SHADOW

Práctica PMDM

Aarón Pérez Villegas 03/03/2015

SHADOW: Juego plataformas en android usando SurfaceView.

Código fuente

El código fuente se encuentra en la dirección github del proyecto:

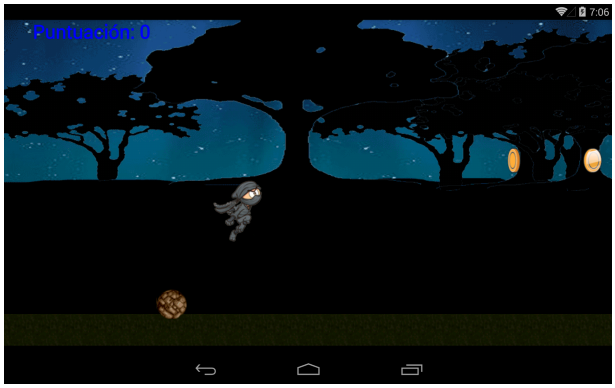
https://github.com/aaronperez/Shadow.git

Funcionamiento de la aplicación

Shadow muestra, mediante SurfaceView, un canvas en el que vamos representando objetos y recoge eventos de tocar la pantalla.

Esta es la imagen inicial al cargar la app.:

Main.xml



Game.xml

Menús

La aplicación no dispone de menús.

Implementación

Manifiesto

En AndroidManifest aplicamos la propiedad landscape en todas las activities para no girar la vista en el dispositivo. Además en la actividad Game le añadimos la propiedad nohistory, para que no vuelva atrás.

Recursos

Drawable

Solo contiene la imagen que hará de icono.

 launcher.png

 back.png

 coin.png

 ground.png

main.jpg


ninja.png


roca.png


temple.png

Layout

El layout de la actividad principal contiene un Button (lanzar el juego) y dos Textviews (muestra las puntuaciones) (*activity_main.xml*).

El layout de la actividad secundaria contiene otro *Listview* (*activity_secundaria.xml*).

Código fuente

Clase Main

El método *onCreate ()* visualiza el Layout principal cargando las puntuaciones, la mejor de preferencias compartidas.

El método *onRestart ()* relanza la música y reescribe las puntuaciones.

El método S*tart ()* lanza la actividad Game.

Clase AdaptadorJ y AdaptadorP

Definen los adaptadores de los *ListViews* que usarán la actividad principal y la secundaria.

Clase AgregarJugador AgregarPartido

Recogen los datos de edición de *Jugadores* y *Partidos* y devuelven un objeto *Jugador* o *Partido* para agregarlo a las tablas.

Clase Gestorjugador y GestorPartido

Define como tratar los objetos *Jugador* y *Partido* en *SQLite*, mediante los métodos *insert*, *delete*, *update*, *List<Objeto>* y *getRow*.

Clase Figura, Coin, RollingStone y Temple

Estas clases construyen desde un bitmap y un contexto una figura animada usando Graphics. Así mediante un sprite generamos una animación al dibujar alternativamente distintas partes del sprite.

Además la clase moneda tiene el método eliminar(), que lo usamos al tocar una moneda, y simplemente la recoloca fuera de la pantalla.

La clase Figura, que usamos con el protagonista, tiene también:

Los métodos tocado() y colisiona(), en cada uno le pasamos un objeto RollingStone y Coin respectivamente y comprueba si coinciden las coordenadas. Devolviendo un booleano.

Clase Game

Crea un objeto Vistajuego (que hereda de SurfaceView) y lo usa de contexto.

Clase HebraJuego

Esta clase en un hilo en el que se apoya la clase VistaJuego, recibe la vista y controla que esté funcionando y recarga el canvas para redibujarlo.

Clase VistaJuego

Esta clase hereda de SurfaceView e implementa SurfaceHolder.Callback.

El constructor asigna los bitmap y los mediaplayer y lanza un HebraJuego pasandole la vista.

El método *onTouchEvent()* recibe el evento de tocar la pantalla y inicia el sonido de salto y pone la variable salto a true.

El método draw() es el encargado de dibujar las distintas figuras cada vez que el canvas se refresca y además comprueba el método acciones() , dónde se generan los eventos del juego y el booleano muerto, para terminar la partida.

```
@Override
   public void draw(Canvas canvas) {
      super.draw(canvas);
      canvas.drawColor(color);
      //background bucle
      z=z-5;
      if(z==-back1.getWidth()) {
         z = 0;
      }
      canvas.drawBitmap(back1, z, 0, null);
      canvas.drawBitmap(back2, z+back1.getWidth(), 0, null);
      //Suelo
      canvas.drawBitmap(ground, 0, alto-125, null);
      //score
      Paint paint = new Paint();
      paint.setColor(Color.BLUE);
```

```
        paint.setAntiAlias(true);
        paint.setFakeBoldText(true);
        paint.setTextSize(60);
        paint.setTextAlign(Paint.Align.LEFT);
        canvas.drawText(getResources().getString(R.string.puntuacion)+": "+score, 90, 60, paint);
        //Monedas
        for(int c=0; c<coins.size();c++) {
            coins.get(c).setPosicion(inicio - 300, coins.get(c).getX() - 10);
            if(coins.get(c).getX()<ancho) {
                coins.get(c).dibujar(canvas);
                if(coins.get(c).getX()<-80){
                    coins.remove(c);
                }
            }
        }
        //Acciones
        acciones();
        //Dibujar prota
        prota.dibujar(canvas);
        //Piedras
        for(int c=0;c < stones.size();c++) {
            stones.get(c).setPosicion(inicio + 140, stones.get(c).getX() - 40);
            if(stones.get(c).getX()<ancho+80) {
                stones.get(c).dibujar(canvas);
                if(stones.get(c).getX()<-80){
                    stones.remove(c);
                }
            }
        }
        if(coins.isEmpty() && stones.isEmpty()){
            templo.setPosicion(0, templo.getX() - 10);
            templo.dibujar(canvas);
        }
        if(muerto){
            hebraJuego.setFuncionando(false);
            String s = getResources().getString(R.string.gameover)+"
"+getResources().getString(R.string.puntuacion)+":";
            canvas.drawText(s+" "+score, 250, inicio-50, gameOver());
            fin();
        }
    }
```

El método acciones() es el controla los eventos de juego comparando la posición sobre el eje de la X y de la Y con respecto a las otras figuras que aparecen.

```
private void acciones(){
        if(salto){
            if(prota.getY()<=inicio && prota.getY()>inicio-320){
                prota.setPosicion(prota.getY()-40, prota.getX());
            }else{
                salto=false;
            }
        }else{
```

```
            if(prota.getY()<inicio){
                prota.setPosicion(prota.getY()+40, prota.getX());
            }
        }
        //Monedas
        if(prota.getY() <= inicio-310){
            for(int y=0;y<coins.size();y++) {
                if (prota.colisiona(coins.get(y))) {
                    coins.get(y).eliminar();
                    score = score + 50;
                    takecoin.start();
                }
            }
        }
        //Piedras
        if(prota.getY() > inicio-50){
            for(int y=0;y<stones.size();y++) {
                if (prota.tocado(stones.get(y))) {
                    ups.start();
                    prota.setMuerto();
                    muerto = true;
                    mp1.stop();
                }
            }
        }
        //Piedras
        if(prota.getX() >= templo.getX()+30){
            level.start();
            prota.setPosicion(inicio,ancho+300);
            muerto = true;
            mp1.stop();

        }
        //Dibujar salto
        if(prota.getY()==inicio){
            prota.setRun();
        }else{
            prota.setJump();
        }
    }
```

El método fin() lo llamamos al finalizar la partida, para la hebra y compara si se ha mejorado la puntuación para guardarla en las preferencias compartidas como la mejor puntuación.

Por último tenemos los métodos Surfaceholder.callback: surfaceCreated(), surfaceChanged() y surfaceDestroyed().