inda13 - Projekt
Javaga

Gustav Dänsel
Lukas Lundmark

15 maj 2014

1 Programbeskrivning

Vi siktar på att skapa en Galaga-klon i Java. Det ska alltså vara en top-down 2D arkadspel. Vi planerar att använda oss av libGDX-biblioteket.

Spelet fungerar så att man kontrollerar ett rymdskepp som ska skjuta ned utomjordingar som kommer från fönstrets övre kant och försöker röra sig nedåt. Målet är att förstöra dem innan de försvinner ut skärmen. Se http://en.wikipedia.org/wiki/Galaga

2 Användarbeskrivning

Vi tänker oss att personer som är sugna på att spela klassiska retrospel utan att köpa en arkadmaskin kan vara en potentiell målgrupp. Annars ingen, tyvärr.

3 Användarscenarie

3.1 Scenarie 1

Sent på kvällen, dagen innan tentamen för SF1626, pluggångesten liger tungt över Kalle. Han känner att han behöver ta en paus från att inte plugga och göra något annat och får ett tips om den supercoola Galagaklonen Javaga. Han laddar ned spelet från thepiratebay och börjar prokrastinera hårt. Han stry sin rymdhjälte med piltangenterna och skriker avfyra! med mellanslag. Vilken upplevelse det är! Han har aldrig varit med om något liknande! Kalle sitter uppe och spelar hela natten och missar tyvärr tentan, mycket sorgligt.

Based on a true story.

3.2 Scenarie 2

Pappa Per är på väg hem från en lång och slitsam dag på jobbet. På vägen hem ser han reklamen för Game On 2.0 på Tekniska Museet. Han kommer och tänka på alla fantastiska kvällar med sina vänner i arkadhallen när han var liten. Han tänker på Galaga och hur kul det var att spela. När han väl kommer hem så bingar han fram en lista på Galagakloner och den som ligger högst upp och har bäst omdömen är ju givetvis Javaga. Per laddar ned spelet och börjar tidsresan till barndomen. Och vilken resa det är! Sicken resa, Mycket fräck! Han känner sina mossiga gamla färdigheter komma tillbaks och upplever sann eufori. Fantastiskt.

4 Testplan

Vi planerar att muta folk till att spela spelet och ge oss bra feedback. Vi planerar att göra detta vid minst två tillfällen, kanske mer om det finns tid. En viss mängd automatiserade tester kan förekomma.

Testanvändaren förväntas spela spelet tills ögonen blöder och därefter berätta för oss hur fantisktiskt det var.

5 Programdesign

Tanken är att dela upp programmet i flera klasser. Primärt så tänker vi oss att vi har dessa klasser:

- Input
- Render
- Game-Logic
- Units
- File I/O

5.1 Input

Här finns metoder för att läsa indata från kontroller såsom tangentbord och mus, samt skicka vidare dessa till relevanta klasser.

5.2 Render

Här görs det tunga jobbet att rita bilden som ska visas på skärmen.

5.3 Game-Logic

I denna klass sköts all logik - såsom hur enheter ska förflytta sig, om skott träffar eller inte och liknande saker.

5.4 Units

Units innehåller beskrivningar och parametrar för alla olika typer av ssskeppsom finns i spelet.

5.5 File I/O

För att läsa och skriva till hårddisk, för t.ex. inställningsfiler.

6 Tekniska frågor

Den största tekniska frågan ligger i hur vi implementerar biblioteket. I det problemer finns det även fler mindre problem, såsom vilken typ av input vi ska använda o.s.v.

Klassiska problem såsom animation sköter det bibliotek (libGDX) vi använder.

7 arbetsplan

7.1 Tidsplan

- Läsa och sätta sig in i biblioteket och dess dokumentiation - 2/5
- Första fungerande prototyp - 9/5
- Testning under helgen 9/5 till 12/5
- Finslipning mer test till 16/5

Planen är att använda GitHub för att samarbeta på koden. Vi tänker oss att vi delar lika på arbetsuppgifterna och arbetar tillsammans. Då projektet är relativt litet så blir det enklare så då vi båda har koll på all kod och vi slipper läsa ikapp.

8 Programkod

src/SpaceObject/Player.java

```
1  package com.me.Javaga.spaceobject;
2
3  import com.badlogic.gdx.Gdx;
4  import com.badlogic.gdx.audio.Sound;
5  import com.badlogic.gdx.graphics.g2d.SpriteBatch;
6  import com.me.Javaga.managers.GameKeys;
7  import com.me.Javaga.managers.GameStateManager;
8
9  import java.util.ArrayList;
10
11 /**
12  * Class for the Players unit. Contains parameters for position
        as well as the sprite used to draw to canvas.
13  * <p/>
14  * Created by Dansel on 2014-04-30.
15  */
16 public class Player extends SpaceObject {
17
18     private static final String FILENAME = "ship.png";
19     private static long time;
20     //private float rotation;
21     //private float scale;
22     private ArrayList<Bullet> bullets;
23     private Sound sound;
24     private int shootLimit;
25
26
27
28     //Call the super-class's constructor
29     public Player(float xPos, float yPos, ArrayList<Bullet>
           bullets) {
30         super(xPos, yPos);
31         HEIGHT = Gdx.graphics.getHeight();
32         WIDTH = Gdx.graphics.getWidth();
33         this.bullets = bullets;
34         init();
35     }
36
37     @Override
38     public void init() {
39         shootLimit = 200;
40         //Set scalefactor
41         setScale(0.4f);
42         spriteSetUp(FILENAME);
43
44         //Create the sprite with some texture
45         sound = Gdx.audio.newSound(Gdx.files.internal("lazer.mp3
               "));
46     }
47
```

```
48      /**
49       * Checks for keypresses and updates the sprite position
50       */
51      @Override
52      public void update() {
53          if (GameKeys.isDown(GameKeys.UP)) {
54              yPos += 5;
55          }
56          if (GameKeys.isDown(GameKeys.DOWN)) {
57              yPos -= 5;
58          }
59          if (GameKeys.isDown(GameKeys.LEFT)) {
60              xPos -= 5;
61          }
62          if (GameKeys.isDown(GameKeys.RIGHT)) {
63              xPos += 5;
64          }
65
66          if (GameKeys.isDown(GameKeys.SPACE)) {
67              if (System.currentTimeMillis() - time > shootLimit)
                    {
68                  time = System.currentTimeMillis();
69                  fire();
70              }
71          }
72
73          xCenter = xPos + sprite.getWidth() / 2;
74          yCenter = yPos + sprite.getHeight() / 2;
75          //Update position
76          wrap();
77          sprite.setX(xPos);
78          sprite.setY(yPos);
79          //Update hitbox
80          hitbox.setCenter(xCenter, yCenter);
81      }
82
83      /**
84       * Draws the players sprite to the canvas at designated xPos
              and yPos.
85       *
86       * @param batch SpriteBatch
87       */
88      @Override
89      public void draw(SpriteBatch batch) {
90          sprite.draw(batch);
91      }
92
93      @Override
94      public boolean checkHealthy() {
95          return isHealthy;
96      }
97
98      @Override
99      public boolean checkForCollision(ArrayList<Bullet>
```

```
        enemyBullets) {
100         for (Bullet bullet : enemyBullets) {
101             if (overlap(bullet)) {
102                 System.out.println("COLLISION!");
103                 return true;
104             }
105         }
106         return false;
107     }
108 
109     /**
110      * Makes sure the players sprite may not move outside the
            window.
111      */
112     public void wrap() {
113         if (xCenter - sWidth / 2 < 0) {
114             xCenter = 0 + sWidth / 2;
115             xPos = xCenter - sprite.getWidth() / 2;
116         }
117 
118         if (xCenter + sWidth / 2 > WIDTH) {
119             xCenter = WIDTH - sWidth / 2;
120             xPos = xCenter - sprite.getWidth() / 2;
121         }
122 
123         if (yCenter - sHeight / 2 < 0) {
124             yCenter = 0 + sHeight / 2;
125             yPos = yCenter - sprite.getHeight() / 2;
126         }
127 
128         if (yCenter + sHeight / 2 > HEIGHT) {
129             yCenter = HEIGHT - sHeight / 2;
130             yPos = yCenter - sprite.getHeight() / 2;
131         }
132         //Okay, so, since the scaling of the sprite doesn't
                change the boundingbox of it we must
133         //manually find the center of the sprite and from that
                number derive the new edges (the visible edges).
134     }
135 
136     private void fire() {
137         sound.play(GameStateManager.getEffectVolume()); // play
                lazer
138         bullets.add(new Bullet(xCenter, yCenter, 90, 20));
139     }
140 
141     public float getX() {
142         return xCenter;
143     }
144 
145     public float getY() {
146         return yCenter;
147     }
148 
```

```
149         @Override
150         public void dispose() {
151             super.dispose();
152             sound.dispose();
153         }
154 }
```