Diseño asistido por computador

Proyecto

Antonio Pérez Contreras
Grupo : Miercoles 18-20

Indice

1. Introducción

Mi proyecto consiste en un juego simulador de un tanque que puede derribar los edificios de una ciudad disparando contra ellos. El objetivo principal era simular de forma realista, dentro de lo que la complejidad computacional permitiera, un sistema físico de objetos con propiedades tales como peso, velocidad... cuyo comportamiento se asemejara en lo posible al comportamiento de las mismas en el mundo real. Esta representación en forma de simulador es, a mi juicio, más interesante que por ejemplo otra cuya meta fuera recrear la realidad mediante un entorno visualmente espectacular ya que por un lado esta última no permitiría una interacción con el mundo tan grande como la primera y por otra parte y más importante no creo que la situación permita llegar a una calidad de imagen lo más mínimamente fotorealista de lo que se pretende y que el software utilizado sea el mejor para llegar a ello. Aparte de todo ello la visión elegida da un mejor ejemplo de un programa computacionalmente complejo aplicado a un programa de diseño asistido por computador.

2. Resolución del problema

La herramienta elegida ha sido el lenguaje de programación OpenGL porque permite el manejo de herramientas de visualización 2D y sobre todo 3D necesarias para llevar a cabo el problema permitiendo además rapidez a la hora de representarlo y compatibilidad.

Como hemos comentado antes el problema consiste en un tanque que puede moverse a través de los edificios de una ciudad pudiendo dispararles por lo tanto se requiere una representación de los siguientes elementos :

- Tanque
- Edificios
- Ciudad como conjunto de edificios con el tanque en mitad de ellos
- Sistema de proyección

Esos son los elementos que se consideran de una representación digamos superficial de problema, si solo quisiéramos mostrar un entorno estático bastaría con lo anterior, pero como hemos comentado antes no es ese nuestro objetivo, necesitamos de otros elementos más implícitos en la naturaleza del problema como son:

- Movimientos del tanque y bala
- Física tanto del tanque, como de los edificios , la bala y la ciudad

Precisamente los primeros elementos corresponderían con la parte de representación mediante el uso de elementos gráficos y los segundos mediante el calculo computacional aplicado a transformar los primeros.

Pasemos ahora a explicar los segundos elementos ya que los otros se explicaran dentro de la sección correspondiente a la forma en que se han implementado.

Un sistema físico

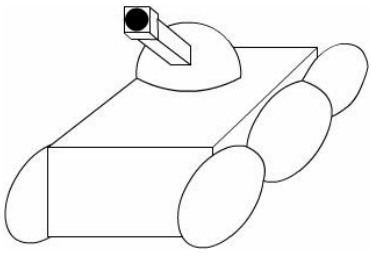
Suponiendo que tenemos el mundo, llamémoslo lógico, completamente a nuestra disposición con sus elementos disponibles para su utilización, tenemos que desarrollar la forma de interaccionar entre ellos.

Lo primero de todos es establecer un punto de referencia en nuestro mundo lógico que represente el punto de referencia del mundo real y las dimensiones de movimiento (evidentemente el mundo lógico que estamos creando es infinitamente más simple que el real donde no existe ningún punto físico de referencia) este se corresponderá con el punto (0,0,0) y las dimensiones con los ejes x,y,z . Nuestros elementos se moverán a lo largo de estos 3 ejes a partir del punto de referencia.

Nuestro primer elemento **el tanque**, necesita una forma de moverse mediante los ejes, suponiendo que nuestro tanque no puede volar limitamos los movimientos a los ejes x,z . Trasladando el origen del tanque a las posiciones deseadas conseguiremos moverlo, esta translación se supone lleva una dirección intrínseca a los ejes, por ejemplo, moverse al noroeste

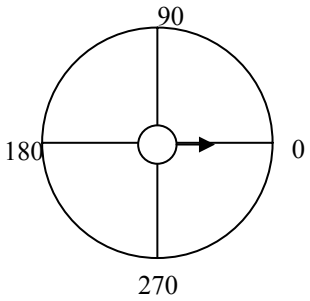
supone una translación positiva en el eje z y otra negativa en el x, vemos como poder calcular la posición del tanque en un momento dado. La forma de implementar todo esto es la siguiente :

Nuestro tanque es así :



pero se representa con :

que indica el centro del mismo y la dirección que lleva. Esta dirección toma como valores los mismos que los grados de rotación en una circunferencia, es decir :



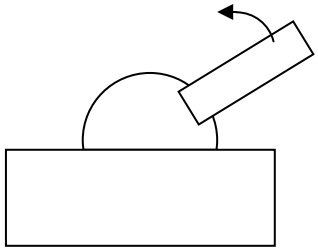
Por propiedades trigonométricas sabremos que un tanque se desplazara en el eje x mediante la formula :

$$x = x + \cos(dirección)$$

y en el eje z mediante la formula :

$$z = z + \cos(dirección)$$

Profundizando un poco más nuestro se mueve sobre 3 puntos, la base del tanque, la torreta del tanque y el cañon del tanque, el primero y el segundo vamos a considerarlos igual, no se permite mover la torreta sobre la base, el cañon sin embargo esta libre y además la rotación del tanque no se hace respecto el eje y como la base sino que se hace sobre el eje x, es decir :



En este caso utilizaremos las mismas consideraciones que la rotación de la base ya que al fin y al cabo la dirección donde apunta el cañon no hara falta para mover el tanque.

La bala de cañon es muy parecida al tanque, también tiene las misma dirección que el tanque en el momento de dispararla, es decir, si el tanque tiene dirección 90 al disparar la bola seguira la misma dirección. La diferencia aquí es que el movimiento de la bola también depende de la inclinación del cañon que hemos visto antes, en este caso si el estuviera completamente vertical tendría dirección 90 y al disparar la bola saldría hacia arriba y no tendria movimiento en los ejes x,z, la formula de moviento de ambas coordenadas sera :

$$x = (x + \cos(direccion))\cdot\cos(inclinacion\_cañon)$$
$$z = (z + sin(direccion))\cdot\cos(inclinacion\_cañon)$$

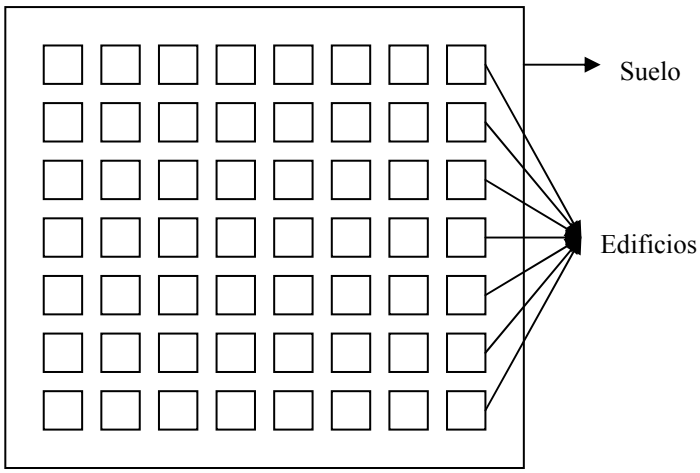
Además de esto la bola sigue el movimiento de una parábola por lo que también se mueve sobre el eje y que dependerá del peso de la bola y por lo tanto de la velocidad que lleve, ademas de depender de la inclinación del cañón, si esta es completamente horizontal apenas se moverá en el eje y, la formula para la coordenada es :

$$y = y + \cos(\text{inclinacion\_cañon})\cdot \text{velocidad}$$

Los edificios son distintos, solamente se moverán cuando estén cayendo y no será un movimiento de translación sino de rotación, esta depende del grado de inclinación del edificio que aumentara conforme al tiempo cuando este cayendo y de la dirección de caída del edificio, cuando una bala choca contra un edificio este caerá en la misma dirección de la bala que le derribo y que sigue el mismo modelo de referencias que esta . Las formulas para hallar el ángulo de rotación en cada eje son :

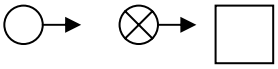
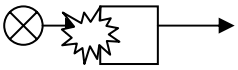
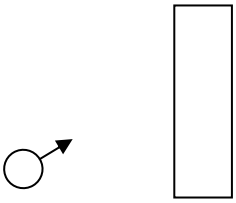
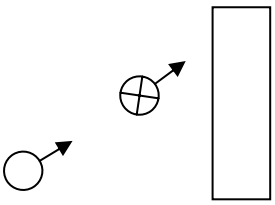
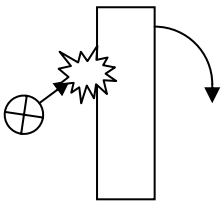
$$x = \text{inclinación} * \sin(\text{dirección})$$
$$z = \text{inclinación} * \cos(\text{dirección})$$

La ciudad esta compuesta de un número de edificios y un tanque colocados sobre el suelo. La forma de representar internamente nuestra ciudad se hace mediante una matriz en la cual cada elemento indica un edificio y en su posición , la distancias entre edificios están prefijadas lo que hará que la ciudad tenga un aspecto de parrilla, vista desde arriba sería :



La posición del tanque en la ciudad no es relevante, siempre y cuando esta no se superponga con la de cualquier edificio. Este esquema de parrilla nos sirve además para algo muy importante que veremos más adelante, es la colisión entre objetos, resultaría muy costoso comprobar si una posición x,y,z esta dentro de los límites de todos los objetos del escenario por lo que como las direcciones entre edificios son prefijadas sabemos todas y cada una de las coordenadas de los edificios de forma anticipada y podremos orientarnos sobre el elemento al que corresponde.

En resumen el funcionamiento del programa consistiría en :

X,Z	1. El tanque lleva se mueve a lo largo de una dirección y tiene una inclinación del cañón 	2. El tanque dispara una bola de cañón que sigue la misma dirección 	3. Al chocar contra el edificio se cae en la misma dirección de la bala 
X,Y			

Ahora vamos a ver como implementamos estos elementos en lenguaje básico de programación.

3. Estructuras de datos

Para poder implementar lo hasta ahora comentado he utilizado unas estructuras de datos propias que paso a comentar :

Tank	Esta es la estructura de datos que representa el tanque. Consta de lo siguiente :
grados	Dirección de inclinación del cañón (radianes)
rbase	Dirección de la base del tanque
rtorre	Dirección de la torreta
rcanon	Dirección de inclinación del cañon (grados)
posz	Posición en Z del tanque
posx	Posición en Y del tanque

La posición en Y esta prefijada y no hace falta guardarla.

Ball	Esta es la estructura de datos que representa la bola de cañon. Consta de lo siguiente :
X	Posición en X
Y	Posición en Y
Z	Posición en Z
Direc	Dirección de movimiento
Grados	Grados de inclinación
Vel	Velocidad de la bala
Mostrar	Flag que indica si se debe dibujar o no

Building	Esta es la estructura de datos que representa un edificio. Consta de lo siguiente :
X	Posición en X
Y	Posición en Y
Z	Posición en Z
H	Altura del edificio
Med	Como los edificios son cuadrados, la mitad del ancho
Direc	Dirección de movimiento
Grados	Grados de inclinación de la caída
Mostrar	Flag que indica si se debe dibujar o no
Tipo	Indica el tipo de edificio que es
rompiendo	Flag que indica si el edificio ha sido destruido
cayendo	Flag que indica si el edificio ha sido derribado
Factor_roto	Indica como de destruido esta el edificio

La posición en Y esta prefijada y no hace falta guardarla.

Para los edificios, como hay 100 en total , se guardan en un vector del mismo tamaño y que actúa como la matriz de posiciones que comentábamos en el apartado anterior, la transformación de una posición i,j de una matriz 10x10 a una sola posición i de un vector de 100 elementos es bastante simple.

El tipo de objeto mayormente utilizado para la mayoría de las variables es flotante dado que nos da un gran grado de precisión muy grande. De haber usado enteros tendríamos que haber creado un escenario gigante para poder llegar una buena suavidad de movimiento, además a la hora de escalar este problema se agravaría.

4. Manejo de primitivas

Para modelar objetos he utilizado las siguientes primitivas incluidas en la librería glut.h:

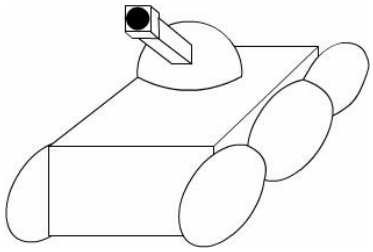
- Esferas
- Toros
- Cuadriláteros
- Tira de cuadriláteros
- Triángulos

Otras primitivas han sido implementadas a partir de primitivas geométricas como son :

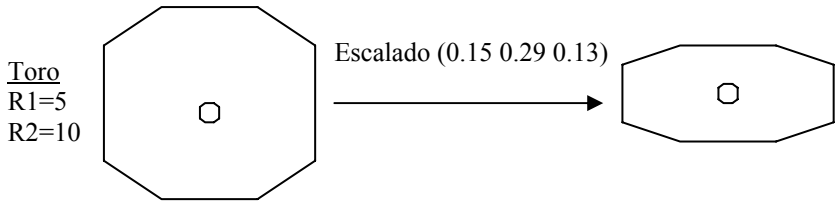
- Cubos
- Pirámides
- Cuñas

4. Manejo de jerarquías

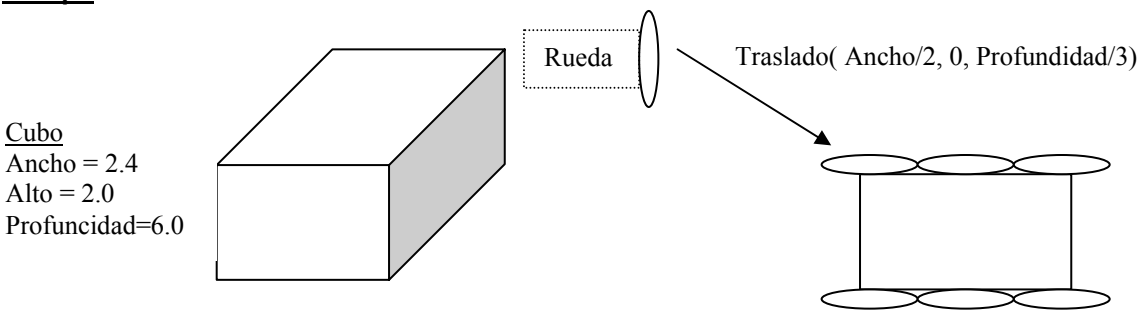
Muchos de los objetos modelados , exactamente los edificios y el tanque, utiliza una estructura jerárquica para dar forma, ahora explicare la jerarquía del tanque que es la más compleja de todas las utilizadas, para su implementación he utilizado “Display List”. Como he mostrado antes el modelo del tanque consiste en :



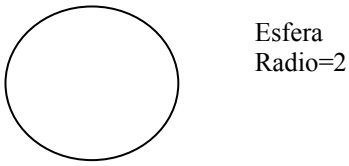
Rueda



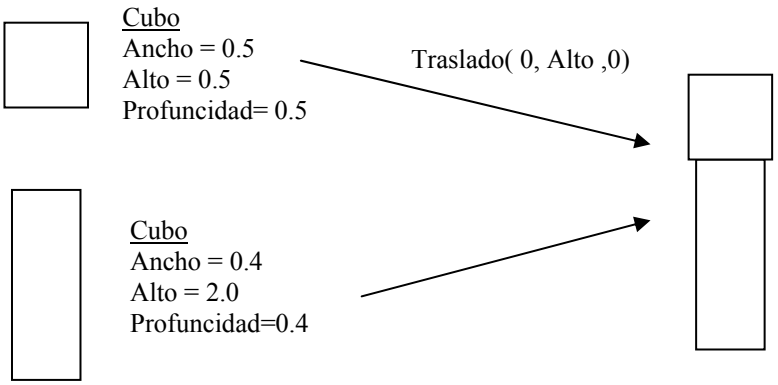
Cuerpo



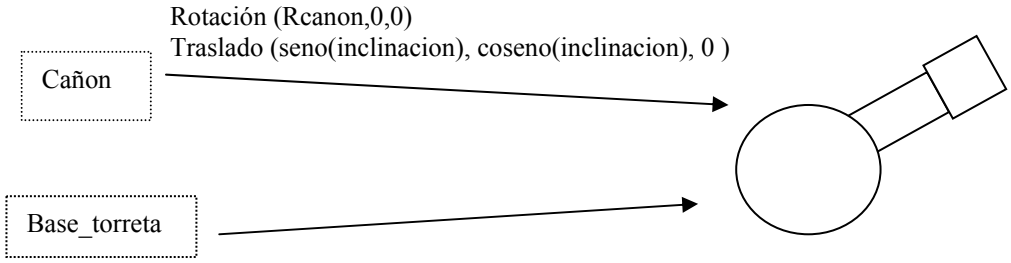
Base_torreta



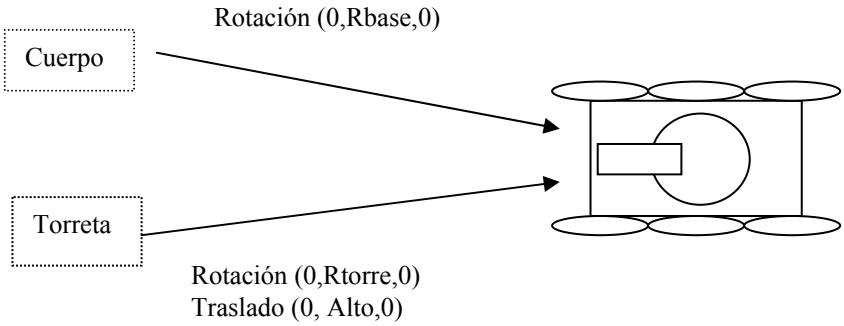
Cañon



Torreta



Tanque



5. Interacción

En este apartado hablaremos de las funciones del ratón sobre el programa. La llamada a la función *Pick* devuelve –1 si el "click" no ha seleccionado ningún objeto o bien el identificador de objeto.

El número de "Hits" siempre será 2 o 1 pues la selección se realiza con la camara situada justo encima del escenario y mirando hacia el eje negativo de Y por lo tanto y teniendo en cuenta que no hay más de dos objetos superpuestos renderizando con esta vista podremos asegurar que si el número de "Hits" es 1 se ha seleccionado el suelo y si es 2 entonces se ha pinchado sobre un edificio o el tanque.

6. Colisiones

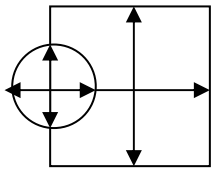
Probablemente la parte de la práctica más importante y mas difícil de realizar es el choque entre objetos. Como hemos comentado desde el principio el objetivo es que el tanque derribe edificios con una bala y por tanto debe haber una colisión entre ellas, al igual también tenemos que detectar colisiones entre el tanque y los edificios y entre los edificios entre sí.

La ciudad vimos antes que está estructurada como una matriz de edificios con una distancia prefijada entre cada uno de ellos, es decir que sabemos la posición del centro de un edificio si sabemos la del anterior y como sabemos el que el primero esta sobre el origen de coordenadas podemos averiguar la posición de todos los edificios, también sabemos el tamaño de cada edificio y por lo tanto las dimensiones que ocupa.

Una colisión entre objetos se considera cuando cualquier punto x,y,z coincide con el punto x,y,z de otro objeto, pues por ejemplo, como sabemos el centro de la bala y además sabemos el espacio de puntos que ocupa cada edificio consideraremos que hay un choque cuando :

El punto X de la bala se encuentre entre :

- Entre la coordenada edificio.x – edificio.mitad y edificio.x + edificio.mitad

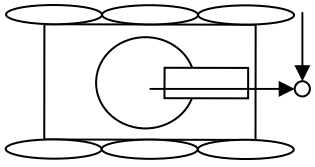


El punto Z analógamente deberá estar entre :

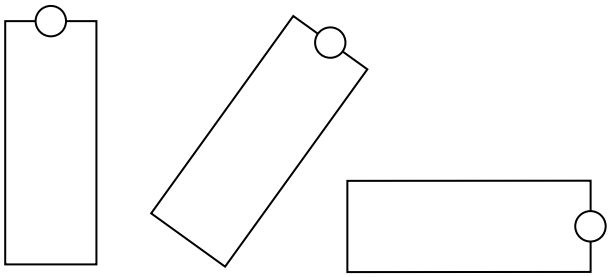
- Entre la coordenada edificio.z – edificio.mitad y edificio.z + edificio.mitad

La coordenada Y es despreciable, la bala no es llega tan alto como para sobrepasar al edificio.

Con la misma filosofía se modela el choque entre edificios y el del tanque con un edificio solo que en cada caso el punto de choque x,y,z considerado es distinto. En el tanque se encuentra en el origen del tanque + la mitad de su tamaño :



Sin embargo el punto considerado del edificio es más complicado porque la colisión se detectara cuando el edificio este cayendo, cuando está parado evidentemente no puede chocar con otro edificio, es por ese motivo que el punto se mueve junto a la caída y además depende de la altura. Por ejemplo :



Como se observa el punto es móvil, este se calcula mediante la expresión :

$$x = sin(inclinación) * \cos(direccion) * altura$$
$$z = sin(inclinación) * sin(direccion) * altura$$

9. Manual de usuario

Ya hemos visto como funciona internamente el programa, ahora explicaremos como utilizarlo.

El programa tiene 3 modos de manejo :

- Vista interior : sirve para manejar el tanque visto desde la perspectiva del tanque. En este caso la cámara se maneja a la vez que se mueve el tanque, no se tiene control sobre ella independientemente del movimiento del tanque.

- Cámara libre : este modo sirve para navegar por el escenario sin estar limitado al movimiento del tanque, con este modo se puede ver el movimiento del tanque pero desde fuera.

- Selección objetos : este modo sirve para destruir edificios y para seleccionar la rotación del tanque más rápidamente y con mayor precisión. La perspectiva es aérea lo que significa que vera la ciudad y el tanque desde arriba.

Las teclas para manejarlo funcionaran o no y sus funciones dependieran del mondo en que se este utilizando. Estas son :

- Teclas validas para los 3 modos :

    - *ESC* : sale del programa
    - *Botón derecho del ratón* : muestra el menú de opciones
    - *Espacio* : el tanque dispara una bola de cañón

- Validas en el modo de vista interna :

    - *Arriba y abajo* : hacen avanzar/retroceder al tanque y a la cámara
    - *Izquierda y derecha* : giran la base del tanque y la cámara
    - *Pagina arriba y abajo* : inclinan el cañón del tanque

- Validas en el modo de cámara libre :

    - *Arriba y abajo* : hacen avanzar/retroceder al tanque
    - *Izquierda y derecha* : giran la base del tanque
    - *Pagina arriba y abajo* : inclinan el cañón del tanque
    - *+ y - :* hacen avanzar/retroceder la cámara
    - *Movimiento de ratón : permite girar la cámara en cualquier eje*

- Validas en el modo de selección objetos :

    - *Arriba y abajo* : desplaza la cámara sobre el eje z
    - *Izquierda y derecha* : desplaza la cámara sobre el eje x
    - *Botón izquierdo del ratón :* selección edificios y tanque.
    - *Movimiento de ratón :* si se ha seleccionado el tanque este puede ser rotado moviendo el ratón. Para deseleccionarlo vuelva a hacer click con el botón izquierdo.

Fallos conocidos

- Existe un fallo que depende de la perspectiva sobre la que se dispare a un edificio en el que no ocurrirá nada. Para solucionar esto rote un poco el tanque y/o desplácese hacia adelanto o atrás.

- En muchas ocasiones la caída del edificio no corresponde completamente con la dirección del impacto y puede incluso variar un poco la ruta dando lugar al choque entre edificios que no se detecta.