



libGDX

Teil 4



© 2012-18 by Thomas Helml



Inhaltsverzeichnis

- ① Features
- ① Das Framework
- ① Life-Cycle
- ① Projekterstellung
- ① Starter Classes
- ① Grundgerüst



Ziele

IT Links

IT libGDX: <http://libgdx.badlogicgames.com>

IT Wiki: <https://github.com/libgdx/libgdx/wiki>

IT libGDX ist

IT ein Java Game Development Framework

IT eine API für alle Plattformen

IT keine Spielengine (wie z.B. Unity)



Features

IT Cross Plattform

- IT Windows
- IT Linux
- IT Mac OS X
- IT Android (2.2+)
- IT BlackBerry
- IT iOS
- IT Java Applet
- IT JavaScript/WebGL



Features

IT Audio

- IT Streaming + Sound FX (ogg, wav, mp3)
- IT direkter HW-Zugriff für PCM Audio

IT Input

- IT Abstraktion für Maus, Touch, Keyboard, Gyro, Kompass
- IT Gesture Detection



Features

- IT Math+Physik Engine

- IT File I/O

 - IT Abstrahiertes File System für alle Plattformen

- IT Grafik

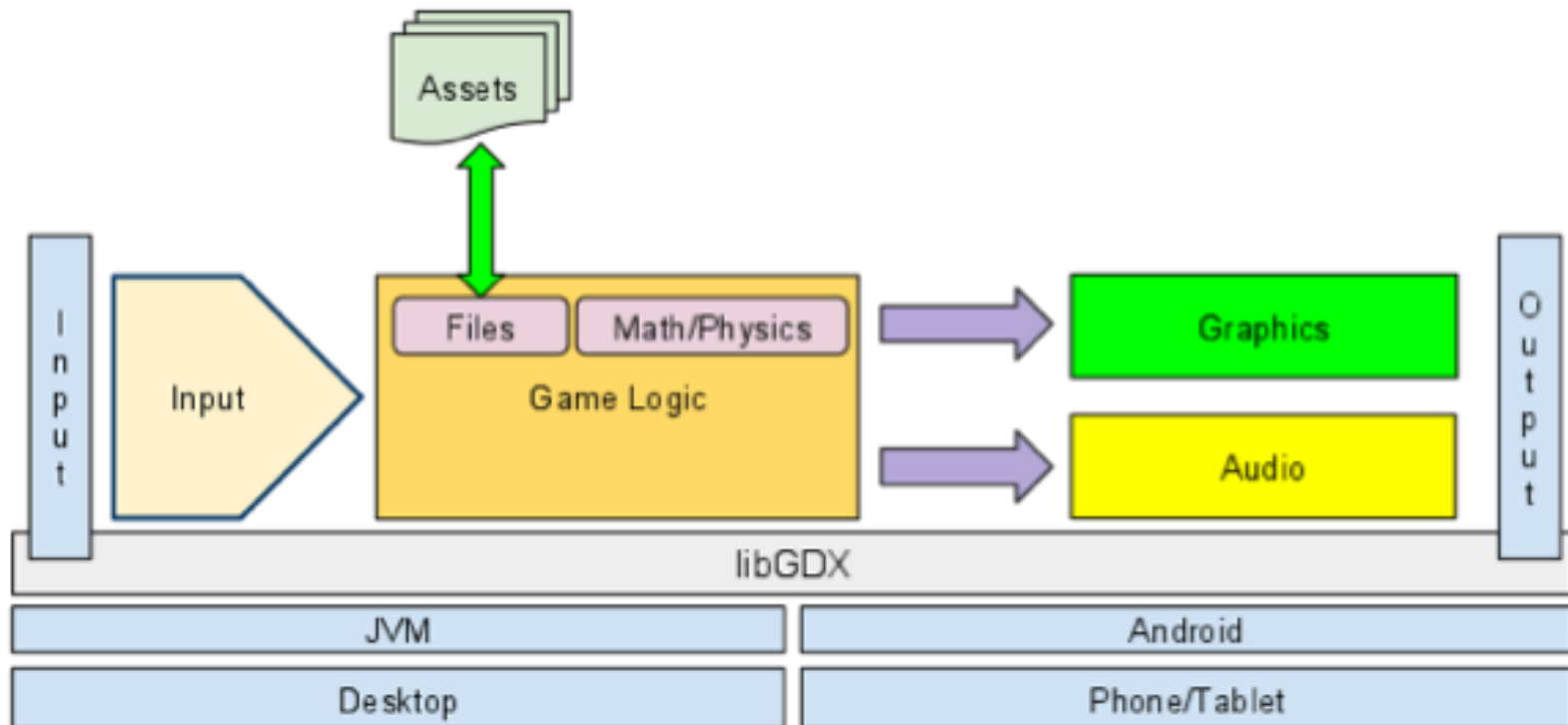
 - IT OpenGL ES 2.0 (OpenGL for Embedded Systems)

 - IT 2D API (z.B. TMX Map Support, ...)

 - IT 3D API



Das Framework





Das Framework

IT 6 wichtige Module:

IT Application

IT Files

IT Input

IT Net

IT Audio

IT Graphics



Das Framework

① Application

- ① Startet Anwendung

- ① Informiert über Events auf App-Ebene

 - ① z.B. Änderung der Fenstergröße

- ① Logging

- ① Abfrage Methoden

 - ① z.B. Speichergröße



Das Framework

IT Files

- IT Abstrahiert Dateizugriff
- IT Methoden inkompatibel mit Java File Klassen

IT Input

- IT Maus, Tastatur, Touch, Gyroskop, ...
- IT Polling und Ereignissteuerung



Das Framework

IT Net

- IT Zugriff auf Ressourcen über HTTP(S)
- IT Socket Kommunikation

IT Audio

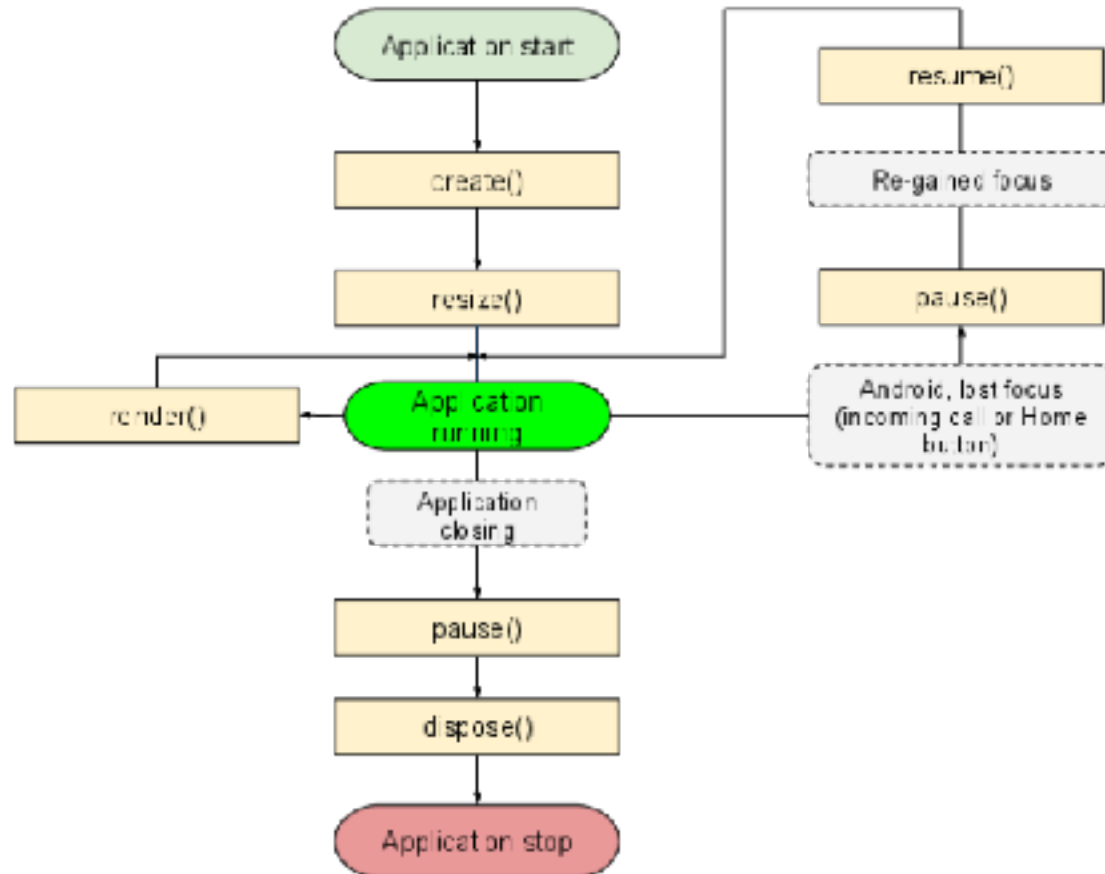
- IT Abspielen von Sound FX und Musik
- IT Direkter HW Zugriff über PCM Audio

IT Graphics

- IT Zugriff auf OpenGL ES 2.0
- IT Video Modus ändern, ...



libGDX-Life-Cycle





libGDX-Life-Cycle

- ① jedes Spiel muss das Interface `ApplicationListener` (bzw. `ApplicationAdapter`) implementieren
- ① diese Klasse muss in den sog. Starterklassen registriert werden
- ① die App ruft je nach Ereignis die entsprechenden Methoden auf



libGDX-Life-Cycle

```
public class MyGame implements ApplicationListener {  
    public void create () {}  
    public void render () {}  
    public void resize (int width, int height) {}  
    public void pause () {}  
    public void resume () {}  
    public void dispose () {}  
}
```



libGDX-Life-Cycle

① `create ()`

① Aufruf nach Anwendungsstart

① `resize(int width, int height)`

① Aufruf nach `create ()`

① jedes Mal, wenn Fenstergröße geändert wird

① `dispose ()`

① Aufruf, wenn Anwendung beendet wird



libGDX-Life-Cycle

IT render ()

- IT Aufruf in unregelmäßigen Abständen
- IT immer wenn Spiel neu gezeichnet werden muss
- IT Spiellogik muss hier behandelt werden
- IT Entspricht Konzept der „Gameloop“



libGDX-Life-Cycle

IT pause ()

- IT Wichtig für Android
- IT Aufruf wenn App in Hintergrund geht (Telefon, Homebutton, ...)
- IT Spielstatus muss/soll hier gesichert werden
- IT Am Desktop: Aufruf nur vor `dispose ()`

IT resume ()

- IT Nur Android: Aufruf nach Ende von `pause ()`



Introduction to libGDX

 <http://youtu.be/BTC922ki2mc>



Aufgabe

① Download libGDX

① [http://libgdx.badlogicgames.com/
download.html](http://libgdx.badlogicgames.com/download.html)

① Projekt Test_libGDX erzeugen+starten



Projekterstellung

libGDX Project Generator

libGDX PROJECT SETUP

Name:

Package:

Game class:

Destination:

Android SDK:

Sub Projects

☒ Desktop ☐ Android ☐ iOS ☐ iOS-mac ☐ HTML

Extensions

☐ Bullet ☐ Freetype ☐ Tools ☐ Controllers ☐ Box2d

☐ Box2dlights ☐ Ashley ☐ Ai

line_warnings

BUILD SUCCESSFUL in 3s

7 actionable tasks: 5 executed, 2 up-to-date

Done!

To import in Eclipse: File -> Import -> General -> Existing Projects into Workspace

To import to IntelliJ IDEA: File -> Open -> YourProjectLib

Advanced Settings

Settings	Description
Maven Mirror...	Replaces Maven Central with this repository
IDEA <input checked="" type="checkbox"/>	Generates IntelliJ IDEA project files
Eclipse <input type="checkbox"/>	Generates Eclipse project files
Offline Mode <input checked="" type="checkbox"/>	Don't force download dependencies
Use Kotlin <input type="checkbox"/>	Use Kotlin as the main language.

[Click for more info on using Gradle without IDE integration](#)



Projekterstellung

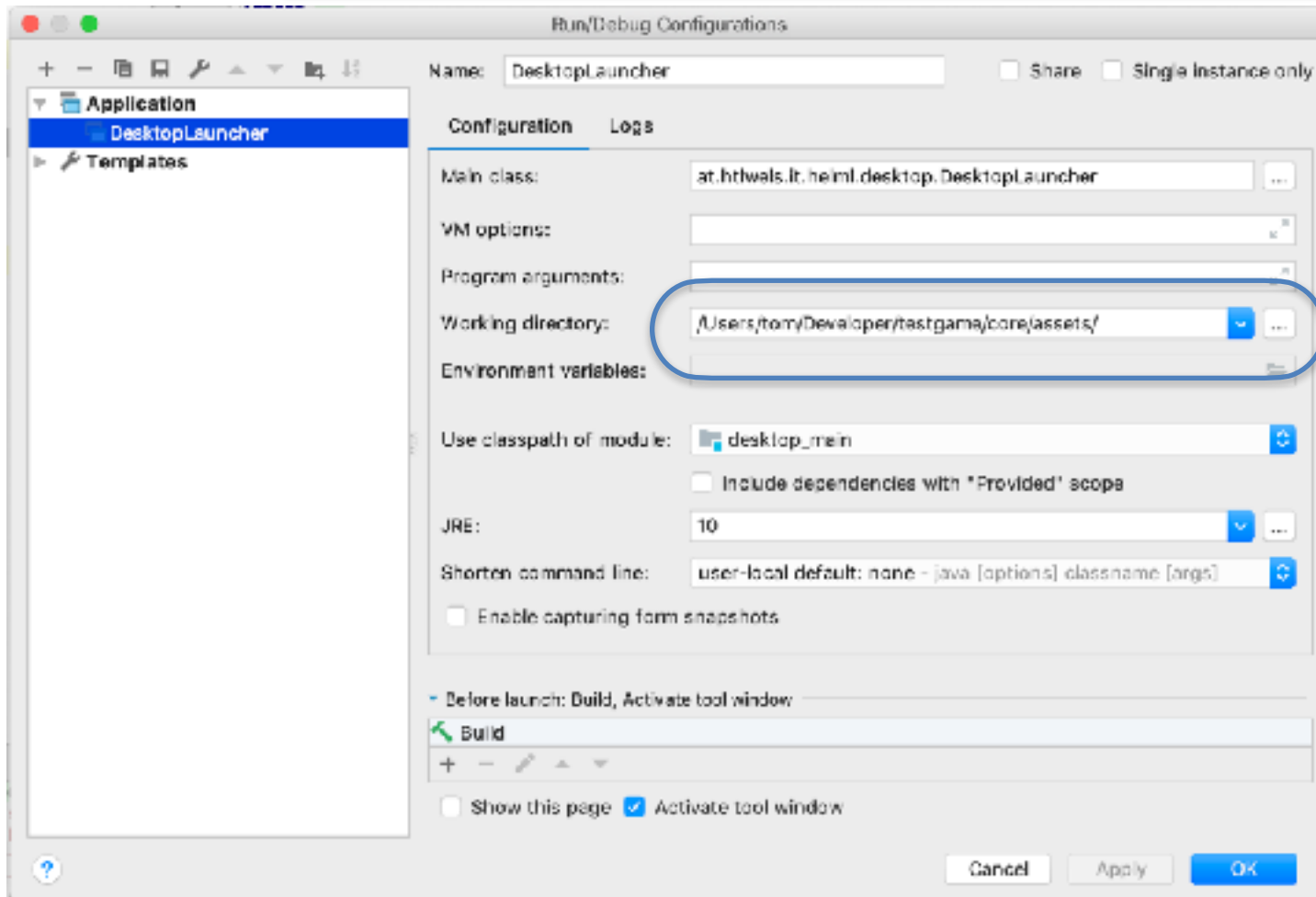
IT File-Open-Projekt

IT nach Build: Runtime Exception

```
WARNING: Please consider reporting this to the maintainers of org.lwjgl.LWJGLUtil$3
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
Exception in thread "LWJGL Application" com.badlogic.gdx.utils.GdxRuntimeException: Couldn't load file: badlogic.jpg
    at com.badlogic.gdx.graphics.Pixmap.<init>(Pixmap.java:149)
    at com.badlogic.gdx.graphics.TextureData$Factory.loadFromFile(TextureData.java:98)
    at com.badlogic.gdx.graphics.Texture.<init>(Texture.java:100)
    at com.badlogic.gdx.graphics.Texture.<init>(Texture.java:92)
```



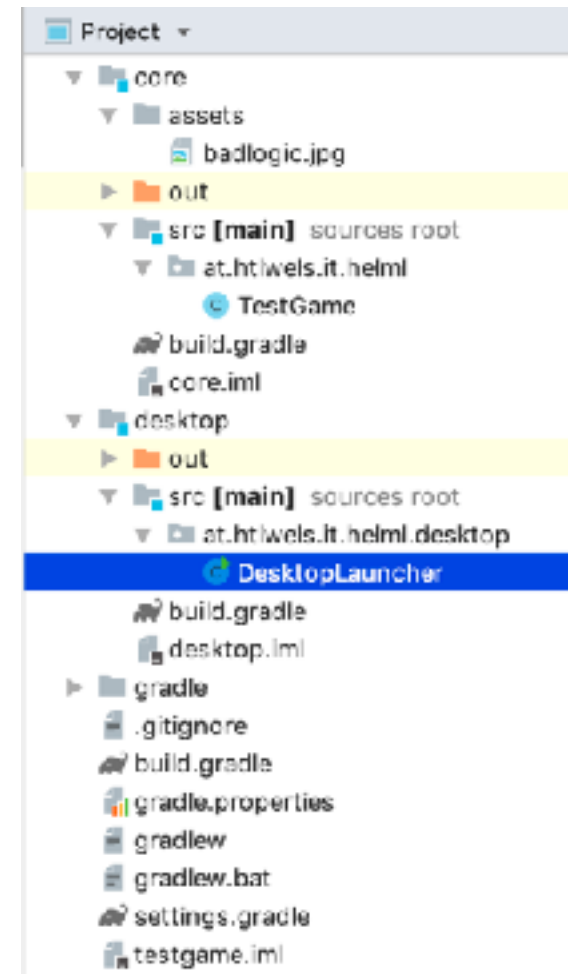
Projekterstellung





Starter Classes

① einziger Code, der plattformabhängig ist





Starter Classes

IT Beispiel Desktop

```
public class DesktopStarter {  
    public static void main(String[] argv)  
    {  
        LwjglApplicationConfiguration config =  
            new LwjglApplicationConfiguration();  
        new LwjglApplication(new MyGame(), config);  
    }  
}
```




Starter Classes

IT Beispiel Android

```
public class AndroidStarter extends AndroidApplication{
    public void onCreate(Bundle bundle){
        super.onCreate(bundle);
        AndroidApplicationConfiguration config =
            new AndroidApplicationConfiguration();
        initialize(new MyGame(), config);
    }
}
```



create()

```
public void create () {  
    batch = new SpriteBatch();  
    img = new Texture("badlogic.jpg");  
}
```

SpriteBatch:



- Benötigt um 2D Rechtecke zu zeichnen (Textures)
- Alle Zeichenoperationen eines SpriteBatch beziehen sich auf Screenkoordinaten
- Links unten Null-Punkt
- max. ein SpriteBatch Objekt je Spiel
- muss freigegeben werden (dispose)



create()

```
public void create () {  
    batch = new SpriteBatch();  
    img = new Texture("badlogic.jpg");  
}
```

Texture:

-  Grafik, die in den Grafikspeicher geladen wird
-  Grafiken MÜSSEN Breite/Höhe haben, die 2er Potenzen sind (16x16, 64x256 ...)



render()

```
public void render () {  
    Gdx.gl.glClearColor(1, 0, 0, 1);  
    Gdx.gl.glClear(GL20.GL_COLOR_BUFFER_BIT);  
    batch.begin();  
    batch.draw(img, 0, 0);  
    batch.end();  
}
```

① `glClearColor(1, 0, 0, 1);`

① `RGBA(1,0,0,1) => Rot`

① letzter Parameter für Alpha-Kanal

① `Gdx.gl.glClear(GL20.GL_COLOR_BUFFER_BIT);`

① OpenGL Befehl für Bildschirm löschen



render()

```
public void render () {  
    Gdx.gl.glClearColor(1, 0, 0, 1);  
    Gdx.gl.glClear(GL20.GL_COLOR_BUFFER_BIT);  
    batch.begin();  
    batch.draw(img, 0, 0);  
    batch.end();  
}
```

① **begin() , end();**

① alle Zeichenoperationen müssen zw. begin() und end() stehen

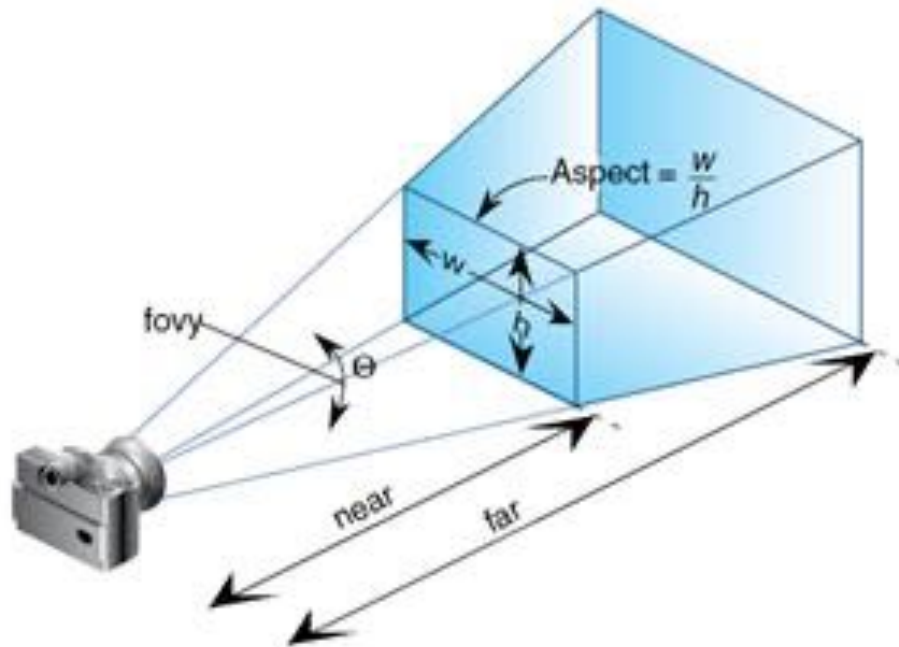
① **draw(img, 0, 0);**

① zeichnet Texture an der Position (0/0)



OrthographicCamera

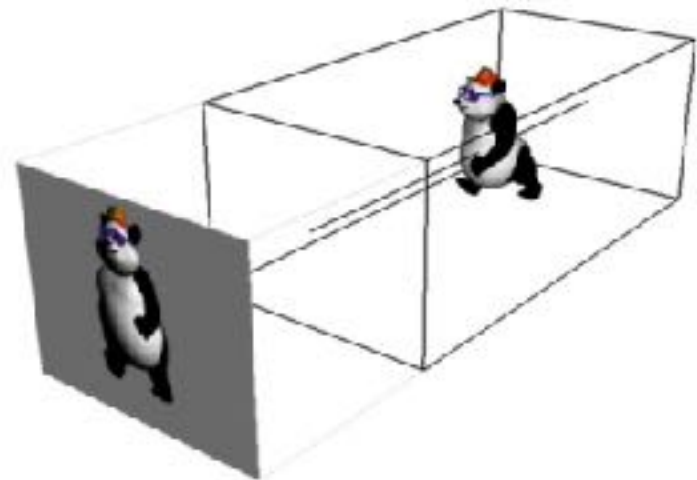
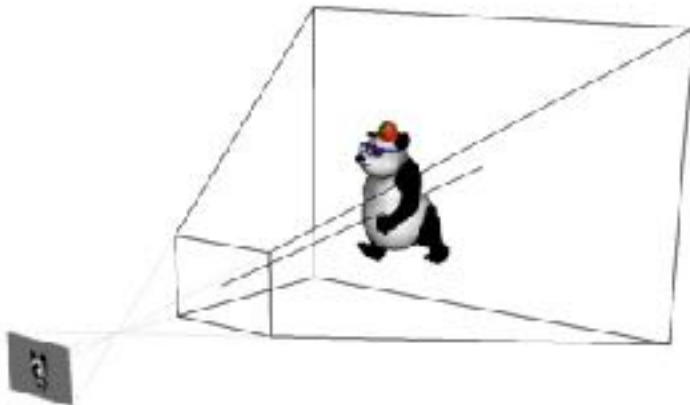
IT Camera „sieht“ Ausschnitt aus realer Welt





OrthographicCamera

Perspektivische Projektion Orthografische Projektion





OrtographicCamera

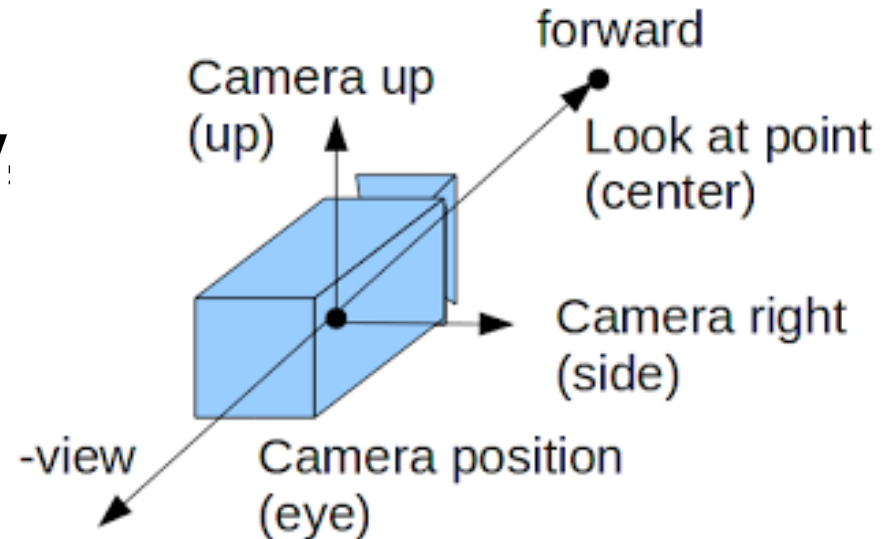
- IT Camera ist vergleichbar mit Kamera IRL
 - IT Bewegen
 - IT Rotieren
 - IT Zoom in/out
 - IT Ändern des Betrachtungspunktes
 - IT ...



OrthographicCamera

① Position der Camera

① Raumkoordinaten (x,y,z)



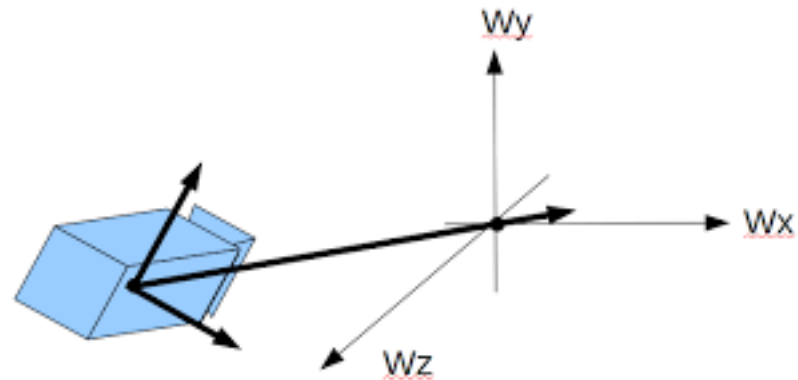
```
cam = new OrthographicCamera( ... );  
cam.position.set(float x, float y, float z);  
cam.update();
```



OrthographicCamera

IT Camera hat Ausrichtung

IT dh. sie „schaut“ auf bestimmte Koordinate in der Welt

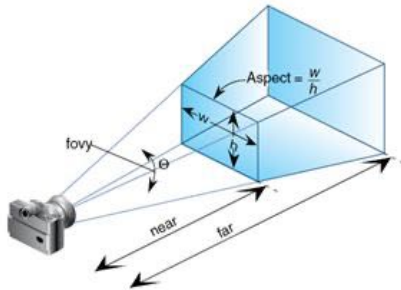


```
cam = new OrthographicCamera( ... );  
cam.lookAt(float x, float y, float z);  
cam.update();
```



OrthographicCamera

IT Camera kann im Raum bewegt werden



```
cam = new OrthographicCamera( ... );  
cam.translate(float x, float y, float z);  
cam.update();
```



OrthographicCamera

IT Rotieren der Camera:

```
cam.rotate ( float angle,  
             float axisX,  
             float axisY,  
             float axisZ);
```

IT Rotation um Achse (bestimmt durch Vektor) um angle Grad



OrthographicCamera

IT Render-Methode

```
@Override
public void render() {
    handleInput();
    // camera Position, etc. aktualisieren
    cam.update();
    // „Zeichenfläche“ aktualisieren mit neuer
    // Camera-Projektion
    batch.setProjectionMatrix(cam.combined);

    Gdx.gl.glClear(GL20.GL_COLOR_BUFFER_BIT);

    batch.begin();
    mapSprite.draw(batch);
    batch.end();
}
```



OrthographicCamera

Constructor:

```
public OrthographicCamera(float viewportWidth,  
                           float viewportHeight)
```

Parameters:

viewportWidth – viewport Breite

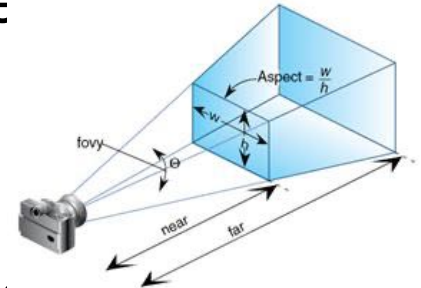
viewportHeight – viewport Höhe



OrthographicCamera

viewPort:

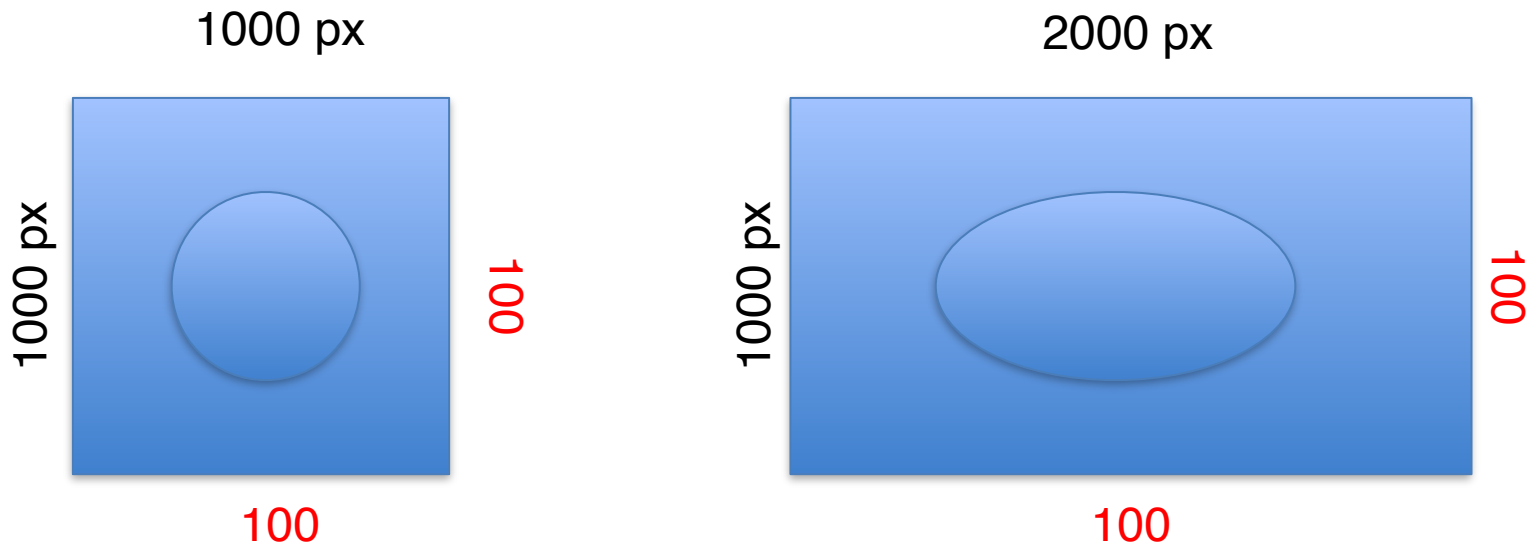
- Größe des sichtbaren Bereichs (=Fenster)
- Angabe in „Welteinheiten“ (was sieht man)
 - Einheit z.B. m, cm, Bausteinbreite
 - nicht in Pixel denken!
- Beispiel: Fullscreen
 - Tatsächl. Auflösung: 800x600 Pixel
 - Weltkoordinaten – sichtbarer Bereich: 40x30m
 - Was passiert, wenn Auflösung geändert wird?





OrthographicCamera

 Bildschirm:





OrthographicCamera

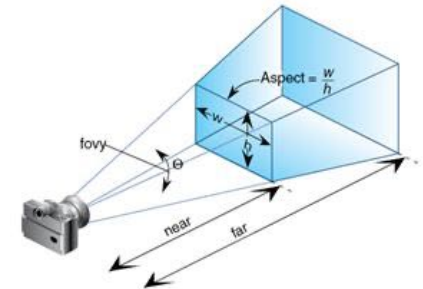
IT LÖSUNG

IT Viewport muss Proportionen des Fensters berücksichtigen!

IT Breite/Höhe, z.B.: 4:3

IT dazu gibt es speziellen Viewport:

```
viewport = new FitViewport(WORLD_WIDTH, WORLD_HEIGHT, cam);
```





Sprite

IT Sprite vs. Texture

- IT Sprite braucht eine Texture (=Grafik)
- IT Sprites können vergrößert/-kleinert werden
- IT Sprites können rotiert werden
- IT Sprites können transparent werden

IT Texture = Oberfläche

IT Sprite = Struktur



Sprite

IT Klasse Sprite

IT beinhaltet

- IT Farbe

- IT Texture

- IT Geometrie (Rotieren, Skalieren, ...)

IT in einem Objekt

- IT Position muss mit `.setPosition()` gesetzt werden!



Sprite

IT Größe der Texturen in der Welt bestimmen:

```
Sprite.setSize(WORLD_WIDTH, WORLD_HEIGHT);
```

IT Create-Methode:

```
mapSprite = new Sprite(new Texture(  
    Gdx.files.internal("sc_map.png")));  
mapSprite.setPosition(0, 0);  
mapSprite.setSize(WORLD_WIDTH, WORLD_HEIGHT);
```