Swing-Umsetzung der Spielebibliothek

Sven Eric Panitz
Hochschule Rhein-Main

In diesem Papier wird die Umsetzung der Spielerahmenbibliothek in Javas Swing-Bibliothek beschrieben

1 Einbinden der Bibliothek

Es gibt bisher keine kluge und automatische Einbindung der Bibliothek in die verschiedenen Realisierungen. Die Quellen müssen manuell zusammen in ein Projekt kopert werden.

Wir gehen davon aus, dass ein Spiel in einer Entwicklungsumgebung (eclipse, intelliJ) entwickelt werden soll. So wird der einfachste Weg sein, in der Entwicklungsumgebung Projekt anzulegen. Dann die Quelltexte der Rahmenbibliothek in das Projekt zu kopieren. Dann die Quelltexte der Swing-Realisierung in das Projekt zu kopieren. Bild- und Klang-Dateien können in eclipse in den Quelltextordner `src` kopiert werden. Zum Testen, ob das Projekt korrekt aufgesetzt wurde, empfielt es sich, das in der Bibliothek enthaltene Spiel `SimpleGame` zu kompilieren und zu starten.

Es spricht nichts dagegen, in dem Projekt nicht nur die Bibliothek der Swing-Realisierung sondern auch die Klassen der javafx-Realisierung zu kopieren und in dem Projekt sowohl das Spiel für swing als auch für javafx zu kompilieren und zu starten.

In der Klasse `SwingGame` gibt es folgende statische Methode, mit der ein Spiel in einem Fenster angezeigt und gestartet wird:

```
public static void startGame(GameLogic<Image> game)
```

2 Die Swing-Umsetzung

Jede Umsetzung der Spielebibliothek muss Implementierungen der Schnittstellen auf die die Bibliothek verweist, beinhalten. Dieses sind auf jedem Fall das `GrphicsTool` und das `SoundTool`. Dann muss es eine Klasse geben, in dem das Spiel in einem Fensterinhalt angezeigt wird und eine zeitgesteuerte SChleife beinhaltet, die die Animation realsiert.

2.1 Zeichnen der Grafiken

Swing verwendet zum Zeichnen auf dem Bildschirm die Klasse `java.awt.Grphics`. Die von dem `GraphicsTool` erwartete Funktionalität wird in dieser Umsetzung entsprechend für diese awt-Klasse umgesetzt.

```
1 package name.panitz.game.framework.swing;
2
3 import java.awt.Graphics;
4 import java.awt.Image;
5 import java.awt.Color;
```

```
6  import java.awt.Font;
7  import javax.swing.ImageIcon;
8
9  import name.panitz.game.framework.GraphicsTool;
10 import name.panitz.game.framework.GameObject;
11
12 public class SwingGraphicsTool implements GraphicsTool<Image>{
13   Graphics g;
14
15   public SwingGraphicsTool(Graphics g) {
16     this.g = g;
17   }
18
19   @Override
20   public void drawImage(Image img, double x, double y) {
21     g.drawImage(img, (int)x, (int)y, null);
22   }
23
24   @Override
25   public void drawRect(double x, double y, double w, double h) {
26     g.drawRect((int)x, (int)y, (int)w, (int)h);
27   }
28
29   @Override
30   public void fillRect(double x, double y, double w, double h) {
31     g.fillRect((int)x, (int)y, (int)w, (int)h);
32   }
33
34   @Override
35   public void drawOval(double x, double y, double w, double h) {
36     g.drawOval((int)x, (int)y, (int)w, (int)h);
37   }
38
39   @Override
40   public void fillOval(double x, double y, double w, double h) {
41     g.fillOval((int)x, (int)y, (int)w, (int)h);
42   }
43
44   @Override
45   public void drawLine(double x1, double y1, double x2, double y2) {
46     g.drawLine((int)x1,(int)y1, (int)x2, (int)y2);
47   }
48
49   @Override
50   public void drawString(double x,double y,int fontSize, String
      fontName, String text){
51     g.setFont(new Font(fontName, Font.PLAIN, fontSize));
52     g.drawString(text, (int)x, (int)y);
53   }
54
55   @Override
56   public  Image generateImage(String name, GameObject<Image> go){
```

```
57      ImageIcon image = new ImageIcon(getClass().getClassLoader().
        getResource(name));
58      go.setWidth(image.getIconWidth());
59      go.setHeight(image.getIconHeight());
60      return image.getImage();
61    }
62
63    @Override
64    public void setColor(double r, double gr, double b) {
65      g.setColor(new Color((float)r, (float)gr, (float)b));
66    }
67  }
```

Listing 1: src/name/panitz/game/framework/swing/SwingGraphicsTool.java

2.2 Klänge

Es gibt verschiedene Wege, um in Java Klänge abzuspielen. Wir verwenden die Klassen aus dem Paket `javax.sound.sampled` um Klänge abzuspielen.

```
1  package name.panitz.game.framework.swing;
2
3  import java.io.BufferedInputStream;
4  import java.io.IOException;
5  import java.io.InputStream;
6
7  import javax.sound.sampled.AudioInputStream;
8  import javax.sound.sampled.AudioSystem;
9  import javax.sound.sampled.Clip;
10 import javax.sound.sampled.LineUnavailableException;
11 import javax.sound.sampled.UnsupportedAudioFileException;
12
13 import name.panitz.game.framework.SoundTool;
14
15 public class JavaSoundTool implements SoundTool<AudioInputStream> {
16
17   @Override
18   public AudioInputStream loadSound(String fileName) {
19     try {
20       InputStream src = getClass().getClassLoader().
       getResourceAsStream(fileName);
21       InputStream bufferedIn = new BufferedInputStream(src);
22       // File laden
23       AudioInputStream audioStream = AudioSystem.getAudioInputStream(
       bufferedIn);
24
25       return audioStream;
26     } catch (UnsupportedAudioFileException |IOException e) {
27       e.printStackTrace();
28     }
29     System.out.println("ups?");
```

```
30      return null;
31    }
32
33    @Override
34    public void playSound(AudioInputStream sound) {
35      Clip clip;
36      try {
37        clip = AudioSystem.getClip();
38        clip.open(sound);
39        clip.start();
40      } catch (LineUnavailableException | IOException e) {
41        // TODO Auto-generated catch block
42        e.printStackTrace();
43      }
44    }
45
46  }
```

Listing 2: src/name/panitz/game/framework/swing/JavaSoundTool.java

2.3 Anzeige des Spiels und Zeitschleife

Um das Spiel anzuzeigen und zu animieren wird die swing-Standardklasse `JPanel` verwendet. Es wird eine Unterklasse definiert. In dieser wird die Methode `paintComponent` so überschrieben, dass die Spielobjekte auf dem Bildschirm gezeichnet werden.

Sie Klasse enthält ein swing-Timer Objekt. Dieses steuert die Animation und ruft in wiederkehrenden Abständen die Methoden zum Bewegen der Spielobjekte auf.

```
1  package name.panitz.game.framework.swing;
2
3  import java.awt.Dimension;
4  import java.awt.Graphics;
5  import java.awt.Image;
6  import java.awt.event.KeyAdapter;
7  import java.awt.event.KeyEvent;
8
9  import javax.sound.sampled.AudioInputStream;
10 import javax.swing.JPanel;
11 import javax.swing.Timer;
12
13 import name.panitz.game.framework.GameLogic;
14 import name.panitz.game.framework.KeyCode;
15 import name.panitz.game.framework.SoundObject;
16
17
18 @SuppressWarnings("serial")
19 public class SwingScreen extends JPanel{
20   GameLogic<Image,AudioInputStream> logic;
```

```
21    Timer t;
22
23    private JavaSoundTool soundTool = new JavaSoundTool();
24
25    public SwingScreen(GameLogic<Image,AudioInputStream> gl) {
26      this.logic = gl;
27
28      t = new Timer(13, (ev)->{
29        logic.move();
30        logic.doChecks();
31        repaint();
32        getToolkit().sync();
33
34        logic.playSounds(soundTool);
35        requestFocus();
36      });
37      t.start();
38
39      addKeyListener(new KeyAdapter() {
40        @Override
41        public void keyPressed(KeyEvent e) {
42          logic.keyPressedReaction(KeyCode.fromCode(e.getKeyCode()));
43        }
44        @Override
45        public void keyReleased(KeyEvent e) {
46          logic.keyReleasedReaction(KeyCode.fromCode(e.getKeyCode()));
47        }
48
49      });
50      setFocusable(true);
51      requestFocus();
52    }
53
54    @Override
55    public Dimension getPreferredSize() {
56      return new Dimension((int)logic.getWidth(),(int)logic.getHeight())
        ;
57    }
58
59    @Override
60    protected void paintComponent(Graphics g) {
61      super.paintComponent(g);
62      logic.paintTo(new SwingGraphicsTool(g));
63    }
64
65  }
```

Listing 3: src/name/panitz/game/framework/swing/SwingScreen.java

2.4 Zusätzlich Knöpfe für das Spiel

Zusätzlich gibt es die Möglichkeit für ein Spiel noch Knöpfe zu definieren. Hierfür wird eine weitere GUI-Komponente definiert. Auch diese ist Unterklasse der Klasse `JPanel`. Sie bündelt die eigentliche Spielleinwand und die Knöpfe zu einer integrierten GUI-Komponente.

```
1  package name.panitz.game.framework.swing;
2
3  import java.awt.BorderLayout;
4  import java.awt.Image;
5
6  import javax.sound.sampled.AudioInputStream;
7  import javax.swing.JButton;
8  import javax.swing.JPanel;
9  import javax.swing.JFrame;
10
11 import name.panitz.game.framework.Button;
12 import name.panitz.game.framework.GameLogic;
13
14 @SuppressWarnings("serial")
15 public class SwingGame extends JPanel {
16   GameLogic<Image, AudioInputStream> logic;
17
18   public SwingGame(GameLogic<Image, AudioInputStream> logic) {
19     super();
20     this.logic = logic;
21     SwingScreen swsc = new SwingScreen(logic);
22     this.setLayout(new BorderLayout());
23     add(swsc, BorderLayout.CENTER);
24     JPanel buttonsPanel = new JPanel();
25     for (Button b :logic.getButtons()){
26       JButton jb = new JButton(b.name);
27       jb.addActionListener(ev->b.action.run());
28       buttonsPanel.add(jb);
29     }
30     add(buttonsPanel, BorderLayout.SOUTH);
31   }
32
33
34   public static void startGame(GameLogic<Image, AudioInputStream> game
     ) {
35     JFrame f = new JFrame();
36     f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
37     f.add(new SwingGame(game));
38     f.pack();
39     f.setVisible(true);
40   }
41
42 }
```

Listing 4: src/name/panitz/game/framework/swing/SwingGame.java

Diese Klasse enthält zusätzlich eine statische Methode, um ein konkretes Spiel direkt in swing zu starten.