WiSe 21/22 GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

Prof. Dr. Winfried Kurth
Alex Tavkhelidze

Praktikum Computergrafik

Folien zu #3

Eingabesteuerung, Wiedergabe von Multimediaformaten, Einführung: komplexere Szenen

Tastatur

Maus

Audio (WAV)

Hierarchische Modelle

2D TextRenderer

Steuerung über Tastatur

- Die Liste aller Funktionen und Tastencodes der Klasse **KeyEvent** entnehmt ihr den **Oracle API-Spezifikationen** hier
- Die Methode keyPressed steuert die **Betätigung** (aka das **Drücken**) jeder Taste
- Die Methode keyReleased steuert das **Lösen** (aka das **Loslassen**) jeder Taste
- Die Methode key**Typed** steuert die **Betätigung** jeder **Zeichen**taste (d.h. Buchstaben, Symbole, Ziffern) – also sobald was gedruckt wird

```
package cg;
//-------------------------------------------------------
...
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
...
//-------------------------------------------------------
public class Main implements KeyListener {
        ...
        public static void main(String[] args) {
            ...
            canvas.addKeyListener(new Main());

            canvas.setFocusable(true);
            ...
        }
//-------------------------------------------------------
                public void keyTyped(KeyEvent e) {
                    int keyCo = e.getKeyCode();
                    if (keyCo == KeyEvent.VK_A) {...}
                    ...
                    }
//-------------------------------------------------------
                public void keyPressed(KeyEvent e) {...}
//-------------------------------------------------------
                public void keyReleased(KeyEvent e) {...}
//-------------------------------------------------------

}
```

Steuerung über Maus (1)

- Die Liste aller Funktionen und Tastencodes der Klasse **MouseEvent** entnehmt ihr den **Oracle API-Spezifikationen** hier
- Die Methode mouseClicked vereint in sich die Ereignisse „**Drücken**“ und „**Lösen**“ einer Maustaste (oder des Mausrades)
- Die Methoden mouseEntered bzw. mouseExited werden aufgerufen sobald der Mauszeiger die Fläche des „abgehörten“ **GUI-Elementes** betritt bzw. verlässt
- Fürs Drehen des Mausrads sind die extra Klassen MouseWheelListener und MouseWheelEvent zuständig

```
package cg;
//-------------------------------------------------------------------------
import javax.swing.JFrame;
//...
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
//...
//-------------------------------------------------------------------------
public class Main implements MouseListener {
	//...
	public static void main(String[] args) {

		JFrame frame = new JFrame("Fenster");
		final int width = 800;
		final int height = 600;
		frame.setSize(width, height);
		//...
		frame.addMouseListener(new Main());
		//...
		frame.setFocusable(true);
		//...
		frame.setVisible(true);
	}
//-------------------------------------------------------------------------
	public void mouseClicked(MouseEvent e) {/*...*/}
//-------------------------------------------------------------------------
	public void mousePressed(MouseEvent e) {System.out.println(e.getClickCount());}
//-------------------------------------------------------------------------
	public void mouseReleased(MouseEvent e) {/*...*/}
//-------------------------------------------------------------------------
	public void mouseEntered(MouseEvent e) {System.out.println(e.getXOnScreen()+ "/ " + e.getX());}
//-------------------------------------------------------------------------
	public void mouseExited(MouseEvent e) {System.out.println(e.getPoint());}
}
```

Steuerung über Maus (2)

- Die Methoden mouseDragged bzw. mouseMoved werden aufgerufen sobald der Mauszeiger über die Fläche des „abgehörten“ **GUI-Elementes** bei (einer bzw. keiner) gedrückten Maustaste bewegt wird

- MouseEvent.BUTTON**3**_MASK fängt idR jedes Drücken der **rechten** Maustaste (BUTTON**1**_MASK wäre für die **linke** Maustaste, BUTTON**2**_MASK – für das Maus**rad**)

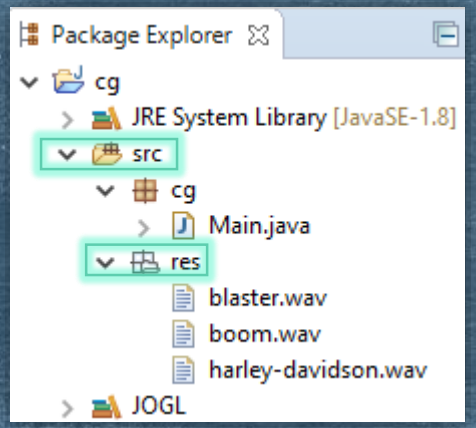
```
package cg;
//-------------------------------------------------------------------------
import javax.swing.JFrame;
import com.jogamp.opengl.awt.GLCanvas;
//...
import java.awt.event.MouseEvent;
import java.awt.event.MouseMotionListener;
//...
//-------------------------------------------------------------------------
public class Main implements MouseMotionListener {
	//...
	public static void main(String[] args) {

		JFrame frame = new JFrame("OpenGL Fenster");
		final int width = 800;
		final int height = 600;
		frame.setSize(width, height);
		//...
		GLCanvas canvas = new GLCanvas();
		frame.add(canvas);//alternativ: frame.getContentPane().add(canvas);
		canvas.addMouseMotionListener(new Main());
		//...
		//setzt den Tastaturfokus auf eine GUI-Komponente (hier: auf die GL-Leinwand)
		//canvas.setFocusable(true);//(bedingt) alternativ: frame.setFocusable(true);
		//...
		frame.setVisible(true);
	}
//-------------------------------------------------------------------------
	public void mouseDragged(MouseEvent e) {
		if (e.getModifiers() == MouseEvent.BUTTON3_MASK) {System.out.println(e.getWhen());}
		}
//-------------------------------------------------------------------------
	public void mouseMoved(MouseEvent e) {System.out.println(e.isShiftDown());}
}
```

alternative Eingabesteuerung (NEWT)

- NEWT weist idR eine höhere Leistung (im Vergleich zu AWT/Swing) auf
- NEWT kann mit AWT/Swing gekoppelt werden
- Oberster GUI-Container ist bereits ein „OpenGL-fähiges“ GLWindow-Fenster

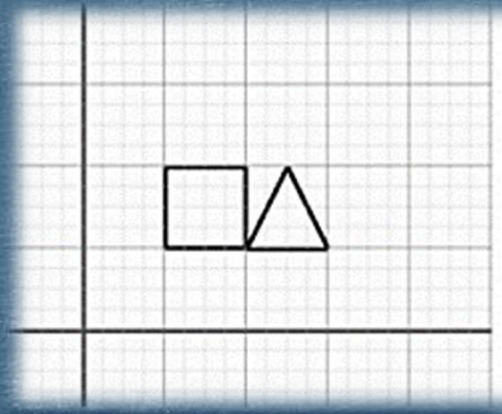
```
package cg;
//-----------------------------------------------------------------------------
import com.jogamp.opengl.GLProfile; import com.jogamp.opengl.GLCapabilities;
import com.jogamp.newt.opengl.GLWindow;
import com.jogamp.opengl.util.FPSAnimator;
import com.jogamp.newt.event.KeyEvent; import com.jogamp.newt.event.KeyListener;
import com.jogamp.newt.event.MouseEvent; import com.jogamp.newt.event.MouseListener;
//-----------------------------------------------------------------------------
public class Main implements KeyListener, MouseListener {
    //...
//-----------------------------------------------------------------------------
public static void main(String[] args) {
        //...
        GLProfile glp = GLProfile.getDefault();
        GLCapabilities caps = new GLCapabilities(glp);

        GLWindow window = GLWindow.create(caps);
        window.setSize(640, 480);
        window.setTitle("NEWT Fenster");
        window.setVisible(true);

        Main mn = new Main(); window.addKeyListener(mn); window.addMouseListener(mn);
        //...
        final FPSAnimator animator = new FPSAnimator(window, 60, true); animator.start();
    }
//-----------------------------------------------------------------------------
    public void keyPressed(KeyEvent e) {System.out.println(e.getKeyChar());}
    public void keyReleased(KeyEvent e) {/*...*/}
//-----------------------------------------------------------------------------
    public void mouseClicked(MouseEvent e) {System.out.println(e.getClickCount());}
    public void mouseEntered(MouseEvent e) {System.out.println(e.getWhen());}
    public void mouseExited(MouseEvent e) {System.out.println(e.getWhen());}
    public void mousePressed(MouseEvent e) {/*...*/}
    public void mouseReleased(MouseEvent e) {/*...*/}
    public void mouseMoved(MouseEvent e) {/*...*/}
    public void mouseDragged(MouseEvent e) {System.out.println(e.getEventType());}
    public void mouseWheelMoved(MouseEvent e) {System.out.println(e.getRotation()[1]);}
}
```

Sound abspielen & stoppen



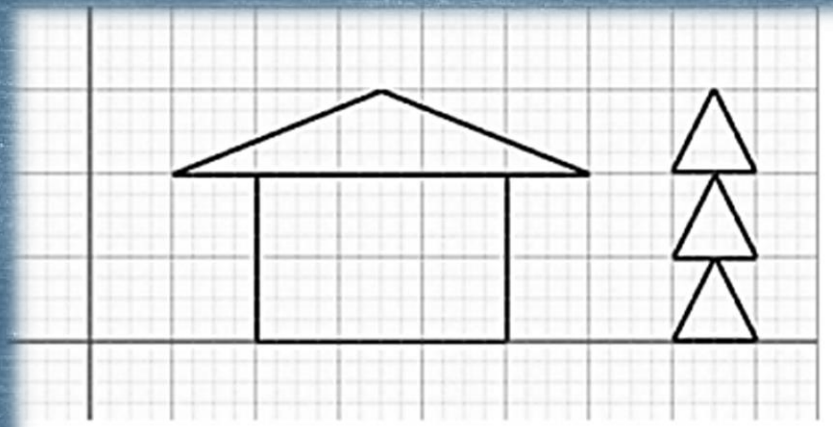
- Java Sound unterstützt die Tondateien mit folgenden Eigenschaften:
  - Dateiformat: AU, AIFF, WAV
    - unkomprimiertes Audio
  - Samplingtiefe: 8-16 Bits
  - Abtastrate: 8-48 kHz
- die Unterstützung von **3D-Sounds** (das heißt, mit Tonabschwächung, Doppler-Effekt und gerichteten Sounds) wird erst mit extra **OpenAL** Java-Paketen mitgeliefert

```
package cg;
//-------------------------------------------------------------------------
...
import java.io.File;
import javax.sound.sampled.AudioInputStream;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.Clip;
...
//-------------------------------------------------------------------------
public class Main implements KeyListener {
      static Clip clip;
      static String[] datei = {"blaster.wav","boom.wav","harley-davidson.wav"};
//-------------------------------------------------------------------------
   public static void main(String[] args) {...}
//-------------------------------------------------------------------------
   public static void soundStart(String datei) {
         try {
               File file = new File("src/res/" + datei);
               AudioInputStream audioIn = AudioSystem.getAudioInputStream(file);
               clip = AudioSystem.getClip();
               clip.open(audioIn);
               clip.start();
            } catch (Exception e) {e.printStackTrace();}
      }
   public static void soundStop() {clip.stop();}
//-------------------------------------------------------------------------
      public void keyPressed(KeyEvent e) {
            int keyCode = e.getKeyCode();
            if(keyCode == KeyEvent.VK_A) {soundStart(datei[0]);}
            if(keyCode == KeyEvent.VK_S) {soundStart(datei[1]);}
            if(keyCode == KeyEvent.VK_D) {soundStart(datei[2]);}
            if(keyCode == KeyEvent.VK_SPACE) {soundStop();}
            }
      ...
}
```



Sample:
translate(1,1)
draw(square)
translate(1,0)
draw(triangle)

Bausteine der Szene



1. translate(2,0)
2. scale(3,2)
3. draw(square)
4. scale(1/3,1/2)
5. translate(5,0)
6. draw(triangle)
7. translate(0,1)
8. draw(triangle)
9. translate(0,1)
10. draw(triangle)
11. translate(-6,0)
12. scale(5,1)
13. draw(triangle)

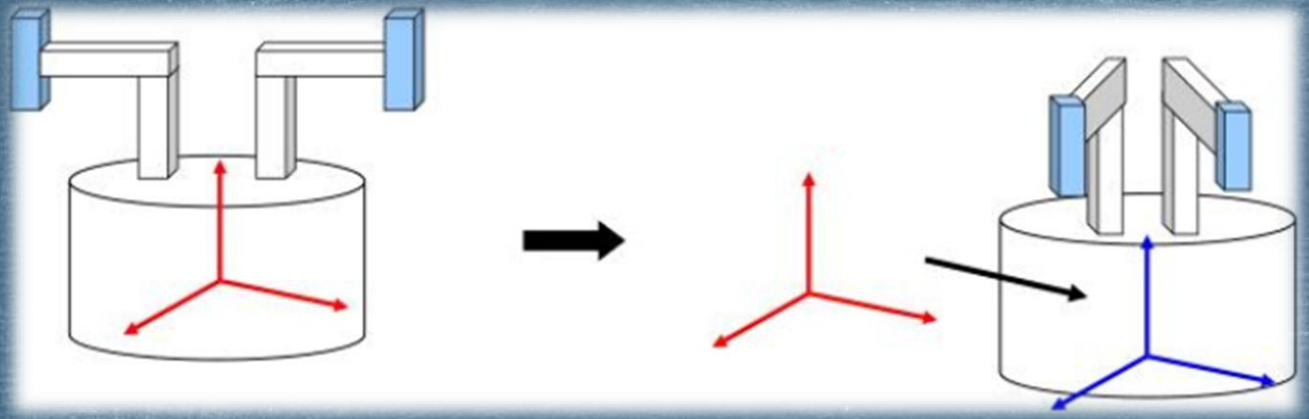
die Szene und deren Aufbau

komplexere Szenen

hierarchische Modelle

Bildquelle: https://media.cheggcdn.com/media/e5c/e5cb7639-d176-4ce5-a0e5-d8617d81cb2b/phpVWgnJm.png

hierarchische Modelle



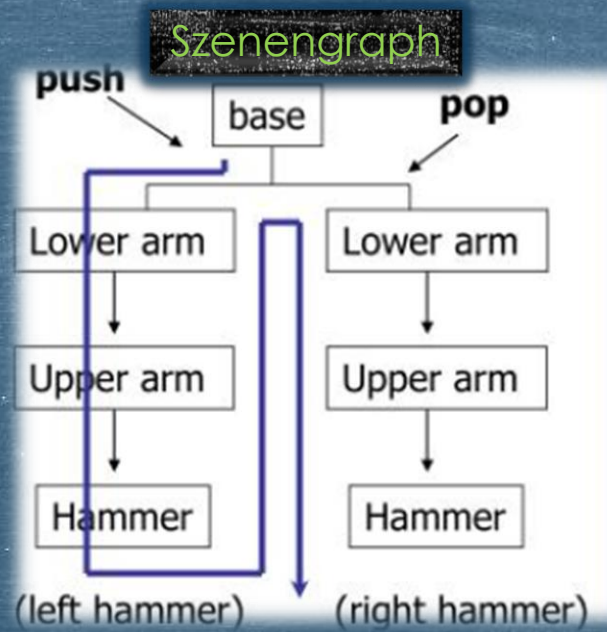
- Das ganze **Konstrukt** besteht aus:
  - einem zylinderförmigen Sockel
    - einem linken Arm – dieser besteht aus:
      - einem quaderförmigen Oberarm
      - einem quaderförmigen Unterarm
      - einem quaderförmigen Hammerkopf
    - einem rechten Arm – dieser besteht aus:
      - einem quaderförmigen Oberarm
      - einem quaderförmigen Unterarm
      - einem quaderförmigen Hammerkopf

Ziel #1:

- Das Konstrukt
  - zu **verschieben**
    - nach rechts
      - um 5 Einheiten

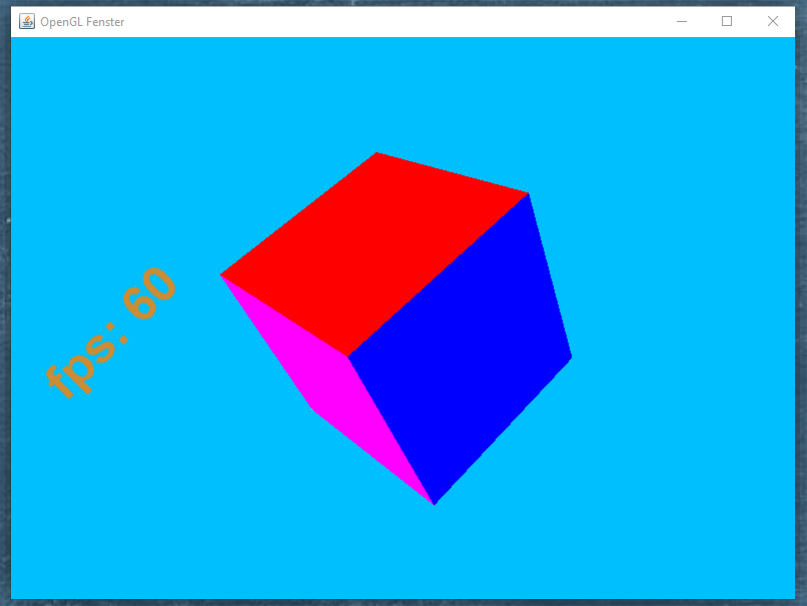
Ziel #2:

- Die Arme
  - zu **drehen**
    - um die Y-Achse
      - um 75 Grad
        - in unterschiedliche Richtungen




```
glTranslate(5,0,0)
Draw_base();
glPushMatrix();

glRotate(75, 0,1,0);
Draw_left_hammer();

glPopMatrix();
glRotate(-75, 0,1,0);
Draw_right_hammer();
```

TextRenderer (2D): Schreiben auf Zeichenfenstern




```
package cg;
//-------------------------------------------------------------------------
...
import java.awt.Font;
import com.jogamp.opengl.util.awt.TextRenderer;
//-------------------------------------------------------------------------
public class Main implements GLEventListener {
      //-------------------------------------------------------------------------
      ...
      TextRenderer renderer;
      //-------------------------------------------------------------------------
      public static void main(String[] args) {...}
      //-------------------------------------------------------------------------
      @Override
      public void init(GLAutoDrawable drawable) {
          ...
          renderer = new TextRenderer(new Font("SansSerif", Font.BOLD, 50));
      }
      //-------------------------------------------------------------------------
      @Override
      public void display(GLAutoDrawable drawable) {
            ...
            renderer.beginRendering(drawable.getSurfaceWidth(), drawable.getSurfaceHeight());
                gl.glMatrixMode(GL2.GL_MODELVIEW);
                gl.glRotatef(45.0f, 0.0f, 0.0f, 1.0f);

                renderer.setColor(1.0f, 0.5f, 0.0f, 0.8f);
                renderer.draw("  fps: " + fps, 150, 100);
            renderer.endRendering();
            ...
      }
      //-------------------------------------------------------------------------
      ...
}
```