LIBGDX APIMoverModelos3D

UNIDADE 5: Mover modelos 3D

Introdución

Neste apartado imos ver como mover os modelos 3D cargados coa clase ModelInstance, facendo uso das matrices.

Para mover os modelos só temos que aplicar o aprendido na Unidade 4.

Preparación:

Creamos unha nova clase de nome EX_3_MoverModelos que sexa chamada pola clase PracticasNovaApi3D.

Movendo os Modelos 3D

A matriz de modelado-vista se atopa na propiedade transform do ModelInstance, e este a su vez está asociado a un Model.

Clase utilizada: Matrix4.

Operacións (todas elas sobrecargadas) sobre a **propiedade transform** do ModelInstance:

- Traslación:

  public Matrix4 setToTranslation(float x,float y, float z): Carga a matriz identidade e posiciona o obxecto nas coordenadas indicadas. **Esta sería a forma que deberíamos utilizar se seguimos o esquema Modelo-Vista-Controlador**.

  public Matrix4 setTranslation(float x,float y, float z): Posiciona o obxecto nas coordenadas indicadas. Non carga a matriz identidade.

- Rotación:

  public Matrix4 setToRotation(Vector3 axis, float degrees): Carga a matriz de identidade e rota a matriz no ángulo indicado en grados e no eixe que teña o valor 1.

  public Matrix4 rotate(float axisX,float axisY,float axisZ,float degrees): Rota a matriz no ángulo indicado en grados e no eixe que teña o valor 1.

- Escala:

  public Matrix4 setToScaling(Vector3 vector): Carga a matriz identidade e escala a matriz no valor indicado no eixe correspondente.

Atención:: Se usamos este método estaremos perdendo a información gardada na matriz de modelado. Para aplicar unha traslación e escala teremos que aplicar o método seguinte:

- Traslación e escala:

  public Matrix4 setToTranslationAndScaling(float translationX,float translationY,float translationZ,float scalingX,float scalingY,float scalingZ): Rota e escala á vez. Carga a matriz de identidade e aplica a rotación e escala nos eixes indicados cun valor de 1.

Imos facer un exemplo.

Seguindo co exercicio anterior (o podedes gardar facendo copy e paste sobre o propio arquivo en Eclipse).

Código da clase EX_3_MoverModelos

Obxectivo: Cargar varios modelos e animalos. Operacións de escala, rotacion e traslación.

```
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.DirectionalLight;
import com.badlogic.gdx.graphics.g3d.utils.CameraInputController;
import com.badlogic.gdx.utils.Array;



public class EX_3_MoverModelos implements Screen {

private PerspectiveCamera camara3d;
private  ModelBatch modelBatch;

private Array<ModelInstance> instances;
private ModelInstance instanceNave;


private Environment environment;
private CameraInputController camController;

private float pos;

public EX_3_MoverModelos(){
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));

        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);

        instanceNave = new ModelInstance(modelNave);
        instanceNave.transform.setToRotation(0, 1, 0, 90);

        for (int cont=1; cont < 10;cont++){
             ModelInstance cubo = new ModelInstance(modelCubo);
             if (cont>3 && cont<6){
             cubo.transform.setToTranslationAndScaling(cont*3, 0, 0, 0.5f, 0.5f, 0.5f);
             }
             else
             cubo.transform.setTranslation(cont*3,0,0);

            instances.add(cubo);

        }

}
```

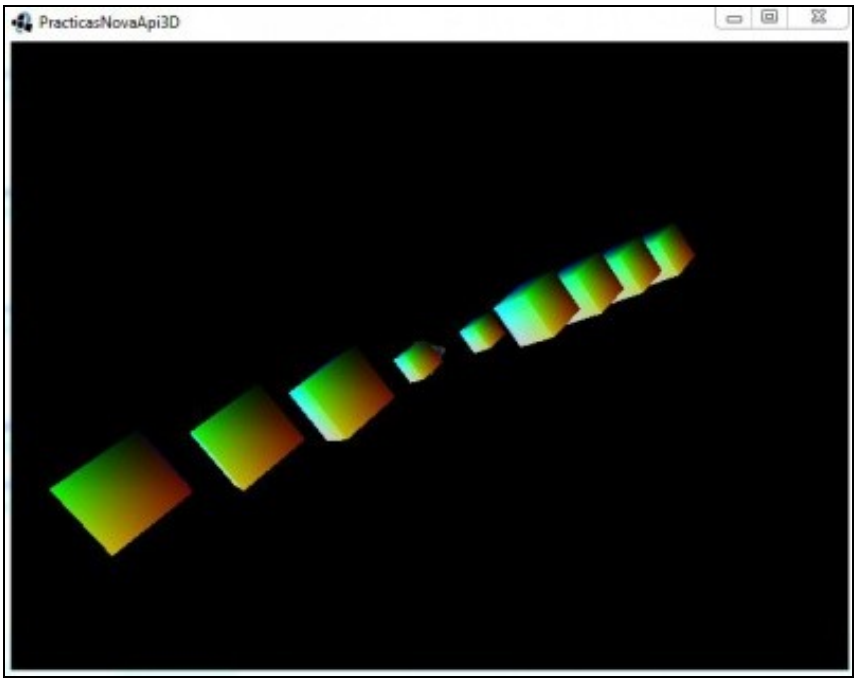
```
@Override
public void render(float delta) {
// TODO Auto-generated method stub
Gdx.gl.glClear(GL20.GL_COLOR_BUFFER_BIT | GL20.GL_DEPTH_BUFFER_BIT);

camController.update();

pos+=delta * 1f;

for (ModelInstance modelI : instances){
modelI.transform.rotate(0,1,0,delta*30f);// Continua a rotación, non empeza de 0
}

instanceNave.transform.setTranslation(pos,0,0);


camara3d.lookAt(pos,0,0);
camara3d.update();

        modelBatch.begin(camara3d);

        modelBatch.render(instances,environment);
        modelBatch.render(instanceNave,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();


        }

        }
```

Comentemos o código:

- Liñas 51-52: Creamos un ModelInstance a partires do Model da nave e rotamos no eixe Y 90 grados a nave, para poñela cara os cubos.
- Liñas 54-62: Creamos o array de ModelInstance. Os trasladamos chamando ó método setTranslate e dous deles (os números 4 e 5) ademais de trasladalos os escalamos á metade do seu tamaño orixinal.
- Liñas 78-80: Rotamos todos os cubos. Como xa están trasladados, a rotación é no punto onde se atopan (como o movemento de rotación da Terra). Chamamos a translate e non a setToTranslate xa que este último inicializa a matriz e polo tanto movería todos os cubos á posición (0,0,0).
- Liña 82: Movemos a nave.
- Liñas 85-86: Facemos que a cámara siga á nave.
- Liñas 90-91: Renderizamos os ModelInstace dos bloques (array) e a nave.

Ó final teremos como resultado isto:



Modelo-Vista-Controlador: Movendo os Modelos 3D

Imos modificar o exercicio anterior para que siga a arquitectura Modelo-Vista.

O controlador xa o sabedes implementar das unidades referidas o xogo 2D polo que por motivos de 'rapidez' non o imos utilizar neste exemplo.

Preparación:

- Crear unha nova clase de nome EX_4_MoverModelosMVC.
- Cambiade a clase PracticasNovaApi3D para que chame á nova clase.

Empecemos creando os modelos:

Código da clase Elemento3D
Obxectivo: Crear o Modelo. Saltamos o paso de crear a clase Mundo para facelo máis sinxelo.

```
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;
public Vector3 rotacion;

public float anguloRotacion;
private int velocidadeRotacion;

private Vector3 temp;

public Elemento3D(Vector3 pos, float escala,Vector3 velocidade, Vector3 rotacion,int velocidadeRotacion){
matriz = new Matrix4();
posicion = pos;
this.escala=escala;
this.velocidade = velocidade;
this.rotacion=rotacion;
anguloRotacion=0;
this.velocidadeRotacion=velocidadeRotacion;

temp = new Vector3();
}


public void update(float delta){

temp.set(velocidade);
posicion.add(temp.scl(delta));
anguloRotacion+=delta*velocidadeRotacion;

matriz.idt();
matriz.translate(posicion);
matriz.scl(escala);
matriz.rotate(rotacion, anguloRotacion);

}

}
```

Como vemos o máis importante a ter en conta é que imos gardar nesta clase a Matriz de modelado que vai usarse para representar a figura. Fixarse como sempre partimos da matriz de identidade e aplicamos a traslación, escala e rotación (liñas 35-38).

- Liñas 24-25: Agora xa non necesitamos un array de ModelInstances. Imos gardar un único ModelInstance por cada tipo de elemento gráfico (unha nave e un bloque).
- Liñas 27-28: Gardamos en obxectos da clase Elemento3D os datos de cada figura.
- Liñas 44-53: Facemos o mesmo que no exercicio anterior (os mesmos datos) pero esta vez son gardados en obxectos que pertencen á clase Elementos3D.

- Liña 76: A cámara mira cara a posición da nave.
- Liñas 81-86: **Moi importante:** Actualizamos a MATRIZ de cada bloque e asinamos ó ModelInstance a matriz gardada.
- Liñas 88-92: Movemos a nave e a rotamos para que estea cara os bloques. Actualizamos a matriz e asinamos ó ModelInstance da nave a matriz gardada.
- Liña 145: Limpamos o array del bloques.

Código da clase EX_4_MoverModelosMVC

Obxectivo: Facemos o mesmo exercicio anterior pero utilizando a matriz gardada na clase Elemento3D.

```
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.DirectionalLight;
import com.badlogic.gdx.graphics.g3d.utils.CameraInputController;
import com.badlogic.gdx.math.Vector3;
import com.badlogic.gdx.utils.Array;
import com.plategaxogo3davanzado.angel.Elemento3D;



public class EX_4_MoverModelosMVC implements Screen {

private PerspectiveCamera camara3d;
private  ModelBatch modelBatch;

private ModelInstance instanceNave;
private ModelInstance instanceBloques;

private Array<Elemento3D>bloques;
private Elemento3D nave;

private Environment environment;
private CameraInputController camController;


public EX_4_MoverModelosMVC(){
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));

                bloques = new Array<Elemento3D>();
                for (int cont=1; cont < 10;cont++){
                if (cont>3 && cont<6){
             bloques.add(new Elemento3D(new Vector3(3*cont, 0, 0),0.5f,new Vector3(0,0,0),new Vector3(1,1,1),30));
                }
                else
             bloques.add(new Elemento3D(new Vector3(3*cont, 0, 0),1f,new Vector3(0,0,0),new Vector3(1,1,1),50));
                }

                nave = new Elemento3D(new Vector3(0,0,0), 1, new Vector3(1f, 0, 0),new Vector3(0,0,0),0);



                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);

                instanceBloques = new ModelInstance(modelCubo);
                instanceNave = new ModelInstance(modelNave);
}

@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();

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

        modelBatch.begin(camara3d);

        for (Elemento3D bloque : bloques){
             bloque.update(delta);
             instanceBloques.transform.set(bloque.matriz);
            modelBatch.render(instanceBloques,environment);

        }

        nave.posicion.x += nave.velocidade.x*delta;
        nave.update(delta);
        nave.matriz.rotate(0, 1,0,90);
        instanceNave.transform.set(nave.matriz);
        modelBatch.render(instanceNave,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();
bloques.clear();


}

}
```

Vexamos as diferenzas:

- Liñas 24-25: Agora xa non necesitamos un array de ModelInstances. Imos gardar un único ModelInstance por cada tipo de elemento gráfico (unha nave e un bloque).
- Liñas 27-28: Gardamos en obxectos da clase Elemento3D os datos de cada figura.
- Liñas 44-53: Facemos o mesmo que no exercicio anterior (os mesmos datos) pero esta vez son gardados en obxectos que pertencen á clase Elementos3D.
- Liña 76: A cámara mira cara a posición da nave.
- Liñas 81-86: **Moi importante:** Actualizamos a MATRIZ de cada bloque e asinamos ó ModelInstance a matriz gardada.
- Liñas 88-92: Movemos a nave e a rotamos para que estea cara os bloques. Actualizamos a matriz e asinamos ó ModelInstance da nave a matriz gardada.
- Liña 145: Limpamos o array del bloques.

Se executades o código o resultado é o mesmo que no caso anterior.

Por eficiencia pode ser mellor ter un array de ModelInstances e renderizalos todos de vez. Isto o podemos facer igual pero teríamos que asinar a cada elemento do array de ModelInstace a matriz gardada no array de Elementos3D.

No blog de Xoppa tedes unha entrada no que amosa como podemos cargar todos os modelos de vez dun só acceso á disco:
http://blog.xoppa.com/loading-a-scene-with-libgdx/

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