Heart of Klaus

Mini Beispiel mit Sprung und Einlesen eines Levels

Sven Eric Panitz
Hochschule Rhein-Main

Dieser kleine Spielansatz demonstriert, wie man einen Level aus einem String einlesen kann und den Einsatz von fallenden und springenden Spielfiguren,

1 Spielfiguren

1.1 Statische Figuren

Dieses Spiel kennt zwar Arten von Spielfiguren, die sich nie bewegen. das sind zum einen Herzen, die von der Spielfigur Klaus einzusammeln sind.

```
1  package name.panitz.game.klaus;
2
3  import name.panitz.game.framework.ImageObject;
4  import name.panitz.game.framework.Vertex;
5
6  public class Heart<I> extends ImageObject<I> {
7
8    public Heart(Vertex corner ) {
9      super("heart.png", corner, new Vertex(0,0));
10   }
11
12 }
```

Listing 1: src/name/panitz/game/klaus/Heart.java

Dann gibt es noch Wände, durch die kein anderes Objekt hindurcch gehen soll.

```
1  package name.panitz.game.klaus;
2
3  import name.panitz.game.framework.ImageObject;
4  import name.panitz.game.framework.Vertex;
5
6  public class Wall<I> extends ImageObject<I> {
7
8    public Wall(Vertex corner) {
9      super("wall.png", corner, new Vertex(0,0));
10   }
11
12 }
```

Listing 2: src/name/panitz/game/klaus/Wall.java

1.2 Fallende Figuren

In dem Spiel fallen immer wieder von oben Apfelweinfässer durch die Spielebene. Hierzu gibt es eine eigene Klasse.

```
1  package name.panitz.game.klaus;
2
3  import name.panitz.game.framework.FallingImage;
4  import name.panitz.game.framework.Vertex;
5
6  public class Barrel<I> extends FallingImage<I> {
```

```
7     public Barrel(Vertex corner) {
8       super("fass.gif", corner, new Vertex(1, 0));
9     }
10
11    public void fromTop(double wi) {
12      getPos().moveTo(
13          new Vertex(Math.random()*(wi - 2*40) + 40, -40));
14    }
15  }
```

Listing 3: src/name/panitz/game/klaus/Barrel.java

Die eigentliche Spielfigur Klaus kann springen und laufen.

```
1  package name.panitz.game.klaus;
2
3  import name.panitz.game.framework.FallingImage;
4  import name.panitz.game.framework.Vertex;
5
6  public class Klaus<I> extends FallingImage<I> {
7    public Klaus(Vertex corner) {
8      super("player.png", corner, new Vertex(0, 0));
9    }
10 }
```

Listing 4: src/name/panitz/game/klaus/Klaus.java

2 Spiellogik

Das eigentliche Spiel. Interessant ist hier, dass das Layout des Spielfeldes in einem String codiert ist, und die Spielobjekte entsprechend der Codierung erzeugt werden.

```
1  package name.panitz.game.klaus;
2
3  import java.io.BufferedReader;
4  import java.io.IOException;
5  import java.io.StringReader;
6  import java.util.ArrayList;
7  import java.util.List;
8
9  import name.panitz.game.framework.AbstractGame;
10 import name.panitz.game.framework.Button;
11 import name.panitz.game.framework.FallingImage;
12 import name.panitz.game.framework.GameObject;
13 import name.panitz.game.framework.GraphicsTool;
14 import name.panitz.game.framework.KeyCode;
15 import name.panitz.game.framework.SoundObject;
16 import name.panitz.game.framework.Vertex;
17
18 public class HeartsOfKlaus<I, S> extends AbstractGame<I, S> {
```

```
19    List<Wall<I>> walls = new ArrayList<>();
20    List<Heart<I>> hearts = new ArrayList<>();
21    List<Barrel<I>> barrels = new ArrayList<>();
22    int energy = 5;
23    int collectedHearts = 0;
24    Klaus<I> klaus;
25
26    static final int GRID_WIDTH=34;
27
28    String level1 = "w                  f  w\n"
29        + "whf          h    www w\n"
30        + "wwwww     wwww       w\n"
31        + "w         h          w\n"
32        + "w       wwww       h w\n"
33        + "w                  w\n"
34        + "w       hf          fhw\n"
35        + "w    wwww       wwwwww\n"
36        + "w                  w\n"
37        + "wh              h  w\n"
38        + "wwwww       wwww   w\n"
39        + "w                  w\n"
40        + "w p     h          hw\n"
41        + "w   wwww       wwwwww\n"
42        + "w                  w\n"
43        + "w                 gw\n"
44        + "wwwwwwww  wwwwwwwwwwww";
45
46    public HeartsOfKlaus() {
47      super(new Klaus<>(new Vertex(0,0)), 22*GRID_WIDTH, 17*GRID_WIDTH);
48
49      klaus = (Klaus<I>) getPlayer();
50
51      BufferedReader r = new BufferedReader(new StringReader(level1));
52      int l = 0;
53      try {
54        for (String line=r.readLine(); line != null; line=r.readLine()){
55          int col = 0;
56          for (char c : line.toCharArray()) {
57            switch (c) {
58            case 'w':
59              walls.add(new Wall<>
60               (new Vertex(col * GRID_WIDTH, l * GRID_WIDTH)));
61              break;
62            case 'h':
63              hearts.add(new Heart<>
64                  (new Vertex(col * GRID_WIDTH, l * GRID_WIDTH)));
65              break;
66            case 'f':
67              barrels.add(new Barrel<>(
68                  new Vertex(col*GRID_WIDTH, l*GRID_WIDTH)));
69              break;
70            case 'p':
```

```
71              getPlayer().getPos().moveTo(new Vertex
72                  (col * GRID_WIDTH, l * GRID_WIDTH-2));
73              break;
74            }
75            col++;
76          }
77          l++;
78        }
79      } catch (IOException e) {
80        e.printStackTrace();
81      }
82      getGOss().add(barrels);
83      getGOss().add(walls);
84      getGOss().add(hearts);
85      getButtons().add(new Button("pause",()->pause()));
86    }
87
88    @Override
89    public void paintTo(GraphicsTool<I> g) {
90      super.paintTo(g);
91      g.drawString(50, 10, "Energy: " + energy);
92      g.drawString(50, 30, "Hearts: " + hearts.size());
93    }
94
95    SoundObject<S> crash = new SoundObject<S>("crash.wav");
96
97    private void playerBarrelCollision() {
98      for (Barrel<I> p : barrels) {
99        if (p.touches(player)) {
100         energy--;
101         playSound(crash);
102         p.fromTop(getWidth());
103       }
104       if (p.getPos().y > getHeight()) {
105         p.fromTop(getWidth());
106       }
107     }
108   }
109
110   private void fallingBarrel() {
111     for (FallingImage<I> p : barrels) {
112       boolean isStandingOnTop = false;
113       for (GameObject<I> wall : walls) {
114         if (p.touches(wall)) {
115           p.restart();
116         }
117         if (p.isStandingOnTopOf(wall)) {
118           isStandingOnTop = true;
119         }
120       }
121       if (!isStandingOnTop && !p.isJumping) {
122         p.startJump(0.1);
```

```
123       }
124     }
125
126   }
127
128   private void checkPlayerWallCollsions() {
129     boolean isStandingOnTop = false;
130     for (GameObject<I> wall : walls) {
131       if (player.touches(wall)) {
132         klaus.stop();
133         return;
134       }
135       if (player.isStandingOnTopOf(wall)) {
136         isStandingOnTop = true;
137       }
138     }
139
140     if (!isStandingOnTop && !klaus.isJumping)
141       klaus.startJump(0.1);
142   }
143
144   private void collectHearts() {
145     Heart<I> removeMe = null;
146     for (Heart<I> heart : hearts) {
147       if (getPlayer().touches(heart)) {
148         removeMe = heart;
149         collectedHearts++;
150         break;
151       }
152     }
153     if (removeMe != null)
154       hearts.remove(removeMe);
155   }
156
157   @Override
158   public void doChecks() {
159     collectHearts();
160     checkPlayerWallCollsions();
161     fallingBarrel();
162     playerBarrelCollision();
163     if (getPlayer().getPos().y > getHeight()) {
164       getPlayer().getPos()
165         .moveTo(new Vertex(GRID_WIDTH, getHeight() - 80));
166     }
167   }
168
169   @Override
170   public void keyPressedReaction(KeyCode keycode) {
171     if (keycode != null)
172       switch (keycode) {
173       case LEFT_ARROW:
174         klaus.left();
```

```
175           break;
176         case RIGHT_ARROW:
177           klaus.right();
178           break;
179         case UP_ARROW:
180           klaus.jump();
181           break;
182         case DOWN_ARROW:
183           klaus.stop();
184           break;
185           default:;
186         }
187     }
188
189     public boolean lost() {
190       return energy <= 0;
191     }
192
193     public boolean won() {
194       return hearts.isEmpty();
195     }
196
197 }
```

Listing 5: src/name/panitz/game/klaus/HeartsOfKlaus.java

Eine kleine Startklasse, um das Spiel mit der javafx-Umsetzung zu starten.

```
1  package name.panitz.game.klaus;
2
3
4  import name.panitz.game.framework.fx.GameApplication;
5
6  public class PlayFX extends GameApplication {
7    public PlayFX(){
8      super(new HeartsOfKlaus<>());
9    }
10   public static void main(String[] args) {
11     PlayFX.launch();
12   }
13 }
```

Listing 6: src/name/panitz/game/klaus/PlayFX.java