1 LIBGDX As colisions

UNIDADE 2: As colisións

1.1 Sumario

1.1.1 Introducción

Para detectar o choque imos utilizar a clase Intersector. Os método de dita clase van devolver un booleano que vai indicar se se produce algunha intersección entre dúas figuras xeométricas.

Tamén pode comprobar se un raio (un punto cun vector dirección) choca cun plano ou figura xeométrica así como calcular distancias entre puntos ou puntos e figuras xeométricas.

Importante: Non é necesario instanciar a clase Intersector. Todos os seus métodos son de clase e polo tanto para chamar a calquera deles só é necesario facer *Intersector.nomeDoMétodo*.

1.1.2 Como funciona a clase Intersector

- Clase **Intersector**: http://libgdx.badlogicgames.com/nightlies/docs/api/com/badlogic/gdx/math/Intersector.html

  Algúns dos métodos que nos ofrece:
  Dita clase ten multitude de métodos que son fáciles de entender xa polo seu nome...

  - **static float distanceLinePoint(float startX, float startY, float endX, float endY, float pointX, float pointY)**
    Devolve a distancia entre unha liña e un punto. Os parámetros que se dan a este método serían a liña, definida por catro coordenadas e un punto, definido por dous coordenadas. Está sobrecargado.
  - **static boolean intersectRayBoundsFast(Ray ray, BoundingBox box)**
    Comproba se un raio (un punto cun vector dirección) intersecciona cun BoundingBox (unha figura xeométrica en forma de cubo ou prima. O veremos na parte 3D).
  - **static boolean overlaps(Circle c1, Circle c2))**
    Devolve true ou false dependendo se os dous círculos indicados se solapan.
    Está sobrecargado e temos métodos que:
    - ◊ static boolean overlaps(Circle c, Rectangle r): se un círculo se solapa cun rectángulo.
    - ◊ static boolean overlaps(Rectangle r1, Rectangle r2): se un rectángulo se solapa con outro rectángulo.

Un exemplo...

Preparación: Agora ides facer unha copia da clase RendererXogo, xa que imos modificala para amosarvos como se poden controlar os choques. Premede co rato sobre a clase, botón dereito e escollede a opción Copy. Despois repetides a operación pero escolledes a opción Paste. Vos preguntará un nome para a clase. Indicade **UD2_5_RendererXogo**.

Modificade a pantalla PantallaXogo para que chame a esta clase.

Código da clase UD2_5_RendererXogo
Obxectivo: Facer que o alien se mova co dedo e comprobar cando choca coa nave espacial.

```
package com.plategaxogo2d.renderer;
```

```
import com.badlogic.gdx.Gdx;
import com.badlogic.gdx.graphics.Color;
import com.badlogic.gdx.graphics.GL20;
import com.badlogic.gdx.graphics.OrthographicCamera;
import com.badlogic.gdx.graphics.g2d.SpriteBatch;
import com.badlogic.gdx.graphics.glutils.ShapeRenderer;
import com.badlogic.gdx.graphics.glutils.ShapeRenderer.ShapeType;
import com.badlogic.gdx.math.Circle;
import com.badlogic.gdx.math.Intersector;
import com.badlogic.gdx.math.Rectangle;
import com.badlogic.gdx.math.Vector3;
import com.plategaxogo2d.angel.AssetsXogo;
import com.plategaxogo2d.angel.Controis;
import com.plategaxogo2d.angel.Utiles;
import com.plategaxogo2d.modelo.Alien;
import com.plategaxogo2d.modelo.Mundo;
import com.plategaxogo2d.modelo.Nave;

/*
 * Comprobamos como funciona a clase Intersector.
 */

public class UD2_5_RendererXogo {

    private OrthographicCamera camara2d;
    private SpriteBatch spritebatch;
    private ShapeRenderer shaperender;

private Mundo meuMundo;


    public UD2_5_RendererXogo(Mundo mundo) {
this.meuMundo = mundo;

        camara2d = new OrthographicCamera();
        spritebatch = new SpriteBatch();
        shaperender = new ShapeRenderer();
    }


private void debuxarAlien(){
Alien alien = meuMundo.getAlien();
spritebatch.draw(AssetsXogo.textureAlien, alien.getPosicion().x,alien.getPosicion().y,alien.getTamano().x,alien.getTamano().y);
}

private void debuxarNave(){
Nave nave = meuMundo.getNave();
spritebatch.draw(AssetsXogo.textureNave, nave.getPosicion().x,nave.getPosicion().y,nave.getTamano().x,nave.getTamano().y);
}


    /**
     * Debuxa todos os elementos graficos da pantalla
     *
     * @param delta
     *            : tempo que pasa entre un frame e o seguinte.
     */
    public void render(float delta) {
Gdx.gl.glClearColor(0, 0, 0, 1);
Gdx.gl.glClear(GL20.GL_COLOR_BUFFER_BIT);

spritebatch.begin();

debuxarNave();

debuxarAlien();

spritebatch.end();

debugger();

    }
```

```
private void debugger(){

shaperender.begin(ShapeType.Line);
shaperender.setColor(Color.YELLOW);

Alien alien = meuMundo.getAlien();
shaperender.rect(alien.getPosicion().x, alien.getPosicion().y, alien.getTamano().x, alien.getTamano().y);

shaperender.setColor(Color.RED);
Nave nave = meuMundo.getNave();
shaperender.rect(nave.getPosicion().x, nave.getPosicion().y, nave.getTamano().x, nave.getTamano().y);

shaperender.end();

}


    public void resize(int width, int height) {

camara2d.setToOrtho(false, Mundo.TAMANO_MUNDO_ANCHO,
Mundo.TAMANO_MUNDO_ALTO);
camara2d.update();

spritebatch.setProjectionMatrix(camara2d.combined);
shaperender.setProjectionMatrix(camara2d.combined);

    }

    public void dispose() {
            spritebatch.dispose();
    }


}
```

Agora, por simplificar, non imos usar a interface InputProcessor e imos chamar directamente a métodos do paquete Gdx.input que nos van devolver a última coordenada pulsada na pantalla. *Modificamos o método render e poñemos este código*.

```
public void render(float delta) {
Gdx.gl.glClearColor(0, 0, 0, 1);
Gdx.gl.glClear(GL20.GL_COLOR_BUFFER_BIT);

if (Gdx.input.justTouched()){
float posx=Gdx.input.getX();
float posy = Gdx.input.getY();

Vector3 dedo = new Vector3(posx,posy,0);
camara2d.unproject(dedo);

Alien alien = meuMundo.getAlien();
alien.setPosicion(dedo.x,dedo.y);
Rectangle rectangulo_alien = new Rectangle(alien.getPosicion().x,alien.getPosicion().y,
alien.getTamano().x,alien.getTamano().y);

Nave nave = meuMundo.getNave();
Rectangle rectangulo_nave = new Rectangle(nave.getPosicion().x,nave.getPosicion().y,
 nave.getTamano().x,nave.getTamano().y);

if (Intersector.overlaps(rectangulo_alien, rectangulo_nave)){
Utiles.imprimirLog("UD2_5_RendererXogo", "RENDER", "CHOCA CON NAVE");
}
}
}
```

Comentarios:

- Liña 5: comproba se se pulsa a pantalla.
- LIñas 6-7: obtén a coordenada en **pixeles de pantalla** da última posición.
- Liñas 9-10: pasa de coordenadas de pantalla a coordenadas do noso mundo. Xa visto anteriormente.
- Liñas 12-13: obtén o alien e o move á posición pulsada.
- Liña 14: crea un rectángulo coa posición e o tamaño do alien.
- Liñas 17-19: obtén a nave e crea un rectángulo coa posición e o tamaño.
- Liñas 21-23: comproba se os dous rectángulos 'chocan' e manda unha mensaxe en caso afirmativo.

Exercicio proposto: poderíamos mover o Alien co dedo. Teríamos que utilizar a interface InputProcessor, ir ó evento adecuado e comprobar se estamos enriba do alien para facer que se move ás coordenadas indicadas polo evento (pasadas previamente a coordenadas do noso mundo).

Recomendación: No caso de xestionar controis de movemento, como frechas de tipo arriba-abaixo-esquerda-dereita, se pode utilizar a clase Circle para representar o dedo pulsado na pantalla e usar o método static boolean overlaps(Circle c, Rectangle r) para comprobar se se premeu a frecha.

Nota: Volvede a facer que a clase PantallaXogo chame á clase RendererXogo.

TAREFA 2.8.B A FACER: Esta parte está asociada á realización dunha tarefa.

1.1.3 Facer que o Alien se mova cos elementos móbiles

Despois de facer a tarefa 2.8 xa controlades ó Alien. Imos facer que cando pase por enriba dun elemento móbil o Alien se mova con él.

Seríades capaces de facelo ?

Como ven pensades temos que poñer o código na clase ControladorXogo e o imos facer no método controlarAlien de dita clase.

Agora temos que pensar se imos necesitar gardar algún valor novo para controlar este estado.

Unha posible solución pode ser a de gardar unha velocidade que se lle sume á velocidade que ten o propio alien. Dita velocidade se modifica en función de se sube a un elemento móbil. Agora xa sabemos como controlar cando o alien intersecciona cun elemento móbil.

Mans a obra.

Nota: eu vos dou unha posible solución pero cada un pode facelo como queira.

Código da clase Alien
Obxectivo: engadimos un campo novo de nome velocidadeMontado. Sumámoslle á velocidade do alien dito valor que será modificado cando o alien suba a un elemento móbil.

```
private float velocidadeMontado;
        ...........
public Alien(Vector2 posicion, Vector2 tamano, float velocidade_max) {
super(posicion, tamano, velocidade_max);

        velocidade = new Vector2(0,0);
setVelocidade_montado(0);
         }
         ...........
        public float getVelocidade_montado() {
return velocidadeMontado;
}

public void setVelocidade_montado(float velocidade_montado) {
this.velocidadeMontado = velocidade_montado;
}

public void update(float delta){
```

```
        setPosicion(getPosicion().x+(velocidade.x+velocidadeMontado)*delta,
                            getPosicion().y+velocidade.y*delta);

    }
```

Agora só queda modificar dito valor e darlle a velocidade do elemento móbil cando estea enriba del. Pero antes de facer isto imos a refinar un pouco o código. Se intentamos facer isto sen máis imos ter que crear dous rectángulos e darlles a posición do alien e a posición de cada un dos elementos móbiles, chamando á clase Intersector para comprobar se se tocan.

En vez disto, podemos modificar a clase Personaxe e crear unha propiedade rectángulo que se modifique cando se cambie de posición (o tamaño do rectángulo non varía).

Código da clase Personaxe
Obxectivo: creamos unha propiedade **rectangulo** de tipo Rectangle que vai gardar a posición actual e o tamaño do Personaxe. Usado pola clase Intersector.

```
import com.badlogic.gdx.math.Rectangle;
...........
public abstract class Personaxe {

private Rectangle rectangulo;

        /**
          * Constructor por defecto
        */
public Personaxe(){
rectangulo = new Rectangle();
}
public Personaxe(Vector2 posicion, Vector2 tamano, float velocidade_max) {
this.posicion = posicion;
this.tamano = tamano;
this.velocidade_max = velocidade_max;

rectangulo = new Rectangle(posicion.x,posicion.y,tamano.x,tamano.y);
}
        ...........

public void setTamanoRectangulo(float width,float height){
rectangulo.setWidth(width);
rectangulo.setHeight(height);
}
        ...........
/**
         * Actualiza a posición do rectángulo asociado á forma do gráfico
         *
         */
public void actualizarRectangulo(){
rectangulo.x=posicion.x;
rectangulo.y=posicion.y;
}

/**
         * Devolve o rectángulo asociado
         * @return rectangulo
         */
public Rectangle getRectangulo(){
return rectangulo;
}

        public void setPosicion(Vector2 posicion) {
            this.posicion = posicion;
            actualizarRectangulo();
        }

       /**
        * Modifica a posición
        *
        * @param x: nova posición x
        * @param y: nova posición y
        */
        public void setPosicion(float x, float y) {
```

```
            posicion.x = x;
            posicion.y = y;
    actualizarRectangulo();
        }

/**
         * Asina un novo tamano
         * @param tamano: o novo tamano.
         */
public void setTamano(Vector2 tamano) {
this.tamano=tamano;
setTamanoRectangulo(tamano.x,tamano.y);
}

public void setTamano(float width, float height) {
this.tamano.set(width,height);
setTamanoRectangulo(width, height);
}

        ...........
}
```

NOTA IMPORTANTE: O método update das clases que derivan da clase Personaxes (como ElementoMobil) debe chamar ó método setPosicion() para moverse:

```
@Override
public void update(float delta) {
        // TODO Auto-generated method stub
       setPosicion(posicion.add(velocidade*delta,0));
        }
```

NON DEBEMOS FACER ASI:

```
@Override
public void update(float delta) {
         // TODO Auto-generated method stub
    posicion.add((velocidade*delta),0);
        }
```

Agora só queda modifica a clase controladorXogo para ver cando coinciden e modificar a velocidadeMontado do alien. **Exercicio proposto:** Intentádeo facelo.

Posible solución:

Código da clase ControladorXogo
Obxectivo: Modificamos o método controlarAlien para controlar cando chocan os rectángulos do alien cos dos elementos móbiles.

```
private void controlarAlien(float delta){

// Actualiza Alien
alien.update(delta);


// Impide que se mova fora dos límites da pantalla
if (alien.getPosicion().x <=0){
alien.setPosicion(0, alien.getPosicion().y);
}
else {
if (alien.getPosicion().x >= Mundo.TAMANO_MUNDO_ANCHO-alien.getTamano().x){
alien.setPosicion(Mundo.TAMANO_MUNDO_ANCHO-alien.getTamano().x, alien.getPosicion().y);
}

}
```

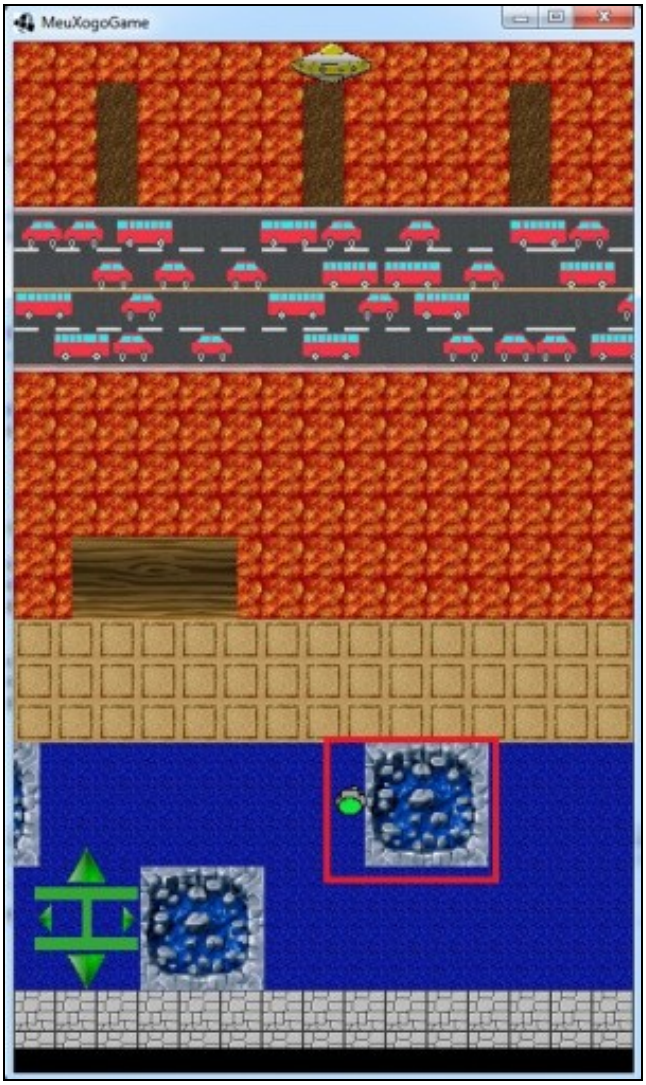
```
if (alien.getPosicion().y <=Controis.FONDO_NEGRO.height){
alien.setPosicion(alien.getPosicion().x,Controis.FONDO_NEGRO.height);
}
else {
if (alien.getPosicion().y >= Mundo.TAMANO_MUNDO_ALTO-alien.getTamano().y){
alien.setPosicion(alien.getPosicion().x, Mundo.TAMANO_MUNDO_ALTO-alien.getTamano().y);
}

}

// Controla que suba enriba dun elemento móvil
alien.setVelocidade_montado(0);
for (ElementoMobil elem : meuMundo.getRochas()){
if (Intersector.overlaps(elem.getRectangulo(), alien.getRectangulo())){
alien.setVelocidade_montado(elem.getVelocidade());
}
}
for (ElementoMobil elem : meuMundo.getTroncos()){
if (Intersector.overlaps(elem.getRectangulo(), alien.getRectangulo())){
alien.setVelocidade_montado(elem.getVelocidade());
}
}


}
```

Se vos fixades e xogades, o alien ten o rectángulo asociado demasiado grande e iso fai que teñamos este efecto:



Como podedes observar o alien está pegado á rocha e é arrastrado.

Para solucionalo podemos facer que o rectángulo do alien sexa máis pequeno (a mitade) pero tamén temos que sobreescribir o método que actualiza o rectángulo xa que o queremos centrado (se non tería a metade de tamaño pero estaría situado abaixo á esquerda.

Código da clase Alien
Obxectivo: axustar o tamaño e posición do rectángulo.

```
public class Alien extends Personaxe {

public Alien(Vector2 posicion, Vector2 tamano, float velocidade_max) {
super(posicion, tamano, velocidade_max);

velocidade = new Vector2(0,0);
setVelocidade_montado(0);

getRectangulo().setSize(tamano.x/2);
}
        ..............
@Override
public void actualizarRectangulo(){

getRectangulo().x = getPosicion().x+getTamano().x/4;
getRectangulo().y = getPosicion().y+getTamano().y/4;

}
       ..............

}
```

1.1.4 Controlar os choques dos coches có Alien

Controlar isto xa non ten problema ningún.

Código da clase ControladorXogo
Obxectivo: Xestionar os choques cos coches.

```
private void controlarAlien(float delta){
                ............

// Controla se lle colle un vehículo
for (ElementoMobil elem : meuMundo.getCoches()){
if (Intersector.overlaps(elem.getRectangulo(), alien.getRectangulo())){
                                  // ALIEN MORTO
}
        }

 }
```

1.1.5 Controlar cando o alien cae na lava ou na auga

Igual que fixemos antes a solución é bastante sinxela. O único complicado é descubrir as coordenadas e tamaño da auga e lava.

Podemos crear un array de zonas perigosas e comprobar se está dentro dunha delas. Tamén temos que definir as zonas seguras, xa que no lugar onde está a nave hai unhas pasarelas que son seguras e teremos que definilas. A definición de ditas zonas a podemos ter na clase Mundo utilizando un array de Rectangles.

Código da clase Mundo
Obxectivo: definimos nun array de Rectangle as zonas perigosas e seguras.
Nota: Estas zonas están definidas para un tamaño de mundo de 300 por 500.

```
public static final Rectangle ZONAS_PERIGOSAS[]={new Rectangle(0,40,300,120),new Rectangle(0,220,300,120), new Rectangle(0,420,300,80
```

```
public static final Rectangle ZONAS_SEGURAS[]={new Rectangle(40,420,20,60),new Rectangle(140,420,20,60), new Rectangle(240,420,20
```

Agora só temos que controlar se o alien pisa algunha delas, **PERO.....** temos que ter en conta cando está enriba dun tronco ou rocha para que non comprobe as zonas. Como podemos sabelo ? Pois mirando o valor da velocidade_montado do alien que cando está enriba dun tronco ou rocha ten un valor diferente a 0.

Código da clase ControladorXogo
Obxectivo: verificar se o alien está nunha zona segura ou perigosa.

```
private void controlarAlien(float delta){
                ..............

// Controla se cae a auga ou lava
if (alien.getVelocidade_montado()==0){
boolean seguro=false;
// Se está nunha zona segura xa non mira as perigosas
for(int cont=0; cont < Mundo.ZONAS_SEGURAS.length;cont++){
if (Intersector.overlaps(Mundo.ZONAS_SEGURAS[cont], alien.getRectangulo())){
seguro=true;
break;
}
}
if (!seguro){
for(int cont=0; cont < Mundo.ZONAS_PERIGOSAS.length;cont++){
if (Intersector.overlaps(Mundo.ZONAS_PERIGOSAS[cont], alien.getRectangulo())){
// ALIEN MORRE
}
}
}
}
```

1.1.6 Avanzando no noso xogo

Relacionados con a xestión de atropellos, lava e auga :) imos xestionar as vidas salvadas/perdidas do alien.

Unha proposta de solución:
A idea e ter nun array as vidas salvadas e mortas e debuxalas na parte inferior da pantalla.

Exercicio proposto: facer o anterior.

Posible solución:
Como sempre temos que pensar que información imos necesitar para engadir esta funcionalidade e onde gardala.

No noso caso dita información será gardada na clase Alien.

Código da clase Alien
Obxectivo: xestionar as vidas salvadas / perdidas.

```
import com.badlogic.gdx.utils.Array;
..........
public class Alien extends Personaxe {

public static enum TIPOS_VIDA{INICIAL,SALVADO,MUERTO};
private Array<TIPOS_VIDA>numVidas;
        ..............
public Alien(Vector2 posicion, Vector2 tamano, float velocidade_max) {
super(posicion, tamano, velocidade_max);

velocidade = new Vector2(0,0);
setVelocidade_montado(0);

getRectangulo().setSize(tamano.x/2);

numVidas = new Array<Alien.TIPOS_VIDA>();
}
        ..............
public Array<TIPOS_VIDA> getNumVidas() {
return numVidas;
```

```
}

public void setNumVidas(TIPOS_VIDA vida) {
numVidas.add(vida);
}

public int getNumVidasSalvadas(){
int num=0;
for (TIPOS_VIDA vida : numVidas){
if (vida == TIPOS_VIDA.SALVADO) num++;
        }

return num;
}
       ..............

}
```

Agora queda debuxar as vidas....como imos debuxalas nunha posición da pantalla podemos gardar dita posición na clase Controis.

Código da clase Controis

Obxectivo: Gardar a posición onde se debuxan as vidas.

```
public final static int POSVIDAS = 60;
```

Agora debuxamos as vidas na posición indicada na clase RendererXogo.

Código da clase RendererXogo

Obxectivo: Debuxar as vidas.

```
..............
private void debuxarVidas(){
Texture textura;
int posx=Controis.POSVIDAS;
for(Alien.TIPOS_VIDA vida : meuMundo.getAlien().getNumVidas()){
if(vida == Alien.TIPOS_VIDA.MUERTO)
textura = AssetsXogo.textureAlienDead;
else if(vida == Alien.TIPOS_VIDA.SALVADO)
textura = AssetsXogo.textureAlienRescue;
else
textura = AssetsXogo.textureAlien;

spritebatch.draw(textura,posx,0,10,10);
posx+=12;
}
}
        ..............
    public void render(float delta) {
Gdx.gl.glClearColor(0, 0, 0, 1);
Gdx.gl.glClear(GL20.GL_COLOR_BUFFER_BIT);

spritebatch.begin();

debuxarFondo();

debuxarNave();

debuxarCoches();
debuxarRochas();
debuxarTroncos();

debuxarAlien();

debuxarControis();
debuxarVidas();
spritebatch.end();
```

```
if (debugger) {
debugger();
}
    }
        ..............
```

Nota: É importante a orde de debuxo xa que se chamamos a debuxarVidas antes que a debuxarControis, o punto negro que conforma a banda inferior tapará as vidas.

E por último temos que modificar á clase ControladorXogo para engadir á vida / morte ó array de vidas en función de se chegamos á nave ou morremos polo camiño...

Código da clase ControladorXogo
Obxectivo: Xestionar as vidas.

```
private void controlarAlien(float delta){
                .............

// Controla se lle colle un vehículo
for (ElementoMobil elem : meuMundo.getCoches()){
if (Intersector.overlaps(elem.getRectangulo(), alien.getRectangulo())){
alien.setNumVidas(TIPOS_VIDA.MUERTO);
}
}

// Controla se cae a auga ou lava
if (alien.getVelocidade_montado()==0){
boolean seguro=false;
// Se está nunha zona segura xa non mira as perigosas
for(int cont=0; cont < Mundo.ZONAS_SEGURAS.length;cont++){
if (Intersector.overlaps(Mundo.ZONAS_SEGURAS[cont], alien.getRectangulo())){
seguro=true;
break;
}
}
if (!seguro){
for(int cont=0; cont < Mundo.ZONAS_PERIGOSAS.length;cont++){
if (Intersector.overlaps(Mundo.ZONAS_PERIGOSAS[cont], alien.getRectangulo())){
alien.setNumVidas(TIPOS_VIDA.MUERTO);
}
}
}
}

}
```

Como está agora encheremos de vidas mortas a parte inferior xa que non nos movemos e os coches seguen pasando por enriba.

Para evitalo imos a engadir á clase Alien un método para inicializar o alien cando morre.

Código da clase Alien
Obxectivo: inicializar o alien cando morre.

```
public void inicializarAlien(){
          setPosicion(100, 20);
          setVelocidade_montado(0);
          setVelocidadeX(0);
          setVelocidadeY(0);
          setTamano(15,15);
          getRectangulo().setSize(tamano.x/2);
       }
```

Código da clase ControladorXogo
Obxectivo: inicializar o alien cando morre.

```
private void controlarAlien(float delta){
                ..................
// Controla se lle colle un vehículo
for (ElementoMobil elem : meuMundo.getCoches()){
if (Intersector.overlaps(elem.getRectangulo(), alien.getRectangulo())){
alien.setNumVidas(TIPOS_VIDA.MUERTO);
alien.inicializarAlien();
}
}

// Controla se cae a auga ou lava
if (alien.getVelocidade_montado()==0){
boolean seguro=false;
// Se está nunha zona segura xa non mira as perigosas
for(int cont=0; cont < Mundo.ZONAS_SEGURAS.length;cont++){
if (Intersector.overlaps(Mundo.ZONAS_SEGURAS[cont], alien.getRectangulo())){
seguro=true;
break;
}
}
if (!seguro){
for(int cont=0; cont < Mundo.ZONAS_PERIGOSAS.length;cont++){
if (Intersector.overlaps(Mundo.ZONAS_PERIGOSAS[cont], alien.getRectangulo())){
alien.setNumVidas(TIPOS_VIDA.MUERTO);
alien.inicializarAlien();
}
}
}
}

        }
```

TAREFA 2.9 A FACER: Esta parte está asociada á realización dunha tarefa.

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