Proyecciones

Tras las transformaciones homogéneas para nuestra geometría ya somos capaces de modelar y situar objetos a lo largo y ancho de nuestra escena virtual. ¿Pero cómo hacer que aparezcan en nuestra ventana del Sistema Operativo? La ventana es 2D y en cambio está mostrando geometria 3D dando sensación de que hay profundidad.

Tenemos que proyectar la geometría en un plano de proyección. Como plano es obviamente 2D y suele situarse por convención en Z = 0 (plano XY). La idea es proyectar primero para "discretizar" después. Por discretizar entendemos pasar del mundo real (float o double) al entero (integer) que es el de los píxels de nuestro monitor.

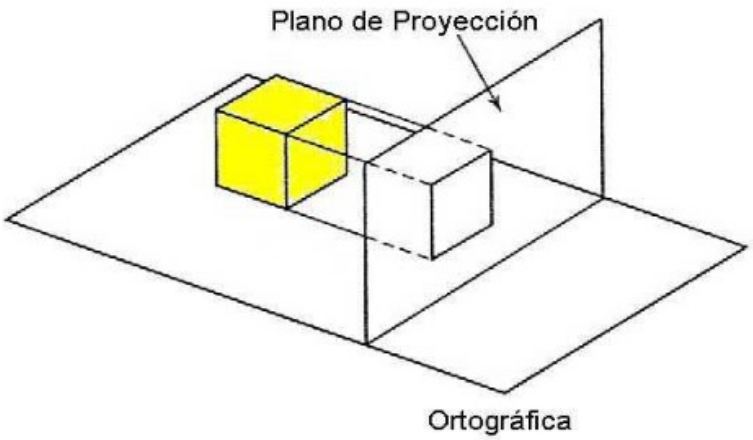
Tipos de proyección
No comentaremos todas las existentes, no será necesario. Nos centraremos en las proyecciones planares. En éstas se define una dirección de visión (viewing) que va desde el observador hasta el objeto en cuestión. Esta dirección se establece por medio de proyectores (líneas) que cortan el plano de proyección generando así la imagen 2D que finalmente aparecerá en pantalla.

Como primera gran clasificación de las proyecciones planares podemos hablar de:

- ·Proyecciones planares
  - ·Paralelas
  - ·Oblicua
  - ·Ortográfica
- ·Perspectiva
  - ·1 punto
  - ·2 puntos
  - ·3 puntos

Nosotros nos centraremos en dos de es tas proyecciones. La ortográfica y la perspectiva de un sólo punto.

Proyección ortográfica
Como proyección paralela que es, cuenta con proyectores paralelos entre ellos. El centro de proyección (COP) se encuentra en el infinito. En el caso de la ortográfica, los proyectores son perpendiculares al plano de proyección. Lo observamos mejor en la figura:



Este tipo de proyección no preserva las dimensiones reales de los objetos según la distancia hasta ellos. Es decir, que si nos acercamos o alejamos de ellos no se producen cambios de tamaño, con lo cual el realismo no es total. Se utiliza tradicionalmente en proyectos de ingeniería del tipo de programas CAD/CAM.

Los parámetros a especificar son las dimensiones de la caja (Xmin, Xmax, Ymin, Ymax, Zmin, Zmax). A los valores MAX y MIN también se les denomina FAR o BACK y NEAR o FRONT.

En OpenGL la podemos definir de la siguiente forma:

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(x_min, x_max, y_min, y_max, z_min, z_max);
```

la última función puede reemplazarse por:

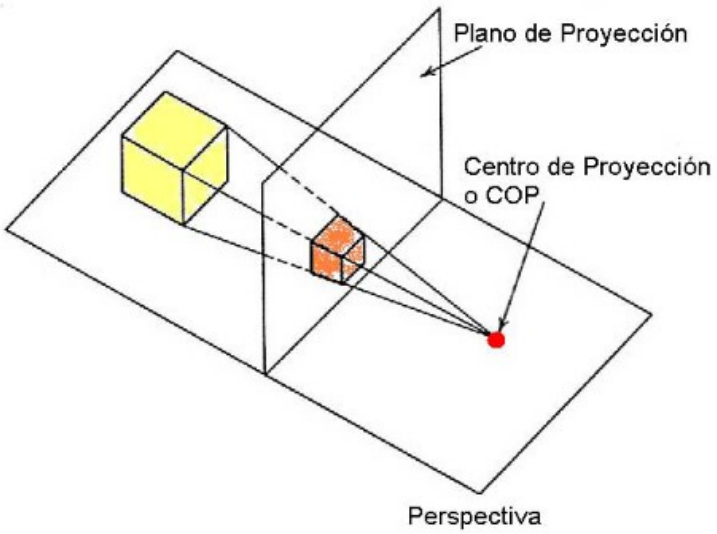
```
gluOrtho2D(x_min, x_max, y_min, y_max);
```

si se aceptan los valores por defecto de Zmin = -1.0 y Zmax = 1.0.

Proyección perspectiva

Esta es la que utilizaremos para dotar del mayor realismo a nuestras aplicaciones. Las proyecciones perspectiva preservan las dimensiones reales de los objetos si nos acercamos / alejamos de ellos.

Por tanto el efecto visual es justo el que necesitamos en casos de apariencia real.



En la figura tenemos una proyección perspectiva con un sólo COP o punto de fuga. Todos los proyectores emanan de él y se dirigen hasta el objeto intersectando el plano de proyección. Como podemos observar los proyectores no son paralelos entre ellos tal.

En OpenGL la podemos definir así:

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective( FOV en grados, Relación de Aspecto, z_near, z_far);
```

Los parámetros que tenemos que especificar son:

· FOV en grados. Es el "field of view" o campo visual. Se refiere al ángulo de abertura vertical.
· Relación de Aspecto o "aspect ratio". Es el cociente entre la anchura (width) y la altura (height) del plano de proyección deseado.
· Los valores NEAR y FAR del volúmen de visualización perspectiva. Iguan que en ortográfica.

Podemos definir también una proyección perspectiva usando la función:

```
glFrustrum(x_min, x_max, y_min, y_max, z_min, z_max);
```

En este caso OpenGL calculará el "frustrum piramidal", es decir, el volúmen de visualización perspectiva más idóneo. Son simplemente dos maneras distintas de acabar creando lo mismo. A cada uno con su elección, la que les parezca más intuitiva y cómoda.

Las distancias NEAR y FAR son siempre positivas y medidas desde el COP hasta esos planos, que serán obviamente paralelos al plano Z = 0. Dado que la cámara apunta por defecto en la dirección negativa de Z, el plano de front (near) estará realmente situado en z = -zmin mientras que el de back (far) estará en z = -zmax.

Hemos de pensar que no hay que proyectar toda la geometría existente sinó tan sólo la que vé la cámara desde su posición y orientación. Tenemos que acotar en este sentido y es por eso que además de definir un plano de proyección y una forma de proyectar, creamos un volúmen de visualización finito con unas fronteras bién marcadas. Todo aquello que no se encuentre dentro del volúmen será rechazado y no proyectado dado que no debe verse.

La Cámara

La cámara son nuestros ojos virtuales. Todo lo que ella vea será proyectado, discretizado y finalmente mostrado en nuestra ventana del sistema operativo. Podemos imaginar que de la cámara emana el volúmen de visualización de forma que se traslada con ella. Los parámetros a definir en cuanto a la cámara son:
Posición XYZ en el mundo 3D. Al igual que un objeto cualquiera, la cámara debe posicionarse. En un juego arcade 3D típico la cámara nos da la visión frontal del mundo y se va moviendo a nuestro antojo con las teclas o el ratón. Cada vez que modificamos la posición varían las coordenadas XYZ de cámara.

Orientación

. Una vez situada debe orientarse
. Yo puedo estar quieto pero girar la cabeza y mirar hacía donde me venga en gana.
La dirección de "AT". Define hacia dónde estoy mirando, a qué punto concretamente.

En OpenGL lo tenemos fácil con:

```
gluLookAt(eyeX, eyeY, eyeZ, atX, atY, atZ, upX, upY, upZ);
```

Esta es la función que determina dónde y cómo está dispuesta la cámara. La posición de la cámara no tiene nada que ver con la proyección que hayamos establecido.

La proyección se define sólo una vez, típicamente al principio del programa en una función inicializadora,

mientras que la cámara se mueve continuamente según nos interese. Añadir a esto que la matriz que se modifica al llamar a esta función no tiene que ser GL_PROJECTION sinó GL_MODELVIEW. OpenGL calculará todas las transformaciones que aplicará al mundo 3D para que manteniendo la cámara en el origen de coordenadas y enfocada en la dirección negativa de las Z's nos dé la sensación de que lo estamos viendo todo desde un cierto lugar. Para ello recordad siempre activar esta matriz antes de llamar a gluLookAt de esta forma:

```
glMatrixMode(GL_MODELVIEW);
```

En cuanto a los parámetros que demanda la función son los siguientes:

· Coordenadas del "eye"

. Es literalmente la posición XYZ dónde colocar la cámara dentro del mundo.

· Coordenadas del "at"

. Es el valor XYZ del punto al que queremos que mire la cámara. Un punto del mundo obviamente.

· Coordenadas del vector "up"

. Si, es un vector y no un punto. Con él regularemos la orientación de la cámara. Este ha de ser el vector que "mira hacia arriba" si entendemos que el vector que mira hacia delante es el que va del "eye" hasta el "at". Variando el "up" variamos la orientación: