Índice general

CAPÍTULO 1

Rendering

1.1. Introducción

Rendering es un término en inglés que denota el proceso de generar una imagen 2D a partir de un modelo digital 3D o un conjunto de ellos, a los que se les llama "escena". Puede ser comparado a tomar una foto o filmar una escena en la vida real.

1.2. ISGL3D

ISGL3D es un *framework* (marco de trabajo) para *iPad*, *iPhone* y *iPod touch* escrito en *Objective-C*, que sirve para crear escenas y *renderizarlas* de forma sencilla. Es un proyecto en código abierto y gratis. En su sitio web oficial: `www.isgl3d.com`, se puede descargar el código de ISGL3D y de forma sencilla este puede ser agregado como un complemento de *Xcode*. Además se pueden encontrar tutoriales, una *Application Programming Interface* (API) y un acceso a un grupo de *google* donde la comunidad pregunta y responde dudas propias y ajenas.

Cuando se crea una aplicación ISGL3D, el núcleo de la misma es la llamada "*view*" ("vista" en Español). Una *view* esta compuesta principalmente por una escena y una cámara:

- Una **escena** (Isgl3dScene3D) es donde los objetos o modelos 3D son agregados como nodos. Todos los nodos pueden ser tanto trasladados como rotados y pueden tener otros nodos hijos; los nodos hijos son trasladados y rotados con sus padres. Así como objetos 3D, se pueden agregar luces de distinto tipo, que generarán en la escena efectos de sombra que luego serán adecuadamente *renderizados* en función de dónde se encuentre y hacia dónde este mirando la cámara.
- Una **cámara** es utilizada para para visualizar la escena desde una posición y un ángulo en particular. La cámara se manipula como cualquier otro objeto o nodo en la escena, se puede trasladar, rotar y hasta indicar hacia dónde quiere uno que la cámara apunte. Es importante ajustar la cámara de manera que su arquitectura sea la que uno busca. Se pueden entonces ajustar ciertos parámetros intrínsecos a esta como por ejemplo su campo visual, su distancia focal, la altura y la anchura del plano imagen, etc.

Es importante entender que el llamado *render* se realiza sumando la información de la escena, objetos 3D y sus hijos, luces, etc.; más la información de dónde se encuentra la cámara, sus características y hacia dónde esta apunta.

Un particularidad de la cámara de ISGL3D es que el parámetro intrínseco "distancia focal" visto la sección **??**, no es directamente configurable. En cambio, el valor que sí se puede alterar es el llamado "FOV", acónimo de "Fiel Of View". El *field of view* de una cámara no es más que su campo visual, y se mide como la extensión angular máxima mapeable en el plano imagen, medida desde el centro óptico *C*. Puede ser medido de forma horizontal o de forma vertical; sin embargo, en ISGL3D es definido verticalmente. Ver figura 1.1.

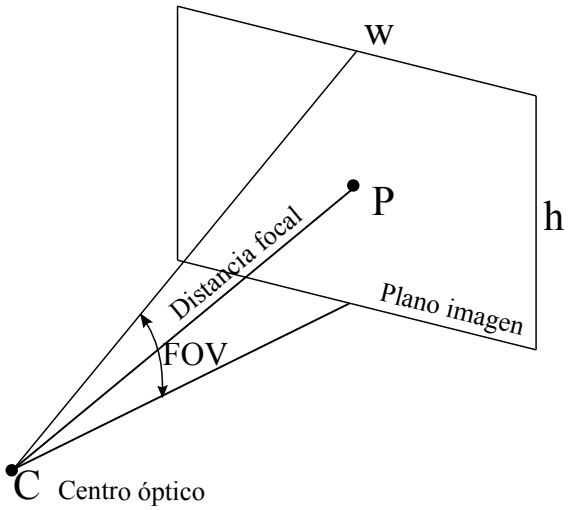

Figura 1.1: Definición gráfica del FOV.

Realizando algo de geometría se ve que la relación entre la distancia focal y el FOV es:

$$FOV = 2.arctg\left(\frac{h}{2.f}\right)$$

donde h denota la altura del plano imagen y f la distancia focal de la cámara.

ISGL3D cuanta con algunas estructuras primitivas que pueden ser usadas como modelos, o incluso combinadas de manera de formar modelos algo más complejos. Las principales estructuras primitivas de ISGL3D son:

- **Isgl3DArrow:** modelo correspondiente a una flecha. Tiene 4 parámetros configurables:
  - *headHeight:* altura de la punta.
  - *headRadius:* radio de la punta.
  - *height:* altura total de la flecha.
  - *radius:* radio de la base.
- **Isgl3DCone:** modelo correspondiente a un cono. Tiene 3 parámetros configurables:
  - *bottomRadius:* radio de la base inferior.
  - *height:* altura del cono.
  - *topRadius:* radio de la base superior.
- **Isgl3DCube:** modelo correspondiente a un cubo. Tiene 3 parámetros configurables:
  - *depth:* profundidad del cubo.

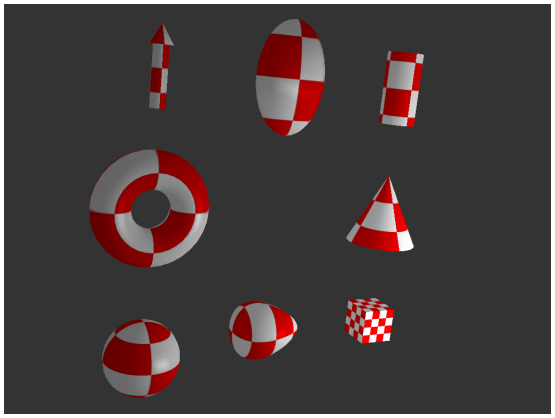

Figura 1.2: Principales primitivas en ISGL3D.

  - *height:* altura del cubo.
  - *width:* anchura del cubo.

- **Isgl3DCylinder:** modelo correspondiente a un cilindro. Tiene 3 parámetros configurables:
  - *height:* altura del cilindio.
  - *radius:* radio del cilindro.
  - *openEnded:* indica si el cilindro cuenta con sus extremos abiertos o no.
- **Isgl3DEllipsoid:** modelo correspondiente a una elipsoide. Cuenta con 3 parámetros configurables:
  - *radiusX:* radio de la elipsoide en la dirección x.
  - *radiusY:* radio de la elipsoide en la dirección y.
  - *radiusZ:* radio de la elipsoide en la dirección z.
- **Isgl3DOvoid:** modelo ovoide. Cuenta con 3 parámetros configurables:
  - *a:* radio del ovoide en la dirección x.
  - *b:* radio del ovoide en la dirección y.
  - *k:* factor que modifica la forma de la curva. Cuando toma el valor 0, el modelo se corresponde con el de una ellipsoide.
- **Isgl3DSphere:** modello correspondiente a una esfera. Tiene un único parámetro configurable:
  - *radius:* radio de la esfera.
- **Isgl3DTorus:** modelo correspondiente a un toroide. Cuenta con 2 parámetros configurables:
  - *radius:* radio desde el origen del toroide hasta el centro del tubo.
  - *tubeRadius:* radio del tubo del toroide.

Para cada primitiva, se debe especificar además, la cantidad de segmentos utilizados en las distintas dimensiones para su creación. Por detalles referirse a la API de ISGL3D en: `http://isgl3d.com/resources/api`. En la figura 1.2 se pueden ver todas las primitivas anteriores.

A veces lo que se quiere no es agregar a la escena una primitiva sino un modelo previamente creado. Los modelos son realizados en herramientas de creado y animación de gráficos 3D como por ejemplo *Blender*, *MeshLab*, *Autodesk Maya* o *Autodesk 3ds Max*. Luego deben ser exportados en un formato llamado *COLLADA*, acrónimo de "COLLAborative Design Activity", que sirve para el intercambio de contenido digital 3D entre distintas aplicaciones de modelado. Por su parte, ISGL3D permite importar modelos pero en un formato llamado *POD*. Se usó entonces, una aplicación llamada *Collada2POD* que lo que hace es convertir modelos tridimensionales en formato *COLLADA* al formato *POD*. *Collada2POD* puede ser descargada gratuitamente de la página oficial de *Imagination Technologies*, su desarrollador: `http://www.imgtec.com/`.

Una vez que se tiene al objeto 3D en el formato *POD*, este puede ser importado en ISGL3D de forma sencilla:

```
Isgl3dNode * _container = [self.scene createNode];

Isgl3dPODImporter * podImporter = [Isgl3dPODImporter podImporterWithFile:@‘‘modelo.pod’’];

Isgl3dSkeletonNode * _model = [_container createSkeletonNode];

[podImporter addMeshesToScene:_model];
```