Medientechnik

Übung

Heute

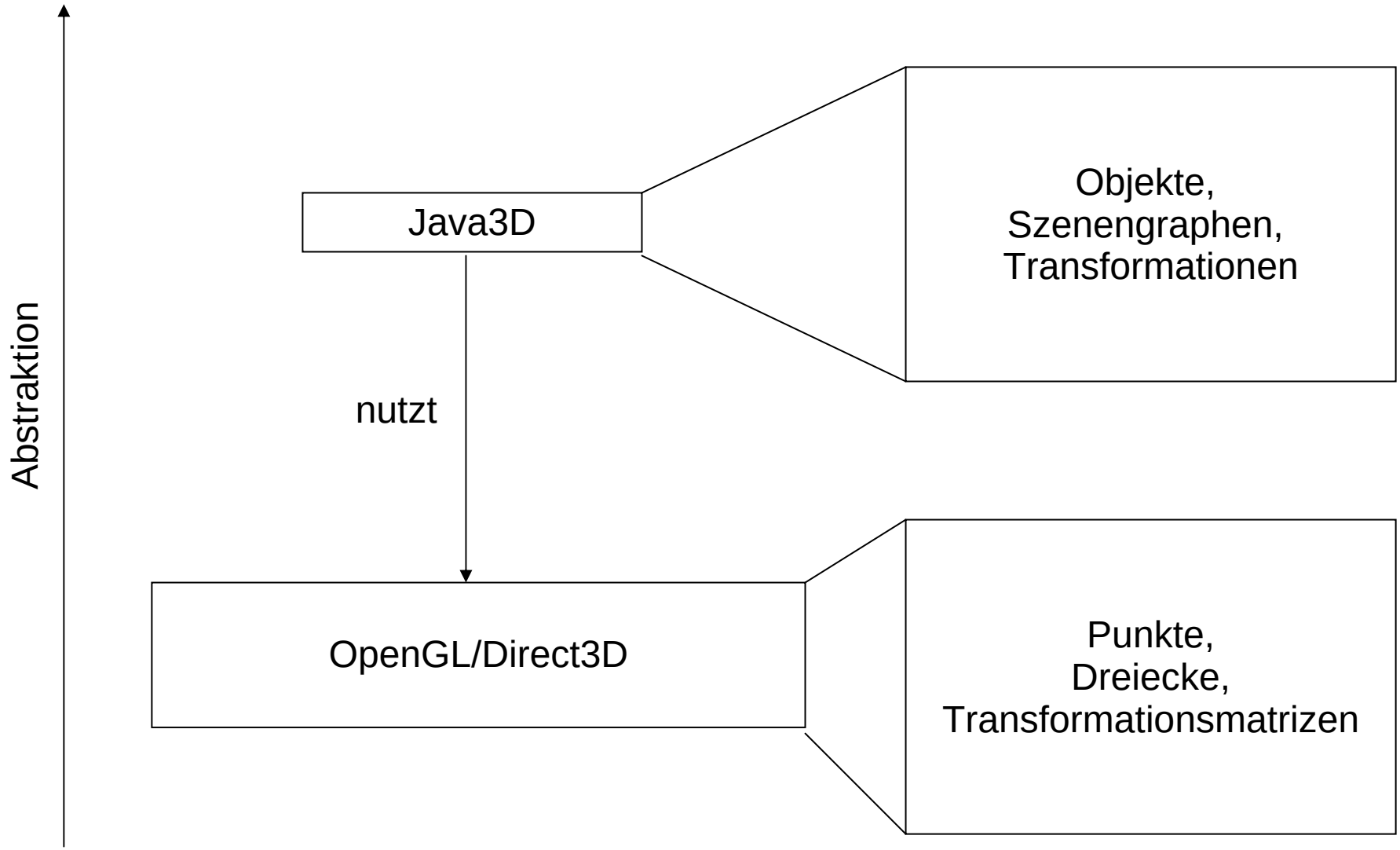
- Java3D:
  - Grundlagen
  - Lichter und Materialien
  - Animationen
- Tastaturinteraktion

Java3D

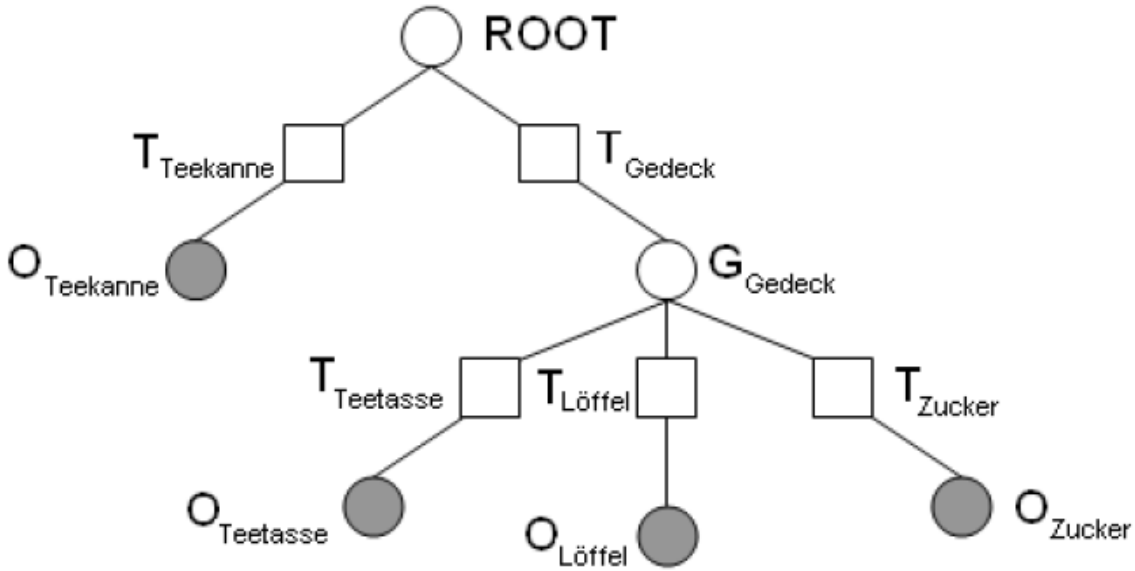
- Bibliothek für dreidimensionale Grafik
- basierend auf low-level Grafikbibliotheken wie OpenGL
- inzwischen Open Source

Sun Java3D-Tutorial: *http://java.sun.com/developer/onlineTraining/java3d/index.html*

Vergleich J3D <> OpenGL/DirectX



Szenengraph



- Gerichteter, azyklischer Graph, der Objekte, Gruppen und Transformationen enthält
- Wird beim Zeichnen von ROOT aus durchwandert
- Änderungen eines Transformationsknotens wirken sich damit auf den gesamten Teilbaum aus
- Java3D nutzt Szenengraphensystem

Grundlage

```
import java.awt.*;
import javax.swing.*;

import javax.media.j3d.*;
import javax.vecmath.*;

import com.sun.j3d.utils.geometry.*;
import com.sun.j3d.utils.universe.*;

public class HelloUniverse extends JFrame {

      public HelloUniverse() {
            super();
            this.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
            GraphicsConfiguration config = SimpleUniverse
                              .getPreferredConfiguration();

            Canvas3D c = new Canvas3D(config);
            this.getContentPane().add(c);
```

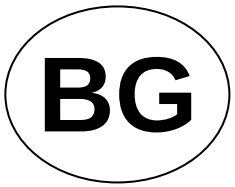
HelloUniverse.java

Grundlage

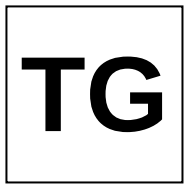
```
        SimpleUniverse u = new SimpleUniverse(c);

        u.getViewingPlatform().setNominalViewingTransform();
    }

    public static void main(String[] args) {
        HelloUniverse hello = new HelloUniverse();
        hello.setSize(800, 600);
        hello.setVisible(true);
    }
}
```

HelloUniverse.java

Szenengraph in J3D

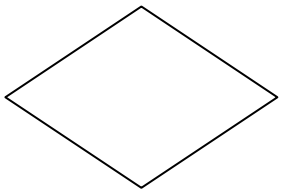


BranchGroup
(Gruppe)

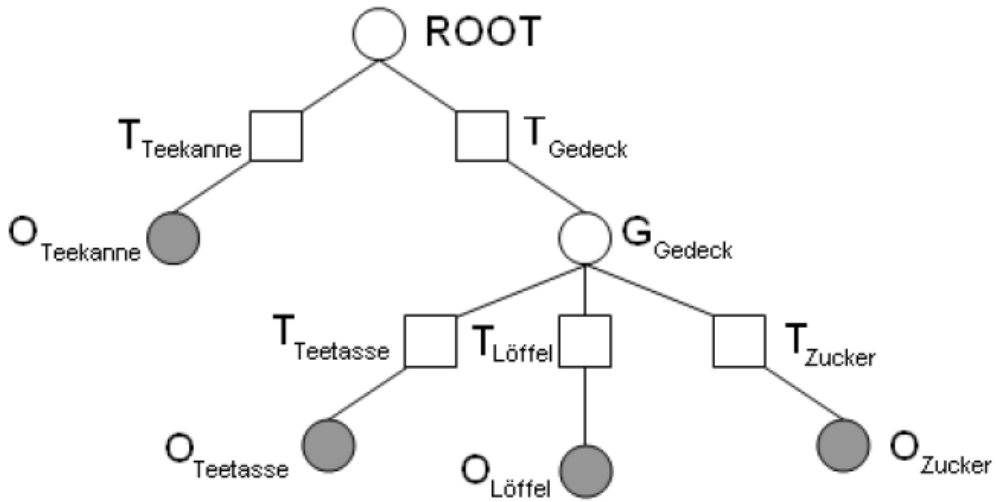


TransformGroup
(Transformation)

z.B. ColorCube, Sphere, Text2D

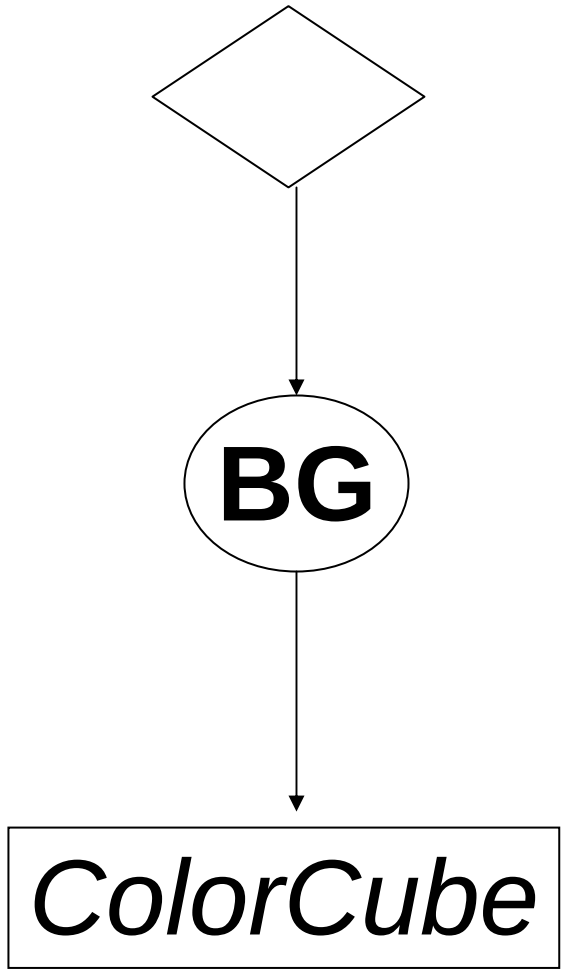


SimpleUniverse (ROOT)



http://download.java.net/media/java3d/javadoc/1.5.0/index.html

Eine erste Szene



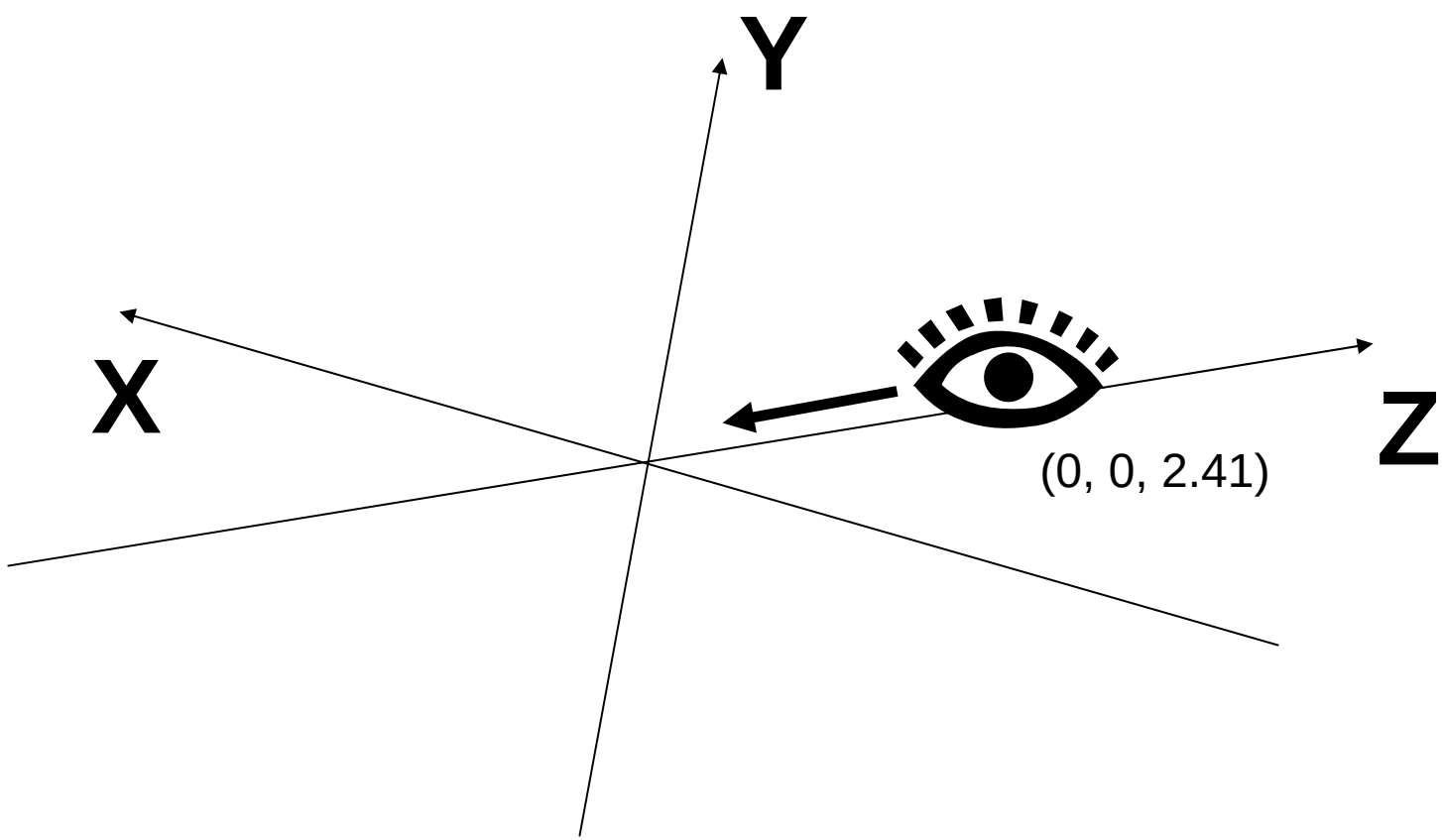
http://download.java.net/media/java3d/javadoc/1.5.0/index.html

Eine erste Szene

```
public HelloUniverse() {
      super();
      this.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
      GraphicsConfiguration config = SimpleUniverse
                        .getPreferredConfiguration();

      Canvas3D c = new Canvas3D(config);
      this.getContentPane().add(c);

      SimpleUniverse u = new SimpleUniverse(c);

      u.getViewingPlatform().setNominalViewingTransform();

      BranchGroup bg = new BranchGroup();
      ColorCube cube = new ColorCube(0.4);
      bg.addChild(cube);
      u.addBranchGraph(bg);
}
```

HelloUniverse.java

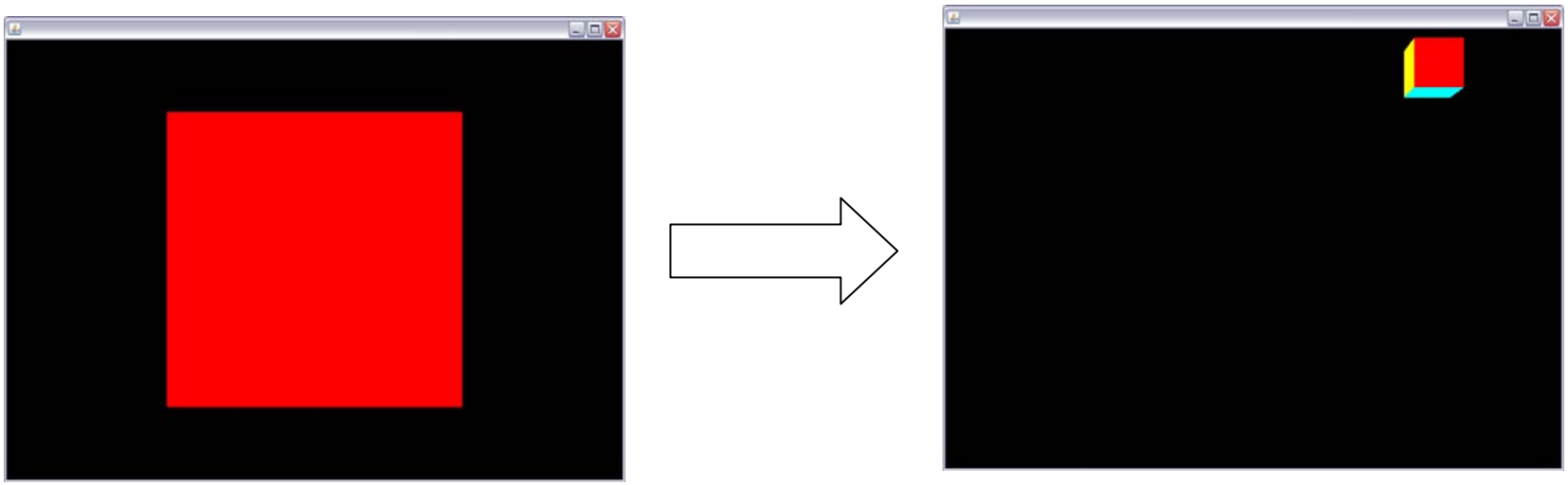
Etwas Geometrie...

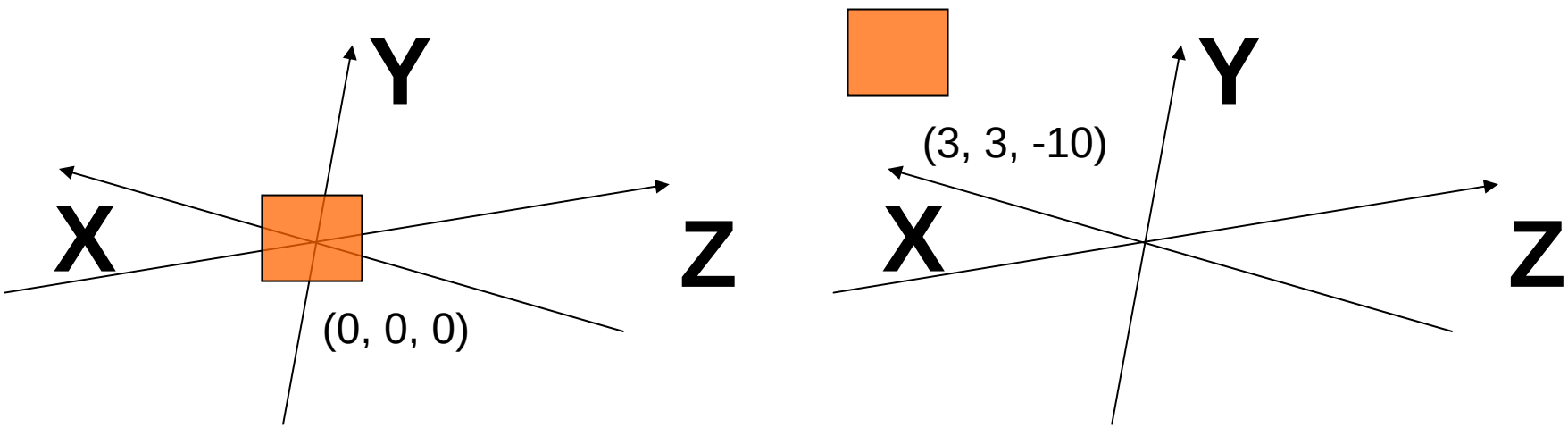


Koordinatensystem ist rechtshändig.

Nach *setNominalViewingTransform()* sitzt die Kamera auf (0, 0, 2.41) und schaut Richtung Ursprung (damit Richtung -Z).

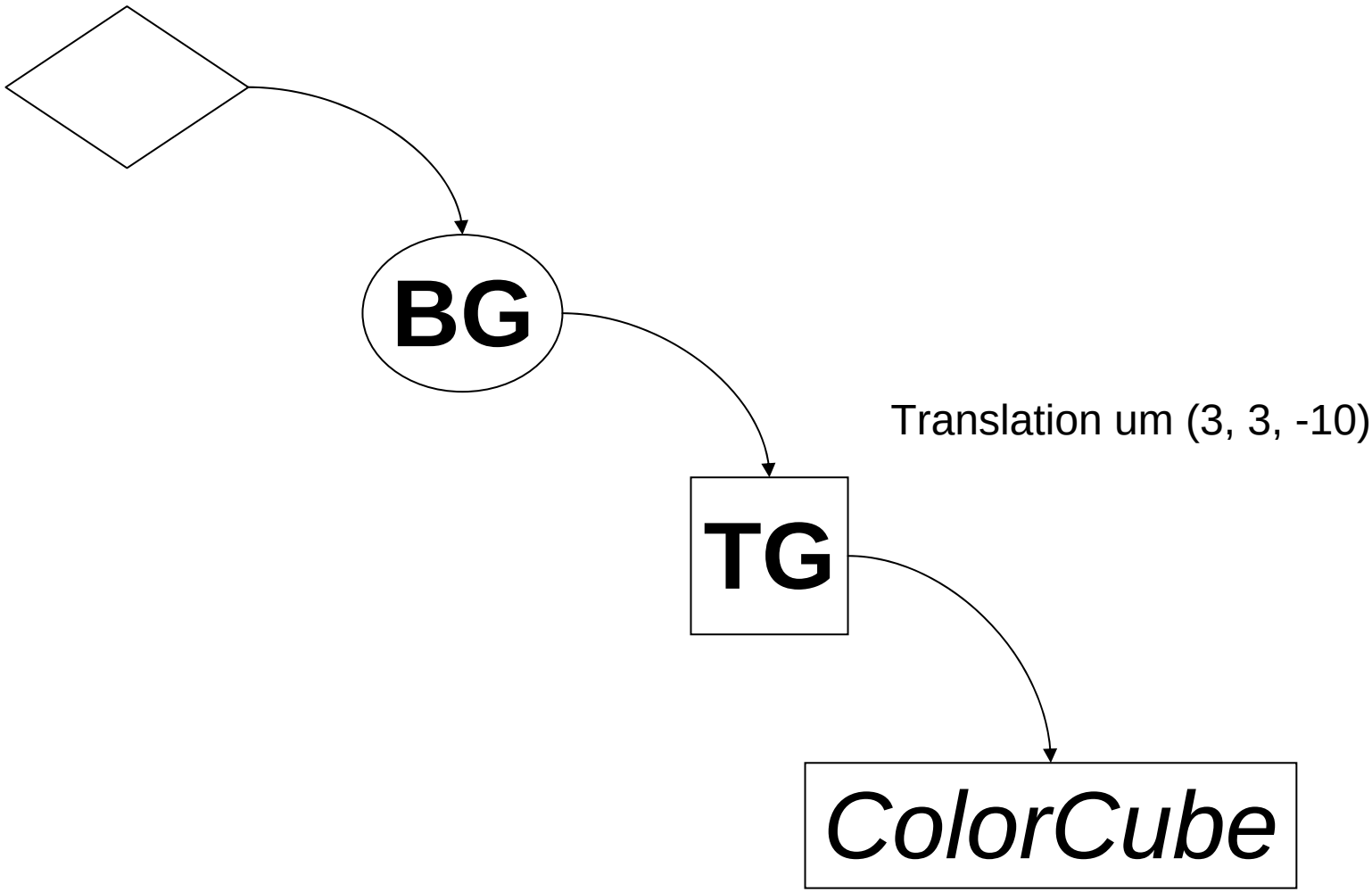
Transformation





http://download.java.net/media/java3d/javadoc/1.5.0/index.html

Transformation im Szenengraph



http://download.java.net/media/java3d/javadoc/1.5.0/index.html

Transformation

```
public HelloUniverse() {
      /* ... */

      BranchGroup bg = new BranchGroup();
      ColorCube cube = new ColorCube(0.4);

      Transform3D verschiebung = new Transform3D();
      verschiebung.setTranslation(new Vector3f(3, 3, -10));

      TransformGroup tg = new TransformGroup(verschiebung);
      tg.addChild(cube);
      bg.addChild(tg);

      u.addBranchGraph(bg);
}
```

HelloUniverse.java

Kombination von Transformationen

```
public HelloUniverse() {
      /* ... */

      BranchGroup bg = new BranchGroup();
      ColorCube cube = new ColorCube(0.4);

      Transform3D verschiebung = new Transform3D();
      verschiebung.setTranslation(new Vector3f(3, 3, -10));

      Transform3D rotation = new Transform3D();
      rotation.rotX(Math.toRadians(60));
      verschiebung.mul(rotation);

      TransformGroup tg = new TransformGroup(verschiebung);
      tg.addChild(cube);
      bg.addChild(tg);

      u.addBranchGraph(bg);
}
```

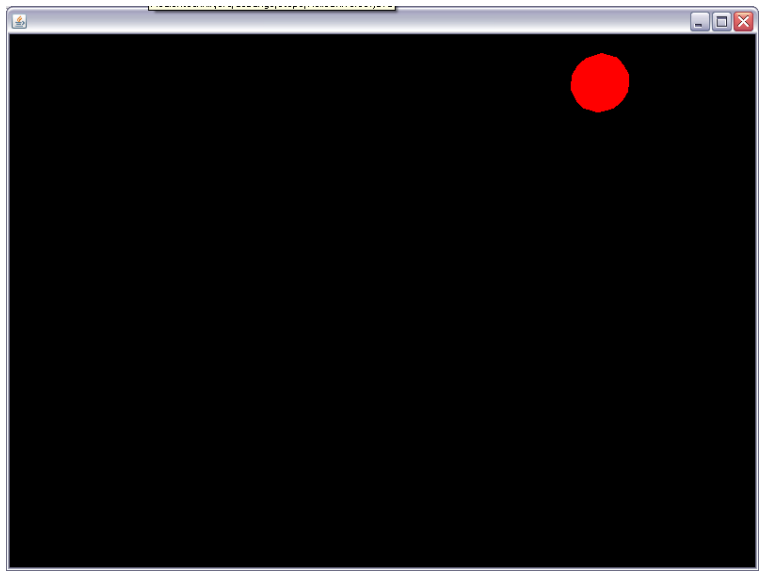
HelloUniverse.java

Appearance

ColorCube hat Farbe und Material vorgegeben, normalerweise sind Objekte (quasi) schwarz!

`Appearance` bestimmt das generelle Aussehen eines Objekts.

Einfachste Variante: Eine Farbe.



http://download.java.net/media/java3d/javadoc/1.5.0/index.html

Appearance

```
public HelloUniverse() {
      /* ... */

      BranchGroup bg = new BranchGroup();
      Sphere ball = new Sphere(0.4f);

      Color3f col = new Color3f(1, 0, 0);
      ColoringAttributes colattr = new ColoringAttributes(col,
                        ColoringAttributes.NICEST);
      Appearance app = new Appearance();
      app.setColoringAttributes(colattr);
      ball.setAppearance(app);

      Transform3D verschiebung = new Transform3D();
      verschiebung.setTranslation(new Vector3f(3, 3, -10));
      /* ... */
}
```

HelloUniverse.java

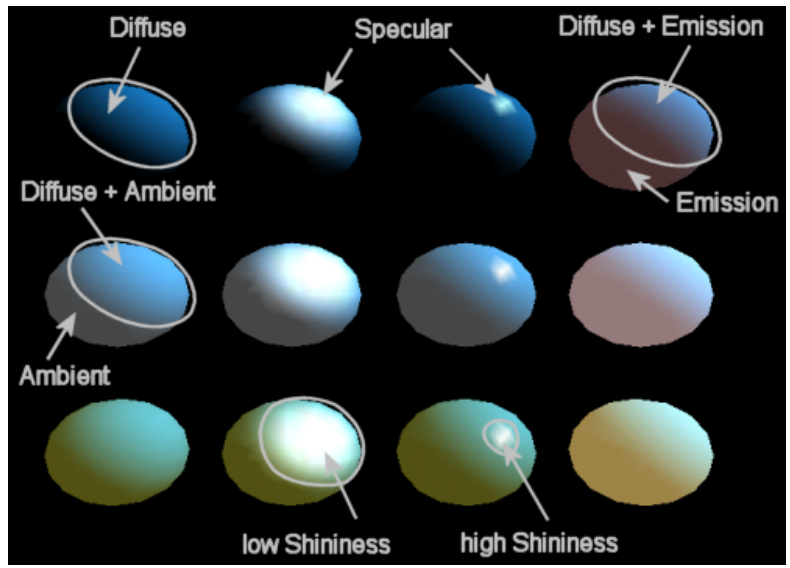
Material

Als Erweiterung einer Farbe lässt sich ein Material setzen.

Materialien bestimmen, wie ein Objekt unter Licht aussieht und setzen sich aus vier verschiedenen Farben zusammen:

- ambient: Grundfarbe des Objekts
- emissive: Farbe in der das Objekt (von sich aus) leuchtet (Glühen)
- diffuse: Licht, das in alle Richtung reflektiert wird
- specular: Licht, das nur in eine Richtung reflektiert wird

+ Shininess: Specular Effekt (von 0 bis 1)



http://jerome.jouvie.free.fr/OpenGl/Tutorials/Tutorial14.php

Material

```
public HelloUniverse() {
      /* ... */

      BranchGroup bg = new BranchGroup();
      Sphere ball = new Sphere(0.4f);

      Color3f white = new Color3f(1, 1, 1);
      Color3f black = new Color3f(0, 0, 0);
      Color3f blue = new Color3f(0, 0, 1);
      Material mat = new Material(blue, black, white, white, 0.6f);
      Appearance app = new Appearance();
      app.setMaterial(mat);
      ball.setAppearance(app);

      Transform3D verschiebung = new Transform3D();
      verschiebung.setTranslation(new Vector3f(3, 3, -10));
      /* ... */
}
```

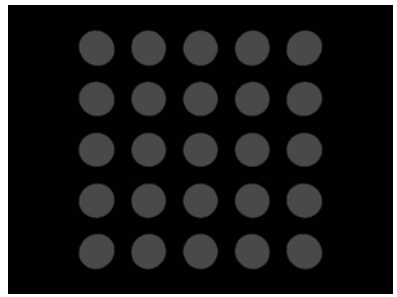
HelloUniverse.java

Licht

Es gibt verschiedene Lichtquellen in Java3D, die wichtigsten sind:

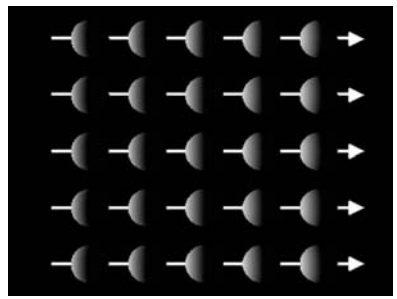
`AmbientLight`

Leuchtet in alle Richtungen gleichmäßig



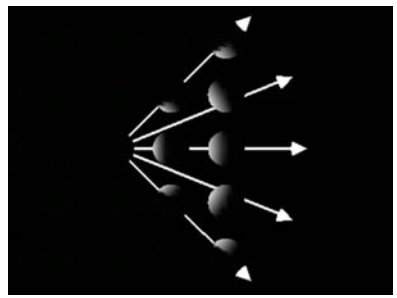
`DirectionalLight`

Parallele Lichtstrahlen, die alle in eine Richtung leuchten



`SpotLight`

Lichtstrahlen stammen aus einem Punkt, verbreiten sich kegelförmig



http://www.sdsc.edu/~nadeau/Courses/Siggraph99/

AmbientLight

```
public HelloUniverse() {
      /* ... */

      Appearance app = new Appearance();
      app.setMaterial(mat);
      ball.setAppearance(app);

      AmbientLight light1 = new AmbientLight(new Color3f(1,1,1));
      BoundingSphere bounds = new BoundingSphere(new Point3d(), 100);
      light1.setInfluencingBounds(bounds);
      bg.addChild(light1);

      Transform3D verschiebung = new Transform3D();
      verschiebung.setTranslation(new Vector3f(3, 3, -10));
      /* ... */
}
```

HelloUniverse.java

DirectionalLight

```
public HelloUniverse() {
      /* ... */
      AmbientLight light1 = new AmbientLight(new Color3f(1,1,1));
      BoundingSphere bounds = new BoundingSphere(new Point3d(), 100);
      light1.setInfluencingBounds(bounds);
      bg.addChild(light1);

      DirectionalLight light2 = new DirectionalLight(
                        new Color3f(.5f, .5f, .5f),
                        new Vector3f(1, 0, -1));
      light2.setInfluencingBounds(bounds);
      bg.addChild(light2);

      Transform3D verschiebung = new Transform3D();
      verschiebung.setTranslation(new Vector3f(3, 3, -10));
      /* ... */
}
```

HelloUniverse.java

Animationen

Einzelne Attribute lassen sich animieren.

`Alpha` stellt dabei eine Art Timer dar, der Werte von 0 bis 1 hochzählt.

Die von `TransformInterpolator` erbenden Klassen nutzen diese Werte, um damit eine Transformation zu manipulieren.

Animationen

```
public HelloUniverse() {
      /* ... */

      TransformGroup tg = new TransformGroup();
      tg.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);

      Alpha rotationAlpha = new Alpha(-1, 2000);
      RotationInterpolator rotator =
              new RotationInterpolator(rotationAlpha, tg);
      rotator.setSchedulingBounds(bounds);
      tg.addChild(rotator);

      tg.addChild(ball);
      bg.addChild(tg);

      u.addBranchGraph(bg);
}
```

HelloUniverse.java

Tastaturinteraktion

```
public class HelloUniverse extends JFrame {

private TransformGroup tg;
private Vector3f position;

      public HelloUniverse() {
            /* ... */
            tg = new TransformGroup();
            tg.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
            position = new Vector3f();

            c.addKeyListener(new KeyAdapter(){
                    public void keyReleased(KeyEvent e){
                            switch(e.getKeyCode()){
                            case KeyEvent.VK_LEFT:
                                    position.x -= 0.1f;
                                    break;
```

HelloUniverse.java

Tastaturinteraktion

```
                    case KeyEvent.VK_RIGHT:
                            position.x += 0.1f;
                            break;
                    case KeyEvent.VK_UP:
                            position.y += 0.1f;
                            break;
                    case KeyEvent.VK_DOWN:
                            position.y -= 0.1f;
                            break;
                    }
                    Transform3D t3d = new Transform3D();
                    t3d.setTranslation(position);
                    tg.setTransform(t3d);
            }
    });
    tg.addChild(ball);
    bg.addChild(tg);

    u.addBranchGraph(bg);
```

HelloUniverse.java