve
2008 Ganador	**2009 Ganador**	**2010 Ganador**
Torque Game Engine Advanced 1.7.1 *GarageGames*	*Unreal Engine 3* *Epic Games Inc.*	*Unreal Engine 3* *Epic Games Inc.*

Figura 2.10
GameEngines finalistas

[64]Jeffrey Fleming, "Game developer Front Line Awards 08" en GameDeveloper, January 2009, Volume 16, Number 1. p. 7 (www.gdmag.com)

[65]GameDeveloperMagazine, http://www.gdmag.com/frontlineawards/hof.html (ví: 6 de junio de 2011)

Unreal Engine es un gran motor, pero al parecer no es muy accesible al público en general, en su página no se puede ver el precio del producto en primera instancia y hay que referirse a soporte o área de contactos para cualquier tipo de información referente al asunto en cuestión.[66] Se estima un precio de 700 dólares para finales de 2007.

De manera que nos queda la familia *Garage Games*, con su motor *Torque Engine Advanced*. Pero no utilizaremos dicho motor para el desarrollo de nuestro producto, sino a su antecesor que es el motor *Torque Game Engine*. El cual fue la fase beta de tan gran motor.

Algo importante de *Garage Games* es que también tiene una sección enfocada a la educación la cual ha sido muy bien aceptada en universidades importantes en todo el mundo, por ejemplo: *University of Nevada* en Reno, Nevada; *University of New Mexico* (Albuquerque, Nuevo México), *Rochester Institute of Technology* (Rochester New York), *ITT Technical Institute* (Norfolk, Vermont), *Southern Alberta Institute of Technology* (Calgary, Canadá), *Technische Fachhochschule Berlin* (Berlín, Alemania), Universidad de los Andes (Bogotá, Colombia), *Universidade Anhembi Morumbi* (Sao Paulo, Brasil), y *Polytechnic University of Puerto Rico (*San Juan, Puerto Rico).

También convence el hecho que tiene un soporte muy amplio para varias tecnologías actuales ya sea tanto en arte como para plataformas, además cuenta con el respaldo de marcas como Ubisoft, NASA, Sony, Microsoft, por mencionar algunas y está asociado con AMD, Autodesk, Nvidia, Intel, Sun y más.

Por lo anterior se ha decidido enfocar el desarrollo en este motor de videojuego, para mayor información al respecto, podemos dirigirnos a la página de

---

[66]Unreal http://www.unrealtechnology.com/ (ví: 20 de Octubre de 2009)

Garage Games,[67] donde encontraremos mayor información sobre el motor, los diferentes productos con los que cuentas. Tendremos que dirigirnos al área de documentación y tutoriales o entrar al foro de Garage Games para leer preguntas y opiniones de toda una comunidad que diariamente ofrece su conocimiento a los nuevos usuarios que se acercan a esta tecnología. Cabe mencionar que el idioma es en inglés.

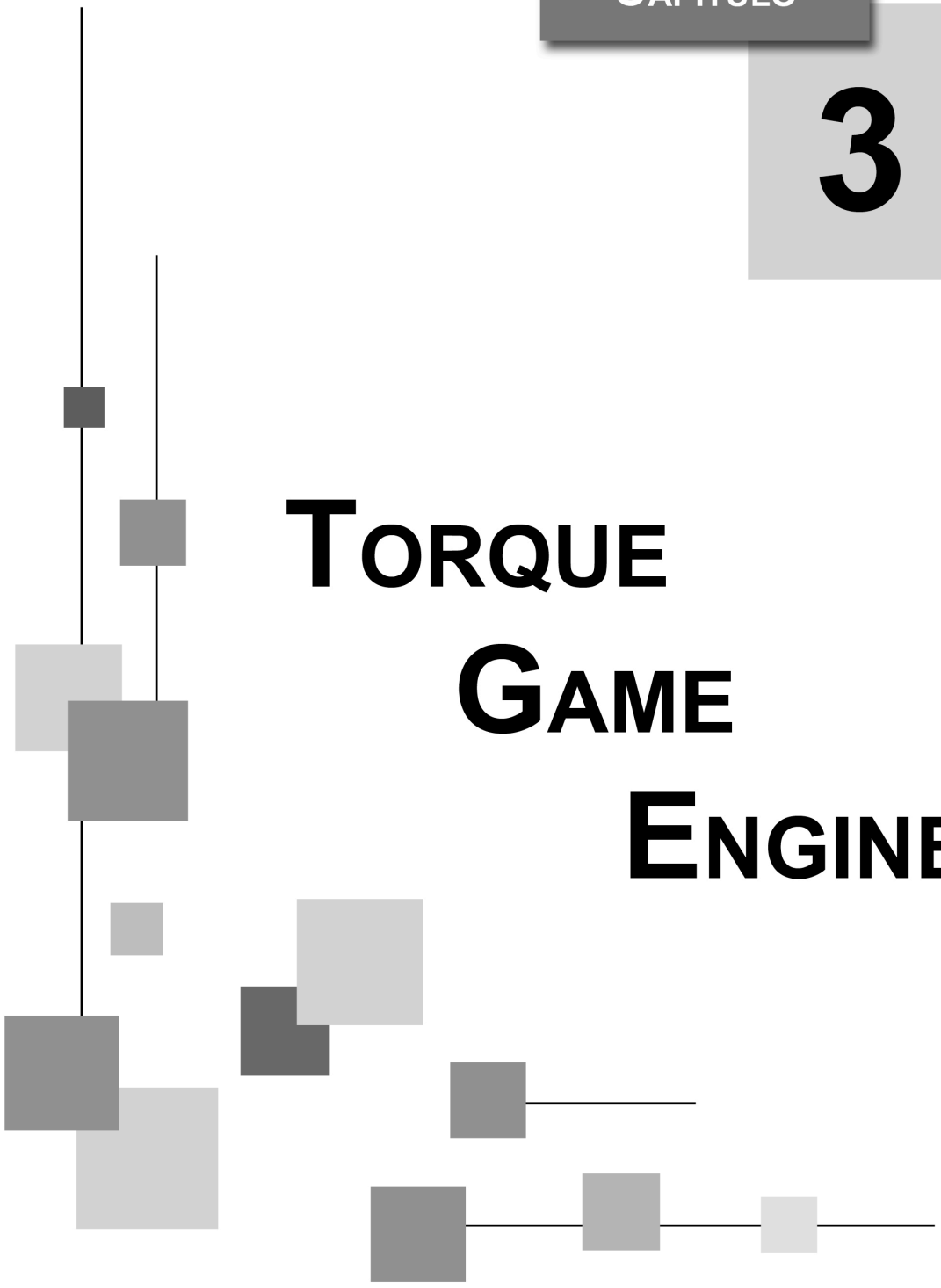
En este capítulo hablamos sobre programas que nos pueden ayudar al momento de escribir nuestro código, herramientas para modelado 3D, algunos conceptos básicos sobre diseño, audio y video, repasamos algunas características importantes sobre software para creación de mundos virtuales y la semejanza que existe con las nuevas tecnologías enfocadas al desarrollo de videojuegos. Teniendo en mente lo anterior, ya contamos con un panorama general de las herramientas necesarias que utilizaremos, conceptos con los que tendremos que convivir a lo largo de nuestro proyecto y lo más importante, vamos concibiendo un pensamiento maduro para enfocar correctamente nuestro producto y las herramientas a utilizar.

[67]Garage Games http://www.garagegames.com/ (ví: 20 de Octubre de 2009)

CAPÍTULO

3

TORQUE GAME ENGINE



TORQUE GAME ENGINE

3.1 ¿QUÉ ES TGE?

"Torque es una solución completa para la creación de juegos, no es solo otro motor de render. Los motores de render son fáciles. Sólo haces una cosa y la haces bien. Éste empieza por brindarnos soporte multiplataforma (Mac/Linux/PC/otras plataformas), multijugadores (2 a 256+ jugadores), un servidor maestro, programación de física, manejo de recursos, rendereo, partículas, GUI (interfaces gráficas de usuario), terrenos, interiores, mallas, animaciones, multitrayectorias, LOD (nivel de detalle), cielos, agua, niebla, efecto de llamaradas, sonidos, herramientas WYSIWYG (editor de interfaces, editor de mundos, constructor de terrenos), etc., etc., todos corriendo fácil y eficientemente de maneja conjunta... uf!!!."[68]

TGE son las siglas para *Torque Game Engine*, cabe mencionar que esta distribución ya está fuera del mercado (enero 2009) y duró mientras se consolidaba la versión completa de lo que hoy conocemos como *Torque Game Engine Advanced* (TGEA) y Torque 3D, pero básicamente el nivel de programación es el mismo a diferencia que las carencias y errores que se encontraron a lo largo de la fase Beta, ahora se encuentran resueltas, sin olvidar

[68] Rick Overman, GarageGames Duggan, Michael. *Torque for teens*: Thomson, 2008. p. 25

que las innovaciones también han sido añadidas a las dos nuevas versiones del TGE. A lo largo de los siguientes capítulos haremos referencia exclusivamente al motor TGE.

El motor de *Torque Game Engine* tiene las siguientes características:[69]

- Provee un scripting robusto (programación), código de red, un motor de edición y creación de mundos y creación de GUIs. El código puede ser compilado en plataformas Windows, Macintosh y Linux.
- Utiliza su propio lenguaje de programación *TorqueScript*. Si tenemos un mínimo de experiencia en lenguaje de programación orientado a objetos no tendremos problemas al momento de programar los códigos para nuestro desarrollo. *TorqueScript* es fácil de aprender.
- Permite publicar directamente nuestros desarrollos en PC, Macintosh y con algunas modificaciones en Linux.
- Está preparado con todo el código necesario para compilar, jugar y capturar elementos en el mundo a partir de gráficos 3D, creación de GUIs, añadir sonidos, y administrar hardware de entrada/salida (I/O).
- Las características de *Torque* con respecto a la creación de terrenos es realmente rápida, automáticamente los podemos construir incluso con LOD (*Level of Detail*).
- Soporta la importación de modelos 3D en formatos tipo DTS y DIF. Los modelos DTS tipo jugador, pueden ser animados usando animación bípeda o animación por medio de cadena de huesos (*skeletal animation*). Usada también para vehículos.
- Entre otras características, el motor interpreta sombras, mapeo de ambientes, neblina volumétrica, y muchos más efectos.

[69] Duggan, Michael. *Torque for teens*: Thomson, 2008. pp. 28-31

- Puede ser utilizado para crear ambientes de un solo usuario o en ambientes bajo servidores o en red enfocados para múltiples usuarios. Está basado en el estándar de arquitectura cliente-servidor.
- Es sumamente eficiente, tanto en memoria de procesador como en ancho de banda, al momento de ser ejecutado bajo red. *Torque* usa bloques de datos estáticos (comprendidos bajo términos de *TorqueScript* como *datablocks*) para datos comunes y compresión en red, además se agregan algoritmos para mantener bajo el tráfico del ancho de banda.
- Torque está designado bajo simulación para manejo de eventos, usando ciclos separados tanto para el cliente como para el servidor.

Hemos mencionado sólo unas cuantas características que engloban todo lo que puede hacer el motor TGE, con lo anterior podemos darnos una idea de los alcances y sus capacidades.

Una de las características que no debemos perder de vista es la parte del *Torque Game Engine Scripting Language*, pero mejor conocida como *TorqueScript*. Hay que considerar que, aunque *TorqueScript* es un lenguaje propio, los desarrolladores deberán estar familiarizados con algunos conceptos básicos de programación. Obviamente no se pide ser experto en lenguaje C/C++ o lenguajes orientados a objetos, pero sí tener idea de cómo se programa en ellos y tener presente algunos conceptos básicos; no obstante, aunque se aprenderá un nuevo lenguaje de programación, tener idea de cómo se manejan dichos lenguajes ayudará en gran manera para familiarizarse rápidamente *TorqueScript*.

Recordando un poco sobre el por qué decidimos usar *Torque Game Engine* como la base principal de nuestro desarrollo, debemos considerar que *GarageGames®*[70] está disponible para casi todas las plataformas (ya sea para desarrollos 3D y 2D) y ha sido una de las plataformas mas licenciadas en los

[70] http://www.garagegames.com/company (ví: 18 de julio de 2011)

últimos años, además de ofrecer soporte a mas de 200 universidades (*Torque Education License*) en todo el mundo en áreas de ciencias de la computación y programas de diseño de juegos.[71] Esto habla por sí mismo, demostrando que *GarageGames* está en constante desarrollo y no sólo se enfoca en el área comercial, sino también brinda opciones para la educación.

TGE ES UNA HERRAMIENTA PARA DESARROLLOS MULTIPLATAFORMA. ADEMÁS CUENTA CON SU PROPIO LENGUAJE PARA PROGRAMAR E IMPLEMENTAR NUESTROS PRODUCTOS. LENGUAJE SUMAMENTE FÁCIL, AUN PARA AQUELLOS QUE SE INICIAN EN LA PROGRAMACIÓN EN TGE POR PRIMERA VEZ.

[71] http://www.garagegames.com/education (ví: 18 de julio de 2011)

3.2 CONCEPTOS QUE UTILIZAREMOS EN *TORQUE*

Ya hemos visto a grandes rasgos los alcances del motor TGE, ahora es momento de visualizar lo que haremos nosotros con ese motor, para ello tendremos que acostumbrarnos a algunos conceptos de trabajo como interior (*interior*), forma (*shape*), bloque de datos (*datablocks*). Pero no hay que casarnos con la terminología aquí usada, aunque los términos que utilicemos pueden llegar a ser parecidos en otras ramas de la industria de desarrollos visuales no significa que sean erróneos. Normalmente suelen ser híbridos de conceptos que al final terminan por referirse al concepto.

3.2.1 FORMAS (SHAPES)

Una forma (shape) o en términos TGE conocidos como objetos DTS, son modelos (objetos) creados a partir de un polígono base ya sea un plano, esfera, rectángulo, etc., por medio de programas de animación y modelado 3D como *LightWave, 3ds Max, Alias, Maya*, entre otros, los cuales ya hemos mencionado anteriormente.

Estos modelos pueden contar con las siguientes características:

- Animación por Esqueleto
- Múltiples texturas
- Texturas animadas
- Múltiples niveles de detalle
- Componentes translúcidos o transparentes
- Múltiples tipos de colisión (‘*Collision Detection’*)

- Y algunos otros[72]

Shape - Forma

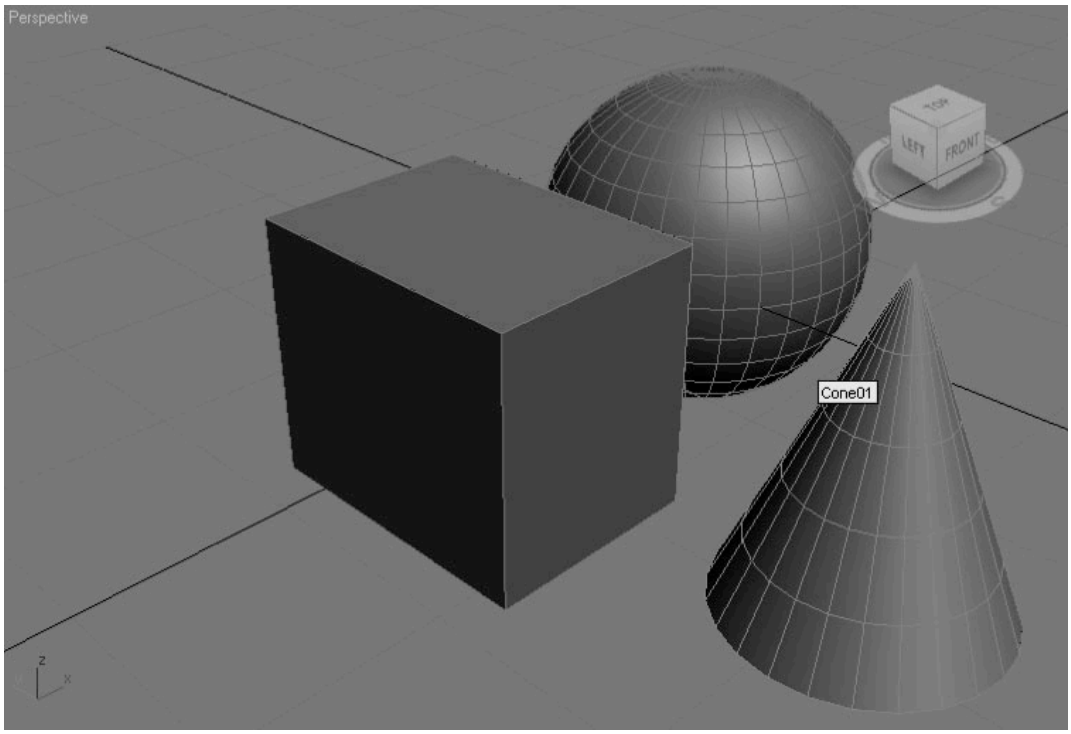


Figura 3.1
Shape – Forma

Dynamix Threespace Shape (DTS) entendida como Forma Dinámica Tridimensional, es una de las dos formas que se van a emplear para la creación de objetos dentro de nuestro entorno. Incluso DTS es la extensión para este tipo de forma, por ejemplo jugador.dts.

Los objetos DTS nos ayudan a exportar objetos no estructurales desde nuestros programas de modelado, por ejemplo, personas, vehículos, objetos de relleno, como árboles y demás formas que darán vida a nuestro ambiente.

[72] Maurina, Edwar F. "The Game Programmer's Guide to Torque", CG Press 2006, p. 16

Shape DTS

Se pueden observar dos *shapes*: *jugador.dts // arbol.dts*

Figura 3.2

Shape DTS

El siguiente modelo utilizado en TGE para las formas (*shapes*) son lo interiores (*interiors*) mejor conocidos como DIF en la terminología de TGE.

3.2.2 INTERIORES (*INTERIORS*)

Los interiores son modelos creados a partir de geometría convexa. Esta clase se conoce como *InteriorInstance* y es utilizada para crear modelos estructurales o que representen algún tipo de estructura, ya sean puentes, edificios, paredes o algún otro tipo de estructura parecida. Y así como dts es la extensión para su tipo de *shape*, de igual manera tenemos que la extensión para los interiores es dif, por ejemplo: interior.dif.

Shape DIF

Se pueden observar dos *shapes: jugador.dts // iglesia.dif*

Figura 3.3

Shape DIF

La pregunta que puede surgir inmediatamente es: ¿por qué no construir interiores con un shape tipo dts? La respuesta es que sí se pueden construir interiores tipo dts pero, estrictamente hablando, ya no sería un *shape* tipo interior; sería un *shape* únicamente. Para entender mejor este detalle hay que aclarar otro término que es el de colisión (*collision*).

La colisión se puede explicar como el proceso de detectar cuando dos o más objetos (obviamente en el mismo entorno) se 'tocan'. Este término se conoce

como COLDET o *collision detection*, y TGE puede soportar un ilimitado número de COLDET para modelos. Lo anterior significa que también tendremos total control sobre el COLDET para nuestro uso.

Entendido un poco lo que es el COLDET (*collision*) podemos retomar la pregunta antes expuesta: ¿por qué no construir interiores con un *shape* tipo *dts*? Ya quedó claro que podemos construir interiores tipo *dts,* pero por obvias razones ya no se consideraría interior, sino un *shape* más dentro de nuestro entorno. El hecho de que se haya creado la clase InteriorInstance es para resolver ciertos conflictos que se generaban cuando se trataba de crear *shapes* de grandes dimensiones o modelos de geometría compleja. De manera que esta nueva forma de *shape* logró resolver los siguientes problemas:[73]

- **Eficiente detección de colisiones:** permitiendo de esta manera controlar eficazmente los procesos del CPU en tiempo real.
- **Mejoras en visibilidad de profundidad:** el uso de portales ('*portals*') se ve mejorado de gran manera, permitiendo optimizar el uso de *render* gráfico ya que en habitaciones o terrenos que el usuario no puede ver, dicha información no sea enviada al procesador de gráficos para su rendereo o interpretación.
- **Eficaz uso de la Iluminación:** se mejora notablemente el proceso de calcular la iluminación y las sombras creadas en el entorno de acuerdo a la presencia del usuario en el recorrido.

[73] Ibid., pp. 17-18

3.2.3 GEOMETRÍA CONVEXA Y CÓNCAVA

Aunque es un tema breve es necesario darle el énfasis justo, ya que es un punto esencial, sumamente importante al momento en que estemos creando nuestros interiores.

En TGE todas nuestras colisiones deben ser convexas, no cóncavas. Al principio puede costar un poco de trabajo identificar estos tipos de geometrías, incluso puede acarrear enormes frustraciones al artista o desarrollador al momento de exportar sus modelos, ya que dentro del entorno generarán problemas y no podrán identificar el error que en ocasiones puede ser tan simple como una geometría cóncava.

Entonces, al tener una superficie cóncava es seguro que exportaremos una mala colisión de nuestro modelo. Para evitar este error, seguiremos unas simples reglas dadas por Edward Maurina en su guía para *Torque*:[74]

- Si alguna de las líneas de nuestro modelo, extendida o alargada infinitamente en ambas direcciones, pasa, corta o atraviesa a través del interior de nuestro modelo, nuestra geometría es cóncava y seguramente generaremos una malla de colisión con errores.
- La alternativa mas ambigua es buscar hoyos o hendiduras en nuestro modelo, si nuestra geometría presenta alguna concavidad, tenemos nuevamente un modelo cóncavo.

En la industria de la imagen tridimensional el término *'convex brush'* es muy común, significa que tenemos un modelo o geometría convexa. La utilidad de tener geometrías convexas es que facilita la combinación de varias estructuras, permitiendo unir estos modelos y crear un interior sin ningún problema.

---

[74] Ibid., pp. 18-19

Otra característica importante a la que deberemos recurrir en nuestros proyectos es al LOD (*level of detail*). LOD es la complejidad en la cual se va a mostrar o representar nuestro modelo tridimensional, de acuerdo a la vista actual que tiene el usuario hacia el modelo. La complejidad de la estructura del modelo se verá afectada ya sea aumentando o disminuyendo sus detalles en respuesta a la cámara (vista), dependiendo del movimiento del usuario, ya sea acercándose o alejándose del objeto respectivamente.

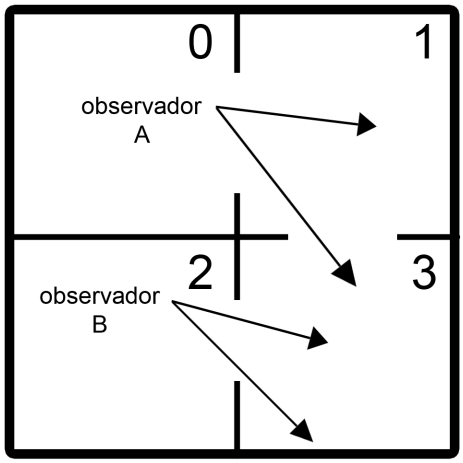
Es momento oportuno de hablar de los portales ('*portals*'). El término '*portal*' es ampliamente usado en la industria de este tipo de desarrollos y, como mencionamos, TGE soporta el *render* o interpretación de interiores. Es decir, que esta interpretación dividirá nuestro interior en sectores.

A continuación tenemos el ejemplo básico de un interior que tiene cuatro habitaciones. Las habitaciones han sido numeradas del 0 al 3 y dicho interior posee 3 puertas.

En el cuarto 0, hay un observador con un cierto ángulo de visión. Podemos ver que su rango de visión (ángulo de cámara) incluye tanto el cuarto 1 como el cuarto 3, de manera que tendrá que ser rendereado el cuarto 0, 1 y 3. En cambio, para el observador B, su ángulo de cámara comprende únicamente el cuarto 3, por lo cual no será necesario renderear los cuartos 0 y 1, solamente los cuartos 2 y 3.

En el caso de no utilizar portales, los cuatro cuartos del interior serían rendereados, y esto implicaría más uso del procesamiento de gráficos en nuestro desarrollo.

Interior con portales



Inspirado en el gráfico Interior con portales. Ibid., p. 20

Figura 3.4

Portal

Una característica importante de la cual estará lleno nuestro desarrollo son los 'callbacks', también es un término utilizado en la industria y éste es un método de consola (normalmente scripts) que está asociado a un objeto de nuestro entorno y que es utilizado por el motor para obtener una respuesta del objeto de acuerdo a algún evento específico.[75]

Por ejemplo, si necesitamos interactuar con alguno de nuestros shapes en nuestro desarrollo, ya sea una puerta, un objeto, etc. Podemos utilizar un callback *onCollision()* o *onPickup(),* y obtendríamos una respuesta inmediata ya sea al momento en que tocamos el objeto o tomamos el objeto respectivamente.

Algo que no puede faltar es el sonido ambiental, ya sea 2D o 3D que también es un concepto utilizado. No se necesita abordar mucho esta idea, ya que el término es obvio. El sonido ambiental puede tener dos fuentes, puede ser 2D, que es un sonido que no aparenta origen y cuando es reproducido sonará igual en

[75] Ibid., p. 21

los dispositivos de salida. Mientras que el sonido 3D, contará al menos con un origen asociado a el, y dicho sonido se verá atenuado o se incrementará de acuerdo a la posición en la que uno se encuentre del origen del sonido. El sonido también afectará los dispositivos de salida de audio que tengamos conectados a nuestro equipo, de manera que si nosotros vamos caminando del lado izquierdo del origen del sonido, nuestra bocina izquierda tendrá mas volumen en su sonido que la bocina derecha, todo esto de acuerdo a la posición relativa en la que nos encontremos en el ambiente.

ALGO QUE DA VIDA A NUESTROS DESARROLLOS ES EL AUDIO. DIFERENCIAR ADECUADAMENTE LOS TIPOS DE SONIDO QUE UTILICEMOS SERÁ DE VITAL IMPORTANCIA PARA LA INMERSIÓN DEL USUARIO. UN 2D PUEDE USARSE EN MENUS Y PANTALLAS DE INICIO, UN 3D PARA LA AMBIENTACIÓN DENTRO DEL RECORRIDO.

3.2.4 MISIONES (MISSIONS)

Este es un término particular de TGE. Seguramente en nuestro desarrollo tendremos más de un mundo o un ambiente que contendrá elementos distintivos de cada escenario. Cada uno de esos ambientes se le conoce como 'nivel'. Por lo regular en *Torque* existirá más de un nivel y cada uno de esos niveles se les conoce con el nombre de misión (*mission*).[76]

Para tener más claro este concepto, será necesario visualizar qué sucede en las misiones. Los archivos que 'contendrán' las misiones, tendrán como extensión .mis y se encuentran dentro de una carpeta específica llamada '*data*' (aspecto que tendremos oportunidad de ver mas adelante). Estos archivos

[76] Ibid., p. 22

contendrán los scripts de creación y posición del contenido de dicha misión, esto significa que podemos pensar en una misión como una colección de elementos que contendrá cierto ambiente o nivel y que se crearán bajo la petición del motor. Estrictamente hablando, la misión es lo que inicialmente se carga al inicio de nuestro recorrido, en ese archivo está contenida la información del terreno, el sol, el cielo, etcétera. Subsecuente a eso, son llamados los scripts de posicionamiento de elementos distintivos de nuestras misiones, como puede ser el caso de árboles, edificios, objetos decorativos, entre otros.

TGE como muchos otros motores en el mercado, es un simulador. Tenemos varios tipos de simuladores, y *Torque* es un simulador por eventos. Esto significa que todas las acciones ejecutadas por el motor serán causadas por algún tipo de evento. TGE maneja una gran variedad de eventos pero estos eventos son puestos en espera y son procesados por el motor en el orden en el cual fueron ocurriendo.[77]

Hay algo muy importante que debemos tener presente, cada objeto que se coloca en la misión cuenta con un '*handle*', es decir, con un identificador que cuenta con dos etiquetas: un número y un nombre.

- **Número:** del lado del cliente nos indica qué objeto es en la misión (de una lista de objetos) y del lado del servidor es el ID del objeto.
- **Nombre:** cuando es colocado un objeto en la misión el nombre por omisión es '*null*', esto significa que no se ha asignado un nombre al objeto. Colocar un nombre al objeto puede ser opcional, pero es muy útil asignárselo (sobre todo para propósitos de *scripting*).

[77] Ibid., p. 23

Handle

Se observan tres *shapes* con sus respectivos *Handle's* (id: nombre), dos de ellos han sido etiquetados (por el programador) con un nombre: árbol y automóvil, mientras que el otro aparece con el nombre *null* (asignado por el motor).

Figura 3.5
Handle

No hay que perder detalle de estos elementos del objeto ya que serán muy ventajosos al momento de querer interactuar con ellos. Trabajar ya sea con el ID o con el nombre del objeto será de gran ayuda cuando estemos programado características especiales entre el usuario y los objetos, ya que permitirán una rápida identificación del mismo sin la necesidad de tanto código para ello.

EL *HANDLE* NO ES UN ADORNO. AUNQUE SE CREA DE MANERA AUTOMÁTICA, SIN LA NECESIDAD QUE EL USUARIO O PROGRAMADOR TENGA QUE ESCRIBIR ALGO, ES IMPORTANTE IDENTIFICAR EL ID DEL OBJETO Y MÁS IMPORTANTE AÚN ETIQUETARLO CON UN NOMBRE SI ESTAMOS SEGUROS QUE LO USAREMOS DESPUÉS.

3.3 ARQUITECTURA CLIENTE-SERVIDOR EN TGE (*CLIENT-SERVER ARCHITECTURE*)

En todo desarrollo es importante saber el tipo de arquitectura que necesitamos o el tipo de arquitectura con la cual vamos a trabajar, pues ella dará la pauta o afectará las decisiones que tomemos.

En nuestro contexto, en el desarrollo de nuestro recorrido virtual, podemos entender el término *arquitectura* como "la organización del sistema de nuestro producto", es decir, "qué parte de nuestro motor hace qué tarea".[78]

En principio, TGE implementa una arquitectura cliente-servidor, lo que significa que una parte del motor actúa como un tipo de controlador (el servidor), mientras que otra parte del motor actuará como control (el cliente).

Durante la ejecución, el cliente y servidor pueden coexistir en el mismo ejecutable, ejecutarse por separado en la misma máquina o ejecutarse por separado en diferentes máquinas conectadas en red. El estándar de cliente-servidor indica que hay solo un servidor, mientras que se puede tener más de un cliente conectado a éste.

El por qué utilizar una arquitectura cliente-servidor para un recorrido virtual es fácil de responder, podemos dar los siguientes argumentos:[79]

- **Primero**: ofrece de una manera significativa y comprensible la labor de dividir tareas y recursos.

[78] Ibid., p. 24
[79] Ibid., p. 25

- **Segundo**: una arquitectura cliente-servidor es adecuada para desarrollos tanto para un solo cliente como para clientes múltiples.
- **Tercero**: en el caso de clientes múltiples, la arquitectura puede ser escalable para N clientes, donde N puede ser escalada para 128 clientes o más.

TGE implementa el modelo cliente-servidor en un solo ejecutable. Esto significa que no importa dónde corra nuestro desarrollo, ya sea en una sola máquina (*standalone*) o a través de una red, siempre contendrá los dos elementos, cliente y servidor.

El presente trabajo está enfocado en programar un producto *standalone* (cliente y servidor en la misma máquina), de manera que no se abordarán temas referentes a desarrollos sobre servidor.

3.3.1 GHOSTS, CONTROL OBJECTS Y SCOPING

Los términos anteriores no deben ser traducidos porque son conceptos de uso para *Torque*. Pero se detallarán a continuación para familiarizarnos con ellos.

Cuando se ejecuta un recorrido, ya sea de forma local o por red, todos los objetos son creados en el servidor y algunos de ellos son duplicados del lado del cliente. Estos objetos duplicados se les llaman *ghosts*. La duplicación de los objetos como *ghosts* del lado del cliente es controlado por el *scoping*. Cada cliente que se une al servidor debe definir un objeto único de control (*control object*). Por lo regular, este tipo de *control object* es un tipo de '*avatar*' (figura bípeda, un vehículo, o algún objeto más), incluso puede ser la misma cámara, el cual va a

identificar al cliente o usuario conectado al servidor. De cualquier manera, independientemente del objeto que sea, será el encargado del *scoping*.[80]

Ghosts y Control Object

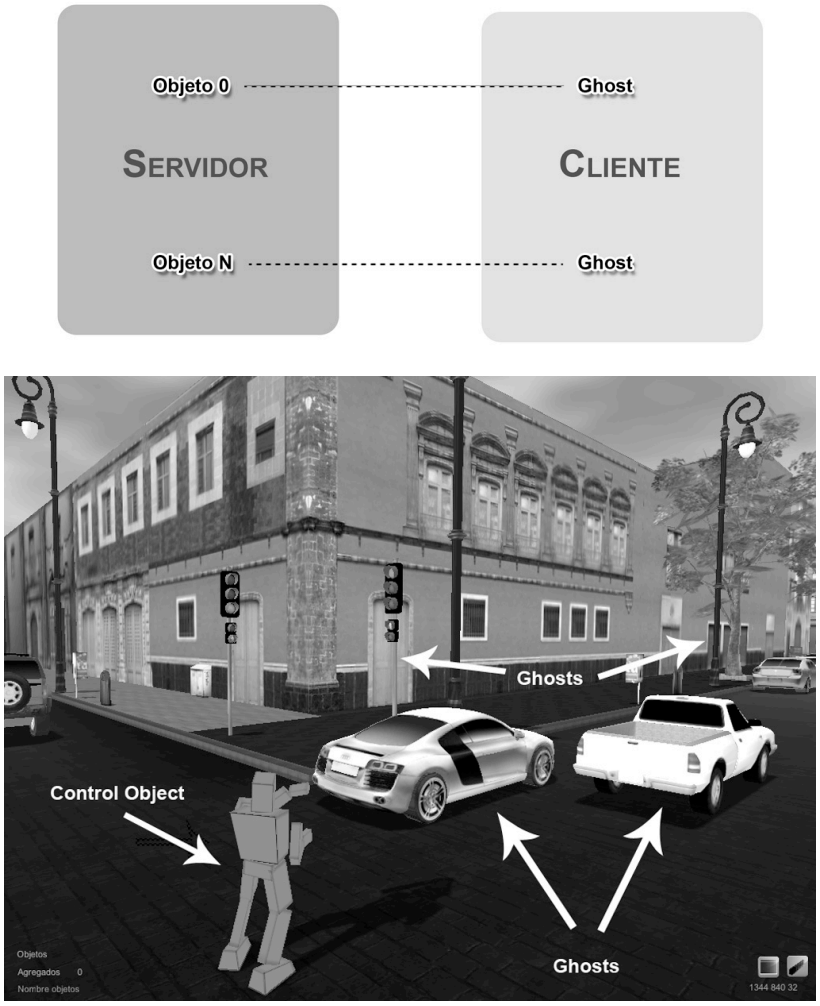


Figura 3.6
Ghosts y Control Object

Aún no hemos terminado la definición de *scoping*, pero antes de abordar dicho término era necesario entender qué es el *control object*. Para TGE, el

[80] Ibid., p. 28

scoping es la forma, el acto de determinar cuáles objetos en el entorno son visibles, audibles, y algún otro requerimiento que sea necesario presentarle al *control object* para que pueda interactuar correctamente en el ambiente. Estos objetos serán los *ghosts* que se desplegarán del lado del cliente para el control object y se mantendrán mientras sean necesarios. No contar con un *control object* implicaría que nuestro ambiente no pueda ser interpretado.

3.3.2 DATABLOCKS

En la categoría de objetos tenemos unos de especial interés denominados '*datablocks*'. Este tipo de objetos se diferencian de los demás debido a:[81]

- Todos los '*datablocks*' son duplicados del servidor a cada cliente (si es el caso de ser multi-cliente).
- Todos los '*datablocks*' XYZ (posición) con un ID 123 (identificador) enviados desde el servidor ofrecen la garantía de tener el mismo XYZ y el mismo ID en el o los clientes.
- Los '*datablocks'* son transmitidos al cliente desde un inicio y no son actualizados después de esto, de manera que son objetos estáticos.
- Los '*datablocks'* tienen propiedades especiales que pueden ser utilizadas por medio de scripts.
- Debido a que los '*datablocks'* son controlados por el servidor, podemos cuidar que sean manipulados, previniendo, así, la modificación de sus propiedades por parte del cliente.

[81] Ibid., p. 29

3.3.3 JERARQUÍA SIM (SIM HIERARCHY)

Como ya hemos mencionado, TGE es un simulador por eventos (*event-driven simulator*) y su estructura organizacional está definida por varias clases, de las cuales la clase principal o la clase base es el *SimObject*, (*Simulation Object*). Esta clase y sus sucesoras forman lo que conocemos como *Sim Hierarchy*, podemos jerarquizarlas de la siguiente manera:[82]

SIMOBJECT:
Es la raíz de las clases para todos los objetos que se simularán.

SIMSET Y SIMGROUP:
Son dos tipos de clases contenedoras. *SimGroup* actúa como clase base para los *GuiControls* y *ScriptGroup*.

SCRIPTOBJECT Y SCRIPTGROUP:
Son dos clases utilizadas para crear clases *script*. Estas clases son especiales ya que nos ayudarán para asociar características y métodos con las clases *script*. De esta manera podemos perfectamente seccionar nuestros *scripts* sin problemas.

SCENEOBJECT:
Es la clase raíz para todos los objetos que son incluidos en el ambiente y añaden los conceptos de posición, render y colisión.

GAMEBASE:
Es la clase raíz para todos los objetos que son colocados en las misiones.

[82] Ibid., p. 31

SHAPEBASE, SHAPEBASEDATA Y CHILDREN:

ShapeBase y *Children* son clases que se utilizan para desplegar modelos. Éstos son usados para representar pequeños objetos del entorno, como vehículos, objetos para usar (*pick-ups*) incluso el mismo usuario como actor en la escena. Soportan geometría y características complejas, así como un número ilimitado de colisiones.

TSSTATIC:

Esta es una clase para renderear objetos ligeros, clase que no incorpora ninguna de las características de la clase *ShapeBase*. Es utilizada para objetos no interactivos, por ejemplo, para agregar adornos o detalles en la escena.

INTERIORS:

Esta clase nos permite desplegar cualquier tipo de estructura en nuestro ambiente. Sus características ya han sido mencionadas en el punto 3.2.2.

SPECIAL EFFECTS:

Como su nombre lo indica, esta clase nos permitirá incorporar una variada gama de efectos especiales en nuestro ambiente, incluyendo audio, visuales y de carácter físicos, como reacciones a la gravedad, al agua o de luces.

3.3.4 I/O TGE (DISPOSITIVOS ENTRADA/SALIDA EN TGE)

Los dispositivos por omisión de entrada para TGE es el ratón (*mouse*) y el teclado, con un poco más de código podemos implementar dispositivos como *gamepads*, *joysticks* u otro dispositivo.

La forma general en cómo se procesan las entradas para TGE es:

- El sistema operativo procesa las entradas y las pasa al TGE *Platform Layer*.

- El TGE Platform Layer identifica y categoriza el tipo de entrada y la pasa al recorrido.
- El recorrido procesa las entradas, si dicha entrada no se encuentra definida simplemente es ignorada.

Pero dicha manera general no es suficiente si queremos agregar más opciones de entrada por teclado o mouse a nuestro desarrollo además de las programadas por omisión, de manera que detallaremos un poco más el proceso de cómo suceden los eventos de entrada para el TGE, pero ahora directamente dentro del desarrollo:[83]

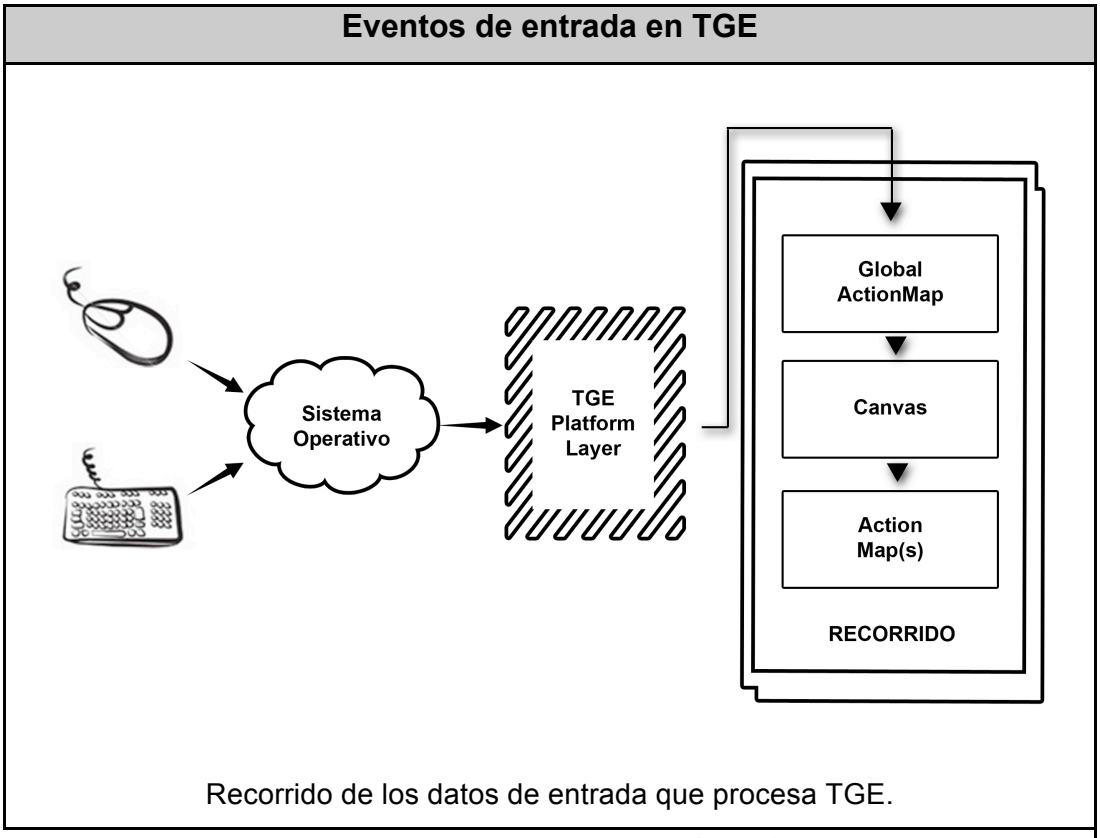


Figura 3.7

Arquitectura de entrada en TGE

[83] Ibid., 32

1. *GlobalActionMap* recibe el dato del *TGE Platform Layer*. Si esta entrada no tiene alguna función asociada pasa al GUI (interfaz gráfica del usuario), específicamente al *Canvas*.
2. El *Canvas* intenta procesar el dato de entrada que ha recibido, pero si no existe algún control del GUI que la procese, dicha entrada seguirá adelante.
3. Finalmente la entrada pasa a alguna entidad activa (no global) del ActionMaps para ser procesada, si no existe algún evento mapeado a esta entrada, el dato es desechado.

Hemos mencionado un término nuevo que es el *ActionMaps*, que es una clase especial dedicada a capturar y redirigir las entradas. Hay dos tipos de *ActionMap*:[84]

- ***GlobalActionMap*:** es el padre para los procesos de entrada y de reemplazo. Esta acción no debería ser expulsada de la pila de procesos del desarrollo.
- ***ActionMap*:** esta acción toma una prioridad baja a comparación de los otros procesos; puede ser ingresada o expulsada de la pila de procesos de acuerdo a las necesidades del desarrollo.

NUESTRO RECORRIDO NECESITARÁ CIERTOS TIPOS DE MAPEO DE LAS ENTRADAS RECIBIDAS ENTRE EL TECLADO Y EL RATÓN, YA SEA PARA LOS MOVIMIENTOS DEL USUARIO Y SUS COMPORTAMIENTOS. INCLUSO UNA MISMA TECLA PUEDE SER USADA PARA DIFERENTES ACCIONES DEPENDIENDO EL EVENTO O MISIÓN.

Con esto hemos cubierto a grandes rasgos los puntos principales de los conceptos que pondremos en práctica a lo largo de nuestro recorrido. Ahora es el

[84] Ibid., p. 33

momento de conocer las herramientas gráficas con las que cuenta TGE para la creación de nuestro desarrollo.

3.4 CONJUNTO DE HERRAMIENTAS EN TGE

En TGE tenemos dos herramientas gráficas básicas el WorldEditor y el GuiEditor, las cuales nos servirán para la construcción de nuestro ambiente y para las pantallas que desplegaremos al cliente respectivamente. Estas son interfaces visuales que tenemos a nuestra disposición, las cuales nos ayudarán a modificar elementos gráficos de nuestro proyecto sin la necesidad de escribir código para ello.

Con el WorldEditor podemos editar cualquier aspecto de nuestro entorno, como el clima, el tipo de terreno, colocar objetos, incluso disponer de eventos especiales como puede ser efectos de luz y otras características que pueda contener nuestro recorrido, todo ello sin la necesidad de salir del WorldEditor. Mientras que el GuiEditor nos permitirá crear las pantallas de usuario, como los menús, organizar los botones, crear controles personalizados para los usuarios, etc. Todo esto lo podremos verificar sin la necesidad de salir del GuiEditor o compilar de nuevo toda la aplicación.

Estas herramientas son de vital importancia por dos razones:

1. **Primero**: nos permite crear ya sea interfaces gráficas personalizables para el usuario o ir creando nuestras misiones de manera visual sin la necesidad de escribir alguna línea de código.
2. **Segundo**: con lo anterior ahorramos tiempo y mejoramos al máximo la correcta programación de estos elementos, ya que la creación de esta parte de código corre a cargo de TGE, estando seguros que no habrá error alguno y estará optimizada dicha programación.

Realmente es una ayuda excepcional, por tal motivo valdrá la pena hablar un poco de estas herramientas con las que contamos en TGE.

3.4.1 WORLDEDITOR

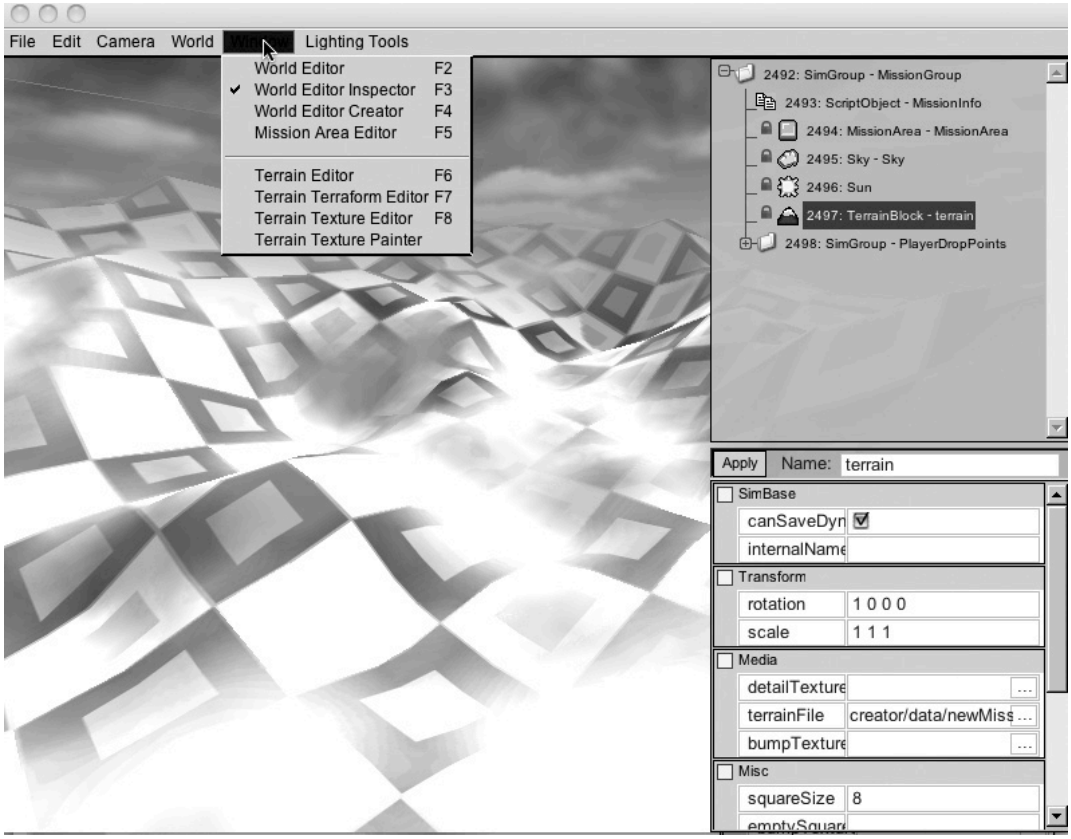
El WorldEditor es una herramienta de TGE que incluye 8 sub-herramientas adicionales, de manera que podemos decir que tenemos 8 sub-editores mas para cuestiones específicas en la edición de nuestro ambiente. Los componentes más importantes son el WorldEditor y el TerrainEditor.[85]

El WorldEditor y sus sub-editores son una herramienta muy poderosa para la construcción de misiones y definición del ambiente por medio de la colocación de objetos o estructuras. Con este instrumento damos forma a nuestros recorridos sin la necesidad de preocuparnos en el código. Podemos crear, colocar, mover, dimensionar, rotar objetos en nuestro entorno, así como modificar las propiedades de dichos objetos.

El TerrainEditor y sus sub-editores (que son parte de las 8 herramientas de WorldEditor) nos permitirán un completo control del paisaje. TGE esta definido para empezar la creación de misiones a partir de grandes áreas al aire libre (por así decirlo) y por medio de sus sub-editores ir dándole forma al panorama. Por ejemplo, podemos crear elevaciones (cordilleras de montañas) con un par de clicks o formar valles de la misma manera. Podemos combinar hasta seis texturas para darle vida a nuestros terrenos.

[85] Duggan, Michael, op. cit., p. 38

WorldEditor



Se observa la interfaz de trabajo del WorldEditor. Se cuenta con un menú exclusivo para seleccionar los sub-editores y del lado derecho tenemos las secciones que contiene tanto el árbol de elementos que integran la mission como el de propiedades para cada uno de dichos elementos.

Figura 3.8

WorldEditor

A continuación describimos a grandes rasgos que hace cada una de esas herramientas:

Herramienta (Tecla de acceso)	Características
WorldEditor (F11)	Compuesto por 8 sub-editores, cada uno permitirá modificar y salvar varios aspectos específicos de las misiones. Usado para modificar misiones existentes o crear nuevas
WorldEditor Manipulator (F2)	Nos permite rotar, mover, escalar objetos que sean colocados en la misión.
WorldEditor Inspector (F3)	Aunado a todas las características del WordEditor, el Inspector nos permitirá modificar las propiedades del objeto seleccionado.
WorldEditor Creator (F4)	Aunado a todas las características del WordEditor, el Creator nos permite colocar nuevos objetos en la misión.
MissionArea Editor (F5)	Nos permite ajustar los límites de nuestra misión y observar los detalles con diferentes puntos de vista de la misión actual.
TerrainEditor (F6)	Con esta herramienta podemos manipular el terreno (forma) por medio del mouse.
TerrainTerraform Editor (F7)	Aunado a todas las características del TerrainEditor, esta herramienta permite cargar 'imágenes' (formas) de terrenos (archivos de terreno) y aplicar algoritmos de generación y filtros para la creación de los mismos.
TerrainTexture Editor (F8)	Aunado a todas las características del TerrainEditor, esta herramienta nos permite seleccionar texturas y aplicarlas por medio de algoritmos que determinarán la mezcla de texturas y su colocación en el terreno.
TerrainTexture Painter (menu → Terrain Texture Painter)	Aunado a todas las características del TerrainEditor, nos permite seleccionar y aplicar hasta seis texturas diferentes al terreno.

Figura 3.9

Sub-Editores del WorldEditor

3.4.2 GUIEDITOR

Recordando un poco, GUI (Graphical User Interface). Es la suma de todos los elementos de control (ventanas, botones, campos de texto, etc.) que son usados para que el usuario pueda interactuar con el desarrollo. Por ejemplo, el programar un botón para iniciar el recorrido, alguna ventana emergente para brindar información sobre algún evento que este sucediendo o una lista que pueda proveer información del sistema y poder controlarlo a nuestro gusto, como estos controles, tenemos un sin fin de posibles funciones para programar y así brindarle mayor control al usuario.[86]

Una definición mas concreta del GUI es la que nos brinda Finney: *"Típicamente es la combinación de gráficos y scripts que contienen la apariencia visual del producto y acepta comandos de entrada provenientes del usuario"*[87]

Uno de los problemas más comunes es poder programar un GUI decente, y en parte se debe a que no se tiene herramientas orientadas para este fin, de manera que tenemos dos problemas, el diseñar un GUI decente y después programar ese GUI. TGE nos proporciona toda una herramienta avanzada para la creación de GUI's de forma visual.

---

[86] GarageGames http://docs.garagegames.com/tge/official/index.html?content/documentation/Artists/Missions/Overview.html (ví: 26 de julio de 2011)

[87] Kenneth C. Finney. "3D Game Programming All in One", Thomson, 2004. p. 19

GUI

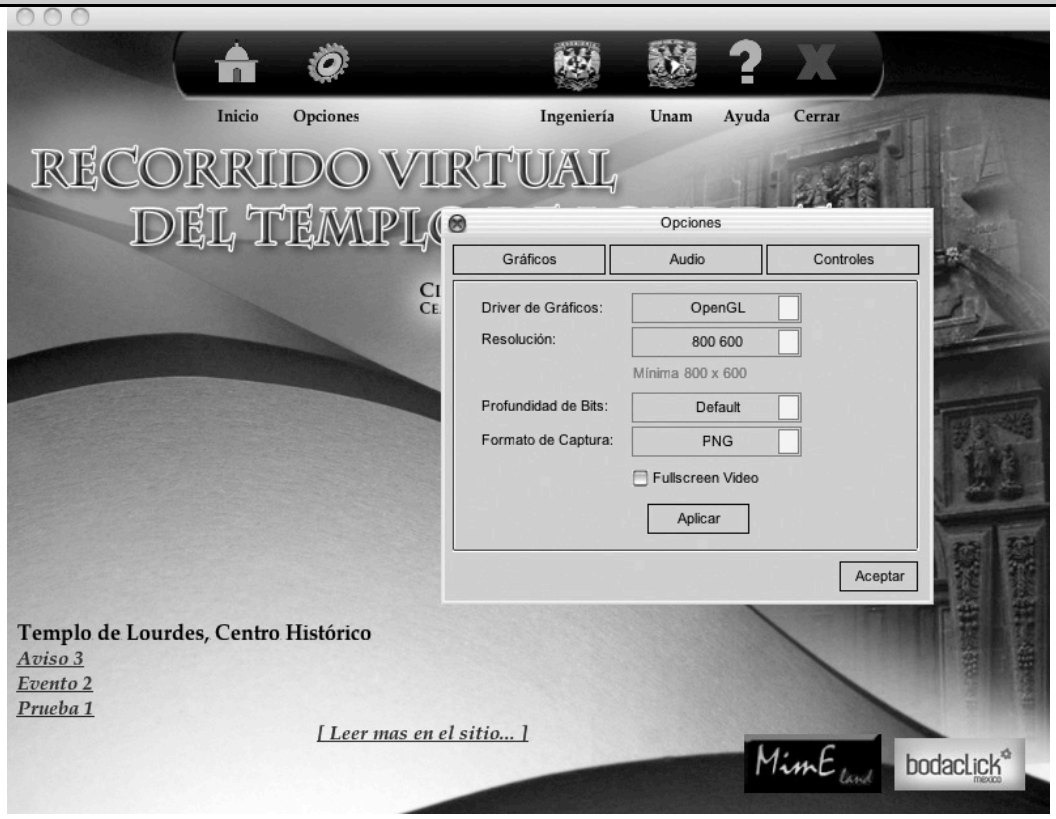


Figura 3.10

Ejemplo de GUI

Al decir que el GUIEditor de TGE es una herramienta avanzada significa que no solo es visual, si no que permite la creación de interfaces de usuario de una manera mas dinámica, permitiendo el arrastre y colocación de elementos visuales directamente en la zona de trabajo del GUIEditor; completando la herramienta contamos con una sección que permite un rápido acceso a las propiedades de los elementos visuales, un menú lateral de campos vacíos para detallar las propiedades, especificar variables, definir funciones, etc., del elemento visual de una manera mas fácil, pero no basta con lo anterior, estas interfaces gráficas son guardadas como archivos tipo scripts, así que podemos editarlos y

agregar funciones extras que escapan a las bondades del GUIEditor, de una forma mas cómoda por medio de nuestro editor de programación.

La interfaz gráfica del GUIEditor se divide en tres secciones:

1. **CONTENTEDITOR**
   Es el lugar donde diseñaremos y colocaremos los elementos gráficos de nuestros GUI's.
2. **CONTENTTREE**
   Aquí se visualiza la jerarquía de los controles que van conformando nuestros GUI's.
3. **GUI INSPECTOR**
   Muestra los campos de las propiedades asociadas al elemento gráfico seleccionado.

Aunque no es parte integral de la interfaz gráfica del GUIEditor el **Toolbar** es la barra que se encuentra en la parte superior y que muestra funciones asociadas y elementos de control que disponemos para la elaboración de nuestros GUI's.

Sin una experiencia mínima en la creación de interfaces de usuario, podemos caer en el error de tener un GUI inaccesible o difícil de entender para el usuario. Aún con manuales y opciones de ayuda directamente en la interfaz, si no somos capaces de poder brindar una interfaz intuitiva y fácil de usar, seguramente nuestro producto por más bueno que sea, terminará por ser mal calificado y desechado por el usuario final.

NO ES FACIL DISEÑAR INTERFACES DE USUARIO, ES COMÚN CAER EN EL ERROR DE LLENAR LA PANTALLA CON 'MIL Y UN MENUS', CREYENDO QUE LA HACEMOS LLAMATIVA. EL CLIENTE ESPERA TENER A LA MANO LAS FUNCIONES MAS COMUNES, NO TODAS LAS INSTRUCCIONES DEL PRODUCTO ENFRENTE DE EL.

TGE no es capaz por si mismo de renderizar las imágenes, en lugar de hacerlo por si mismo, utiliza el API ('Application Programming Interface') gráfico de OpenGL. Esto es una gran ventaja ya que no es necesario programar nosotros mismos los controladores necesarios para la interpretación de las imágenes utilizadas. TGE incluye un conjunto de librerías que son extensiones de OpenGL para estos propósitos.

Las librerías del GUI de TGE controlan la interfaz de usuario y han sido diseñadas exclusivamente para ese propósito. El **Canvas** es el padre o el contenedor de los objetos gráficos ACTIVOS en la jerarquía actual del GUI. Se encarga de comunicar las entradas provenientes del teclado o del 'mouse', de actualizar las regiones necesarias y controlar los eventos del cursor y obviamente de mantener el refresco de imagen, es decir, cuando es necesario dibujar el siguiente cuadro en la pantalla. [88] Así como el canvas es el padre de todos los objetos gráficos activos en la pantalla en cierto momento, la clase **Control** es el padre de las clases de todos los controles del sistema del GUI.

[88] Ibid., p. 25

3.5 TORQUESCRIPT

Hemos estado haciendo mención varias veces de lo importante que es TorqueScript para nuestro desarrollo, de manera que es momento para hablar de las bondades que nos ofrece.

Antes de empezar hay que hacer hincapié en que, no es completamente necesario dominar ciertos lenguajes de programación, como C/C++ o lenguajes orientados a objetos, pero si valdría la pena tener conocimiento de su manejo ya que permitirá asimilar de una manera mucho mas fácil y rápida el uso del TorqueScript. Pero aún, si no se estuviera familiarizado con estos conceptos, TorqueScript resulta un lenguaje de programación fácil de aprender.

TorqueScript es un lenguaje flexible y cuenta con características y capacidades muy similares a los lenguajes de programación orientada a objetos así como sintaxis muy parecida a la de C/C++.[89]

Es necesario dar una reseña de lo que es TorqueScript antes de explicar detalles sobre su programación y alcances:[90]

SINTIPO (TYPELESSNESS)

Las variables en TGE pueden convertirse en lo que se decida mientras sea necesario o en caso contrario son usadas sin cambio alguno. Las variables pueden ser globales cuentan con el símbolo $ como prefijo, mientras que las locales se les antepone el símbolo %. Las variables locales únicamente pueden ser accesadas dentro de la subrutina o función, mientras que las globales son accesibles donde sea. Es decir, puedo tener una variable $caminar que me indica

---

89 Kenneth C. Finney. “Advanced 3D Game Programming All in One”, Thomson, 2005, p. 4

90 Ibid., pp. 22-23

que el usuario podrá caminar a cierta velocidad en cualquier parte del recorrido, pero puedo tener el mismo nombre de variable dentro de una función llamada entrarCuarto, en ese caso mi variable estaría definida de la siguiente manera %caminar y podría tener una valor diferente al de caminar global, siendo accesible únicamente dentro de esa función, de manera que cuando saliera de esa función (entrarCuarto), caminar tomaría el valor global nuevamente.

SINCASO (CASELESSNESS):

TorqueScript ignora la forma en la que esta escrita una variable o función cuando la interpreta. Es decir, no importa que la variable o función se escriba %correr, %CORRER o %Correr, así mismo para el caso de las funciones. Aunque hay esquemas a seguir al momento de escribir este tipo de datos pero se verán más adelante.

FINAL DE SENTENCIA (STATEMENT TERMINATION):

El final de sentencia será con el símbolo; (punto y coma o 'semicolon').

OPERADORES:

Al igual que en lenguajes como C/C++ o java, la mayoría de ellos son los operadores comunes.

ESTRUCTURAS DE CONTROL:

Como C/C++ o Java, TorqueScript provee las estructuras: *if-then-else, for, while y switch.*

FUNCIONES:

Podremos crear funciones con la capacidad de regresar valores y los argumentos que se pasen a la función pueden ser mediante variables o referencias. Cuando definimos una función debe empezar con *function.*

OBJETOS:

TGE provee una amplia gamma de objetos y tipos de objetos para TorqueScript y a partir de ellos podemos crear ScriptsObjects.

PAQUETES:

Los paquetes, como en muchos otros lenguajes de programación, son usados por TorqueScript para establecer funciones dinámicas y polimorfismo (usados ampliamente en lenguajes orientados a objetos). Es decir, es la capacidad de que varias clases derivadas de alguna otra utilicen el mismo método de manera diferente, dependiendo de que paquete este en proceso. De manera que podemos activar y desactivar paquetes de acuerdo a las necesidades que se estén estableciendo.

Con este pequeño repaso de los alcances básicos de TorqueScript, salta la pregunta, ¿Es necesario utilizar scripts en nuestros desarrollos? La verdad es que no. El motor TGE esta programado completamente en C++ y la mayoría del desarrollo que se elabore estará totalmente programado en TorqueScript o al menos un 99%.

Esta pregunta salta a la mente porque es necesario diferenciar entre los scripts y el TorqueScript. Los scripts son pequeños pedazos de código o líneas de instrucciones, normalmente fuera del programa base o el programa principal y que son almacenados en texto plano o escritos directamente sobre la consola y son interpretados en tiempo real para su ejecución, a diferencia de un programa que debe ser compilado para que se ejecute. Y no es sumamente necesario que estén en el mismo lenguaje, ya que pueden estar escritos en otro lenguaje para aumentar las funcionalidades.

De modo que una cosa son los scripts que podemos manejar dentro de TGE y otra muy diferente es TorqueScript, que es el lenguaje de programación enfocado para desarrollos con TGE.

Retomando la pregunta, ya nos dimos cuenta que no es necesario usar scripts para nuestro desarrollo, pero al momento de usar scripts podremos hacer mas eficiente nuestra programación. Además de que es más fácil escribir los scripts para comprobar eventos que estar programando y compilando cada vez que hagamos un cambio en nuestros programas.

En nuestro caso, los scripts serán convenientes porque usarán componentes pre-existentes en nuestro motor y ayudaran a completar nuevas tareas. Es decir, usaremos scripts para accesar a características de nuestro motor y poder aumentar la experiencia virtual de nuestro desarrollo.[91]

Hemos conocido los alcances de los scripts e incluso sabemos que podemos hacer uso de ellos con nuestro motor TGE, pero en nuestro caso nos enfocaremos más en el uso de TorqueScript para la programación de nuestro producto. Solo que fue necesario aclarar puntos ya que, aunque TorqueScript no son scripts íntegramente, tiene todas las funcionalidades de los scripts mas las ventajas del uso de Torque como plataforma. De manera que en ciertos casos pueden actuar como scripts, aún estando escritos en lenguaje TorqueScript.

Ahora que conocemos las generalidades de TorqueScript, avancemos un poco mas en lo que vendrá siendo realmente el esqueleto o la parte medular de nuestro desarrollo. La programación.

3.5.1 PROGRAMANDO EN TORQUESCRIPT

Torque script tiene como referencia para la estructura y programación el lenguaje C/C++ y lenguajes orientados a objetos, como Java. Estos archivos

[91] Maurina, op. cit., p. 98

deberán tener como extensión '.cs' para que puedan ser compilados por el motor TGE.

El fin de este apartado no es salir programando en TorqueScript, eso ya compete a cada uno el acercarse al lenguaje de programación y adecuar las necesidades a su proyecto. El objetivo es:

- Entender y leer fácilmente el código.
- Prevenir bugs.
- Buena arquitectura y modularidad.
- Compatibilidad de plataforma y compilador.

3.5.2 ARQUITECTURA EN TORQUESCRIPT

Daremos un vistazo general de los principales puntos que debemos tener presentes. El orden estándar de cómo se conforma un archivo de código es el siguiente:

- Cabecera de comentarios
- #includes
- #defines
- Estructuras locales
- Variables locales estáticas
- Prototipos de funciones estáticas
- Funciones estáticas y globales
- Clase de métodos
- Métodos de consola

Todos los archivos deberían, digo deberían porque normalmente no se hace o en alguno se nos escapa escribir ese par de líneas al principio de nuestro archivo de código, como decía, se debería empezar con un 'comentario de cabecera'. El formato que se recomienda es el siguiente, aunque no es obligación el hacerlo:

```
//-----------------------------------------------------------------------------
// Torque Game Engine
// Copyright (C) GarageGames.com, Inc.
// Empresa desarrolladora:
// Participantes o desarrolladores del script:
//-----------------------------------------------------------------------------
```

Estructurar el código de la misma manera, le dice al lector que el programa ha sido elaborado por un programador profesional, que no está descubriendo en cada pedazo de código, la forma de organizar un archivo por primera vez. Es importante notar que el código producido por GarageGames tiene un aspecto profesional y es un buen ejemplo de cómo empezar a escribir nuestros programas. Un profesional en consonancia también hace sus códigos de manera fácil para encontrar las partes principales de un módulo de código cuando es necesario que alguien más tenga que leerlo.

Con respecto a las variables debemos evitar exponerlas públicamente. Si la clase necesita exponer alguna variable, entonces una función de acceso debe ser creada. Realmente una función de acceso hace nada, pero regresa una variable, la cual debe ser integrada desde el código resultante, esta forma será mas rápida (pero no necesariamente mas pequeña).

Hay que evitar también en lo más posible, el uso de variables globales. En dichas variables es demasiado fácil que su valor sea cambiado por alguna otra clase o función y no nos demos cuenta del cambio y esto hace que el 'debug' sea extremadamente difícil. Si existe verdaderamente la necesidad de crear una variable global, una clase especial se debe crear para envolver el valor y acceder así al valor, ésta es una manera mas conveniente de trabajar, ya que así es mas fácil de ver cuando el valor ha sido cambiando. Éstas variables deben ser declaradas como privadas, de esta manera protegemos que sean utilizadas por alguien mas que no sea la función de acceso.

LA FALTA DE EXPERIENCIA EN LA PROGRAMACIÓN NOS HACE CAER EN EL ERROR COMÚN DE MANEJAR VARIABLES GLOBALES. SON MUY COMODAS Y NOS FACILITAN LA VIDA, PERO SE VIVE AL FILO DE LA NAVAJA, YA QUE SI POR ALGUNA RAZÓN UN VALOR NO ESPERADO CAE EN ESA VARIABLE, PUEDE GENERAR UN ERROR GRAVE Y NOS SERÁ DIFICIL DESCUBRIR QUE DICHO ERROR ES GENERADO POR LA VARIABLE.

Con respecto a las funciones deben utilizarse únicamente dentro de un solo archivo (esto no incluye los métodos de una clase), además deben ser declaradas como estáticas para prevenir que no sean de tipo pública.

Comentarios

Es mejor utilizar el comentario de tipo // en lugar de la notación /* */. Probablemente el estilo de comentario /* */, puede conducir a la confusión cuando tengamos comentarios anidados. Además, desde el /* hasta el */ puede abarcar varias líneas y suele pasar que resulta difícil vincular los puntos inicial y final del código comentado (un error común es pensar que dos comentarios diferentes son en realidad un solo comentario).

En grupos de funciones, podemos agregar comentarios para los bloques de la siguiente manera:

```
//-----------------------------------------------------------------------------
//   Este grupo de funciones ejecutan las operaciones de render
//   para un objeto
//-----------------------------------------------------------------------------
```

Marcamos los grupos de funciones heredadas:

```
//-----------------------------------------------------------------------------
//   Metodos derivados SceneObject
//-----------------------------------------------------------------------------

     // interface de transformacion
     void setTransform(const MatrixF&);

     // Interface de render
     void renderObject(SceneState*, SceneRenderImage*);
     bool prepRenderImage(SceneState*, const U32 stateKey,
                          const U32 startZone,
                          const bool modifyBaseZoneState = false);
```

Incluso en el ejemplo anterior vemos cómo poder cortar nuestro código cuando la línea puede llegar a ser mas grande que la vista de nuestro visor de código; en este caso es conveniente hacer nuestros cortes después del símbolo de coma (,) de esta forma no cortamos de tajo la idea que estamos leyendo.

También es conveniente preceder cada función en el código por un comentario que especifique la función y qué es lo que hace:

```
//---------------------------------------------------------------------------
// Function name:  MyObj::myMethod
// Summary:        This is what my method does.
//---------------------------------------------------------------------------
```

Se han de preguntar por qué en esta parte dejo los comentarios en inglés; debemos entender algo importante, después de todo, al final no solo programadores en español leerán nuestro código y pues parece que se ha llegado a la idea que el idioma universal es el inglés, así que también deberemos hacer nuestros comentarios en inglés, de esta manera estamos dándole universalidad a nuestro código.

El propósito del comentario es permitir al lector localizar rápidamente el inicio y final de las funciones en el archivo del documento, así como explicar que hace cada una de ellas. Es asombroso lo que hace una o dos frases con una descripción de la función, crean un código muy accesible y nos permiten ahorrarnos un año en la lectura del código o un poco más si el mismo carece de una estructura en su elaboración.

Tabuladores

Nunca se debe utilizar espaciado tipo tabulador en cualquiera de nuestros archivos de código. El tabulador depende de la aplicación o la impresora utilizada y casi siempre resulta en una mala alineación de nuestro código. Es nuestra responsabilidad asegurarnos que nuestro editor de código no introduzca espacios de tabulación en el código que estamos escribiendo.

Aún así, debemos entender que los espacios de tabulación no siempre son los mismos, difieren en espaciado, pueden ser 4, 7, 10 o un número diferente de espaciado, por ejemplo: 1 Tab = 3 espacios. Si queremos usar el tabulador como espaciado, debemos asegurarnos que el espaciado debe estar definido con una

inserción de 3 espacios. Esta inserción de tres espacios es catalogada como el tabulador estándar en la escritura de códigos.

Declaración de variables

La consideración más importante al momento de nombrar nuestras variables es que el nombre debe ser preciso, que describa completamente y sin ambigüedades lo que nuestra variable representa

Las variables deben tener un solo propósito en el programa. Como ejemplo, no hay que usar una variable 'i' como un índice, y después usemos la misma variable para calcular el número de estrellas en la galaxia.

No debemos usar tipos intrínsecos, ¿a qué me refiero con esto?, los tipos intrínsecos son los tipos nativos de un lenguaje de programación. Los tipos de variable se sustituyen para permitir la compilación.

usar	se puede pero no se recomienda
F32	float
F64	double
S32	int
S16	short
S8	char
U32	unsigned int
U16	unsigned short
U8	unsigned char

Durante la programación de nuestros scripts, lo entenderemos mejor.

Todas las variables, se anotarán de la forma 'lowerCamelCase' (empiezan con letra en minúscula y las demás palabras en la variable empiezan con una letra mayúscula, un ejemplo: asiEsCamelCase). Tampoco hay que utilizar

guión bajo (_) para separar las palabras en los nombres de las variables. En pocas palabras, las variables locales no requieren un prefijo, pero si necesita la letra inicial ser en minúscula.

Declaración de Funciones

Las funciones se declaran en “lowerCamelCase”

```
function exampleFunction()
{
  …
}
```

Las funciones pueden ser locales o globales. Una función que nada mas es utilizada dentro de un archivo de código, es declarada como local. Una función que es llamada desde otro archivo, o fuera del archivo donde ha sido programada, debe declararse como global en la cabecera del archivo y debe tener la primera letra de su nombre en mayúscula para distinguirla de las funciones locales.

```
// buen nombre para una función global
getCurrentTime()
{
}

// buen nombre para una función local (estática)
static getNextInterior()
{
}
```

Paréntesis y Sangría

Los paréntesis deben tratar de cerrarse en la misma línea.

```
if ( flag )
  {
    for ( int i=0; i<10; i++ )
    {
      // algo de código
    }
    // algo de código
  }
  else
  {
    // algo de código
  }
```

Con respecto a las tabulaciones, deben de respetar un cierto criterio de espaciado (4, 7, 10), aunque esta parte ya la estudiamos en la parte de 'tabuladores', localizada párrafos arriba.

En las declaraciones de tipo 'for', 'if' ó 'while', dichas declaraciones debemos escribirlas en su propia línea. Ésta hace mucho más fácil de rastrear y depurar nuestro código y ayuda a dar un criterio más amplio para darnos cuenta si nuestra declaración se ha ejecutado.

```
// no se recomienda
if ( obscureCondition == true )  obscureVar += strangeVar;

// esta es una forma correcta
if ( obscureCondition == true )
    ObscureVar += strangeVar;
```

Warnings de Compilador

Aunque muchas veces solo son avisos, debemos procurar deshacernos de las advertencias (‘warnings’) que el compilador nos mande sobre nuestro código.

Advertencia: debemos tener en cuenta que muchas veces los compiladores, o la configuración del compilador, sobre todo los motores multiplataforma, producen ‘*warnings’* sin una aparente causa; al momento no ocasionan conflictos y creemos que podemos dejarlos de lado, y aunque estemos completamente seguros que nuestras líneas de código están bien escritas**,** aún así, no podemos dejar sin atención el ‘warning’. Es recomendable preguntar a personas versadas en el tema para que nos ayuden a eliminar el *warning*.

Un ejemplo de cómo probar y eliminar *warnings*

Signed/Unsigned warning

A menudo en nuestro código, accidentalmente podemos mezclar enteros con y sin signo, recordemos que en *TorqueScript* no es necesario declarar el tipo de variable que estamos usando. Por ejemplo el container.size() devuelve un entero sin signo, mientras sea comparado con un ‘int’, lo anterior no causa problema hasta que el ‘int’ sea un número negativo.

```
void insertAt( Container container, int location, int value )
    {
        if (container.size() > location)
        {
            return; // can't insert
        } else {
            container[location] = value;
        }
    }
```

Si ‘location’ fuera un número negativo (probablemente -1), haremos que nuestra prueba falle. Aunque podemos solucionar esta dificultad: cambiemos el parámetro de ‘location’, convirtiéndolo a ‘unsigned int’ (entero sin signo).

Después aplicaremos pruebas para asegurarnos que ‘location’ nunca tomará un valor negativo, ya teniendo en consideración esto, con confianza podemos hacer a un lado los *warnings* con respecto a nuestro probable entero negativo.

La primera solución nos fuerza a corregir la advertencia, o en el peor de los casos, movemos el problema hacía otra parte del código. La otra solución es mejor o peor, dependiendo del punto de vista del programador.

Con respecto a lo anterior, dentro de las ventajas de *Torque*, lo que se aconseja es utilizar los tipos propios de Torque, que ya estudiamos en párrafos pasados, en este caso serían U32 y S32. Aunque esta no es una opción en el caso de librerías externas, pero es de gran ayuda en varias situaciones.

Ejemplo de posibles errores de asignación.

```
// Este código muestra posibles errores de asignación (warnings)

    S32 a;
    if ( (a=size) )
    {
      // hace algo
    }

    // Podemos hacer esto:

    S32 a;
    if ( (a=size) != 0)
```

```
{
  // hace algo
}

// o incluso mejor:

S32 a = size

if ( a )                    // o tambien: if (a != 0)
{
  // hace algo
}
```

Acabamos de comprobar que para llegar al mismo punto hay una gran variedad de modelos y maneras de organizar nuestro código, incluso la manera de programar varía con respecto a cada uno de nosotros.

Debemos tener claro qué es lo que estamos haciendo y hacia dónde queremos llegar; sobre todo tener nuestro código lo más limpio y depurado posible, así evitamos líneas, líneas de código inservibles, y mejor aún, obtendremos el beneplácito de quien lea nuestro código. La práctica nos dará la experiencia para mejorar día con día.

Hay que considerar que lo que acabamos de leer no es exclusivo para TorqueScript, sino que son las reglas básicas que un buen programador debe seguir y tener en consideración, no sólo al escribir código para TGE, sino también para cualquiera de sus desarrollos.

Con esto damos por terminado la visión general de lo que es *Torque Game Engine* (TGE), así como la descripción de las herramientas que tenemos para la creación de nuestros recorridos, y cómo es que *TorqueScript* es un lenguaje de programación fácil de aprender y, sobre todo, de usar. Recalquemos que

TorqueScript no es sólo un lenguaje creado al azar, sino que tiene todo un equipo de desarrollo detrás de él que ofrece ayuda, tutoriales, blogs e información para los clientes con el fin de proveer una plataforma completa, tanto en los alcances del lenguaje de programación como de soporte, esto con el objetivo de dar a sus clientes la satisfacción de poder solucionar sus dudas ocasionales.

CAPÍTULO

4

DISEÑO DEL PROYECTO

DISEÑO DEL PROYECTO

4.1 DEFINICIÓN DE REQUERIMIENTOS

Se considera una buena práctica, antes de empezar a desarrollar el producto, tener un esquema general de la aplicación que se va a construir. De esta manera tenemos un proceso más fluido, una idea global de los cambios que van ocurriendo y en qué tiempo se va dando, así como cuáles son los eventos siguientes y lo más importante, saber en dónde estamos en el proyecto.

4.1.1 DOCUMENTO DE DISEÑO

Cada desarrollo es un mundo, cada uno con sus procesos, sus limitantes, sus alcances; así como existe una aplicación para cada situación, también tendremos un documento de diseño diferente para cada uno de esos programas.

Cada documento de diseño es manejado de forma diferente para cada uno de nuestros proyectos, es decir, muy difícilmente tendremos más de un documento igual a algún otro. Con esto quiero decir que no debemos casarnos con una manera específica para escribir nuestros documentos de diseño, siempre variarán indudablemente y estas variaciones serán debido a las circunstancias y

procedimientos que se emplearan para la resolución de los problemas que vayamos identificando a lo largo del producto. Ya sea por respuesta a los requerimientos del cliente o la identificación de requisitos necesarios en el programa.

Nuestro documento debe proporcionar dos puntos claros al equipo de desarrollo:[92]

- **Identificar los detalles:** Si nuestro documento de diseño es incapaz de proporcionar detalles de nuestro recorrido, seguramente no tenemos un documento de diseño en nuestras manos. Una de las características imprescindibles en todo documento de diseño, es que éste sea capaz de brindar al equipo de desarrollo absolutamente todos los pormenores, los datos, las referencias con las que tendrán que lidiar en la producción del producto final. Por ejemplo, el dónde irá tal elemento, qué hará con respecto a qué situación, inclusive hasta el más mínimo detalle como las coordenadas de un objeto en el ambiente.
- **Brindar una visión general:** Es una manera de describir los alcances y cómo es que se piensa que se verá nuestro producto cuando esté finalizado. Es como platicar un cuento de lo que sucederá en nuestro recorrido y del porque es tan bueno.

La mayoría de los casos, los documentos de diseño pueden a llegar a ser profundos en los detalles, convirtiendo nuestro documento en un enredado tumulto de hojas por leer, haciendo casi incomprensible el uso del mismo. En la actualidad tenemos bastantes herramientas para crear nuestro documento de diseño y los programadores están empezando a inclinarse cada vez más por estas herramientas como los blog's o los wiki's. Estos últimos han tenido un gran éxito, permitiendo tener de una manera mas organizada la información, ya que es fácil

[92] Duggan, Michael, op. cit., pp. 43-44

crear vínculos entre páginas para acceder a la información. Por ejemplo, se puede dar un panorama general de cierta acción en el recorrido, pero si el diseñador quiere ahondar mas en el tema, probablemente dicha información tenga un vínculo que al seleccionarlo nos lleve a otra página con características particulares de dicha acción.

De cualquier manera, no importa el medio por el cual se elabore el documento de diseño, lo importante es poder transmitir la información necesaria que ayudará a la creación de nuestro recorrido. Lo puntos substanciales que debe tener el documento lo veremos a continuación:[93]

Visión de Conjunto

Antes de abordar la idea completamente del recorrido, los detalles, las funciones, incluso las especificaciones, necesitamos establecer los objetivos del desarrollo. Aunque no es una regla, la mayor parte de las veces, dichos objetivos están escritos en la visión de conjunto del proyecto. Tener la visión concreta del recorrido es a menudo la piedra angular de la construcción del proyecto. Sin ella, seguramente será mas complicado la creación del mismo e incluso al final pueda tener una base inestable que acarre mas de un problema.

¿Cómo puedo escribir mis objetivos? No hay hilos conductores, ni recetas específicas para una correcta creación de nuestros objetivos. Depende en mayor parte de la experiencia y sobre todo de la información recabada para plantear con seguridad los objetivos.

El punto medular para una buena creación de objetivos es pensar en lo que queremos ver y que servicios otorgar en nuestro desarrollo. Una idea para ello es

[93] Ibid., p. 44

seguir la metodología denominada SMART, la cual ha sido definida por James Lewis en su libro *"Fundamentals of Project Management"*[94]

La metodología SMART (del ingles, inteligente) nos pide hacer nuestros objetivos de una forma:

- Specific (específicos)
- Measurable (medibles)
- Attainable (alcanzables)
- Realistic (realistas)
- Time-Limited (con tiempo límite)

Ya que tenemos los objetivos, al menos evaluados de forma concreta es el momento de pasar al siguiente nivel de nuestro documento de diseño.

CADA DESARROLLO ES ÚNICO, AUNQUE GUARDEN UN BAJO PERFIL EN SIMILITUDES, COMO PROCESOS, FUNCIONES, APLICACIÓN HACIA EL USUARIO FINAL, NO OLVIDAR QUE EL ÉXITO ES LA PREPARACIÓN PREVIA QUE SE LE PONE AL PROYECTO. NUNCA HACER SUPOSICIONES, NI DEJAR ZONAS GRISES, MUCHO MENOS APLICAR LA FILOSOFÍA DE 'AL RATO LO VEMOS'.

Audiencia o Usuarios Finales

No podemos desarrollar algo si no tenemos la idea bien clara de cuáles serán las personas que usarán nuestra aplicación. En pocas palabras, conocer bien la audiencia final de nuestro recorrido será la clave del éxito o fracaso de nuestro recorrido, si serán jóvenes, adultos, niños o los tres universos. Este es un detalle muy especial, ya que si nuestro recorrido básicamente está desarrollado

[94] Loc. Cit.

para un cierto universo de personas puede convertirlo en una aplicación muy fácil de realizar o en caso contrario, convertirse en un objeto mas exhaustivo de análisis al tener que concordar estudios para el tipo de usuario final al que va dirigido el recorrido. Para ello tendremos que obtener un estudio estadístico y tener personas cercanas al proyecto de acuerdo al rango valorado de edades o personas a las cuales estará dirigido el producto final. Un ejemplo, si planeamos un recorrido por una universidad, seguramente la mayor parte de usuarios serán jóvenes universitarios y seguramente nuestro equipo de desarrollo oscilarán entre esas edades. Punto a favor sería que si al mayor porcentaje de ellos les gusta el recorrido final, seguramente a una gran parte de los usuarios finales también les gustará.

Sin embargo, para alcanzar un amplio mercado y que nuestra aplicación sea bien recibida por el público valdrá la pena considerar los siguientes puntos:[95]

- **Geográficos:** Los zona geográfica dónde vive nuestra audiencia final es de vital importancia. Aunque un recorrido se ve altamente influenciado por el lugar que vamos a representar, el hecho de conocer la zona geográfica de los usuarios nos permitirá mejores estudios de viabilidad al momento de crear la aplicación como estructuras de entrega y modo de manejo de la información.
- **Demográficos:** Como ya se había mencionado, conocer datos demográficos de nuestra público final, como edad, género y demás características permitirán un mejor manejo de usabilidad y estructura visual de nuestro producto.
- **Psicográficos:** Un término usado tanto en psicología como en posicionamiento y estudios de mercado. Al final nos indica valores, actitudes y creencias de nuestros usuarios finales. Obviamente si nuestro desarrollo final fuera un recorrido por un templo religioso, tendríamos que

[95] Ibid., p.45

cuidar el aspecto creyente de manera de no caer en algún tipo de censura que pudiera ofender algún estrato de esa población.

Alcances del Proyecto

Conocer los alcances de nuestro recorrido nos permitirá estructurar de una manera bien definida los pesos y trayectorias de nuestras metas a alcanzar. Si no tenemos establecidos los alcances de nuestro desarrollo, seguramente no se cumplirán con los tiempos establecidos al tener fugas de procedimientos.

Para lograr un estudio viable de alcances del proyecto debemos centrar nuestra atención en los siguientes puntos:

- **Plataforma:** En que tipo de sistema correrá nuestra aplicación. Es para computadora, para dispositivos móviles, ejecución por internet, consolas de videojuegos.
- **Género:** Aunque el término es comúnmente usado en videojuegos, películas, programas de televisión, la expresión no debe pasar desapercibida por nosotros. Si tuviéramos que hacer algún recorrido por un casino o un bar (y se espera que sea lo mas apegado a la realidad), seguramente el género o tipo de producto no estaría disponible para todo tipo de personas.
- **Modo de Usuario:** Saber si el recorrido se va a ejecutar de modo multi-usuario o solo por un usuario.
- **Vista de Usuario (POV: point of view):** El tipo de vista que tendrá la cámara durante el recorrido. Primera persona, tercera persona o algún otro tipo de modalidad.
- **Contenido:** Qué tipo de contenido se manejará en el recorrido.
- **Estilo:** Pensar en la estética del juego. Considerar la apariencia y el esquema de color a utilizar y sobre todo procurar en gran medida no afectar o agredir el humor del usuario.

- **Interface (Término del ingles):** Pensar en el modo de navegación del usuario durante el recorrido, cómo representaremos la información que vamos a desplegar en pantalla y componentes visuales a usar, éstos y demás factores debemos considerar al momento de estar construyendo nuestros GUI's.
- **Duración:** Planear de forma concreta el tiempo del recorrido. ¿Será en tiempo real, se verá afectado por turnos, tendrá un límite de tiempo o dependerá del usuario su estadía en el mismo?
- **Audio:** El audio se aplicará de forma 3D o 2D, habrá efectos o será sintético.
- **Tecnología:** ¿Qué tipo de tecnología emplearemos para la creación de nuestro recorrido? Y sobre todo, fijarnos en las limitaciones que tendremos al momento de usar dichas tecnologías. No considerar con antelación este tipo de problemas puede acarrear problemas graves a medida que el proyecto esté avanzado.

DEBEMOS PONER ATENCIÓN EN LOS ALCANCES DEL PROYECTO. SON LA BASE QUE SUSTENTARÁ TODO NUESTRO TRABAJO A REALIZAR PARA NUESTRO PRODUCTO, PERO PRINCIPALMENTE EN EL PUNTO DE TECNOLOGIA. NO HABER CONSIDERADO UNA LIMITACIÓN PUEDE PONER EN RIESGO TODO EL PRODUCTO Y MAS SI DICHA LIMITACIÓN FUESE UNA CARACTERÍSTICA PRIMORDIAL EN EL PRODUCTO.

A grandes rasgos, esta parte del documento de diseño contiene las características principales que tendrá nuestro recorrido. Qué características hemos decidido que estarán disponibles al usuario. Cuáles serán las ventajas que se ofrecerán como producto de venta, en cambio si solo nuestro estudio se ve enfocado a la cuestión técnica, éste será otro apartado que estudiaremos mas adelante.

El conjunto de características también debe explicar cómo es que el usuario se moverá durante el recorrido, mencionar los aspectos mas destacados durante

dicha experiencia y sobre todo plantear la realidad en la que el usuario será capaz de interactuar. Si no pudiéramos definir estos puntos desde un principio, seguramente nuestro desarrollo no tendrá el impacto que deseamos generar.

Desglose del Trabajo

Es una de las partes mas interesante, concisas, y la que mas frustraciones o alegrías nos brindará. Ésta etapa no es mas que dividir nuestro proyecto en tareas y sub-tareas, las cuales se asignarán a los miembros del equipo de trabajo y se estimarán los tiempos que se tardarán en realizar las tareas planteadas.

En ocasiones plantearnos tareas nos puede resultar un poco complicado, no por saber identificarlas si no por la falta de precisión en las mismas. Para evitar ambigüedades en nuestras tareas podemos considerarlas como específicas y/o generales. Hay más métodos que nos pueden ayudar a identificar tareas para nuestros proyectos, caso que la Ingeniería de Software nos puede ayudar. Pero para no complicarnos empezaremos con la metodología básica de tipificación.[96]

- **Tareas Específicas:** Son los pasos necesarios para completar una característica del proyecto. Un ejemplo sería la creación del 'avatar' del usuario (si es que hubiera). Para la realización de la figura, ya sea nosotros o el equipo de creación, se necesitaría de alguien describiera la caracterización del modelo, un artista para el trabajo de arte (dibujar el modelo), un modelador 3D para la elaboración y animación de la figura en tres dimensiones. Cada una de estas son tareas son tareas específicas.
- **Tareas Generales:** Son los pasos generales que se aplican a casi todas las características en el producto, por ejemplo: para crear la interfaz, el estilo, la historia, y el recorrido, incluyendo la garantía de calidad o el manejo de pruebas.

[96] Ibid., p. 50

Estimación del Tiempo

Una vez que hemos identificado las tareas a realizar es momento de estimar el tiempo de realización para cada una de ellas.

Antes de continuar hay que considerar que hay eventos interdependientes que pueden interponerse en la realización de una tarea. No podemos empezar a realizar un 'test' de cierto mundo, si no están colocados todos los objetos en la ambientación, pero no podemos finalizar los objetos hasta que el jefe de modelado no los haya revisado y no podemos empezar la creación de los objetos si no han sido dibujados por el artista. Pasos como los que hemos descrito, son puntos en un proyecto en los cuales nada se puede hacer si la tarea anterior no ha sido completada. Dichos puntos se consideran acontecimientos importantes que pueden convertirse en cuellos de botella si no son identificados correctamente dentro de nuestra línea de tiempo.

Estimación de Costos

Aunque nuestro trabajo no plantea una revisión de costos, es necesario mencionarlo. Identificar los costos es muy importante para que nuestro proyecto tenga un final feliz. Un desarrollo de esta envergadura obliga entregar la fusión en un paquete del trabajo de artistas, diseñadores, programadores, escritores, ingenieros de audio, etc. Identificar estos roles, sus costos, su formas de trabajo y el tiempo que nos llevará entregarlo es la pauta para estimar el costo total de nuestro producto.

En esa estimación hay que considerar: empelados, licencias, contratos externos (outsourcing), equipo y demás detalles necesarios para la realización de nuestro recorrido.

En nuestro caso, contamos con un lugar de trabajo, una computadora y el equipo humano (uno mismo) para completar el recorrido, es decir, los costos ya van incluidos en la realización del trabajo.

Detalles a Considerar

Sabemos que tenemos más de una forma de escribir un documento de diseño, incluso que tenemos herramientas digitales que nos pueden ayudar a la realización de nuestro escrito. Dependiendo la forma en que se entregará el documento podemos incluir o no una parte extra al documento. Nos referimos a los activos de producción (elementos inherentes al recorrido como objetos, modelos, elementos naturales, terrenos, clima, bocetos), el diseño de interface, diseño de audio y especificaciones técnicas.

Si tenemos la oportunidad de crear un documento de diseño ya sea digital o por medio de herramientas interactivas, indudablemente será necesario que también contenga este material. No hay que olvidar que el documento de diseño es la base de nuestro desarrollo, es el manual, el archivo de consulta, es el que contiene toda la información necesaria para el éxito de nuestra aplicación. No contar con un documento de diseño hará que nuestro proyecto y nuestro equipo de desarrollo carezcan de los elementos necesarios para la elaboración de los activos de producción en primera instancia.

No entraremos a detalle con los puntos anteriores, pero se darán pautas que podremos considerar al elaborar el documento de diseño para esta sección.

Para la parte de los activos de producción, dividiremos las misiones en pequeños objetos de estudio, brindando al lector información sobre los niveles del recorrido (por ejemplo, transiciones entre misiones), describir por qué son importantes dicho número de niveles o por qué son necesarias las misiones. En ocasiones los diseñadores de las misiones esquematizan en mapas las formas de los ambientes mientras que los modeladores o desarrolladores de entornos ofrecen una descripción ‘física’ del entorno así como funciones del escenario.

Algunas cosas que debemos tener presente al describir nuestros escenarios son:[97]

- **Clima:** Si es de día o de noche o prevalece alguna situación natural como viento o lluvia.
- **Iluminación:** Si es natural, por ejemplo, iluminación creada por el sol o será artificial, a través de luminarias ya descritas en capítulos anteriores.
- **Objetos:** Aquí hablaremos de los artefactos o escenografía que se incorporará en los ambientes.
- **Escala:** Si fuera necesario describir con precisión las dimensiones del entorno o algún tipo de medida a considerar por el recorrido. Por ejemplo la extensión de alguna misión.
- **Tema:** Si el recorrido tiene algún tema en particular o es abierto al público en general.
- **Recorrido:** Se describe el estilo de viaje o recorrido que hará el usuario dentro de las misiones.

Aunque puede ser una cualidad externa al estudio preliminar, vale la pena considerar si existe algún elemento extra durante el recorrido, por ejemplo, inteligencia artificial, usuarios ajenos inmersos en el ambiente, guías añadidas que provean información dentro del recorrido (otros actores en la escena), etc. Este tipo de elementos se considerarán como otro punto llamado **Personajes**.

Ya hemos hecho estudios sobre la parte de Diseño de Interfaces y Diseño de Audio, de manera que no ahondaremos más en el tema en este apartado. En cambio para la parte de Especificaciones Técnicas es un área importante de nuestro documento de diseño donde por lo menos debemos detallar información como:[98]

---

[97] Ibid., p. 52
[98] Ibid., p. 53

- **Bases de Datos, Servidores y tipo de Red:** Si usaremos bases de datos, que tipo de servidor necesitaremos o si el recorrido se montará sobre algún tipo de red.
- **Programas de Desarrollo:** Si necesitamos software especializado para la creación o programación del recorrido.
- **Video:** Referencia a la parte visual como resoluciones de pantalla mínimas o especificaciones de hardware apropiado para la ejecución del recorrido.
- **Internet:** No confundir con la parte de Red, podemos montar nuestro recorrido en una red interna, por ejemplo, un recorrido virtual dentro de una empresa que estará accesible únicamente a equipos dentro de un complejo, el cual no necesariamente tiene que conectarse a internet o en caso contrario, tiene que conectarse a internet para recibir cierta información necesaria para su ejecución.
- **Plataforma:** En que tipo de sistema se ejecutará nuestro recorrido, Windows, Macintosh, Internet, algún otro tipo de sistema. Recomendaciones mínimas y especificación de hardware recomendadas para su óptima ejecución.
- **Seguridad:** Si se necesita un tipo de seguridad añadida al recorrido o puede ser de libre distribución y ejecución. ¿Se necesita algún tipo de seguridad?

En nuestro caso se propone una estructura diferente para el documento de diseño del recorrido por el Templo de Lourdes, Centro Histórico, Ciudad de México. Hay que aclarar que tendremos dos secciones en el documento de diseño, una sección para el cliente y la otra para el equipo desarrollador. En muchas ocasiones es necesario tener un documento que podamos presentar al cliente. Al cliente no le importa en demasía detalles técnicos o procesos exhaustivos en el desarrollo de la aplicación, pero seguramente esa información será de vital importancia para el equipo desarrollador. De manera que se presenta un documento de diseño dividido en dos partes, la primera es información general

del proyecto que sirve como presentación al cliente, no obstante no significa que esta información no sea de importancia al equipo desarrollador, es decir, aunque no tenga datos significativos del proceso de diseño y programación del recorrido, brinda un panorama general al equipo de trabajo para darse una idea de los alcances del proyecto. Mientras que la segunda parte si destaca elementos informativos detallados de todas las funciones que tendrá el recorrido, datos que permitirán una mejor evaluación de las tareas y procesos involucrados para la realización y terminación del recorrido.

Con esto damos por terminada la parte de creación del documento de diseño. Es necesario comentar que no es obligación seguir éste tipo de ejemplo para la redacción del documento de diseño, hay muchas formas de escribirlo, pero hemos puntualizado al menos la forma, características y elementos mínimos que debe contener. Ya dependerá del equipo de trabajo la forma más adecuada para estructurarlo y elaborarlo.

A continuación se muestra el documento de diseño para el recorrido virtual por el Templo de Lourdes, Centro Histórico, Ciudad de México.

RECORRIDO VIRTUAL POR EL TEMPLO DE LOURDES
CENTRO HISTÓRICO, CIUDAD DE MÉXICO
Versión 1.0

1. Descripción General:

El recorrido virtual por el Templo de Lourdes tiene como finalidad brindar la oportunidad de conocer la belleza arquitectónica de la Iglesia, así como conocer las obras de arte que se exponen en ella, recreando el entorno tanto de los alrededores como del mismo templo lo más fielmente posible. Ofreciendo además la oportunidad que el usuario pueda interactuar con el ambiente recreado del Templo de Lourdes a través de la colocación de objetos en la Iglesia.

2. Visión General:

Escenario

El escenario del recorrido virtual por el Templo de Lourdes está dividido en dos mundos, el primero se desenvuelve por los alrededores del templo. Presentando una extensión de una cuadra por cada lado de la Iglesia, haciendo un mayor énfasis en las esquinas de 16 de Septiembre, Bolívar y Venustiano Carranza, situadas en el Centro Histórico de la Ciudad de México. Mientras que en el interior del Templo de Lourdes se considera el segundo mundo, mostrando la nave principal y la capilla lateral con la que cuenta. Actualmente el recorrido esta situado durante el día de la Ciudad de México.

Recorrido

El recorrido es completamente a gusto del usuario, aunque está delimitado en sus alrededores el usuario puede desenvolverse sin ningún problema dentro del recorrido tanto por los alrededores como en el interior del Templo de Lourdes.

El inicio del recorrido empieza en la calle 16 de Septiembre, localizada al Norte del Templo de Lourdes, durante esta etapa del recorrido el usuario podrá tomar notas y fotografías de los alrededores. Estas notas y fotografías serán almacenadas en el equipo las cuales podrán ser consultadas durante el recorrido o fuera de el.

RECORRIDO VIRTUAL POR EL TEMPLO DE LOURDES
CENTRO HISTÓRICO, CIUDAD DE MÉXICO
Versión 1.0

También cuenta con una función de acercamiento, permitiendo configurar el acercamiento a utilizar. Con esta cualidad el usuario podrá observar con mejor detalle los objetos cercanos a el.

Durante el recorrido en el interior del Templo de Lourdes se activan tres funciones extras al usuario. Visualización de Objetos, Colocación de Objetos y Movimiento de Objetos. La función de Visualización de Objetos permitirá al usuario dar clicks sobre algún elemento decorativo del Templo de Lourdes, como santos o cuadros y de esta manera observar mas de cerca el objeto seleccionado así como una breve reseña o pasaje alusivo al mismo. Mientras que las funciones de Colocar y Mover Objetos permitirán interactuar al usuario con el entorno del Templo Lourdes. Estos objetos pretenderán representar adornos florales y arreglos decorativos de una Florería real, los cuales estarán creados lo más fielmente posible al original.

Apariencia

El recorrido trata de ser lo mas fiel posible a los alrededores del Templo de Lourdes, mostrando las fachadas: color, diseño, disposición, de los edificios existentes en estos momentos (años 2010-2011), elementos decorativos y de información que se encuentran en las calles aledañas como semáforos, letreros de nombres de calles, etc. Emulando los sonidos y las condiciones ambientales de un día normal entre las 10:00 y 12:00 horas.

Con respecto al interior del Templo de Lourdes, las condiciones son similares al entorno real, mostrando en el recorrido efectos de luz ambiental y disposición real de los objetos a lo largo del ambiente representado.

3. Información Comercial:

Audiencia

El recorrido virtual por el Templo de Lourdes esta dirigido a todo tipo de audiencia. Especialmente a aquellos que deseen conocer el inmueble ya sea por propósitos de información, recolección de datos inventaríales, visitas

RECORRIDO VIRTUAL POR EL TEMPLO DE LOURDES
CENTRO HISTÓRICO, CIUDAD DE MÉXICO
Versión 1.0

turísticas y de investigación histórica o técnica.

Hay un estrato especial de la audiencia al cual va dirigido el recorrido por el Templo de Lourdes, personas que necesitan conocer la iglesia porque tendrán algún evento en el reciento y no pueden desplazarse hacia el inmueble ya sea por motivos de tiempo, lugar o impedimentos de diferente índole. Permitiendo así representar el templo de la manera mas fiel y real posible al usuario, haciéndolo sentir inmerso e incluso como si estuviera presente en la Iglesia, incluyendo a esta representación la oportunidad de adornar el templo con diferentes elementos decorativos que tendrá a su disposición.

Plataforma

El recorrido por el Templo de Lourdes puede ser ejecutado en equipos con sistema operativo Windows y Macintosh. La aplicación es ejecutable y compatible para ambos sistemas debido a que son los más comunes y populares en el mercado.

Requisitos del Sistema

Para equipos con sistema operativo **Windows:**

Requerimientos Mínimos

- Microsoft Windows XP o superior
- Procesador 1 GHz
- OpenGL o DirectX® 9.0 o posterior
- 512 MB RAM (1GB para Vista/7)
- 600 megabytes (MB) de espacio disponible en disco duro.
- OpenGL o DirectX® compatible con funciones 3D de transformación e iluminación.
- CD: 8X
- Tarjeta de sonido, altavoces o auriculares.
- Modem o adaptador de Red ya sea Ethernet o WiFi para conexión a Internet
- Proveedor de servicios de Internet (ISP)

RECORRIDO VIRTUAL POR EL TEMPLO DE LOURDES
CENTRO HISTÓRICO, CIUDAD DE MÉXICO
Versión 1.0

Requerimientos Recomendados

- Microsoft Windows XP o superior
- Procesador 2 GHz
- OpenGL o DirectX® 9.0 o posterior
- 1 GB RAM (2GB para Vista/7)
- 600 megabytes (MB) de espacio disponible en disco duro.
- Acelerador gráfico nVidia o ATI con funciones 3D de transformación e iluminación.
- CD: 8X
- Tarjeta de sonido, altavoces o auriculares.
- Modem o adaptador de Red ya sea Ethernet o WiFi para conexión a Internet
- Proveedor de servicios de Internet (ISP)

Para equipos con sistema operativo **Macintosh:**

Requerimientos Mínimos

- Intel Mac.
- 512 MB en RAM
- MAC OSX 10.5 o superior.
- acelerador gráfico nVidia o ATI
- 600 megas libres en disco duro
- Unidad de CD
- Tarjeta de sonido, altavoces o auriculares.
- Adaptador de Red ya sea Ethernet o WiFi para conexión a Internet
- Proveedor de servicios de Internet (ISP)

Requerimientos Recomendados

- Intel Mac.
- 1 GB en RAM
- MAC OSX 10.5 o superior.
- acelerador gráfico nVidia o ATI
- 600 megas libres en disco duro
- Unidad de CD

RECORRIDO VIRTUAL POR EL TEMPLO DE LOURDES
CENTRO HISTÓRICO, CIUDAD DE MÉXICO
Versión 1.0

- Tarjeta de sonido, altavoces o auriculares.
- Adaptador de Red ya sea Ethernet o WiFi para conexión a Internet
- Proveedor de servicios de Internet (ISP)

Comparación de Características

A diferencia de la mayoría de los recorridos virtuales existentes en el mercado, el recorrido virtual por el Templo de Lourdes presenta particularidades llamativas al usuario como recorrido no lineal, interacción con el ambiente, opciones de escribir y tomar fotografías durante el paseo, por mencionar algunas funciones, además de ser un producto multi-plataforma permite una fácil comercialización, debido a la portabilidad de su tamaño (600 megas MB), por lo cual puede distribuirse por medios físicos como CD, USB o estar accesible al público a través de Internet.

4. Copyright

Las normas y principios que regulan los derechos morales y patrimoniales del recorrido virtual por el Templo de Lourdes serán dados por la empresa a la cual será entregado el producto final. Dicha normatividad corresponderá a la necesidad del cliente de acuerdo a al modo de difusión de la aplicación.

RECORRIDO VIRTUAL POR EL TEMPLO DE LOURDES
CENTRO HISTÓRICO, CIUDAD DE MÉXICO
Versión 1.0

5. Recorrido

El diseño del recorrido por el Templo de Lourdes tendrá como entrada pantallas donde se presentarán los créditos de las empresas participantes y finalizará con un video introductorio de la historia del templo de Lourdes.

Al finalizar los créditos y el video introductorio se desplegará el menú principal que contendrá un apartado de información general del recorrido y otro mas para la configuración de las funciones que estarán disponibles para el usuario.

Se contará con una ventana que permitirá seleccionar el tipo de recorrido al que desea ingresar el usuario, entre las opciones se tendrá un recorrido completo, uno alterno para equipos que no cuenten con un acelerador gráfico y uno mas con el recorrido inmediato por el Templo de Lourdes.

El recorrido principal contará con dos misiones, la primera será por los alrededores del Templo de Lourdes y la segunda consistirá dentro de la Iglesia. Este recorrido principal detalla dos modos, el primero será con la mayoría de los detalles que se encuentran actualmente en las calles aledañas a la Iglesia, mientras que el recorrido alterno tendrá en su ambiente sólo los elementos fundamentales para permitir que el usuario puede reconocer el ambiente que rodea al templo, considerando el mínimo de elementos necesarios para dicho caso.

Existirá la tercera opción que permitirá ingresar directamente al usuario a través del recorrido por el Templo de Lourdes.

La misión perteneciente al recorrido por el Templo de Lourdes es única para los dos tipos de recorridos que tendrá, no importando si es con mínimo detalle o es el recorrido completo, las características del recorrido por el interior de la Iglesia serán las mismas para cualquier tipo de selección de recorrido.

RECORRIDO VIRTUAL POR EL TEMPLO DE LOURDES
CENTRO HISTÓRICO, CIUDAD DE MÉXICO
Versión 1.0

Durante el recorrido por la misión principal o la misión alterna, selección que escogerá el usuario al principio del recorrido, se podrá tomar notas a través de una dialogo que admitirá la escritura de texto para las notas. De la misma manera se podrá tomar fotografías que serán almacenadas directamente en el equipo. Para contemplar detalles de los alrededores se ha agregado una función adicional que permite acercamientos (‘zoom’) en cualquier momento.

Después que el usuario haya recorrido los alrededores a su gusto, podrá ingresar al interior del Templo de Lourdes. En este momento se iniciará la segunda misión del recorrido. Dentro del Templo de Lourdes, aparte de las funciones que ya se describieron, se activarán tres funciones adicionales:

Visión de Objetos: Permitirá contemplar ciertos objetos que se encontrarán al interior del Templo de Lourdes, brindando información del mismo y un acercamiento al objeto seleccionado.

Colocación de Objetos: De acuerdo a convenios establecidos se agregará un distribuidor de arreglos florales en la aplicación para acceder a los adornos destacados que otorgó el empresario para uso del recorrido por el Templo de Lourdes.

Movimiento de Objetos: Los objetos que sean colocados al interior del Templo de Lourdes tendrán la cualidad de ser desplazados a lo largo del templo, con la finalidad de poder adornar el templo al gusto del cliente. Estos objetos permanecerán en el Templo mientras la misión permanezca activa. Saliendo del interior del Templo de Lourdes, los objetos ya no permanecerán en el interior de la Iglesia.

Los movimientos del usuario durante el recorrido por el Templo de Lourdes serán por medio del teclado y el ratón. Las configuraciones del teclado que sean programadas por omisión en el recorrido, podrán ser configuradas a gusto del usuario.

RECORRIDO VIRTUAL POR EL TEMPLO DE LOURDES
CENTRO HISTÓRICO, CIUDAD DE MÉXICO
Versión 1.0

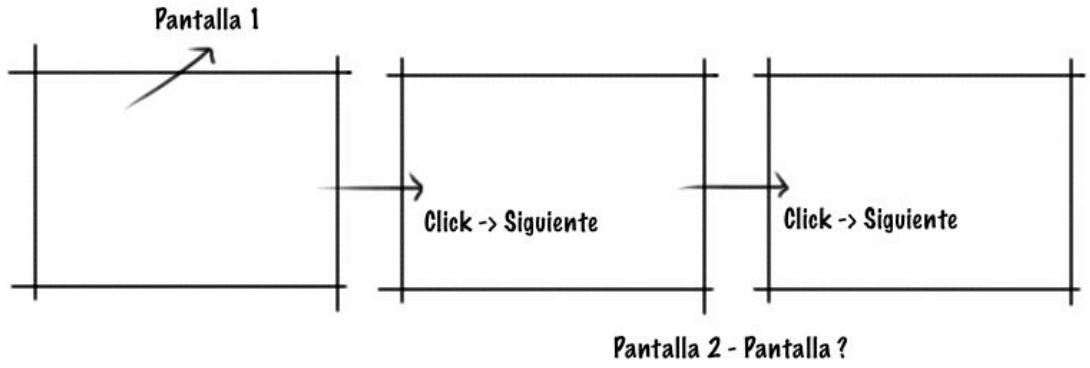
Al finalizar el recorrido, tanto las notas como las fotografías tomadas durante la ejecución del programa estarán accesibles al usuario en cualquier momento, ya que se encontrarán almacenadas en el equipo donde se ejecuta la aplicación.

6. Pantallas y Controles (Descripción detallada)

Pantallas

La disposición de los créditos de entrada será de la siguiente manera:

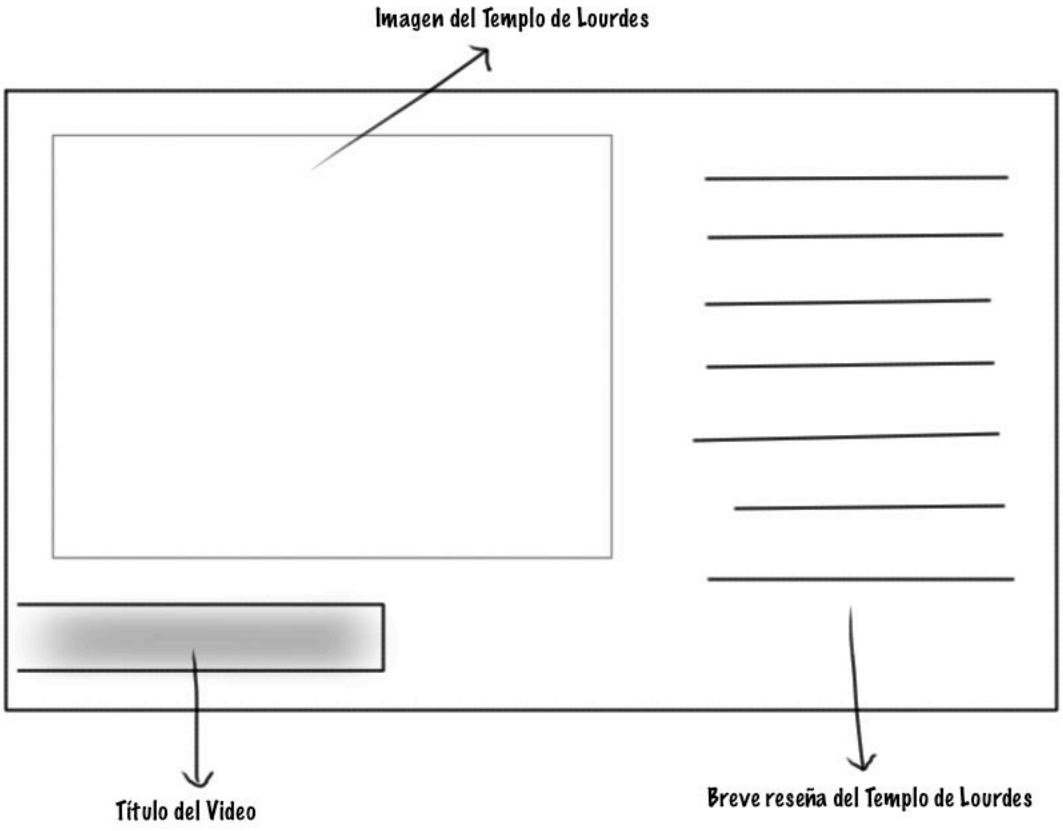
Pantalla 1: Empresa desarrolladora del producto.
Pantalla 2: Créditos para la empresa del motor del recorrido.
Pantalla 3-?: Créditos necesarios para el recorrido.



Las pantallas contarán con un máximo de 3 segundos por cada una de ellas. La primer pantalla estará obligada a pasar completamente, las siguientes pantallas con un click en cualquier sitio de la ventana del recorrido indicará al sistema que se desea pasar a la siguiente pantalla.

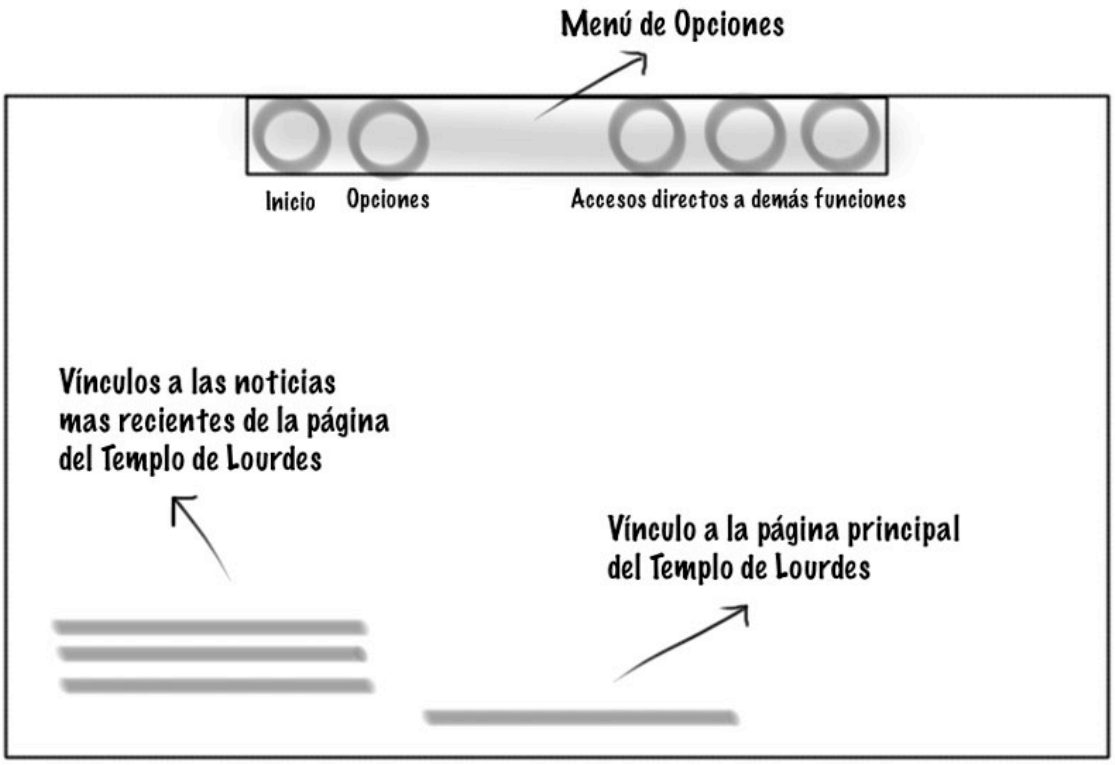
Al finalizar los créditos iniciales se desplegará en pantalla un video introductorio con una breve reseña del Templo de Lourdes. Se propone que el video no dure más de un minuto y que en el video se esboce alguna fachada exterior del Templo de Lourdes. No contará con la opción de salto.

RECORRIDO VIRTUAL POR EL TEMPLO DE LOURDES
CENTRO HISTÓRICO, CIUDAD DE MÉXICO
Versión 1.0



Al finalizar el video introductorio del Templo de Lourdes se desplegará en pantalla el menú principal del recorrido. En la parte superior tendrá una barra de opciones. Un acceso para el inicio del recorrido, uno más para las configuraciones globales del recorrido: video, audio, controles. Y un espacio para accesos directos como ayuda, salir del recorrido y si fuera necesario espacio adicional dentro del menú para vínculos importantes. Se desea también que contenga un apartado que muestre las últimas noticias que se han publicado en la página del Templo de Lourdes.

RECORRIDO VIRTUAL POR EL TEMPLO DE LOURDES
CENTRO HISTÓRICO, CIUDAD DE MÉXICO
Versión 1.0

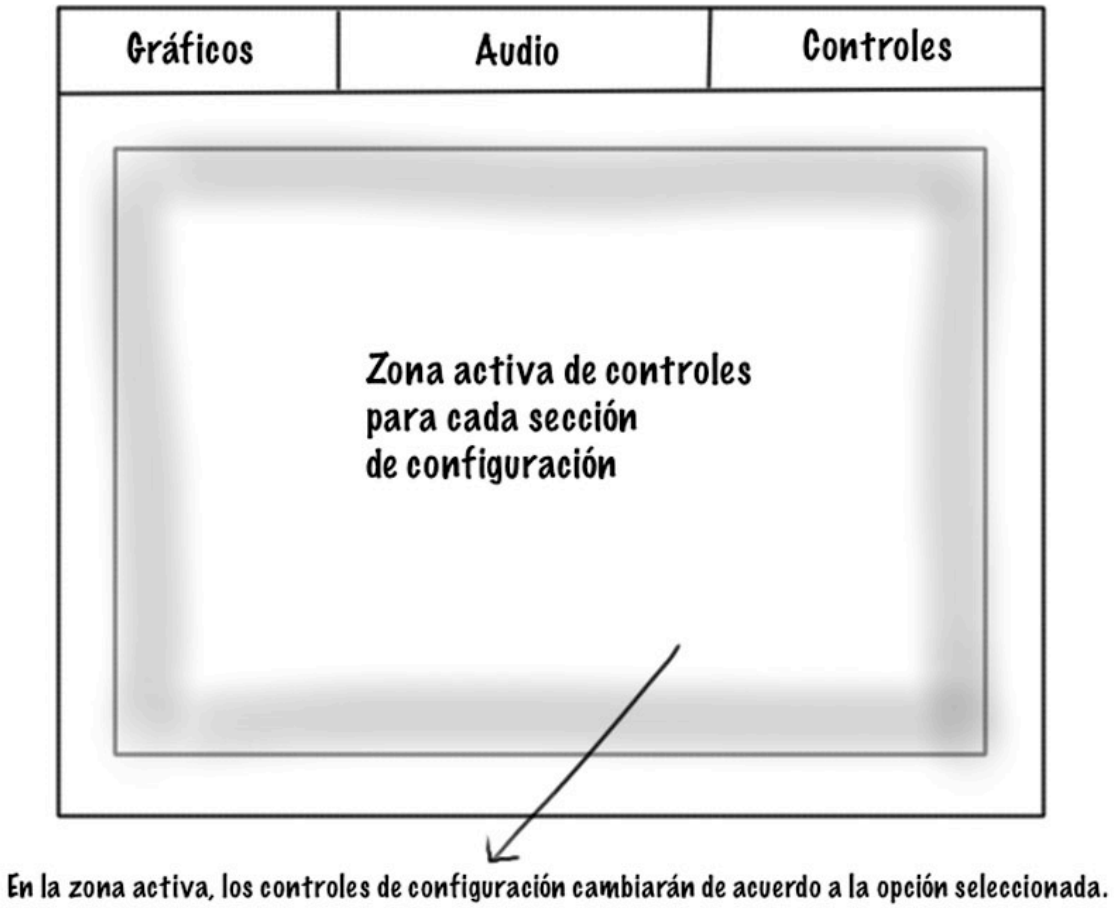


El acceso de Inicio abrirá una ventana que mostrará las tres opciones de recorrido a escoger. Recorrido Principal: Contendrá el máximo de detalles en objetos en los alrededores del Templo de Lourdes (recomendado para equipos con acelerador gráfico), Recorrido Alterno: También permitirá hacer un recorrido por los alrededores del Templo de Lourdes, pero el ambiente tendrá el mínimo de detalle admisible para podre representar el entorno y pueda orientarse el usuario en las vecindades de la Iglesia. Recorrido por el Templo de Lourdes: Opción que envía al usuario directamente al interior del Templo de Lourdes.

El acceso de Opciones permitirá la configuración general de los modos de video, audio y controles. La parte de Video tendrá las opciones de resolución de pantalla, tanto para pantalla completa o en modo ventana, la profundidad de bits que se empleara durante el recorrido y el tipo de formato

RECORRIDO VIRTUAL POR EL TEMPLO DE LOURDES
CENTRO HISTÓRICO, CIUDAD DE MÉXICO
Versión 1.0

que se utilizará para las tomas fotográficas. También incluirá la opción para seleccionar el tipo de visualización, modo ventana o pantalla completa. Para la sección de Audio se tendrán controles que permitan escoger el volumen de los diferentes tipos de sonido que tengamos en la aplicación. Y la última sección Controles permitirá visualizar que botones del teclado están configurados para las funciones que tendrá a su disposición el usuario dentro del recorrido. Se permitirá configurar los botones de las funciones de acuerdo a la selección que elija el usuario, se recomienda que el cambio de tecla sea mediante un 'doble click' sobre la tecla asignada a la función.



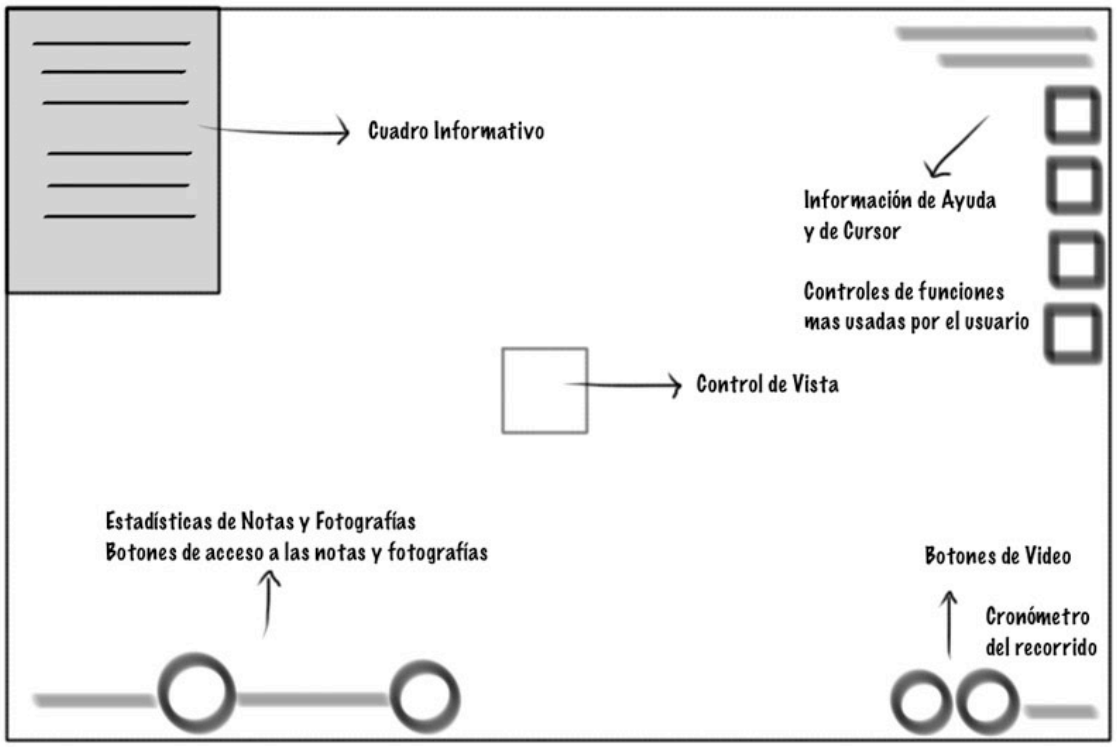
RECORRIDO VIRTUAL POR EL TEMPLO DE LOURDES
CENTRO HISTÓRICO, CIUDAD DE MÉXICO
Versión 1.0

Los accesos directos a las demás funciones quedarán en valoración de acuerdo a las necesidades que vayan surgiendo durante el desarrollo del recorrido. Se recomienda que si alguno de esos accesos debe proporcionar algún tipo de configuración, se utilice como base el ejemplo de Opciones.

Para el entorno base que se tendrá durante el recorrido por el Templo de Lourdes se estructuró una interface que tenga las funciones mas usadas al alcance del usuario en un par de clicks.

Del lado izquierdo superior habrá un cuadro de control, que desplegará información de los objetos que estén presentes durante el recorrido, así como el nombre de usuario y algún elemento seleccionado por medio de una mira que estará colocada al centro de la pantalla. Del lado derecho superior estará escrita una leyenda indicando que botón esta asignado para la ayuda y con que botón se activa el puntero del mouse. Del lado derecho estarán los botones con las funciones más comunes que estarán disponibles para el usuario: Notas, Fotografías, Agregar Objeto, Cursor, Información. En la parte inferior de la pantalla se tendrá un menú dedicado para las notas y fotos. Dicho menú mostrará en pantalla los datos en tiempo real sobre el número de notas que se han tomado y el número de archivos guardados en disco. De igual manera se tendrá la información para las fotos actuales y las fotos guardadas en disco. Junto a cada menú está su respectivo botón que enviará al usuario a la carpeta donde se encuentran almacenados estos archivos. Del lado inferior derecho se tendrán dos controles de Video, un botón de color rojo que permitirá la mayor resolución a pantalla completa, y el siguiente control que admitirá el cambio de Video, ya sea a pantalla completa o modo ventana, estos cambios de pantalla será en la resolución activa en la que esta corriendo el recorrido, para hacer cambios en resolución de pantalla será necesario salir del recorrido y cambiarla manualmente desde el menú inicio en la parte de Opciones, y al final un control que llevará la cuenta del tiempo que ha permanecido el usuario dentro del recorrido.

RECORRIDO VIRTUAL POR EL TEMPLO DE LOURDES
CENTRO HISTÓRICO, CIUDAD DE MÉXICO
Versión 1.0

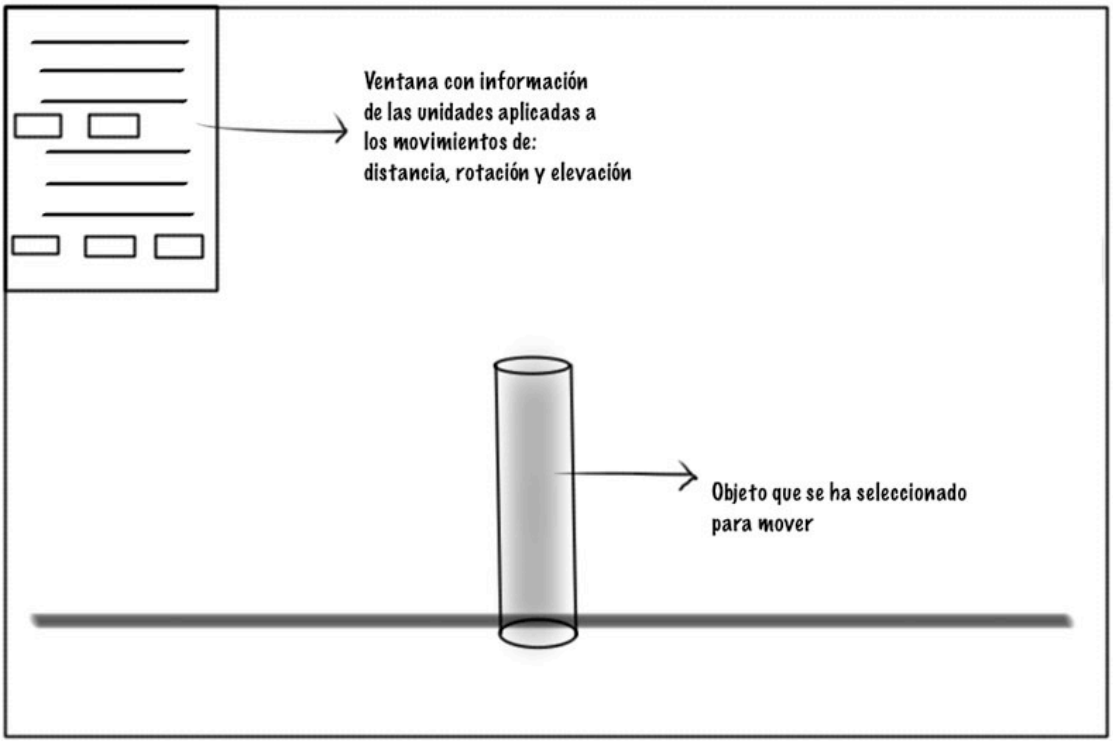


La disposición del diseño de interface anterior permanecerá a lo largo de todo el recorrido y aplicará para cualquier misión disponible.

Para las funciones adicionales que se activarán en el interior del Templo de Lourdes contarán con la siguiente presentación. Para la función de 'Visualizar Objeto', dicha función se activará con el botón izquierdo del ratón, la ventana que mostrará la información contará con dos cuadros principales. Del lado izquierdo de la ventana habrá un cuadro de texto que mostrará la información perteneciente al objeto que se ha seleccionado, puede ser una descripción o una reseña del mismo. Del lado derecho de la ventana se verá la imagen del objeto seleccionado, aunado a esto, con los botones del mouse el usuario podrá hacer acercamientos y rotar la imagen que esta observando. La siguiente función 'Mover Objeto' únicamente desplegará en pantalla una ventana del lado superior izquierdo, la cual mostrará en pantalla los controles de movimiento: mover, rotar y elevar con sus respectivas unidades

RECORRIDO VIRTUAL POR EL TEMPLO DE LOURDES
CENTRO HISTÓRICO, CIUDAD DE MÉXICO
Versión 1.0

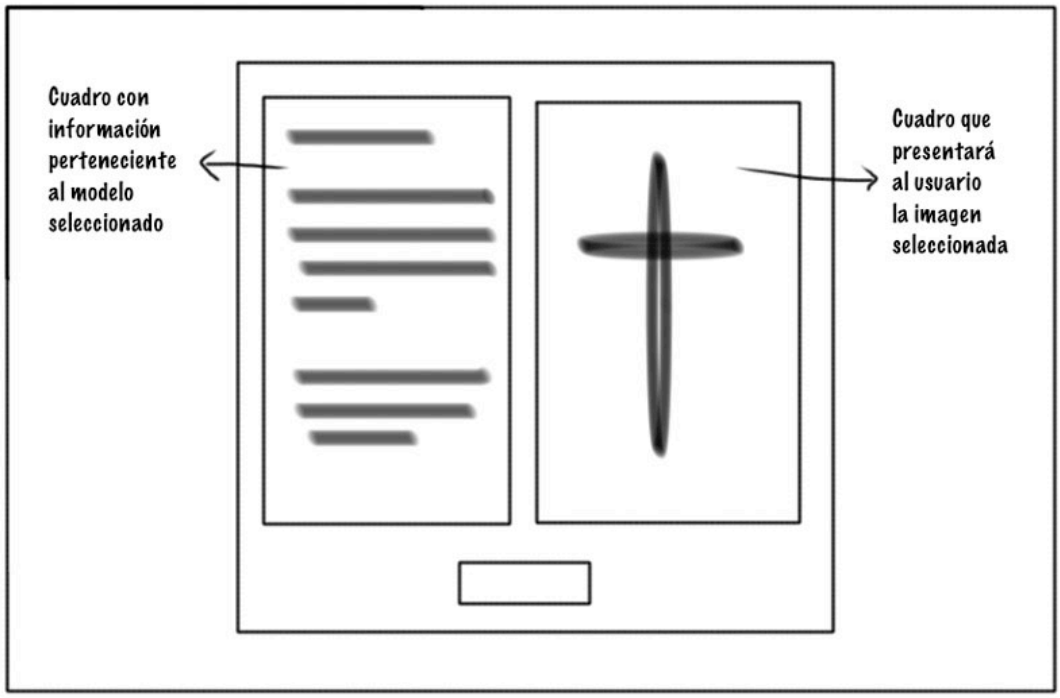
de medida, estas unidades de medida podrán modificarse directamente en la ventana de 'Mover Objeto' por medio de botones, ya sea para aumentar o disminuir las cantidades. El movimiento del objeto será por medio de los botones de flechas y obedecerá a las cantidades que se muestran en el cuadro de 'Mover Objeto'.



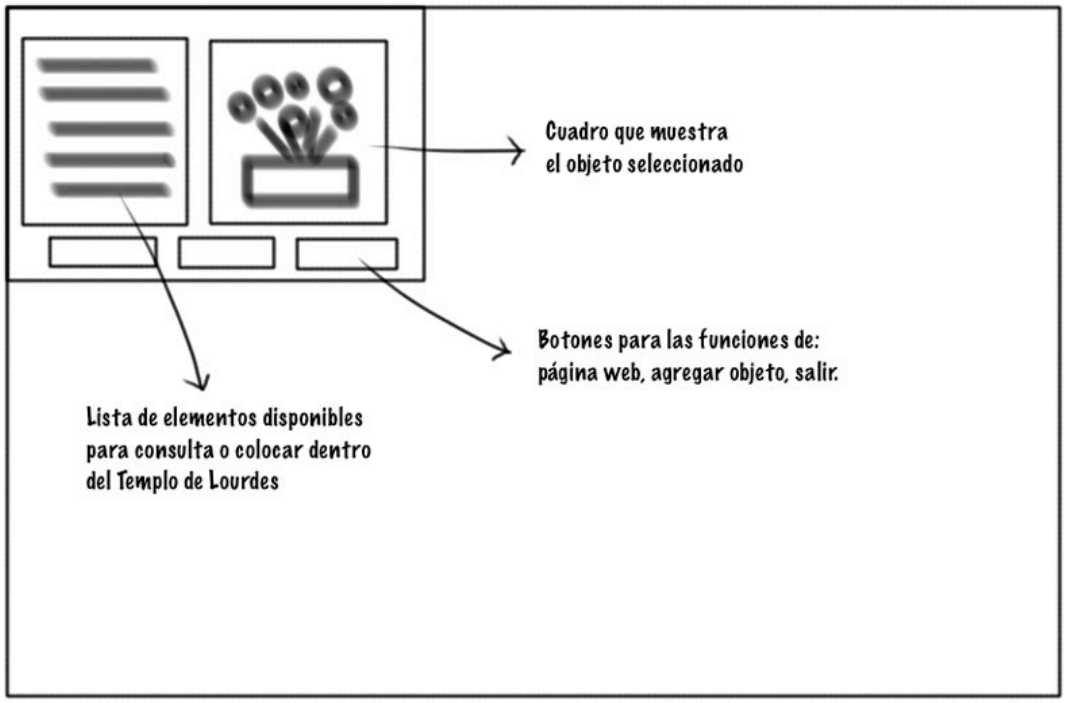
Para la siguiente función activa en el Templo de Lourdes, 'Agregar Objeto'. La ventana perteneciente a este comando desplegará la información como se muestra a continuación. Presentará una ventana en la parte superior izquierda con dos cuadros principales. El cuadro izquierdo mostrará una lista de elementos que pueden ser colocados al interior del Templo de Lourdes y el cuadro derecho una imagen del objeto a colocar. Estos elementos al ser seleccionados contarán con dos opciones, seleccionar el control 'vía web', para consultar mayor información directamente en la página del vendedor o el control de 'agregar objeto', para colocar el adorno en el interior de la Iglesia.

RECORRIDO VIRTUAL POR EL TEMPLO DE LOURDES
CENTRO HISTÓRICO, CIUDAD DE MÉXICO
Versión 1.0

'Ver Objeto'



'Agregar Objeto'



RECORRIDO VIRTUAL POR EL TEMPLO DE LOURDES
CENTRO HISTÓRICO, CIUDAD DE MÉXICO
Versión 1.0

Controles

Los controles configurados por defecto son:

Tecla:	Función asociada:
Esc	Salir del recorrido y ventanas emergentes.
A	Caminar a la izquierda.
D	Caminar a la derecha.
W	Caminar hacia adelante.
S	Caminar hacia atrás.
Q	Correr.
Flecha arriba	Mover objeto hacia adelante.
Flecha abajo	Mover objeto hacia atrás.
Flecha Izquierda	Mover objeto hacia la izquierda.
Flecha Derecha	Mover objeto hacia la derecha.
Espacio	Mover objeto enfrente.
Retroceso	Borrar objeto.
Tabulador	Mostrar cursor.
R	Seleccionar nivel de acercamiento.
E	Usar función de acercamiento.
F	Tomar fotografía.
Shift + Flecha arriba	Mover objeto adelante.
Shift + Flecha abajo	Mover objeto atrás.
Shift + Flecha izquierda	Rotar objeto.
Shift + Flecha derecha	Rotar objeto.
Botón izquierdo ratón	Ver información del objeto.
Botón derecho ratón	Menú mover objeto.

RECORRIDO VIRTUAL POR EL TEMPLO DE LOURDES
CENTRO HISTÓRICO, CIUDAD DE MÉXICO
Versión 1.0

7. Contexto del Recorrido

El recorrido por el Templo de Lourdes se desarrolla en los años 2010-2011 por las cuadras aledañas a la Iglesia. La extensión es de una cuadra por lado permitiendo de esta manera conocer los alrededores y tener una perspectiva más real de los establecimientos y edificios que se encuentran en las inmediaciones de la Capilla.

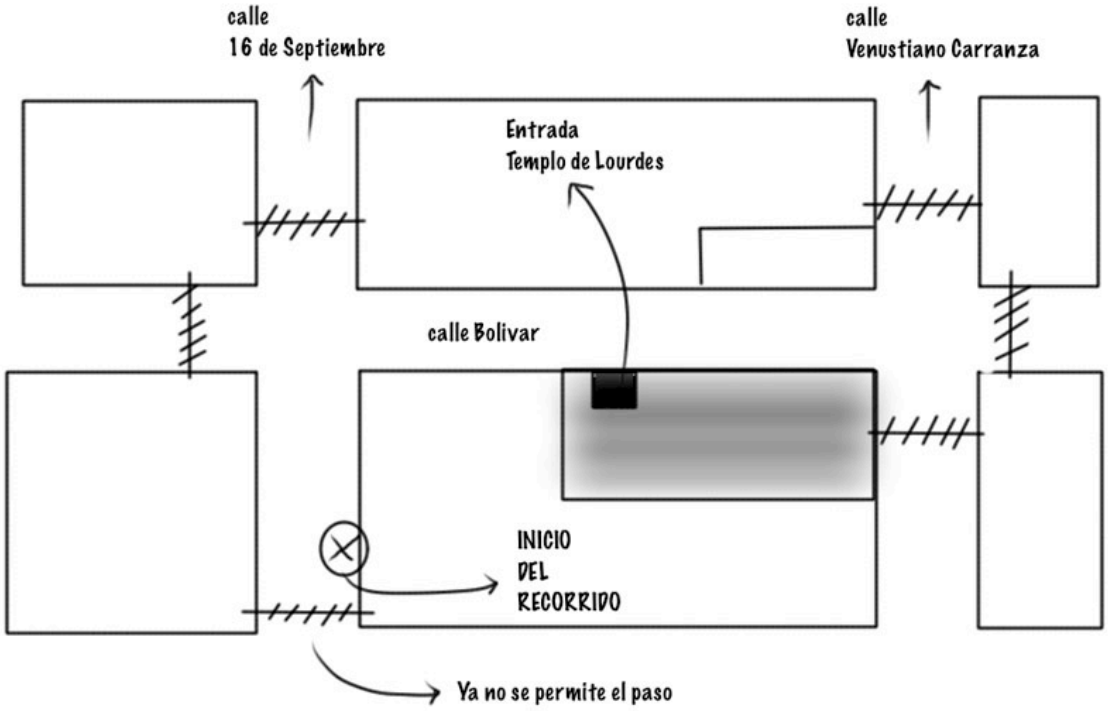
El recorrido por el interior del Templo de Lourdes muestra las características y disposición actuales de los elementos decorativos que se encuentran en la Iglesia.

Las dimensiones son las mas acercadas a la realidad del entorno, permitiendo al usuario tener una experiencia en el recorrido lo mas verídica a al ambiente.

Al inicio del recorrido el usuario se situará a la salida del Club de Banqueros sobre la calle 16 de Septiembre y podrá recorrer la parte Este de la misma que tiene dirección a la calle de Bolívar. Sobre la calle de Bolívar podrá recorrer toda la calle hasta la siguiente cuadra que es Venustiano Carranza. A la mitad de la calle de Bolívar se encuentra la entrada al Templo de Lourdes. Al ingresar a la Iglesia se activa la siguiente misión, correspondiente al interior del Templo de Lourdes.

No se podrá recorrer todo el contorno del Templo de Lourdes. En el mapa siguiente se detalla el camino que estará disponible para el usuario y los lugares: de inicio del usuario y la localización del Templo de Lourdes en el mapa de la primera Misión.

RECORRIDO VIRTUAL POR EL TEMPLO DE LOURDES
CENTRO HISTÓRICO, CIUDAD DE MÉXICO
Versión 1.0



La escala representativa es 1:1 con respecto a la altura del avatar utilizado en el recorrido que es 1.70 m. La escala que se utiliza es lo mas cercana a la realidad, pero puede haber variaciones en ciertos objetos del entorno.

Las condiciones meteorológicas están dispuestas para representar un clima de mañana, entre las 10 y 12 del día en la zona ya detallada, incluso la orientación de los edificios y la posición del sol están situados de acuerdo a la realidad.

La física que se representa en el recorrido por el Templo de Lourdes esta diseñada para el comportamiento que cualquier persona debería tener en la vida real. El caminar y correr están adaptados para características comunes, no permitiendo la posibilidad que el usuario pueda saltar durante el recorrido.

RECORRIDO VIRTUAL POR EL TEMPLO DE LOURDES
CENTRO HISTÓRICO, CIUDAD DE MÉXICO
Versión 1.0

8. Audio

Para las sesiones de audio dentro del recorrido se prefiere música con toques novohispanos, arpas o violines y de estilo ambiental. Para los menús se pretende colocar un estilo de música ambiental con toques de cuerdas. Para el video música sintética pero que permita una lectura suave durante la duración del video.

Para el entorno ambiental se harán grabaciones de audio dentro del mismo ambiente real que engloba el recorrido, tratando de evocar lo más fielmente posible las sensaciones y sonidos persistentes en el lugar. Para los efectos de sonido, se procurara hacer grabaciones reales de la acción que se está realizando. Para tales efectos se utilizarán grabaciones por medios físicos como video o capturadora de sonido o en su efecto se utilizará sonido digital.

9. Especificaciones Técnicas

Para la elaboración del recorrido por el Templo de Lourdes se necesitan equipos con sistema Operativo Windows y Macintosh. Por lo menos uno de cada sistema.

Las características mínimas son:

- Procesador 2 GHz
- 2 GB RAM
- Acelerador gráfico nVidia o ATI con funciones 3D de transformación e iluminación a 256 MB.

El software que se empleará para la compilación en sistemas Windows es: Microsoft Visual C++. Para los sistemas Macintosh: Xcode.

El programa para la construcción del entorno virtual es Torque Game Engine licenciado a 100 dólares, con libre acceso para instalar en ambos sistemas operativos.

Con respecto al demás software, como para la construcción de modelos 3D, Diseño Gráfico y producción de audio, se dejará al equipo de trabajo escoger la mejor opción.

RECORRIDO VIRTUAL POR EL TEMPLO DE LOURDES
CENTRO HISTÓRICO, CIUDAD DE MÉXICO
Versión 1.0

El formato de entrega del recorrido por el Templo de Lourdes será por medio físico, ya sea en CD, USB o algún dispositivo de almacenamiento masivo o descargable a través de Internet. Esto significa que la aplicación no tendrá un tamaño mayor a 700 MB, permitiendo su grabación en un CD. Meta a la que se debe llegar al finalizar el proyecto.

10. Apéndice

Las referencias técnicas avanzadas pertenecientes a los programas de diseño, programación y construcción del recorrido podrán ser consultadas en las siguientes ligas.

Microsoft Visual C++
http://msdn.microsoft.com/en-us/visualc/aa336395

Xcode
http://developer.apple.com/support/xcode/

Torque Game Engine
http://www.garagegames.com/documentation/tge

Torque Game Engine Developers
http://tdn.garagegames.com/wiki/

Las omisiones o características adicionales o faltantes deberán ser consultadas con el líder de proyecto a cargo del equipo de trabajo. Dichas omisiones deberán anexarse en un Apéndice al presente Documento para ser consultadas y aprobadas por las dependencias participantes.

El documento anterior es un ejemplo de cómo podemos escribir nuestros documentos de diseño, aunque no se describe a gran detalle las funciones y características presentes en la aplicación, el documento brinda un panorama general del recorrido a realizar.

Recordando un poco, se comentó que las especificaciones técnicas, aunque estrictamente hablado no pertenecían al documento de diseño, las hemos mencionado para recordar que es necesario tenerlas presentes. Las especificaciones técnicas pueden mencionarse en un apéndice que hace referencia a un documento anexo o incluso alguna página en internet.

ES SEGURO QUE SEGIREMOS ESCRIBIENDO NUESTROS DOCUMENTOS DE DISEÑO EN PAPEL, AL MENOS LA PARTE QUE ESTARÁ DIRIGIDA A NUESTRO CLIENTE, PERO ES NECESARIO EMPEZAR A UTILIZAR HERRAMIENTAS DIGITALES COMO LOS 'WIKIS' O LAS HERRAMIENTAS EN LA NUBE, POR EJEMPLO, LAS DE COLABORACIÓN EMPRESARIAL Y GESTIÓN DE PROYECTOS.

También se omitió la parte de definición de tareas y tiempos en el proceso creativo y de programación. Por condiciones del proyecto no es necesario realizar dicha labor.

4.2 ARQUITECTURA DEL RECORRIDO

Este es un tema pequeño pero es necesario tener una breve descripción de la arquitectura inicial de Torque Game Engine y del por qué esta organizado de dicha manera.

Antes de empezar cualquier proyecto es necesario tener definida la arquitectura de nuestra aplicación. La organización es vital par poder identificar de manera más rápida elementos, programas, módulos de ejecución, hasta posibles fallas que puedan pasar por alto.

Recordando, La arquitectura de Torque Game Engine es de tipo Cliente-Servidor, es decir siempre habrá un módulo actuando de manera remota, incluso si la aplicación es ejecutada de manera local.

Al abrir la carpeta de Torque Game Engine, observamos la siguiente disposición:

Organización de TGE	
/	Es el directorio raíz y representa el directorio desde donde el ejecutable estará colocado para correr. Este es el directorio base visible para TGE y los scripts.
/ common	Este directorio contiene archivos comunes entre los productos desarrollados en TGE. La intensión de este directorio es tener programas, imágenes, modelos, etc., que puedan ser reusados al menos en prototipos, incluso en los productos finales.
/ creator	Este directorio contiene las herramientas adicionales proporcionadas por TGE para la elaboración de GUI's y entornos gráficos.
/ tutorial.base	Este es el directorio del recorrido que incluye todos los scripts, imágenes, modelos, etc., que serán empleados en la aplicación.
/ tutorial.base / client	Esta carpeta contiene todos los elementos que pertenecen a las interfaces, elementos gráficos, preferencias del cliente, archivos y scripts que corresponden al entorno del usuario.
/ tutorial.base / data	Esta carpeta contiene todos los elementos que estarán dispuestos a lo largo del recorrido, modelos 3D, texturas, contenido de las misiones, elementos de los terrenos, audios o sonidos, videos, etc.
/tutorial.base / server	Este directorio contiene los scripts relacionados directamente con el recorrido. No confundir con funciones scripts que son implicaciones del cliente (usuario, carpeta 'client')

Figura 4.1

Arquitectura (Directorios) TGE

La organización de los directorios anteriormente descrita, es a partir de la disposición que sugiere TGE para los productos hechos con el motor. Como se ha mencionado, la arquitectura depende de la experiencia que se tenga, de manera que si en cierto momento vemos que dicha organización no cumple con los requisitos que estamos proponiendo, podremos modificar la distribución de las carpetas, eliminar un par de ellas, hasta poder agregar carpetas que aumenten significativamente características del recorrido final, incluso hasta proponer una organización propia.

Los archivos main.cs que se encuentran justamente en cada una de las carpetas significativas: /(raíz) , / tutorial.base, se consideran como los módulos de control de cada carpeta. Es decir, en ellos encontraremos scripts y funciones de carga para los elementos significativos de la carpeta a la cual están asignados.

Por ejemplo, main.cs de la carpeta raíz, la primer línea significativa de código hará referencia a la carpeta que contiene nuestro recorrido: $defaultGame = "tutorial.base"; ¿Cuál es el significado de tutorial.base?, es la carpeta que contiene un proyecto en blanco para la creación de cualquier tipo de producto que deseemos desarrollar en TGE. Contiene la base de los scripts necesarios para el inicio de nuestros recorridos. Como ya sabemos, podemos modificar para la arquitectura básica de TGE para empezar desde cero la programación de nuestro recorrido o utilizar el prototipo base para la elaboración del mismo.

Dentro de la carpeta ‘tutorial.base’ encontramos otro archivo main, dicho archivo contiene scripts relacionados con la carga de la misión inicial de nuestro recorrido, inicializan los scripts de la carpeta ‘client’, la carpeta server, así como las interfaces de usuario.

La arquitectura base que se empleará para el recorrido por el Templo de Lourdes se basa en la propuesta de TGE con una seria de adecuaciones específicas para una mejor organización perteneciente al recorrido. A continuación se presentan las adecuaciones:

Arquitectura base para el recorrido por el Templo de Lourdes	
/	Directorio raíz.
/ common	Carpeta para reutilización de scripts.
/ creator	Proporciona las herramientas adicionales.
/ TSTK	Paquete de scripts adicionales que proporcionan características especiales de manejo de archivos y funciones matemáticas.
/ lourdes	Carpeta perteneciente al recorrido y contiene todos los modelos, interfaces, elementos gráficos, archivos y scripts.
/ lourdes / client	Archivos que pertenecen al entorno del usuario.
/ lourdes / data	Archivos de modelos, audio, elementos gráficos que pertenecen al recorrido.
/ lourdes / fotos	Fotos tomadas por el usuario.
/ lourdes / notas	Notas escritas por el usuario.
/ lourdes / server	Archivos relacionados directamente con funciones del servidor.
/ lourdes / client / interfaces	Elementos gráficos y scripts relacionados con las interfaces de usuario.
/ lourdes / client / scripts	Scripts de funciones pertenecientes al usuario.
/ lourdes / server / scripts	Scripts de funciones que se ejecutarán del lado del servidor.

Figura 4.2
Arquitectura del recorrido por el Templo de Lourdes

La arquitectura que se presenta es la base con la que se empezará a trabajar el recorrido, no significa que a lo largo del desarrollo no haya cambios o se agreguen mas carpetas para conveniencia de la aplicación.

Con la arquitectura ya dispuesta se puede empezar la creación del recorrido por el Templo de Lourdes, ya conocemos la estructura general de las carpetas y a medida que vayamos desarrollando el producto sabremos dónde colocar cada uno de los elementos y scripts pertenecientes a la aplicación.

4.3 DISEÑO GRÁFICO, MODELADO 3D Y PROGRAMACIÓN (RECORRIDO EXTERNO)

Recordado un poco, hemos visto que el Diseño Gráfico es una forma de comunicación en donde se vea envuelta la imagen, los colores, cada elemento visual que compone la escena. Crear un ambiente visual por medio de imágenes lo mas real posible es la meta que tenemos que cumplir.

A lo largo de este tema se combinarán los procesos creativos de diseño gráfico y modelado 3D. Esto se debe a que las tareas no involucran únicamente un paso a la vez, si no que la construcción primaria requiere que los dos puntos sean llevados a la par.

4.3.1 CREACIÓN DEL ENTORNO VISUAL

¿Por qué empezar con la parte visual y no la de programación? En la mayoría de las aplicaciones empresariales es muy común empezar por la parte de programación y dejar al final la parte visual. Lo más importante para los equipos de desarrollo es tener al menos, el esqueleto de la aplicación funcionando y el diseño vendrá después, incluso resulta ser en algunos casos la parte final del proyecto.

Retomando la pregunta anterior, no debemos olvidar que Torque Game Engine ofrece un par de herramientas excepcionales para la creación de nuestras misiones y apariencia de nuestro recorrido. Y que mejor que empezar con la visualización de nuestro entorno utilizando las herramientas que nos proporciona TGE sin olvidar que la parte de script corre por cuenta de la aplicación.

Consultando el Documento de Diseño observamos que la primera misión se establece en los alrededores del Templo de Lourdes, para ser precisos, entre las calles de 16 de Septiembre y Venustiano Carranza y la longitud es de casi la cuadra completa. Para hacer el levantamiento habrá que ir al lugar y tomar fotos de los alrededores y utilizarlas como texturas para las fachadas de las calles.

El Documento de Diseño indica que el recorrido será entre las 10:00 y 12:00 de la mañana, valdrá la pena tomar las fotografías durante dicho periodo de tiempo. Aunque el motor también simulará el entorno climático de esas horas, no hay que perder la oportunidad que nuestras imágenes sean lo mas fieles a la realidad. Hay que señalar que no necesariamente hay que tomar fotografías a esas horas, con un buen trabajo en la imagen, se puede arreglar para dejarla en óptimas condiciones para el recorrido, pero haciéndolo de la manera antes descrita, facilitaremos el trabajo de los artistas que tendrán que preparar las imágenes para el recorrido.

4.3.2 PROCESO DE LEVANTAMIENTO

Para la creación de las misiones usaremos la herramienta 'World Editor'. Para ello deberemos tener nuestra copia de TGE en nuestro espacio de desarrollo.

Antes de continuar eliminaremos de nuestra copia las carpetas y archivos innecesarios, dejando únicamente los que ya han sido puntualizados con respecto a la arquitectura del recorrido por el Templo de Lourdes. Como la aplicación estará diseñada para los sistemas operativos Windows y Macintosh, valdrá la pena también tener el ejecutable del otro sistema en la misma carpeta.

Al abrir la aplicación llamada Torque Demo se abrirá la pantalla principal del motor, mostrando en pantalla links a las herramientas más comunes de TGE para empezar el desarrollo de nuestra aplicación.

Pantalla de Inicio

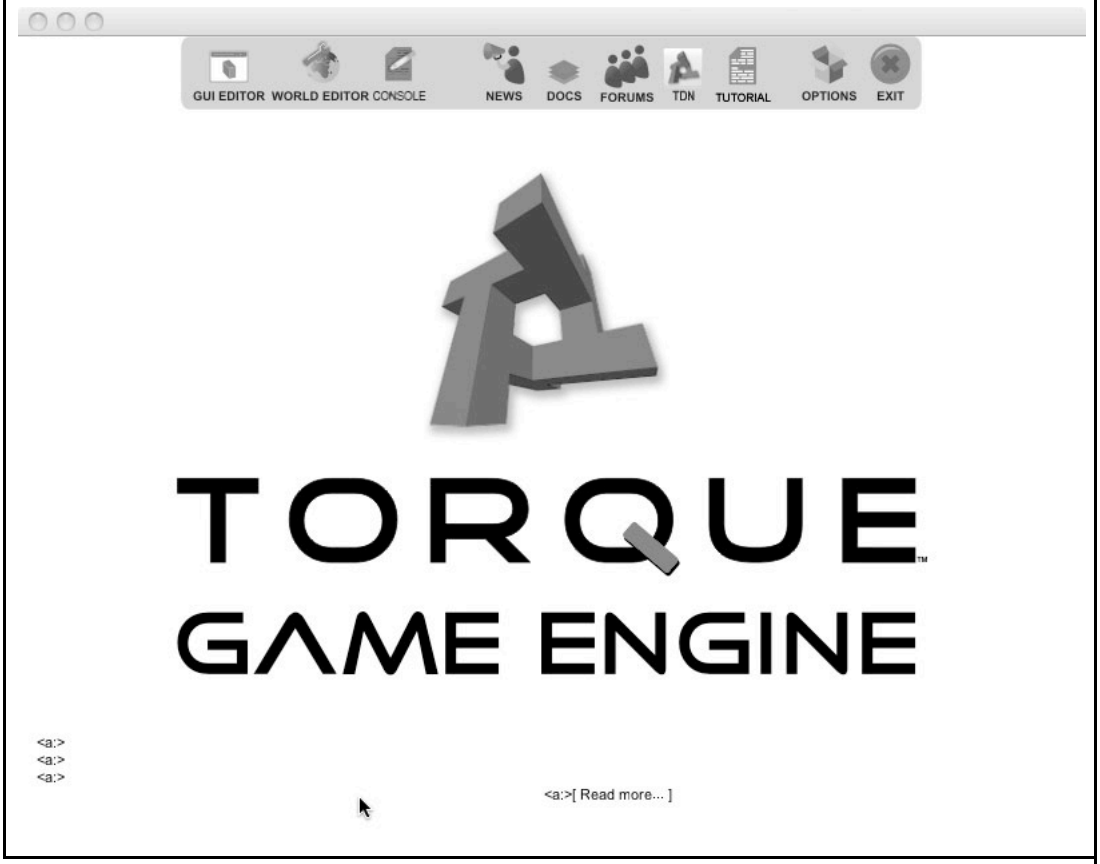


Figura 4.3
Prototipo inicial de TGE

En la parte superior encontramos los accesos para las herramientas 'GUI Editor' y 'World Editor'. Si la experiencia con Torque es inicial, se recomienda ampliamente empezar con el prototipo base que proporciona TGE.

Los procesos de análisis, identificación de herramientas y usos de las aplicaciones 'GUI Editor' y 'World Editor' quedarán del lado del equipo que desarrolle la aplicación. En los siguientes capítulos únicamente se hará referencia

a las herramientas utilizadas para el desarrollo del recorrido por el Templo de Lourdes.

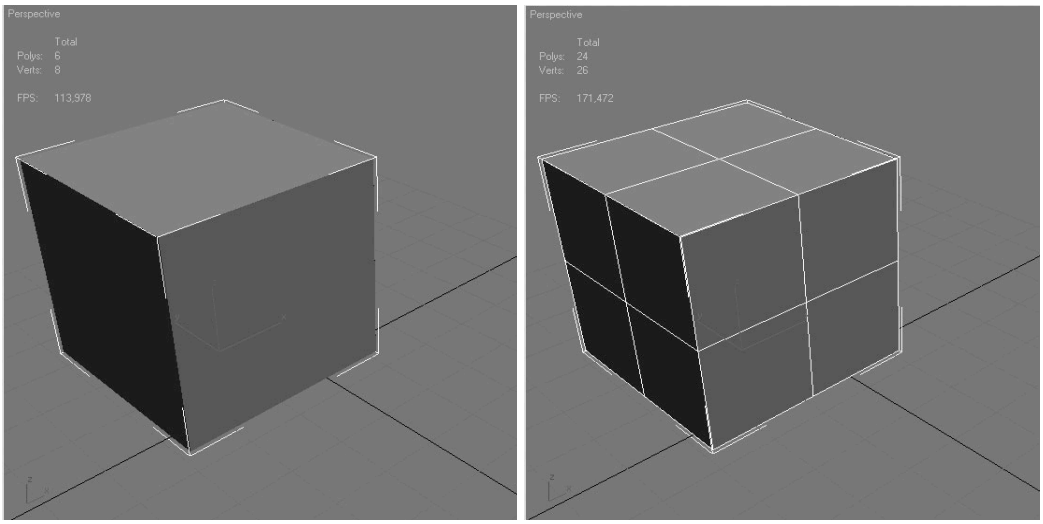
Para la creación de las fachadas que contendrá nuestro recorrido será a través del programa 3DS Max, programa de modelado para la creación de objetos 3D.

Hay que tener presente que en la creación de modelos 3D puede ser por medio de 'NURBs', 'mesh' (mayas) o 'polygon' (polígonos), debemos tener claro que TGE utiliza sus propios tipos de objetos para que puedan ser importados por el motor. En este caso, al diseñar nuestros modelos en 3DS max, deben estar hechos en 'mesh' o 'polygon object' para que la importación sea correcta.

Otro punto importante es que no nos podemos dar el lujo de hacer modelos con un gran número de polígonos, por ejemplo, un cubo tiene 6 caras, si cada cara es un cuadrado, podemos decir que esta hecha con 6 cuadrados, es decir con 6 polígonos. El número total de la imagen es 6 polígonos. Pero si cada cuadrado lo dividimos en cuatro partes iguales, cada cuadrado estrictamente estará hecho por 4 cuadrados, de manera que nuestro cubo ahora estará hecho por (6 x 4) = 24 polígonos. Esto solo índica que estamos haciendo una figura con un excesivo e innecesario número de polígonos. El cubo se verá de la misma forma y sin perder detalle ya sea con 6 o 24 polígonos.

¿Por qué nos debe importar el número de polígonos? Para TGE no hay límite de polígonos para un objeto, de manera que podemos diseñar un objeto 3D con un número de polígonos indeterminado. Sin embargo no siempre tendremos la seguridad que nuestro desarrollo se ejecutará en equipos con aceleradores gráficos. Esto significa que a mayor número de polígonos existentes en nuestro modelo 3D, la carga de procesos para el acelerador gráfico o procesador se multiplicará considerablemente, provocando que el equipo que está ejecutando la aplicación pueda demorar su carga o incluso cerrar la aplicación.

Conteo de polígonos



En la figura del lado izquierdo se muestra un cubo con las caras sin dividir, el número de polígonos es 6; mientras el cubo del lado derecho, cada cara ha sido dividida en 4 partes, de manera que el conteo de polígonos sera de 24.

Figura 4.4
Conteo de polígonos

Si estamos seguros que no siempre nuestra aplicación se ejecutará en equipos con un acelerador gráfico potente, capaz de renderizar completamente todo el ambiente creado, deberíamos pensar en detalles como calidad de imágenes y número de polígonos para nuestros objetos en entorno virtual.

Para evitar andar con los ojos vendados en este aspecto, TGE nos propone una lista dónde nos describe el tipo de objeto y un posible número de polígonos con los que debería contar el modelo.[99]

[99] garagegames.com. http://tdn.garagegames.com/wiki/DTS/FAQ/General_FAQ (ví: 2 de Septiembre de 2011)

Número de polígonos por objeto	
Personajes	2250 polígonos
Vehículos	1500 polígonos
Objetos para personaje	500 polígonos
Objetos generales	400 polígonos

Figura 4.5 Polígonos por Objeto recomendados por TGE

Aunque la lista anterior hace referencia directamente a lo propuesto por TGE, en el mercado ya contamos con equipos que al menos cumplen inmediatamente con los requerimientos mínimos pedidos por el sistema. En el recorrido por el Templo de Lourdes se empleó para los modelos la siguiente tabla:

Número de polígonos por objeto	
Personajes	No aplica
Vehículos	2500 polígonos
Objetos para personaje	No aplica
Objetos generales	1000 polígonos

Figura 4.6 Polígonos por objeto en el recorrido por el Templo de Lourdes

Ya que tenemos la información necesaria para empezar el modelado de nuestros objetos que estarán presentes en el ambiente de nuestro recorrido, solo hace falta saber cómo es que debemos tratar las imágenes para que también sean compatibles con la importación en el motor.

Para el manejo de imágenes se utilizará el programa de Photoshop, el cual nos permitirá optimizar las imágenes tanto en dimensiones como en peso. Las imágenes que han sido tomadas están en alta calidad, lo cual es necesario optimizar las imágenes y fotografías para el proceso de adaptación bajo los

estándares que TGE nos exige. Sin olvidar que existen otras opciones de programas de edición de imágenes como PixelMator, InkScape, GIMP, por mencionar algunos.

Los formatos que soporta TGE son jpg, gif y png y deben estar escalados en potencia de 2, la dimensión mínima es 64 x 64, puede ser 128 x 128, 256 x 256 y en casos extremos 512 x 512. Aunque se pueden usar dimensiones como 1024 x 1024 no se recomienda, ya no por cuestión del hardware del equipo si no que TGE puede colapsar. Incluso se puede usar combinaciones como 64 x 256 o 128 x 512, siempre y cuando sean potencias de 2.

Con la información anterior ya es momento de iniciar la tarea del modelado de las fachadas colindantes con el Templo de Lourdes. Hay dos opciones para modelar las fachadas, diseñar una fachada con detalles como volados, volúmenes y adornos pertenecientes a ellas o utilizar las bondades de la fotografía y simplemente utilizar un modelo plano.

Probablemente detalles en las fachadas no acarrearían un aumento considerable en el número de polígonos en nuestros ambiente, pero habrá que hacer un estudio evaluativo de cuantas fachadas se colocarán y si es necesario tener tanto detalle en ellas. Considerando que el recorrido debe ser capaz de ejecutarse incluso en equipos que no tengan un acelerador gráfico los esfuerzos deben centrarse en crear modelos lo mas reales posibles con un mínimo número de polígonos en ellos. Por tal razón se optó por crear fachadas planas y texturizar los modelos con las fotografías ya tratadas para la correcta importación al motor.

Solo basta decidir el formato de exportación para TGE, tenemos dos tipos de modelos soportados por el motor, DIF y DTS. En el caso del recorrido por el Templo de Lourdes , aunque las fachadas pueden considerarse objetos estructurales y ser exportados como DIF, hay que recordar que los objetos DIF

son para interiores y modelos que cuentan con un alto nivel de detalle. De manera que la exportación se hará con un formato tipo DTS.

Tanto para las exportaciones de tipo DIF como DTS hay que ser precavidos, tanto el modelo como las imágenes utilizadas por el objeto se encuentren en la misma carpeta y el nombre de la imagen no puede cambiar si ya ha sido exportado el modelo, es decir, si una imagen no se encuentra en la carpeta o la imagen después de la exportación cambió de nombre, el modelo mostrará áreas grises o tendrá una textura transparente en el lugar donde estaría representada la textura.

Hasta el momento no hemos tocado nada del motor para la programación del recorrido pero ya contamos con los modelos para que sean colocados en la misión correspondiente. El momento de meter mano a la parte del motor ha llegado. Recordando, la arquitectura de TGE es del tipo cliente-servidor, es decir habrá scripts, archivos, objetos tanto del lado del cliente como del servidor y no hay que olvidar que de alguna manera los archivos cercanos al cliente pueden ser modificados por el usuario. Aunque la aplicación esta diseñada para correr de modo ‘standalone’ es decir de forma independiente y no necesitará conectarse a un servidor remoto, ya que el servidor será simulado de forma local en el mismo equipo, hay que trabajar de la forma en cómo el motor funciona. La cuestión ahora es, ¿dónde y en que lugar se colocarán los objetos y scripts para la carga de los modelos ya diseñados?

Antes de decidir en qué lugar va cada elemento que nosotros diseñemos y programemos, hay que recordar que tenemos dos carpetas tanto en la arquitectura de TGE como en la arquitectura diseñada para el recorrido. ¿Qué significa tener esas dos carpetas llamadas ‘client’ y ‘server’? Aunque podemos ejecutar las aplicaciones creadas en TGE en una maquina sin la necesidad de que se conecten a un servidor remoto, TGE siempre actuará bajo el estándar cliente-servidor, como ya se dijo, simulará un servidor al momento de inicializar la

aplicación. De manera que en la carpeta de 'client' tenemos todos los archivos o scripts que pertenecen al usuario: imágenes, funciones, archivos, etc. Mientras que en la carpeta 'server' tenemos todos los scripts, funciones, objetos, etc., que estarán bajo control del servidor. Por ejemplo, si se realizara una aplicación donde se simulara el recorrido por un museo, y todas las obras de arte estuvieran en la carpeta del cliente, seguramente el cliente podría manipular las imágenes creando sus propias obras de arte y recreando un recorrido bizarro que no correspondería a las obras de arte que existieran en el museo. O por ejemplo, si el script de carga de objetos estuviera del lado del cliente y el usuario se dedicara a re-estructurar la programación de carga de objetos, seguramente los objetos desaparecerían o aparecerían unos en lugar de otros. Tener las carpetas 'client' y 'server' es para identificar que scripts y archivos estarán del lado del servidor y que otros del lado del cliente. Aunque nuestro desarrollo no comprende esta parte de programación, es necesario mencionarla para tener presente la forma de trabajo con TGE.

Con el ejemplo anterior ya tenemos una mejor idea de cómo ir estructurando y organizando los elementos dentro de la arquitectura de nuestro recorrido. En este caso tanto el script de carga de objetos como los objetos no deberán estar del lado del cliente, si no del lado del servidor o en una carpeta especial para objetos dentro del recorrido.

Revisando la arquitectura propuesta para el recorrido por el Templo de Lourdes, se observa que hay una carpeta llamada 'data' y se menciona que esa carpeta está destinada para todos los archivos multimedia de la aplicación. De manera que todos los modelos y objetos que serán colocados en las misiones del recorrido irán dentro de dicha carpeta. Mientras que los scripts de carga de los objetos, no tenemos duda alguna de colocarlos del lado del servidor.

Para una mejor organización del recorrido se creó una carpeta llamada 'shapes' la cual contendrá todos los objetos que se agregarán en las misiones exteriores. En está carpeta estarán los modelos de las fachadas de los edificios,

elementos decorativos como carros, árboles y demás objetos existentes en la periferia del Templo de Lourdes.

De la misma manera como se procedió para la recolección de imágenes pertenecientes a las fachadas, se hará para los objetos que se encuentran en las calles cercanas al Templo de Lourdes, como lámparas, maceteros, árboles, etc. Para la misión principal se pretende que sea lo mas fiel al contexto real, por lo que hay que hacer una recolección exhaustiva de elementos que existen en el entorno. Todos los modelos creados para este fin serán enviados a la carpeta 'shapes'.

4.3.3 PROGRAMACIÓN

En este momento entramos directamente con el uso del motor y de las herramientas con las que contamos.

Básicamente nuestro programa es un recorrido virtual por el Templo de Lourdes que se encuentra en la ciudad de México; aunque muchas de las tareas para la elaboración de la aplicación pueden llevarse simultáneamente con un buen estudio y división de tareas, se platicarán los pasos consecutivos que se fueron planeando para la elaboración del recorrido.

Aunque el documento hace mas énfasis a la parte gráfica del recorrido, también hay que tener presentes que existen muchas tareas asociadas a las funciones ahí descritas, por ejemplo, ¿cómo vamos a nombrar las notas o fotografías?, ¿con qué formato se presentarán las descripciones?, ¿cuándo se activan ciertas funciones del usuario?, etc. Todas estas características inherentes a las funciones las trataremos en éste tema.

Aunque ciertos scripts de programación se pueden llevar de manera conjunta con el levantamiento del ambiente del recorrido, lo haremos de manera

individual para cada paso de los procesos que se involucraron en la creación de la aplicación.

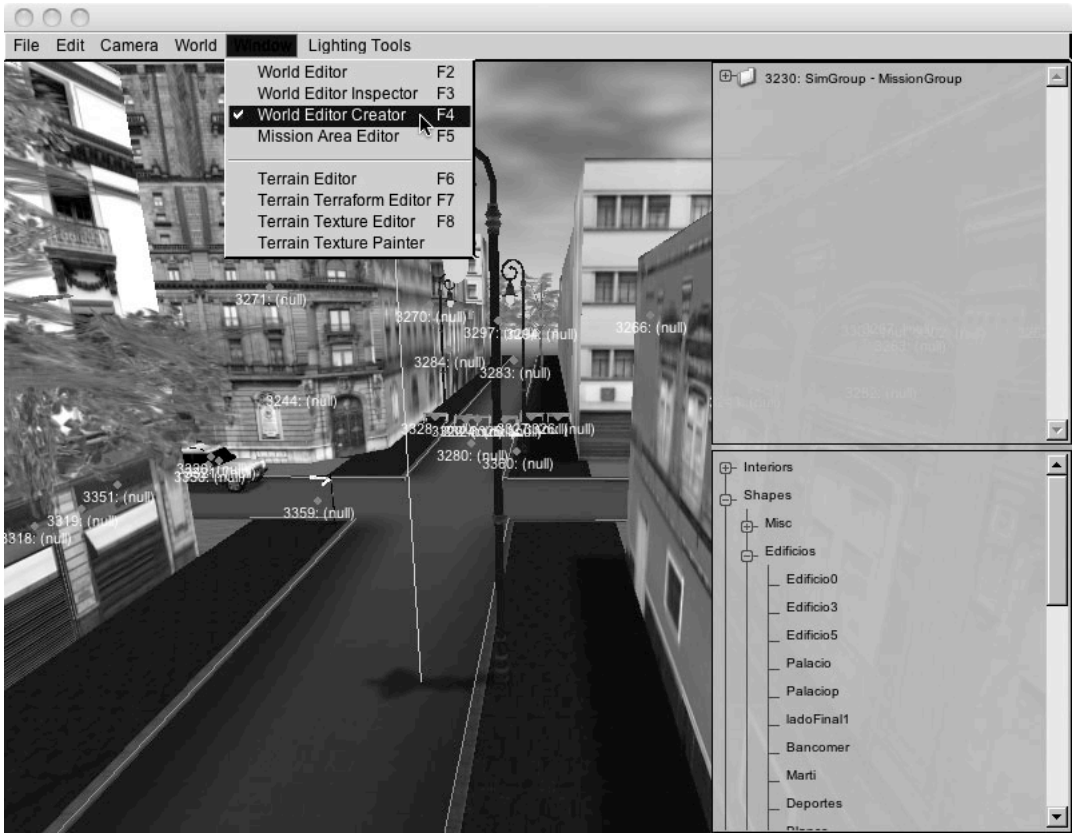
4.3.4 ELABORACIÓN DE LAS MISIONES

Empezaremos con el levantamiento de la escenografía colindante al Templo de Lourdes. Para ello con antelación ya se debió haber programado el script de carga para los ´shapes´ que se van a emplear como fachadas y tener ya almacenadas las texturas que se emplearán para los tipos de terrenos.

Para crear los entornos que contendrán nuestras misiones utilizaremos la herramienta 'World Editor', se activa pulsando el botón que tiene el mismo nombre en la parte superior o con la tecla F11. Ver Figura 4.7.

Para la ambientación no tenemos problemas de programación porque el motor se encarga de realizar esa tarea. Sólo depende el colocar los edificios de acuerdo a los mapas de zona y orientación deseada. Para la parte del terreno, la forma de trabajarlo es utilizar la opción 'TerrainBlock', que sea crea por omisión en cada misión nueva. Para modificar las características necesitamos seleccionar la opción 'World Editor Inspector' y seleccionar el bloque de terreno. En las opciones de configuración podemos escoger las texturas que han sido trabajadas para el terreno y se cargan en la opción de 'detailTexture'. Ver Figura 4.8.

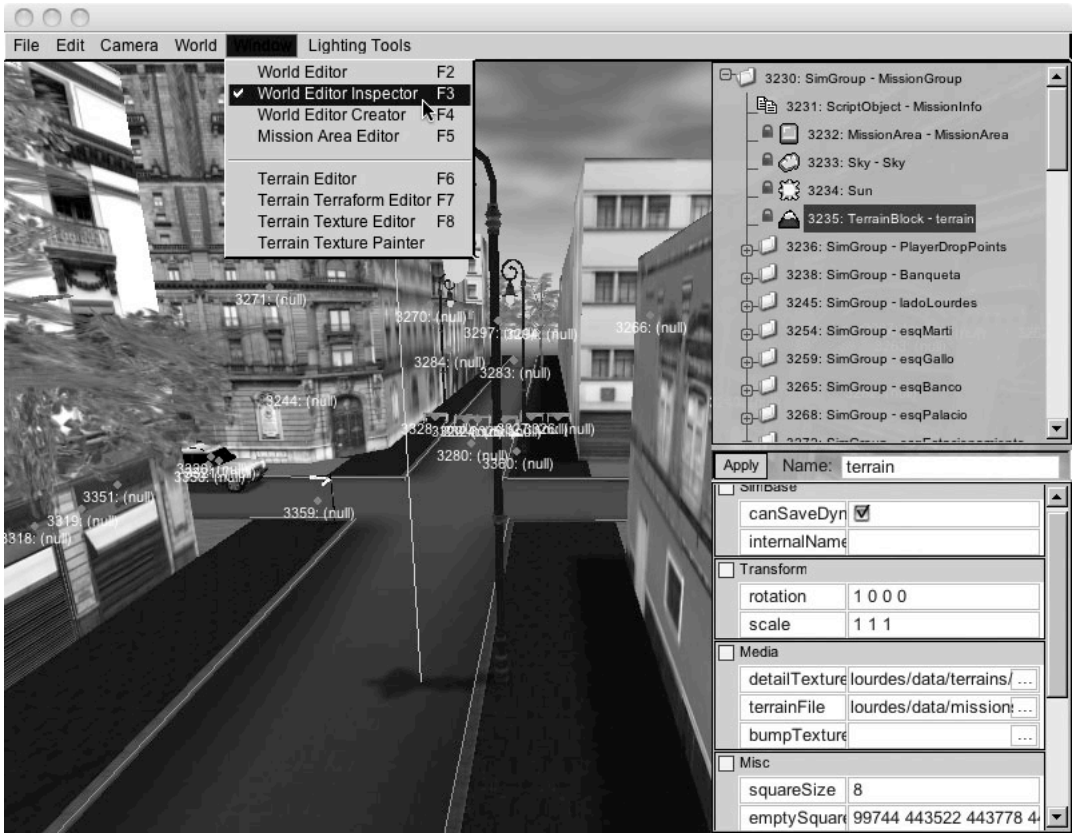
Pantalla del 'World Editor'



Del lado derecho tenemos una lista de todos los elementos que han sido cargados por el motor para armar el escenario.

Figura 4.7
World Editor

Edición del Terreno



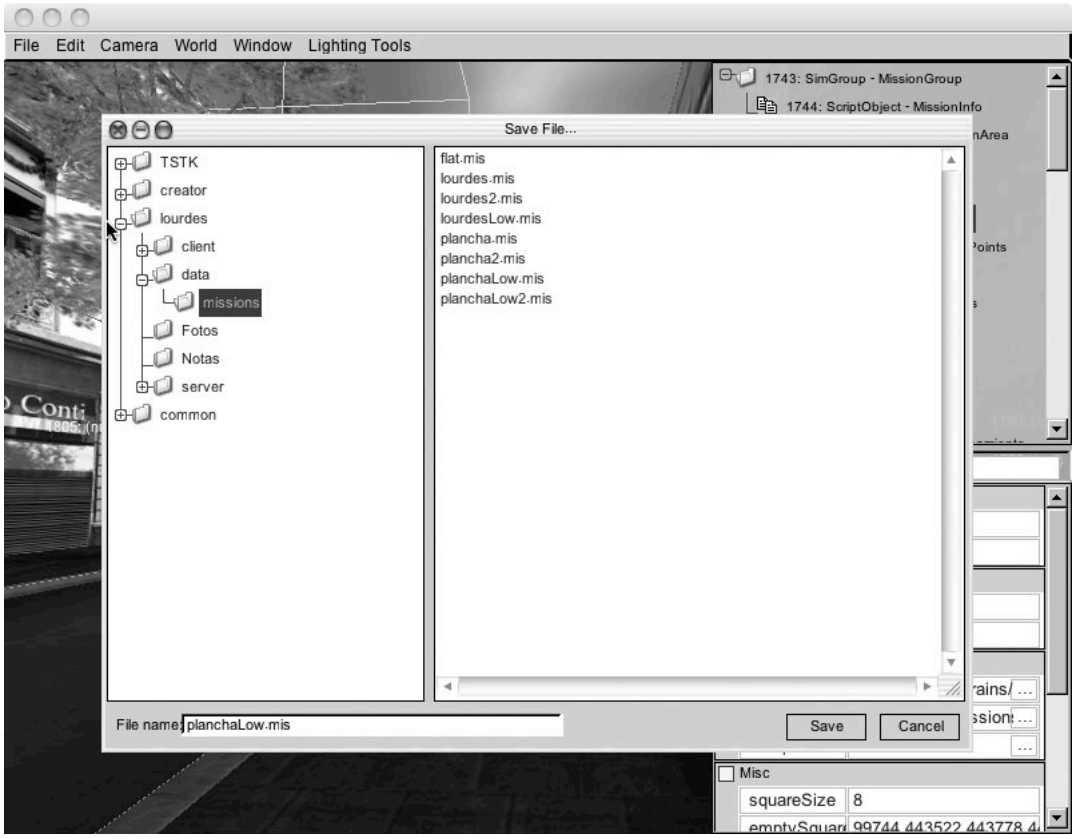
En la parte superior derecha se tiene la sección que muestra una lista de los elementos que contiene la misión actual. En la parte inferior tenemos las configuraciones que se pueden aplicar (incluso mas a nivel script) al objeto seleccionado.

Figura 4.8
Opciones para el terreno

El lugar mas adecuado para guardar los archivos de las misiones (.mis) es en la carpeta 'data' la cual contiene un folder llamado 'missions'. Al guardar la misión es necesario que sea con un nombre significativo de la misión porque este será necesario para identificar el tipo de ambiente que se simulará en el recorrido, para ello en el menú se elige la opción 'File – Save Mission As…'. De la misma

manera, para acceder a los archivos guardados de las misiones será con el botón F11 en el menú 'File – Open Mission'.

Archivos de Misiones



Se pueden crear mas de una misión y pueden ser guardadas en la misma carpeta sin ningún problema.

Figura 4.9
Archivos de las Misiones

Una misión se considera terminada cuando todos los elementos que debe contener la escena, así como las funciones que pertenecen a cada misión han sido creadas. De manera que el siguiente paso es empezar a programar las funciones que estarán disponibles para el usuario dependiendo la misión en la que se encuentre.

Aunque el proceso de la creación de las misiones y la programación de los scripts son tareas que se pueden elaborar de manera simultanea, no es recomendable ya que puede haber omisiones durante la elaboración de los scripts por falta de elementos. Por ejemplo, ¿cuál es el formato para los nombres de las notas o fotografías que se van almacenar durante el recorrido?, ¿cómo es el formato que se desplegará para la información de los objetos seleccionados?. Detalles como los anteriores puede generar errores durante la elaboración del recorrido.

SI AÚN NO SE CUENTA CON LA SUFICIENTE EXPERIENCIA EN TGE LO MAS RECOMENDABLE ES INICIAR CON LA CREACIÓN DE LAS MISIONES Y DESPUÉS CON LA PROGRAMACIÓN DE LAS FUNCIONES. POR DOS RAZONES: LAS FUNCIONES PODRÁN SER REVISADAS DURANTE LA MISIÓN Y LA SEGUNDA, SI EXISTE ALGUN ERROR EN EL RECORRIDO SE PUEDE IDENTIFICAR Y CORREGIR.

4.3.5 PROGRAMACIÓN DE FUNCIONES

Para este apartado hay recurrir al Documento de Diseño para conocer qué funciones corresponden a la misión que se esta diseñando. Para ello se puede recurrir a herramientas para el diseño de la programación, como los diversos diagramas que apoyan de una forma gráfica la construcción de los algoritmos que vamos a utilizar para las funciones.

Sin embargo, la elaboración de un diagrama escapa a las dimensiones del proyecto, ya que en la mayoría de los casos en los que se diseñan estas herramientas gráficas, para una entrada se tiene un conjunto de posibles decisiones con sus respectivas salidas. En este caso se tiene un conjunto de entradas (básicamente cada botón del teclado hace la función de una entrada) que están en espera de acciones, y por ser un simulador de eventos, cada entrada procesará, o la respuesta del valor anterior o estará a la expectativa del próximo

evento, por lo cual se dificulta encontrar una herramienta que facilite gráficamente la comprensión de los procesos que se llevan a cabo durante la ejecución del programa.

Para ello tenemos el Documento de Diseño que de una forma sintetizada y general hace referencia a los procesos involucrados durante el recorrido por el Templo de Lourdes, no obstante se detallarán mas adelante la estructura de dichas funciones.

Consultando el Documento de Diseño el menú principal contará con una barra superior y una opción que mostrará en pantalla las últimas noticias publicadas en el sitio web del Templo de Lourdes, esta última opción será a través de internet.

Antes de empezar con la programación de las funciones es recomendable preparar el aspecto visual de la pantalla inicial. Para ello utilizaremos la otra herramienta que provee TGE, el 'GUI Editor'.

El 'GUI Editor' provee varias módulos visuales que facilitarán la forma en que vincularemos los controles gráficos con los scripts que se programen. Aunque se tiene un número limitado de controles que podemos utilizar para las interfaces gráficas, son suficientes para las funciones comunes que podemos encontrar en las interfaces para el usuario.

La barra superior contendrá 3 controles principales: inicio, configuración y salir, y un módulo inferior que desplegará en pantalla información preveniente de internet. Para los controles superiores se utilizarán botones que llamarán a una función que ejecutará la tarea seleccionada, mientras que para el módulo inferior la tarea será un poco mas compleja ya que se tiene que utilizar herramientas adicionales como los 'web feed' o medios de redifusión de contenido web.

Los medios de redifusión de contenidos en páginas de internet se utilizan para suministrar información actualizada a clientes que estén suscritos a la fuente de información. En pocas palabras, cuando una página web difunde contenido o actualiza alguno, sus suscriptores recibirán en sus equipos o medios digitales de información las novedades que se han publicado, es decir, el cliente ya no necesita visitar la página para saber las ultimas noticias de algún sitio, si no que la página se encarga de enviarle la información reciente al suscriptor, ya dependerá del usuario visitar o no la página web, el cliente ya esta informado, aún sin haber visitado el sitio directamente.

Para el primer botón: inicio, basta nombrar la misión a la cual se desea acceder y esperar a que el motor cargue el recorrido. De ahí el porque se hizo tanto énfasis en considerar un nombre apropiado para las misiones, ya que si vamos a programar mas de una misión, dichos nombres deberán ir en el script de los botones de inicio. Para el botón: configuración, solo necesitamos un script que muestre una ventana emergente que contenga las opciones de gráficos, audio y controles. El botón: salir, cortará la conexión al servidor y de esa manera cerrar la aplicación.

El módulo inferior merece un poco de mas detalle ya que utilizaremos herramientas adicionales para mantener la información actualizada al momento que la aplicación se inicialice. Para ello utilizaremos el formato RSS, que no hay que confundir como fuente web. La fuente web es el medio de redifusión de la noticia mientras que RSS es el formato de dicha fuente. Entre los formatos mas comunes para las fuentes web tenemos el Atom y RSS.

Sería absurdo e innecesario programar toda la fuente web si ya existen instrumentos que lo hacen por nosotros, lo recomendable es utilizar estas herramientas para beneficio de la aplicación. Para ello hay que organizar en primera instancia, cómo se distribuirán las actualizaciones de la página y después conseguir un proveedor que gestione la información.

La página a la cual hay que enlazar la aplicación es:

http://www.mimeland.net/lourdes/

Haciendo un estudio de la arquitectura de la página y cómo se publican las noticias en el sitio, se decidió que el formato a emplear para la difusión será RSS y el gestor de la información será FeedBurner.

Se decidió utilizar FeedBurner porque ofrece ventajas admisibles para la aplicación, una de ellas y la mas importante es que cuenta con la redirección del 'feed' (fuente web), es decir, la dirección del 'feed' de la página es:

www.mimeland.net/lourdes/?feed=rss2

y ha sido redireccionado a:

feeds.feedburner.com/TemploDeLourdes

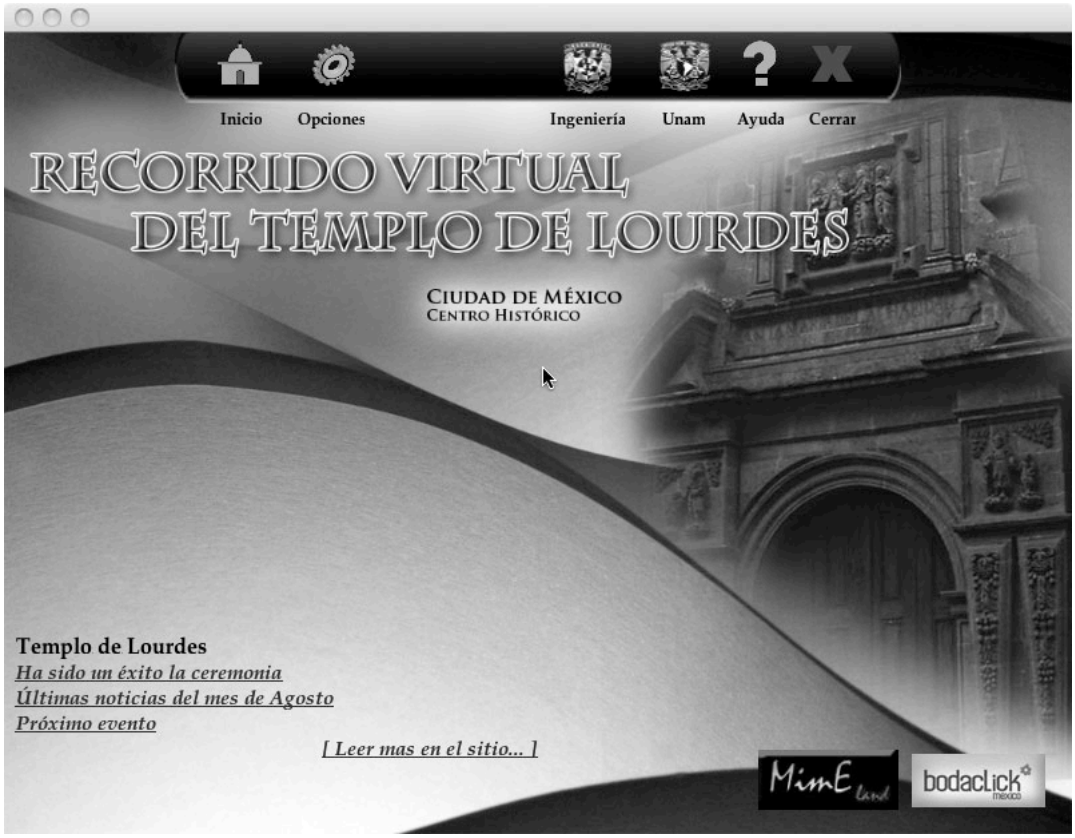
lo anterior permite controlar los formatos y obviamente la redirección de las actualizaciones de las noticias a donde nosotros deseemos, al recorrido por ejemplo, además de aminorar el tráfico de red empleado por la aplicación. Es más fácil y menos costoso recibir la información ya con un formato específico, que hacer un barrido completo de una página para encontrar noticias nuevas o saber cuáles han sido o no actualizadas.

De esta manera aseguramos que las noticias nuevas o actualizadas en la página del Templo de Lourdes sean enviadas a la pantalla principal del recorrido, cuando el equipo donde se está ejecutando la aplicación esté conectado a internet. De esta manera informamos directamente al cliente sin la necesidad de ingresar directamente al sitio web.

Con lo anterior ya es posible finalizar el proceso tanto de la parte gráfica como de los scripts necesarios para el menú inicial del recorrido.

Guiándonos por la propuesta inicial del Documento de Diseño el menú inicial queda conformado de la siguiente manera:

Menú Inicial



En la parte superior contamos con los botones para el inicio y configuración del recorrido, en la parte inferior un módulo que despliega en pantalla los títulos de las últimas tres noticias actualizadas de la página web.

Figura 4.10
Menú inicial del recorrido por el Templo de Lourdes

Para la opción del botón de inicio, la propuesta pide que se tenga la facilidad de escoger tres tipos de recorridos. El recorrido completo, el cual contendrá todos los elementos existentes en las inmediaciones del Templo de Lourdes, un recorrido alterno que únicamente cuente con los elementos indispensables para que el usuario pueda orientarse en los alrededores de la Iglesia y la última opción que permita entrar directamente al Templo de Lourdes.

¿Cuál es la finalidad de contar con tres recorridos?. Estamos de acuerdo que no todos los equipos serán capaces de ejecutar la aplicación en su modo completo y se pretende que el recorrido sea capaz de correr aún en un equipo con las características mínimas. Para ello se han elaborado dos tipos de recorridos por el exterior del Templo de Lourdes, uno de ellos con un nivel de detalle admisible que permita al usuario orientarse adecuadamente a los alrededores de la Iglesia, contando con la mayoría de los detalles presentes en el ambiente, esta misión solamente podrá ejecutarse sin problemas en equipos con aceleradores gráficos. El recorrido alterno cuenta con un nivel de detalle básico, como fachadas y algunos adornos existentes para que el usuario pueda reconocer el lugar cuando se encuentre ahí, este recorrido está planeado para equipos con tarjetas de video comunes. Mientras que el recorrido por el interior del Templo de Lourdes, no importando si el equipo cuenta o no con un acelerador gráfico, sea capaz de ejecutarse en el equipo sin problemas.

Aunque aún no se entra en detalle con estas propiedades con las que se tendrán que elaborar las misiones, es necesario ir pensando en los procesos de carga tanto para el motor como para el equipo del usuario.

La ventana emergente que aparecerá al momento de dar click al botón de inicio que se encuentra en el menú inicial de la aplicación tendrá la siguiente organización. En la parte superior un campo que permita al usuario escribir su nombre, de esta manera se podrá reconocer al usuario que esta utilizando la aplicación, por si mas de una persona utiliza el recorrido. Un texto de bienvenida que explique la finalidad del recorrido y brinde un panorama general del recorrido. Y tres botones que indiquen el tipo de recorrido al cual se desea ingresar, ya sea el recorrido principal, el recorrido alterno o directamente al interior del Templo de Lourdes.

Ventana - botón Inicio

Figura 4.11
Ventana emergente del botón Inicio

Con respecto al botón Opciones, los scripts serán mas robustos. Para la sección de Gráficos, debemos asegurar una correcta resolución para las interfaces de usuario y el recorrido, para ello el programa deberá hacer un análisis del sistema en segundo plano al momento de inicializarse la aplicación, a partir de ahí considerar las resoluciones óptimas tanto para la pantalla completa como en modo ventana y al finalizar mostrar una lista de posibles resoluciones de pantalla para que el usuario seleccione la que mas convenga. Aunque para cuando la lista se muestre, el programa será capaz de seleccionar la resolución mas adecuada para el equipo. Para ello se colocará un botón en la ventana de 'Opciones – Gráficos'

en la parte inferior de la ventana. Para la sección de Audio, no hay mucho que programar, basta un módulo con una barra que permita la selección de volumen de los sonidos ambientales, de los objetos y el volumen general. Para ello basta configurar las preferencias de sonido con una variable asociada a las barras de desplazamiento. Y por último, para la sección de Controles, se desplegará un cuadro el cual mostrará las teclas asociadas por omisión a las funciones con las que contará el usuario durante el recorrido. Para esta sección se necesita que los controles puedan configurarse de acuerdo a la comodidad del usuario y que al finalizar el recorrido sigan permaneciendo las preferencias seleccionadas por el cliente cuando el recorrido inicie nuevamente.

Los controles que desplegará la sección opciones serán: Una lista de las resoluciones de pantalla, la profundidad de bits con la que se ejecutara la aplicación, el formato con que se desea tomar las fotografías y la opción de pantalla completa o modo ventana. Un botón al final de la ventana que permita que al sistema seleccionar la resolución mas adecuada al equipo. En la sección de Audio, tres controles que permitan escoger el volumen de los sonidos de la aplicación y para la sección de Controles una ventana que muestre la lista de las funciones y las teclas asociadas con alguna opción de modificar la tecla asociada a la función.

Cuando se seleccione alguna misión, ya sea la principal o la alterna, antes de iniciar el recorrido todos los elementos que corresponden a la misión se cargarán desde un principio, de manera que hay que tener presente que entre mas elementos haya en el recorrido, mayor será el procesamiento de video.

Ventana - botón Opciones

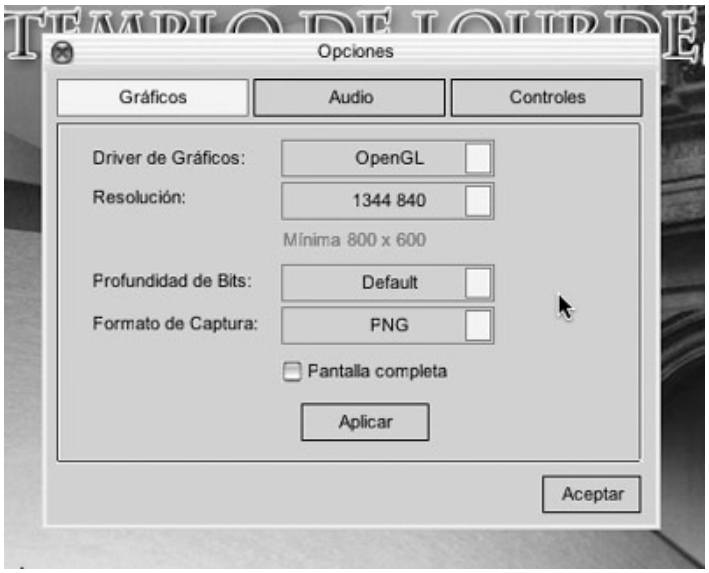


Figura 4.12
Secciones del botón Opciones

Cuando el usuario se encuentre dentro del recorrido tendrá a su disposición ciertas funciones, que dependiendo en qué misión se encuentre, ya sea en el exterior del Templo de Lourdes o en el interior, estarán disponibles para su uso. De manera que se empezará por programar las funciones que estarán activas durante el recorrido externo.

Antes de continuar es necesario definir concretamente la interfaz de usuario durante el recorrido. Basándonos en el documento de diseño, se propone que la interfaz gráfica tenga los controles mas comunes a disposición inmediata del usuario.

Del lado izquierdo habrá un módulo que desplegará en pantalla, el nombre de usuario, el tipo de objeto que está seleccionando y cierta información para uso del sistema. No todos los objetos que seleccione el cliente deberán aparecer en la

ventana de información. Aquellos que se consideren prescindibles no aparecerán nombrados en la ventana.

Para lograr lo anterior se necesitará de un modulo del 'GUI Editor' llamado 'CrossHairHud', este elemento se puede considerar como un punto de mira, el cual tendrá como objetivo reconocer el elemento al que se está apuntando y obtener datos del objeto como su tipo, su nombre y en ciertos casos saber si es un objeto colocado por el usuario o es un objeto del sistema.

Del lado derecho se encontrarán los servicios para el usuario, como el agregar notas, tomar fotografías, agregar objetos. Mientras que en la parte inferior se encontraran datos informativos sobre las notas y las fotografías. Y en la parte inferior derecha, botones para el manejo de la resolución y cambio de modo gráfico.

No obstante, se puede definir la interfaz gráfica del usuario antes de programar las funciones que estarán disponibles en esta misión. Además, asumiendo que dicha interfaz se utilizará para todo el recorrido se puede crear desde un principio antes de programar los scripts para cada uno de los elementos descritos anteriormente.

Las funciones básicas disponibles para el usuario durante todo el recorrido es la de escribir notas, tomar fotografías y escoger el modo ventana o pantalla completa.

Interfaz Gráfica del Recorrido

Organización de los elementos deseados de la interfaz de usuario durante el recorrido por cualquiera de las misiones de la aplicación.

Figura 4.13
Interfaz Gráfica durante el recorrido

Empezando por la función 'Notas', se necesita que las notas que se vayan escribiendo durante todo el recorrido estén disponibles para su lectura, incluso después que el recorrido haya finalizado. Para esta función hay que tener presente que necesitamos manejo de ficheros y el uso de cadenas de texto, entre algunas características mas.

Se necesita que el archivo esté disponible después que haya finalizado el recorrido, significa que debe estar almacenado para su posterior lectura. Y

tomando en cuenta que más de un usuario podrá utilizar la aplicación, inclusive el mismo día, hay que almacenar varias notas. También se puede dar el caso que el mismo día un mismo usuario utilice la aplicación mas de una vez y si el programa no es capaz de resolver este problema seguramente tendríamos notas duplicadas.

La forma mas adecuada de tratar las notas es por fecha, con eso aseguramos que no haya mas de un archivo por día. El siguiente punto es saber qué notas le corresponden a qué usuario; para solventar esta situación se ha colocado un cuadro de texto en la ventana de acceso a la misión para que el usuario que utilizará la aplicación escriba su nombre. Con los dos datos anteriores podemos solucionar el problema de saber a qué usuario le corresponde qué nota. Sólo faltaría escoger el tipo de archivo con que se guardarán las notas, así como el formato en cómo se escribirán.

Para empezar necesitamos un formato común de lectura / escritura entre los dos sistemas en que se va a ejecutar la aplicación (Windows, Macintosh). Podríamos evitar hacer la evaluación del formato programando un script privativo de la aplicación, haciendo que todas las notas se guarden en un archivo con extensión .nr (NotasRecorrido) y la aplicación pueda decodificar para su lectura y escritura. Pero el problema no es hacer una nota que pueda leerse desde la aplicación, si no que necesitamos un archivo que aún después de usar la aplicación podamos leerlos, incluso en cualquiera de los dos sistemas operativos. Existen varios programas comunes entre los sistemas operativos para edición y lectura de texto, unos de pago como la suite de Office que es capaz de leer archivos .doc o .docx ya sea en Windows o Macintosh o programas de código abierto como Open Office que también hacen lo mismo. El inconveniente es que estamos sujetos a que en el equipo donde se ejecute la aplicación esté instalado un software especializado para su lectura o escritura. Lo que se necesita es explotar al máximo las capacidades nativas con las que cuenta cada sistema operativo, para ello, un archivo de lectura / escritura común en los dos sistemas

son los archivos de texto .txt, además de que no se necesita software adicional para su manejo.

Faltaría únicamente el formato de presentación. Los archivos de texto .txt carecen de un formato enriquecido, además que al tomar notas no es necesario dar estilo al texto, pero si es importante resaltar las notas escritas. Considerando lo anterior se pensó en generar notas por línea o párrafo. Así denotamos la importancia de cada nota y su fácil localización.

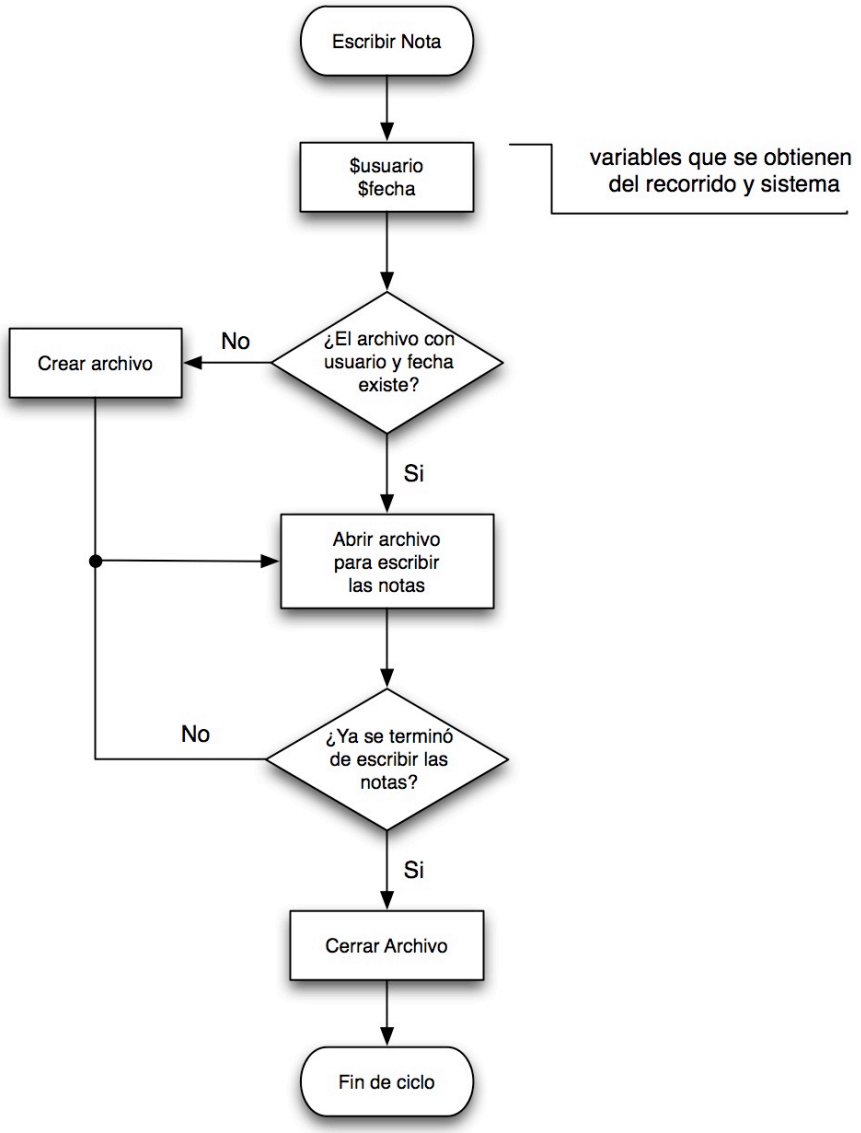
Con los datos anteriores ya es posible evitar los posibles errores que pueden ocurrir al generar los archivos, el título de las notas será de la siguiente manera:

Usuario_Fecha.txt

El formato anterior es facilitar la localización de las notas en la carpeta dónde serán almacenadas. Iniciar con el nombre del cliente para ubicar inmediatamente al usuario que escribió la nota y la fecha para saber cuándo fue escrita la nota. Si mas de un usuario usa la aplicación el mismo día, se generarán varias notas donde la fecha no se modificará pero el usuario si. Y de la misma manera evitamos generar notas duplicadas, ya que si la búsqueda de la nota se hace por día y usuario, y la nota ya existe simplemente se abre la nota y se sigue escribiendo al final del último renglón escrito.

En la siguiente página se muestra el diagrama de flujo para el proceso:

Diagrama de Flujo para las Notas



Proceso para la escritura de notas.

Figura 4.14
Diagrama de flujo para la función notas

Lo anterior sólo es la forma en cómo se procesará la creación del archivo de notas, falta la forma gráfica en cómo se recibirán y desplegarán las notas.

Al dar click sobre el botón ‘Notas’, se abrirá una ventana emergente desplegando en pantalla el contenido del documento notas, si no existe el documento, éste se crea con un título con el formato siguiente:

Archivo de notas de: USUARIO, con fecha: DÍA_MES_AÑO

… a partir del siguiente renglón se añadirán las notas que escriba el usuario.

En la parte inferior se colocará un botón o sección para que el usuario pueda agregar las notas que desee. Tomando en cuenta que es un archivo tipo txt y no tenemos la oportunidad de dar estilo a las notas, cada idea escrita por el usuario será separada por un salto de línea, para lograr lo anterior, al momento de escribir las notas se desplegara una ventana adicional donde se podrán escribir las ideas.

Se añadirá de igual manera en la parte inferior un botón que permita imprimir la nota por si el usuario desea tener una copia de sus notas en papel. El botón simplemente usará las bondades del sistema para abrir el archivo con la aplicación nativa y ya corresponderá al usuario imprimirlo desde el menú de la aplicación.

Cabe mencionar que si el usuario jamás selecciona el botón de Notas, ningún archivo será creado.

La interfaz para el control anterior se muestra en la siguiente figura:

Ventanas para las Notas

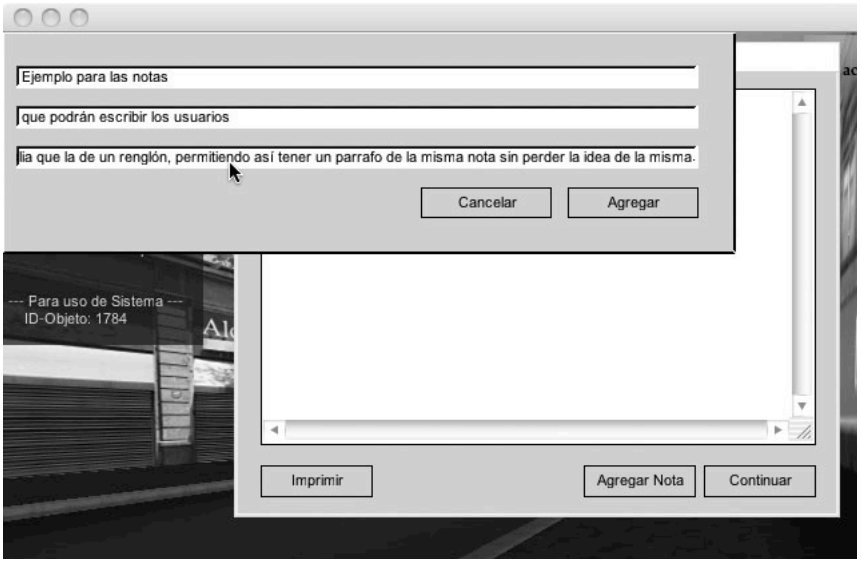


Imagen 4.15.a

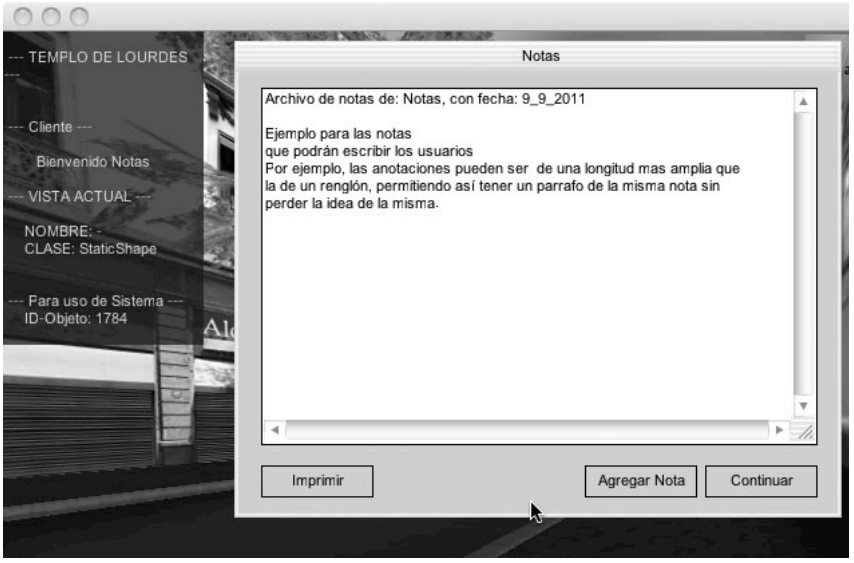


Imagen 4.15.b

En la imagen 4.15.a se observa la ventana para el ingreso de notas.
En la imagen 4.14.b la ventana de previsualización de las notas durante el recorrido con sus respectivos botones: 'agregar notas' e 'imprimir'.

Figura 4.15
Ventana de Notas

Para la siguiente función, la cual permite al usuario tomar fotografías durante cualquier momento en el recorrido, el procedimiento es muy similar al anterior.

Al dar click en el botón 'Tomar Foto', se abrirá una ventana emergente que indicará el procedimiento para tomar las fotografías, con la opción de quitar este aviso en las subsecuentes tomas fotográficas.

Durante las tomas fotográficas, la interfaz gráfica que permanece durante todo el recorrido será remplazada por una vista limpia del recorrido. Mientras el usuario permanezca en la toma de fotografías la interfaz gráfica permanecerá sin ningún elemento gráfico del menú.

Cuando el usuario tome una fotografía, se mostrará en pantalla un aviso que indicará que la foto ha sido tomada, en ese momento el cliente puede agregar una nota a la foto que ha tomado o seguir tomando fotografías. Si el usuario decide agregar una nota a la fotografía, en la parte inferior del aviso estará disponible un botón para agregar la nota. Al agregar las notas, en la parte inferior de la ventana aparecerá el nombre de la fotografía a la cual se le esta asociando la nota. Dicha nota aparecerá en el archivo de notas. Cabe mencionar que si en ese momento no se le agrega la nota a la fotografía, posteriormente ya no se podrá añadir.

El recorrido utiliza dos tipos de formato para guardar la imagen, jpg o png. La selección de este formato se hace antes de iniciar el recorrido en la parte del menú 'Opciones' en la sección de gráficos, por omisión el formato a utilizar es png. Las fotografías no se almacenarán en alta resolución, por dos razones: si se necesita enviarlas por internet ya están preparadas para ello o por cuestiones de espacio en disco del usuario.

De la misma forma en que las notas se almacenan para su posterior uso, las fotografías también se almacenarán en disco para uso del cliente.

El diagrama de flujo para la toma de fotografías se muestra a continuación:

Diagrama de flujo para fotografías

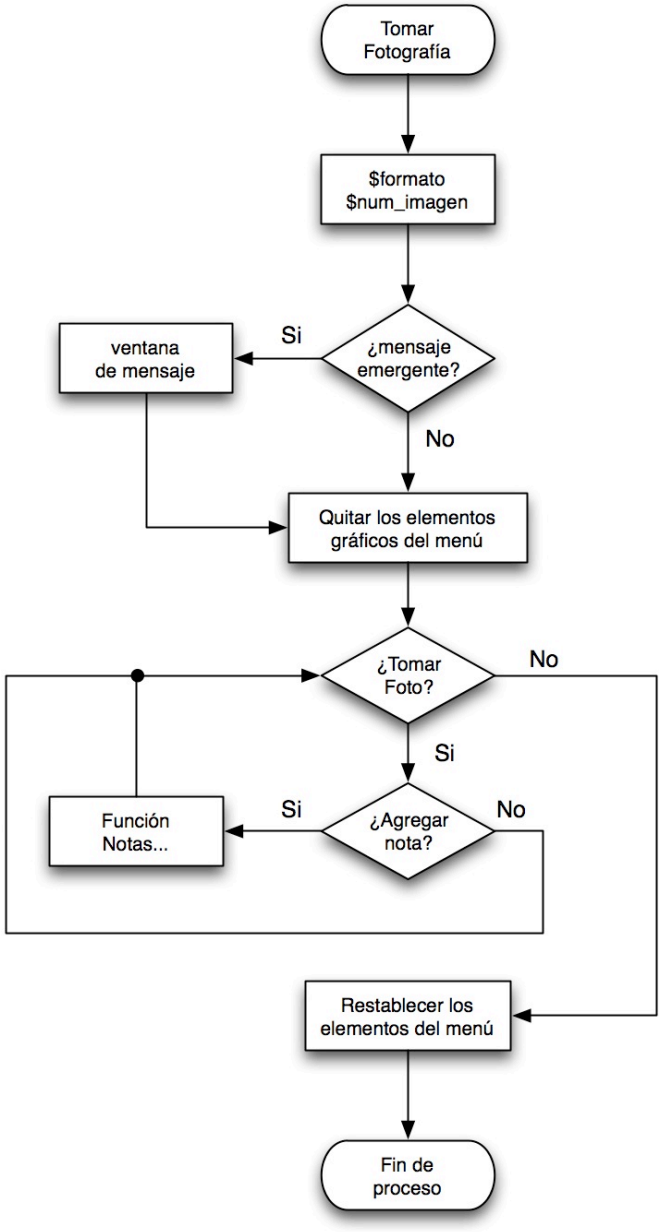


Figura 4.16
Diagrama de flujo fotografías

Ya que tenemos estas opciones funcionando, se puede iniciar la programación de los elementos que se encuentran en la parte inferior. Estos controles muestran la información en tiempo real de las notas y fotografías tomadas, incluyendo un acceso a los archivos que han sido almacenados.

Para estos controles se necesita un módulo que posee el 'GUI Editor' llamado GuiTextCtrl, el cual permite asignarle el valor de una variable y cambiar el dato cada ves que éste sea actualizado.

Este control servirá para mostrar en pantalla el número de notas que se han escrito y el número de archivos guardados en el equipo. Lo mismo para las fotografías, número de fotografías tomadas durante el recorrido y número de archivos de imagen almacenados.

Para el número de notas y fotografías tomadas durante el recorrido habrá funciones asociadas a cada elemento, llevando el conteo mientras se activen estos controles. Con respecto a los archivos almacenados es mas fácil ya que nada mas se necesitará una función que al iniciar la aplicación haga un conteo de archivos por carpeta, para asegurar que no se lean archivos ocultos o de sistema, solo se contarán elementos con extensiones txt, para el caso de las notas y las imágenes archivos jpg y png.

Los botones a lado de cada elemento de conteo llevarán directamente al usuario a la carpeta destino donde se encuentran los archivos almacenados para su posterior uso o eliminación.

Y del lado inferior izquierdo un control para llevar el conteo del tiempo de permanencia del usuario en el recorrido.

Controles inferiores

En la parte inferior se observan los módulos de conteo de archivos y el conómetro del lado iderecho.

Figura 4.17
Controles de conteo de archivos y tiempo

Con lo anterior se puede concluir con la primer etapa del desarrollo del recorrido. Quedando pendiente la segunda parte del recorrido, la misión mas importante que es el recorrido por el Templo de Lourdes. La cual ya no deberá ser tan difícil, al menos en el aspecto de diseño porque ya tenemos una experiencia previa con la realización de la misión del exterior del templo y la programación de algunas funciones previstas en el Documento de Diseño.

4.4 DISEÑO GRÁFICO, MODELADO 3D Y PROGRAMACIÓN (RECORRIDO INTERNO)

En esta sección se hablará de lo que es en sí, la parte fundamental de la aplicación, el recorrido por el Templo de Lourdes.

No era conveniente hablar inmediatamente del proceso de creación de la ambientación y programación del recorrido por el Templo de Lourdes sin un panorama general, tanto del modo de recrear el ambiente como programar las funciones que estarán involucradas en esta misión. Con la experiencia anterior se puede preparar la misión con un menor margen de error y con una mayor visión de lo que se necesita para el recorrido.

4.4.1 MODELADO Y LEVANTAMIENTO TEMPLO DE LOURDES

Para el modelo y levantamiento del Templo de Lourdes se necesitó una visita directamente al lugar, así como investigación de planos y documentos pertenecientes al templo. A falta de archivos que pudieran facilitar la labor se tuvo que construir el interior de la Iglesia por medio de fotografías y mediciones presenciales que facilitaran la construcción del lugar.

A partir de las fotografías armar un modelo del lugar y a partir de las mediciones crear una maqueta del lugar para que se pudiera representar la estructura y modelar el inmueble lo mas fielmente posible al real.

Toma de fotografías del Templo

Ejemplo de las fotografías que sirvieron para la reconstrucción del Templo de Lourdes.

Figura 4.18
Fotografías para la reconstrucción del Templo

A diferencia del modelado de las fachadas que se colocaron en el recorrido exterior, el Templo de Lourdes guarda un mayor número de detalles en su interior. Por ejemplo, las paredes cuentan con columnas y volados sobre las paredes, se puede agregar mas de una colisión para un objeto, pero eso implicaría un aumento de polígonos en la estructura del templo, lo que se necesita es hacer la estructura de la iglesia lo mas cercana a la real sin un gran número de polígonos. No sería problema si aseguráramos que el recorrido será ejecutado en equipos con aceleradores gráficos, pero se pretende que el recorrido por el interior del Templo de Lourdes sea capaz de correr en la mayoría de los equipos existentes

en el mercado actual. Para ello utilizaremos los objetos tipo DIF, los cuales están enfocados en estructuras conocidas como 'interiores', estos objetos permiten optimizar las colisiones así como el manejo de estructuras con niveles de detalles mas precisos a diferencia de los DTS.

El modelado de estos elementos es básicamente igual que para cualquier objeto que deseemos importar en TGE, a diferencia que se necesitarán herramientas adicionales para una correcta exportación de nuestro programa de modelado 3D.

Existe un programa de Garagegames llamado 'Constructor'[100] que está dedicado exclusivamente al modelado de interiores y es completamente gratuito. El problema de éste programa es que hay que dedicarle un buen tiempo para su utilización y manejo ya que es un poco complicado crear los interiores, pero lo interesante es que al exportar nuestros interiores con este programa estamos seguros que están optimizados de una forma adecuada para TGE. Por esta razón, lo mas razonable es modelar el interior del Templo de Lourdes con el programa que se ha utilizado para la creación de todos los modelos 3D y exportar el interior no directamente a TGE si no al programa 'Constructor' y de ahí exportarlo a TGE.

Para la construcción del Templo de Lourdes, no se hizo todo el modelo en una sola pieza, primero se realizó un estudio de aquellos elementos que se repitieran a lo largo de la Iglesia, y después de aquellos que mantuvieran su singularidad con respecto a los demás elementos decorativos. Lo primero que se hizo fue modelar el esqueleto del templo, nave principal y capilla lateral.

A partir de los elementos terminados y unidos, se realizó una exportación previa del modelo para comprobar si no existían errores de modelado o fallas al momento de texturizar el modelo.

---

[100] Constructor http://www.garagegames.com/products/constructor (ví: 11 de septiembre de 2011)

EN MODELOS DEMASIADOS GRANDES, LO MAS CONVENIENTE ES IR ARMANDO EL MODELO POR PARTES E IR PROBANDO LAS EXPORTACIONES PARA IDENTIFICAR POSIBLES FALLAS Y PODER ARREGLARLAS INMEDIATAMENTE. SUELE PASAR QUE AL FINALIZAR EL MODELO COMPLETO, HAYAN FALLAS EN LA ESTRUCTURA O TEXTURAS Y SEA MUY DIFICIL CORREGIRLAS YA TERMINADO EL MODELO.

Al asegurar que no existen fallas de exportación en la estructura preliminar del Templo de Lourdes, se empezó con la siguiente tarea, modelar y agregar los elementos decorativos que fueran repetitivos en toda la estructura. En este caso, los altares laterales son idénticos, por lo que solo fue necesario modelar un solo altar, crear copias del mismo y montarlos a lo largo de la nave principal del templo.

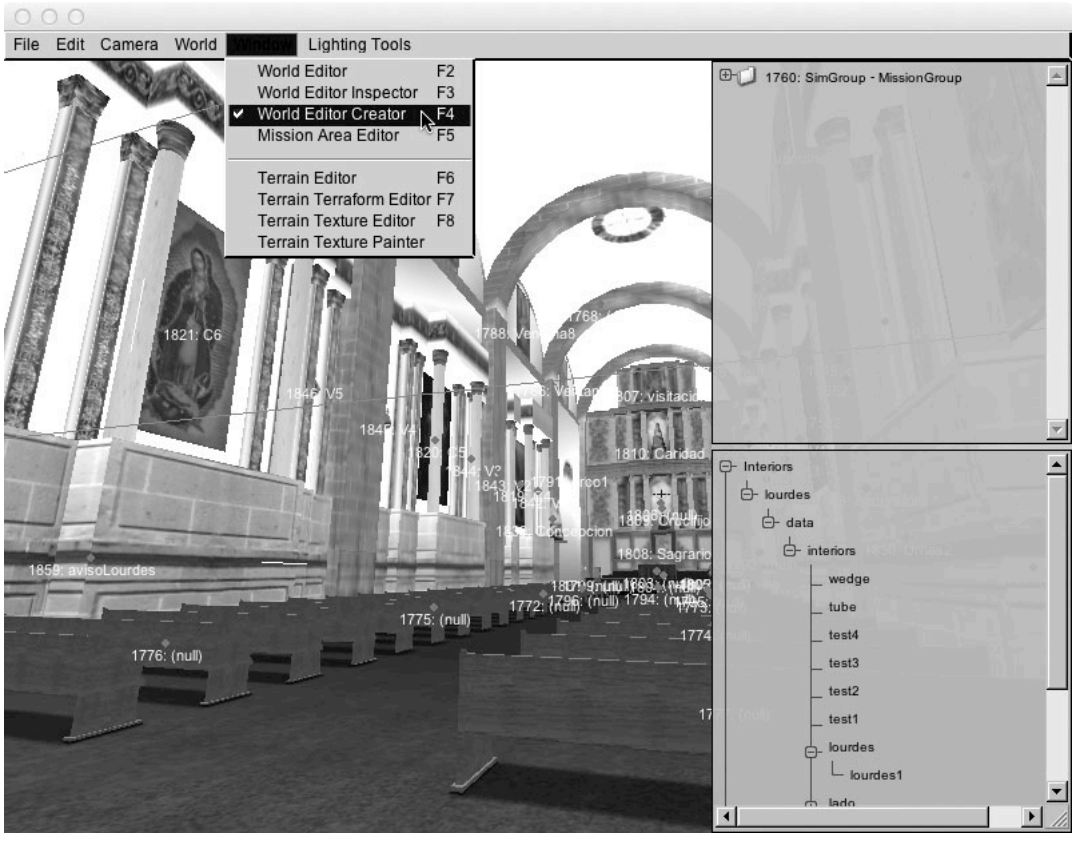
Asegurando que no hay problemas con la estructura preliminar se terminó el modelado del Templo de Lourdes con el altar principal y detalles de la capilla lateral. Para este momento la reconstrucción de la Iglesia esta en su fase final, solo falta su exportación en formato DIF y agregarla a la nueva misión que está por iniciarse.

En el sitio de Garagegames existen herramientas adicionales que se pueden instalar en los mas comunes programas de modelado 3D, entre estas herramientas existen programas que pueden anexarse al programa de modelado que estemos utilizando para exportar nuestros modelos. Entre ellos contamos con uno que sirve para exportar nuestro modelo, no como DIF si no como un archivo tipo ‘Constructor’ tipo .csx,y de ahí exportar nuestro modelo como un ‘interior’ y así agregarlo a la misión correspondiente.

Exportar el templo de Lourdes como un DIF, aseguramos que las colisiones estarán optimizadas y evitaremos una carga excesiva al procesador. Solo faltaría agregar los elementos decorativos como bancas y escalinatas que encontramos en los altares, tanto del principal como el de la capilla lateral.

Concluida la exportación solo falta agregar el modelo creado a la misión correspondiente. A diferencia de los modelos tipo dts, los interiores no son necesarios cargarlos desde un script, TGE es capaz de identificarlos justo al momento de inicializarse y mediante la herramienta de 'World Editor' en la opción 'World Editor Creator', en la ventana inferior derecha exploramos la carpeta, de que hemos seleccionado para guardar nuestros interiores. La arquitectura básica de TGE nos propone un folder llamado 'interiors' justamente en la carpeta data para colocar nuestros modelos. Solo basta seleccionar el 'interior' y se colocará inmediatamente en nuestra misión actual.

Interior del Templo de Lourdes



Desde el World Editor Creator también se colocan nuestros interiores.

Figura 4.19
Interior del Templo de Lourdes

Solo falta colocar los objetos decorativos que no entrarán en la descripción prevista para el Templo de Lourdes. Revisando el Documento de Diseño, se pide que al entrar al templo se habiliten funciones para el usuario como colocar objeto, ver detalles del objeto, entre otras. Pero no necesariamente todos los objetos deben de entrar en esta categoría de ser detallados cuando el usuario de click sobre ellos.

Por ejemplo, bancas, puertas y algunos objetos extras no necesitan ser detallados en el recorrido, éstos modelos serán creados de la misma manera en como se hicieron los modelos para la misión exterior. Hay que resaltar esta parte, ya que los elementos que tendrán ciertas funciones sobre ellos, no serán exportados de la misma manera.

Al finalizar la tarea habrá acabado la primera parte de la misión interior. Como se había comentado, si se está iniciando en desarrollos de éste tipo, lo mas adecuado es empezar con el ambiente y finalizado o al menos teniendo ya una gran parte de el se puede empezar con la programación de funciones y scripts asociados a la misión.

4.4.2 SCRIPTS DE FUNCIONES ASOCIADAS AL TEMPLO DE LOURDES

Las funciones que se deben contemplar para el interior del Templo de Lourdes son:

- Ver detalles de objetos.
- Colocar objetos.
- Mover objetos colocados por el usuario.

Aunque no hay jerarquía para empezar la programación de las tres funciones adicionales, es mejor hacer un estudio previo para ordenar las fases en que serán programadas.

Si se desea empezar con la parte fácil sería por la función de 'detalles de objetos', porque si se empieza por la de 'mover objetos', quedaría un vacío preliminar, ya que se pide que los objetos que se muevan sean únicamente colocados por el usuario, lo conveniente sería en ese caso empezar por la de 'colocar objetos'.

¿Qué necesidad se tiene el colocar objetos dentro del Templo de Lourdes?. El perfil prioritario de la aplicación está enfocado en clientes que deseen conocer el inmueble, esencialmente aquellos deseen conocer un poco de los tesoros que se encuentran en la Iglesia, así como aquellos que pretendan celebrar algún evento religioso en el mismo.

Pensando en aquellos que deseen contratar una celebración en la misma se ha decidido en brindar la oportunidad que el mismo usuario pueda adornar el templo a su gusto.

EL PROPÓSITO MAS IMPORTANTE ES PERMITIR QUE LA APLICACIÓN SEA COLABORATIVA, ES DECIR, QUE UN SOLO PRODUCTO ABRA SU ABANICO DE POSIBILIDADES PERMITIENDO INCORPORAR AL PROGRAMA MAS DE UN SERVICIO. DE ESTA MANERA SE ACCEDE A UN MAYOR NÚMERO DE FUNCIONES Y SOBRE TODO, QUE ES LO QUE SE AMBICIONA, HACER EL RECORRIDO POR EL TEMPLO DE LOURDES, LO MAS INTERACTIVO POSIBLE. NO DEJANDO EL RECORRIDO, EN UN PASEO PROGRAMADO Y SIN UNA INTERACCIÓN MAYOR QUE LA DE CAMINAR O CONOCER LOS ALREDEDORES.

Por ello, además del recorrido por el Templo de Lourdes se pensó en agregar servicios adicionales que tuvieran que ver de alguna manera con la Iglesia. Considerando que el poder adornar la Iglesia al gusto del cliente sería una buena opción, se concluyó que sería preciso contactar con florerías que ofrecieran servicios para ceremonias y arreglos florales en la ciudad de México.

El estudio preliminar para esta función es hacer una lista de los arreglos florales que se agregarán durante el recorrido por el Templo de Lourdes, la florería seleccionada fue 'RC Ambientación Floral'.[101] Los arreglos florales seleccionados se modelaron para convertirlos en shapes tipo dts y así agregarlos al recorrido.

Es momento de hablar de una de las características ya antes vistas, los 'datablocks'[102]. Recordando un poco, los datablocks son duplicados del servidor al cliente y cuentan con un ID que es el mismo tanto del lado del servidor como del cliente, además cuentan con características que pueden ser manipuladas por medio de scripts, sin olvidar que los datablocks son controlados por el servidor, de esta forma cuidamos que las propiedades estáticas de los objetos sean manipuladas por el cliente

Esta característica es muy importante, ya que ante todo, debemos de cuidar el rendimiento del equipo en el que se está ejecutando la aplicación. Al ser duplicados del servidor estamos asegurando que aunque son objetos dentro del recorrido no afectarán en demasía el rendimiento del procesador gráfico.

Entre mas objetos haya en la misión o el modelado de los mismos no haya sido depurado para una óptima exportación (número de polígonos, texturas) el número de procesos por el acelerador gráfico aumentará considerablemente y si la RAM de la computadora como la tarjeta de video no son capaces de soportar la carga de gráficos, el equipo empezará a sufrir retrasos en sus procesos, ejemplos

[101] Rc Ambientación Floral, http://rcambientacionfloral.com/ (ví: 12 de septiembre de 2011)
[102] Capítulo 3, tema 3.2.

comunes: los movimientos dentro de la aplicación se tornan lentos, hay demora considerable en la carga de las misiones, incluso se ralentiza la aplicación y en el peor de los casos, la aplicación sufre un error grave que haga que se cierre.

Considerando lo anterior y que no podemos asegurar que el equipo sea capaz de soportar la carga de objetos en la misión, se utilizará este tipo de objetos para incorporarlos en la misión.

El tipo de archivos de exportación es el mismo que el de los shapes normales, incluso su extensión también es tipo dts, pero la carga es diferente, los shapes normales que se han agregado en las misiones han sido por carga directa y colocados por medio del 'World Editor', en cambio los shapes que seleccionaremos para agregar en el interior del Templo de Lourdes serán agregados por medio de script.

Los datablock's cuentan con un ID que los identifica al momento de ser agregados en la misión al igual que su posición XYZ dentro del ambiente. Al momento de ser agregados en la misión este identificador único (ID) es irrepetible durante todo el recorrido, el detalle radica en que no se tiene control de qué número de identificador se asigna a cada datablock y para acceder a las características de los datablocks necesitamos su ID.

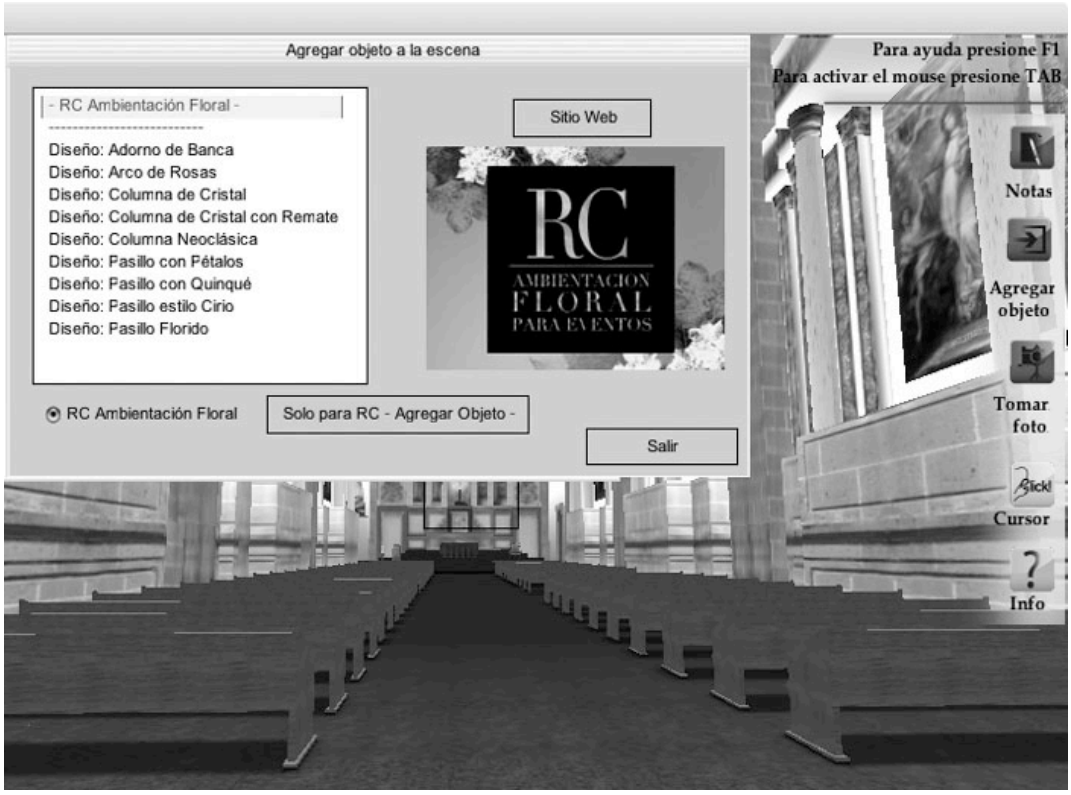
Para solventar este problema contamos con el 'CrossHairHud', cuadro central que permite seleccionar el objeto al cual se esté apuntando. Gracias a el tenemos la oportunidad, por medio de scripts, obtener el ID de sistema o el nombre del objeto al cual se esta observando.

Resolviendo el problema de obtener el ID del objeto o el nombre del shape o datablock, se resuelve un gran problema para las funciones que necesitamos habilitar para el usuario.

UNO DE LOS ASPECTOS BÁSICOS EN LOS RECORRIDOS VIRTUALES ES PODER OBTENER INFORMACIÓN DE LOS ELEMENTOS QUE SE ENCUENTRAN DENTRO DEL AMBIENTE. SI NO SE ES CAPAZ DE SOLVENTAR, AL MENOS ESTE DETALLE, SEGURAMENTE NUESTRO RECORRIDO NO SERÁ INTERACTIVO PARA EL USUARIO Y PASARÁ A SER OTRO RECORRIDO AMBIENTAL.

Para la interfaz de usuario que permitirá agregar objetos al recorrido deberá contar con los elementos visuales ya mencionados en el Documento de Diseño. Para ello se necesitó la lista completa de los elementos a colocar y las imágenes de los diseños propuestos por la empresa foral.

Interfaz gráfica para el menú, Agregar Objeto



Del lado izquierdo esta la lista de los elementos disponibles para agregar durante el recorrido y del lado derecho una previsualización del objeto.

Figura 4.20
Interfaz para la función Agregar Objeto

Concluida la interfaz gráfica para el menú ya se puede iniciar la parte de programación de los scripts. Se necesita contemplar las siguientes tareas:

- Script para agregar objetos en la escena. Botón Agregar Objeto.
- Script que seleccione de la lista un elemento de la misma y pase el dato al script de agregar objeto. Control izquierdo de la ventana.
- Script que permita. a partir del objeto seleccionado de la lista, abrir el sitio de internet correspondiente al mismo.
- Script que permita la visualización del objeto seleccionado en ventana. Módulo derecho de la ventana.

Hay diferencias importantes al cargar éste tipo de objetos, aunque los shapes normales también se consideran datablocks, la diferencia radica en que los objetos estáticos que se encuentran durante todo el recorrido se consideran como: 'StaticShapeData' mientras que los objetos que se puedan agregar a la misión se consideran como: 'ItemData'.

¿Cuál es la diferencia entre los StaticShapeData y los ItemData?. Aunque hay características importantes que los distinguen, por ejemplo, colisiones, interacción con el usuario, etc, para el caso del recorrido por el Templo de Lourdes se hace la siguiente distinción, que aquellos elementos de uso decorativo únicamente se considerarán como StaticShapeData mientras aquellos elementos con los cuales se quiera que haya una interacción con el usuario se catalogarán como ItemData.

Concluido el modelado, exportación y scripts necesarios para que la aplicación reconozca los elementos en lista, es momento de añadir el script de agregar objeto.

Los objetos que se agreguen a la misión se considerarán como Items en lugar de shapes. Y el script al momento de agregar el objeto deberá asegurar al

menos dos detalles. Ser capaz de reconocer el ID con el que cuenta el objeto al ser seleccionado y reconocer el nombre asociado al Item.

La pregunta que salta inmediatamente es ¿Se permiten nombres duplicados durante la escena?. Estrictamente hablando no. Cuando se esta creando el ambiente y se están colocando árboles a lo largo del recorrido y se desea nombrarlos por medio del Handle (Identificador y Etiqueta), no se pueden nombrar los arboles con un mismo nombre. Sin embargo, al momento de agregar un objeto en la escena durante el recorrido, el Item al que se esta llamando, tiene por omisión un nombre, si no fuera así, sería imposible reconocer el objeto, la diferencia radica en el identificador ya que no será el mismo al objeto anterior, aún siendo un duplicado del anterior.

Por ejemplo, un elemento de la lista es el objeto decorativo 'Arco de Rosas', ¿cómo reconoce la aplicación que se necesita agregar un arco de rosas?. Los únicos datos que se tienen son: El nombre del archivo .dts y el nombre con el que se diferencia al Item de los demás Item's. No hay mas datos con los que podamos contar del objeto. Pero ¿por qué los objetos que ya están en la misión cuentan con un ID incluso con un nombre? El ID es imposible obtenerlo antes de colocar el objeto en la misión, porque mientras el Shape o Item no sea colocado en escena, dicho objeto no tendrá un ID que lo reconozca ya que el ID es otorgado por el servidor justo al momento de ser colocado en la escena. A comparación de los objetos que ya se encuentran en la escena, estos tampoco contaban con un ID hasta el momento que fueron colocados en la misión durante la carga de la misma al iniciar el recorrido.

Al tener únicamente estos dos campos para interactuar con el objeto hay que sacar provecho de ellos. Al llamar un objeto de la lista para ser colocado, por ejemplo, arco de rosas, el script debe ser capaz de reconocer el nombre del objeto en la lista y después buscar éste elemento (el nombre del objeto) en la lista de objetos almacenados para que posteriormente sea agregado a al escena. Y si el

script permite quedarse con este nombre para que al momento de colocar el objeto se pueda vincular el ID con el nombre que se tiene en ese momento, no estamos duplicando objetos en la escena si no que estamos relacionando el nombre de un objeto con varios ID dentro de la escena.

Al resolver este problema ya contamos con dos elementos imprescindibles, el ID y el nombre del objeto para las demás funciones.

Para la función de ir al sitio web de la empresa o el arreglo floral es menos complicado, ya que a cada elemento se le asocia su respectivo link y lo único que se hace es llamar la respectiva aplicación programada por omisión en el sistema para abrir el navegador e insertar el sitio web para visitar la página.

De la misma manera funciona el módulo de visualizar el objeto seleccionado, aunque en este caso, para evitar problemas con conexión a internet se decidió colocar las imágenes dentro de la aplicación.

Solo faltaría agregar el script al botón ‘agregar objeto’ al menú de la interfaz del recorrido.

La siguiente tarea a realizar es la de ‘mover objeto’, dicha tarea está vinculada a la de agregar objeto. Para esta función es necesario vincular varias funciones al programa principal que hace referencia al script mover objeto.

La función tiene que acceder a los datos que ya tiene la función de agregar objeto. Obviamente no se puede mover un objeto, sin haberlo agregado antes, de antemano los datos de dicho objeto, pre-existen en la aplicación. Para mover un objeto se necesita obtener o su nombre o su ID, esta referencia nos la da el ‘CrossHairHud’.

Hasta el momento, el 'CrossHairHud' es un elemento que ha sido mencionado varias veces durante la programación del recorrido, pero no se ha contemplado su importancia. Ya sabemos que a partir de el, podemos obtener datos importantes del objeto que estamos observando. La ventaja de éste módulo es el seleccionar un espacio de la interfaz de usuario dedicada exclusivamente para obtener datos del entorno, como su ID, el nombre, el tipo de objeto, entre otras características.

Sabiendo las bondades de este módulo, simplemente basta con vincular la funcionalidad del elemento del 'GUI Editor' con el script que accede o contiene los datos del elemento observado. A partir de ahí solo basta obtener los datos del objeto que se está observando para que sean contemplen como argumentos del script asociado a la función 'mover objeto'.

La función mover objeto trabaja de la siguiente forma, a partir del 'CrossHairHud' se "leen" los datos del objeto que se esta observando, estos datos al dar click al botón secundario pasan a formar parte del argumento de la función 'mover objeto' y son procesados de la siguiente manera:

- Se obtiene el ID del objeto observado…

Es pertinente mencionar la siguiente observación. ¿Por qué se utiliza el ID y no el nombre del objeto observado?. Es muy probable que exista mas de un objeto agregado con el mismo nombre dentro del recorrido, por ejemplo, se desea agregar una hilera de adornos de banca a lo largo del pasillo principal de la Iglesia. Claramente cada uno de estos objetos tendrá el mismo nombre, si la función se valiera del nombre para mover el objeto ocurrirían dos probables errores: se moverían todos los objetos con el mismo nombre o la aplicación generaría un 'warning' en el mejor de los casos, al decidir no mover ningún objeto por no saber cual de todos mover o en el peor de los casos generar un error general y cerrar la aplicación al no procesar correctamente los argumentos de la función. Pero ya

sabemos que los ID generados por el servidor son únicos e irrepetibles, de manera que si seleccionamos trabajar con el ID del objeto en lugar del nombre del objeto, estamos asegurando que el script no genera algún error con el argumento elegido.

Retomando cómo la función 'mover objeto' procesa los argumentos se continua de la siguiente manera:

- Se obtiene el ID del objeto observado.
- El ID se convierte en argumento de la función 'mover objeto'
- La función 'mover objeto' obtiene la posición del objeto
- Se programa un script para un botón en la interfaz de usuario 'mover objeto', o se asigna la función a una tecla para desplazar el objeto de acuerdo a la magnitud seleccionada.
- El objeto se traslada el número de veces con respecto a la magnitud seleccionada.

No solo basta con desplazar el objeto de forma horizontal, probablemente se desee también mover el objeto de manera vertical. Para ello, habrá que agregar la magnitud seleccionada con respecto al eje Z.

Como se esta trabajando en un ambiente tridimensional, el movimiento básico sería sobre el eje horizontal, adelante, atrás, izquierda, derecha; pero existe una magnitud que podría pasar desapercibida que sería el eje vertical, moverlo hacia arriba o hacia abajo.

Hay dos maneras posibles para trabajar éste argumento: agregar una variable nueva para que se procese sobre el eje Z o utilizar el mismo argumento con el que se esta trabajando sobre el eje horizontal para desplazar el objeto verticalmente.

Mover Objeto

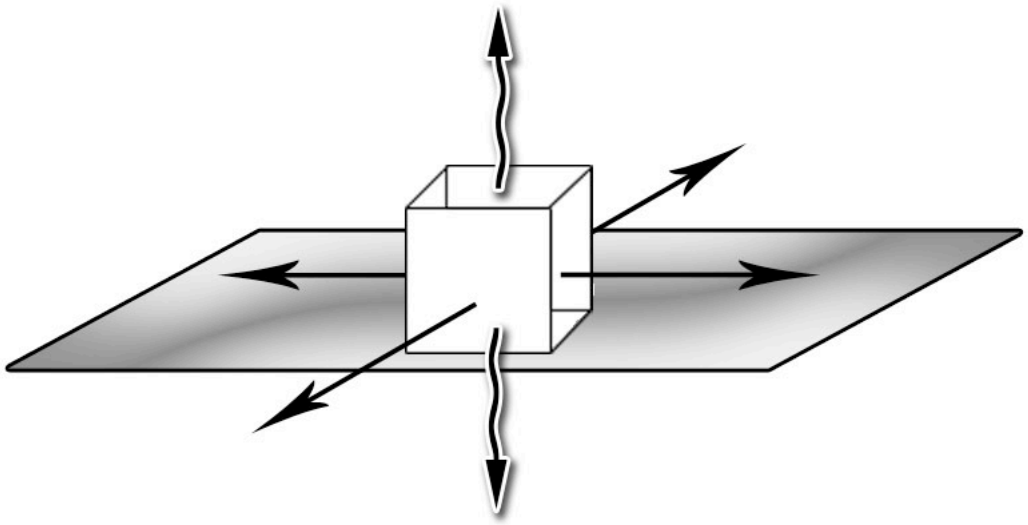


Figura 4.21
Ejes para desplazar un objeto

De la figura anterior se observa los ejes que debemos contemplar al momento de mover un objeto. Se pueden agregar restricciones para únicamente mover el objeto sobre el eje horizontal pero lo conveniente es poder agregar el dato para permitir mover el objeto también sobre el eje Z. La solución que se escogió es agregar también el argumento para desplazar el objeto sobre el eje horizontal sobre el eje vertical.

Al tener la función trabajando correctamente sobre el objeto basta concluir la interfaz de usuario que se utilizará sobre este control. La referencia concreta se obtiene del Documento de Diseño, a partir de el se programó la siguiente interfaz:

Interfaz Mover Objeto

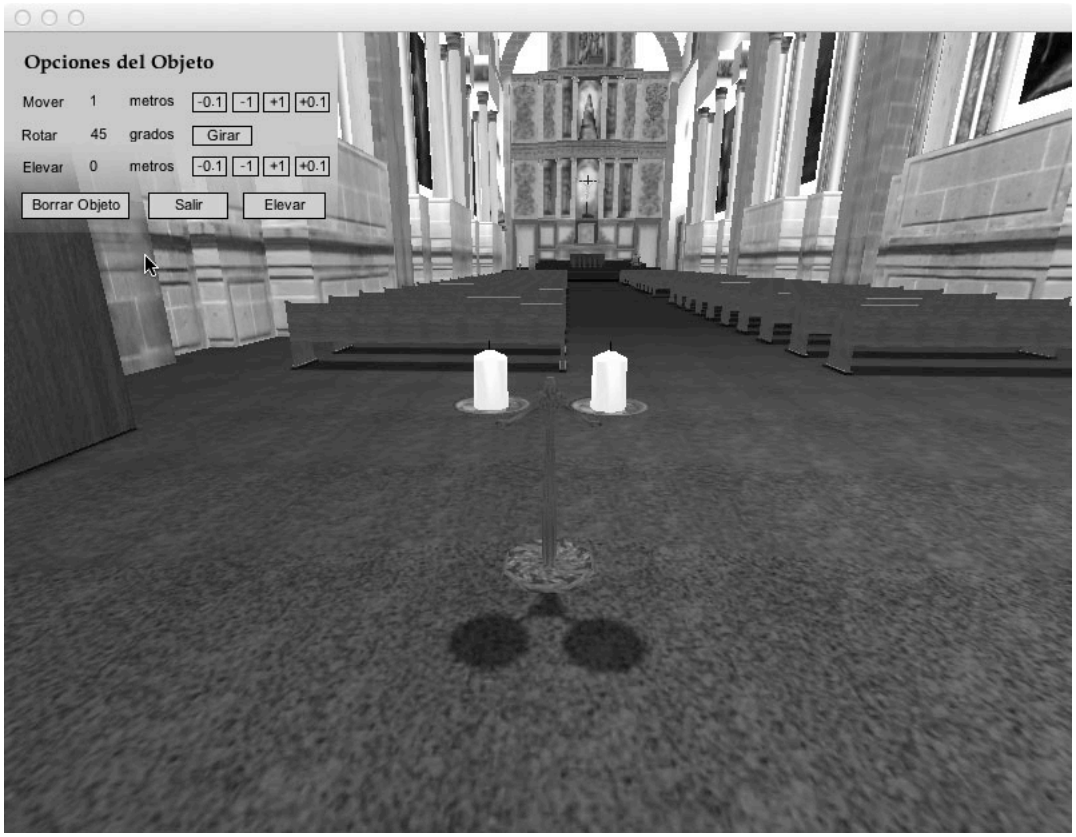


Figura 4.22
Interfaz 'Mover Objeto'

Aunque en el Documento de Diseño no se habla de la posibilidad de eliminar un objeto agregado, es una acción inherente a la función de 'Mover Objeto'. Esta operación se agrego directamente en el menú de mover objeto.

Para las acciones de mover, elevar y rotar objeto, se agregaron botones especificando la magnitud que uno desea mover, las magnitudes aparecen justo delante de cada movimiento a realizar sobre el objeto en uso.

Con la finalización de la tarea anterior, bastará únicamente la función pendiente que es la de 'Previsualizar Objeto'.

La función de ‘Previsualizar Objeto’ depende de varios factores:

- Todos los objetos agregados por el usuario tendrán esta característica.
- No todos los objetos en escena podrán previsualizarse.
- Además se deberá contar con un texto detallando el objeto o brindando un contexto del mismo.

Para la opción de previsualizar los objetos, se seleccionó una carpeta exclusiva para ellos. Lo anterior es para tener un mayor control sobre los objeto, ya que, aunque se modelan de la misma forma que todos los anteriores, se consideró que agregarlo dentro de la carpeta ‘shapes’ sería mas difícil su ubicación y manejo de los mismos. Dicha carpeta se denominó ‘lourdes’ ubicada en el folder ‘data’.

Ya teniendo la organización adecuada para estos elementos basta con programar la interfaz de usuario y sus respectivos scripts.

Siguiendo las instrucciones para esta función que brinda el documento de diseño se observa que la interfaz también cuenta con dos módulos. Uno de ellos desplegando texto y el otro mostrando la previsualización del objeto, con la diferencia que éste módulo, a diferencia del anterior (agregar objeto) que solo se mostraba una imagen del elemento, aquí se pide poder visualizar el objeto en tres dimensiones, acercarlo y alejarlo, incluso rotarlo.

Retomando la decisión de utilizar formatos para ciertos elementos, como las imágenes o el texto, se continuará con la misma filosofía. Como ya habíamos comentado, TGE es capaz de procesar archivos creados con extensiones personalizadas, pero no se puede estar seguro si siempre, aun con el cambio de sistema operativo pueda seguir dándose la lectura de forma correcta. En cambio si utilizamos formatos universales, se asegura que la lectura de los archivos sea exitosa. De manera que las descripciones que se mostrarán en pantalla también

contarán con un formato accesible para las dos plataformas. Para este primer módulo, se tendrá un archivo asociado al objeto, cuando se seleccione un objeto, se buscará el archivo asociado al objeto para su lectura y se desplegará en pantalla.

Para la función de previsualizar el objeto en tres dimensiones, el problema se agrava en demasía, ya que el motor no cuenta con esas características particulares. Para resolver este conflicto se tiene que agregar estas funciones al motor directamente. Es decir, modificar el código fuente del motor y compilar de nuevo para crear los ejecutables.

ANTES DE INICIAR CUALQUIER PROYECTO, SE DEBE HACER UN ESTUDIO PRELIMINAR DE LAS HERRAMIENTAS DISPONIBLES Y DE LOS ALCANCES DE LAS MISMAS, SABER SI OFRECEN SOPORTE TÉCNICO O PODEMOS MANIPULAR LA ESTRUCTURA PARA ADAPTARLAS A NUESTRAS NECESIDADES. SI NO SE HACE, CABE LA POSIBILIDAD QUE NO SE PUEDA CON ALGUNA TAREA Y TODO EL DESARROLLO SE VENGA ABAJO.

Esta tarea pudo haber sido un problema gravísimo para el desarrollo del producto. Lo que ayudó considerablemente al proyecto para concluir esta tarea con éxito fue que al comprar el motor TGE, se otorga la licencia con todo y el código fuente, una característica excepcional. Y obviamente, sin el soporte técnico al cual se tuvo que recurrir para pedir orientaciones adicionales a las pertenecientes al desarrollo del recorrido, las cuales fueron de gran ayuda, no se hubiera podido lograr la programación correcta para esta función.

Lo primero que se hizo fue agregar instrucciones adicionales al motor para que estuvieran disponibles al momento de crear los scripts para esta función. Cuando se programaron correctamente el código necesario se compiló el motor nuevamente, tanto para Macintosh como para Windows. Es necesario aclara una cosa. Los scripts que se programan, la misma versión sirve tanto para Macintosh como para Windows, lo que se necesita es compilar el ejecutable para cada

sistema operativo, por tal razón, cuando se realizan modificaciones al código fuente, se debe compilar de nuevo el motor para crear los ejecutables respectivos para cada plataforma.

Ya que se tienen las modificaciones en el motor y ha sido compilado correctamente, se puede continuar con los scripts necesarios para la función 'Previsualizar Objeto'. Para éste módulo, como la mayor parte del código se encuentra directamente compilado en el ejecutable, solo es necesario utilizar los comandos programados para tal efecto. La facilidad de programar directamente en el motor las funciones a implementar, es que ya no se necesita hacer un script tan robusto.

Solo falta vincular la interfaz creada con sus respectivos botones quedando de la siguiente manera.

Interfaz Ver Objeto

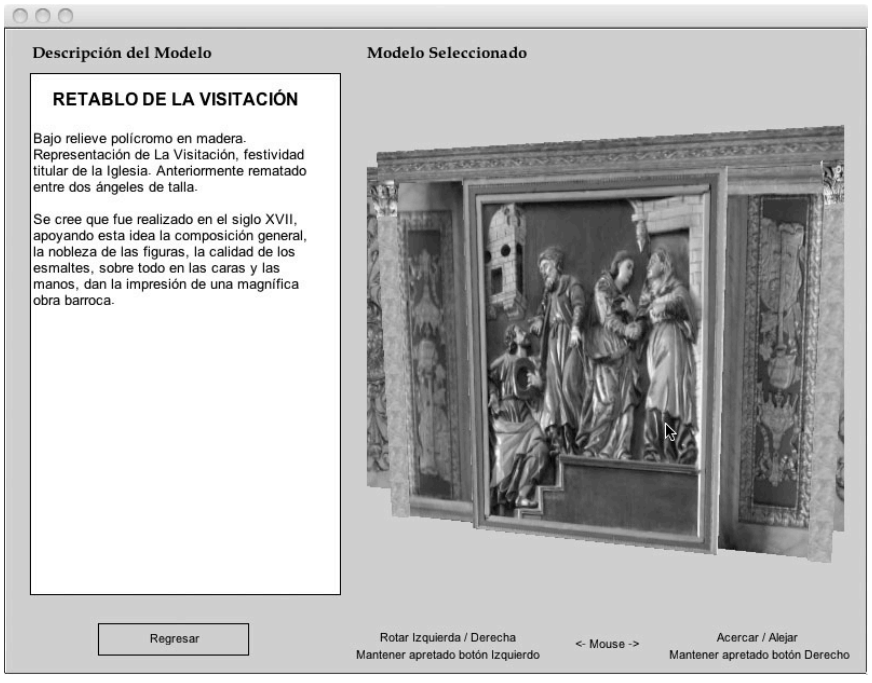


Figura 4.23
Visualizar Objeto

Con esto damos por terminada la programación de las funciones disponibles al interior del Templo de Lourdes.

Solo hace falta agregar los archivos de ayuda, los cuales estarán disponibles a lo largo del recorrido. Se utilizará un control similar al que se utilizó para la parte de mostrar texto en pantalla de la función 'Previsualizar Objeto' y se programó el script para el botón F1, se escogió esta tecla ya que la mayoría de los programas, tienen asociado este botón para la ayuda.

Con esto se concluyen las tareas asociadas para la creación de los scripts para las funciones que estarán disponibles al usuario durante todo el recorrido. Básicamente la aplicación esta terminada, solo falta el proceso de pruebas para encontrar errores y sean corregidos.

4.4.3 AUDIO

Para la parte de audio, que se considera la parte final de la programación del recorrido por el Templo de Lourdes se implementó tanto para el menú principal como para cada una de las misiones presentes en el recorrido.

Los sonidos a emplear fue tanto para los botones como sonidos ambientes que amenizaran el recorrido durante las misiones. No hay mucho que decir sobre el aspecto de audio. Para el sonido ambiental del recorrido exterior, se utilizó equipo de grabación y se obtuvo directamente de los alrededores de la Iglesia, mientras que para la misión que corresponde al Templo de Lourdes así como la música del menú principal se utilizaron melodías novohispanas y con toques barrocos respectivamente. Hay una función que escanea cuando una misión o un elemento esta abierto, basta con activar la melodía o sonido cuando este activo el elemento y parar la melodía cuando éste se desactive.

4.5 VERIFICACIÓN Y PRUEBAS DEL PRODUCTO

Al revisar la aplicación en ambos sistemas, se convino a la tarea de entregar el recorrido para sus pruebas a varios tipos de usuarios con los que se creería que habría un mayor uso.

En esta etapa se entrego una versión beta a 10 usuarios para que la probarán y dieran sus opiniones. La muestra consistió en 5 hombres y 5 mujeres, de las cuales 2 personas tanto para hombres como mujeres fueron de mediana edad y 3 fueron jóvenes. Las edades para la gente de edad madura fue de 40 a 50 años mientras que para los jóvenes fue de 20 a 35 años.

Lo que se pidió fue claro, que opinarán sobre la usabilidad, estabilidad, instalación, funciones y apariencia. Aunado a los detalles anteriores también que brindaran una opinión general sobre el producto y qué mejoras podría haber o incluso que ofrecieran ideas para la aplicación. Para evaluar los características se pidió que calificarán de 0 a 10 cada propiedad.

La información adicional que se proporcionó a los usuarios al entregar el programa fue que para poder usar la aplicación, únicamente tenían que copiar y pegar la carpeta en cualquier lugar de su computadora.

Los datos arrojados por las pruebas se dividieron por edad de los usuarios. Un bloque sería el de la gente de edad madura y el otro bloque serían los jóvenes. Los resultados que se obtuvieron son los siguientes:

Para los usuarios de 40 a 50 años se arrojaron los siguientes resultados.

Usuarios de 40 a 50 años

	Usabilidad	Estabilidad	Instalación	Funciones	Apariencia
Hombres	**5**	9	10	9	9
Mujeres	**3**	9	10	8	9

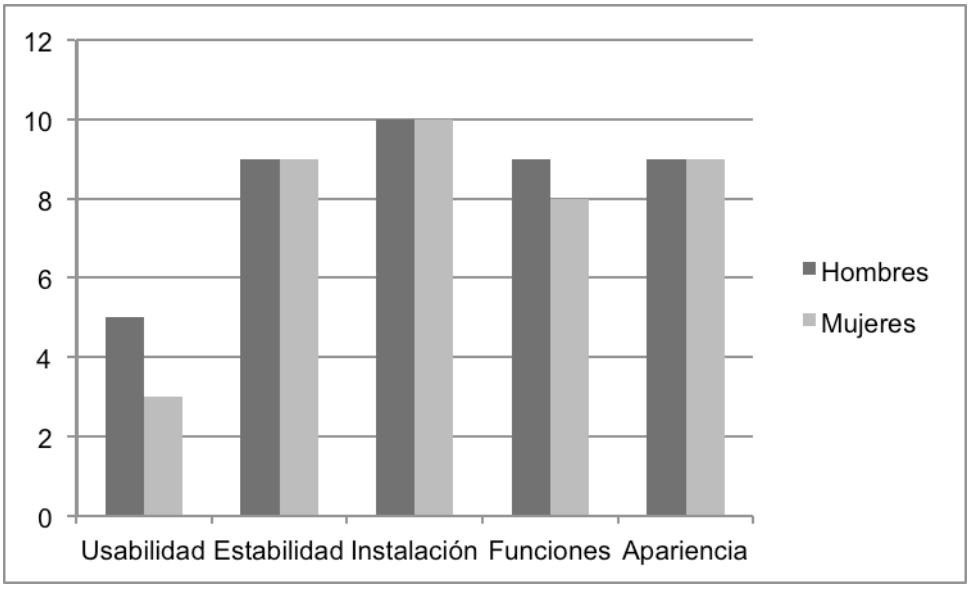


Figura 4.24
Resultados arrojados por los usuarios de 40 a 50 años.

Las opiniones generales que se obtuvieron de esta encuesta fueron las siguientes:

- Complicado uso de los controles.
- Error de salto en el video de inicio.
- Confusión al ingresar a la carpeta donde se encuentran los archivos guardados.
- Modificar el modo de pantalla durante el recorrido.
- Errores al mover los objetos a través de ellos.

- Falta de información durante el recorrido para saber que funciones se pueden usar.
- Letras mas grandes.

Esta información fue proporcionada únicamente por las mujeres, ya que no hubo mucha retroalimentación de parte de los hombres.

Para los usuarios de 20 a 35 años se arrojaron los siguientes resultados

Usuarios de 20 a 35 años

	Usabilidad	Estabilidad	Instalación	Funciones	Apariencia
Hombres	9	10	10	7	9
Mujeres	10	9	10	8	8

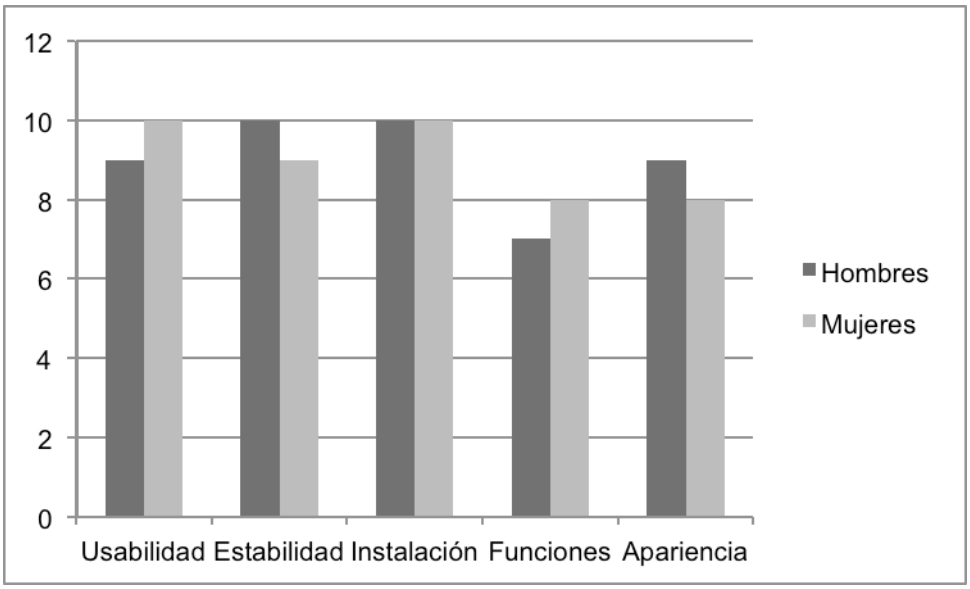


Figura 4.25
Resultados arrojados por los usuarios de 20 a 35 años.

Las opiniones surgidas por éste intervalo de personas fueron las siguientes:

- Error de salto en el video inicial.
- Errores durante la transición del recorrido al dar click en los links que muestran los archivos almacenados.
- Errores al mover los objetos.
- Avisos de qué funciones están disponibles durante la misión en curso.
- Error durante el uso del cursor, en ocasiones no obedece la tecla tabulador.
- En las misiones exteriores no es tan fluida la aplicación.
- Los santos se ven de forma rara.

Se proporcionaron las siguiente ideas al producto:

- Hacer que al seleccionar un producto se cree automáticamente toda una fila a lo largo del pasillo.
- Agregar botón de saltar video.
- Poder encimar o elevar objetos uno sobre otro.
- Mayor facilidad al mover objetos, combinación de teclas o que sean mas representativas.
- Si es posible, simplificar los pasos anteriores.
- Poder mover un objeto justo frente a uno.
- Tener un conteo de los objetos que se han colocado en la iglesia y si es posible, agregar esta información al archivo de notas.
- Al ingresar al Templo de Lourdes, salir nuevamente a las calles aledañas de la Iglesia.
- Que tenga acceso a redes sociales.
- Así como existe un menú de colocar objetos, también exista un menú para seleccionar diferentes piezas musicales para ceremonia.
- Escoger tipo de recorrido: día o noche.
- Agregar un botón que lleve a la página de la Arquidiócesis de México.

Así como sucedió en la primera encuesta, la mayor parte de las opiniones fueron por parte de las mujeres, las observaciones por parte de los hombres fue nula.

Haciendo un estudio de las encuestas realizadas, a los hombres solo les importó conocer los alrededores y el interior del Templo de Lourdes y no les interesó mucho la opción de adornar la iglesia. La respuesta común fue que se cumplía con el objetivo de conocer los alrededores y la iglesia.

En cambio, las mujeres fueron las que aportaron un mayor número de observaciones al recorrido así como ideas para fortalecer la aplicación.

Valorando las respuestas recibidas y errores generados durante la prueba al recorrido se obtuvieron las siguientes conclusiones.

Para la gente de edad madura fue que no se pudo obtener datos que tuvieran un peso significativo, ya que al calificar el recorrido con puntuaciones demasiado bajas con respecto a la usabilidad, se entiende que probablemente no utilizaron la aplicación en su totalidad.

El problema que salta inmediatamente es tratar de resolver el problema de la usabilidad para cierto grupo de personas. Aunque existe un detalle que no se termina de entender del todo, si la baja calificación de usabilidad se generó a partir de los controles o a una mala organización con respecto a las interfaces de usuario asociadas a la aplicación. Ya que en apariencia dieron notas muy altas.

En cambio con la gente mas joven, se recibieron detalles mas específicos y brindaron mayores ideas que pueden ser añadidas a la aplicación.

De la evaluación de los errores que se encontraron fueron los siguientes:

1. Errores al cambiar de resolución la aplicación.

2. Error en el video inicial.
3. Errores durante el movimiento de objetos y colocación.
4. Falta de información.
5. Error de cursor.

Las ideas que se pueden agregar al recorrido que son viables son:

1. Hacer mas fluido el movimiento de los objetos o tener combinación de teclas.
2. Movimiento rápido de los objetos.
3. Que no se encuentre en cero el valor por omisión para la altura del objeto en la ventana de 'Mover Objeto'.
4. Conteo de objetos agregados en la Iglesia.
5. Si el usuario escogió iniciar con la misión exterior, al ingresar al Templo de Lourdes poder salir nuevamente a la misión en la que se encontraba.
6. Agregar botón para saltar video.
7. Agregar vínculo hacia la página de la Arquidiócesis de México.

Ahora solo es necesario realizar las pruebas pertinentes y solventar los errores encontrados, así como aplicar las recomendaciones viables para la mejora del producto.

4.6 IDENTIFICACIÓN Y CORRECCIÓN DE ERRORES

Ya se cuenta con una lista de errores y mejoras que se pueden aplicar al producto. Es momento de tratar estos defectos y funcionalidades de una manera comprensible.

El primer paso para gestionar los defectos es comprenderlos.[103] Para ello Humphrey nos propone unas ideas:

- Registrar los defectos.
- Registrar información y datos suficientes sobre cada defecto.
- Analizar los datos obtenidos de cada defecto.
- Idear formas para corregir los defectos.

Antes de continuar vale la pena hablar de los defectos de diseño. Estos defectos pueden ser ocasionados por varias razones. Uno de ellos puede deberse a un mal diseño. Estos errores aparecen frecuentemente cuando hay falta de experiencia en la elaboración, por ejemplo, del Documento de Diseño y entre las causas puede ser que no se comprendió una función, se diseño una función con datos que no eran los correctos o incluso ignorar alguna condición para el manejo de argumentos. También se debe a una falta de información, exceso de trabajo, falta de descanso entre otras particularidades referentes directamente al equipo humano.

---

[103] Humphrey, Watts S. Introducción al Proceso Software Pesonal, Addison Wesley, 2001. p. 147

Una de las causas mas comunes es que aunque se entiende el diseño del problema y se elabora un documento de diseño correcto, el problema radica en no haber entendido correctamente el contexto del problema.

Cuando los diseñadores no han conocido el contexto del problema es probable que se interpreten los datos de forma literal y mas aún cuando el usuario explica las condiciones del problema con sus propias palabras o la experiencia que el cree tener al momento de explicar lo que necesita y lo necesario es entender las implicaciones desde el punto de vista operativo.

4.6.1 ANÁLISIS Y SOLUCIÓN DE ERRORES

Continuando con el análisis de los errores encontrados empezaremos con el primer error: Errores en la resolución de la aplicación.

Analizando el problema, el error se genera cuando de la aplicación en modo pantalla completa utiliza un programa externo, sucede cuando:

- Cuando se corre la aplicación en modo pantalla completa y se accede a un link externo, la aplicación correr en modo ventana para poder utilizar el programa externo al cual se esta llamando.
- De igual manera sucede el error en modo pantalla completa cuando se accede a las carpetas que contienen los archivos almacenados de fotografías o notas.

El error prevalece mientras la aplicación se encuentra en modo pantalla completa. La siguiente lista menciona los detalles que ocurren cuando sucede el problema:

- La aplicación cambia de pantalla completa a modo ventana cuando se necesita acceder a una aplicación externa como el explorador.
- Esto sucede porque no se puede manipular la aplicación de terceros mientras la aplicación corre en pantalla completa.
- Cuando la aplicación corre en modo ventana no hay forma de regresar la aplicación a pantalla completa hasta que el usuario salga al menú principal y habilite nuevamente la pantalla completa.
- En ocasiones la aplicación deja de responder cuando se quiere acceder a ella nuevamente.

Los problemas ocurren justamente después del cambio automático del modo de pantalla. Para solucionar el problema se decidió por controlar el modo gráfico cada ve que se accede a una aplicación, es decir, no permitir que la aplicación haga el cambio automático, si no cambiar de manera controlada el modo de pantalla y agregar un botón que permita habilitar nuevamente el modo de pantalla completa y continuar con el recorrido de forma normal.

Botones para cambio de modo pantalla

Del lado inferior derecho se agregaron dos controles de pantalla. El primero cambiará la aplicación a la máxima resolución en pantalla completa. El segundo hará un cambio de modo ventana a pantalla completa.

Figura 4.26 Botones para cambio de modo de pantalla

Debido a que el modo controlado de cambio de pantalla exige que se maneje una resolución menor que la de pantalla completa se agregó un botón adicional que permite correr la aplicación en modo ventana a la máxima resolución posible.

Con la adición de estos botones se resolvió el error de pérdida de control sobre la aplicación al tener la necesidad de salir de ella, así como manejar durante el recorrido ambos modos de pantalla sin la necesidad de regresar al menú principal.

Continuando con la lista de errores, el siguiente a tratar es el problema del video inicial. Los errores reportados son los siguientes:

- No se puede saltar el video.
- Al finalizar el video permanece por un tiempo la pantalla oscura y después aparece el menú principal.

Estos errores pueden ser tratados con facilidad ya que están vinculados directamente a la función que se encarga del manejo de video. Considerando la opción de salto de video, se eligió el no permitir al usuario saltar el video ya que se considera importante que el cliente lea por un espacio de 45 segundos el contexto por donde se estará recreando la ambientación

Con respecto al error del cambio de video a menú principal, el descuido se debió que al momento de terminar el video automáticamente, el control encargado de ejecutar el video no se detiene si no que sigue reproduciendo, por tal razón el video termina y el control sigue reproduciendo una pantalla en negro.

La solución para el problema anterior es fácil, simplemente es necesario controlar la reproducción del video añadiendo una función que se encargue de detener el control multimedia y salte directamente el menú principal.

Y no es opción agregar un control adicional para permitir el salto de video.

Para el siguiente problema sobre el manejo de los objetos colocados al interior del templo de Lourdes se enlistan los siguientes detalles:

- El manejo de colisiones por parte del motor es un detalle que no permite una movilidad casi natural de lo objetos colocados.
- No se permite atravesar objetos, haciendo muy difícil el manejo de los movimientos.
- El encimar objetos es imposible por cuestiones físicas como la gravedad.
- Pensar en una función que permita un movimiento rápido de los objetos frente al usuario.

Los datos anteriores brindan un panorama de los errores que hay que solventar. Haciendo las pruebas pertinentes en ambos sistemas, mas que un error, un detalle que valdría la pena considerar a fondo es quitar las colisiones a los objetos.

La colisión es una propiedad que permite saber cuando un objeto colinda o está ‘tocando’ a otro. Obviamente esta condición asemeja las particularidades de los objetos en la vida real, si un objeto choca con otro sucederá alguna de las dos situaciones siguientes:

- El objeto que se esta moviendo chocará con algún otro y moverá dicho objeto.
- El objeto que se esta moviendo chocará con algún otro y no permitirá un libre movimiento del mismo.

Si se elimina la colisión de los objetos se puede mejorar significativamente el movimiento de los mismos y realmente no es necesaria la colisión para estos elementos.

Después de haber reestructurado los modelos, tanto el movimiento como los errores de gravedad desaparecieron. El movimiento se volvió mas natural y la dificultad aminoró considerablemente.

Básicamente el error consistió en una sola cualidad inherente al objeto. Si se hubiera decidido atacar los errores de manera individual, probablemente se hubiera llegado a la misma solución, pero seguramente con una carga de trabajo y tiempo mucho mayor.

LAS FASES INICIALES PARA LA SOLUCIÓN DE ERRORES: IDENTIFICAR, RECOLECTAR, ANALIZAR, NOS PERMITIRÁN ABORDAR LOS ERRORES DE UNA MANERA FORMAL Y NOS FACILITARÁN LAS TAREAS DE CORRECCIÓN DE ERRORES. INCLUSO SI SE TIENE UN ESTUDIO GLOBAL DE ERRORES, SUELE PASAR QUE CON LA CORRECCIÓN DE UN ERROR SE SOLUCIONA MAS DE UN PROBELMA.

La siguiente dificultad a considerar es la falta de información. Aunque esto no es en si un error de la aplicación, si no un problema cultural, ya que la mayoría de los mexicanos no leemos el instructivo antes de usar un programa o un artefacto.

Pero si mas de la mitad de los usuarios a los que se les brindó la aplicación para las pruebas, opinó que era carente la información, no en el instructivo, si no durante el recorrido, había que solucionar ese inconveniente.

La pregunta es: ¿cómo atacar un problema que básicamente no se puede considerar como error?. Tanto los errores como las mejoras pueden considerarse en forma general como una tarea por solucionar. Y si las tareas a solucionar las vemos como defectos, se puede aplicar la misma metodología que nos propone Humphrey.

La decisión fue el agregar ventanas emergentes durante las fases mas importantes del recorrido, es decir obligar al usuario a leer al menos detalles importantes de la aplicación. Los puntos que se consideraron importantes para informar al usuario durante el recorrido fueron los siguientes:

- Al inicio del menú principal.
- Antes de escoger alguna misión.
- Al inicio del recorrido externo.
- Al inicio del recorrido por el Templo de Lourdes.

En conjunto, las cuatro ventanas emergentes informarán completamente al usuario de las funciones disponibles. Cada ventana informará recomendaciones pertinentes para cada sector del recorrido.

La ventana al inicio del menú informará de las opciones con las que cuenta la aplicación, la ventana de las misiones informará de forma somera las diferencias entre cada misión que se puede seleccionar y las dos ventanas faltantes cada una informará de las funciones activas que puede utilizar el usuario de acuerdo a la misión en la que se encuentre.

Por último basta solucionar el error del cursor. Un detalle que no fue fácil encontrar ya que no siempre ocurría, Por mencionar datos relevantes, el error jamás transcurrió durante la ejecución en el sistema Macintosh. Y cuando se utilizaba la aplicación en sistemas Windows, podía o no pasar. Sinceramente jamás se logro encontrar el error por el cual se produce tal defecto.

El error ocurre cuando:

- Se corre la aplicación en sistemas Windows.
- Cuando se activa el cursor con la tecla tabulador y al presionarla nuevamente no se desactiva el cursor.

- En cambios de modo de pantalla, sobre todo en modo ventana, no responde la tecla tabulador cuando el cursor está activo.

A pesar que se hicieron pruebas exhaustivas para encontrar el defecto no se logró identificar que es lo que producía la falta de respuesta al presionar la tecla tabulador. La conclusión fue que se debería a un error en el sistema operativo de Windows, ya que dicho error jamás se pudo reproducir en equipos Macintosh.

Al no contar con la suficiente información y datos que pudieran ayudar a resolver el problema de qué podría estar provocando el error del cursor, se decidió agregar un botón extra a la interfaz gráfica con una función asociada, que tanto en sistemas Macintosh como Windows principalmente, logrará desactivar el cursor por medio de la aplicación y no por uso del sistema.

Cursor y ventana emergente

Figura 4.27
Ventana emergente y botón de cursor

Con la resolución del problema del cursor se concluyen las tareas dedicadas a la eliminación de errores, solo falta validar las mejoras propuestas por los usuarios.

MUCHAS VECES ALGUNOS ERRORES NO ESTÁN VINCULADOS DIRECTAMENTE CON LA APLICACIÓN. POCAS VECES UN PROGRAMA, PARA TODOS SUS PROCESOS DEPENDE EXCLUSIVAMENTE DE SI MISMO. PARA ELLO HAY QUE REALIZAR ANÁLISIS EXHAUSTIVOS, PARA DESLINDAR LA APLICACIÓN DE ALGÚN DEFECTO QUE SE ENCUENTRE E INGENIAR IDEAS PARA SOLUCIONAR EL DEFECTO.

6.6.2 IMPLEMENTACIÓN DE MEJORAS

Las mejoras proveen al usuario, facilidad de uso de la aplicación que esta manejando. Normalmente las mejoras viene por parte de los usuarios finales, donde dan ideas y posibles mejoras que pueden agregarse a la aplicación.

De las ideas que proveyeron los usuarios que usaron la aplicación las siguientes se consideraron como viables para incorporarse a la aplicación:

1. Hacer mas fluido el movimiento de los objetos o tener combinación de teclas.
2. Movimiento rápido de los objetos.
3. Que no se encuentre en cero el valor por omisión para la altura del objeto en la ventana de ‘Mover Objeto’.
4. Conteo de objetos agregados en la Iglesia.
5. Si el usuario escogió iniciar con la misión exterior, al ingresar al Templo de Lourdes poder salir nuevamente a la misión en la que se encontraba.
6. Agregar vínculo hacia la página de la Arquidiócesis de México.

La idea uno quedo solventada de manera inmediata en pasos anteriores cuando se decidió eliminar del objeto la colisión.

Con respecto a la idea dos, gracias al hecho de haber eliminado la colisión es posible crear una función que mueva el objeto mucho mas rápido. Se ideo una forma de mover el objeto de manera rápida y muy fácil.

Se diseño una función que permita mover el objeto seleccionado justo frente al usuario. Para utilizar esta función se debe tener abierta la ventana de mover objeto y en lugar de utilizar las flechas para trasladar el objeto, basta con apretar la tecla ‘espacio’ y el objeto se moverá inmediatamente justo frente al usuario. De esta manera se evita la utilización de teclas y configuración de magnitudes para un movimiento rápido del objeto deseado. Con esta función se optimiza sobremanera la forma de trasladar objetos.

Para la mejora tres, es un error demasiado obvio que no debió haber ocurrido, de antemano no se puede programar valores de movimiento por omisión con magnitud nula. La corrección fue casi inmediata ya que los datos que se muestran en pantalla son variables que se inicializan con un dato, modificar este valor por la magnitud deseada no implica mayor costo en el proyecto.

La mejora cuatro es un poco mas laboriosa su elaboración como su explicación, por lo que es conveniente dejarla al final y continuar con las demás propuestas para la aplicación.

La idea cinco puede considerarse como una omisión en el Documento de Diseño, ya que si el recorrido por la parte exterior permite ingresar al Templo de Lourdes, es elemental salir nuevamente del Templo de Lourdes si así lo desea el usuario. En cambio, si desde un inicio se selecciona la recorrido por el Templo de Lourdes, sería ilógico que se permitiera la salida del interior de la Iglesia.

Para implementar esta mejora fue necesario numerar las misiones y guardar este valor mientras se estuviera dentro del templo de Lourdes. Al salir del templo de Lourdes, consultar el valor de la misión y cargar nuevamente la misión de acuerdo al valor guardado.

Para la idea seis, el acceso directo a la página de la Arquidiócesis de México es un dato que nunca se consideró, sin embargo, desde un principio de hablo de la barra superior localizada en el menú principal de la aplicación, dicha barra serviría para posibles vínculos que fueran necesarios agregar al recorrido. Una mejora fácil de solucionar.

Retomando la mejora cuatro, que habla sobre agregar un listado de todos los adornos colocados en el interior de la Iglesia. Que aunque no esta descrita en el Documento de Diseño, no se puede pasar por alto, aunque no se específico del por qué es necesario tener este conteo en la base del recorrido, es innegable que este dato es sumamente necesario para el usuario. Si la finalidad de agregar adornos en el templo de Lourdes es tener una idea de cómo quiere que luzca, al finalizar de arreglar el templo, lo menos que se podría querer es tener un listado de todos los objetos que se utilizaron para adornar la Iglesia.

Aunque se pudo haber omitido esta mejora, se consideró indiscutible que estuviera presente en la aplicación. No se iba a poner al cliente, después de finalizada su labor de adornar el templo, a contar todos los adornos colocados y mucho menos pedirle que los clasificara por nombre y anotar cuantos fueron de cada uno.

La observación anterior sirve como pauta para empezar a diseñar la función que se encargará de actualizar los datos y al finalizar mostrarlo al usuario.

Recordando un poco, los datos accesibles para el sistema de los objetos agregados por el usuario, es el nombre y la localización del archivo. Incluso

podemos agregar un valor adicional, que es el ID, porque para este momento ya se encontrará el objeto en la escena.

¿Qué datos se necesitan para poder desplegar el número de elementos agregados al entorno?. Basta con el nombre y el número de adornos.

Los datos con los que ya contamos es el nombre del adorno agregado, lo que hace falta es saber que número de objetos se han agregado por adorno.

Una de las soluciones factibles es agregar un script que lleve un conteo de elementos por nombre. La función se encargará de llevar el control de los elementos agregados a escena. Cada vez que se agregue un adorno, tomará el nombre del adorno, y lo enviará a una función que se encargará de clasificar el objeto y llevar el conteo del mismo. De la misma forma, cuando un elemento sea borrado, tomará el nombre y en lugar de agregar el valor, lo restará del total de objetos por nombre agregados.

Con esta forma no se comete el error de tener sumas negativas, ya que no se puede empezar a restar algún adorno si el mismo no ha sido agregado con anterioridad.

Solo basta saber cómo se va a desplegar la información ya almacenada. Lo mas correcto es que esta información aparezca en el archivo de notas que ha creado el usuario o crear un nuevo archivo con la lista de elementos agregados.

Para evitar un aumento considerable de archivos y convertir la aplicación en algo cansado al usuario por aumentar el manejo de archivos, se optó por agregar esta información al documento de notas justo al final del mismo.

El script que se encargará de realizar esta tarea lo hará de la siguiente manera:

1. Cuando el usuario decida salir de la misión al menú principal, se abrirá el archivo de notas para escritura.
2. Si existen objetos agregados en la escena, empezara un barrido de la lista que contiene los nombres de los adornos que han sido agregados, de lo contrario cierra el archivo de notas y se accede al menú principal.
3. Tomará el nombre del adorno y el valor asociado y lo escribirá al final del archivo notas.
4. Seguidamente continuará con el siguiente elemento, tomando su nombre y valor.
5. Concluido el barrido de la lista se finaliza la escritura del archivo notas y se accede al menú principal.

De esta manera se puede brindar al usuario la información del número de elementos agregados a escena y el nombre de los adornos.

Con esto se da por concluida la fase de pruebas de la aplicación y se procede a la validación de las pruebas finales para preparar el producto para la entrega final.

4.7 Validación y Producto Final

Con este tema se concluye el desarrollo del recorrido por el Templo de Lourdes, con anterioridad se han hecho las pruebas, tanto del lado del desarrollo como de los clientes finales, se han encontrado defectos que han sido analizados y solucionados así como se han estudiado las propuestas para mejoras del producto, las cuales han sido evaluadas y en algunos casos se han aceptado para su incorporación en el recorrido. Otras mas quedarán pendientes para alguna próxima versión del producto.

Siempre existirán mejoras para un producto, indiscutiblemente las mejoras proveen al usuario ventajas ya sea, sobre el mismo producto o sobre productos existentes en el mercado. Sin embargo, hay que saber decir hasta dónde es el alcance del producto que se esta programando. No saber dónde parar, solo es prueba de un inconsistente y débil trabajo evaluativo anterior. Ya que no se tiene una base sólida a la cual hacer referencia de los límites de la aplicación.

NO TENER CLARAS LAS METAS A LAS CUALES SE QUIERE LLEGAR EN UNA APLICACIÓN, ARRASTRARÁ PROBLEMAS AL FINAL DEL PRODUCTO, YA QUE SE CAERÁ EN EL PERFECCIONISMO, QUE NO ES LO MISMO QUE EL PERFECCIONAMIENTO. POR TAL RAZÓN, LA MAYORÍA DE LAS APLICACIONES CUENTAN CON VERSIONES QUE SON MUESTRAS DE SU PERFECCIONAMIENTO.

Teniendo claro que las mejoras pertinentes han sido añadidas y los errores corregidos solo falta validar la aplicación.

4.7.1 VALIDACIÓN

En general podemos entender la validación como la comprobación de que el producto este bien formado, tanto de forma visual como en su codificación y que se ajuste a la estructura definida, en este caso sería al Documento de Diseño y si existiese, a los diagramas de código programados para las funciones.

Antes de continuar hay que identificar un par de conceptos que pueden confundirse. Boehm (Boehm 1979) expresó la diferencia entre validar y verificar:[104]

- Validación: ¿Estamos construyendo el producto correcto?
- Verificación: ¿Estamos construyendo el producto correctamente?

La diferencia radica que en la verificación comprobamos que el programa está de acuerdo a las especificaciones previstas, como los requerimientos, funciones, etc. Mientras que en la validación aseguramos que el producto satisfaga al cliente.

La etapa de verificación se fue llevando a lo largo de la programación del recorrido, comprobando que las funciones fueran las correctas y que no generaran errores, que las interfaces gráficas respondieran tanto al diseño como a las necesidades del recorrido, por mencionar algunos ejemplos.

Podría creerse que la etapa de validación también ya se llevo acabo, en el momento en haberse dado una fase beta, la cual ocurrió al momento de entregarse la aplicación para pruebas, aunque estrictamente hablando, puede formar parte de las etapas de validación no necesariamente puede ser así.

---

[104] Sommerville Ian, Ingeniería de Software, 7ª Edición, Prentice Hall. p. 472

Existen pruebas de validación como las pruebas Alfa y Beta[105], que son pruebas de aceptación por parte de usuario, en donde el cliente valida todas las funciones existentes en la aplicación. La parte Alfa de la prueba se realiza por un cliente en el lugar de desarrollo, mientras que el desarrollador va observando la forma de interactuar del cliente con la aplicación y se van anotando los errores, registros de uso y observaciones pertinentes a la prueba, mientras que la fase Beta, se realiza por clientes pero en sus lugares habituales de trabajo, no estando presente el desarrollador, donde las observaciones ya no son apuntadas por el desarrollador si no que los registros de errores y observaciones son anotados por el cliente e informa de las irregularidades al desarrollador. En el caso de la aplicación, justamente se aplico la prueba Alfa y Beta .

La validación intenta demostrar que el software es justamente el que quiere el cliente, que satisface sus requerimientos.[106] La validación final se da del lado del equipo desarrollador, entregar la aplicación a un cierto número de usuarios sirve para obtener datos estadísticos y realizar pruebas de fiabilidad del programa, pero estos datos deben ser procesados y estudiados por parte del desarrollador, para así demostrar que la aplicación satisface los requerimientos del cliente y las necesidades de los usuarios.

Con los datos obtenidos de las pruebas y verificaciones que se han realizado al producto, la validación del producto ya puede llevarse a cabo:

- Las verificaciones hechas durante todo el desarrollo del recorrido aseguran que las funciones y el diseño esta apegado al Documento de Diseño obtenido a partir de los requerimientos establecidos.

---

[105] Pressman Roger, Ingeniería de Software. Un enfoque práctico. 5ª Edición, Mac Graw Hill. p. 316
[106] Sommervile, p. 474

- El prototipo entregado a los usuarios facilitó datos para la corrección de errores e implementar mejoras que estuvieran apegadas al Documento de Diseño.
- La información recibida con respecto a la usabilidad y fiabilidad de la aplicación permitió establecer parámetros de medida que apoyaron la sustentabilidad de que se esta entregando una aplicación factible a las necesidades del usuario final.

No obstante, no basta con la validación como prueba final para poder entregar un producto para el mercado. Un tema muy controversial y difícil de medir es la calidad de un producto. Controversial por el aspecto de que la calidad es diferente para cada usuario y esta se da de acuerdo a su experiencia y factores que marcan la satisfacción de sus necesidades.

El ejemplo en este caso sería la calidad de el recorrido virtual visto desde el punto de vista de una persona que pocas veces ha tenido contacto con este tipo de productos a comparación con alguna otra que tiene una basta experiencia en videojuegos, seguramente el punto de vista de la calidad puede variar significativamente.

4.7.2 PRODUCTO FINAL

En todo desarrollo de software, la integración es una etapa que suele pasar desapercibida, no el ámbito de construcción, si no que pocas veces puede considerarse como etapa de estudio. Generalmente se piensa que si todos los módulos o programas funcionan bien, seguramente sucederá lo mismo al momento de unirlos. No necesariamente tiene que pasar de ésta manera.

En nuestro caso se puede enfocar la parte de integración como dos tareas diferentes:

- Integración de funciones como un todo en el producto final.
- Integración del producto final en ambas plataformas.

No se había hablado antes de la parte de integración por el hecho de que a medida que las funciones se van construyendo, automáticamente pasaban a formar parte de la aplicación en si, en este caso jamás hubo división de tareas, si no que desde un principio la aplicación fue tomando forma.

Pude considerarse que la forma de integración del recorrido virtual por el Templo de Lourdes fue una integración incremental,[107] donde se fue construyendo por partes bien definidas y se estuvieron aplicando las pruebas en pequeños segmentos con lo cual fue mas fácil aislar y corregir los errores que fueran surgiendo.

En cambio, la integración con respecto a la plataforma donde se va a ejecutar la aplicación es diferente, el hecho de construir la aplicación con las bondades de que un solo desarrollo aplica para ambas plataformas (Windows y Macintosh) no precisamente ayuda con la parte de una integración funcional.

La integración incremental es muy fácil de implementar e incluso de sobrellevar, sin embargo, como sucedió en este caso, el haberse enfocado directamente al desarrollo no se preveo la idea que podría fallar ya trabajando en el sistema operativo y uno de esos errores fue el problema del cursor que si bien actuó correctamente en uno no significa que en el otro hiciera lo mismo.

[107] Pressman, p. 312

NUNCA HAY QUE OLVIDAR QUE LAS PRUEBAS DE INTEGRACIÓN SON SUMAMENTE IMPORTANTES ANTES DE ENTREGAR UN PRODUCTO. LAS PRUEBAS DE INTEGRACIÓN SON TÉCNICAS SISTEMÁTICAS PARA CONSTRUIR LA ESTRUCTURA DEL PROGRAMA Y AL MISMO TIEMPO IR DETECTANDO ERRORES ASOCIADOS CON LA INTERACCIÓN DE LA APLICACIÓN.

Para algunos, la calidad hace referencia a los estándares y procedimientos organizacionales de calidad que comprueban si el equipo de desarrollo los esta siguiendo o no, para algunos otros, la calidad depende de lo que se ha denominado "cultura de la calidad"[108] donde todos nos comprometamos a que el desarrollo sea elaborado y entregado con un alto nivel de responsabilidad.

La calidad a la que debemos aspirar es al compromiso de entregar un producto bajo los estándares y procedimientos organizacionales vigentes que hacen referencia al cumplimiento de normas establecidas para el tipo de producto que se está entregando, aunado a ello, con la suficiente información, manuales de usuario, instructivos de uso y con la seguridad que obtendrá del programa la satisfacción y los beneficios que espera conseguir al emplearlo, contando con la garantía de un soporte técnico y la fiabilidad que no generará problemas graves en su uso ya sea en sus equipos o de manera personal. Obviamente para lograr lo anterior debe haber toda una gestión de procesos encaminados a entregar un producto confiable.

Entonces, si la finalidad es entregar una aplicación confiable, el producto final debe contar con todas esas características necesarias.

[108] Sommerville, p. 588

Asegurar que sea capaz de correr en las dos plataformas sin ningún problema, entregar un producto capaz de contar con versiones futuras (mejoras y procedimientos adicionales que no fueron contemplados durante la etapa de requerimientos), tener un servicio de soporte cercano al cliente, ya sea por correo electrónico, número telefónicos o redes sociales, facilitar información suficiente del producto para su manejo y usabilidad y sobre todo, mantener la ventana abierta a actualizaciones periódicas por carencias o problemas inherentes a la aplicación.

Finalizar la aplicación para que esté preparada para todos los retos que se presenten, justamente después de que sea entregada e incluso distribuida a los usuarios, es lo que puede considerarse como la entrega final de un producto de calidad.

Para el caso del recorrido por el Templo de Lourdes, la entrega será por medio físico a través de un CD o por descarga vía Internet, cumpliendo justamente con el parámetro de portabilidad. La aplicación será capaz de conectarse a internet e informar al usuario si hay actualizaciones disponibles para su descarga, así cómo una configuración básica y prioritaria para sistemas de bajo poder de procesamiento para la primera vez que la aplicación sea instalada y ejecutada, las configuraciones por defecto son las siguientes:

- Instalación Básica: Copiar y pegar carpeta en cualquier lugar del equipo que desee el usuario.
- Inicio en modo ventana.
- Resolución mínima: 800 x 600.
- Profundidad de bits: Profundidad estándar utilizada por el sistema.

Las demás configuraciones correrán a cargo del usuario permitiendo configurar la aplicación a conveniencia y dichas configuraciones quedarán guardadas para el próximo inicio de la aplicación sin la necesidad de configurar nuevamente el recorrido.

Material físico como manuales de usuario e instructivos no estarán disponibles, sin embargo se encontrarán dentro del CD o descarga vía web para su posterior lectura e impresión por parte del usuario.

¿CÓMO SABER CUANDO UNA APLICACIÓN ESTÁ FINALIZADA? CUANDO CUMPLE CON EL OBJETIVO ORIGINAL. SI EL PRODUCTO SATISFACE CON TODOS LOS REQUERIMIENTOS Y SE ADECÚA A LAS NECESIDADES DEL CLIENTE PODEMOS ASEGURAR QUE HEMOS CUMPLIDO CON LA META PROPUESTA.

Conclusión

CONCLUSIONES

El desarrollo de recorridos virtuales interactivos es una realidad. Se tienen aplicaciones semejantes que en la mayoría de los casos, brindan poca o nula interacción con el usuario. Algunos de ellos son recorridos exclusivamente informativos, por ejemplo, presentan la disposición de un inmueble con la posibilidad de tener una panorámica de 360º y un movimiento pobre a lo largo del mismo. Algunos otros, con un poco de mayor enfoque, presentan al usuario información (no sólo la visualización del entorno) de lo que esta observando, lo cual no significa que haya una interacción recorrido-usuario.

Los alcances de los recorridos virtuales interactivos son enormes, aplican tanto para información general como para programas de capacitación. Si una empresa recluta gente del extranjero, puede recurrir a aplicaciones de este estilo para que el trabajador (antes de conocer el espacio físico de la empresa) pueda recorrer la misma e incluso si fuese necesario, interactuar con ambientes programados dentro del recorrido.

Una vertiente que no se ha explotado aún pero que se puede considerar imprescindible es el área educativa. Si proyectos como los recorridos virtuales interactivos se desarrollaran para escuelas que no cuentan con la suficiente infraestructura para albergar grandes laboratorios o contar con instrumental para recrear eventos, por ejemplo de física, se puede recurrir a desarrollos de este tipo

para generar entornos lo más reales posibles; con esto el alumno podrá entender, resolver e interactuar con la aplicación los problemas que se le presenten para solucionarlos. Si se brindan herramientas de este tipo y no sólo se quedan los ejercicios en hojas y lápiz, la comprensión de los problemas puede darse de una manera más fácil. Aunque los alcances del proyecto no apuntaron hacia un desarrollo educativo, es una propuesta que podría tomar forma.

Como consideramos que los recorridos virtuales interactivos pueden ofrecer más oportunidades que las opciones comunes, se decidió por una aplicación de este tipo para el Club de Banqueros de la Ciudad de México, cuyo problema principal (como mencionamos en los primeros capítulos) es la dificultad que tienen sus integrantes para visitar personalmente las instalaciones del templo de Lourdes.

La idea principal era recrear el entorno, tanto de los alrededores del templo como al interior del mismo, pero no ajeno a lo anterior, también se pedía que el usuario pudiera escribir sus reseñas durante el recorrido así como ofrecer la posibilidad de guardar imágenes del entorno para que pudiera acceder a ellas cuando necesitara. Pensando que los clientes que usarían la aplicación realizarían su evento en el templo, se pidió un extra como estimulo al cliente: tener la opción de adornar el interior del templo con cierta gama de flores existentes en el mercado.

Obviamente la exigencia de los requerimientos rebasaba la programación de un recorrido virtual ordinario. Además, uno de los inconvenientes principales al que se tuvo que enfrentar fue el bajo costo del proyecto, en comparación con los desarrollos comunes de realidad virtual. Acercarse a tecnologías alternas (no precisamente para la programación de recorridos virtuales) que permitieran la elaboración de ambientes tridimensionales, permitió resolver satisfactoriamente el problema.

Para ello se decidió utilizar una plataforma conocida como '*game engine'* la cual permite recrear ambientes tridimensionales a bajo costo con la posibilidad de agregar programación adicional para nuevas funciones.

La ventaja de utilizar plataformas de este tipo es que la programación se ve reducida exclusivamente al desarrollo del recorrido, pues el motor se encarga de comunicarse con la plataforma para ejecutar la aplicación.

Con lo anterior se logra resolver la afinidad del recorrido con respecto al sistema operativo donde se va a ejecutar la aplicación. Es decir, en algunos casos, ciertas plataformas permiten que un solo desarrollo pueda ejecutarse en los sistemas operativos más comunes como Windows, Macintosh o Linux.

En el caso del recorrido por el templo de Lourdes, el '*game engine'* que se utilizó fue *Torque*, que permite tanto la elaboración de entornos tridimensionales como facilitar la compatibilidad de un solo desarrollo con sistemas Windows o Macintosh. Obviamente lo anterior reduce el tiempo de programación y los costos de una manera significativa.

Además con desarrollos de esta magnitud se puede dar solución a proyectos pendientes en México, como la digitalización de acervos y recintos culturales:

> *"De acuerdo con Ivonne Lonna Olvera, académica del Departamento de Arte de la Universidad Iberoamericana (UIA), son contados los museos que han intentado digitalizar su acervo.*
>
> *"Algunos lo intentan, ponen en línea el catálogo de su acervo, con imágenes de muy mala resolución, como si la descargaras de Google", dice.*
>
> *"Los recorridos virtuales deberían ser de tal manera que la gente pueda tener una inmersión en el sitio, que pueda ver en donde está expuesta la obra, que*

se mueva por el espacio y que si le llama la atención una pieza pueda acercarse y tener la apreciación de esa pieza, pero que también tenga una descripción bien documentada, y que además se pueda descargar en alta resolución".

La convergencia digital de algunos museos mexicanos es preocupante, comenta Lonna Olvera. "El Museo Tamayo es de arte contemporáneo y se está durmiendo y eso no puede ser. En otros casos hay problemas técnicos, a veces las páginas están mal construidas, el sistema de navegación está mal hecho o más bien no existe. Los recorridos virtuales están, pero es una visita rápida y no te permite detener en la obra y apreciarla, verla a grandes resoluciones y tener al mismo tiempo la información".

Pedro Ángeles Jiménez, coordinador del Archivo Fotográfico "Manuel Toussaint" del Instituto de Investigaciones Estéticas (IIE) de la UNAM, señala que en cuanto a la catalogación y digitalización de los acervos, a México le falta mucho por hacer. "En México tenemos una asimetría enorme, en términos de cómo nuestras instituciones representan sus acervos en línea. Basta con comparar las páginas de los museos nacionales ante los extranjeros. Si consultamos la página del Museo del Louvre o del Museo Británico, vemos que casi todas sus piezas están disponibles en Internet, muy diferente a que si consultamos las páginas de los museos nacionales. La asimetría es inmensa", dice.

Aunque Ángeles Jiménez destaca la labor que el INAH ha emprendido con los proyectos de digitalización del acervo de algunos museos, opina que los catálogos deben ser de excelente calidad. "Vale la pena decirle al INAH que las imágenes no deben ser tan pequeñas como se muestran. La gente no está pensando en robarse nada, lo que se necesita es acceso a la información. Si vemos el catalogo del British Museum, las imágenes no son de dos centímetros, son de buena resolución y accesibles al público"…

Una de las ventajas que ofrece la digitalización y que en México los museos no están aplicando es la disposición en línea del acervo completo, incluido lo que se guarda en bodega.

"Los museos deberían ofrecernos no sólo lo que se ve en la sala, sino también las piezas que están guardadas en las bodegas, a las que sólo podríamos tener acceso de esta manera", comenta Pedro Ángeles Jiménez."[109]

Romper el paradigma de utilizar lenguajes tipo VRML para desarrollar un producto de realidad virtual permitió acercar este tipo de aplicaciones a un contexto ordinario. Lo anterior nos permite objetar la opinión común, según la cual desarrollar recorridos virtuales es sinónimo de inversiones elevadas, ya sea por el hecho de tener equipo de computo exclusivo para programar aplicaciones virtuales o por la necesidad de contar con programadores experimentados en el área.

Ahora el Club de Banqueros cuenta con una aplicación innovadora que permitirá estar mas cerca de sus clientes, brindándoles la oportunidad de conocer el templo de Lourdes sin la necesidad de estar presentes en el lugar, así como interactuar con el entorno para que el cliente pueda planear su evento de una manera mas significativa por medio de las funciones adicionales presentes en el recorrido. Confiando que la aplicación haya superado las expectativas del cliente y sea aprovechada al máximo por sus usuarios.

[109] El Universal, http://www.eluniversal.com.mx/cultura/66703.html (ví: 21 de Octubre de 2011)

APÉNDICE

APÉNDICE 1

CONTROL DE VERSIONES

¿Qué es el *control de versiones*? Es el proceso de gestionar múltiples versiones de una pieza de información, ya sea un programa, un archivo, datos, etc.[110]

Manejar el control de versión de un archivo de forma manual es un error muy común, un ejemplo es el crear de un archivo diferentes copias del mismo, ya sea colocándolos en diferentes carpetas, o en una misma carpeta pero con diferentes nombres, regularmente se utiliza el nombre mas la fecha de modificación. Sin embargo contamos con software especializado que nos permite llevar un adecuado control de versiones de nuestros archivos de forma automática y mas sencilla.

Pero, ¿para qué me sirve tener un control de versiones?. Un control de versiones es sumamente importante, ya sea cuando trabajemos en forma colaborativa o de manera individual, el control de versiones nos da las siguientes ventajas:[111]

- Automáticamente contamos con el historial y la evolución del archivo. Es decir, por cada cambio que se realice al archivo, existirá una anotación diciéndonos, quién lo hizo, cuándo lo hizo, qué cambio hizo y por qué lo hizo.

---

[110] O'Sullivan Bryan, Mercurial: The Definitive Guide. http://hgbook.red-bean.com/read/how-did-we-get-here.html. (ví:26 de Septiembre de 2011)
[111] Loc. Cit.

- Cuando estamos trabajando de forma colaborativa, aplicaciones de control de versiones simplifican en gran manera el trabajo en equipo. Por ejemplo, si se hicieron cambios simultáneamente en un archivo y estos son potencialmente incompatibles, el control de versiones nos ayuda a identificarlos y arreglarlos.
- Una de las mas grandes ventajas es permitir recobrar antiguas versiones si por alguna razón se cometió un error y se es incapaz de poder corregirla.
- Y como el nombre lo indica, ayuda a trabajar simultáneamente en varias versiones de un proyecto.

Existen varias formas de identificar este tipo de programas:

- Control de revisión (Revision Control, RCS).
- Software de gestión de configuración (Software Configuration Management, SCM).
- Administrador de código.
- Control de Versión (Version Control, VCS)

Tenemos varias herramientas dedicadas al control de versiones:

SUBVERSION

Es la herramienta de control de versiones mas popular en el mercado y desarrollada para remplazar a la herramienta CVS (no confundir con la abreviatura de control de versiones), la cual mantiene una arquitectura centralizada de cliente-servidor.

GIT

Es una herramienta de control de versiones desarrollada para manejar los desarrollos del kernel de Linux. Sale sobrando decir que su rendimiento en sistemas Linux es formidable, sin embargo en Windows su rendimiento no es tan envidiable en comparación a otras herramientas de control.

CVS

Es probablemente la herramienta de control mas usada en todo el mundo. Así como subversión, también cuenta con una arquitectura cliente-servidor.

MERCURIAL

No es una herramienta tan popular como subversión, ya no es tan visual y sus comando son directamente en consola, sin embargo guarda un gran rendimiento en cualquier sistema, inclusive Macintosh además de tener gran ventaja sobre otros en su manejo de versiones.

Escoger una herramienta adecuada dependerá de la experiencia de cada equipo de desarrollo, no existe la herramienta de control de versiones que se adecue a todas las necesidades. Para el caso del desarrollo por el Templo de Lourdes se decidió utilizar Mercurial como herramienta como control de versiones para el proyecto.

Entre las ventajas que se encontraron para decidir utilizar Mercurial fueron:

- Fácil instalación en los sistemas operativos mas populares. Windows, Linux y Macintosh. Necesario al menos en dos de los tres sistemas para el proyecto.
- Facilidad de los comando de uso.
- Detalladas revisiones de control, como los ‘benchmarks’ (puntos de referencia).
- Fácil uso de importación y exportación de historiales de control de revisiones de otros servidores.
- Los repositorios creados con mercurial no necesitan mantenimiento.
- Incluso Mercurial puede importar historiales de revisión de CVS.

Además de las ventajas anteriores, si deseamos exportar nuestro desarrollo a otro herramienta de control de versiones, cuenta con un comando ‘convert’ que

nos permitirá exportar nuestro historial de revisiones a las siguientes herramientas: Subversion, CVS, GIT, Darcs.

CONTAR CON UNA HERRAMIENTA DE CONTROL DE VERSIONES NOS AHORRARÁ MUCHO TRABAJO, YA QUE TENDREMOS CONTROLADO TODO EL HISTORIAL Y EVOLUCIÓN DE NUESTRO PROYECTO. INCLUSO CONTAMOS CON HERRAMIENTAS EXTRAS COMO INSERCIÓN DE CÓDIGO VIA REMOTA O CONSULTA DE AVANCES VIA INTERNET.

ÍNDICE DE GRÁFICOS

BIBLIOGRAFÍA

- Parra Márquez, Juan Carlos, A.A.V.V. Introducción Práctica a La Realidad Virtual. Concepción. Ediciones U. Bío-Bío, 2001.

- Gutiérrez A, Mario A., A.A.V.V. Stepping into Virtual Reality. London: Springer, 2008.

- Grigore C. Burdea, Philippe Coiffet. Virtual Reality Technology. Second ed: John Wiley & Sons, Inc., 2003.

- Slater M., Wilbur S. A Framework for Immersive Virtual Environments (Five): Speculations on the Role of Presence in Virtual Environments, 1997.

- Avgerakis, George. Digital Animation Bible: Creating Professional Animation with 3ds Max, Lightwave, and Maya: McGraw-Hill, 2003.

- Rodriguez, Edward. Computer Graphic Artist. Delhi: Global Media, 2007.

- Wayne, Carlson. A Critical History of Computer Graphics and Animation: The Ohio State University, 2003.

- Sherrod, Allen. Game Graphics Programming: Thomson, 2009.

- Whyte, Jennifer. Virtual Reality and the Built Environment: Architectural Press, 2002.

- Lawrence, R.J. Housing Dwellings and Homes: Design Theory, Research and Practice: John Wiley & Sons, 1987.

- Gordon Bob, Gordon Maggie. The Complete Guide to Digital Graphic Design: Thames & Hudson, 2005.

- Paterson, Ian. A Dictionary of Colour: A Lexicon of the Language of Colour: Thorogood Publishing Ltd, 2003.

- Tracy Diane, Tom Cassidy. Colour Forecasting: Blackwell Publishing Ltd, 2005.

- Levkowitz, Haim. Color Theory and Modeling for Computer Graphics, Visualization, and Multimedia Applications: Kluwer Academic, 1997.

- Gauthier, Jean-Marc. Building Interactive Worlds in 3d: Virtual Sets and Pre-arVisualization for Games, Film, and the Web: Elsevier, 2005.
- Acharya, Tinku. Image Processing Principles and Applications: Wiley-Intersciencie, 2005.
- Eberly, David H. 3d Game Engine Architecture: Morgan Kaufmann, 2006.
- Eberly, David H. 3D Game Engine Design A practical approach to Real-Time computer graphics: Morgan Kaufmann, 2001.
- Les Pardew, Alpine Studios. Game design for teens: Thomson, 2004.
- Wilde, Martin. Audio programming for interactive games: Elsevier, 2004.
- Austerberry, David. The technology of video and audio streaming: Elseiver, 2005.
- Duggan, Michael. Torque for Teens: Thomson, 2008.
- Maurina, Edwar F. The Game Programmer's Guide to Torque, CG Press 2006
- Kenneth C. Finney. 3D Game Programming All in One, Thomson, 2004
- Kenneth C. Finney. Advanced 3D Game Programming All in One, Thomson, 2005
- Humphrey, Watts S. Introducción al Proceso Software Personal, Addison Wesley, 2001
- Pressman Roger, Ingeniería de Software. Un enfoque práctico. 5ª Edición, Mac Graw Hill, 2005
- Sommerville Ian, Ingeniería de Software, 7ª Edición, Prentice Hall, 2002