“Metodología para generar un proyecto interactivo tridimensional: Paseo virtual por la zona arqueológica Tenochtitlán en 3D”

Trabajo escrito en la modalidad de desarrollo
de un caso práctico que para obtener el título
de ingeniero en computación, presenta:

ROSY YALIM AGUIRRE CRUZ

ASESOR: M. EN I. ELIO VEGA MUNGUÍA

UNAM – Universum - FES Aragón
MÉXICO, 2009

Agradecimientos.

Agradezco primero a Dios por darme la oportunidad de formar parte de este increíble proyecto.

Agradezco enormemente el apoyo de Universum para la realización del proyecto y en especial al Maestro en Ciencias Manuel E. González Casanova Almoina por aceptarme dentro de su equipo de trabajo.

Mi especial agradecimiento por su apoyo y enseñanza al Ing. Francisco Salgado Rodríguez, sin él no existiría este proyecto, ni mi participación en el mismo.

Gracias a mis hijos por el apoyo incondicional que me dieron para la realización de este proyecto. Caleb y Yaretzi los quiero.

Gracias a mi madre por darme ánimos para terminar el proyecto, además de su valioso tiempo con mis hijos, mientras trabajaba en este proyecto.

Gracias a mi familia y amigos que estuvieron apoyándome con su ánimo para que no desertara del proyecto.

Gracias a Dashiell por ser mi incondicional en este proyecto, tu apoyo y ayuda fueron de gran utilidad.

Gracias a J por mostrarme mi realidad, lo que me impulsó a iniciar este proyecto en mi vida.

Espero que este escrito apoye y ayude a desarrollar nuevos proyectos con mira al desarrollo de nuevas tecnologías de programación orientadas a la realidad virtual y la programación en tiempo real de México.

Índice.

Prefacio.

El objetivo de este texto es organizar la información necesaria para ayudar a realizar proyectos tridimensionales de realidad virtual, dando los consejos necesarios para que la implementación de este tipo de proyectos se realice de forma ágil y eficaz, haciendo la creación más rápida y con el mínimo de complicaciones.

Para su consulta se incluyen algunos antecedentes de forma sencilla, para no saturar de información este manual.

De igual forma se abordan los temas con un breve antecedente histórico y descriptivo, para después dar las pautas para su integración en un proyecto 3D interactivo. Se busca adaptar los temas para un auditorio lo más amplio posible; el lector tendrá acceso a conocimientos básicos en el manejo de las aplicaciones computacionales aquí utilizadas, así como también de conceptos de física y geometría analítica para comprender mejor algunos temas.

Es importante destacar que el tipo de proyectos que se podrán generar con las especificaciones y recursos descritos en el escrito presente, se podrán proyectar en salas especiales para proyección 3D, con las características particulares que en este caso tiene la sala "Ventana de Euclides" del museo de ciencias Universum. Dichas características serán descritas más adelante.

1. Introducción.

1. 1 Motivación.

La computación gráfica es una de las ramas de las Ciencias de la computación de mayor impacto social y acercamiento al público en general. Actualmente, casi todo sistema o programa interactivo tiene una interfase gráfica... no sólo en aplicaciones específicas de graficación matemática y estadística, sino también en los lenguajes de programación visual, las interfases gráficas con sistemas de ventanas, la metáfora del escritorio y en los sistemas de diseño asistido[1].

La computación gráfica con la ayuda de la realidad virtual interactiva crea proyectos de gran interés a nivel educativo, social y comercial. Desafortunadamente suelen costar mucho trabajo, organizacional, funcional y monetario, por lo que el objetivo de este documento es minimizar los costos para que estos proyectos tengan, por fin, el auge que merecen sin los costos excesivos que representan.

1.2 Aplicaciones.

Son muchas las aplicaciones para la computación gráfica, desde la aparición de la PC y su entorno gráfico, la mayoría de los usuarios tendemos a usar aplicaciones gráficas por que su uso es de fácil comprensión, además mejora la visualización de lo que estamos haciendo.

El desarrollo de videojuegos y entornos gráficos de simulación facilitan la comprensión de los procesos que interactúan en ellos. Esto ha hecho más rentable el desarrollo de proyectos gráficos orientados para usuarios de cualquier edad. Así también los entornos de programación se vuelven gráficos para facilitar y agilizar el desarrollo de proyectos de cualquier tipo, aún más los que son para desarrollar aplicaciones gráficas: se vuelven más complejos y más completos, más detallados, con una base de datos que puede personalizarse, pueden incluso recrear condiciones de prueba para verificar que el proyecto desarrollado no tenga errores, en especial si se trata de uno que se llevará a cabo en la vida real, como una tubería de gasoducto en una plataforma de exploración o un sistema de cableado eléctrico dentro de una construcción, por citar algunos ejemplos.

1 Claudio Delrieux, Introducción a la computación gráfica, Apuntes, pag.8.

1.3 Desarrollo Histórico.

El objetivo de la *Visión por Computadora* (*Computer Vision*) es permitir a una computadora procesar e interpretar un entorno a través de información visual. Debido a la importancia de la información visual en las actividades humanas éste ha sido un tema estudiado con gran interés en los últimos años. Otros nombres utilizados habitualmente además de *Computer Vision* son *Image Understanding* (Procesamiento de imagen) y *Machine Vision* (Visión artificial).

Los diferentes componentes de la visión tridimensional por computadora pueden verse en el diagrama de bloques siguiente:

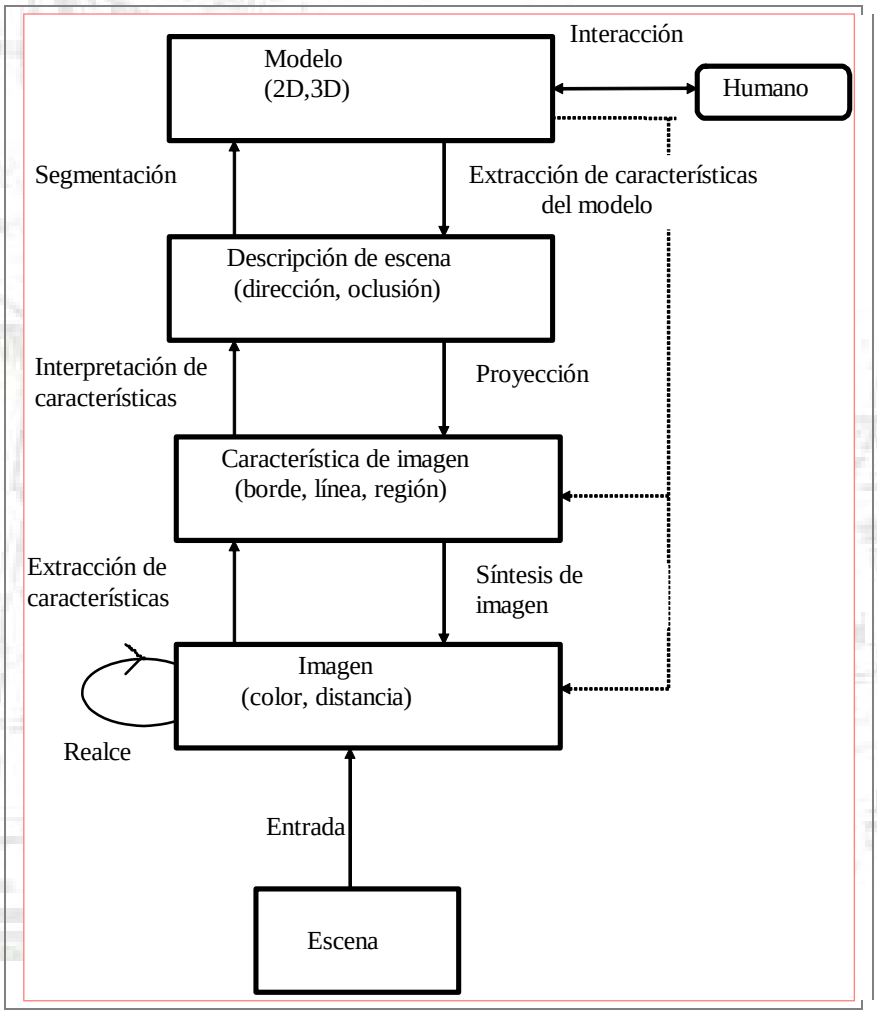


Figura 1. Componentes de la visión por computadora

En primer lugar se adquiere información visual de una escena 3-D (intensidad luminosa, color o distancia). El resto del proceso depende de los objetivos, pero un

proceso típico sería el siguiente:

1- Extracción de características (de bordes a regiones), que pueden o no corresponder a bordes o superficies de la escena real. Este es el llamado procesamiento de bajo nivel.

2- Descripción de la escena a partir de las características deseadas.

3- Reconocimiento basado en modelos previamente almacenados. Este punto y el anterior son el procesamiento de alto nivel.

Figura 2. Sinóptico del esquema de visión por computadora

El proceso general pero no siempre efectivo es el Bottom_Up, siguiendo de abajo hacia arriba de la figura 1. Si se imponen suficientes restricciones en la clase y orientación de los objetos en la escena, puede usarse la aproximación *Top-Down*, en que se compara la imagen con proyecciones creadas a partir de los modelos conocidos.

La teoría de la Visión por Computadora fue establecida por Marr[2] a finales de los 70, a partir de sus estudios de la visión humana. Según Marr, el proceso de la visión transforma las imágenes originales en *tres niveles de representación*: *primal sketch*, *2.5D sketch* y *representación 3-D*. Se pasa sucesivamente de la imagen original (pixeles) al *primal sketch* (características), al *2.5D sketch* (superficies), y finalmente a la *representación 3-D* (objetos).

En el ***primal sketch*** se obtiene una serie de características de la imagen, obtenidas a partir de los cambios de intensidad. Estos cambios suelen llamarse bordes, aunque no tienen el significado físico correspondiente. Para la obtención de estos bordes se utiliza una serie de filtros. Una vez obtenidos, estos bordes se agrupan de acuerdo

2 Marr, David, Neurocientífico Inglés. Mediante su modelo de la función del cerebelo, es posible emular la visión humana por computadora, a través de la interpretación de la información visual. Es conocido como el fundador de la disciplina de la Neurociencia Computacional.

a su posición y orientación.

El siguiente paso, el ***2.5D sketch*** es una etapa intermedia entre las representaciones **2D** y **3D**. Puede ser un mapa de profundidades de la escena, o bien la obtención de las normales a las superficies presentes en la misma. La entrada a esta etapa son las características obtenidas y agrupadas en el *primal sketch.* Existen varias maneras de abordar esta etapa. Normalmente se continúa la aproximación *bottom-up* del *primal sketch*, en el sentido de que no se utiliza ningún conocimiento sobre la escena, sino que se usan otras pistas, como conocimiento de la iluminación, o de la posición de las cámaras. Básicamente es el reconocimiento de la iluminación o sombras que dan profundidad a la escena.

La representación 3-D es la etapa final del paradigma de Marr. Se trata ya de una representación de la escena centrada en los objetos, permitiendo la descripción de los mismos independientemente del punto de vista. Para esta etapa ya se debe usar una aproximación top-down, basada en modelos de los objetos. Esta es la etapa más difícil, y su implementación con éxito parece todavía remota.

2. Requerimientos.

2.1 Introducción.

Un proyecto tridimensional interactivo debe cumplir con ciertos requisitos para no ser identificado como un videojuego. El principal requisito es que tenga la característica de visión estereoscópica (3D), además, que tenga varias interacciones por parte del usuario (para ser interactivo) y no de la computadora. Si agregamos una ambientación lo más real posible y buscamos que los personajes que intervienen sean fáciles de usar por el usuario, entonces tenemos un buen proyecto en nuestras manos.

Se debe plantear un objetivo específico y detallado de lo que hará o se espera del proyecto una vez que haya sido terminado.

Para que el desarrollo del proyecto sea más ágil, describiré algunos consejos útiles en forma de recuadro, y enumeraré pasos a seguir dentro del documento para consulta rápida.

2.2 Requerimientos de Hardware.

Para llevar a cabo aplicaciones interactivas tridimensionales debemos contar con ciertas características en nuestro equipo de cómputo para el desarrollo óptimo de una aplicación similar a la que presentamos como ejemplo en este documento.

Estas características son:

- ✓ Procesador Intel Pentium 4 (mínimo) 3GHz (mínimo)
- ✓ Sistema operativo Windows XP o superior
- ✓ DirectX Versión 9 o superior
- ✓ Tarjeta de video Nvidia Procesador Quadro FX 3400 / 4400
- ✓ BUS PCI Express x 16
- ✓ Memoria RAM 2GB
- ✓ Cañones (2)
- ✓ Pantalla de proyección traslúcida polarizada
- ✓ Joystick inalámbrico (u otro dispositivo, i. e. wii)
- ✓ Filtros de polarización circular.
- ✓ Lentes polarizados circulares.

Cabe señalar que el desarrollo de una aplicación tridimensional puede empezarse con una PC. Para la visualización real del proyecto que se está generando, se debe contemplar el contar con los requerimientos aquí mencionados para observar a detalle todo el procedimiento.

Descripción de los elementos.

La tarjeta de video permite la proyección simultánea de 2 imágenes, éstas se proyectan a través de los cañones (conectados a cada una de las salidas de la tarjeta de video). Las imágenes de los cañones necesitan ser filtradas y proyectadas, usando la técnica de polarización de la luz (que consiste en la supresión de ondas de luz en una dirección o polaridad), hacia una pantalla especial que no rompa el plano de polarización. Recibimos la señal visual con un filtro fijo a 90 ° (lentes polarizados) que nos darán dos imágenes una para cada ojo, las cuales interpretaremos como imagen 3D.

Este es el mejor método de proyección de video (o cine) 3D, pues se conserva la calidad de imagen, los fantasmas disminuyen y el cansancio visual del espectador es mínimo, el inconveniente es que se requiere equipo especial de proyección y pantallas especiales.

2.3 Requerimientos de Software.

Es importante contar con una herramienta de diseño para crear proyectos con mejor perspectiva de la realidad, así mismo es mucho más fácil detallar los objetos que se usarán para la interacción del mismo.

Texturas y formas se verán beneficiados en el proyecto tridimensional si ponemos un poco más de atención durante el desarrollo estructural de los objetos dentro de una aplicación de diseño.

Existen diversas herramientas de diseño en el mercado, las que se usaron para este proyecto en particular son:

- ✓ 3D Studio Max. Es un programa de creación de gráficos y animación 3D más utilizados por los desarrolladores de videojuegos, aunque también

en el desarrollo de proyectos de animación como películas o anuncios de televisión, efectos especiales y en arquitectura. Dispone de una sólida capacidad de edición, arquitectura de plugins y una tradición en plataformas Microsoft Windows.

- ✓ Virtools Ver. 3.5. Provee una herramienta eficiente para crear complejos sistemas VR (Realidad Virtual) que potencialmente se ejecutan en una PC, usando esterovision.
- ✓ PhotoShop. Herramienta útil en el retoque de archivos utilizados para simular luz y sombra o texturas dentro del proyecto.

Para la empezar con un proyecto, recomiendo archivar la información que se va a trabajar de la siguiente forma, a partir de la carpeta donde se instala el software de Realidad Virtual y manejo de eventos, en este caso, Virtools/:

Proyecto general:

… /virtools/nombre-proyecto/nombre-proyecto.cmo

Debido a que se requieren gran variedad de recursos multimedia tales como: audios, imágenes, modelos tridimensionales, animaciones, etcétera, es necesaria la creación de una serie de carpetas virtuales con el fin de mantener un orden y una estructura que permita realizar las actualizaciones y el mantenimiento del sistema de una manera mas eficiente. Es importante que, si se cuenta con un equipo de trabajo, todos los integrantes sepan la estructura de archivo de información a fin de que cualquiera de ellos pueda encontrar la información que necesite al momento.

La estructura propuesta es la que se enlista a continuación:

… /virtools/samples/nombre-proyecto/recursos/
… /virtools/samples/nombre-proyecto/2DSprites/
… /virtools/samples/nombre-proyecto/3DSprites/
… /virtools/samples/nombre-proyecto/Scripts/
… /virtools/samples/nombre-proyecto/Caracteres/
… /virtools/samples/nombre-proyecto/sonidos/
… /virtools/samples/nombre-proyecto/texturas/

Archivos para su ejecución automática:

…/virtools/nombre-proyecto/

Ayuda, tutoriales y demo:

…/Virtools/ documentation/

Carpetas de DEMOS de ayuda de Virtools (building blocks):

…/virtools/documentation/CMOS/BBexamples

3. Geometría del espacio.

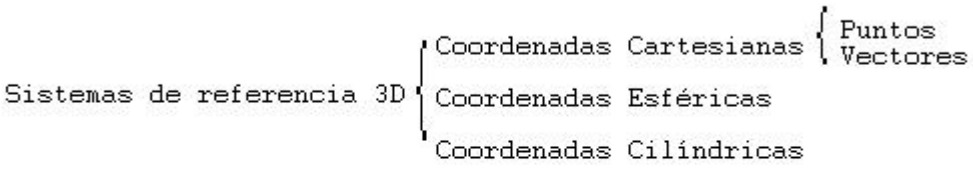
3.1 Sustento teórico.

Un punto base para la realización de un proyecto interactivo tridimensional es la geometría y arquitectura del escenario. Para ello tomamos la información de fotografías, maquetas, planos, libros, etc. que nos ayuden a recrear un conjunto de objetos tridimensionales que simulen a los reales.

La ambientación que se desea obtener para el proyecto dependerá de la información real que obtengamos para ello. Mientras más real se requiera, mayor será la investigación.

Geometría cartesiana.

Asigna valores numéricos a cada punto del plano o del espacio. Describe las entidades geométricas mediante fórmulas (ecuaciones) que verifican estos números. Utilizando la geometría clásica que define las entidades geométricas mediante propiedades descriptivas; como ejemplo podemos citar: lugar de puntos equidistantes de uno dado, camino más corto entre dos puntos, etc. Para los procesos que lo requieran.



Representación de líneas 3D.

Se realiza mediante 2 o más funciones (fórmulas) expresando relaciones entre las coordenadas de los puntos que constituyen las líneas.

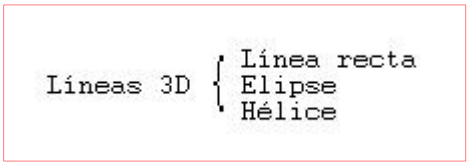


Figura 4. Cuadro sinóptico de las líneas de 3D.

Representación de superficies.

Se realiza mediante una ecuación relacionando las tres coordenadas de los puntos que contiene o a través de ecuaciones paramétricas con dos parámetros[1]:

- Ecuación explicita
- Ecuación implícita
- Ecuación paramétrica

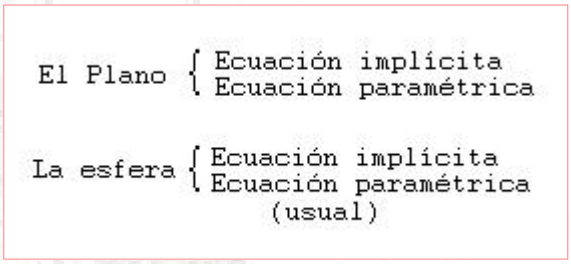


Figura 5. Cuadro sinóptico para las ecuaciones de plano y esfera.

Transformaciones 2D/3D.

Transformaciones afines 2D/3D.

- Al aplicarlas a todos los puntos del polígono (no sólo a los vértices) el resultado es otro polígono. Al aplicarlas a todos los puntos de un poliedro (no sólo a vértices y aristas) el resultado es otro poliedro. Para transformar un polígono o un poliedro basta con transformar sus vértices. En el poliedro se transforman planos en planos.
- Traslación.
- Escalado respecto al origen.
- Escalado respecto a un punto.

1 Síntesis de imágenes por ordenador, E. T. S. I. I. Sevilla, José Cortés Parejo, pag. 5.

- Rotación alrededor del origen. En la práctica sólo se consideran rotaciones alrededor de uno de los ejes cartesianos (Planas) en 3D, así, una coordenada permanece constante y las otras dos se transforman.
- Rotación alrededor de un punto.
- Simetría/Reflexión. En 3D, en la práctica, sólo se consideran simetrías respecto a uno de los planos cartesianos, así, dos coordenadas quedan constantes la tercera cambia de signo.
- Shear (inclinación, distorsión).
- Coordenadas homogéneas. Es la representación de un punto de R^2 mediante un vector y todos sus proporcionales:
  - La representación en coordenadas homogéneas permite representar las coordenadas afines como aplicaciones lineales en el espacio proyectivo (puntos de R^2 junto con los puntos de infinito/direcciones)
  - Puntos y vectores (direcciones) son tratados como una misma entidad geométrica.
  - La aplicación de transformaciones sucesivas a un mismo punto (concatenación) puede realizarse multiplicando las matrices asociadas a cada transformación[2].

Transformaciones no afines 3D.

- Taper (afilar). Las caras de un objeto poliédrico se transforman en general en superficies curvadas.
- Twist (retorcer).
- Bend (doblar).

Transformaciones de encuadre y perspectiva[3].

- Transformación de encuadre 2D. Es la que transforma un rectángulo de R2 en otro, es una afinidad consistente en un escalado seguido de una traslación; su utilidad consiste en aplicar el contenido del primer rectángulo (Ventana/ window) en el segundo (Puerta/ viewport). El rectángulo origen se asocia con el área de trabajo del usuario, el rectángulo imagen se considera que es la zona de dibujo sobre el monitor u otro dispositivo. Esto permite independizar el sistema de referencia del usuario y sus unidades (centímetros, pulgadas, etc.) del sistema de referencia del monitor y sus unidades (pixeles).

2 Idem, pag. 6-8.
3 Idem, pag. 50 - 67

- Zoom y Pan. El zoom consiste en realizar un acercamiento o alejamiento de la imagen en pantalla a partir de una vista dada. El "Pan" o "paneo" (como se conoce comúnmente) consiste en fijar un punto de visión dentro del escenario a partir del cual podemos ver varias vistas de éste.
- Recorte. Es el proceso en el cual determinamos que parte del dibujo original debe incluirse en el viewport.
- Recorte de segmentos. Puede ser respecto a una ventana:
  - Totalmente exterior (invisible)
  - Totalmente interior (visible)
  - Parcialmente exterior – interior.

- Recorte de polígonos. El proceso de recorte debe proporcionar un único polígono cerrado y conexo para que pueda incluir un área. Si cada polígono tiene predeterminadas sus abscisas y ordenadas mínima y máxima la mayoría de ellos pueden descartarse. Mediante el algoritmo de Cohen – Sutherland el polígono se recorta sucesivamente contra cada lado (extendido) de la ventana[4]. Ver imagen:

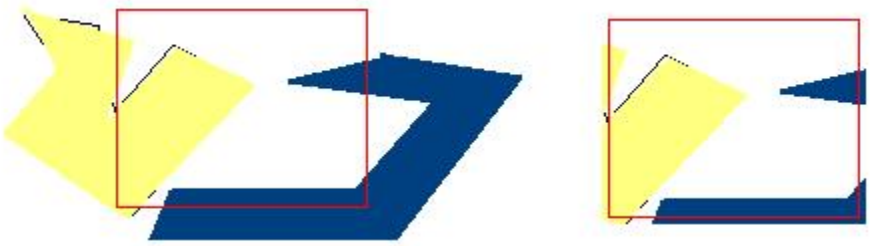


Figura 6. Recorte de polígonos usando el algoritmo de Cohen – Sutherland.

Se debe buscar emplear el menor número de polígonos para que el archivo sea ejecutado lo más rápido posible (ejecución en tiempo real).

Las imágenes se manejan de dos formas en la computadora:

1. Mapa de puntos (bitmaps)
2. Geometría vectorial

La representación por bitmaps se trata de un arreglo de renglones y columnas, donde cada uno de los elementos de la matriz representa un punto en el plano.

4 Idem, pag. 67 – 69.

En la representación por geometría vectorial se guarda la forma precisa de las formas gráficas dentro de un sistema de coordenadas cartesianas (x,y).

Existen tres métodos para la construcción por computadora de un entorno tridimensional:

1. Poligonal
2. Nurms
3. Patch

MODELADO DE SUPERFICIES.

En una amplia gama de aplicaciones se desea modelar una superficie libre en un espacio tridimensional. En este proyecto el objetivo es representar la realidad y cuidar la calidad de imagen, resulta claro que el mundo que nos rodea está lleno de formas caprichosas, de curvas no euclídeas.

Los dos métodos más populares para esculpir superficies son el de Bézier[5] y las splines.

SUPERFICIES DE BÉZIER.

Curvas de Bézier.
La curva se calcula por aproximación a una serie de puntos que ha introducido el usuario. De esta manera se consiguen curvas más suaves que si se calculase por interpolación.

Bézier calcula las coordenadas de un punto de la curva a partir del peso que los puntos de control tienen sobre él. Ese peso viene dado por la distancia al punto de control y una distribución binomial ordinaria. El punto se obtiene sumando esos pesos.

Por lo tanto, dados n+1 puntos de control, P_0, P_1,...P_n por sus coordenadas cartesianas (x_i,y_i)el polinomio de Bézier es

$$P(0) = \sum_{i=0}^{n} P_i B_{i,n}(0)$$

donde

5 Bézier es ingeniero de la Renault y pionero en el mundo del CAD, ya que su método fue el primero que se utilizó para modelar superficies por ordenador.

$$B_{i,n}(\theta)=\binom{n}{i}\theta^{i}\ (1-\theta)^{n-i}$$

Desdoblada en componentes:

$$x(0)=\sum_{i=0}^{n} x_i B_{i,n}(0)$$

$$y(0)=\sum_{i=0}^{n} y_i B_{i,n}(0)$$

El número de vértices n+1 determina el orden del polinomio, n. Para aumentar el nivel de detalle, es necesario aumentar la "libertad de la curva para girar": aumentando el orden del polinomio aproximador.

Propiedades de las curvas de Bézier:

- Una curva de Bézier pasa por el primero y último puntos de control.
- La curva es tangente a los lados del polígono abierto que va de P_0 a P_n en estos puntos.
- Cada punto de control ejerce una perceptible atracción principalmente sobre el tramo de curva próximo a él.
- Las curvas de Bézier pueden ser cerradas y cortarse a sí mismas.
- Una curva de Bézier está siempre en la cavidad convexa que contiene a los puntos de control.
- El control de la curva es global.
- Si se altera la abscisa de un punto de control, la curva sufrirá un estiramiento de abscisas, quedando invariantes las ordenadas. Lo mismo es válido para éstas.
- Aumentando la multiplicidad de un punto de control, la curva tiende a aproximarse más a él.

Modelar superficies de Bézier[6].
Para modelar superficies de Bézier se usan dos conjuntos de curvas de Bézier. Se deben introducir (m+1)(n+1) puntos de control, y la superficie se obtiene como el producto cartesiano de las curvas.

SUPERFICIES SPLINE

En diseño se usaba una lámina de madera flexible que era curvada por el dibujante para obtener contornos. La curva que resultaba era la más suave que se podía

[6] 1974, Pierre Bézier presenta el método bézier.

obtener, ya que la lámina tiende a minimizar su energía de flexión para no romperse. Dicha lámina, en inglés se llama *Spline*.

Curvas B-spline en el plano.

Un enfoque alternativo es construir el polinomio en trozos. Las curvas de bézier están referidas a la base de Bernstein, y cada punto de control está asociado con todos los vectores de la base polinómica. Para que cada punto afecte sólo a un intervalo, sería suficiente con asociarlo a un único vector de la base. Es necesario también añadir condiciones de continuidad en la curvatura de la función, si no la curva obtenida no parecerá suave.

La formulación de una curva B-spline con n+1 punto de control P_0, P_1, ... P_n es:

$$P(0) = \sum_{k=0}^{n} P_k N_{k,t}(0)$$

El grado de las funciones mezcla $N_{k,t}(\theta)$es t-1, que es definida por el usuario o programador; será una concatenación de trozos curvos de grado t-1. Suele emplearse t=4, obteniéndose polinomios de grado 3. Son las difundidas splines cúbicas.

OTRAS APROXIMACIONES

MÉTODO DE HERMITE.

Se basa en interpolación, no en aproximación. Utiliza como funciones de peso los polinomios de la base de Hermite, y se caracteriza porque no sólo interpola los puntos de control, sino también las derivadas, lo que da una cierta facilidad para modelar la continuidad de la superficie.

SUPERFICIES DE COONS

Es uno de los primeros métodos propuestos para modelar superficies. Se basa en definir cuatro curvas que definan la frontera de la superficie a modelar y mezclarlas. No se puede conseguir continuidad de primer orden al juntar dos superficies de Coons, por lo que apenas se utilizan.

SUBDIVISIÓN DE POLIEDROS.

Es una aproximación totalmente distinta para el modelado de superficies de forma libre. Se basa en alterar la posición de los vértices de un poliedro base, añadiendo otros durante el proceso y subdividiendo los lados. Presenta el inconveniente de la dificultad de manejo para el usuario y su más difícil implementación, comparado con los otros métodos.

MODELADO DE SÓLIDOS

El objeto rey es el sólido en el modelado de superficies, aquí un breve resumen de los métodos utilizados.

INSTANCIACION DE PRIMITIVAS.

El modelador de sólidos debe llevar incluida una librería de primitivas sólidas; es decir, un conjunto de objetos que el usuario no tiene que describir al programa.

Estos objetos están parametrizados internamente, de manera que el usuario, a partir de la forma característica del objeto del que se trate, pueda precisar sus dimensiones.

OPERACIONES BOOLEANAS REGULARIZADAS.

Las operaciones booleanas entre sólidos son el más popular de los métodos para combinar objetos más o menos sencillos y obtener otros.

Es un método muy intuitivo, potente y agradable de usar, lo que explica su amplia difusión.

Es apropiado, por razones de consistencia, el empleo de operaciones booleanas regularizadas. Estas son cerradas: es decir, al aplicarse a dos sólidos siempre se obtiene otro sólido.

MODELO DE ESQUEMAS O ALÁMBRICO

La información que tiene el programa sobre el sólido es un conjunto de líneas, típicamente las aristas del objeto. En realidad, es impropio considerar este método

como de modelado de sólidos, pues no hay datos suficientes que permitan tratar al objeto como tal. Se usa en algunos entornos CAD 3D. Ha quedado obsoleto.

MODELADO B-REP

B-rep (*boundary representation*) o representación de fronteras define los objetos en función de la superficie que los encierra: vértices, aristas y caras. Algunos B-reps incluso restringen a que las caras sean triángulos y polígonos convexos. Por lo general, en estos sistemas las superficies curvas y las formas libres se poligonizan, con la consiguiente pérdida de resolución y el aumento de las necesidades de memoria para el almacenamiento de la estructura de datos. Las B-reps tratan primordialmente con poliedros.

Baumgart[7] introdujo los llamados operadores de Euler. Son una herramienta fundamental para las B-reps, pues se aplican a poliedros de Euler y se obtienen poliedros de Euler. Con esto, se pueden ofrecer al usuario, primitivas consistentes y herramientas para modificarlas. Operan añadiendo y eliminando vértices, aristas y caras.

CSG MODELADO C-REP

Este es el método más utilizado en la implementación de modeladores de sólidos. Está orientado a las operaciones booleanas, por lo que hereda toda su potencia, facilidad de uso y consistencia. Permite, además, calcular con relativa sencillez las propiedades físicas de los sólidos que modela, lo que resulta más atractivo para aplicaciones CAD de diseño en las que los objetos que se necesitan no son excesivamente complejos. Maneja de igual manera superficies curvas que poliédricas.

C-rep viene de representación constructiva. Es más conocido como CSG (geometría sólida constructiva) y surge como la estructura de datos más natural que se pueda aplicar al proceso de diseño a partir de operaciones booleanas.

El método se basa en ofrecer primitivas sólidas que el usuario instancia y combina con operadores booleanos. Un modelador CSG almacena la información en forma de árbol binario, donde los nodos son operaciones booleanas y las hojas, las primitivas.

[7] 1974.

Los árboles CSG son fáciles de representar en pantalla, pues las operaciones lógicas no se resuelven entre volúmenes 3D, sino en una dimensión. Al disponer de una información total sobre la masa del objeto, es apropiado para el cálculo de propiedades físicas.

BARRIDOS

Este método consiste en barrer una región del espacio con una superficie plana. El barrido puede ser trasnacional, con dirección perpendicular o no al plano de la superficie, o rotacional. Permite conseguir volúmenes difíciles de generar incluso con operaciones booleanas.

Vossler propone un método basado en reconocimiento de patrones para pasar a CSG. Imponiendo ciertas restricciones a la superficie con la que se efectuará el barrido y presuponiendo la existencia de ciertas primitivas CSG, va tomando tramos de la superficie y buscando la primitiva con la que coincidirá tras el barrido.

DESCOMPOSICIÓN ESPACIAL

En estos métodos se divide el espacio en celdillas cúbicas y se estudia qué objetos están en cada celdilla. Es un proceso que recuerda la rasterización de primitivas de dibujo. Es el conocido efecto de rectas dentadas, por analogía, a las celdillas se las conoce con el nombre de voxel[8] (elementos de volumen).

Como ocurre en pantalla, aumentar la resolución da más exactitud al modelo. Esto se traduce en aumentar el número de vóxeles con los que se describe la escena.

ENUMERACIÓN ESPACIAL

Consiste en partir toda la escena con un número prefijado de vóxeles iguales, formando una red regular. Para representar un objeto, sólo debemos preocuparnos de qué celdas ocupa y apuntarlo.

[8] La palabra proviene de la contracción del término en inglés “volumetric pixel” es la unidad cúbica que compone un objeto tridimensional.

ÁRBOLES OCTALES

Nacen del excesivo gasto de memoria para almacenar una enumeración espacial.

Se basa en un DAC (esquema algorítmico divide y vencerás), y funciona de la siguiente forma. Se considera toda la escena como un solo voxel. Si está totalmente llena o totalmente vacía, se acaba. Si no, se parte en 8 cubos (8 octantes) y se aplica la misma idea recursivamente a cada uno. El proceso terminaría cuando se alcance un cierto umbral (el tamaño del voxel es demasiado pequeño para ser significativo), asignándose en este caso al voxel el estado de ocupado.

REPRESENTACIÓN POR BARRAS

Ideado por Montani y Scopigno[9], también propone una mayor compactación de la enumeración espacial, pero con otro enfoque totalmente distinto. Piénsese que un árbol octal es muy sensible a la situación de los objetos y un pequeño desplazamiento de un objeto puede conllevar un aumento de la recursión de varios niveles.

Lo que se propone es que una hilera de vóxeles consecutivos ocupados por el mismo objeto se considera una barra y un objeto se considera compuesto en barras de distintas longitudes. Identificada una barra, sólo hay que almacenar los dos vóxeles que la delimitan.

Sus mayores inconvenientes son el gasto de memoria y la pérdida de información intrínseca cuando se manejan superficies curvas.

SUPERFICIES ALGEBRAICAS

Hay una serie de superficies curvas muy útiles como primitivas y que admiten una ecuación matemática sencilla. En el caso de esferas, elipsoides, cilindros, conos, paraboloides y tiroides. La forma más natural de tratarlos es a partir de sus propias ecuaciones, además de ser la solución óptima desde el punto de vista del consumo de memoria. Sin embargo, en ciertos modeladores, como B-reps restringidos, se hará precisa una poligonalización de estos objetos (descomposición espacial).

[9] En 1997 Cigoni, Montani y Scopigno presentaron el algoritmo *DeWall*, que hace posible el cálculo de una triangulación de Delaunay usando *divide and conquer* para cualquier número de dimensiones.

METABALLS: MODELADO DE SÓLIDOS CON CAMPOS ESCALARES.

Los métodos antes vistos para el modelado de sólidos resultan complicados para el modelado de cuerpos humanos o de otros animales.

EL MODELO

Fue desarrollado simultánea e independientemente por K. Omura y J. Blinn[10]. Éste último lo utilizó para modelar mapas de densidades de electrones de estructuras moleculares. La idea es que un objeto 3D se modela como una isosuperficie de un campo escalar que es generado por una serie de primitivas consideradas puntuales.

El valor del campo creado por una primitiva P_i en un punto del espacio vendrá dado por una función de R^3 a R de dicho punto:

$$V_i(x, y, z) = f(x, y, z)$$

Si asumimos por un momento que la función *f* creadora del campo depende sólo de la distancia del punto (x, y, z) a la primitiva y que sólo tenemos una primitiva, las isosuperficies serán esferas. En la siguiente figura se muestran dos primitivas generadoras de un campo como el descrito, lo suficientemente alejadas como para que una no interfiera en el campo de la otra.

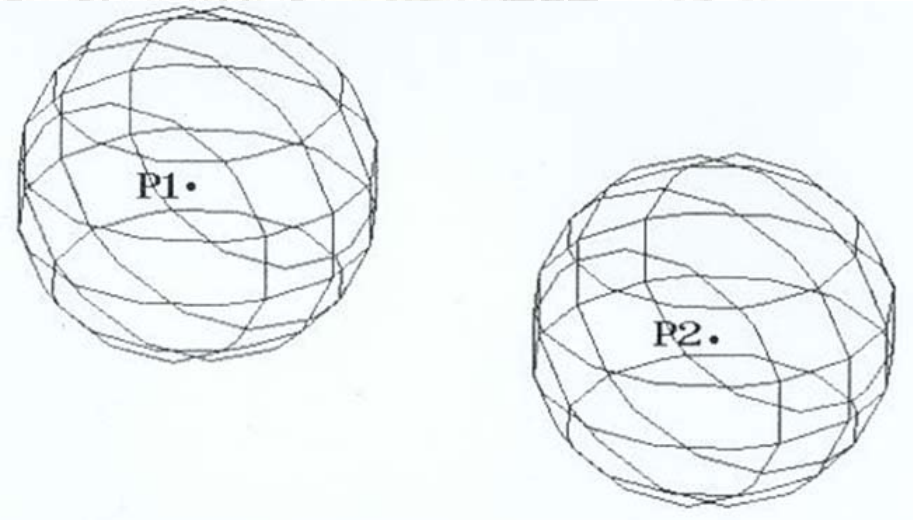


Figura 7. Metaballs. Inicio.

[10] 1983. Kouichi Omura, James Blinn

Si se usan varias primitivas, entonces se suma el campo aportado por cada una de ellas, y las isosuperficies pueden adquirir formas complicadas:

$$V(x,y,z) = \sum_{i=1}^{N} V_i(x,y,z)$$

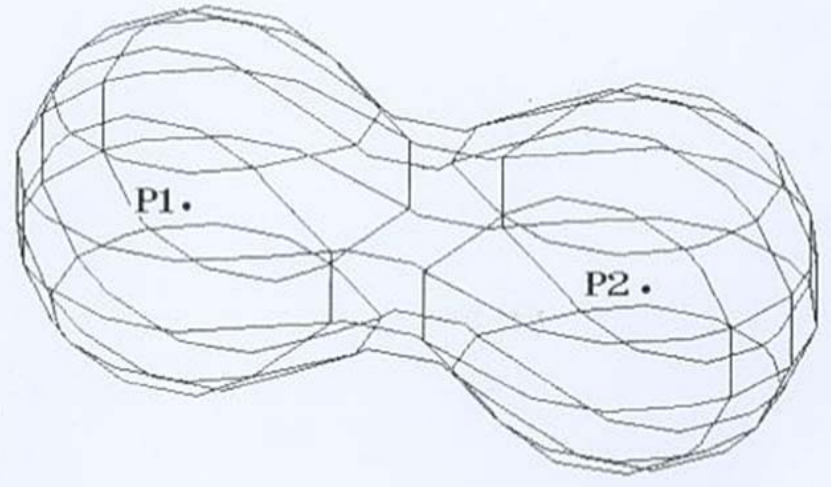


Figura 8. Metaballs. Segunda parte.

La isosuperficie de potencial T (>0) se expresaría como una función implícita:

$$V\ (x,\ y,\ z) = T$$

El efecto de aproximar dos primitivas: la suma de sus respectivos campos en el espacio entre ellas hace que aparezca superficie donde en un método clásico no lo habría, y se produce una deformación por la atracción de una esfera sobre la otra, obteniéndose una forma que recuerda un cacahuate.

Una importante desventaja del método es la dificultad de asignar texturas a las primitivas.

BLOBBY MODEL

Hay que tener en mente que Blinn pretendía modelar orbitales moleculares, así que su función de campo fue la de la densidad de un átomo de hidrógeno:

$$V_i(x,y,z) = b_i \exp(-a_i g_i(x,y,z))$$

El factor a_j (>0) afecta a la decaída exponencial del campo, y el b_j a su intensidad. Al aumentar b_j, se consiguen primitivas más grandes.

3.2 Metodología.

Para un proyecto interactivo se recomienda que los objetos sean construidos por conjuntos de polígonos (caras) de las cuales identificamos aristas y vértices. Con el fin de minimizar el uso de memoria RAM al momento de ejecutar la simulación en tiempo real.

Es importante utilizar el mínimo de aristas y vértices. Siempre que no afecten la visualización del proyecto terminado.

Los objetos se deben modelar con el mínimo de polígonos posible. Aunque esto reduce el detalle en los objetos, si podemos obtener mayor rapidez de ejecución en tiempo real.

Los polígonos utilizados en el modelado deben preferentemente ser de tipo cuadrangular o triangular pues es mejor la forma para texturizar.

Se debe cuidar que los objetos tridimensionales no contengan polígonos con caras internas (confirmando que la Normal del polígono nos está proyectando la cara visible), polígonos internos (que no se vean, como la cara de un objeto que da al suelo) pues nos resultarán en un gasto no necesario de memoria al momento de su ejecución en TR (Tiempo Real).

3.3 Desarrollo.

El diseño de los objetos y del entorno tridimensional puede ser realizado en varios programas como 3DStudio Max, CAD, MAYA, Vue 6 Xtream, etc. La decisión de cuál usar depende del desarrollador del proyecto y de varios factores que por su complejidad no se describirán en el presente documento.

En este caso, se realizó la arquitectura de templos y pirámides usando 3DStudio Max 9. Hay que tomar en cuenta que no sólo se pueden crear escenarios conocidos, también podemos recrear un ambiente celular o espacial. En la imagen presentada a continuación se puede observar que la utilización de polígonos se reduce al mínimo.

Se observan líneas delimitando cada uno de los elementos arquitectónicos; mismas que se utilizarán dentro del proyecto como el sistema de colisión mediante el cual definiremos las estructuras como sólidas para no utilizar un exceso de recursos del sistema para ello.

Con este sistema, evitaremos que los objetos en movimiento atraviesen estructuras que consideramos sólidas. O nos ayudarán a definir algún procedimiento especial dentro del sistema como sería subir escaleras por ejemplo.

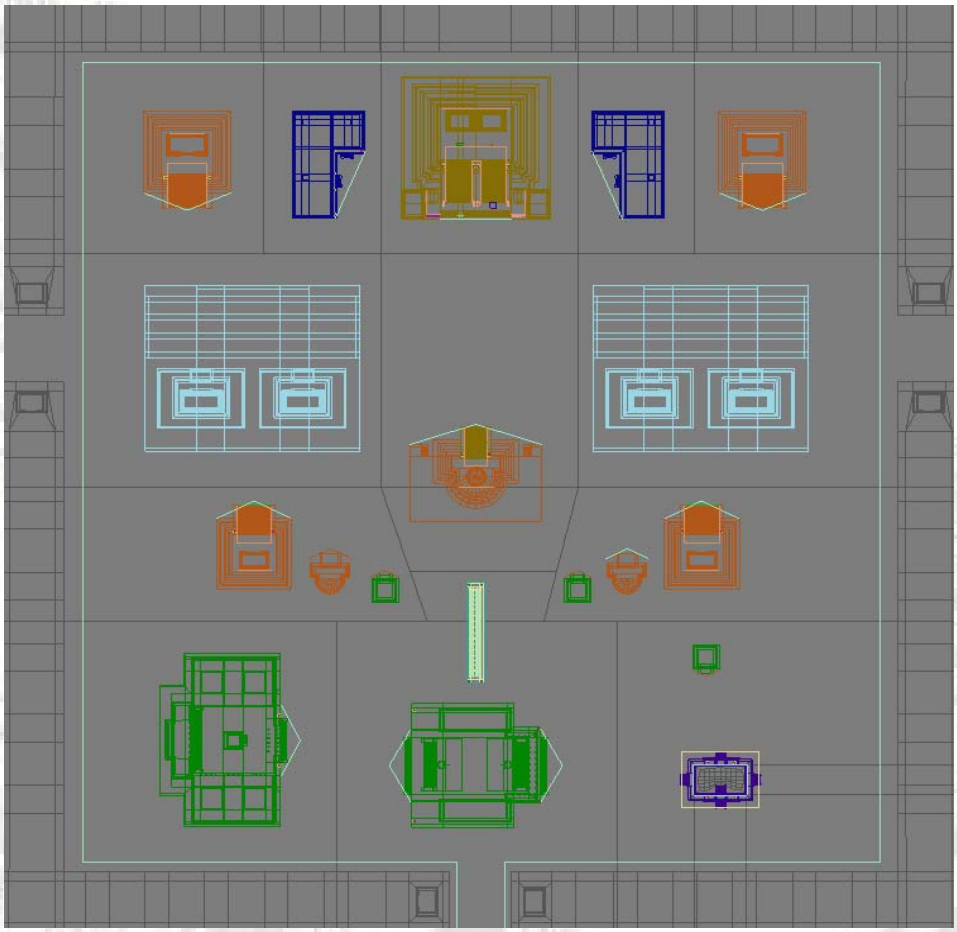


Figura 9. La representación del escenario 3D.

Con esta arquitectura, investigada previamente, tenemos definido el escenario sobre el cual realizaremos la ambientación, iluminación, rendering, etc. Debemos revisar con cuidado la arquitectura hecha para evitar errores posteriores, como el número de polígonos, la normal hacia la cara visible, etc.

Revisar que la arquitectura del proyecto esté perfecta según los lineamientos que deseamos en nuestro proyecto, puede resultar monótono y aburrido, sin embargo,

nos ayudará a evitar trabajo repetitivo cuando el proyecto tenga un avance considerable por errores dentro de la arquitectura. Y el término del proyecto podrá ser más rápido sin estas correcciones innecesarias.

Los modelos realizados en 3D Max Studio se exportan dentro de un archivo a Virtools para poderlos utilizar. Esto se hace mediante un programa (archivo complementario, Plugin) que identifica y revalida los formatos utilizados. En este caso <Virtools Export Plugins>.

Para este efecto, recomendamos utilizar la siguiente dirección:
C:/virtools/virtools proyectos/virtools export/virtools export plugins

Una vez que el escenario se encuentra dentro del sistema de virtools, continuamos con el siguiente paso en el desarrollo de nuestro proyecto interactivo 3D que es la textura del escenario. ¿Cómo debe lucir? Se usará la información recabada para que el proyecto cumpla con nuestras expectativas.

Para este efecto, utilizaremos las herramientas de textura y luz que se describen a continuación. Todo archivo creado para textura o luz, que se use para el proyecto, se recomienda sea archivada en la siguiente dirección:
…/virtools/samples/nombre-proyecto/texturas /

4. Texturas.

4.1 Sustento teórico.

Las texturas y los mapas de luz-sombra (conocidos como *lightmap*, los veremos en el siguiente capítulo) nos ayudan a dar una ilusión de realidad al proyecto, sin embargo cuando queremos que éste se utilice en tiempo real debemos cuidar que no se sobrecargue el proyecto con estos detalles, pues resulta en un proyecto que será lento durante la ejecución en Tiempo Real.

La textura es una cualidad abstracta que nos ayuda a distinguir y reconocer objetos (rugoso o liso por ejemplo). Los archivos de textura se crean con diferentes herramientas por lo que pueden ser archivos bmp, jpg, gif, dib, etc. Usualmente se utilizaremos los jpg, jpeg, pues resultan en menor tamaño de bits que los otros.

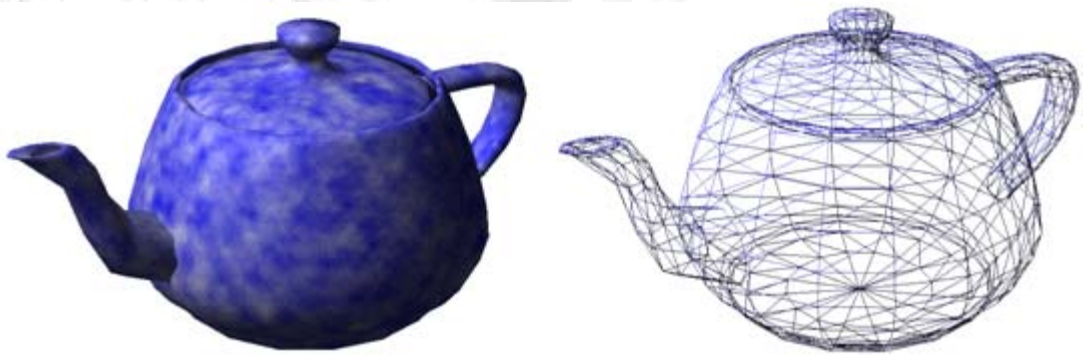


Figura 10. Ejemplo de texturas en bitmaps.

Por texturización entendemos en el proceso de asignar una imagen (bitmap) a un polígono, de manera que en lugar de ver este de un color plano, o un gradiente de colores, veremos la imagen proyectada en él[1].

Actualmente se dispone de funciones específicas para el mapeado de texturas (pegar imágenes en la superficie de las primitivas dibujadas), añadiendo realismo a la escena. En este trabajo se explica el proceso de mapeado de texturas a través de un sencillo ejemplo, para posteriormente emplear las texturas en la aplicación.

[1] Información de monografías.com

Parámetros de las texturas

A cada textura le podemos asignar unas ciertas características. El primero de ellos es el filtro de visualización. Una textura es un bitmap, formado por un conjunto de pixeles, dispuestos de manera regular. Si la textura es de 64 x 64 pixeles, y la mostramos completa en una ventana de resolución 1024 x 768, se escalarán estos pixeles, de manera que cada pixel de la textura (de ahora en adelante texel) ocupará 16 x 12 píxeles en la pantalla.

1024 pixels ancho / 64 texeles ancho = 16;
768 pixels alto / 64 texeles alto = 12;

Eso quiere decir que lo que veremos serán "cuadrados" de 16 x 12, representando cada uno un texel de la textura. Visualmente queda muy poco realista ver una textura 'pixelizada', de manera que le aplicamos filtros para evitarlo. El más común es el 'filtro lineal', que se basa en hacer una interpolación en cada píxel en pantalla que dibuja. A pesar de que pueda parecer costoso a nivel computacional, esto se hace por hardware y no afecta en absoluto al rendimiento.

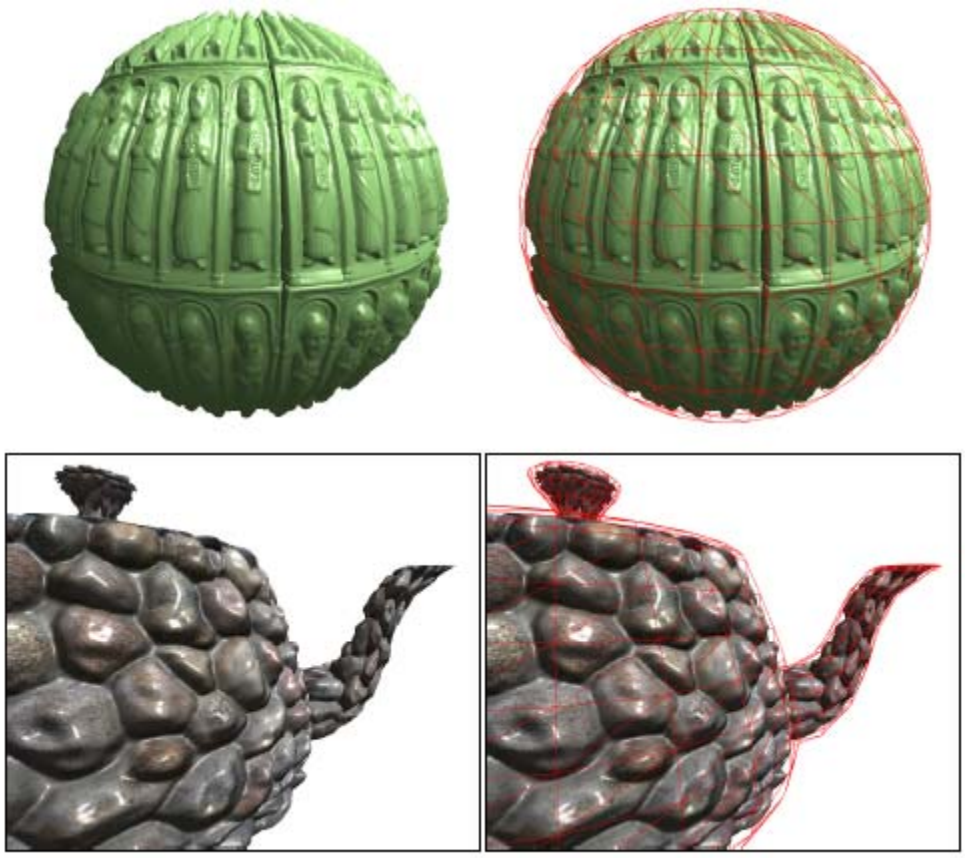


Figura 11. Ejemplo de texturas irregulares en bitmaps.

Normalmente en una escena suelen haber muchas texturas diferentes. Si dibujamos toda la geometría desordenadamente, es posible que cada pocos triángulos tengamos que cambiar de textura. El proceso de cambio de textura tiene un cierto coste, por eso es recomendable organizar nuestra geometría, haciendo grupos de superficies que tengan la misma textura, de manera que hagamos los mínimos cambios posibles. Asimismo, podemos hacer lo mismo con los materiales en el caso que hagamos uso de ellos.

MODELOS DE COLOR.

Un modelo de color es un método para explicar las propiedades o el comportamiento del color en algún contexto particular.

Luz es una banda de frecuencia estrecha en el espectro electromagnético. Otras bandas de frecuencia son ondas de radio, microondas, ondas infrarrojas y rayos x.

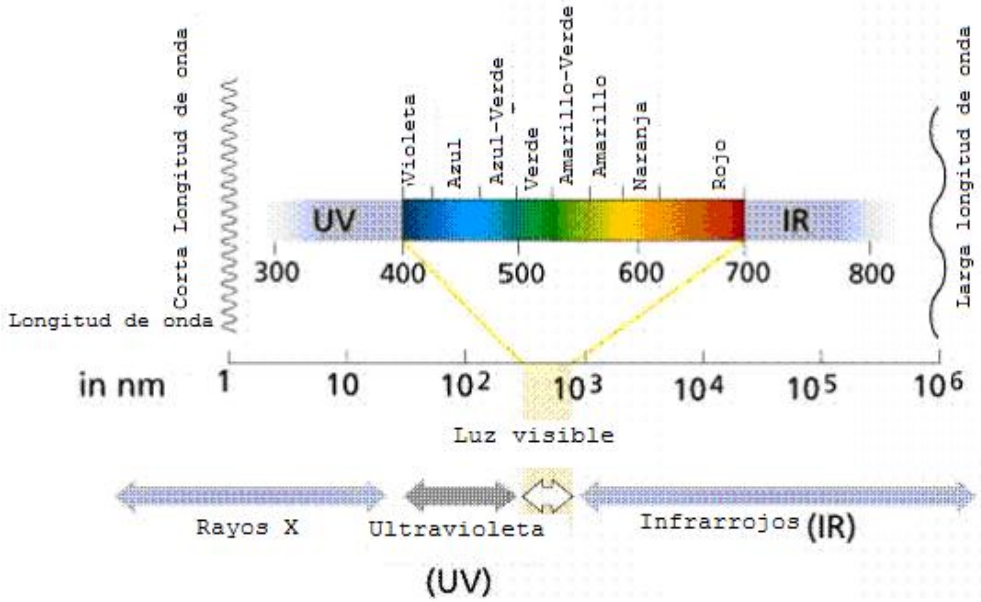


Figura 12. Esquema de luz. Frecuencia de onda.

La longitud de onda y frecuencia de la onda monocromática son inversamente proporcionales, con la constante de proporción como la velocidad de la luz c:

$C = \lambda f$ donde: longitud de onda=λ; frecuencia=f_0

La CIE[2] define los siguientes modelos de color[3]:

[2] Commission International de l'Eclairage.
[3] Hearn – Baker, Gráficos por computadora, pags. 115 – 125.

1. Modelo de color XYZ.

Conjunto de primarios X, Y, Z que representan vectores en un espacio tridimensional de color aditivo. Cualquier color Cλ se expresa como:

Cλ xX + yY + zZ, donde x=azul, y=rojo, z=amarillo.

2. Modelo RGB. Este es el que utilizamos en el proyecto.

Basado en la teoría de los tres estímulos de la visión se obtiene con un rango de sensibilidad pico en longitudes de onda (nm):

Rojo (630 nm)

Verde (530 nm)

Azul (450 nm)

3. Modelo YIQ

Es el modelo utilizado por National Television System Commitee (NTSC) para formar la señal de video compuesta YIQ que se basa en los conceptos del modelo XYZ de la CIE.

Y = xX + yY + zZ {modelo XYZ

I = naranja – cian {amplitud de banda 1.5 MHz

Q = verde – magenta {amplitud de banda 0.6 MHz

4. Modelo CMY

Modelo de color que define como colores primarios CIAN, MAGENTA y AMARILLO; sirve para describir la salida de colores a dispositivos de salida impresa.

5. Modelo de color HSV.

Utiliza descripciones de color:

matiz (H) + saturación (s) + valor (v)

6. Modelo HLS

Utiliza la misma ideología del HSV, con las variantes:

Matiz (H) + brillantez (L) + saturación (s)

4.2 Metodología.

Las texturas y los mapas de luz/sombra (*lightmap*) son el siguiente paso en el desarrollo de una aplicación tridimensional y realidad virtual. Debemos cuidar que las texturas puedan ser reutilizadas (por un objeto grande o por varios objetos), que den una apariencia real a los objetos y que se puedan modificar fácilmente.

Las texturas generadas para los edificios del escenario virtual, se almacenan en una carpeta especialmente destinada para ello. Así se evita que el sistema quede demasiado grande y se vuelva lento al ejecutarlo en tiempo real.

Para optimizar la ejecución del proyecto se opta por reducir el tamaño de los archivos utilizados de textura o mapa de luz/sombra, si éste es muy grande. Y se hace repetición del mismo archivo a lo largo del espacio que se desea ambientar, muy recomendable técnica para pisos o grandes espacios a llenar.

4.3 Desarrollo.

Es importante cuidar la optimización de los recursos, para reducir el tamaño del archivo ya sea mapa de luz/sombra o textura, podemos utilizar dentro de virtools la herramienta de Edit parameter y manualmente reducir el número de bits para que sea el menor posible sin que afecte nuestro proyecto. Para este efecto, se hace lo siguiente: dentro del menú de texturas <colocarse sobre la imagen botón derecho del Mouse> seleccionar el tamaño del "SLOT". Ahí mismo, se reduce el formato de video al mínimo (16 bits RGB 555*). Como se muestra a continuación:

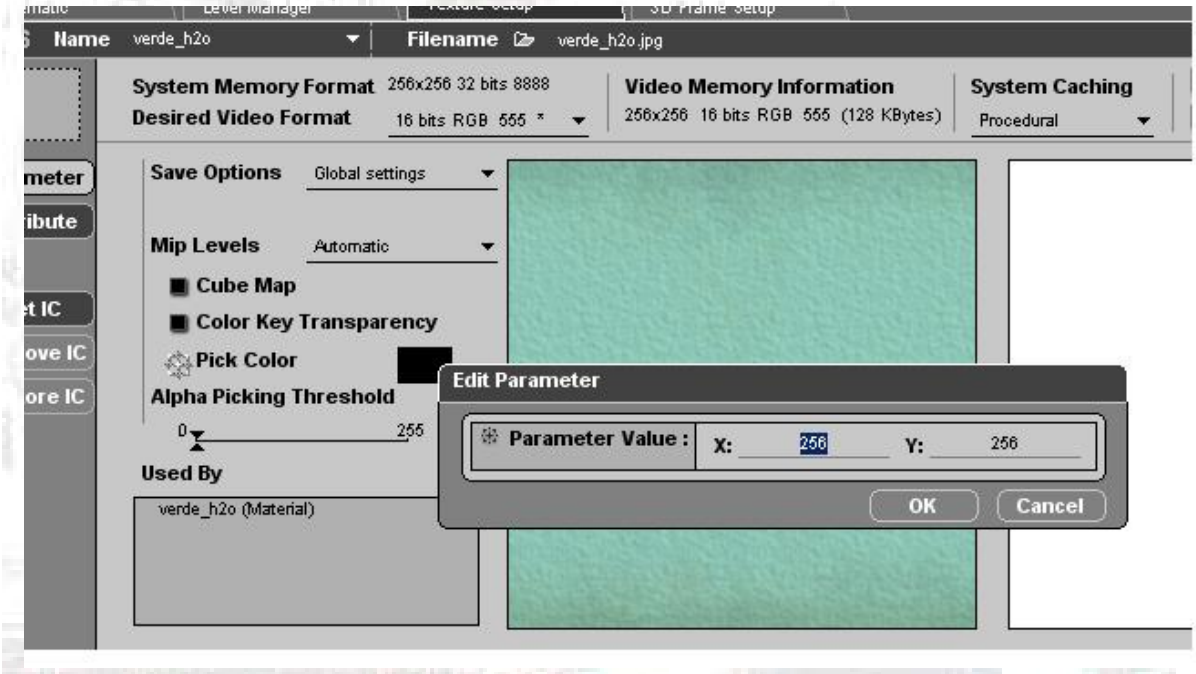


Figura 13. Menu Texture Setup, dentro de Virtools para manipular texturas o lightmaps.

Estas texturas se pueden mejorar o modificar independientemente de la arquitectura a la que pertenezcan; así una textura puede utilizarse para uno o más elementos dentro del ambiente de 3D. Así reducimos el uso excesivo de imágenes similares por varias repeticiones.

También debemos tomar en cuenta que cada vez que se importe el proyecto por alguna modificación hecha se debe confirmar que los archivos de texturas no se repitan dentro del menú general pues sobrecargan el proyecto para su óptimo desempeño en tiempo real.

Este es un archivo de textura con transparencia en formato png, utilizado para "ahorrar" polígonos y sustituir a la geometría por una textura de poco peso:

Figura 14. Imagen png, utilizada dentro del entorno 3D, para dar una apariencia real al objeto.

Este es un archivo utilizado para la simulación de luz y sombra de un objeto (*lightmap*). Imagen en archivo de extensión jpg, generada dentro del proyecto a partir de las condiciones de luz y cámara establecidas en el mismo. Recrea la apariencia de luz-sombra a cada objeto 3D de el escenario tridimensional.

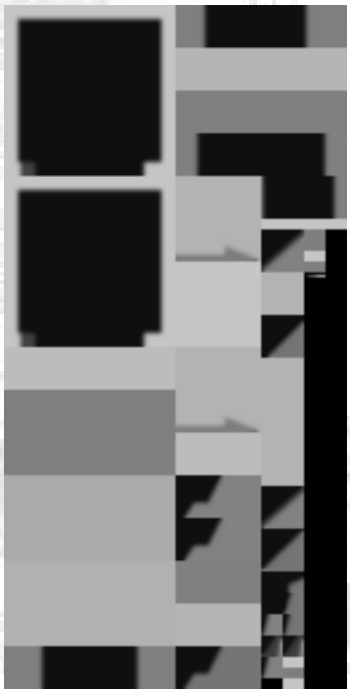


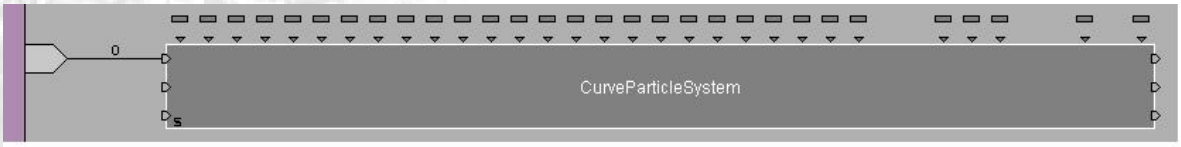
Figura 15. Imagen jpg, lightmap creado en virtools.

Ambos archivos pueden ser reutilizables a conveniencia del proyecto, pues solo se necesita hacer referencia a ellos dentro del proyecto en virtools.

La simulación de eventos físicos como lo son los sistemas de partículas (humo, agua) son truqueados a partir del uso de texturas asignadas a secciones de planos, ya que con el uso de las texturas se puede procesar de una manera mas rápida y eficiente la generación de partículas planas con apariencia de la textura o material que se le asigne.

Para darle mayor realismo al proyecto, se hace uso de efectos tales como humo, etc. En este caso se muestra el BB que dará el efecto de los braseros humeando, al mismo tiempo que lo usamos como un indicador de la ruta a seguir por el caballero águila durante el juego. Ver la ilustración siguiente:

Figura 16. Building Block. Estructura de programación de virtools para generación de partículas.



Con este BB[4], el proyecto tiene más realismo y además se activa según la ruta y la zona que debe visitar el caballero águila. Se aprecia en la toma siguiente:

4 Building Block, Bloque de estructura de Virtools, contiene la información de programación para ejecutar una acción por, para, con el objeto u objetos asignados a él.

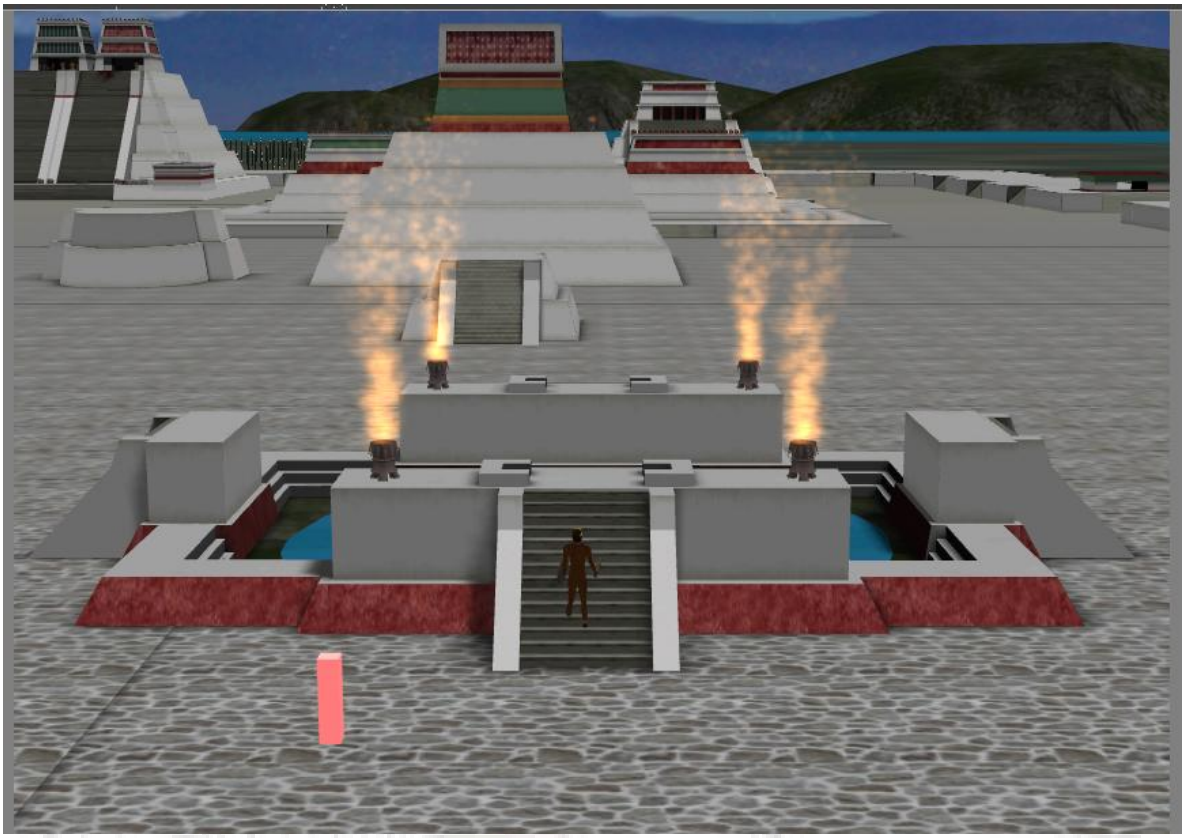


Figura 17. Ejemplo de generación de partículas: Humo.

5. Iluminación y ambientación del escenario.

5.1 Sustento teórico.

El siguiente paso importante en la realización de un proyecto tridimensional en tiempo real es la iluminación y ambientación del entorno. Los *lightmap* son las imágenes que nos ayudan a dar profundidad a la imagen 3D con la ilusión del brillo de luz y la proyección de sombras sobre los objetos o el entorno, en otras palabras, son los archivos con la información de luz-sombra de los objetos que utilizaremos en el proyecto, para dar una ilusión lo más real posible: sin doble sombra, muy oscuros o en extremo claros y de preferencia en formato jpg, por mencionar algunos aspectos.

Factores de iluminación[1]*:*

- Percibimos imágenes formadas por luz
- La luz que se ve de un punto depende de:
  - la luz que llega de la fuente de iluminación
  - la orientación del punto
  - las características del material
  - la luz que llega de otros puntos
  - la posición del observador

Reflexión de la luz*:*

- La luz se puede reflejar:
  - perfectamente especular
  - imperfectamente especular
  - difusa

Modelo de iluminación*:*

- Luz reflejada por una superficie:

luz ambiente + reflexión difusa + reflexión especular

$$I = k_a I_a + k_d I_d + k_s I_s$$

[1] Información obtenida de http://www.sc.ehu.es/ccwgamoa/clases

Luz ambiente:

- k_aI_a
- I_a es la luz ambiente en la escena
- k_a es una propiedad del material
- Es una simplificación del modelo global de iluminación
- Substituye a la contribución de la luz que no llega directamente de las fuentes de luz

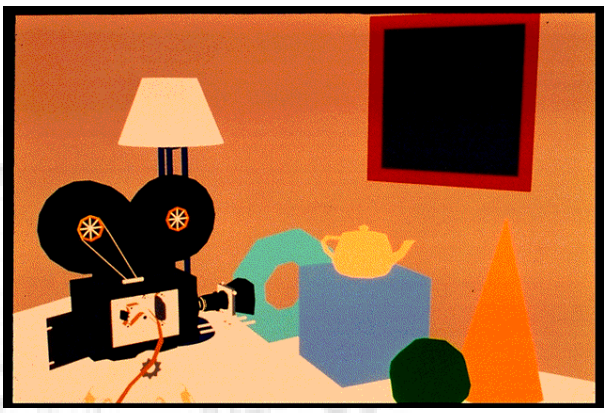


Figura 18. Imagen representativa de luz ambiente

Luz con reflexión difusa:

- Ley de Lambert
- $k_dI_d = k_d I_p \cos\theta = k_d I_p (n \bullet l)$
- I_p es la luz de una fuente puntual
- k_d es una propiedad del material
- Es independiente del observador y la componente principal del color del objeto

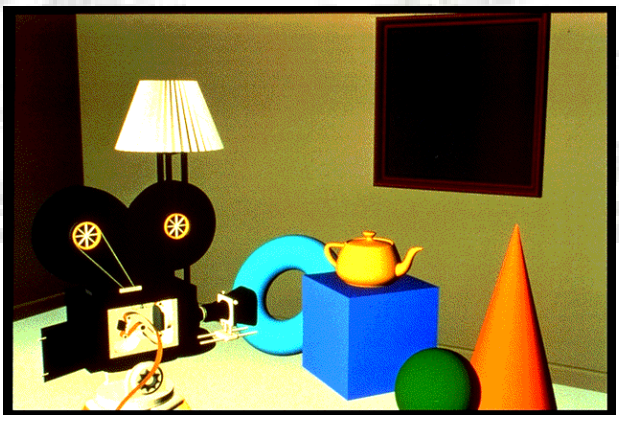


Figura 19. Imagen representativa de luz con reflexión difusa

Luz con reflexión especular:

- La reflexión perfectamente especular solo produce incidencia en un punto de visión
- Reflexión imperfectamente especular
- La intensidad disminuye al alejarnos de la dirección del rayo reflejado
- Ley de Fresnel[2]: $\cos^s \alpha$
- Produce los brillos de los objetos
- $k_s I_s = k_s I_p \cos^s \alpha = k_s I_p (r \bullet v)^s$
- I_p es la luz de una fuente puntual
- k_s es una propiedad del material
- s determina la dispersión
- Cálculo del vector reflejado r

$r = r_1 + r_2$ [1]

$- r_1 = l + (-r_2) \Rightarrow r_1 = - l + r_2$ [2]

Substituyendo en [1]:

$r = - l + r_2 + r_2 = 2r_2 - l$ [3]

$r_2 = (n \bullet l)\, n$ [4]

Substituyendo en [4]:

$r = 2 (n \bullet l)\, n - l$ [5]

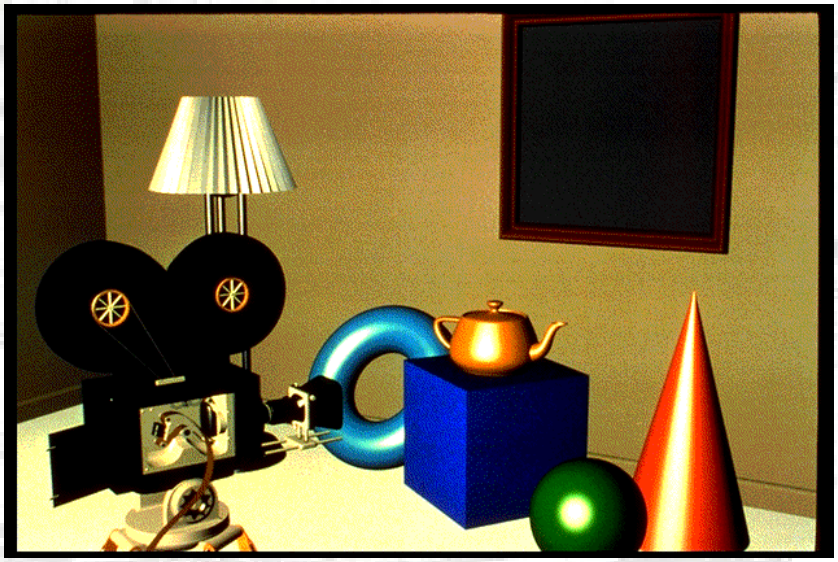


Figura 20. Imagen representativa de luz con reflexión especular.

Sustituyendo, tenemos ahora el modelo de iluminación:

- Luz reflejada por una superficie:

[2] Augustin.-Jean Fresnel fue un físico francés que contribuyó significativamente a la teoría de óptica ondulatoria. Fresnel estudió el comportamiento de la luz tanto teórica como experimentalmente.

luz ambiente + reflexión difusa + reflexión especular

$$I = k_a I_a + k_d I_p (n \bullet l) + k_s I_p (r \bullet v)^s$$

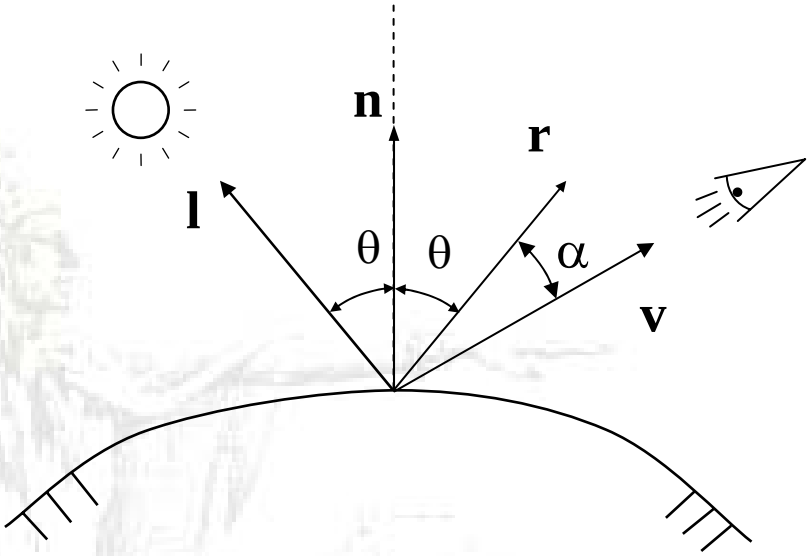


Figura 21. Modelo de iluminación.

5.2 Metodología.

Se debe plantear un sistema de iluminación que recree la ambientación que se desea manifestar. Cuidar que la creación de luz y sombras utilice el mínimo de recursos.

Si se desea una ambientación cercana a la realidad, se deben cuidar detalles como la doble sombra (generada por la utilización de 2 luces opuestas), la textura de los materiales (se distorsionan cuando reciben un exceso o un mínimo de luz), el tamaño de la sombra generada (utiliza muchos recursos de maquina), la intensidad de la luz (no siempre el blanco_radiante resulta), etc.

Dentro de virtools tenemos 3 tipos de luz: direccional (utilizan dirección vectorial), ambiental (tipo point) y spot (que ilumina directamente a un objeto determinado).

Generalmente, en todo proyecto se utiliza una luz ambiental y una luz direccional (en la dirección que se desee). A partir de éstas, se podrán hacer las variaciones que sean necesarias para que el proyecto adquiera el ambiente de luz que se desea obtener.

5.3 Desarrollo.

Después de una serie de intentos se determino que el mejor sistema de iluminación para el proyecto Tenochtitlan consiste en la utilización de 6 luces de tipo direccional. De estas, 4 se disponen en cada posición cardinal, apuntando hacia el centro del escenario. La quinta se dispone arriba sobre la arquitectura con inclinación perpendicular al plano xz. La sexta se orienta en un ángulo de 45° con respecto al plano xz y tiene la misma posición que la anterior, esta luz es la que producirá las sombras de la arquitectura, para una simulación más realista. Ver imagen siguiente:

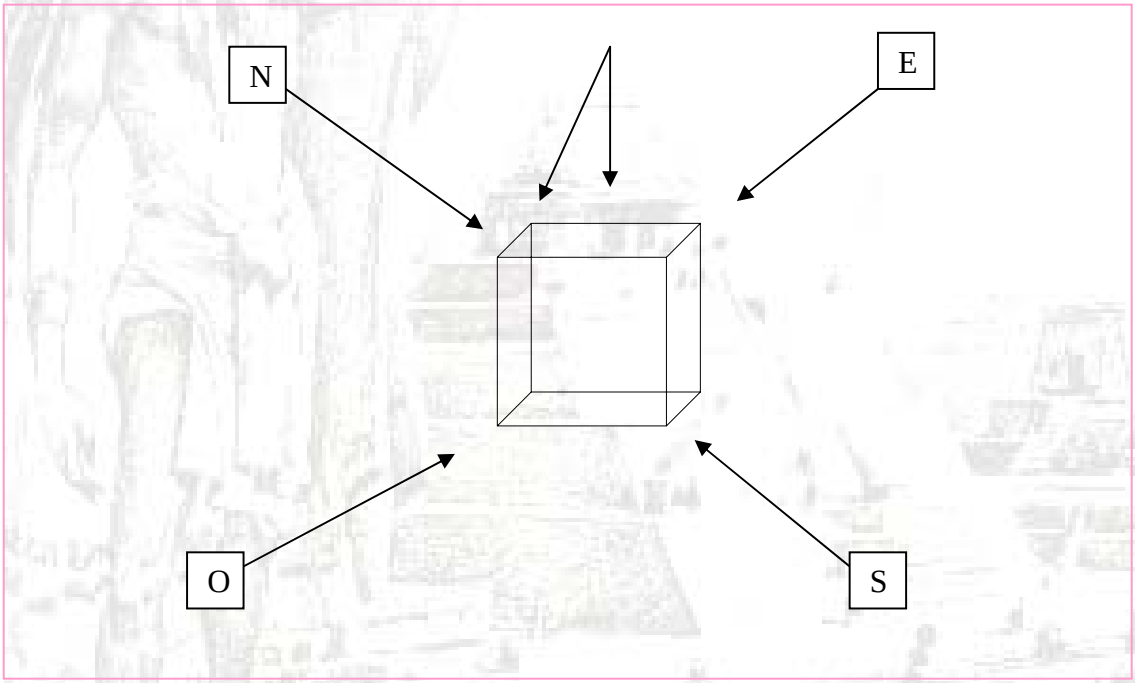


Figura 22. Representación de la ubicación de las luces empleadas para el proyecto Tenochtitlan 1.0.

Se utilizó para este efecto el BB llamado *Static Lightmap*, el cual se programa en el *Level Script*. Este bloque genera mapeos de luz, que sobrepuestos sobre la arquitectura recrean luz_y_sombra. Para su utilización se deben especificar 3 parámetros que se componen de grupos arbitrarios de objetos 3D de nuestra arquitectura (esta agrupación se hace previa a su asignación).

El primer parámetro Receiving Objects es para los objetos que reciben la luz directa y las sombras de otros objetos.

El segundo parámetro Occluder Objects es para ese grupo que hace sombras sobre el primero.

El tercer parámetro es para el grupo de luces que deseamos que iluminen nuestra arquitectura.

Cuando se genera un "mapeo de luz" al mismo tiempo se generan 3 entidades ligadas al mismo. El objeto generado *LM_nombre.del.objeto*, trae consigo su Material *LM_nombre.textura.del.objeto*, su Malla *LM_nombre.malla.del.objeto* y su Textura *LightMap_nombre.del.objeto*.

Es importante indicar que cuando se trabaja en crear los lightmap del proyecto, durante el tiempo de pruebas, se debe cuidar no sobrecargar la base de datos (BD), pues cada vez que se genere una prueba, esta creara los 4 elementos indicados anteriormente, así que se deben borrar los generados anteriormente antes a fin de evitar que se desborde la BD.

El BB llamado LightMap no se utilizo para este proyecto, pues la arquitectura es grande y podría crear problemas con la utilización de recursos, pues este bloque genera las imágenes de luz y sombra en tiempo real lo cual podría afectar el uso de los recursos para los objetos con animación.

Los objetos elaborados en 3D Max Studio se pueden exportar para su utilización en Virtools mediante un archivo complementario <Plugin> el cual indentifica y revalida los formatos utilizados. El programa para esto es Virtools Export Plugins.

6. Cámaras.

6.1 Sustento teórico.

El funcionamiento de la visión estereoscópica es básicamente el siguiente: Se obtienen dos imágenes de una misma escena, desde dos puntos de vista diferentes (normalmente ligeramente diferentes). Una vez establecida la correspondencia entre puntos de las dos imágenes que corresponden a un mismo punto de la escena, mediante una sencilla triangulación puede hallarse la distancia de ese punto a las cámaras. Esto, que dicho así parece muy sencillo, plantea los siguientes grandes problemas:

1- Definición del modelo de las cámaras y calibración de las mismas. Una vez definido el modelo de cámara, se trata de hallar los parámetros que relacionen las coordenadas de la imagen en pixeles con las correspondientes a algún sistema de referencia del mundo real, en unidades métricas.

2- Obtención de los puntos correspondientes en ambas imágenes (matching). Es el mayor problema de la estereovisión, a causa de su ambigüedad. En principio, un punto de una imagen puede corresponder a cualquier punto de la otra. Se utiliza una serie de restricciones geométricas y físicas para tratar de reducir la ambigüedad de la correspondencia.

3- Reconstrucción 3-D. Una vez realizado el matching, el cálculo de la profundidad es inmediato, y sólo plantea algún problema de implementación práctica, ya que se obtiene mediante un sistema de ecuaciones sobredeterminado. El inconveniente puede ser que el número de puntos para el que se halló correspondencia sea escaso, lo que obliga a realizar una interpolación para la obtención de un mapa de profundidades denso.

Consideremos un sistema ideal, con dos cámaras idénticas y perfectamente alineadas (el mismo plano XY). La figura muestra la planta de un sistema esquematizado de estereovisión. En estas condiciones se demuestra que la coordenada Z se puede calcular fácilmente mediante la expresión,

$$Z = \lambda - [(\lambda \cdot B) / (x2 - x1)]$$

Donde: λ es la distancia focal de las lentes de las cámaras, B es la distancia entre las cámaras y x1 y x2 son las coordenadas del objeto en cada cámara. También se puede calcular las coordenadas XY del objeto mediante expresiones igualmente sencillas. El modelo de cámara utilizado es el pin-hole, con un centro óptico o foco

situado a una distancia λ del plano imagen. Es un modelo sencillo que se adapta bien a las características ópticas y geométricas de la mayor parte de las cámaras modernas. La diferencia de coordenadas x2-x1 entre las proyecciones de un mismo punto en las dos imágenes recibe el nombre de disparidad.

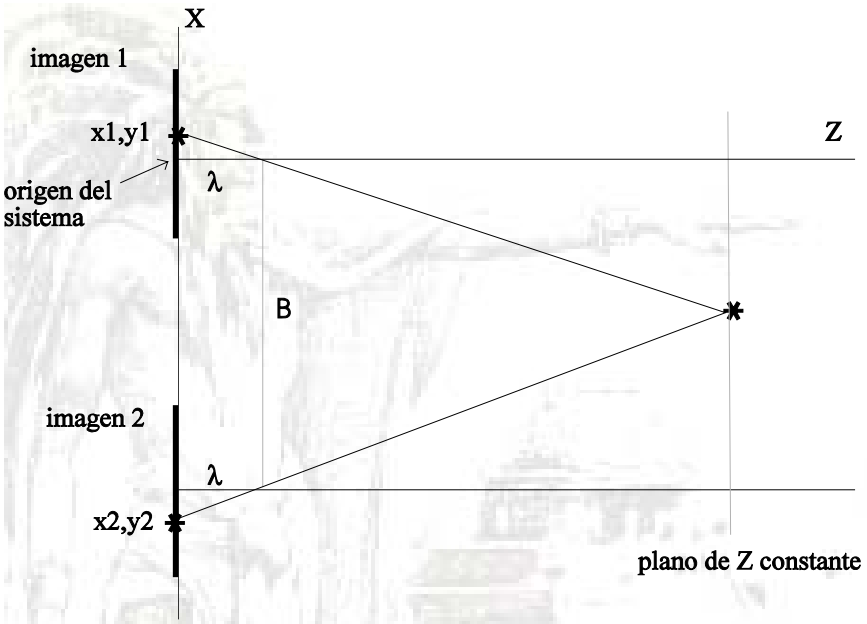


Figura 23. Esquema de un sistema idealizado de estereovisión.

En la práctica nunca se presenta esta situación ideal. Muchas veces interesa que las cámaras formen un cierto ángulo, para definir un campo común más idóneo para la aplicación. Incluso en el caso de que no fuera así, y que mediante un cuidadoso montaje mecánico se consiguiera un alineamiento perfecto de las cámaras, no hay garantías de que los planos imagen (es decir los detectores) y las ópticas correspondientes estén asimismo alineados. Esto supone que las relaciones entre las coordenadas X-Y-Z de un punto de la escena y las coordenadas x-y de su proyección no son las teóricas vistas con anterioridad, sino que deben obtenerse mediante un procedimiento de calibración. En alguno de estos métodos de calibración (método de la matriz de perspectiva) la relación entre coordenadas en el mundo y en la imagen no es función explícita de los parámetros del sistema (λ, B), sino que viene dada por una matriz de calibración que incluye todos los aspectos del mismo (modelo óptico, factores de escala, rotaciones y traslaciones de los planos imagen). La calibración se lleva a cabo tomando imágenes de un cierto número de puntos cuyas coordenadas espaciales son conocidas. El número de puntos suele ser elevado, y se minimiza el error mediante mínimos cuadrados o alguna técnica similar.

El segundo problema que se mencionó anteriormente es la correspondencia o *matching*. Se trata de encontrar cuáles son los dos puntos correspondientes en las dos imágenes diferentes del mismo escenario. El problema es a priori ambiguo. Cualquier punto en el segundo plano imagen podría ser el correspondiente de cualquier punto en el primero. Para atacar esta ambigüedad se usan ciertas restricciones, geométricas y físicas. La más utilizada es la restricción epipolar, basada en la geometría relativa del sistema estéreo. Es posible utilizar esta restricción para transformar el sistema en uno ideal de cámaras paralelas, de modo que los puntos correspondientes se hallen en la misma línea en ambas imágenes. Esta transformación se llama rectificación epipolar, y simplifica en gran manera el *matching* posterior.

Los algoritmos de *matching* pueden clasificarse en dos grandes categorías: basados en características, que extraen puntos de interés en la imagen (normalmente bordes), sobre los que llevan a cabo el *matching*, y basados en área, que llevan a cabo la correlación de los niveles de gris en ventanas de las distintas imágenes, considerando que en el entorno de puntos correspondientes los patrones de intensidad deben ser similares. La gran ventaja de los algoritmos basados en área es que en la actualidad existen a precios razonables dispositivos orientados al procesamiento de imagen capaces de llevar a cabo convoluciones y correlaciones en tiempo real, con performances muy superiores a los procesadores de propósito general. También es posible una combinación de ambos sistemas, implementando la correlación sólo en puntos de interés previamente identificados. En nuestro caso el número de puntos de profundidad requerido no es excesivamente elevado, por lo que esta parece ser la elección óptima. En todo caso debe tenerse en cuenta que, al contrario que la calibración, no puede decirse que el problema de la correspondencia en estéreo visión esté resuelto, sino que sigue siendo un problema abierto, en el que las soluciones implementadas suelen ser en gran medida dependientes de la aplicación concreta.

La reconstrucción 3-D no plantea problemas en sí misma, pero en caso de que el número de correspondencias encontradas sea bajo es posible que deba llevarse a cabo una interpolación, que puede ser más o menos sofisticada dependiendo de si se quieren mantener las discontinuidades en profundidad presentes en la escena. En nuestro caso el número de puntos requeridos no es alto, por lo que la interpolación no sería necesaria.

6.2 Metodología.

Existen métodos diversos para representar una imagen 3D; para ver una escena, debemos establecer mediante coordenadas cartesianas la ubicación del observador (o cámara) desde donde se verá el modelo tridimensional. Frente a este punto de vista se establece un plano de visión en el cual se proyecta la información geométrica de las formas geométricas 3D.

Cuando el sistema utiliza varias cámaras que se controlan desde la aplicación dependiendo de su desarrollo en realidad virtual, conviene no especificar una cámara en el archivo de ejecución VRdisplay.cfg sustituyendo el nombre de ésta por el signo <_> (guión bajo). En la siguiente línea:

```
View_0_0 both   _   0   0   1024   768
```

De esta sencilla forma se consigue que el objeto cámara no sea fijo y se pueda interactuar con varias cámaras en la ejecución en tiempo real del proyecto.

6.3 Desarrollo.

Algunos de los objetivos dentro del desarrollo del sistema fueron:

1. sistema de colisión. Para 2 objetos en movimiento (uno es la cámara) que detecten el sistema de colisión, pero que la cámara no pierda el objeto que enfoca, que no atraviese los objetos y que no quede atrás de los objetos.
2. intentar hacer un cambio de cámaras cuando ocurren diferentes acciones (subir escaleras, dar una vuelta, bajar escaleras) para no perder la imagen.
3. seguir la imagen en movimiento rápido, y evite crear mareos.
4. interacción si hay otro personaje para cambiar de posición y tipo si fuera necesario.
5. ver el comportamiento del giro durante la ascensión y descenso de escaleras. Detectar cuando deja de subir la cámara en relación a que gira el objeto focal.
6. establecer un detector de piso – escalera – no piso – no escalera
7. reinicio de valores para la cámara y el objeto a fin que no se bloqueen las condiciones del sistema de colisión.

La cámara como objeto dentro de la programación de bloques de virtools, consta de dos características a definir: Posición y Orientación.

La característica de posición se define en base al plano de coordenadas xyz y especifica precisamente la posición que la cámara tendrá en el proyecto.

La característica de orientación define hacia dónde estará orientada la cámara. Es decir, hacia dónde ve.

El procedimiento para asignar una cámara en virtools es el siguiente:

1. Dar click en el icono de cámara.
2. Aparece en <level manager> la opción de "Camera"
3. En new camera asignar <set as active camera>
4. Establecer las coordenadas y/o características en <target camera setup>

A continuación se asigna el controlador de la cámara:

1. En <level manager> dar click en botón derecho en <level> y <create a new script>
2. En <schematics> ya aparecen dos pestañas y se podrá asignar el o los controladores que se necesiten como joystick, Mouse, teclado.

Para crear una cámara que siga al personaje:

- La opción ADD CHILD no funciona bien, se debe especificar algún parámetro extra.
- Se agregaron los bloques SET POSITION + SET EULER ORIENTATION para recrear el movimiento de la cámara siguiendo al personaje.

- Se agregaron los bloques ROTATE AROUND + SET EULER ORIENTATION para recrear el movimiento de la cámara alrededor del personaje.

- Es más factible observar el movimiento si se sitúa la imagen desde el TOP VIEW de la cámara (llamado VISOR)

El siguiente aspecto fundamental para presentar mejor el proyecto fue la asignación de varias cámaras. La asignación y montaje de cada una se determinó según las necesidades de visualización que se esperaban. El manejo y control de cámara queda determinado por el esquema siguiente:

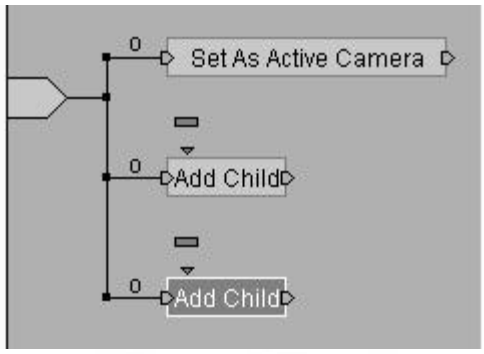
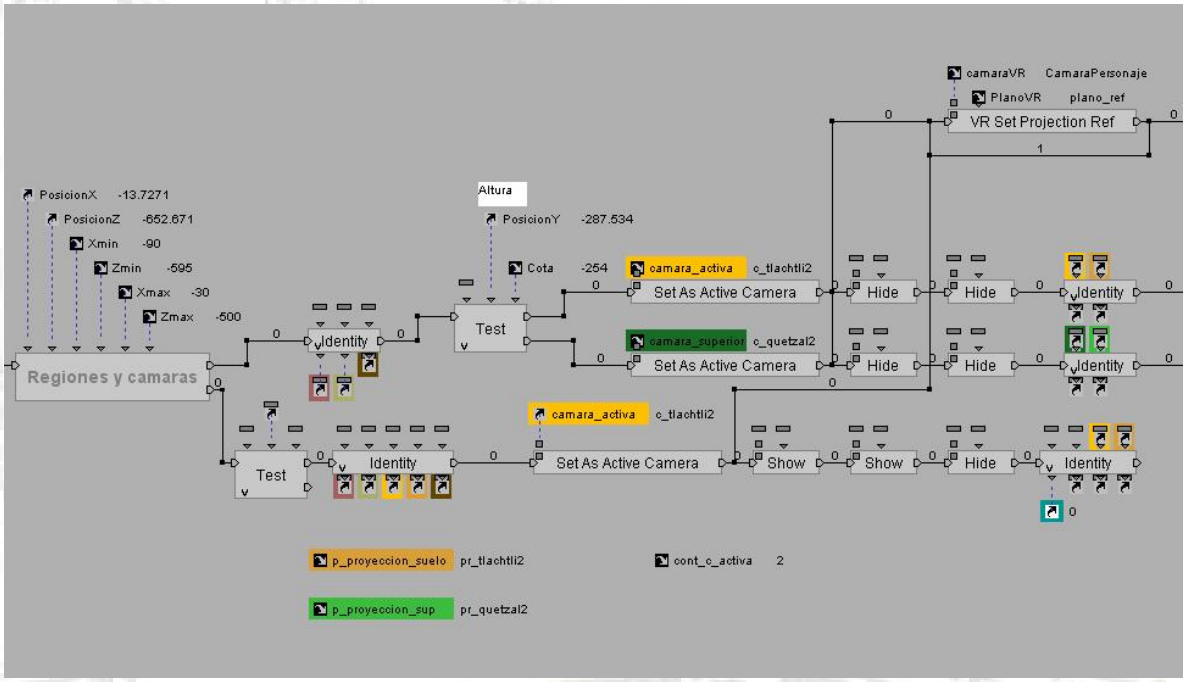


Figura 24. Building Block de Virtools para el manejo de cámaras.

Dentro del proyecto se determina cuál será la cámara activa. Usando el siguiente código:

Figura 25. Esquema de programación en virtools para el control de cámaras activas.



La cámara activa será determinada por las coordenadas, el lugar y la acción que se está llevando a cabo.

Para ello se toman en cuenta unos planos invisibles de referencia, que a la vez se usaron como planos de colisión.

El sistema de colisión tiene por objetivo que dos objetos en movimiento (considerando uno de ellos como la misma cámara) no choquen entre ellos al efectuar el movimiento, no atraviesen objetos considerados sólidos, no queden dentro de los objetos sólidos durante la interacción dependiente del movimiento de uno con el otro.

SISTEMA DE DETECCIÓN DE COLISIONES

1. Se utilizó un equipo de 4 cámaras con sensores
2. Se cambia la posición del caballero cuando gira
3. Se cambia la posición de la cámara cuando el caballero gira
4. Se implementó un sistema de sensado de distancia vertical para controlar la subida y bajada de escaleras.

En la imagen a continuación, se especifican los parámetros de cámara y plano de referencia, mediante los cuales se controla el cambio de cámaras para el proyecto.

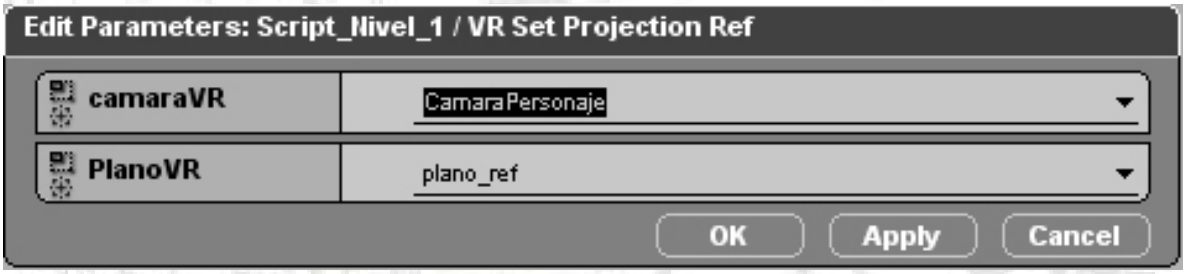


Figura 26. Building Block en el que se especifica cámara y plano de referencia.

Se generó un sistema de cambio de cámaras mediante la asignación de variables según las restricciones del proyecto. Con esto, se amplió la posibilidad de movimientos de los personajes dinámicos, así como la definición de la estructura sólida de los objetos fijos en un mayor área de movimiento.

Eso nos determina en dónde se encuentra el caballero águila, si está entrando o saliendo del edificio o si se encuentra en una zona abierta. Así, la cámara activa siempre nos va a dar la mejor vista de la acción que se realiza. Además podemos en este mismo bloque asignar el video que se mostrará si es que se encuentra dentro del área de algún edificio, como podemos notar en el código siguiente:

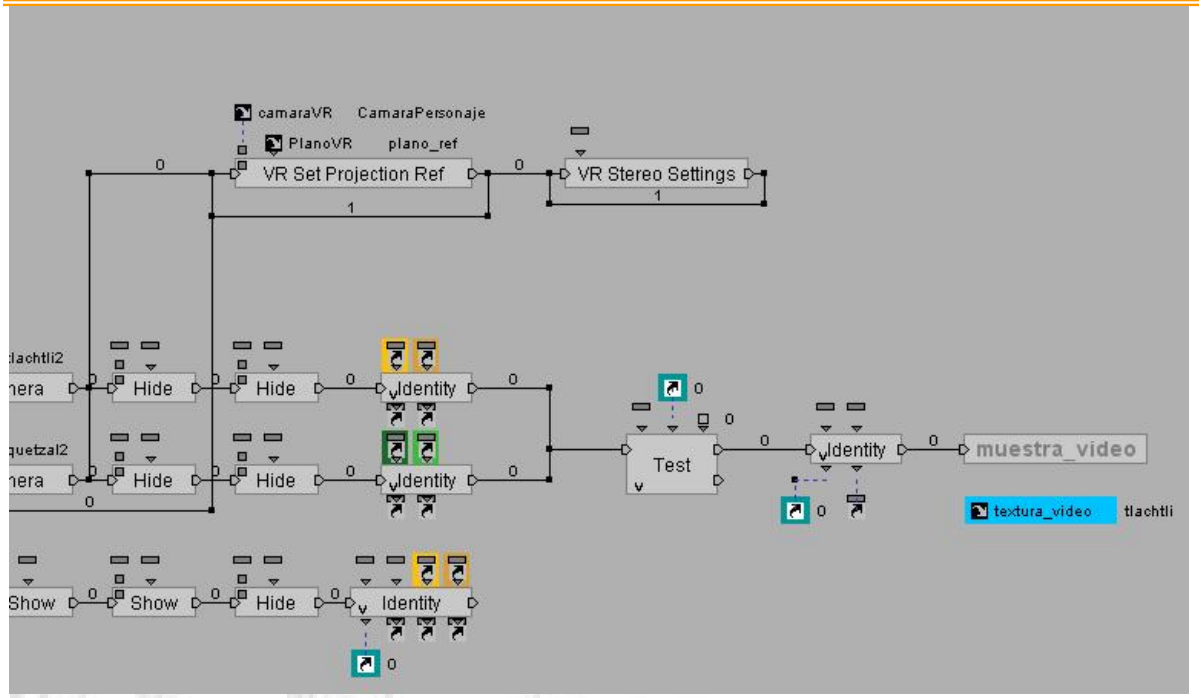


Figura 27. Programación en virtools para la interacción de cámaras.

Como extra, se integro música ambiental al proyecto, esto se hace utilizando el BB siguiente:

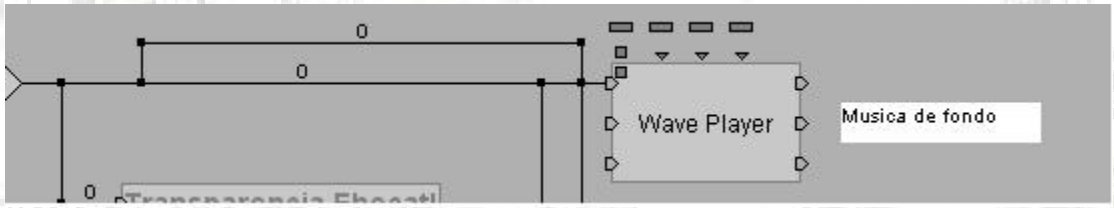


Figura 28. Building Block que ayuda a integrar sonidos al sistema.

Puede hacer una programación independiente para que la música varíe según las actividades o efectos que se deseen en el proyecto. Es opcional.

7. Estereoscopia.

7.1 Sustento teórico.

Breve Historia sobre la estereoscopia.

		
Euclides. (330 a.C. – 275 a.C.) Matemático griego. Poco se conoce a ciencia cierta de la biografía de Euclides, pese a ser el matemático más famoso de la Antigüedad.	Da Vinci. (1452-1519), artista florentino y uno de los grandes maestros del renacimiento, famoso como pintor, escultor, arquitecto, ingeniero y científico. Sus investigaciones científicas —sobre todo en las áreas de anatomía, óptica e hidráulica— anticiparon muchos de los avances de la ciencia moderna.	Kepler. (Würtemburg, actual Alemania, 1571-Ratisbona, id., 1630) Astrónomo, matemático y físico alemán.

Figura 29. Retratos. Estereoscopía.

Parece que Euclides y el genial Leonardo da Vinci ya observaron y estudiaron el fenómeno de la visión binocular, siendo considerados los pioneros en este tema. También el famoso astrónomo Kepler llevó a cabo estudios sobre la estereoscopia. Curiosamente la estereoscopia precedió a la fotografía. Fue un físico escocés, Sir Charles Wheatstone, quién en Junio de 1838 describió primero con cierto rigor el fenómeno de la visión tridimensional y construyó luego un aparato con el que se podían apreciar en relieve dibujos geométricos: el Estereoscopio.

	Años más tarde, en 1849, Sir David Brewster diseñó y construyó la primera cámara fotográfica estereoscópica, con la que obtuvo las primeras fotografías en relieve. Construyó también un visor con lentes para observarlas. Posteriormente, Oliver Wendell Holmes, en 1862, construyó otro modelo de estereoscopio

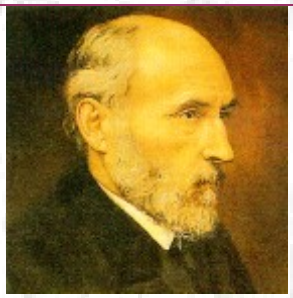
	de mano que se hizo muy popular a finales del siglo XIX. Con él podían verse en relieve fotografías estereoscópicas montadas sobre un cartón. Se crearon extensas colecciones y se pusieron a la venta. Podían encontrarse fotografías en relieve de cualquier parte del mundo.
	Esta es una muestra de una vieja fotografía estereoscópica montada sobre cartón. Para poderla ver en relieve era necesario un visor tipo Holmes como el que aparece en la fotografía de la primera página. Actualmente, gracias a la informática, puede verse en la pantalla del ordenador sin necesidad del antiguo estereoscopio.
	Ilustres científicos, como el Premio Nobel aragonés Don Santiago Ramón y Cajal, utilizaron la estereoscopía para presentar sus trabajos científicos. Don Santiago presentó parte de sus muestras para microscopio en fotografías estereoscópicas. Era un gran aficionado a esta técnica y a la fotografía en general.

Figura 30. Imágenes ilustrativas a la estereoscopía.

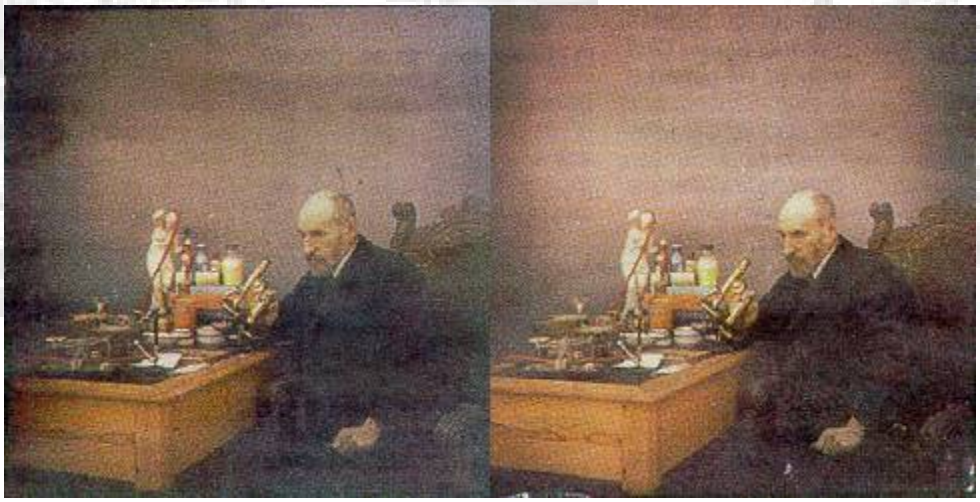


Figura 31. Autorretrato estereoscópico de Don Santiago Ramón y Cajal.

Durante los años 30, hubo un resurgir de la estereofotografía a raíz de la aparición de cámaras 3D con película de 35mm. como la Realist o la ViewMaster, que facilitaban al aficionado la obtención de este tipo de imágenes. Desgraciadamente, estas cámaras ya no se fabrican, y son hoy en día objeto de colección y sólo pueden encontrarse en tiendas de material de ocasión.

Figura 32. Estereofotografía.

También en el arte algunos pintores han usado la representación estereoscópica. Por ejemplo, Salvador Dalí utilizó un dispositivo de espejos similar al de Wheatstone para mostrar algunos de sus trabajos. Pueden verse actualmente en el museo de Figueras.

(Fotografía: IMAX Corporation)

En los años 50 se intentó la explotación comercial de películas 3D y aparecieron los primeros títulos, pero con escasa incidencia en el mercado cinematográfico. No pasaron de ser meras curiosidades para el público. Además, algunas de las películas que se realizaron presentaban problemas de visión, por no conocer algunos de los técnicos de la época toda la problemática que conlleva una película estereoscópica, lo que ocasionaba molestias visuales que hicieron que una parte del público rechazara este tipo de cine. No sería hasta los años 80 cuando se conseguirían los resultados más espectaculares, con los sistemas de gran formato de película, como el de IMAX, para conseguir imágenes de alta resolución en pantallas gigantescas, tras grandes inversiones en investigación y medios.

En los años noventa, los avances de la informática permiten presentar imágenes 3D en monitores de ordenador y utilizarlas para presentaciones en CAD, Medicina, cartografía y otras muchas aplicaciones. Los ordenadores permiten además generar espectaculares imágenes de síntesis en relieve, para aplicaciones científicas, industriales o de entretenimiento. Recientemente la NASA ha utilizado

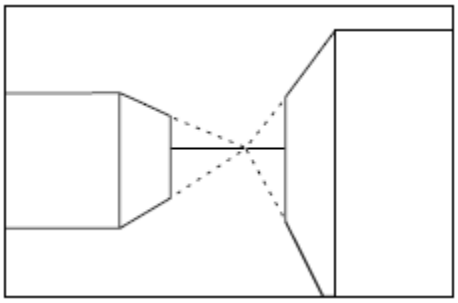
la estereoscopia como una herramienta para ver en 3D y analizar las imágenes de Marte enviadas por la sonda Pathfinder.

Teoría sobre la percepción en 3D.

Existen claves que el ojo humano usa para interpretar la profundidad de un ambiente en 3D (Tercera dimensión). Algunos están incluso presentes en imágenes 2D:

Perspectiva

Los objetos se observan pequeños mientras mas lejos están y convergen con la línea paralela de visión a la distancia.



Tamaño de los objetos conocidos

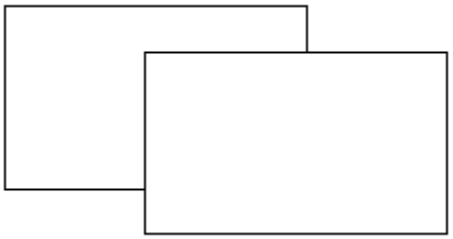
Se espera que algunos objetos sean menores que otros. Si un elefante y una taza de te parecen del mismo tamaño, imaginamos que el elefante se encuentra en la lejanía.

Detalles

Los objetos cercanos se ven con gran detalle, mientras que los lejanos con menos (nitidez y difuso).

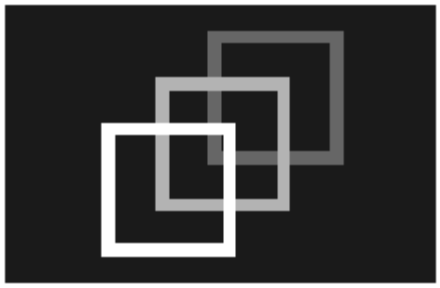
Occlusion

Un objeto que oculta a otro se asume que se encuentra delante de éste.



Luz Sombras

Los objetos cercanos son más brillantes que los distantes.



Movimiento Relativo

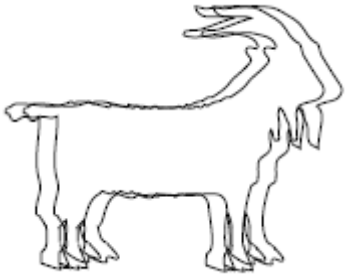
Los objetos lejanos parecen más lentos al moverse que los que se encuentran más cercanos.

Figura 33. Imágenes alusivas a la percepción 3D.

Existen otras claves que no se encuentran presentes en imágenes 2D y son:

Disparidad Binocular

Esta es la diferencia en las imágenes proyectadas sobre la parte posterior del ojo (y sobre la corteza visual) porque los ojos están separados horizontalmente por la distancia interocular.



Alojamiento

Es la tensión muscular necesaria para cambiar la distancia focal del lente del ojo para enfocar una distancia en particular.

Convergencia

Es la tensión muscular requerida para rotar cada ojo y así cambiar el punto focal de visión.

Para obtener un par estéreo pueden emplearse diferentes procedimientos. Con una sola cámara podemos obtener las dos imágenes, en dos tiempos, desplazando la cámara una distancia similar a la de separación de los ojos, unos 65mm. Naturalmente, el sujeto no debe moverse entre las dos tomas, por lo cual este procedimiento sólo sirve para fotografiar objetos inmóviles. Si queremos obtener fotografías 3D en movimiento debemos emplear una cámara estéreo especial, una cámara convencional dotada de un accesorio especial con espejos o bien dos cámaras disparadas sincronizadamente.

El funcionamiento de la **visión estereoscópica** es básicamente el siguiente: Se obtienen dos imágenes de una misma escena, desde dos puntos de vista diferentes (normalmente ligeramente diferentes). Una vez establecida la correspondencia entre puntos de las dos imágenes que corresponden a un mismo punto de la escena, mediante una sencilla triangulación puede hallarse la distancia de ese punto a las cámaras. Esto, que dicho así parece muy sencillo, plantea los siguientes grandes problemas:

1- **Definición del modelo de las cámaras y calibración de las mismas.** Una vez definido el modelo de cámara, se trata de hallar los parámetros que relacionen las coordenadas de la imagen en pixeles con las correspondientes a algún sistema de referencia del mundo real, en unidades métricas.

2- **Obtención de los puntos correspondientes en ambas imágenes (*matching*)**. Es el mayor problema de la estereovisión, a causa de su ambigüedad. En principio, un punto de una imagen puede corresponder a cualquier punto de la otra. Se utiliza una serie de restricciones geométricas y físicas para tratar de reducir la ambigüedad de la correspondencia.

3- **Reconstrucción 3-D**. Una vez realizado el *matching*, el cálculo de la profundidad es inmediato, y sólo plantea algún problema de implementación práctica, ya que se obtiene mediante un sistema de ecuaciones sobredeterminado. El inconveniente puede ser que el número de puntos para el que se halló correspondencia sea escaso, lo que obliga a realizar una interpolación para la obtención de un mapa de profundidades denso.

Descripción de los elementos.

La tarjeta de video permite la proyección simultánea de 2 imágenes, éstas se proyectan a través de los cañones (conectados a cada una de las salidas de la tarjeta de video). Las imágenes de los cañones necesitan ser filtradas y

proyectadas, usando la técnica de polarización de la luz (que consiste en la supresión de ondas de luz en una dirección o polaridad), hacia una pantalla especial que no rompa el plano de polarización. Recibimos la señal visual con un filtro fijo a 90 ° (lentes polarizados) que nos darán dos imágenes una para cada ojo, las cuales interpretaremos como imagen 3D.
Este es el mejor método de proyección de video (o cine) 3D, pues se conserva la calidad de imagen, los fantasmas de disminuyen y el cansancio visual del espectador es mínimo, el inconveniente es que se requiere equipo especial de proyección y pantallas especiales.

Sistemas de presentación de imágenes estereoscópicas

Para poder observar correctamente una imagen estereoscópica, cada ojo debe ver únicamente la imagen que le corresponde. Para ello se han ideado diversos sistemas que se describen brevemente a continuación:

VISIÓN LIBRE PARALELA 

Los ojos observan cada uno su imagen correspondiente, manteniendo sus ejes ópticos paralelos, es decir, como si mirásemos al infinito. Sólo puede usarse este método con imágenes no superiores a 65 milímetros entre sus centros. Es el método usado para ver las imágenes de los libros con estereogramas de puntos aleatorios ("ojo mágico").

VISIÓN LIBRE CRUZADA 

Las imágenes se observan cruzando los ejes ópticos de los ojos. El par estéreo se presenta invertido, es decir, la imagen derecha está situada a la izquierda y viceversa. Podemos ayudarnos mirando un lápiz situado entre nuestros ojos y las imágenes. Este método debe usarse con imágenes de dimensiones superiores a **65 milímetros** entre sus centros, aunque la imagen virtual aparece más pequeña.

ANÁGLIFO 

Se utilizan filtros de colores complementarios, como rojo-azul, rojo-verde o ámbar-azul. La imagen presentada por ejemplo en rojo no es vista por el ojo que tiene un filtro del mismo color, pero sí que ve la otra imagen en azul o verde. Este sistema, por su bajo costo, se emplea sobre todo en publicaciones, así como también en monitores de ordenador y en el cine. Presenta el problema

de la alteración de los colores, pérdida de luminosidad y cansancio visual después de un uso prolongado. Normalmente se sitúa el filtro rojo en el ojo izquierdo, y el azul en el ojo derecho. Existen dos sistemas muy similares en ámbar-azul: el sistema SpaceSpex de 3DTV Corporation (Naranja-Azul) y el ColorCode 3-D, empleado ya en alguna película formato Imax y comercializado también en algún DVD en 3D. Este sistema es bastante eficaz, pero el filtro azul del ojo derecho es demasiado oscuro para una visión cómoda. En general el sistema anaglifo no es cómodo para usarlo durante un tiempo prolongado, pero por su fácil uso y bajo costo es el que se utiliza para el proyecto Tenochtitlan descrito en este texto.

Figura 34. Fotografía: ©Alfredo González

POLARIZACIÓN

Se utiliza luz polarizada para separar las imágenes izquierda y derecha. El sistema de polarización no altera los colores, aunque hay una cierta pérdida de luminosidad. Se usa tanto en proyección de cine 3D como en monitores de ordenador mediante pantallas de polarización alternativa. Hoy día es el sistema más económico para una calidad de imagen aceptable. Este es el sistema que se utiliza para los proyectos de esta metodología propuesta.

Figura 35. Fotografía: ©Alfredo González

ALTERNATIVO

Con este sistema se presentan en secuencia y alternativamente las imágenes izquierda y derecha, sincronizadamente con unas gafas dotadas con obturadores de cristal líquido (denominadas LCS, Liquid Crystal Shutter glasses o LCD, Liquid Crystal Display glasses), de forma que cada ojo ve solamente su imagen correspondiente. A una frecuencia elevada, el parpadeo es imperceptible. Se utiliza en monitores de ordenador, TV y cines 3D de última generación.

Figura 36. Fotografía: ©Alfredo González.

HEAD MOUNTED DISPLAY (HMD)

Un HMD es un casco estereoscópico que porta dos pantallas y los sistemas ópticos para cada ojo, de forma que la imagen se genera en el propio dispositivo. Su principal uso hasta ahora ha sido la Realidad Virtual, a un costo prohibitivo y de forma experimental, aunque al bajar de precio aparecen otras aplicaciones lúdicas, como los videojuegos.

Figura 37. Fotografía artefacto para realidad virtual.

MONITORES AUTO-STEREO

Se están desarrollando prototipos de monitores que no precisan gafas especiales para su visualización. Todos ellos emplean variantes del sistema lenticular, es decir, microlentes dispuestas paralela y verticalmente sobre la pantalla del monitor, que generan una cierta desviación a partir de dos o más imágenes (normalmente de 2 a 8).

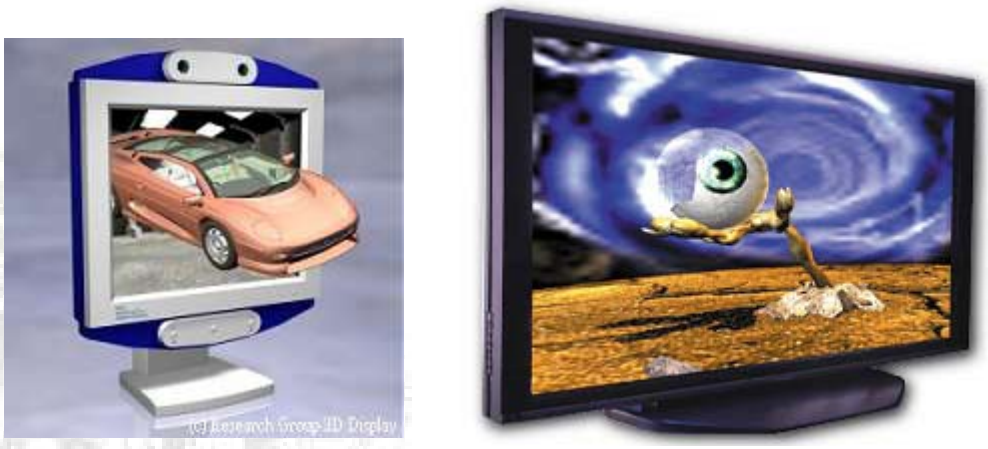


Figura 38. Tercera dimension

EFECTO PULFRICH

El llamado Efecto Pulfrich fue descubierto por el médico alemán Carl Pulfrich en 1922. El fenómeno es la percepción de un efecto estereoscópico cuando se observa una imagen en movimiento horizontal sobre un plano y con un filtro oscuro situado delante de uno de los ojos. Debido a la menor luminosidad que percibe el ojo con el filtro, la imagen llega al cerebro con un retardo de unas centésimas de segundo. Por tanto, en la estereopsis el cerebro percibe la misma imagen pero con una pequeña diferencia de posición horizontal, lo que genera el efecto estereoscópico. No es propiamente un sistema de visualización estéreo, ya que no se parte de un par de imágenes sino de una única imagen 2D animada. Sin embargo pueden obtenerse efectos estereoscópicos muy espectaculares filmando con una única cámara en movimiento. Se han ideado incluso sistemas para generar el movimiento de las imágenes sin mover la cámara (CircleScan 4D). Diversas cadenas de televisión han presentado en alguna ocasión filmaciones preparadas para efecto Pulfrich.

CHROMADEPTH™

El sistema ChromaDepth™ de ChromaTek Inc. se basa en la desviación que producen los diferentes colores del espectro. En un prisma, la luz se desvía ligeramente dependiendo de su longitud de onda: más desviación en el rojo, menos en el azul. La información de profundidad se codifica por colores. Las gafas

especialmente diseñadas para ver éstas imágenes disponen de unos cristales transparentes con microprismas. Cuando la imagen, denominada CyberHologram™, se observa con las gafas HoloPlay™ (para imágenes de ordenador) o C3D™ (para imágenes impresas), la imagen 2D se convierte en tridimensional. La desventaja de este sistema es la pérdida de información cromática, pero la ventaja sobre el anáglifo es que las imágenes pueden verse también en 2D.

Aplicaciones de la estereoscopia

Actualmente diversos campos científicos y técnicos se benefician de la estereoscopia. Por citar algunos:

- TOPOGRAFÍA Y ESTUDIO DEL TERRENO

Una de las aplicaciones prácticas más antigua es la visualización y medición del relieve terrestre mediante fotografías aéreas.

- ESTUDIO DE LA TIERRA Y OTROS PLANETAS

De forma similar a la fotografía aérea, NASA ha obtenido numerosas vistas tridimensionales de fotografías de la Tierra obtenidas desde satélites, así como también de otros planetas de nuestro Sistema Solar.

- CAD (Diseño Asistido por Computador) y CAE (Ingeniería Asistida por Computador)

Es una poderosa herramienta para diseño y visualización de prototipos, con un importante ahorro en tiempo y dinero durante el desarrollo. Los más importantes paquetes y estaciones de diseño por ordenador, como IBM, HP, DEC, Sun o Silicon Graphics, soportan actualmente la visualización estereoscópica mediante gafas LCS, como las de Stereographics o VRex.

- MEDICINA

Es uno de los campos en los que la estereoscopia proporciona más ayuda para la enseñanza, la interpretación de imágenes para el diagnóstico o como ayuda en las intervenciones.

- INGENIERIA MOLECULAR

En ingeniería molecular, sin la visualización estéreo en estaciones de diseño sería muy difícil crear nuevas moléculas complejas.

- TELEPRESENCIA

Sistemas de video-cámaras estéreo permiten operar en entornos peligrosos u hostiles con la máxima precisión. NASA ya tiene cierta experiencia respecto a sistemas de telepresencia submarina.

- REALIDAD VIRTUAL

La técnica denominada Realidad Virtual básicamente es una interacción usuario-ordenador en la que se generan las imágenes estereoscópicas en tiempo real, introduciendo al espectador en un escenario 3D artificial. Por citar algunas, se encuentran las siguientes aplicaciones:
En arquitectura, donde la Realidad Virtual nos permite navegar por el interior del edificio antes de que se construya.
En arqueología, permite recrear edificios y ciudades de viejas civilizaciones, o ayudar en la restauración de monumentos.
En medicina, es posible simular intervenciones quirúrgicas o navegar por el interior del cuerpo humano para planificar operaciones o en la enseñanza. Los sistemas llamados "Realidad Aumentada" superponen a una imagen real otra generada por ordenador. Esto permite que el cirujano vea sobre la zona de intervención una imagen sintética tridimensional, con indicaciones precisas en un punto de interés especial. Puede ser de gran ayuda en operaciones delicadas, como por ejemplo en el cerebro.
En la industria automovilística, es posible situarse al volante de un automóvil antes de fabricarlo.
En la industria aerospacial, en simuladores de vuelo de aviones o para simular entornos de naves espaciales u operaciones en el espacio.
Del cálculo exacto de los parámetros de visión estereoscópica depende mucho el realismo del entorno virtual en todas estas aplicaciones.

7.2 Metodología.

Para que el archivo tenga las características 3D, existe una herramienta en virtools, que nos ayudará con la recreación 3D para su ejecución en tiempo real.

Es importante hacer notar que aunque el sistema quede habilitado para un despliegue tridimensional, éste sólo se apreciará si y sólo si se tienen las herramientas para mostrarlo.

Para habilitar el sistema 3D, el procedimiento es relativamente sencillo:

1. Se crea un nuevo Script en RV-Script.
2. Utilizamos los Building Block <VR Set Projection Ref> y <VR Stereo Settings> y los "cerramos" en ciclo en sí mismos.
3. La especificación en ambos es la siguiente:
    a. Front Camera
    b. Plano de referencia
    c. Distancia interpupilar o interocular = 0.65

Figura 39. Distancia interpupilar.

En la siguiente imagen, se aprecia la estructura en que se conectan éstos parámetros para simular la imagen 3D del proyecto. Para ello, cabe notar la interacción básica que existe entre el control de cámaras con el plano de referencia para poder ejecutar de una forma coherente la simulación de las imágenes en 3D dentro de la ejecución en tiempo real del proyecto.

Es importante también verificar que la ejecución de las distintas variables del proyecto no produzca efectos aberrantes en la muestra 3D por que se puede traducir en cansancio, mareo o dolor de ojos de los espectadores. Por eso, es importante realizar varias pruebas previas.

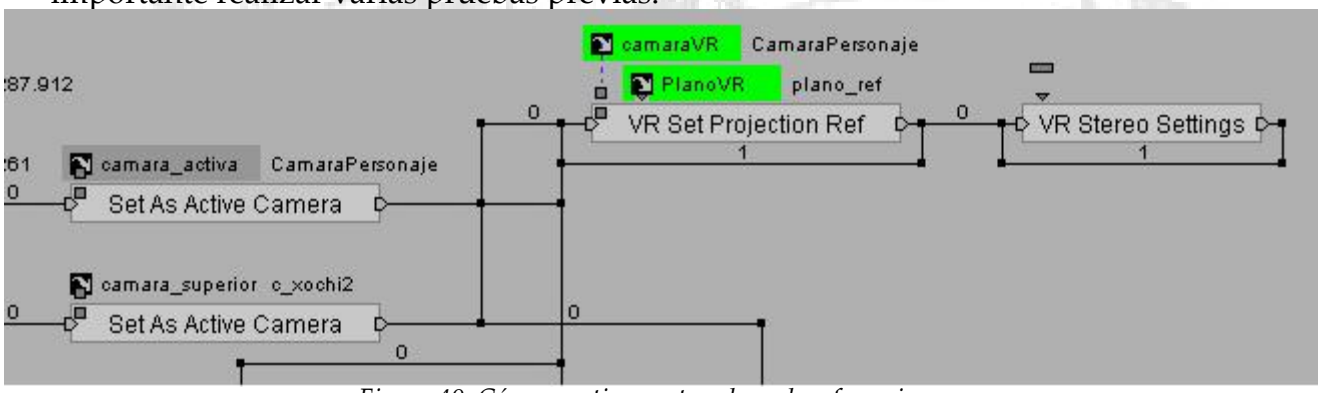


Figura 40. Cámara activa contra plano de referencia.

7.3 Desarrollo.

Los bloques Building Blocks <VR Set Projection Ref> y <VR Stereo Settings> son los responsables de la simulación 3D del proyecto.
Se observa en la figura la configuración para su óptimo desempeño.

El bloque de cámara es variable según la estructura programada y la asignación de la cámara depende de la variable <camara_activa> y <cámara_superior>.

En el bloque inferior derecho, está el sistema de colisión de cámaras, con el cual se envían las señales para cada cambio de cámaras según se requiera durante la ejecución en tiempo real del proyecto.

En el bloque de interdistancia pupilar <VR Stereo Settings>, asignamos el valor 0.65 descrito anteriormente, que es la distancia de separación entre las dos imágenes que creará el sistema y que saldrá en cada cañón de proyección para mostrar la imagen tridimensional en pantalla.
La variación en la asignación de este valor, se traduce en una mala emulación de la figura tridimensional, en la sensación de mareo o dolor de cabeza y ojos para los espectadores de la imagen por un tiempo prolongado o simplemente en la nula apreciación del sistema en tercera dimensión.

En el bloque de proyección <VR Set Projection Ref> se especifican los valores de cámara y plano de referencia, a partir de los cuales se generarán las imágenes dobles para recrear la imagen tridimensional. Estos valores, dada la estructura del proyecto, también son variables y dependen de las variables (valga la redundancia): <camaraVR> y <PlanoVR> las cuales van ligadas directamente al sistema de colisión descrito anteriormente.

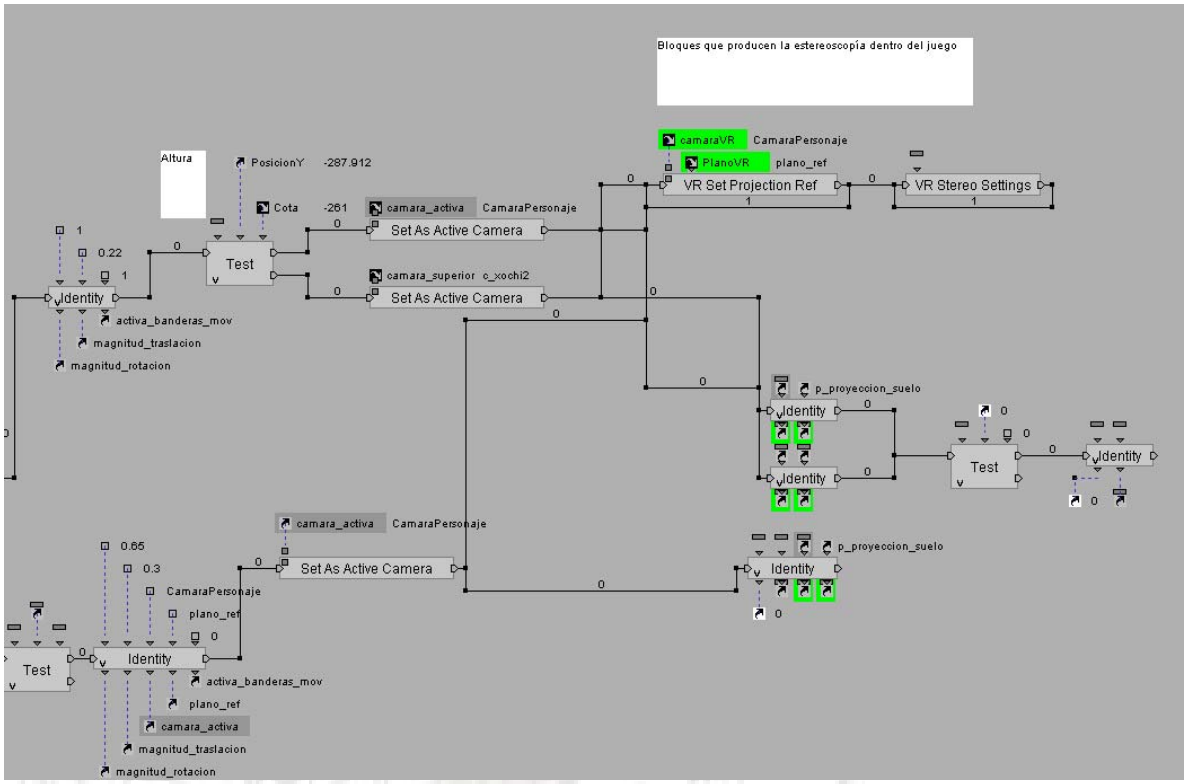


Figura 41. Sistema de colisión.

8. Proyecto Interactivo.

8.1 Animación.

El objetivo del proyecto es crear una animación en realidad virtual y 3D que describa algunos aspectos de la gran Tenochtitlan y al mismo tiempo sea de gran atractivo para los infantes. Se desea obtener un juego de metas, donde cada meta completada nos lleve al siguiente nivel de juego. El tiempo máximo de duración es de 15 min.

Objetivo 1. Como el lugar es amplio visto desde la visión del caballero águila se necesita una guía de ubicación para saber en que punto del mapa nos encontramos.

Para resolver esto, utilizamos un plano en 2D del complejo arquitectónico y un punto rojo para representar al caballero. Considerando el tamaño de la imagen 2D de 250x250 pixeles y el mundo virtual 889x928.5 Unidades Lineales tenemos:

Δx, Δz imagen 2D.
δx, δz mundo virtual.

$$\left(\frac{\Delta x}{\delta x}\right) = \frac{250}{889} = 0.28121$$

$$\left(\frac{\Delta z}{\delta z}\right) = \frac{250}{928.5} = 0.26925$$

x′= x (0.28121) + 135.0584
y′= y (-0.26925) + 68.48248

Así obtenemos las coordenadas que se asignan al punto rojo de forma automática al movimiento del objeto (en este caso, caballero águila). Ver imagen:

Figura 42. Imagen del sistema 3D.

La programación para obtener este resultado es la siguiente:
Se puede observar la utilización de la fórmula antes diseñada para este efecto. El BB (Set component) es utilizado para detectar la posición del caballero águila. El BB (Set 2D position) es el que va a manipular la posición del punto rojo en el mapa.

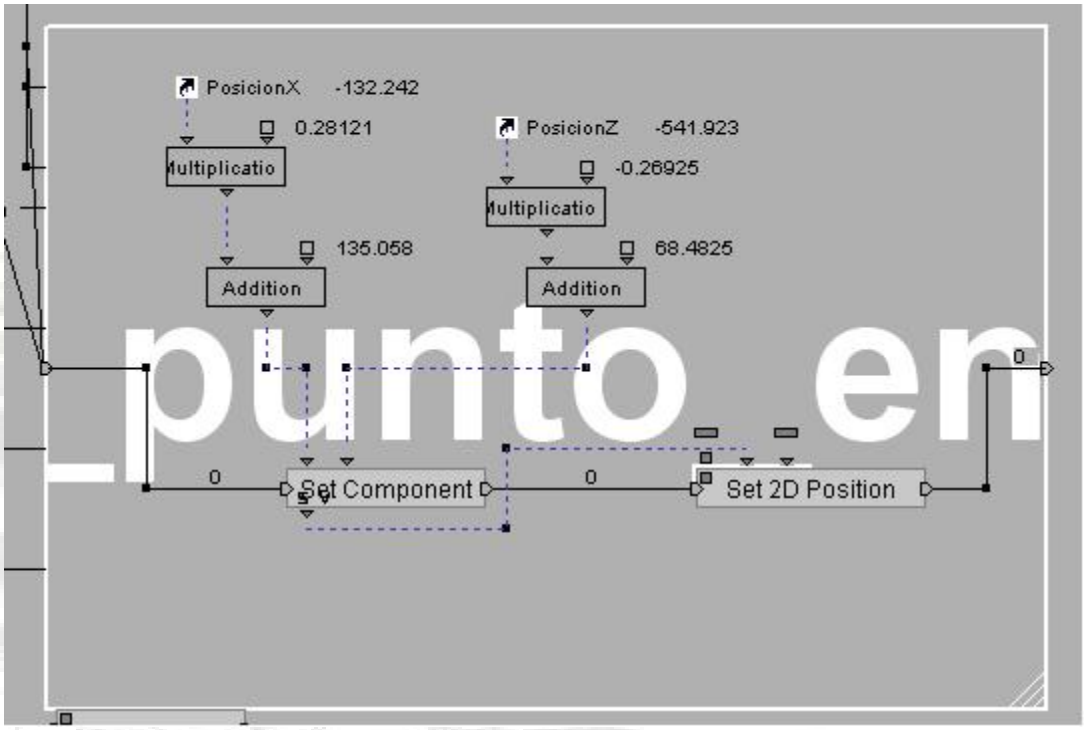


Figura 43. Control de mapa en el proyecto.

Programación de eventos

Para que las actividades que se desarrollan dentro del programa tuvieran mejor desempeño, se modificaron algunos aspectos de programación, estos son:

1. El Building Block de la estructura de <coordenadas de colisión> se estableció con los siguientes parámetros:

- Xmin que indicará el mínimo de valor que puede tener la coordenada X
- Zmin que indicará el mínimo de valor que puede tener la coordenada Z
- Xmax que indicará el máximo de valor que puede tener la coordenada X
- Zmax que indicará el máximo de valor que puede tener la coordenada Z
- CamaraActiva indicará cuál cámara queda activa según las restricciones anteriores.
- Camara_Sup indicará cuál es la cámara superior
- Cota_Sup indicará al sistema cuál es la cota superior dentro establecido para el sistema de colisiones.
- TexturaVideo es un indicador de video.

2. Se agregaron los Building Block de colisionador, lo que impide que el personaje atraviese las paredes.

3. Se agrego un Building Block <test> para controlar la velocidad del personaje haciéndolo más rápido si se observa de lejos y mas lento si observa de cerca. También se cambio de la misma forma para la rotación.

4. Se pusieron condiciones iniciales para el programa. Aparece el menú de inicio, se inicializan las variables de condición, se actualiza el plano con el punto rojo de posición del personaje, mismo que aparece en el punto de entrada definido previamente.

5. Se habilitaron las cámaras para el punto de Ojo de Agua y demás estructuras que no habían sido consideradas dentro del diseño original.

6. Se habilitaron los videos de cada estructura que participa en alguna de las rutas del juego.

7. También se agregaron las condiciones de colisión para las cámaras. Así éstas no atraviesan las paredes.

8. Se inició la investigación previa de la información que se desplegará dentro del programa acerca de la Zona Arqueológica Tenochtitlán. La cual deberá estar sustentada en base a investigaciones antropológicas realizadas por expertos en el campo. La forma en que se muestre esta información dependerá del departamento de diseño.

9. Se habilitaron nuevas cámaras para el Templo Mayor con el objetivo de que se observe mejor el detalle realista en los pequeños objetos que conforman esta estructura.

8.2 Personajes.

Para el desarrollo y elaboración de personajes se utilizó la herramienta 3DStudio Max, para afinar detalles. La imagen realizada en 3D se exporta a la plataforma de Virtools, donde conserva sus características físicas y se le pueden adaptar las diversas programaciones que tiene Virtools para la ejecución de movimientos varios.

Entre la variedad de personajes que se necesitarán para el proyecto, se debe definir cuáles necesitan ser detallados al máximo y cuáles no necesitarán serlo. Así evitaremos saturar el sistema con la ejecución excesiva de polígonos en tiempo real, optimizando la velocidad y desempeño del proyecto.

Algunas características del personaje, pueden tomarse directamente de la programación ya establecida de Virtools, incluso algunos personajes, para ahorrar tiempo de elaboración. Dentro de las librerías predeterminadas se encuentran: caminar, correr, detenerse, girar, entre otras. Así reutilizando los comandos para actividades simples, podemos agilizar la ejecución del proyecto sin necesidad de programar cada acción a cada personaje, según se requiera.

Entre los movimientos considerados para el personaje del proyecto (caballero águila) tenemos: caminar hacia adelante, caminar hacia atrás, correr, saltar, dar vuelta a la derecha, dar vuelta a la izquierda, subir escaleras, bajar escaleras.

Para el personaje de sacerdote no se consideraron movimientos específicos, pues no es necesaria para la ejecución del proyecto.

Es importante definir bien los eventos físicos del personaje para poder asignar las restricciones, cámaras, sistemas de colisión, etcétera; necesarios para que el proyecto cumpla con las expectativas esperadas

Para agregar un personaje predeterminado del sistema Virtools se siguen los siguientes pasos:

1. Seleccionar la opción <Virtools Resources>
2. Entrar a la opción <characteres>
3. Entrar a la opción <skin character>
4. Seleccionar el que deseemos (i.e. Jane)
5. Con el mouse hacemos el arrastre del personaje al área de dibujo.
6. Para la programación entrar a la sección de <Level Manager> y buscar el Building Block que aparece con su nombre
7. Entramos a la selección de propiedades de movimientos en <Virtools Resources> <characteres> <animations> <skin character animation> <Nombre_personaje>
8. Seleccionamos las propiedades que se deseen para ese personaje haciendo una selección de los mismos y un arrastre al Building Block del personaje

9. Un control predeterminado de virtools es el Building Block <character keeps on floor> el cual es en sí mismo un sistema de colisión automático
para todo lo que se defina en el proyecto como grupo <suelo>. Con éste, el personaje siempre se mantendrá con "los pies sobre la tierra" literalmente.
Este control lo encontramos en el menú <schematics>
10. Otro control importante para personajes es el <character controller> mediante el cual indicamos la animación a seguir por nuestro personaje. Este
building block está directamente ligado al controlador con que deseemos manipular nuestro proyecto, entre éstos están:
<keyboard controller> <joystick controller> y otros.

8.3 Variables de inicio.

Dentro de todo sistema se establece un inicio para que el programa se ejecute de forma correcta.

Para ello, se deben establecer los parámetros con los que se desea iniciar siempre el sistema que se está desarrollando, dentro de los cuales consideras:

- Posición de inicio de cámaras.
- Posición de inicio de personajes.
- Secuencia inicial de sonidos (si necesitara).

Para mejor referencia ver imágenes anexas:

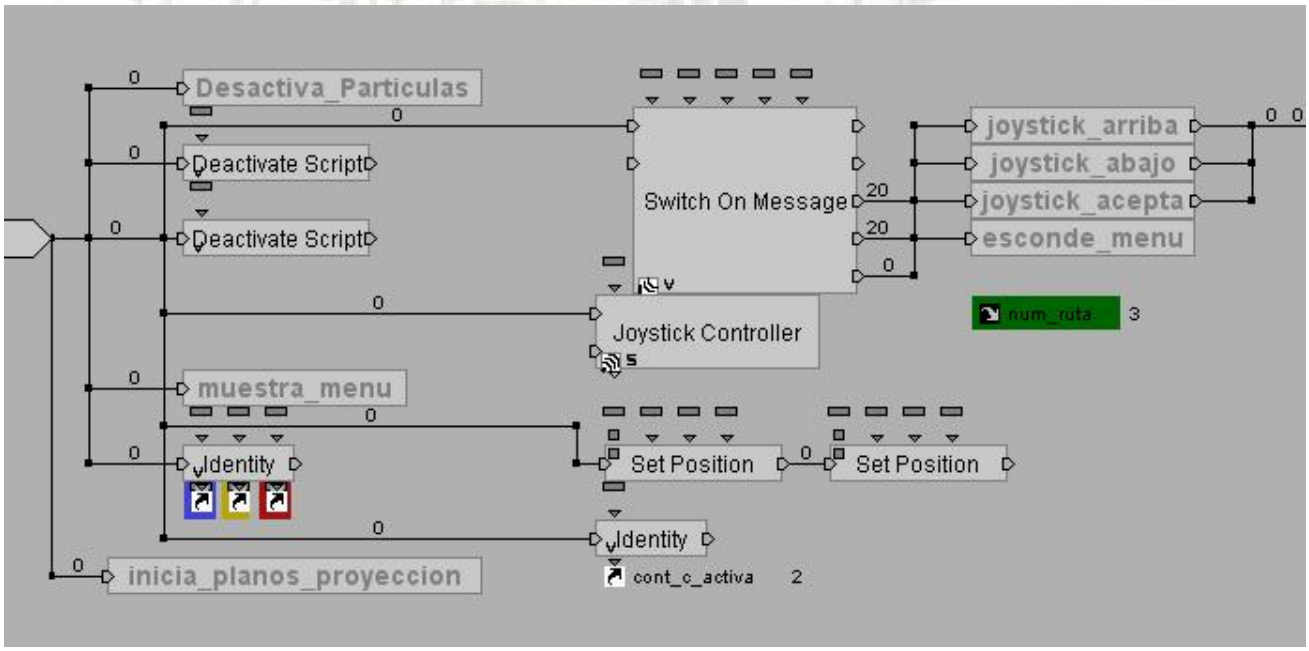


Figura 44. Reinicio de variables.

En esta imagen, se define el valor de las variables de inicio, la posición de cámaras inicial así como la posición de personajes. En general, podemos definir todas las condiciones con que esperamos inicie el proyecto, podemos incluir imágenes, sonidos, efectos, reinicio de valores o conservación de los mismos a la última ejecución, cambio de personajes, etc.

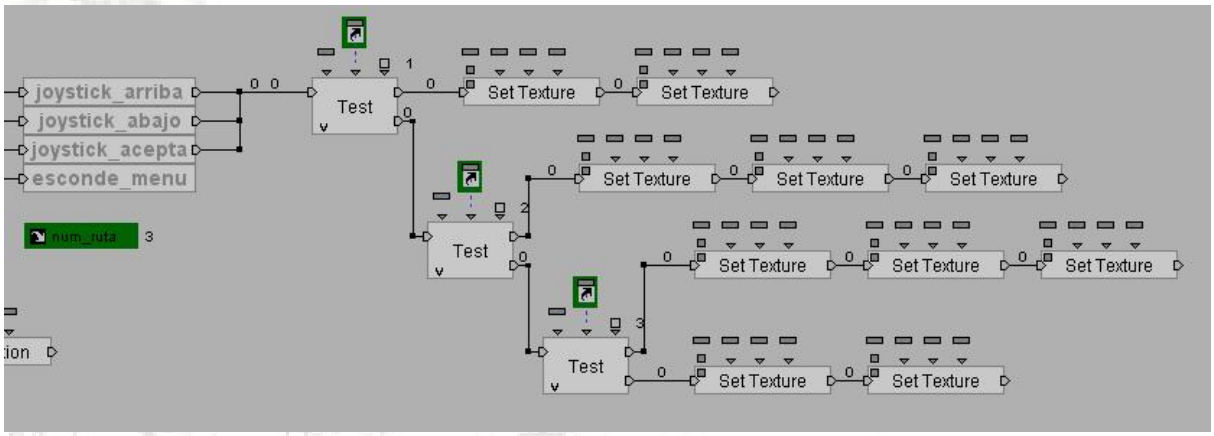


Figura 45. Menú de variables.

En esta imagen se observa como la relación con el joystick va a modificar algunas características de inicio. En este caso altera las texturas de iniciación de los objetos.

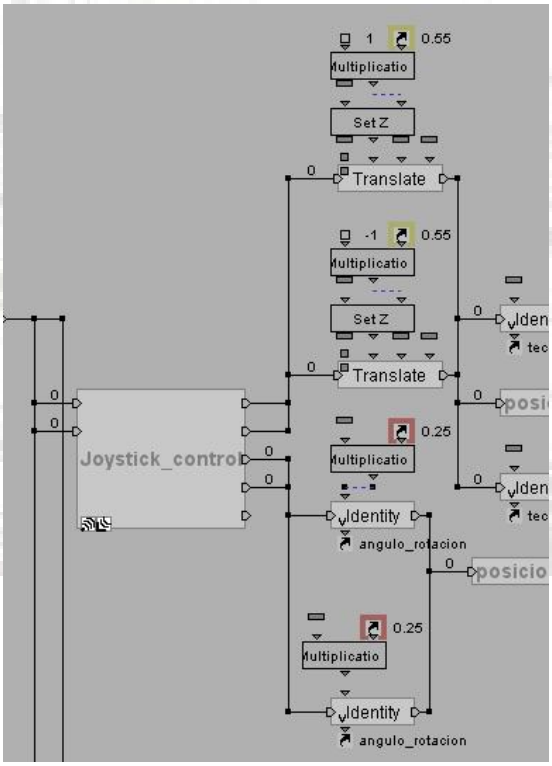


Figura 46. Control de joystick.

En el proyecto desarrollado, el joystick tiene un papel fundamental en el desarrollo de las características del mismo, como en este caso donde se muestra la relación ligada entre el movimiento del personaje en el plano 3D con el punto de ubicación dentro del mapa, para hacerlo en forma real y automática.

USO DE GUIONES

El desarrollo de guiones para la forma en que se espera se desenvuelva el proyecto, es importante pues ayuda a no perder la intención inicial que motivó la creación del mismo.

Se sugiere, para este efecto crear la carpeta:

…/Virtools/ documentacion/guion/

Donde guardará en archivos de texto, los guiones y las adaptaciones que vayan surgiendo dentro del desarrollo del mismo. Tanto para el desarrollador en jefe como para el equipo de trabajo.

USO DE ARCHIVOS PARA ANALISIS Y DISEÑO

Dentro del desarrollo del proyecto se pueden encontrar "vacíos" que necesitamos llenar de forma independiente.

Por ejemplo las formas preestablecidas por el sistema para caminar o correr, podrían no corresponder a la idea original que esperamos de nuestro personaje. Así, debemos elaborar nuestros propios sistemas, a fin que el resultado sea lo más cercano a lo que necesitamos.

Recomiendo utilizar un par de carpetas para organizar esta información de investigación, para tener un mejor control de los archivos que durante el desarrollo del proyecto vamos necesitando.

Para las imágenes de movimiento que podemos necesitar para sustituir las preestablecidas, para crear nuevas animaciones o movimientos que no tenga el sistema establecido, haremos una investigación de videos donde se muestren los movimientos deseados. Utilizaremos la carpeta escrita a continuación para archivar esta información:

…/Virtools/ documentacion/imagenes_movimiento/

Para las imágenes de referencia útiles para cubrir los detalles del proyecto, utilizaremos la dirección:

…/Virtools/ documentacion/imagenes_referencia/

8.4 Crear versión ejecutable.

Una vez que se desarrolla un proyecto, debemos considerar que el lugar donde se va a ejecutar o mostrar no necesariamente debe tener instalados todos los programas que requerimos para desarrollarlo, probarlo y ejecutarlo. Por lo que es más conveniente crear un archivo .exe mediante el cual trasladaremos el sistema desarrollado de un lugar a otro sin tanto problema. Como nota: los archivos ejecutables sólo son compatibles con Windows.

La propuesta es utilizar el 3DVIA player para ver las aplicaciones interactivas 3D a través del web browser: Internet Explorer, Firefox, Mozilla, Netscape. Con el sistema operativo: Microsoft Windows (2000, XP o Vista)

Por el tiempo de desarrollo de este proyecto, no se hicieron pruebas al respecto.

Otro método sencillo de visualizar un proyecto en virtools, es guardarlo con extensión <vmo> y verlos en cualquier navegador. En caso que no lo "cargue", pues hay que "bajar" el plugin mencionado anteriormente.

Para el proyecto de Tenochtitlan se creó un archivo con extensión .bat para recrear la ejecución automática del archivo con extensión .cmo que crea Virtools del proyecto.

Glosario y sugerencias.

SUGERENCIAS. TIPS ADICIONALES:

Cabe mencionar que el archivo VRpack.cfg es el archivo de configuración para que la aplicación funcione.

F2. Sirve para cambiar el nombre de cualquier elemento dentro del sistema Virtools.

Para agregar más elementos (llamados BUILDING BLOCKS) a la programación de Scripts basta con arrastrarlos con el Mouse al lugar donde se desea.

Si da doble click sobre el elemento se abre un texto de ayuda, sobre como usarlo o características del mismo.

Si el elemento tiene salidas (como en el caso de los controladores, i. e. KEYBOARD) basta con seleccionarlo y presionar < O > para agregar más salidas según se requieran.

Utilizamos < ALT > < L > para agregar parámetros de entrada.

Se utiliza un "sistema de colisión" para delimitar los objetos dentro del sistema y que éstos no sean atravesados por los objetos en movimiento como son las cámaras. Generalmente son objetos 2D que se usan como "bardas" para delimitar los objetos físicamente en forma virtual. Únicamente se les atribuyen características de objeto de colisión para que los objetos en movimiento no atraviesen como haría un fantasma, los objetos del entorno.

En caso necesario se pueden generar scripts propios con el fin de definir un detalle deseado en especial. Se utiliza para simplificar métodos dentro de la aplicación. Una vez desarrollado el procedimiento deseado, se presiona < G > y sin soltar, se define un recuadro con el Mouse de los elementos que se desean incluir en la función.

Para abrir/ cerrar éstas funciones basta con < doble click > sobre ellas.

Podemos incluir parámetros de operación (suma, resta, etc) estando en el área

SCHEMATICS y tecleando al mismo tiempo < ALT > < P >

Para utilizar parámetros es necesario el *building block* de transferencia < *identity* >, estando en el área de trabajo se agregan con < CTRL > <doble click botón izquierdo del Mouse >

Menús de elementos:

a) transformaciones > 3D > básico > ROTATE / TRANSLATE (rotación y traslación)
b) controllers (controladores 3D) > JOYSTICK / TECLADO / MOUSE / OTROS
c) building blocks / collisions (definen un grupo de objetos)
d) logics / calculator / identity
e) condicional (IF) / logics / test / test

Glosario de imágenes.

Bibliografía Internet.

http://csdl2.computer.org/comp/mags/cg/1983/06/04037618.pdf
http://jgarzas.googlepages.com/GARZASTesisDesignKnowledge.pdf
http://es.wikipedia.org/wiki/Triangulaci%C3%B3n_de_Delaunay
www.monografías.com
http://www.users.red3i.es/~stereoweb/historia.htm
http://www.sc.ehu.es/ccwgamoa/clases
http://es.wikipedia.org/wiki/Augustin_Fresnel
www.chromatek.com

Bibliografía.

- ✓ "Gráficos por computadora", HEARN Donald – BAKER Pauline, Prentice Hall, 2ª. Ed., México, 1995, [tr. José Julián Díaz Díaz].
- ✓ Manual 3DMAX Studio.
- ✓ "Stereoscopy", N. A. Valyus. Focal Press (Londres) 1966.
- ✓ "Stereo graphics. Developers Handbook", StereoGraphics Corporation, 1997.
- ✓ "Foundations of the Stereoscopic Cinema", Lenny Lipton. Van Nostrand Reinhold, 1982.
- ✓ "The CrystalEyes Handbook. Theory and methods for developing high-quality stereo images.", Lenny Lipton, de StereoGraphics Corporation. San Rafael, California.
- ✓ "Principles of Stereoscopy", Herbert McCay's.
  Primera edición, 1948. Segunda edición, 1951 con el título "Three-Dimensional Photography"
- ✓ "The theory of Stereoscopic transmission and its application to the motion picture" , Raymond and Nigel Spottiswoode's. University of California Press, 1953.
- ✓ "Introduction to 3D", H. Dewhurst's. The Macmillian Co., NY 1954.

- ✓ "Stereoscopic Cameras", K.C.M. Symons.
- ✓ "Stereo Computer Graphics and Other True 3D Technologies", David F. McAllister.
- ✓ "Stereo Photography", Fritz G. Waack de la German Stereoscopic Society, 1979.
- ✓ "Special Effects. High Tech Filmmaking", David Hutchison. Starlog Photo Guidebook Vol. 3. Starlog Press (New York) 1981.
- ✓ Science&Vie. "La photo et les images synthétiques". (Número fuera de serie), Diciembre 1984.
- ✓ "Binocular Vision", K. N. Ogle, W.B. Saunders Co., Philadelphia, 1950.