TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

Mobile Game Programming

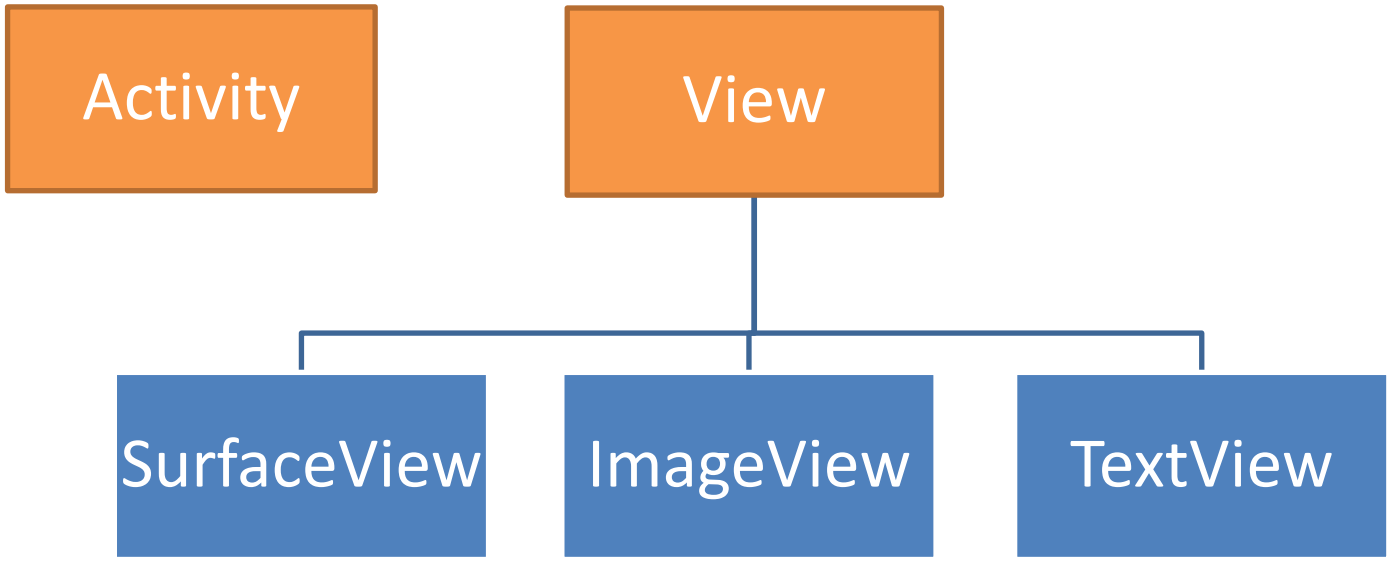
Android API

Andrea Diwald
Vienna University of Technology
WS 2014

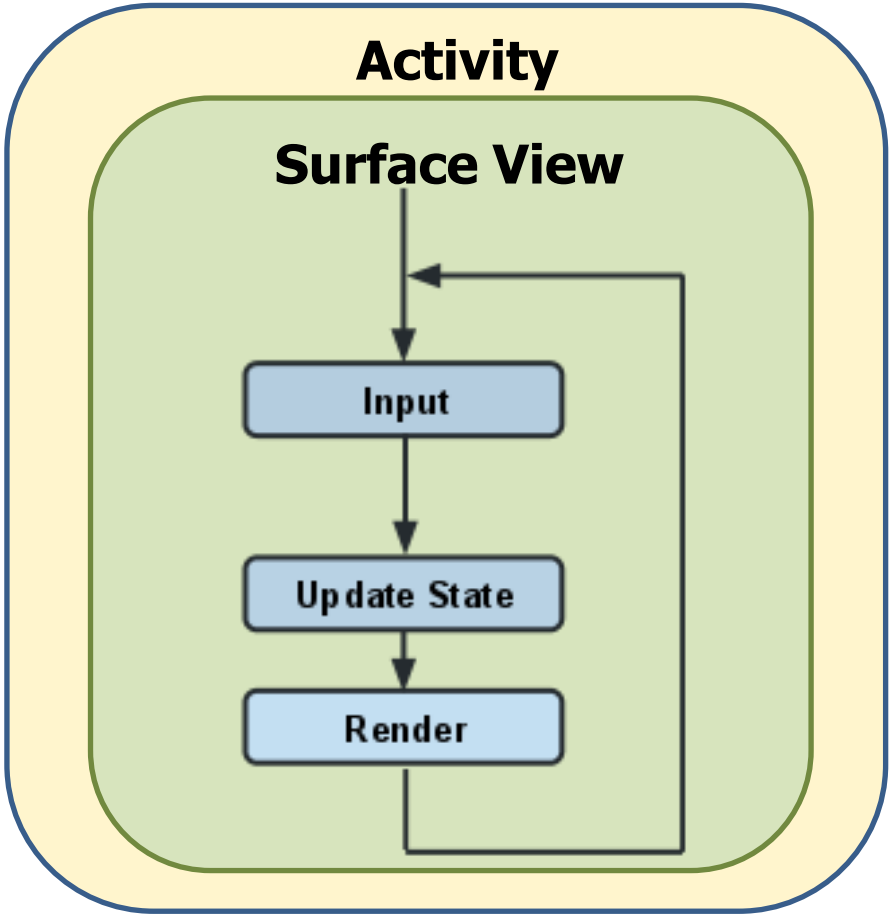
Inhalt

- Android API
  - Activity
  - SurfaceView
    - Canvas
    - Touch Screen/Tastatur
  - Thread/GameLoop
  - Collision detection
  - Ressources
  - Debugging: LogCat
- Tipps & Tricks

Android Api - Class Hierachy

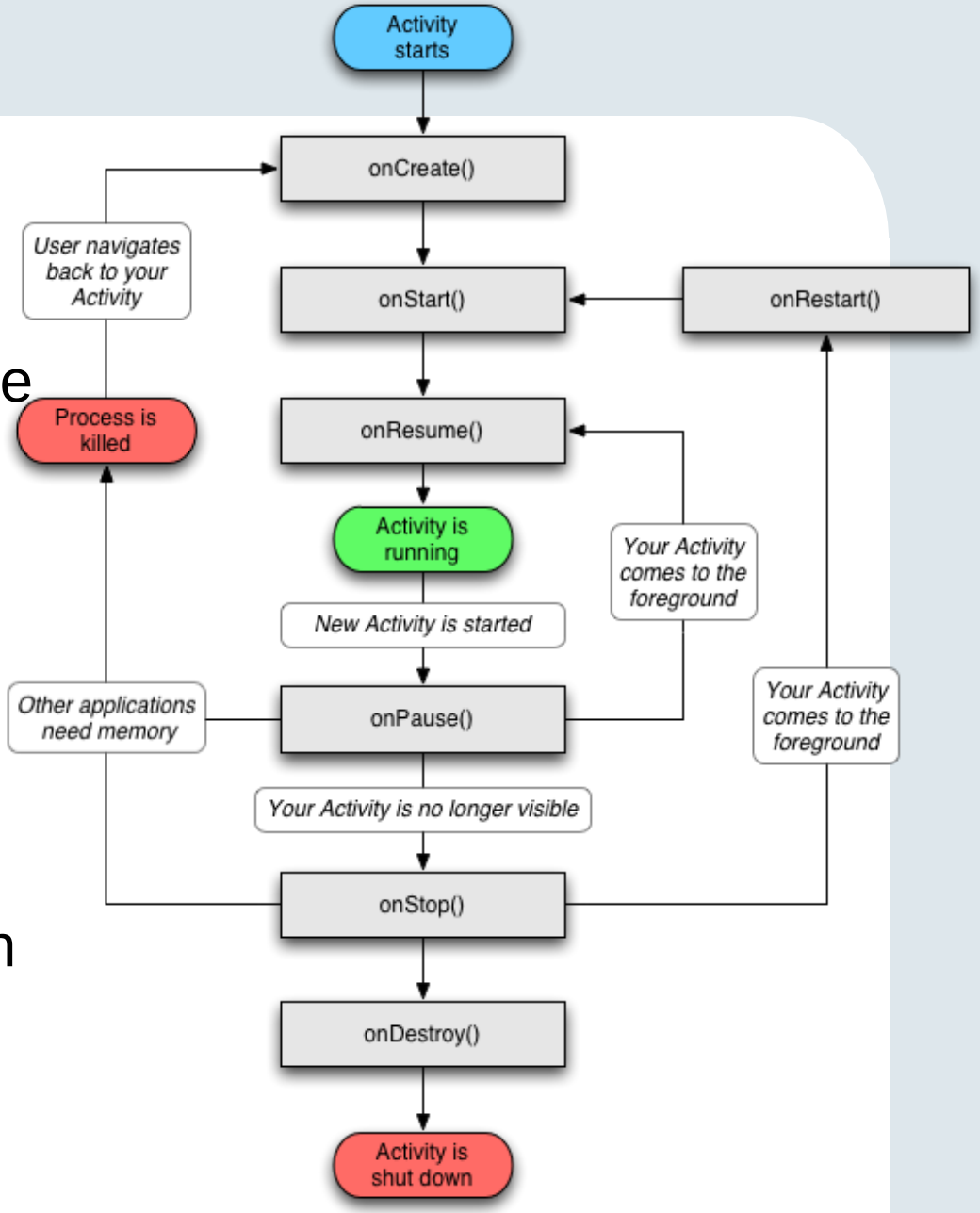


Game Loop Grundlagen



Activity

- Applikationskomponente
- Stellt Screen bereit
- Schnittstelle für Userinteraktion
- Mehrere Activities möglich
- Eine Activity kann auch andere Activities starten



Activity

- Beispielcode:

```
public class MainActivity extends Activity {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(new MySurfaceView(this));//Screeninhalte setzen

        //hier erzeugen wir Objekte wie Views und Threads
    }
    protected void onDestroy() {
    // nützlich um einige Objekte beim Beenden zu beenden zB Datenbank
    }
    protected void onPause(){
    //was soll passieren wenn die Activity pausiert wird?
    }
    protected void onResume(){
    }
}
```

Activity

- Jede neue Activity muss im AndroidManifest.xml deklariert werden!

```
<manifest …>
 <application…>
   <activity
      android:name=".ExampleActivity">
      …
   </activity>
 </application>
</manifest>
```

SurfaceView

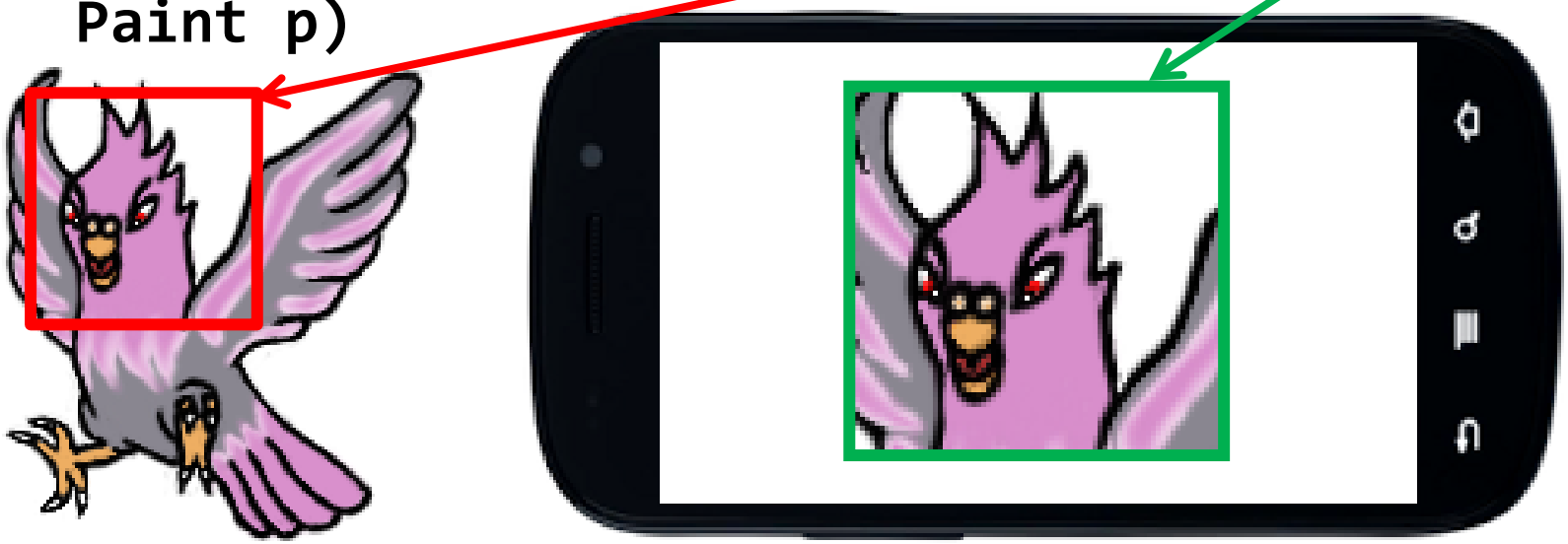
- Oberfläche zum Zeichnen
  - Bilder
  - Verschiedene Formen
  - Text
- Bietet auch Interaktion:
  - **`onTouchEvent`** fängt Touch-Eingaben ab (x und y Koordinate für Position am Screen)
  - **`onKey (View v, int keyCode, KeyEvent event)`** liefert Tastatureingaben ab (keyCode steht für eine bestimmte Taste)
- Sollte SurfaceHolder.Callback implementieren um drunter liegende Oberflächen zu realisieren und eigene Sichtbarkeit zu überprüfen

SurfaceView Canvas

- Figuren wie Rechtecke, Ovale, Linien,…
  - `drawRectangle(new Rect(int x, int y, int width, int height), Paint p);`
  - `fillRectangle(new Rect(int x, int y, int width, int height), Paint p);`
  - `drawLine(startX, startY, stopX, stopY, paint);`
- Klasse Paint dient dazu das Design festzulegen
  - Strichstärke `setStrokeWidth(int staerke)`
  - Strichart `setStyle( Style s)`
  - Farbe `setColor(Color c)`

Zeichnen von Bildern

- `drawBitmap(Bitmap bitmap, int left, int top, Paint p) //p kann auch null sein`
- `drawBitmap(bitmap, Rect src, Rect dst, Paint p)`



SurfaceView Touch und Key Events

- TouchEvents:
  - **`MotionEvent.`*`ACTION_DOWN`*** *:* auf den Touchscreen wird aufgedrückt
  - **`MotionEvent.`*`ACTION_UP`***: Finger wird vom Touchscreen abgehoben
  - **`MotionEvent.`*`ACTION_MOVE`***: Finger wird aufgedrückt und dabei bewegt
  - **`MotionEvent.`*`ACTION_SCROLL`***: Scrollbewegung

KeyEvents:

- **`KeyEvent.KEYCODE_BACK`**: Zurück Button
- **`KeyEvent.KEYCODE_MENU`**: Menü Button
- **`KeyEvent.KEYCODE_HOME`**: Home Button

SurfaceView Code1

```
public class MySurfaceView extends SurfaceView implements
   SurfaceHolder.Callback {
   private MainThread thread;
   public MainGamePanel(Context context) {
        super(context);
        getHolder().addCallback(this); // hinzufügen des
        Callbacks um Events abzuprüfen
        setFocusable(true); // um events handeln zu können
   }
    public void surfaceCreated(SurfaceHolder holder) {
   //Content laden
   thread = new MainThread(getHolder(), this); // gameloop
   erstellen
   thread.setRunning(true);
   thread.start();
   }
```

SurfaceView Code2

```
public void surfaceDestroyed(SurfaceHolder holder) { //Thread sauber
   beenden
…}
public boolean onTouchEvent(MotionEvent event) {
   if (event.getAction() == MotionEvent.ACTION_DOWN) {//Touch event
        //event.getX() oder event.getY() liefert Koordinaten
   }
   return super.onTouchEvent(event);
}
protected void onDraw(Canvas canvas) { //Es wird erst gezeichnet,
   wenn der Thread diese Methode aufruft!
   super.onDraw(canvas);//Hier wird gezeichnet, zum Beispiel:
   Paint p= new Paint();
   p.setColor(Color.BLUE);
   p.setStrokeWidth(10);
   canvas.drawRect(new Rect(20, 20, 60, 60), p);
}
```

SurfaceView Canvas – unser Zeichenblatt

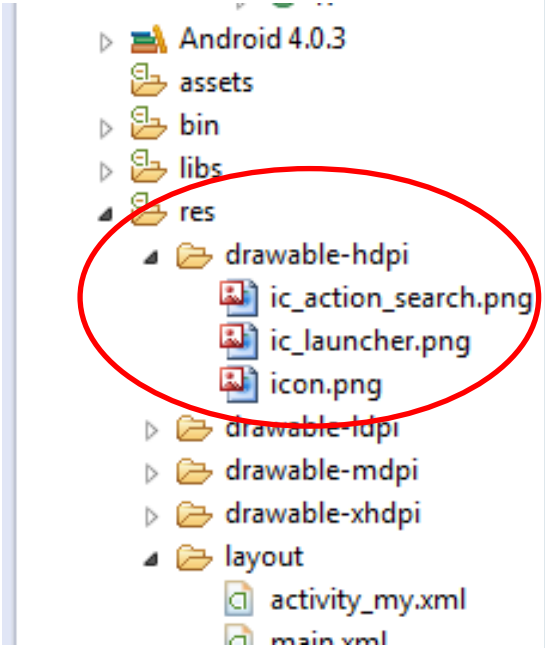
- Grafiken werden in derselben Reihenfolge wie im Code geschrieben übereinander gemalt
- Beispiel:

```
canvas.drawBitmap(hintergrund, 0, 0, null);
canvas.drawBitmap(kakadu, 150, 250, null);
canvas.drawBitmap(hase, 190, 250, null);
```

- Ladet alle Bilder EIN Mal beim Initialisieren der View (NICHT in der Schleife -> Ineffizient und verlangsamt das Spiel enorm!)

Bitmaps erstellen

- **`Bitmap bild= BitmapFactory.`*`decodeResource(`*** ***`getResources(),R.drawable.filename);`***
- Bild muss kein .bmp sein, .png oder .jpg gehen ebenso!
- Bilder werden im res Ordner abgespeichert und im automatisch generierten R.java gelistet für den Zugriff



Was ist ein Thread?

- „Handlungsstrang“, der Parallelisieren erlaubt
- Muss Klasse „Thread“ ableiten
- Methoden:
  - **`Start()`**: startet den Thread und ruft run() auf
  - **`Run()`**: muss überschrieben werden, beinhaltet die Logik des Threads
  - **`Sleep()`**: pausiert Thread für gewisse Anzahl an Millisekunden zB.: **`Sleep(5000);`**
  - **`Join()`**: beendet den Thread

Thread- Game Loop

- Verantwortlich für gleichmäßiges Update des Spiels und des Screens
- Spiel soll unabhängig von der Smartphone Leistung gleich schnell laufen
- Eine Schleife mit speziellen Eigenschaften
- Soll keine anderen Methoden blocken

```
public class GameLoopThread extends Thread {
   private SurfaceHolder surfaceHolder;
   private MySurfaceView myView;
   private boolean running; //läuft das Spiel gerade?
   public void setRunning(boolean running){
        this.running = running;
   }
```

Thread – Game Loop

```
public GameLoopThread(SurfaceHolder surfaceHolder, MySurfaceView v) {…}
   public void run() {
        //frameabhängigkeit hinzufügen
        while (running) {
        Canvas c=null; //neuzeichnen
        try{
                c = this.surfaceHolder.lockCanvas(null);
                synchronized (this.surfaceHolder){
                v.onDraw(c);
                updateGame(); //Spielinhalte updaten
        }
        finally{
                if (c != null){
                this.surfaceHolder.unlockCanvasAndPost(c);}
        }
}
```

Collision detection

- Leider nicht im API enthalten
- Muss manuell programmiert werden
- Kleine Hilfen wie **`intersect()`** Abfrage von Rectangle:
  - Bitmaps wie Rechtecke behandeln
  - **`rechteck1.intersect(rechteck2)`** liefert zurück, ob sich die Rechtecke überschneiden

Beispiel:

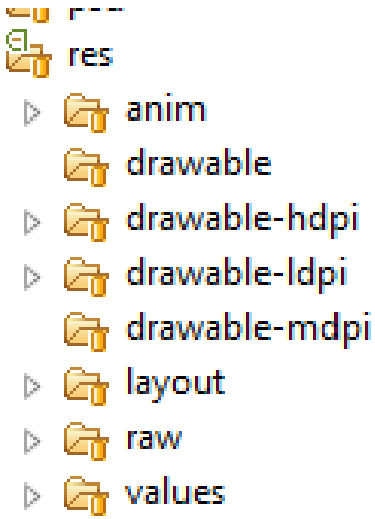
```
Rect rect1= new Rect(bild1x, bild1y,
      bild1x+bild1.getWidth(),bild1y+bild1.getHeight());
Rect rect2= new Rect(bild2x, bild2y,
      bild2x+bild2.getWidth(),bild2y+bild2.getHeight());
boolean collision= rect1.intersect(rect2);
```

Weitere Informationen

- Activities
  - http://developer.android.com/guide/components/activities.html
  - http://developer.android.com/training/basics/firstapp/starting-activity.html
- SurfaceView
  - http://android-coding.blogspot.co.at/2011/05/drawing-on-surfaceview.html
  - http://developer.android.com/reference/android/view/SurfaceView.html
- Touch & Key Events
  - http://developer.android.com/reference/android/view/KeyEvent.html
- Game Loops
  - http://obviam.net/index.php/the-android-game-loop/
  - http://www.javacodegeeks.com/2011/07/android-game-development-basic-game_05.html

Ressourcen

- Effizienter und einfacher Zugriff auf Ressourcen
- „res“ Ordner liefert Unterordner
  - Drawable (weitere Ordner für angepasste Auflösung)
  - Anim (Animationsfiles in xml)
  - Raw (Musikfiles, Videos)
  - Layout (bestimmte Layouts definieren für den Screen)
  - Value (Strings.xml um Texte zu definieren)



R.java

- Automatisch generiert um alle Ressourcen zu verwalten
- Im Ordner „gen“- NICHT MANUELL BEARBEITEN!!!

MainThread.java | DroidzActivi... | DaemonFight.... | strings.xml

```
1  /* AUTO-GENERATED FILE.  DO NOT MODIFY.
7
8  package demon.hunter;
9
10 public final class R {
11     public static final class anim {
12         public static final int hcup=0x7f040000;
13     }
14     public static final class attr {
15     }
16     public static final class drawable {
17         public static final int attackmenue=0x7f020000;
18         public static final int backgroundgreen=0x7f020001;
19         public static final int backgroundwueste=0x7f020002;
20         public static final int barrel=0x7f020003;
21         public static final int basket=0x7f020004;
22         public static final int button=0x7f020005;
23         public static final int button2=0x7f020006;
24         public static final int cockarillokl=0x7f020007;
25         public static final int cockarilloklback=0x7f020008;
26         public static final int daemonmenuebg=0x7f020009;
27         public static final int einhorn=0x7f02000a;
28         public static final int einhornkl=0x7f02000b;
29         public static final int farmer=0x7f02000c;
30         public static final int farmer2=0x7f02000d;
31         public static final int farmer2back=0x7f02000e;
32         public static final int farmer2front=0x7f02000f;
33         public static final int farmer2right=0x7f020010;
34         public static final int farmer3=0x7f020011;
35         public static final int farmer3front=0x7f020012;
36         public static final int farmer3right=0x7f020013;
37         public static final int farmerback=0x7f020014;
```

Debugging mit LogCat

- Nicht System.out.print() benutzen
- Android hat seine eigene Message Ausgabe
- Mehrere Messagearten:
  - `Log.d(String titel, String message)` Debug Message
  - `Log.e(String titel, String message)` Error Message
  - ….

Durch Filter kann man einzelne Messagebereiche ausblenden zum Beispiel alle Error Messages sollen nicht angezeigt werden

LogCat

ers | Data Source Explorer | Snippets | Console | LogCat | Call Hierarchy

Search for messages. Accepts Java regexes. Prefix with pid:, app:, tag: or text: to limit scope. debug

L...	Time	PID	TID	Application	Tag	Text
D	09-16 22:54:0...	542	559	net.obviam.droidz	drin	bin drinnen
D	09-16 22:54:0...	542	559	net.obviam.droidz	test	ich zeichne
D	09-16 22:54:0...	542	559	net.obviam.droidz	drin	bin drinnen
D	09-16 22:54:0...	542	559	net.obviam.droidz	test	ich zeichne
D	09-16 22:54:06.028	542	559	net.obviam.droidz	drin	bin drinnen
D	09-16 22:54:0...	542	559	net.obviam.droidz	test	ich zeichne
D	09-16 22:54:0...	542	559	net.obviam.droidz	drin	bin drinnen
D	09-16 22:54:0...	542	559	net.obviam.droidz	test	ich zeichne
D	09-16 22:54:0...	542	542	net.obviam.droidz	MainGamePanel	Surface is being destroyed
D	09-16 22:54:0...	542	559	net.obviam.droidz	drin	bin drinnen
W	09-16 22:54:0...	542	559	net.obviam.droidz	dalvikvm	threadid=11: thread exiting with uncaught exception (group=0x f8)
E	09-16 22:54:0...	542	559	net.obviam.droidz	AndroidRun...	FATAL EXCEPTION: Thread-72
E	09-16 22:54:0...	542	559	net.obviam.droidz	AndroidRun...	java.lang.NullPointerException
E	09-16 22:54:0...	542	559	net.obviam.droidz	AndroidRun...	at net.obviam.droidz.MainGamePanel.onDraw(MainGamePanel.java:
E	09-16 22:54:0...	542	559	net.obviam.droidz	AndroidRun...	at net.obviam.droidz.MainThread.run(MainThread.java:54)

Tipps & Tricks

- Wer ein Android Phone hat -> besser zum Testen als der Emulator
- Habt keine Angst davor mehrere Activities zu nutzen
- Fehlermeldungen genau durchlesen
- Weniger ist mehr: auch ein einfaches Spiel macht viel Programmieraufwand (KEINE RPGs)
- Erzeugt Bitmaps nur einmal beim Initialisieren
- Don‘t panic: Im Tuwel Forum wird euch geholfen

Danke für die Aufmerksamkeit

?