# LIBGDX Tarefas entregar UD4

## Sumario



## ACLARACIÓNS TAREFAS UD4

Lembrar que na unidade 3D se pode escoller unha das dúas opcións.

- Facer as tarefas indicadas.
- Facer un xogo proposto.

A máis doada de facer son as tarefas.

## TAREFAS

Nota: Se escolledes facer as tarefas non será necesario entregar o xogo.

**Tarefa 4.1**

ENLACE A MIRAR.

Deseña un cadrado que vaia dende -0.5f a 0.5f, tanto no eixe x como no y.

Podes facelo de dúas formas:

- Con dous triángulos (crear dous obxectos mesh).
- Debuxar un cadrado.

**Pistas:**

- Definir un único Mesh que tería catro vértices e seis índices.
- Con 3 índices formas un triángulo. Con 6 podes formar dous facendo un cadrado. Se empezas a numerar de abaixo-dereita-arriba-esquerda (os catro vértices) coa secuencia de índices: 0-1-2 (forma un triángulo) e 0-2-3 (formas o outro triángulo) temos un cadrado.
- Lembra modificar o número de índices no método render do Mesh e no constructor o número de índices e vértices.

- Comprimide o código fonte de dita clase nun zip co voso nome e apelidos e subídeo xunto coas outras actividades.

**Tarefa 4.2**

ENLACE A MIRAR.

Modifica a clase UD4_2_Camara2D e fai que a cámara ortográfica non visualice o triángulo verde. modificando o ViewFrustrum da cámara (planos far e near).

- Comprimide o código fonte de dita clase nun zip co voso nome e apelidos e subídeo xunto coas outras actividades.

**Tarefa 4.3**

ENLACE A MIRAR.

Modifica o arquivo vertex.vert e fai que o triángulo mida o dobre de alto.

- Comprimide o arquivo vertex.vert nun zip co voso nome e apelidos e subídeo xunto coas outras actividades.

**Tarefa 4.4**

ENLACE A MIRAR.

Crea unha nova clase de nome UD4_TAREFA_4_4 partindo do código da clase UD4_4_ProgramShader (o código está posto ó final do enlace e mirar) fai que a cámara rote arredor do triángulo vermello.

Pistas:

- Necesitarás chamar ó método rotateAround.

  Dito método espera recibir como parámetros os seguintes:

  - Primeiro parámetro: Un punto cara onde ten que mirar a cámara. No noso caso é o: new Vector3(0,0,0).
  - Segundo parámetro: O vector que ten que rotar.
    Para sabelo repasa o gráfico da regra da man dereita. Para saber o eixe tedes que imaxinar unha lanza atravesando a vosa testa. Así, se a lanza é o eixe X moveredes a cabeza de arriba-abaixo,....e así por cada eixe. Se queredes que se mova nun eixe determinado poñeredes como segundo parámetro: new Vector3(1,0,0). Neste caso se movería arriba-abaixo.
  - Terceiro parámetro: o ángulo a rotar. Teredes que multiplicalo por Gdx.graphics.getDeltaTime() para indicar que rote X grados cada segundo.

**Lembrar chamar ó método update da cámara** e engadir o código ó método render.

Nota: Lembrar que segundo vimos nos consellos de programación é mellor facer un só new no constructor e poñer: vector.set(x,y,z) para darlle un valor.

- Comprimide a clase nun zip co voso nome e apelidos e subídeo xunto coas outras actividades.

**Tarefa 4.5**

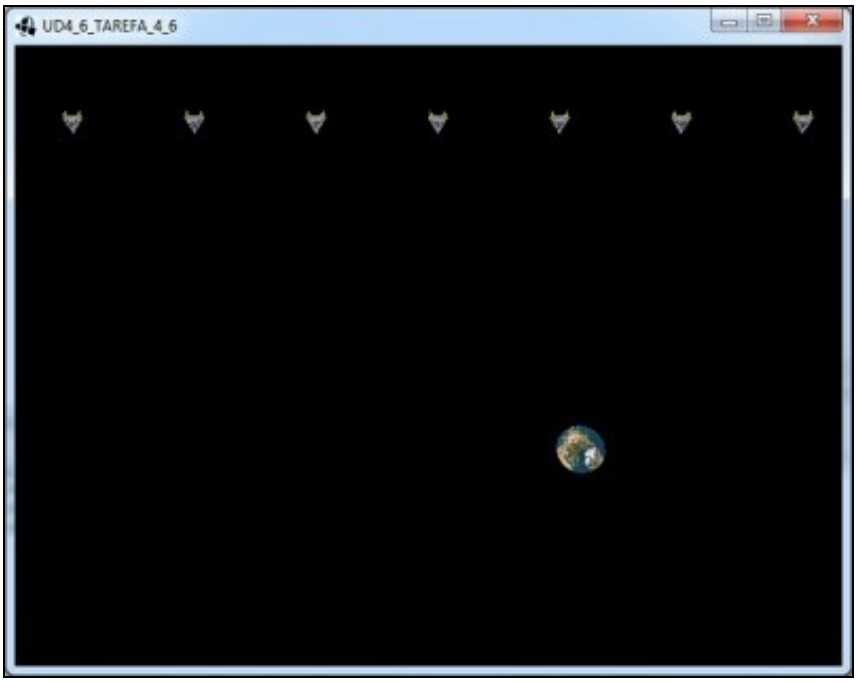
ENLACE A MIRAR: Animacións en 3D.

ENLACE A MIRAR: Carga Modelos3D.

Imos crear un novo escenario no que dispoñemos de naves que temos que levar á Terra. As naves están na parte superior e a Terra está xirando na inferior.

Aviso: Neste código non estamos a separar o Modelo-Vista-Controlador como fixemos na parte 2D. Isto o facemos por motivos de tempo, pudendo aplicar todos os conceptos aprendidos no desenvolvemento do xogo 2D.

Unha imaxe do que queremos conseguir:




**Preparacion:**

- Crea unha clase de nome Terra, que teña este código:

**Código da clase Terra**
**Obxectivo:** Clase que garda a información necesaria para mover a Terra.

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

/**
 * Elemento 3D con rotación
 * @author ANGEL
 *
 */
public class Terra extends Elemento3D{

public float velocidadeRotar;
public Vector3 eixeRotar;
private float anguloRotacion;

public final float POSINICALX=-400;
public final float POSFINALX=400;


public Terra(Vector3 pos, float escala,Vector3 velocidade,float velocidadeRotar,Vector3 eixeRotar){
super(pos,escala,velocidade);
this.velocidadeRotar=velocidadeRotar;
this.eixeRotar = eixeRotar;

}

public void update(float delta){

super.update(delta);

anguloRotacion +=delta*velocidadeRotar;
matriz.rotate(eixeRotar,anguloRotacion);
```

```
}

}
```

- Descomprime o arquivo seguinte e copia o seu contido (tierra.obj / tierra.jpg) ó cartafol /assets/modelos/ da versión Android.

Media:Tierra.zip

- Crea unha nova clase que derive de Game de nome UD4_6_TAREFA_4_6. Cambia os diferentes proxectos para que carguen a nova clase.

A continuación vos deixo o código da clase **pero faltan liñas por programar**. Indico o que hai que facer nos comentarios dentro do código:

**Código da clase UD4_TAREFA_4_5**
**Obxectivo:** Implementar o escenario pedido.

```
import java.util.ArrayList;

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;
import com.plategaxogo3d.exemplos.Elemento3D;
import com.plategaxogo3d.exemplos.Terra;

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

public class UD4_TAREFA_4_5 extends Game {

private Mesh meshNave;
private Mesh meshTerra;

private ArrayList<Elemento3D> naves;
private  Terra terra;

private ShaderProgram shaderProgram;

private Texture texturaNave;
private Texture texturaTerra;
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);


        /* CÓDIGO POR FACER
         * Falta cargar o meshTerra có modelo gardado en modelos/tierra.obj
         */


FileHandle imageFileHandle = Gdx.files.internal("modelos/ship.png");
texturaNave = new Texture(imageFileHandle);
imageFileHandle = Gdx.files.internal("modelos/tierra.jpg");
texturaTerra = new Texture(imageFileHandle);

camara3d = new PerspectiveCamera();

naves = new ArrayList<Elemento3D>();
naves.add(new Elemento3D(new Vector3(-380f,0f,-250f), 20f, new Vector3(0,0,0)));
naves.add(new Elemento3D(new Vector3(-250f,0f,-250f), 20f, new Vector3(0,0,0)));
naves.add(new Elemento3D(new Vector3(-120f,0f,-250f), 20f, new Vector3(0,0,0)));
naves.add(new Elemento3D(new Vector3(+10f,0f,-250f), 20f, new Vector3(0,0,0)));
naves.add(new Elemento3D(new Vector3(+140f,0f,-250f), 20f, new Vector3(0,0,0)));
naves.add(new Elemento3D(new Vector3(+270f,0f,-250f), 20f, new Vector3(0,0,0)));

        /* CÓDIGO POR FACER
         * Falta engadir unha nave na posición (+400f,0f,-250f)
         * Falta instanciar a terra (propiedade terra) na posición (0,0,100f), escala 25f, velocidade (100f,0,0) e eixe a mover (0,1
         */


}

private void controlarTerra(float delta){
terra.update(delta);
if ((terra.posicion.x >= terra.POSFINALX) | (terra.posicion.x <= terra.POSINICALX)){
terra.velocidade.x=-1*terra.velocidade.x;
}

}

@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);

        /* CÓDIGO POR FACER
         * Falta colocar a cámara na posición (0,500f,0) e que mire cara a posición (0,0,0).
         * Dependendo da resolución da vosa pantalla (e a resolución que utilizades na versión Desktop) pode ser necesario aumentar
         *   Podedes probar con diferentes valores ó de 500f máis alonxados como 600f,...
         * LEMBRAR ACTUALIZAR A CAMARA
         */

controlarTerra(Gdx.graphics.getDeltaTime());
for (Elemento3D nave: naves){
nave.update(Gdx.graphics.getDeltaTime());
}

shaderProgram.begin();

texturaNave.bind(0);
shaderProgram.setUniformi("u_texture", 0);
for (Elemento3D nave: naves){
shaderProgram.setUniformMatrix("u_worldView", camara3d.combined.cpy().mul(nave.matriz));
meshNave.render(shaderProgram, GL20.GL_TRIANGLES);
}


texturaTerra.bind(1);
shaderProgram.setUniformi("u_texture", 1);
shaderProgram.setUniformMatrix("u_worldView", camara3d.combined.cpy().mul(terra.matriz));
```

```
meshTerra.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=5000f;
camara3d.near=0.1f;
camara3d.update();
}

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

}

}
```

- Comprimide a clase nun zip co voso nome e apelidos e subídeo xunto coas outras actividades.

**Tarefa 4.6**

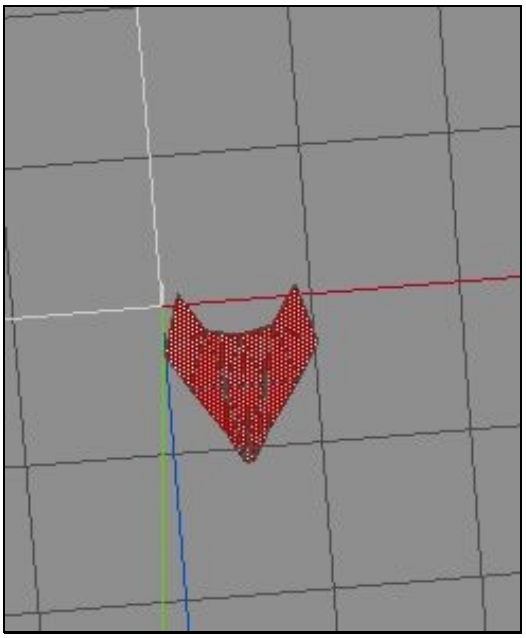
ENLACE A MIRAR.

**Preparación:** Partindo do código da clase UNIDADE4_TAREFA_4_5 feito na tarefa 4.5, crea unha nova clase de nome UNIDADE4_TAREFA_4_6 (copia e pega a tarefa anterior e vos pedirá un novo nome).

O obxectivo é facer que cando se preme na pantalla a nave se mova cara a terra e controlar se choca con ela ou se perde. Ó premer unha vez se moverá a primeira nave, ó premer outra vez a segunda e así continuamente ata que se acaben as naves. Nese intre volveremos a empezar.
Podedes descargar a versión PC desta tarefa neste enlace: Media:Tarefa4_6.zip
Esta é a versión máis sinxela.

Para controlar os choques tedes varias opcións:

- A máis sinxela é utilizar a clase Sphere. Aínda que as naves non teñan esa forma se pode empregar sen problemas. Tede en conta a escala:



Así podedes controlar os choques de esfera con esfera.

- Máis complexa: neste solución teredes que controlar o choque dunha esfera cun BoundingBox. Neste caso non temos a función que o controle e teríamos que implementala nos.

Unha opción é a implementada por Jim Arvo no se libro: A Simple Method for Box-Sphere Intersection Testing by Jim Arvo from "Graphics Gems", Academic Press, 1990:

```
public static boolean intersectsWith(BoundingBox boundingBox, Sphere sphere) {
    float dmin = 0;

    Vector3 center = sphere.center;
    Vector3 bmin = boundingBox.getMin();
    Vector3 bmax = boundingBox.getMax();

    if (center.x < bmin.x) {
        dmin += Math.pow(center.x - bmin.x, 2);
    } else if (center.x > bmax.x) {
        dmin += Math.pow(center.x - bmax.x, 2);
    }

    if (center.y < bmin.y) {
        dmin += Math.pow(center.y - bmin.y, 2);
    } else if (center.y > bmax.y) {
        dmin += Math.pow(center.y - bmax.y, 2);
    }

    if (center.z < bmin.z) {
        dmin += Math.pow(center.z - bmin.z, 2);
    } else if (center.z > bmax.z) {
        dmin += Math.pow(center.z - bmax.z, 2);
    }

    return dmin <= Math.pow(sphere.radius, 2);
}
```

- Para mover as naves só tedes que engadirlles unha velocidade ó eixe z.
- Podedes variar a tarefa e facela tan complexa como queirades.

  Así podedes (optativo):

◊ Darlle a cada nave un valor aleatorio de velocidade, indicando cunha cámara ortográfica na parte superior á altura de cada nave á velocidade asinada.
◊ Controlar cun raio cando o usuario preme na pantalla e o fai sobre a nave esta empece a moverse.
◊ Levar un contador de naves que chegaron á Terra e nave que se perderon.
◊ Facer que a cámara siga á nave cando estea en movemento ata que desapareza.

- Comprimide a clase nun zip co voso nome e apelidos e subídeo xunto coas outras actividades.

## XOGO PROPOSTO

Nota: Se escolledes facer o xogo non será necesario entregar as tarefas.

Por motivos de tempo non é necesario separar a programación en Modelo-Vista-Controlador, pero eu vos aconsello a lo menos facer a parte de Modelo creando as clases base para gardar a información das figuras 3D e a clase Mundo.

Despois podedes facer o control na propia clase que renderiza.

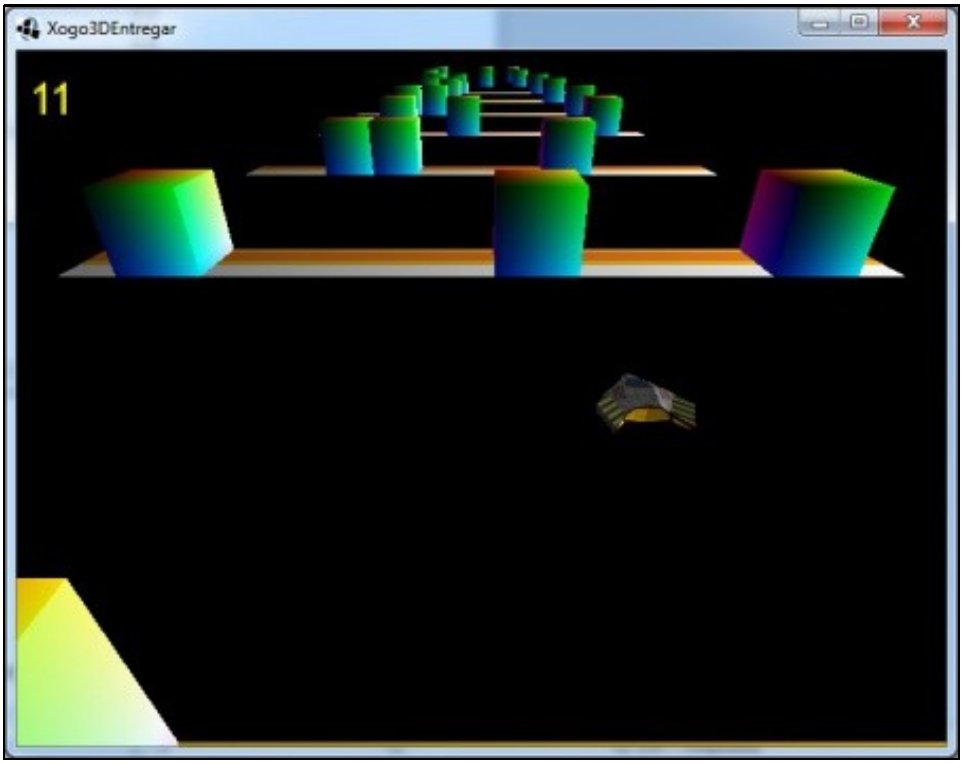
**O xogo a desenvolver:**

### Nome e datos principais

- **NOME:** ESQUIVA.
- **HISTORIA:** Perdido polo espazo tes que chegar a Terra o antes posible. Estás polo que parece ser unha vía espacial no que non paran de aparecer grandes bloques de plasma.

  A túa misión é aguantar 120 segundos ata chegar á Terra esquivando os bloques de plasma.
  Cada 30 segundos conseguirás aumentar a velocidade da nave para chegar antes á Terra.

- **OBXECTIVO:** SOBREVIVE DURANTE 120 SEGUNDOS.
- **CATEGORÍA:** Casual.



### Gráficos

Tedes neste zip os gráficos para realizar o xogo: Media:LIBGDX_assets3D.zip

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