LIBGDX Carga Modelos3D

UNIDADE 4: Carga de Modelos 3D

Sumario

Deseñando os nosos modelos 3D

Loxicamente os nosos modelos 3D non vai ser creados 'manualmente', se non utilizando programas gráficos.

Algún deles:

- Blender: http://www.blender.org/

    Moi completo pero tamén bastante complexo.

- Wings3D: http://www.wings3d.com

    Bastante máis sinxelo.

Nos imos usar o wings3d aínda que os alumnos podedes utilizar calquera que manexedes.

A continuación tedes un enlace con tres vídeos (e o códec de vídeo por se non o visualizades) de como se manexa o Wings3D e como podemos crear un obxecto 3D e asinarlle unha textura.

Enlace a vídeos explicativos Wings3D

Conversión de formatos en 3D

En canto os formatos gráficos que podemos cargar están, entre outros:

- obj: https://en.wikipedia.org/wiki/Wavefront_.obj_file
- md2: https://en.wikipedia.org/wiki/MD2_%28file_format%29
- fbx: http://en.wikipedia.org/wiki/FBX
- g3dj: Formato g3d en mode texto Json (se pode editar).
- g3db: Formato g3d en binario. É o mais rápido de cargar e o que menos ocupa.

Para pasar dun formato a outro podemos utilizar unha ferramenta: **fbx-conv**.

Podemos ver o código fonte en: https://github.com/libgdx/fbx-conv

Podemos descargar a versión compilada en: http://libgdx.badlogicgames.com/fbx-conv/

Información na wiki: https://github.com/libgdx/fbx-conv

O proceso para pasar dun obj a un g3db sería:

- Descargar a versión compilada do enlace anterior.
- Descomprimide e abride unha consola.
    - Se estades en Linux hai que copiar a librería **libfbxsdk.so** a /usr/lib.
    - Se estades en Windows debedes ter instalado o paquete VC 2010 Redistributable Package.
- Vos situades onde se atopan os executables e escolledes a versión correspondente ó voso S.O.:
    - fbx-conv-win32.exe: Windows
    - fbx-conv-lin64: Linux
    - fbx-conv-mac: Mac
- A orde para pasar por exemplo de obj a g3db seria:

```
fbx-conv-win32.exe modelo_entrada.obj modelo_saida.g3db
```

NOTA: Sempre que queiramos acelerar a carga dos modelos deberemos de pasalos a binario.

Carga de Modelos 3D

Carga Modelos obj

Para cargar un modelo obj temos que facer uso da clase ModelLoader.

O proceso sería:

- Crear unha instancia de dita clase:

```
ModelLoader loader = new ObjLoader();
```

- Chamar ó método loadModel:

```
Model model = loader.loadModel(Gdx.files.internal("modelos/ship.obj"));
```

Dito método espera recibir como parámetro o arquivo obj.

Este método devolve un obxecto da clase Model.

Para nos un Model ven ser como un Mesh, o que sucede é que nun model poden ser cargados moitos Mesh de vez, polo que para ter unha referencia a un concreto faremos o seguinte:

- Cargamos o Mesh:

```
private Mesh meshNave;
        ............
        meshNave = model.meshes.get(0);
```

No noso caso só temos un Mesh no obj.

Exemplo:

Imos ver un exemplo completo, engadindo un modelo dunha nave e movéndoa cara a cámara, facendo que a cámara siga a nave cando a pase.

Preparación: Crear unha clase de nome UD4_6_CargaModelos3D, que derive da clase Game e sexa chamada pola clase principal das diferentes versións (desktop, android,...).

Crea un cartafol de nome **modelos** dentro do cartafol assets da versión Android.

Leva ó cartafol **assets/modelos** os arquivos ship.mtl, ship.obj e ship.png que se atopan no seguinte arquivo comprimido:

Media:Ship.zip

Creamos unha clase Elementos3D que garde a información de calquera elemento 3D. É igual á clase Cubo que utilizamos no exercicio anterior:

Código da clase Elementos3D
Obxectivo: Definir unha clase que garde o que necesitamos dun obxecto 3d.

```
import com.badlogic.gdx.math.Matrix4;
import com.badlogic.gdx.math.Vector3;

public class Elemento3D {

public Matrix4 matriz;
public Vector3 posicion;
public float escala;
public Vector3 velocidade;
private Vector3 temp;

public Elemento3D(Vector3 pos, float escala,Vector3 velocidade){
matriz = new Matrix4();
posicion = pos;
this.escala=escala;
this.velocidade = velocidade;

temp = new Vector3();

}

public void update(float delta){

temp.set(velocidade);
posicion.add(temp.scl(delta));


matriz.idt();
matriz.translate(posicion);
matriz.scl(escala);


}

}
```

Agora imos ver o código de carga da nave.

```
import com.badlogic.gdx.Game;
import com.badlogic.gdx.Gdx;
import com.badlogic.gdx.assets.loaders.ModelLoader;
import com.badlogic.gdx.files.FileHandle;
import com.badlogic.gdx.graphics.GL20;
import com.badlogic.gdx.graphics.Mesh;
import com.badlogic.gdx.graphics.PerspectiveCamera;
```

```
import com.badlogic.gdx.graphics.Texture;
import com.badlogic.gdx.graphics.g3d.Model;
import com.badlogic.gdx.graphics.g3d.loader.ObjLoader;
import com.badlogic.gdx.graphics.glutils.ShaderProgram;
import com.badlogic.gdx.math.Vector3;

/**
 * Cargando modelos 3D
 * @author ANGEL
 */

public class UD4_6_CargaModelos3D extends Game {

private Mesh meshNave;
private Elemento3D nave;
private ShaderProgram shaderProgram;

private Texture textura;
private PerspectiveCamera camara3d;



@Override
public void create() {
// TODO Auto-generated method stub

shaderProgram = new ShaderProgram(Gdx.files.internal("vertex.vert"), Gdx.files.internal("fragment.frag"));
if (shaderProgram.isCompiled() == false) {
Gdx.app.log("ShaderError", shaderProgram.getLog());
System.exit(0);
}


ModelLoader loader = new ObjLoader();
        Model model = loader.loadModel(Gdx.files.internal("modelos/ship.obj"));
        meshNave = model.meshes.get(0);

FileHandle imageFileHandle = Gdx.files.internal("modelos/ship.png");
textura = new Texture(imageFileHandle);

camara3d = new PerspectiveCamera();
nave = new Elemento3D(new Vector3(0,-1f,-20f), 2f, new Vector3(0.1f,0.1f,2f));

}


@Override
public void render() {

Gdx.gl20.glClearColor(0f, 0f, 0f, 1f);
Gdx.gl20.glClear(GL20.GL_COLOR_BUFFER_BIT|GL20.GL_DEPTH_BUFFER_BIT);

Gdx.gl20.glEnable(GL20.GL_DEPTH_TEST);

nave.update(Gdx.graphics.getDeltaTime());

camara3d.lookAt(nave.posicion);
camara3d.update();

shaderProgram.begin();
textura.bind(0);
shaderProgram.setUniformi("u_texture", 0);
shaderProgram.setUniformMatrix("u_worldView", camara3d.combined.cpy().mul(nave.matriz));
meshNave.render(shaderProgram, GL20.GL_TRIANGLES);



shaderProgram.end();

Gdx.gl20.glDisable(GL20.GL_DEPTH_TEST);

}
```

```
@Override
public void resize (int width,int height){
// Definimos os parámetros da cámara
float aspectRatio = (float) width / (float) height;
camara3d.viewportWidth=aspectRatio*1f;
camara3d.viewportHeight=1f;
camara3d.far=1000f;
camara3d.near=0.1f;
camara3d.lookAt(0,0,0);
camara3d.position.set(0f,0f,5f);
camara3d.update();
}

@Override
public void dispose(){
shaderProgram.dispose();
meshNave.dispose();

}

}
```

- Liña 21: Definimos o obxecto Mesh que vai representar a nave.
- Liñas 41-43: Cargamos o obj utilizando a clase ModelLoader.
- Liña 49: Instanciamos a nave pasándolle uns datos concretos.
- Liñas 64-65: Facemos que a cámara mire cara a nave. LEMBRAR CHAMAR Ó UPDATE.
- Liña 71: Renderizamos a nave.
- Liña 97: Liberamos a memoria.

TAREFA 4.5 A FACER: Esta parte está asociada á realización dunha tarefa.

Carga Modelos g3db

Para cargar estes modelos debemos facer uso da clase AssetManager.

Como isto é optativo na parte 2D non fai falla que o provedes.

Para cargar un modelo g3db previamente obtido coa ferramenta fbx-conv debemos facer o seguinte:

- Crear un obxecto da clase AssetManager (que xa podemos ter creado se utilizamos esta forma de carga dos gráficos do xogo):

```
AssetManager assetManager = new AssetManager();
```

- Chamamos ó método load para cargar o modelo e esperamos a que remate de cargar.

```
assetManager.load("modelos/ship.g3db", Model.class);

        assetManager.finishLoading();
```

- Obtemos o obxecto da clase Model como fixemos no caso anterior:

```
Model model = assetManager.get("modelos/ship.g3db", Model.class);
```

- Cargamos o Mesh que se atopa no Model (só temos un).

```
meshNave = model.meshes.get(0);
```

O código completo:

```
AssetManager assetManager = new AssetManager();
        assetManager.load("modelos/ship.g3db", Model.class);

        assetManager.finishLoading();

        Model model = assetManager.get("modelos/ship.g3db", Model.class);
        meshNave = model.meshes.get(0);
```

Modelos 3D xa feitos

Temos diferentes sitios web onde obter modelos 3D.

- http://opengameart.org/
- http://tf3dm.com/

-- Ángel D. Fernández González -- (2014).