

1 LIBGDX APICargaModelos3D

UNIDADE 5: Carga Modelos 3D.

1.1 Sumario

- 1 Introducción
- 2 Carga de Modelos
 - ◆ 2.1 Un só modelo
 - ◆ 2.2 Múltiples modelos

1.1.1 Introducción

O obxectivo deste punto é amosar como podemos cargar os modelos utilizando a nova API 3D.

Neste punto podedes cargar calquera modelo que atoparedes por Internet ou un voso.

Lembrar que é mellor ter o modelo en formato binario que en formato obj.

Tedes [neste enlace](#) como cambiar de formato.

Tamén imos facer uso da clase `AssetManager` [explicada anteriormente](#).

Preparación:

- Descomprime o arquivo `Ship.zip` e copia o seu contido ó cartafol `/assets/` da versión Android: [Media:Ship.zip](#)
- Copia a clase `EX_1_DefinicionLuz` do [punto anterior](#) e pon de nome `EX_2_CargaModelos`.
- Crea unha nova clase de nome `EX_2_CargaModelos` e copia o contido seguinte:

Código da clase `EX_2_CargaModelosInicial`

```
import com.badlogic.gdx.Gdx;
import com.badlogic.gdx.Screen;
import com.badlogic.gdx.graphics.GL20;
import com.badlogic.gdx.graphics.PerspectiveCamera;
import com.badlogic.gdx.graphics.g3d.Environment;
import com.badlogic.gdx.graphics.g3d.ModelBatch;
import com.badlogic.gdx.graphics.g3d.ModelInstance;
import com.badlogic.gdx.graphics.g3d.attributes.ColorAttribute;
import com.badlogic.gdx.graphics.g3d.environment.DirectionallLight;
import com.badlogic.gdx.graphics.g3d.utils.CameraInputController;

public class EX_2_CargaModelos implements Screen {

    private PerspectiveCamera camara3d;
    private ModelInstance modeloInstancia;
    private ModelBatch modelBatch;

    private Environment environment;
    private CameraInputController camController;

    public EX_2_CargaModelosInicial() {
        camara3d = new PerspectiveCamera();
        camController = new CameraInputController(camara3d);
        Gdx.input.setInputProcessor(camController);

        modelBatch = new ModelBatch();
        environment = new Environment();
        environment.set(new ColorAttribute(ColorAttribute.AmbientLight, 0.4f, 0.4f, 0.4f, 1f));
```

```

        environment.add(new DirectionalLight().set(1f, 1f, 1f, 1f, 0f, 0f));

    }

    @Override
    public void render(float delta) {
        // TODO Auto-generated method stub
        Gdx.gl.glClear(GL20.GL_COLOR_BUFFER_BIT | GL20.GL_DEPTH_BUFFER_BIT);

        camController.update();

        modelBatch.begin(camara3d);

        modelBatch.end();
    }

    @Override
    public void resize(int width, int height) {
        // TODO Auto-generated method stub

        camara3d.fieldOfView=67;
        camara3d.viewportWidth=width;
        camara3d.viewportHeight=height;

        Gdx.input.setInputProcessor(camController);

        camara3d.position.set(0f,0f,15f);
        camara3d.lookAt(0,0,0);
        camara3d.near=1;
        camara3d.far=300f;
        camara3d.update();

    }

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

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

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

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

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

        modelBatch.dispose();

    }

}

```

- Modifica a clase PracticasNovaApi3D para que chame á clase EX_2_CargaModelos

1.1.2 Carga de Modelos

1.1.2.1 Un só modelo

A carga de modelos utilizando a clase AssetManager non ten moito problema:

- Instanciamos a clase AssetManager.

```
AssetManager assets = new AssetManager();
```

- Chamamos ó método load cargando o modelo obj ou g3db se o convertimos anteriormente.

```
assets.load("ship.obj", Model.class);
assets.finishLoading();
```

- Obtemos o Model a partires do modelo cargado. A partires de aquí é o mesmo que no [explicado no punto anterior](#).

```
Model model = assets.get("ship.obj", Model.class);
modeloInstancia = new ModelInstance(model);
```

- Renderizamos o ModelInstance (é o que determina a posición do modelo no espazo 3D).

```
@Override
public void render(float delta) {
    // TODO Auto-generated method stub
    Gdx.gl.glClear(GL20.GL_COLOR_BUFFER_BIT | GL20.GL_DEPTH_BUFFER_BIT);

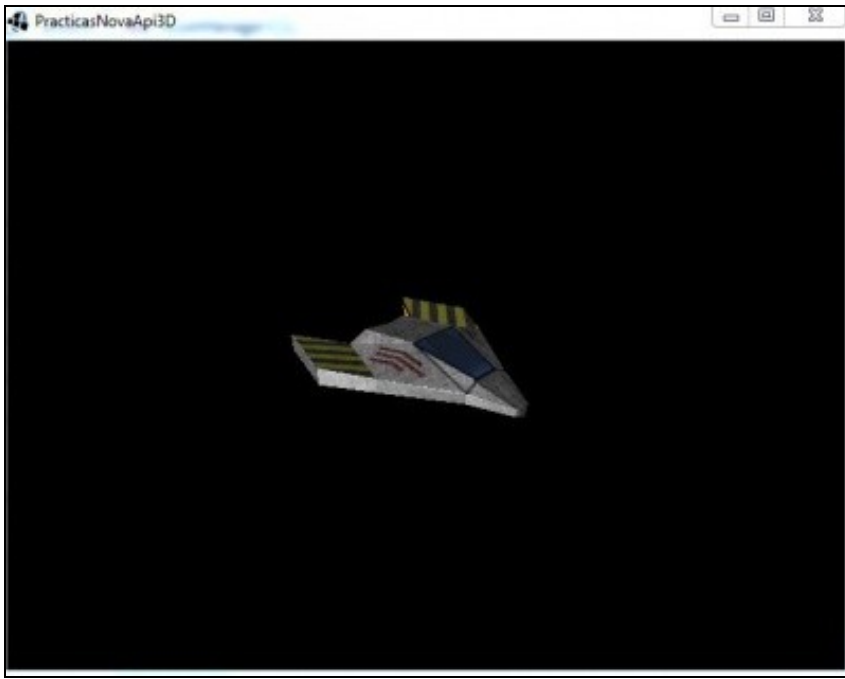
    camController.update();

    modelBatch.begin(camara3d);

    modelBatch.render(modeloInstancia, environment);

    modelBatch.end();
}
```

Ó final teremos como resultado isto:



1.1.2.2 Múltiples modelos

Preparación:

- Descargade este arquivo e o descomprimides no cartafol /assets/ da versión Android: [Media:LIBGDX_cubo.zip](#).

Como veredes ven o obj en forma binaria.

Agora imos cargar este cubo xunto coa nave. Para facelo temos que definir un array de `ModelInstance`.

Lembrar que a clase `ModelInstance` vai gardar nunha matriz a posición, rotación e escalado do modelo cargado.

Polo tanto:

- Definimos o array de `ModelInstance`:

```
private Array<ModelInstance> instances;
```

- Instanciamos o array, normalmente no constructor:

```
instances = new Array<ModelInstance>();
```

- Cargamos os modelos no `AssetManager`:

```
AssetManager assets = new AssetManager();
assets.load("ship.obj", Model.class);
assets.load("cube.g3db", Model.class);
assets.finishLoading();

Model modelNave = assets.get("ship.obj", Model.class);
Model modelCubo = assets.get("cube.g3db", Model.class);
```

- Unha vez cargados os modelos, os engadimos ó array:

```
instances.add(new ModelInstance(modelNave));  
instances.add(new ModelInstance(modelCubo));
```

Nota: Fixarse que agora o array ten dous ModelInstance (na posición 0 está a nave e na 1 o cubo). No seguinte punto imos explicar como mover cada un deles, polo de agora só poñemos o código e o explicamos posteriormente.

- Movemos o cubo para que non coincida na mesma posición cá nave.

```
instances.get(0).transform.setToTranslation(2, 2, 2);
```

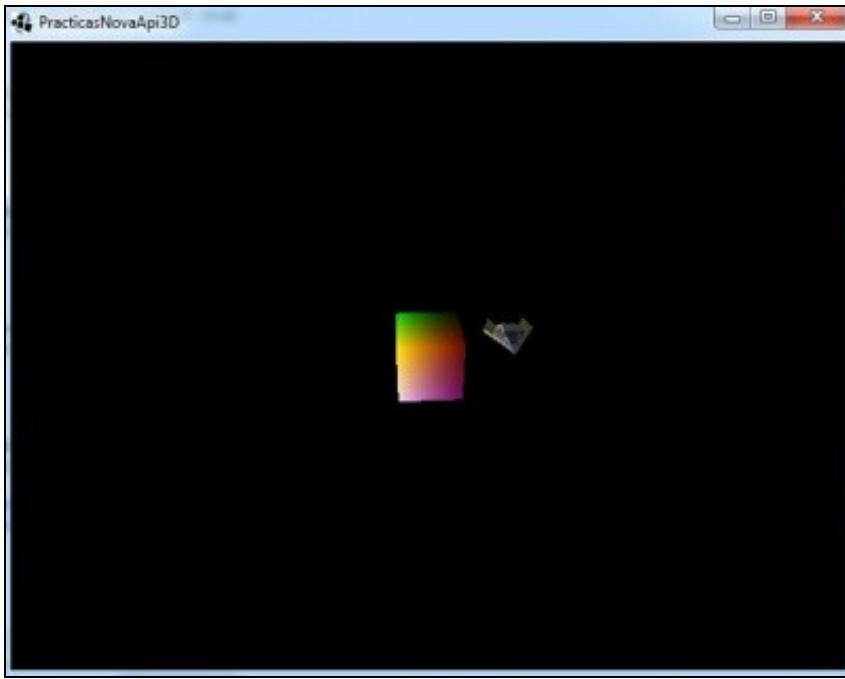
- Renderizamos o array completo:

```
@Override  
public void render(float delta) {  
    // TODO Auto-generated method stub  
    Gdx.gl.glClear(GL20.GL_COLOR_BUFFER_BIT | GL20.GL_DEPTH_BUFFER_BIT);  
  
    camController.update();  
  
    modelBatch.begin(camara3d);  
  
    modelBatch.render(instances, environment);  
  
    modelBatch.end();  
}
```

- Liberamos o array:

```
@Override  
public void dispose() {  
    // TODO Auto-generated method stub  
  
    modelBatch.dispose();  
    instances.clear();  
  
}
```

Dará como resultado isto:



Código da classe EX_2_CargaModelos

Obxectivo: Visualizar varios modelos.

```
import com.badlogic.gdx.Gdx;
import com.badlogic.gdx.Screen;
import com.badlogic.gdx.assets.AssetManager;
import com.badlogic.gdx.graphics.GL20;
import com.badlogic.gdx.graphics.PerspectiveCamera;
import com.badlogic.gdx.graphics.g3d.Environment;
import com.badlogic.gdx.graphics.g3d.Model;
import com.badlogic.gdx.graphics.g3d.ModelBatch;
import com.badlogic.gdx.graphics.g3d.ModelInstance;
import com.badlogic.gdx.graphics.g3d.attributes.ColorAttribute;
import com.badlogic.gdx.graphics.g3d.environment.DirectionallLight;
import com.badlogic.gdx.graphics.g3d.utils.CameraInputController;
import com.badlogic.gdx.utils.Array;

public class EX_2_CargaModelosInicial implements Screen {

    private PerspectiveCamera camara3d;
    private ModelBatch modelBatch;

    private Array<ModelInstance> instances;

    private Environment environment;
    private CameraInputController camController;

    public EX_2_CargaModelosInicial() {
        camara3d = new PerspectiveCamera();
        camController = new CameraInputController(camara3d);
        Gdx.input.setInputProcessor(camController);

        modelBatch = new ModelBatch();
        environment = new Environment();
        environment.set(new ColorAttribute(ColorAttribute.AmbientLight, 0.4f, 0.4f, 0.4f, 1f));
        environment.add(new DirectionallLight().set(1f, 1f, 1f, 1f, 0f, 0f));

        instances = new Array<ModelInstance>();

        AssetManager assets = new AssetManager();
        assets.load("ship.obj", Model.class);
        assets.load("cube.g3db", Model.class);
        assets.finishLoading();
    }
}
```

```

        Model modelNave = assets.get("ship.obj", Model.class);
        Model modelCubo = assets.get("cube.g3db", Model.class);

        instances.add(new ModelInstance(modelNave));
        instances.add(new ModelInstance(modelCubo));

        instances.get(0).transform.setToTranslation(2, 2, 2);

    }

    @Override
    public void render(float delta) {
        // TODO Auto-generated method stub
        Gdx.gl.glClear(GL20.GL_COLOR_BUFFER_BIT | GL20.GL_DEPTH_BUFFER_BIT);

        camController.update();

        modelBatch.begin(camara3d);

        modelBatch.render(instances, environment);

        modelBatch.end();
    }

    @Override
    public void resize(int width, int height) {
        // TODO Auto-generated method stub

        camara3d.fieldOfView=67;
        camara3d.viewportWidth=width;
        camara3d.viewportHeight=height;

        Gdx.input.setInputProcessor(camController);

        camara3d.position.set(0f, 0f, 15f);
        camara3d.lookAt(0, 0, 0);
        camara3d.near=1;
        camara3d.far=300f;
        camara3d.update();

    }

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

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

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

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

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

```

```
modelBatch.dispose();  
instances.clear();  
  
}  
  
}
```

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