

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 ligger 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 fädigheter 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 fantastiskt 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 skeppsom 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 problemet 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

- Läs och sätta sig in i biblioteket och dess dokumentation - 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
13  * as well as the sprite used to draw to canvas.
14  * <p/>
15  * Created by Dansel on 2014-04-30.
16  */
17 public class Player extends SpaceObject {
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>
30         bullets) {
31         super(xPos, yPos);
32         HEIGHT = Gdx.graphics.getHeight();
33         WIDTH = Gdx.graphics.getWidth();
34         this.bullets = bullets;
35         init();
36     }
37
38     @Override
39     public void init() {
40         shootLimit = 200;
41         //Set scalefactor
42         setScale(0.4f);
43         spriteSetUp(FILENAME);
44
45         //Create the sprite with some texture
46         sound = Gdx.audio.newSound(Gdx.files.internal("lazer.mp3
47         "));
48     }
49 }
```

```

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              {
69                  time = System.currentTimeMillis();
70                  fire();
71              }
72      }
73
74      xCenter = xPos + sprite.getWidth() / 2;
75      yCenter = yPos + sprite.getHeight() / 2;
76      //Update position
77      wrap();
78      sprite.setX(xPos);
79      sprite.setY(yPos);
80      //Update hitbox
81      hitbox.setCenter(xCenter, yCenter);
82  }
83
84  /**
85   * Draws the players sprite to the canvas at designated xPos
86   * and yPos.
87   *
88   * @param batch SpriteBatch
89   */
90  @Override
91  public void draw(SpriteBatch batch) {
92      sprite.draw(batch);
93  }
94
95  @Override
96  public boolean checkHealthy() {
97      return isHealthy;
98  }
99
100  @Override
101  public boolean checkForCollision(ArrayList<Bullet>

```

```

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

```

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