LIBGDX Creando o mundo

UNIDADE 2: Creando o mundo

Sumario

O fondo de pantalla

Para empezar com algo fácil imos poñer o fondo de pantalla ó noso xogo. O gráfico xa o tedes no cartafol xa que viña na tarefa 2.3 co nome LIBGDX_itin1_fondoxogo.jpg.

Exercicio proposto: Debuxade o gráfico do fondo.

Solución. **Código da clase RendererXogo**
Obxectivo: Debuxar o fondo do xogo.

```
.............
private void debuxarFondo(){
spritebatch.draw(AssetsXogo.textureFondo,
                                0,0,Mundo.TAMANO_MUNDO_ANCHO,Mundo.TAMANO_MUNDO_ALTO);
}
        .............
        public void render(float delta) {
Gdx.gl.glClearColor(0, 0, 0, 1);
Gdx.gl.glClear(GL20.GL_COLOR_BUFFER_BIT);

spritebatch.begin();

debuxarFondo();

debuxarAlien();
debuxarNave();

debuxarCoches();

spritebatch.end();

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

Os inimigos: Elementos móbiles parte I

Nota: Agora entramos mais en temas de programación que do propio framework. Neste punto teremos como límite a imaxinación.

Temos que ter en conta varias consideracións:

- Á hora de afrontar este tipo de inimigos que desaparecen e aparecen polo lado contrario temos dúas opcións:
  - ◊ Eliminar o inimigo de array cando chega ó final e crear un novo.
  - ◊ Mover o inimigo de volta á posición inicial.

A segunda opción é máis sinxela. Imos a utilizar esta opción para os coches e para as rochas usaremos a primeira para ver como se eliminan os obxectos do array.

- Ademais temos que tomar a decisión de se o número de inimigos (coches-autobuses) é sempre o mesmo ou ben o modificamos de forma aleatoria, de tal forma que poden aumentar ou diminuír. Nos imos a escoller a opción máis sinxela. Se algún alumno quere facer a complexa pode facelo.

A forma máis sinxela de crear os inimigos é engadilos un a un na clase Mundo.

```
public class Mundo {
        .............
 public final static Vector2 TAMANO_COCHES = new Vector2(20,15);
 public final static Vector2 TAMANO_AUTOBUSES = new Vector2(30,15);
 public final static Vector2 TAMANO_ROCA = new Vector2(60,60);
 public final static Vector2 TAMANO_TRONCO = new Vector2(80,40);

        .............

 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),60f);

 coches = new Array<ElementoMobil>();
 coches.add(new ElementoMobil(new Vector2(0,345),TAMANO_AUTOBUSES.cpy(),50,ElementoMobil.TIPOS_ELEMENTOS.AUTOBUS));
 coches.add(new ElementoMobil(new Vector2(70,345),TAMANO_COCHES.cpy(),50,ElementoMobil.TIPOS_ELEMENTOS.COCHE));
 coches.add(new ElementoMobil(new Vector2(95,345),TAMANO_COCHES.cpy(),50,ElementoMobil.TIPOS_ELEMENTOS.COCHE));
 coches.add(new ElementoMobil(new Vector2(115,345),TAMANO_COCHES.cpy(),50,ElementoMobil.TIPOS_ELEMENTOS.COCHE));
 coches.add(new ElementoMobil(new Vector2(140,345),TAMANO_AUTOBUSES.cpy(),50,ElementoMobil.TIPOS_ELEMENTOS.AUTOBUS));
 coches.add(new ElementoMobil(new Vector2(210,345),TAMANO_AUTOBUSES.cpy(),50,ElementoMobil.TIPOS_ELEMENTOS.AUTOBUS));
 coches.add(new ElementoMobil(new Vector2(240,345),TAMANO_COCHES.cpy(),50,ElementoMobil.TIPOS_ELEMENTOS.COCHE));
 coches.add(new ElementoMobil(new Vector2(278,345),TAMANO_COCHES.cpy(),50,ElementoMobil.TIPOS_ELEMENTOS.COCHE));

 coches.add(new ElementoMobil(new Vector2(40,380),TAMANO_COCHES.cpy(),35,ElementoMobil.TIPOS_ELEMENTOS.COCHE));
 coches.add(new ElementoMobil(new Vector2(70,380),TAMANO_COCHES.cpy(),35,ElementoMobil.TIPOS_ELEMENTOS.COCHE));
 coches.add(new ElementoMobil(new Vector2(105,380),TAMANO_COCHES.cpy(),35,ElementoMobil.TIPOS_ELEMENTOS.COCHE));
 coches.add(new ElementoMobil(new Vector2(150,380),TAMANO_AUTOBUSES.cpy(),35,ElementoMobil.TIPOS_ELEMENTOS.AUTOBUS));
 coches.add(new ElementoMobil(new Vector2(180,380),TAMANO_AUTOBUSES.cpy(),35,ElementoMobil.TIPOS_ELEMENTOS.AUTOBUS));
 coches.add(new ElementoMobil(new Vector2(220,380),TAMANO_COCHES.cpy(),35,ElementoMobil.TIPOS_ELEMENTOS.COCHE));
 coches.add(new ElementoMobil(new Vector2(265,380),TAMANO_AUTOBUSES.cpy(),35,ElementoMobil.TIPOS_ELEMENTOS.AUTOBUS));

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

}
```

Ó principio definimos o tamaño que queremos que teñan os diferentes elementos (pódense cambiar) e despois engadimos ó array de coches un a un cada un dos coches/autobuses que aparece en pantalla.

Fixarse que chamamos á funcion cpy para crear unha copia do vector tamaño e mandala ó constructor.

Esta sería a forma máis sinxela.

Agora temos que controlar que cando o coche chegue ó extremo, este volva ó principio.

Dito código o teremos que poñer na clase ControladorXogo.

Código da clase ControladorXogo
Obxectivo: controlar cando o coche/autobús chega ó extremo.

```
private void controlarCoches(float delta){

for(ElementoMobil coche: meuMundo.getCoches()){
coche.update(delta);
```

```
if (coche.getPosicion().x>=Mundo.TAMANO_MUNDO_ANCHO){
coche.setPosicion(-40, coche.getPosicion().y);
}
}

}
```

Agora queda por poñer os coches que *veñen en sentido contrario*.

Imos engadir ó noso mundo dous liñas de coches:

Código da clase Mundo

```
..............
 coches.add(new ElementoMobil(new Vector2(0,400),TAMANO_AUTOBUSES.cpy(),-45,ElementoMobil.TIPOS_ELEMENTOS.AUTOBUS));
 coches.add(new ElementoMobil(new Vector2(30,400),TAMANO_COCHES.cpy(),-45,ElementoMobil.TIPOS_ELEMENTOS.COCHE));
 coches.add(new ElementoMobil(new Vector2(70,400),TAMANO_COCHES.cpy(),-45,ElementoMobil.TIPOS_ELEMENTOS.COCHE));
 coches.add(new ElementoMobil(new Vector2(95,400),TAMANO_COCHES.cpy(),-45,ElementoMobil.TIPOS_ELEMENTOS.COCHE));
 coches.add(new ElementoMobil(new Vector2(115,400),TAMANO_COCHES.cpy(),-45,ElementoMobil.TIPOS_ELEMENTOS.COCHE));
 coches.add(new ElementoMobil(new Vector2(140,400),TAMANO_AUTOBUSES.cpy(),-45,ElementoMobil.TIPOS_ELEMENTOS.AUTOBUS));
 coches.add(new ElementoMobil(new Vector2(210,400),TAMANO_AUTOBUSES.cpy(),-45,ElementoMobil.TIPOS_ELEMENTOS.AUTOBUS));
 coches.add(new ElementoMobil(new Vector2(240,400),TAMANO_COCHES.cpy(),-45,ElementoMobil.TIPOS_ELEMENTOS.COCHE));
 coches.add(new ElementoMobil(new Vector2(278,400),TAMANO_COCHES.cpy(),-45,ElementoMobil.TIPOS_ELEMENTOS.COCHE));
                .............
 coches.add(new ElementoMobil(new Vector2(0,365),TAMANO_COCHES.cpy(),-65,ElementoMobil.TIPOS_ELEMENTOS.COCHE));
 coches.add(new ElementoMobil(new Vector2(70,365),TAMANO_AUTOBUSES.cpy(),-65,ElementoMobil.TIPOS_ELEMENTOS.AUTOBUS));
 coches.add(new ElementoMobil(new Vector2(115,365),TAMANO_COCHES.cpy(),-65,ElementoMobil.TIPOS_ELEMENTOS.COCHE));
 coches.add(new ElementoMobil(new Vector2(140,365),TAMANO_AUTOBUSES.cpy(),-65,ElementoMobil.TIPOS_ELEMENTOS.AUTOBUS));
 coches.add(new ElementoMobil(new Vector2(240,365),TAMANO_COCHES.cpy(),-65,ElementoMobil.TIPOS_ELEMENTOS.COCHE));
 coches.add(new ElementoMobil(new Vector2(278,365),TAMANO_AUTOBUSES.cpy(),-65,ElementoMobil.TIPOS_ELEMENTOS.AUTOBUS));

                .............
```

Estes coches irán de dereita a esquerda. Como o facemos ?

Temos dous problemas a ter en conta:

- A parte controladora: xa que agora os coches deben aparecer na parte dereita cando cheguen ó final. O tema do movemento xa está solucionado, xa que se vos fixades, **a velocidade dos coches engadidos ó noso mundo E NEGATIVA**.
- A parte visual, xa que teremos que voltear a imaxe.

Parte conntroladora:

Código da clase ControladorXogo
Obxectivo: controlar cando o coche/autobús chega ó extremo esquerdo ou dereito dependendo da velocidade.

```
private void controlarCoches(float delta){

for(ElementoMobil coche: meuMundo.getCoches()){
coche.update(delta);
if (coche.getVelocidade()>0){// Vai de esquerda a dereita
if (coche.getPosicion().x>=Mundo.TAMANO_MUNDO_ANCHO){
coche.setPosicion(-Mundo.TAMANO_AUTOBUSES.x, coche.getPosicion().y);
}
}
else{// Vai de dereita a esquerda
if (coche.getPosicion().x<=-coche.getTamano().x){
coche.setPosicion(Mundo.TAMANO_MUNDO_ANCHO, coche.getPosicion().y);
}
}
        }
}
```

Varias consideracións:

- Se imos de esquerda a dereita, cando chegue ó final (coche.getPosicion().x>=Mundo.TAMANO_MUNDO_ANCHO) o posicionamos na parte esquerda da pantalla pero fora da visión do xogador, para facer que vaia aparecendo. Dos dous elementos móbiles, o máis grande é o autobús e por iso os poñemos nesa posición en negativo (para que quede fora da pantalla).
- Se imos de dereita a esquerda temos que ter en conta que o coche é máis pequeno que o autobús e polo tanto cando o trasladamos á posición inicial (á parte dereita) temos que movelo un pouco máis a dereita que o autobús xa que se non en cada pasada o coche se irá achegando máis ó autobús.

O podedes probar có código anterior.

Para solucionalo:
Código da clase ControladorXogo
Obxectivo: controlar cando o coche/autobús chega ó extremo esquerdo ou dereito dependendo da velocidade.

```
private void controlarCoches(float delta){

for(ElementoMobil coche: meuMundo.getCoches()){
coche.update(delta);
if (coche.getVelocidade()>0){// Vai de esquerda a dereita
if (coche.getPosicion().x>=Mundo.TAMANO_MUNDO_ANCHO){
coche.setPosicion(-Mundo.TAMANO_AUTOBUSES.x, coche.getPosicion().y);
}
}
else{// Vai de dereita a esquerda
if (coche.getPosicion().x<=-coche.getTamano().x){
if (coche.getTipo()==ElementoMobil.TIPOS_ELEMENTOS.COCHE)// E un coche enton necesitamos situalo un pouco a dereita se non vai pisan
coche.setPosicion(Mundo.TAMANO_MUNDO_ANCHO+Mundo.TAMANO_AUTOBUSES.x-Mundo.TAMANO_COCHES.x, coche.getPosicion().y);
else
coche.setPosicion(Mundo.TAMANO_MUNDO_ANCHO, coche.getPosicion().y);
}
}
}

}
```

Parte de renderizado: Para solucionar este problema temos varias aproximacións:

◊ Ter outra textura cargada na clase AssetsXogo que sexa a mesma pero volteada.
◊ Utilizar a clase TextureRegion. Esta clase a volveremos a ver na sección de animación en programación avanzada.

Dita clase posúe o método flip que serve para 'voltear' a imaxe nun ou nos dous eixes x,y.

Un exemplo sería:

```
TextureRegion texreg = new TextureRegion(textura);
texreg.flip(true, false);
spritebatch.draw(texreg,coche.getPosicion().x,coche.getPosicion().y,coche.getTamano().x,coche.getTamano().y);
```

Nota: Lembrar que os new´s debe intentar evitarse dentro do render. É mellor ten un único new cando instanciemos a clase e chamar ó método setRegion e asinarlle a textura que queiramos rotar.

◊ Utilizando o método draw clase SpriteBatch.

◊ Poñendo unha velocidade negativa. Desta forma os coches se moverán de dereita a esquerda. Pero temos que ter en conta que desta forma a posición 'visual' dos coches / autobuses non coinciden coa posición 'real' no noso mundo. O podemos comprobar se activamos o debuger na clase **RendererXogo** e debuxamos os coches.

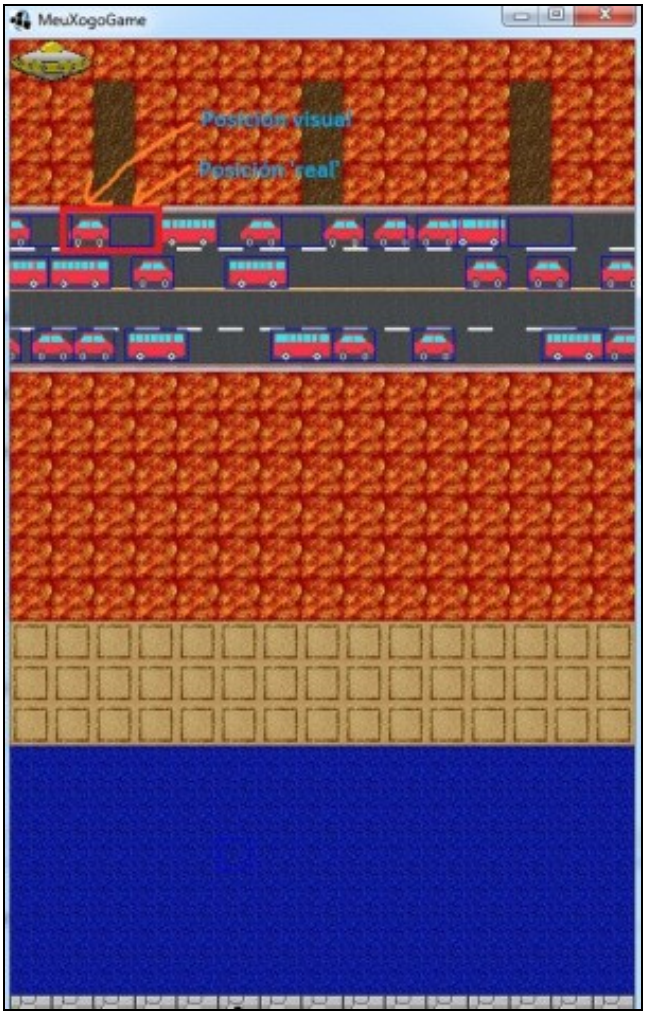
Código da clase RendererXogo
Obxectivo: Debuxar os coches que veñen de dereita a esquerda.

```
private void debuxarCoches(){
Texture textura=null;

for (ElementoMobil coche : meumundo.getCoches()){
switch(coche.getTipo()){
case COCHE:
textura = AssetsXogo.textureCoche;
```

```
break;
default:
textura = AssetsXogo.textureAutobus;
break;
}
if (coche.getVelocidade()<0){
spritebatch.draw(textura,coche.getPosicion().x,coche.getPosicion().y,-coche.getTamano().x,coche.getTamano().y);
}
else{
spritebatch.draw(textura,coche.getPosicion().x,coche.getPosicion().y,coche.getTamano().x,coche.getTamano().y);
}


}
}
```

```
private void debugger(){

            shaperender.begin(ShapeType.Line);
            shaperender.setColor(Color.BLUE);
            for (ElementoMobil coche : meumundo.getCoches()){
                shaperender.rect(coche.getPosicion().x,coche.getPosicion().y,coche.getTamano().x,coche.getTamano()
            }
            shaperender.end();

    }
```

Dará como resultado isto:



Para solucionalo só temos que desprazar o ancho do coche/autobús á súa posición.

Código da clase RendererXogo
Obxectivo: Facer coincidir a posición real coa posición visual.

```
private void debuxarCoches(){
Texture textura=null;

for (ElementoMobil coche : meumundo.getCoches()){
switch(coche.getTipo()){
case COCHE:
textura = AssetsXogo.textureCoche;
break;
default:
textura = AssetsXogo.textureAutobus;
break;
}
if (coche.getVelocidade()<0){
spritebatch.draw(textura,coche.getPosicion().x+coche.getTamano().x,coche.getPosicion().y,-coche.getTamano().x,coche.getTamano().y);
}
else{
spritebatch.draw(textura,coche.getPosicion().x,coche.getPosicion().y,coche.getTamano().x,coche.getTamano().y);
}


}
}
```

Fixarse como na liña 14 engadimos á posición do coche/autobús o seu tamaño.
NOTA: No desenvolvemento do xogo os alumnos poden escoller a solución que crean máis conveniente.

Engadimos agora a outra liña de coches que faltaba:

Código da clase Mundo

```
coches.add(new ElementoMobil(new Vector2(0,365),TAMANO_COCHES.cpy(),-65,ElementoMobil.TIPOS_ELEMENTOS.COCHE));
 coches.add(new ElementoMobil(new Vector2(70,365),TAMANO_AUTOBUSES.cpy(),-65,ElementoMobil.TIPOS_ELEMENTOS.AUTOBUS));
 coches.add(new ElementoMobil(new Vector2(115,365),TAMANO_COCHES.cpy(),-65,ElementoMobil.TIPOS_ELEMENTOS.COCHE));
 coches.add(new ElementoMobil(new Vector2(140,365),TAMANO_AUTOBUSES.cpy(),-65,ElementoMobil.TIPOS_ELEMENTOS.AUTOBUS));
 coches.add(new ElementoMobil(new Vector2(240,365),TAMANO_COCHES.cpy(),-65,ElementoMobil.TIPOS_ELEMENTOS.COCHE));
 coches.add(new ElementoMobil(new Vector2(278,365),TAMANO_AUTOBUSES.cpy(),-65,ElementoMobil.TIPOS_ELEMENTOS.AUTOBUS));
```

Loxicamente podedes engadir ou quitar coches para facelo máis fácil ou máis complicado :).

TAREFA 2.6 A FACER: Esta parte está asociada á realización dunha tarefa.

Os inimigos: Elementos móbiles parte II

Neste punto teredes movéndose os coches e as rochas polo voso xogo. Agora imos facer unha pequena modificación que vai consistir en eliminar os elementos móbiles do array e volver a engadilos. Poderíamos engadilos nunha posición aleatoria para facelo máis complicado e tamén poderíamos modificar a velocidade. Quen quera facelo pode (xa comentado como solución alternativa na tarefa 2.6). Para facelo máis sinxelo na explicación que ven a continuación só imos traballar con tres troncos, un por cada fila.

O proceso é o mesmo que nos coches e rochas. Só temos que cambiar ó método da clase ControladorXogo.

Código da clase Mundo
Obxectivo: creamos os troncos.

```
..............
 private Array<ElementoMobil>troncos;
 public Mundo(){
                ..............
 troncos = new Array<ElementoMobil>();
 troncos.add(new ElementoMobil(new Vector2(100,220),TAMANO_TRONCO.cpy(),-50,ElementoMobil.TIPOS_ELEMENTOS.TRONCO));
 troncos.add(new ElementoMobil(new Vector2(60,260),TAMANO_TRONCO.cpy(),40,ElementoMobil.TIPOS_ELEMENTOS.TRONCO));
```

```
 troncos.add(new ElementoMobil(new Vector2(150,300),TAMANO_TRONCO.cpy(),-70,ElementoMobil.TIPOS_ELEMENTOS.TRONCO));
                  ..............
         }
public Array<ElementoMobil> getTroncos() {
return troncos;
}

         ..............
```

Código da clase RendererXogo
Obxectivo: debuxamos os troncos.

```
..............
private void debuxarTroncos(){
for (ElementoMobil tronco : meumundo.getTroncos()){
spritebatch.draw(AssetsXogo.textureTronco,tronco.getPosicion().x,tronco.getPosicion().y,tronco.getTamano().x,tronco.getTamano().y);
}

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

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

spritebatch.begin();

debuxarFondo();

debuxarAlien();
debuxarNave();

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


spritebatch.end();
                ..............
    }
```

Agora chega a diferenza que é eliminar os elementos do array na clase ControladorXogo cando as rochas cheguen os límites.

Código da clase ControladorXogo
Obxectivo: actualiza e elimina/creo os troncos ó chegar ós limites.

```
..............
private void controlarTroncos(float delta){
for(ElementoMobil tronco: meuMundo.getTroncos()){
tronco.update(delta);
if (tronco.getPosicion().x<-Mundo.TAMANO_TRONCO.x){
meuMundo.getTroncos().add(new ElementoMobil(new Vector2(MathUtils.random(Mundo.TAMANO_MUNDO_ANCHO, Mundo.TAMANO_MUNDO_ANCHO+Mundo.TAM
meuMundo.getTroncos().removeValue(tronco, true);

}
if (tronco.getPosicion().x>Mundo.TAMANO_MUNDO_ANCHO){
meuMundo.getTroncos().add(new ElementoMobil(new Vector2(MathUtils.random(-2*Mundo.TAMANO_TRONCO.x,-Mundo.TAMANO_TRONCO.x),
tronco.getPosicion().y),Mundo.TAMANO_TRONCO.cpy(),tronco.getVelocidade(),ElementoMobil.TIPOS_ELEMENTOS.TRONCO));
meuMundo.getTroncos().removeValue(tronco, true);
}

}


}
public void update(float delta) {

controlarCoches(delta);
```

```
controlarRochas(delta);
controlarTroncos(delta);

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

Comentamos o código:

- Liña 5: comprobamos se a rocha alcanza o límite (ven de dereita a esquerda).
- Liña 6: se é o caso, creamos unha nova rocha antes de eliminala xa que necesitamos darlle a posición Y que ten a rocha e tamén a velocidade. Neste exemplo non se xera unha velocidade aleatoria quedando como posible exercicio. Como se ve, cando fai o new da posición temos o seguinte código:

  ```
  new Vector2(MathUtils.random(Mundo.TAMANO_MUNDO_ANCHO,
  Mundo.TAMANO_MUNDO_ANCHO+Mundo.TAMANO_TRONCO.x)
  ```

  Isto xera un número aleatorio entre o tamano de mundo no ancho e o mesmo tamano máis o ancho do tronco. Desta forma xeramos unha posición aleatoria no eixe x.

- Liñas 10-12: o mesmo pero para o caso de que a rocha vaia de esquerda a dereita.

Lembrade que xa vimos como eliminar un elemento do array.

A nave espacial

Chega o momento de mover a nosa nave.

Como sempre, os pasos son:

- Cargar a textura na clase AssetsXogo.
- Crear a clase que derive de Personaxe e que terá toda a información da nave. Chamarémoslle Nave.
- Crear o obxecto nave na clase Mundo e o método get asociado.
- Debuxar a nave na clase Renderer.
- Controlar a nave na clase ControladorXogo. Aquí imos facer unha pequena trampa (é optativo facelo) e vou pasar o código que fai que a nave se pare en cada plataforma á clase Nave ó seu método update.

Empecemos:

A primeira parte xa está feita da tarefa 2.3.

Creamos a clase que deriva de Personaxe. O facemos no paquete modelo.

Código da clase Nave

```
public class Nave extends Personaxe {

public Nave(Vector2 posicion, Vector2 tamano, float velocidade_max) {
super(posicion, tamano, velocidade_max);
velocidade=velocidade_max;
}

@Override
public void update(float delta) {
setPosicion(getPosicion().x+delta*velocidade, getPosicion().y);

// Se chega ó final do recorrido cambiamos de dirección
if (posicion.x >=Mundo.TAMANO_MUNDO_ANCHO-getTamano().x){
setPosicion(Mundo.TAMANO_MUNDO_ANCHO-getTamano().x,getPosicion().y);
velocidade=-1*velocidade;
} else if (posicion.x<=0) {
setPosicion(0,getPosicion().y);
velocidade=-1*velocidade;
```

```
}
}

}
```

Varios comentarios:

- Liña 5: a nave dende o comenzo está movéndose, polo que dende o principio ten unha velocidade.
- Liñas 13-19: cando a nave chega os límites temos que facer que se mova en dirección contraria. O conseguimos multiplicando por -1 a súa velocidade.

  Imaxinemos que a velocidade da nave é positiva, por exemplo 50. Isto quere dicir que a nave vai de esquerda a dereita. Cando chegue ó final da pantalla multiplicamos por -1 e teremos unha velocidade de -50 e se moverá de dereita a esquerda. O mesmo na outra dirección. Por que temos asinada unha posición nas liñas 14 e 17 ? Imaxinemos o caso que a nave vai de dereita a esquerda e chega a -0.8f. Lembrade que en cada iteración restamos delta * velocidade e case nunca vai dar 0 exacto. Por iso o límite sempre ten que estar feito con >= ou <=. Neste caso cambiaríamos de velocidade a positivo e imaxinemos que a nave se move a -0.3f. Como vemos segue sendo negativa e polo tanto volvería a entrar no if do límite alcanzado. Volvería a cambiar de velocidade e volvería a ir cara a esquerda.

Crear o obxecto nave na clase Mundo e o método get asociado. Xa feito anteriormente (tarefa 2.4).

Debuxar a nave na clase Renderer. Xa feito anteriormente (tarefa 2.4).

Controlar a nave na clase ControladorXogo

Para mover a nave só temos que chamar ó método update.

Código da clase ControladorXogo
Obxectivo: mover a nave.

```
..............
private void controlarNave(float delta){
meuMundo.getNave().update(delta);
}

public void update(float delta) {

controlarCoches(delta);
controlarRochas(delta);
controlarTroncos(delta);

controlarNave(delta);

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

Podedes probar como a nave se move nos límites da pantalla.

Agora falta que a nave se quede parada, digamos 3 segundos en cada unha das plataformas. Vos atrevedes a facelo ?

Pistas:

- As posición en que debe parar a nave son: entre 33-35; entre 133-135; entre 227-229 (lembrar que non podemos poñer unha condición de 'posición nave = 33' xa que nunca vai dar o número exacto.
- A idea é que cando a nave chegue a algunha destas posicións paremos a nave. Como ? Temos varias opcións, poñendo a velocidade a 0 ou utilizar unha variable booleana que indique cando debe estar parada. Cando a variable valga true facemos que non entre no código que que cambia a posición en función da velocidade.
- Cando saibamos que debe estar parada, poñeremos en marcha un cronómetro (utilizar delta) e cando chegue a 3 segundos (ou pode ir dende 3 a 0) debemos de cambiar a variable booleana para que xa se poda mover a nave.

Posible solución:

Código da clase Nave
Obxectivo: facer que a nave se pare 3 segundos en cada plataforma.

```
public class Nave extends Personaxe {

private final int TEMPO_MOVENDOSE=3;
/**
         * Leva o tempo que está parado. Cando chegue a 0 volve a moverse.
         */
private float tempo;

private boolean parado;

public Nave(Vector2 posicion, Vector2 tamano, float velocidade_max) {
super(posicion, tamano, velocidade_max);
// TODO Auto-generated constructor stub
velocidade=velocidade_max;
parado=false;
tempo=0;
}

@Override
public void update(float delta) {
// TODO Auto-generated method stub
if ((posicion.x > 33 && posicion.x < 35) ||
(posicion.x > 133 && posicion.x < 135) ||
(posicion.x > 227 && posicion.x < 229))
{
parado=true;
}
if (parado){
tempo+=delta;
if (tempo > TEMPO_MOVENDOSE){
tempo=0;
parado=false;
}
}
if (!parado){
setPosicion(getPosicion().x+delta*velocidade, getPosicion().y);

// Se chega ó final do recorrido cambiamos de dirección
if (posicion.x >=Mundo.TAMANO_MUNDO_ANCHO-getTamano().x){
setPosicion(Mundo.TAMANO_MUNDO_ANCHO-getTamano().x,getPosicion().y);
velocidade=-1*velocidade;
} else if (posicion.x<=0) {
setPosicion(0,getPosicion().y);
velocidade=-1*velocidade;
}
}
}

}
```