1 LIBGDX tarefa 2 6 Alternativa

Tal como comentaba anteriormente temos moitas formas de programar o xogo, algunhas máis óptimas que outras.

Normalmente canto máis flexible é a solución máis complexa é de facer, pero despois nos permite engadir novos elementos moito máis rápido.

Vos vou amosar unha posible solución ó problema dos elementos móbiles. Loxicamente non tedes que facela, simplemente queda como posible alternativa á forma que estamos a desenrolar, existindo outros métodos seguro que máis óptimos...

Código da clase ElementoMovil
Obxectivo: Representa calquera elemento móbil do xogo.

```
public class ElementoMobil extends Personaxe {

public static enum TIPOS_ELEMENTOS {COCHE, AUTOBUS, TRONCO, ROCA};
private TIPOS_ELEMENTOS tipo;

/**
         * Filas que se correponden con os elementos do xogo que se moven
         * @param fila: dende 0 ata 3 son para os coches; dende 4 a 6 son para os troncos
         */
public ElementoMobil(){

}
public ElementoMobil(Vector2 posicion, Vector2 tamano, float velocidade_max,TIPOS_ELEMENTOS tipo) {
super(posicion, tamano, velocidade_max);
this.setTipo(tipo);
// TODO Auto-generated constructor stub

velocidade=velocidade_max;
}

public TIPOS_ELEMENTOS getTipo() {
return tipo;
}
public void setTipo(TIPOS_ELEMENTOS tipo) {
this.tipo = tipo;
}


@Override
public void update(float delta) {
// TODO Auto-generated method stub

setPosicion(posicion.x+delta*velocidade,posicion.y);


}

}
```

Nesta clase non hai moito que comentar.

Agora creo unha nova clase que vai ter nun array todos os elementos móbiles.

Código da clase Elementos
Obxectivo: Representa todos os elementos móbiles do xogo.

```
public class Elementos {

private int posfilas[]={220,260,300,345,365,380,400,40,100};
private int velocidades[]={-35,95,-55,50,-45,35,-65,30,-50};
```

```
private Array<ElementoMobil> elementos;
private float tempoParaCrearMax;
private float tempoParaCrearMin;

private int numElementos;

private float crono;


public Elementos(float tempomin,float tempomax,int numelementos){
elementos = new Array<ElementoMobil>();
setTempoparacrearMax(tempoParaCrearMax);
crono = tempoParaCrearMax;
setTempoparacrearMin(tempomin);
setNumelementos(numelementos);
}

private int getFila(ElementoMobil.TIPOS_ELEMENTOS tipoelem){
int fila=0;

switch(tipoelem){
case AUTOBUS:
case COCHE:
fila = MathUtils.random(3,6);
break;
case ROCA:
fila = MathUtils.random(7,8);
break;
case TRONCO:
fila = MathUtils.random(0,2);
break;
default:
break;

}
return fila;

}

public void engadirElemento(ElementoMobil.TIPOS_ELEMENTOS tipoelem){
int fila=0;
fila = getFila(tipoelem);

ElementoMobil elemento = new ElementoMobil();

elemento.posicion = new Vector2();
elemento.tamano = new Vector2();

switch(tipoelem){
case AUTOBUS:
elemento.setTipo(TIPOS_ELEMENTOS.AUTOBUS);
elemento.setTamano(30,15);
break;
case COCHE:
elemento.setTipo(TIPOS_ELEMENTOS.COCHE);
elemento.setTamano(20,15);
break;
case ROCA:
elemento.setTipo(TIPOS_ELEMENTOS.ROCA);
elemento.setTamano(60,60);
break;
case TRONCO:
elemento.setTipo(TIPOS_ELEMENTOS.TRONCO);
elemento.setTamano(80,40);
break;
default:
break;

}


elemento.velocidade = velocidades[fila];
// Os pares sean da esquerda
```

```
if(velocidades[fila] >0)
elemento.setPosicion(-elemento.tamano.x, posfilas[fila]);
else
elemento.setPosicion(Mundo.TAMANO_MUNDO_ANCHO, posfilas[fila]);

for (ElementoMobil elem : elementos){
if (Intersector.overlaps(elem.getRectangulo(), elemento.getRectangulo())){
return;
}
}
elementos.add(elemento);


}

public void update(float delta){
crono-=delta;
for (ElementoMobil elem : elementos){
elem.update(delta);
}
for (ElementoMobil elem : elementos){
if (elem.velocidade<0){ // Vai cara a esquerda
if (elem.posicion.x <= -elem.tamano.x){
elementos.removeValue(elem, true);
break;
}
}
else{
if (elem.posicion.x >= Mundo.TAMANO_MUNDO_ANCHO){
elementos.removeValue(elem, true);
break;
}
}
}
}

public Array<ElementoMobil>getElementos(){
return elementos;
}

public float getCrono(){
return crono;
}
public void setCrono(float tempo){
crono=tempo;
}

public float getTempoparacrearMax() {
return tempoParaCrearMax;
}

public void setTempoparacrearMax(float tempoparacrear) {
this.tempoParaCrearMax = tempoparacrear;
}

public float getTempoparacrearMin() {
return tempoParaCrearMin;
}

public void setTempoparacrearMin(float tempoparacrearmin) {
this.tempoParaCrearMin = tempoparacrearmin;
}

public int getNumelementos() {
return numElementos;
}

public void setNumelementos(int numelementos) {
this.numElementos = numelementos;
}

}
```

Expliquemos previamente dita clase:

- Liñas 3-4: representan cada unha das filas de elementos móbiles. Indican a súa posición e velocidade. Lembrar que todos os elementos móbiles de cada liña teñen a mesma velocidade.
- Liña 7: array que ten todos os elementos móbiles de todas as filas. Por simplificar, este array vai levar todos os coches (un obxecto desta clase), todas as rochas (outro obxecto desta clase) e todos os troncos (outro obxecto desta clase).
- Liñas 8-9: esta clase vai crear aleatoriamente os elementos. Cando creamos un novo elemento xeramos un número aleatorio entre estes dous números, que será o tempo que terá que pasar ata instancialo.
- Liña 11: número máximo de elementos na pantalla. Por exemplo, podemos facer que non haia máis de 20 coches en pantalla (incluídas todas as filas).
- Liña 13: cronómetro usado para crear novos elementos.
- Liñas 24-44: método getFila: método que devolve unha fila de forma aleatoria onde se vai crear un novo elemento móbil. Depende do tipo (coche, tronco e rocha) teremos rango de filas diferentes.
- Liñas 46-93: método engadirElemento: engade un novo elemento. Xa falaremos da clase Intersector no punto de xestión das colisións.
- Liñas 95-114: método update: actualiza a posición de todos os elementos. Se chegan os limites eliminao do array.

Código da clase Mundo
Obxectivo: Engadimos os elementos móbiles.

```
private Elementos vehiculos;
    private Elementos troncos;
    private Elementos rocas;

    public Mundo(){
        alien = new Alien(new Vector2(100,20), new Vector2(15,15),100);
        nave = new Nave(new Vector2(0,480),new Vector2(40,20),60);

        vehiculos = new Elementos(1.5f,3f,20);
        troncos = new Elementos(2f,5f,6);
        rocas = new Elementos(2f,5f,3);
    }

public Elementos getVehiculos(){
return vehiculos;
}
public Elementos getTroncos(){
return troncos;
}
public Elementos getRocas(){
return rocas;
}
    ...................
```

- Estamos a engadir 20 coches, 6 troncos e 3 rochas. Se fai de forma aleatoria e a medida que se van eliminando do array se van engadindo novos elementos. Isto se fai na clase controladora.

Código da clase ControladorXogo
Obxectivo: Xestionamos os elementos móbiles.

```
private void controlarVehiculos(float delta){

// Actualizamos posición e elininamos
Elementos vehiculos = meuMundo.getVehiculos();
vehiculos.update(delta);

// Engadimos os vehiculos
 if (vehiculos.getCrono()<=0){
vehiculos.setCrono(MathUtils.random(vehiculos.getTempoparacrearMin(),vehiculos.getTempoparacrearMax()));
if (vehiculos.getElementos().size<vehiculos.getNumelementos()){
int rand = MathUtils.random(1);
ElementoMobil.TIPOS_ELEMENTOS tipo;
if (rand==0)
tipo = TIPOS_ELEMENTOS.AUTOBUS;
```

```
else
tipo = TIPOS_ELEMENTOS.COCHE;

vehiculos.engadirElemento(tipo);
vehiculos.engadirElemento(tipo);
}
}

// Actualizamos posición e elininamos
Elementos troncos = meuMundo.getTroncos();
troncos.update(delta);

// Engadimos os troncos
if (troncos.getCrono()<=0){
troncos.setCrono(MathUtils.random(troncos.getTempoparacrearMin(),troncos.getTempoparacrearMax()));
if (troncos.getElementos().size<troncos.getNumelementos()){
troncos.engadirElemento(TIPOS_ELEMENTOS.TRONCO);
}
}

// Actualizamos posición e elininamos
Elementos rocas = meuMundo.getRocas();
rocas.update(delta);

// Engadimos os troncos
if (rocas.getCrono()<=0){
rocas.setCrono(MathUtils.random(rocas.getTempoparacrearMin(),rocas.getTempoparacrearMax()));
if (rocas.getElementos().size<rocas.getNumelementos()){
rocas.engadirElemento(TIPOS_ELEMENTOS.ROCA);
}
}



}

public void update(float delta){

controlarVehiculos(delta);

}
```