1 LIBGDX MVC

UNIDADE 2: Modelo-Vista-Controlador

1.1 Modelo-Vista-Controlador

A miña idea inicial era non facer esta parte obrigatoria e poñer todo o control do xogo na clase PantallaXogo, facendo optativo o que vou explicar a continuación, pero resultaría demasiado complicado de manter as dúas opcións nun curso a distancia xa que o sitio onde van ir os métodos variaría.

A idea é crear unha clase Controladora que vai mover e controlar todo o que pasa no noso xogo. Quen non queira facer esta parte tería que implementar todos os métodos na clase PantallaXogo, pero eu recomendo facelo como vou explicar a continuación.

Empecemos:

Tal cal fixemos no exercicio anterior poderíamos establecer o movemento de todos os personaxes, pero nos estamos a seguir un modelo no que separamos o control (mover os gráficos) da súa visualización (render), e desta forma non o faríamos.

O concepto é moi sinxelo.

Dividimos o xogo en tres partes:

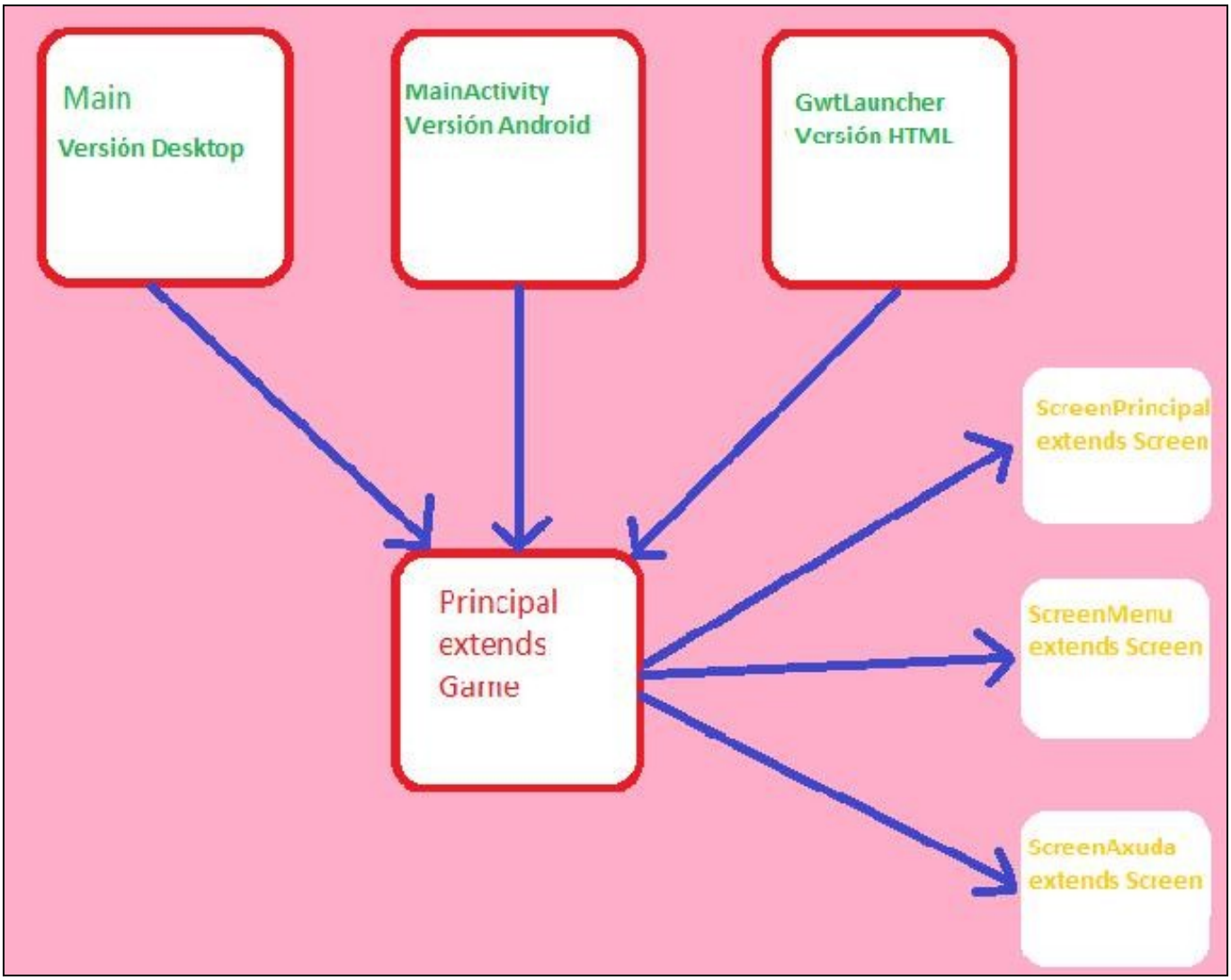
- ◊ Modelo: onde están definidas as clases que nos serven de base para o noso mundo. Isto xa o estamos a facer agora. Serían as clases Mundo, Personaxe, Piexe, Gancho, Pescador.
- ◊ Vista: aquí só se debuxan os obxectos que conforman o noso mundo e a cámara en base as súas propiedades de posición e tamaño.
- ◊ Controlador: controla todo o que sucede no noso mundo e modifica as propiedades dos obxectos que pertencen ó Modelo.

A maiores imos ter unha zona de nome Pantallas que serán cada unha das pantallas do noso xogo e que terán as interfaces que permiten xestionar os eventos e que informarán á clase controladora de que evento se trata.

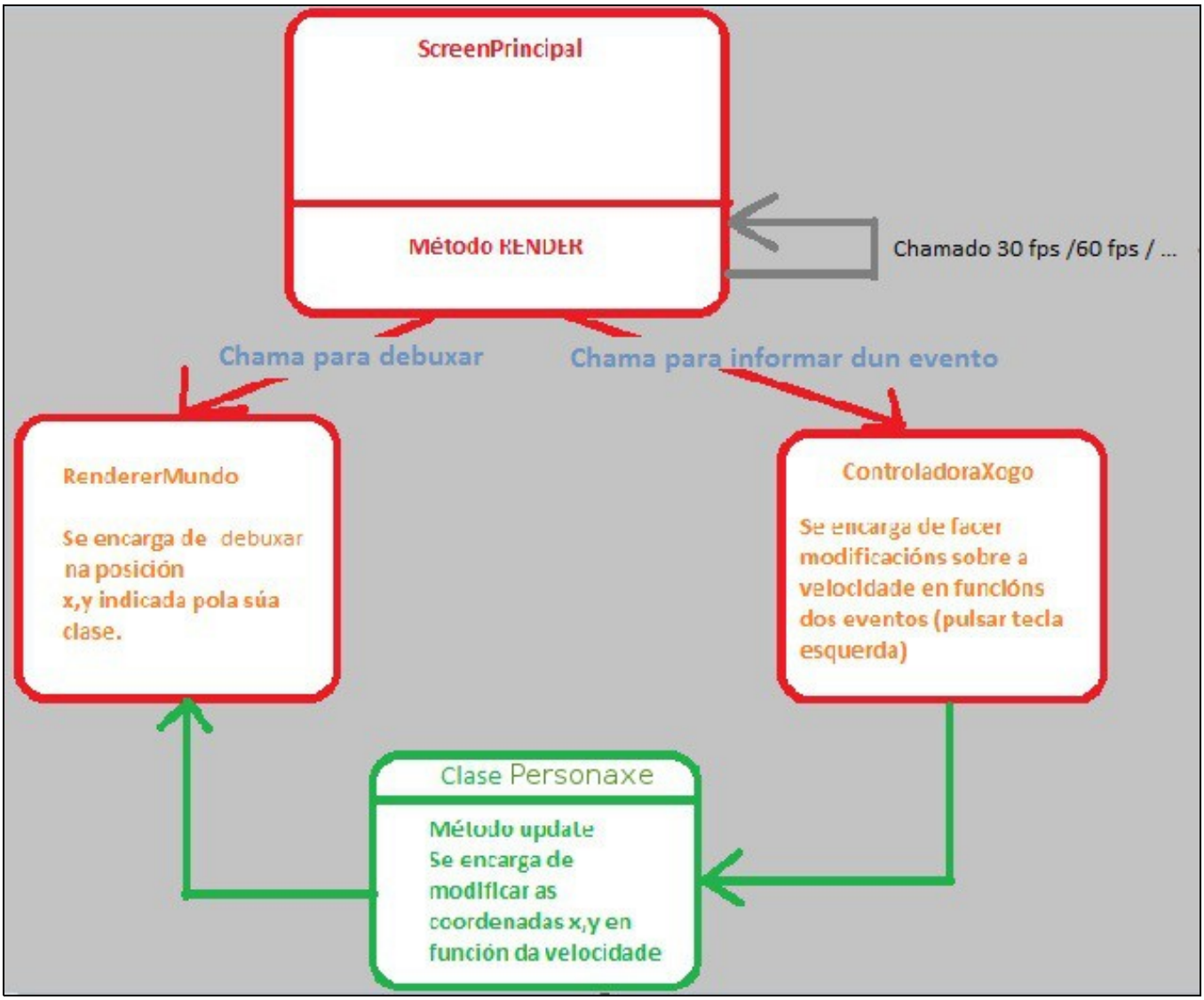
Polo tanto imos ter:

- ◊ Paquete modelo: onde estarán as clases que nos van servir para gardar a información de cada personaxe do noso xogo. Aquí estará a clase Mundo e unha clase por cada tipo de personaxe que conforme o noso xogo (incluído disparos, plataformas,...).
- ◊ Paquete renderer: clase que vai visualizar os personaxes do noso mundo. Só vai debuxar, non vai ningún tipo de control.
- ◊ Paquete pantalla: clases que derivan da clase Screen e que van a implementar as interface´s que nos van permitir iteracionar co xogo (pulsar a pantalla e recoller dito evento) e chamar á clase Controladora informando de dito evento.
- ◊ Paquete controlador: onde se atopan as clases que van encargarse de controlar todos os eventos (choque entre dous personaxes, disparo que alcanza a un inimigo,...) e recoller a información sobre a iteración do usuario coa pantalla dende a clase que se atopa no paquete pantalla.

Graficamente será algo parecido a isto:



Agora por cada pantalla:



Aplicando estes conceptos ó noso xogo:

- Creamos o paquete **com.plategaxogo2d.renderer** e movemos a clase **RendererXogo** a dito paquete.
- Creamos o paquete **com.plategaxogo2d.controlador**.
- Dentro de dito paquete **creamos a clase ControladorXogo**.
- Modificamos o constructor de dita clase para pasarlle un obxecto da Mundo xa que o imos necesitar para controlar os personaxes do noso mundo.
- Creamos un método update cun parámetro delta que vai ser chamado pola clase PantallaXogo de forma continua.
- Creamos dentro da clase PantallaXogo un obxecto da clase controladora (ó igual que fixemos coa renderer) e chamamos ó método update.

Traducido a código, creremos unha nova clase de nome **ControladorXogo** nun paquete novo de nome **com.plategaxogo2d.controlador**:

Código da clase ControladorXogo

```
package com.plategaxogo2d.controlador;

import com.plategaxogo2d.modelo.Mundo;

public class ControladorXogo {
Mundo meuMundo;

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

 }
```

```
/**
         * Vai chamar a todos os métodos para mover e controlar os personaxes
         * Tamén xestionará os eventos producidos polo usuario e que veñen dende a clase PantallaXogo
         * @param delta
         */
public void update(float delta){


}

}
```

Código da clase PantallaXogo
Obxectivo: modificamos a clase para que chame á clase ControladorXogo.

```
public class PantallaXogo implements Screen {

private MeuXogoGame meuXogoGame;
private RendererXogo rendererXogo;
private ControladorXogo controladorXogo;

Mundo meuMundo;

public PantallaXogo(MeuXogoGame meuXogoGame) {

meuMundo = new Mundo();

this.meuXogoGame = meuXogoGame;
rendererXogo = new RendererXogo(meuMundo);
controladorXogo = new ControladorXogo(meuMundo);
}

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

rendererXogo.render(delta);
controladorXogo.update(delta);
}

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

1.2 Movendo os gráficos co MVC

Agora para mover os gráficos só será necesario chamar o método update de cada personaxe e que sexa dito método o que cambie a posición en función da velocidade.

Cando a posición cambie, a clase RendererXogo visualizará a nova posición.

En código:

Código da clase ElementoMobil
Obxectivo: modificamos a clase para que mova o elemento móbil en función da velocidade.

```
public class ElementoMobil extends Personaxe{

public ElementoMobil(Vector2 posicion, Vector2 tamano, float velocidade_max) {
super(posicion, tamano, velocidade_max);
setVelocidade(velocidade_max);
}
@Override
public void update(float delta) {
// TODO Auto-generated method stub

posicion.add((velocidade*delta),0);
}
```

```
}
```

Código da clase Controlador

Obxectivo: chamamos ó método update de cada coche. O nome do método será controladorCoches xa que a parte de mover tamén terá código de control (visto posteriormente).

```
public class ControladorXogo {

Mundo meuMundo;

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

}

private void controlarCoches(float delta){

for(ElementoMovil coche: meuMundo.getCoches()){
coche.update(delta);
}

}
/**
         * Vai chamar a todos os métodos para mover e controlar os personaxes Tamén
         * xestionará os eventos producidos polo usuario e que veñen dende a clase
         * PantallaXogo
         *
         * @param delta
         */
public void update(float delta) {

controlarCoches(delta);
}

}
```

Se probades o código podedes comprobar como os coches se moven á velocidade indicada na clase Mundo cando as instanciamos...

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