TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

188.912 UE Multimedia

# Drawables und Animationen in Android

Sascha Wiplinger

WS 2014

# Inhalt

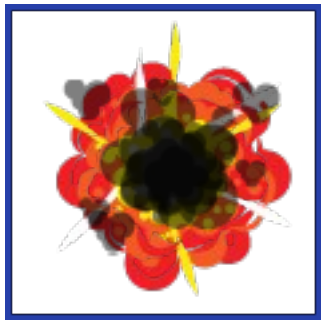
- Einführung
- Drawables
- Tween Animationen
- Frame-by-Frame Animation

# Inhalt

- Einführung
- Drawables
- Tween Animationen
- Frame-by-Frame Animation

# Was animieren?

- Spielegrafiken



- Splash Screens
  - Winner Ceremony
  - Game Over Screen

**GAME OVER**

# Was bietet Android?

- Drawables

- Framework für Animationen
  - View Animationen ← Für Splashscreens diese verwenden
  - Property Animationen ← ...oder diese.
  - (Drawable Animationen) ← Diese nicht verwenden!

# Inhalt

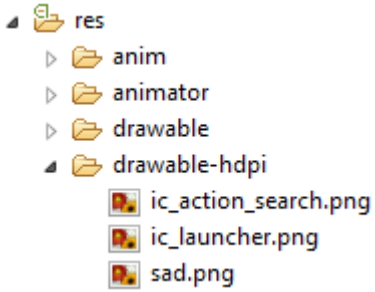
- Einführung
- Drawables
- Tween Animationen
- Frame-by-Frame Animation

# Drawable Ressourcen

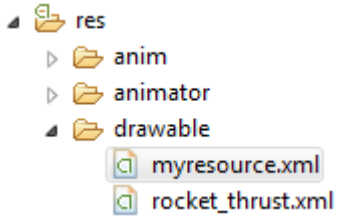
- „Etwas, das man zeichnen kann.“
- **Bitmap**, **Shape**, …
- Zugriff in Java
  - `getRessources().getDrawable(R.drawable.resource_name)`
- Bzw. Attribute in XML-Dateien
  - `android:drawable = …`
  - `android:src = …`
  - `android:background = …`
  - …
- mit dem Pfad
  - `“@drawable/resource_name“`

# Bitmap Ressourcen als Drawable

- PNG (bevorzugt), JPEG (OK) oder GIF (zu vermeiden)
- Ablage in **`res/drawable-...`**
  - **`hdpi`**
  - **`mdpi`**
  - **`...`**




- alternativ auch als Bitmap XML definierbar
  - Wurzelelement **`<bitmap>`**
  - ablegen in **`res/drawable`**
- Vorteil: Verschiedene Attribute für
  - Anti-Aliasing,
  - Tile Mode,
  - ...



# Bitmap Ressourcen als Drawable

- XML

```xml
<?xml version="1.0" encoding="utf-8"?>
<bitmap
xmlns:android="http://schemas.android.com/apk/res/android"
  android:src="@drawable/sad"
  android:tileMode="repeat">
</bitmap>
```

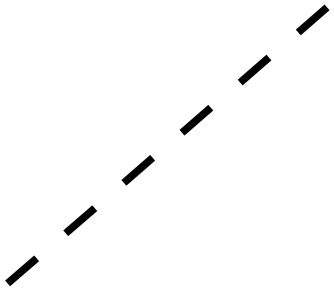
- Java

```java
Drawable pic =
getResources().getDrawable(R.drawable.myresource);
pic.setBounds(10, 10, pic.getIntrinsicWidth(),
pic.getIntrinsicHeight());
pic.draw(canvas);
```

# Shape Ressourcen

- Zeichnen einfacher Formen

```
<?xml version="1.0" encoding="utf-8"?>
<shape
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:shape=["rectangle"|"oval"|"line"|"ring"]>
   <corners ... />
   <gradient ... />
   <padding ... />
   <size ... />
   <solid ... />
   <stroke ... />
</shape>
```

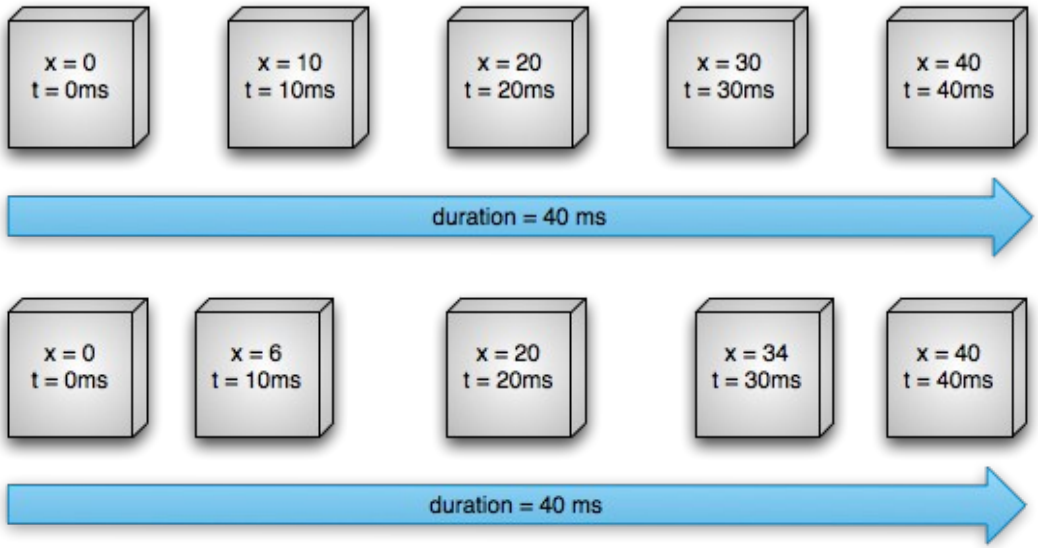


# Inhalt

- Einführung
- Drawables
- Tween Animationen
- Frame-by-Frame Animation

# Tween Animationen

- Interpolation
- zwischen Anfangs- und Endwert
- über Zeit

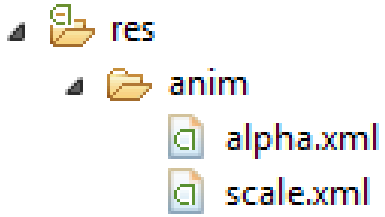


# Tween Animationen

- Für Splash Screens
- 2 Arten von Tween Animationen im Android API
  - View Animationen
  - Property Animationen
- in Java-Code oder XML definierbar
- Definition in XML empfohlen!
  - lesbar
  - wiederverwendbar
  - austauschbar

# View Animationen - Aufbau

- Ordner: **`res/anim`**
- Mögliche Transformationen
  - Translation **`<translate>`**
  - Skalierung **`<scale>`**
  - Rotation **`<rotate>`**
  - Transparenz **`<alpha>`**
- Parameterangaben als Attribute
- Parameter für Zeit:
  - Dauer **`android:duration`**
  - Offset **`android:startOffset`**
  - Wiederholung **`android:repeatMode, android:repeatCount`**



# View Animationen - Codebeispiel

- Animation in XML

```xml
<?xml version="1.0" encoding="utf-8"?>
 <scale xmlns:android="http://schemas.android.com/apk/res/android"
     android:id="@+id/scaleImageView"
     android:fromXScale="0.0"
     android:toXScale="1.5"
     android:fromYScale="0.0"
     android:toYScale="1.5"
     android:duration="2000"
     android:pivotX="50%"
     android:pivotY="50%"
     android:repeatCount="infinite"
     android:repeatMode="reverse">
 </scale>
```

- Einbinden in Java

```java
Animation scaleAnimation = AnimationUtils.loadAnimation(this, R.anim.scale);
ImageView scaleImageView = (ImageView) findViewById(R.id.scaleImageView);
scaleImageView.startAnimation(scaleAnimation);
```

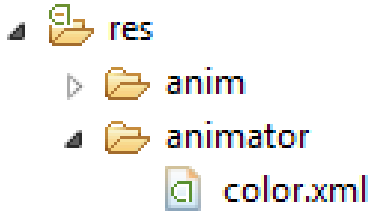
# View Animationen - Zusätzliches

- Werden auf Inhalte von Views angewandt
  - Die Views werden nicht tatsächlich verändert!
- `<set>` zum Gruppieren von Transformationen
- Animationen parallel
  - mit **android:startOffset** zeitversetzt
- Attribut für Interpolatoren
  - `android:interpolator`
  - Diverse Interpolatoren vom API zur Verfügung gestellt
    - Package: `android.view.animation`

# View Animationen

DEMO

# Property Animationen




- Ab API Level 11 (Android 3.0)
- Mächtiger, als View Animationen
  - Animation beliebiger Felder der Objekte (nicht nur von Views)
  - Position zum Klicken ändert sich mit
- Beschreibung der Animation ähnlich wie bei View Animationen
  - Definition in XML, Einbinden und Starten über Java
  - Wurzelelement `<animator>` bzw. `<objectAnimator>`
  - Attribute für Start-, Endwerte, Interpolatoren, **Framerate**...
  - `<set>` unterstützt paralleles, sequentielles,... Abspielen

# Property Animationen - ValueAnimator

- Erweitert abstrakte Klasse Animator
- Basisklasse für Property Animationen
- Wurzelelement `<animator>`
- `start()` spielt Animation ab

| ValueAnimator |
| --- |
| TimeInterpolator |
| TypeEvaluator |
| int duration |
| int startPropertyValue |
| int endPropertyValue |
| start() |

- Berechnungen pro Frame
  - Vergangene **Zeit** (Intervall [0.0, 1.0])
  - Animationsfortschritt mittels **Interpolator** (Intervall [0.0, 1.0])
  - eigentlichen Wert mittels **TypeEvaluator**

# Property Animation - ValueAnimator

- Notification: Neuer Frame berechnet
- Update auf Objekte muss selbst implementiert werden!
  - ähnlich zu OnClickListener
  - Interface **`ValueAnimator.AnimatorUpdateListener`** implementieren
  - Abstrakte Methode **`onAnimationUpdate()`**
  - **`getAnimatedValue()`** liefert aktuellen Wert als **`Object`**

# Property Animationen - TypeEvaluator

- Interface
- Gibt an, wie eine bestimmte Property berechnet wird
- Verwendet zur Berechnung
  - Die Timing-Daten des Animators
  - Anfangs- und Endwerte
- Verfügbare Implementierungen
  - IntEvaluator
  - FloatEvaluator
  - ArgbEvaluator
- Alles andere selbst implementieren.

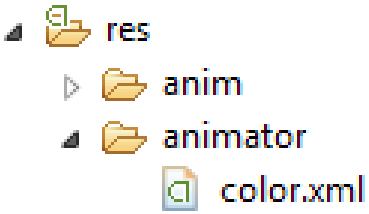
# Property Animationen – ObjectAnimator

- In XML Wurzelelement **<objectAnimator>**
- Unterklasse von ValueAnimator
- hat Zielobjekt → Update automatisch
- Verwendung einfacher, aber mehr Einschränkungen, als ValueAnimator
  - Zielobjekt braucht **Getter-** und **Settermethode**
  - für Property mit dem angegebenen **Namen**
  - mit **Typ** entsprechend dem Start- und Endwert des ObjectAnimators

# Property Animationen - Codebeispiel

- XML in **res/animator**

```
<?xml version="1.0" encoding="utf-8"?>
<objectAnimator xmlns:android="http://schemas.android.com/apk/res/android"
  android:propertyName="textColor"
  android:valueFrom="#000000"
  android:valueTo="#FF0000"
  ...
</objectAnimator>
```




- Einbinden in Java

```
TextView colorTextView = (TextView) findViewById(R.id.colorTextView);
ObjectAnimator textAnimator =
   (ObjectAnimator) AnimatorInflater.loadAnimator(this, R.animator.color);
textAnimator.setEvaluator(new ArgbEvaluator());
textAnimator.setTarget(colorTextView);
textAnimator.start();
```

# Property Animationen

DEMO

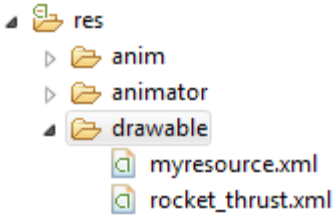
# Tween Animationen – Vergleich

| | View Animation | Property Animation |
|---|---|---|
| **Auf Views anwendbar** | Mit vorgegebenen Transformationen | Ja |
| **Auf beliebige Objekte anwendbar** | Nein | Ja |
| **Immer an gezeichneter Position klickbar** | Nein | Ja |
| **Erweiterbar** | Nein | Ja |
| **< API Lvl 11 (Android 3.0) unterstützt** | Ja | Nein |
| **Komplexität** | Einfacher | Aufwändiger |

# Inhalt

- Einführung
- Drawables
- Tween Animationen
- Frame-by-Frame Animation

# Drawable Animation




- Frame-by-Frame Animation
- Bitmaps werden der Reihe nach in eine View gezeichnet
- XML
  - in **res/drawable**

```
<animation-list xmlns:android="http://schemas.android.com/apk/res/android"
    android:oneshot="true">
    <item android:drawable="@drawable/rocket_thrust1" android:duration="200" />
    <item android:drawable="@drawable/rocket_thrust2" android:duration="200" />
    <item android:drawable="@drawable/rocket_thrust3" android:duration="200" />
</animation-list>
```

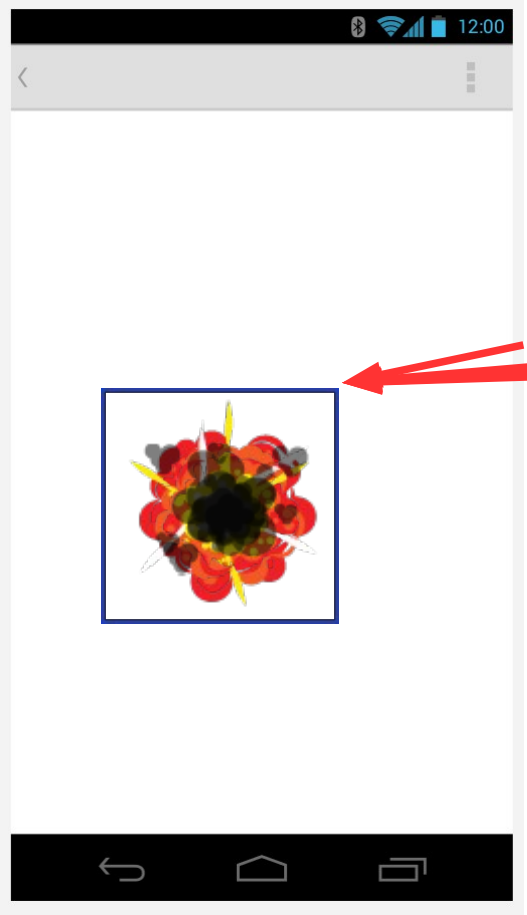
- Einbinden in Java

```
ImageView rocketImage = (ImageView) findViewById(R.id.rocket_image);
rocketImage.setBackgroundResource(R.drawable.rocket_thrust);
AnimationDrawable rocketAnimation = (AnimationDrawable) rocketImage.getBackground();
rocketAnimation.start();
```

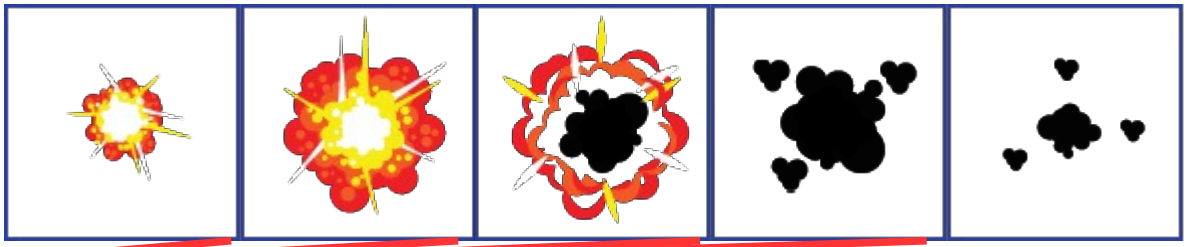
# Drawable Animation – PROBLEM!

- **PROBLEM:**
  - Drawable Animationen verwenden **`setBackground()`**
  - wir Zeichnen im Spiel am Canvas
    - `drawRectangle(new Rect(int x, int y, int width, int height), Paint p);`
  - Keine Möglichkeit, mit Drawable Animationen am Canvas zu positionieren
- **ABHILFE: Sprite Animation**
  - Selbst implementieren

# Sprite Animation




sprite sheet:



von vorne beginnen?

```
canvas.drawBitmap(spriteSheet, sourceRect, destRect, null);
```

# Sprite Animation

- Informationen für die Klasse:
  - Spritesheet (Bitmap Resource)
  - Anzahl der Frames (int)
  - Position am Canvas (int x, int y)
  - Bilder pro Sekunde (int)
  - Animation in Endlosschleife? (boolean)
- → Konstruktor

# Sprite Animation - Beispielklasse

```
public class SimpleAnimatedSprite {
...
   //Getter- und Settermethoden für Position, z.B.:
   public int getX() { … }
   public void setX(int x) { … }
...
   //prüft, ob der nächste Frame angezeigt werden soll, sofern die Animation noch
   //nicht abgeschlossen ist.
   public void update(long gameTime) { … }

   //zeichnet den aktuellen Frame des sprite sheets auf das Canvas
   public void draw(Canvas canvas) {
      if (!isStopped) {
         Rect destRect =
            new Rect(getX(), getY(), getX() + spriteWidth, getY() + spriteHeight);
         canvas.drawBitmap(spriteSheet, sourceRect, destRect, null);
      }
}
```

# Sprite Animation

DEMO

# Anforderungen

- Splash Screens
  - Mindestens einen
  - Tween Animation
- Spielgrafiken
  - Sprite Animation (eigene Klasse)

# Quellen

- Android Animation Framework
  - Animation and Graphics Overview | Android Developer
    http://developer.android.com/guide/topics/graphics/overview.html
  - Animation Resources | Android Developer
    http://developer.android.com/guide/topics/resources/animation-resource.html
  - Drawable Resources | Android Developer
    - http://developer.android.com/guide/topics/resources/drawable-resource.html
- Sprite Animation
  - Sprite Animation with Android | Against the Grain - Game Development
    - http://obviam.net/index.php/sprite-animation-with-android/

# Fragen?