Programmiererfahrungen mit Android

Arthur Thiessen & Senad Licina

Wissenschaftliches Rechnen
Fachbereich Informaitk, Uni Hamburg

24.03.2010

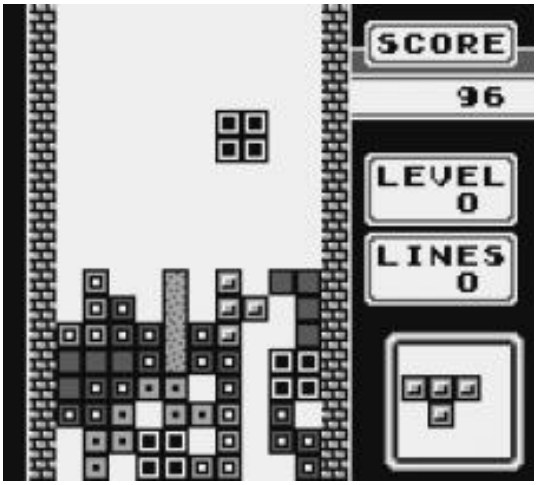
Tetroid

Spielgeschichte
Tales of a Lengendary Videogame

- Juni 1984 - Alexei Leonidowitsch Paschitnow erfindet Tetris
- Pentomino dient als Vorlage
- Lizenzproblematik
- Nintendo Gameboy

Spielkonzept
Tetris for beginners

- 4 Blöcke = 1 Stein
- Steine fallen von Oben
- volle Zeilen werden gelöscht
- Punktesystem
- Levelsystem
- Game Over!



Steine

- Datentyp
  - Array of Array of int !
- Operationen
  - Positionsveränderung
  - Rotation

```
{
  { 0, 0, 0, 0, 0 },
  { 0, 1, 1, 0, 0 },
  { 0, 0, 1, 1, 0 },
  { 0, 0, 0, 0, 0 },
  { 0, 0, 0, 0, 0 }
}
```


Spielbrett

- Datentyp
  - Array of Array of int !
- Operationen
  - Stein ins Brett legen
  - Zeilen Löschen
  - Kollisionsabfrage
  - Game Over!

```
{
 {1,0,0,0,0,0,0,0,0,0,0,1},
 {1,0,0,0,0,0,0,0,0,0,0,1},
 {1,0,0,0,0,0,0,0,0,0,0,1},
 {1,0,0,0,0,0,0,0,0,0,0,1},
 {1,0,0,0,0,0,0,0,0,0,0,1},
 {1,0,0,0,0,0,0,0,0,0,0,1},
 {1,0,0,0,0,0,0,0,0,0,0,1},
 {1,0,0,0,0,0,0,0,0,0,0,1},
 {1,0,0,0,0,0,0,0,0,0,0,1},
 {1,0,0,0,0,0,0,0,0,0,0,1},
 {1,0,0,0,0,0,0,0,0,0,0,1},
 {1,0,0,0,0,0,0,0,0,0,0,1},
 {1,0,0,0,0,0,0,0,0,0,0,1},
 {1,1,1,1,1,1,1,1,1,1,1,1}
}
```

Spieldynamik und Grafik

- Thread & Schleife vs. Timer
  - runOnUiThread();
- Views anpassen
  - (TextView)findViewById(R.id.MyTextView);
  - gewünschte funktionen aurfufen
  - fertig!
- GridLayout & ImageView
  - Elemente lassen sich immer seperat auswählen
  - Fehlschlag! :(
- FrameLayout & RectViews
  - RectView extrends View
  - beinhaltet ein ShapeDrawable
  - performance

Übergabe von Daten zwischen Activities

- Daten werden über das Intent an eine andere Activity übergeben
  - mit der ,,putExtra"-Methode werden Daten in dem Intent gelagert
  - mit der ,,getExtras"-Methoden werden Daten vom Intent ausgelesen
- in Intents können nur Bestimmte Datentypen übergeben werden
- java.io.Serializable vs. android.os.Parcelable

Übergabe von Daten zwischen Activities

„alte" Activity

```
Intent meinIntent = new Intent(this, NeueActivity.class);
meinIntent.putExtra("meinString", "have you tried to turn
                                   it OFF and ON again?");
meinIntent.putExtra("meinInteger", 42);
startActivity(meinIntent);
```

„neue" Activity

```
Bundle meinBundle = getIntent().getExtras();
String meinString = (String)bundle.get("meinString");
int meinInteger = (int)bundle.get("meinInteger");
```


Die Idee

- Zwei Spieler sollen gegeneinander spielen können
- jeder von seinem Gerät aus
- Sichtbar auf dem Display
  - das eigene Spielfeld
  - das Spielfeld vom Gegner
  - nächster Stein
  - Punktzahl

Anforderungen

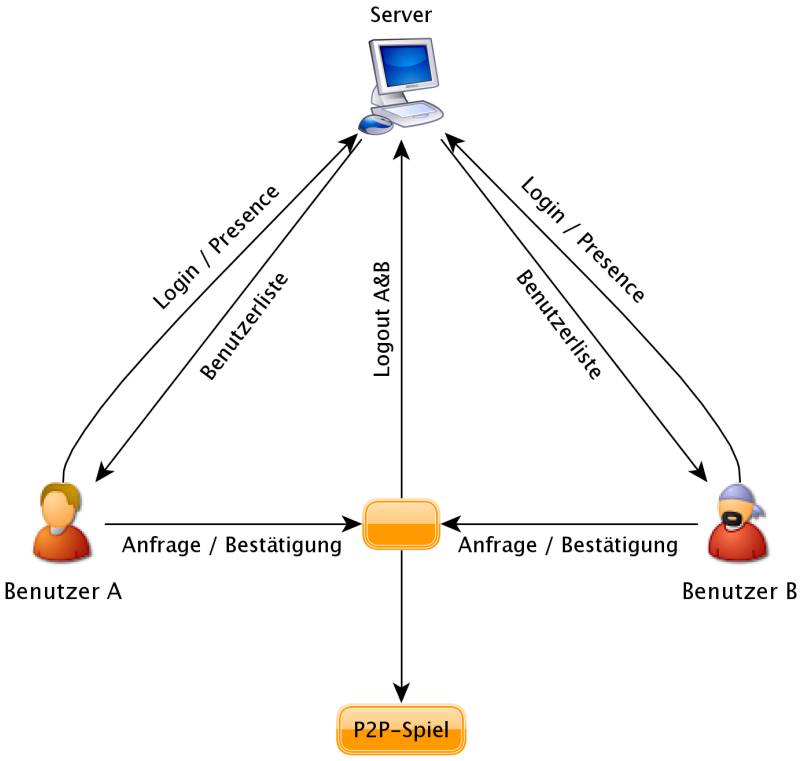
Server

- Möglichst geringe Bedeutung (zwecks P2P)
- Übernimmt die Rolle des Vermittlers von Spielern

Client

- Möglichst Peer2Peer
- Bluetooth/Internet
- Hohe Performance
- Bedienbarkeit (Tasten/Sensoren)
- Minimaler Traffic

Überblick



Tetroid Server
reaktiver Akteur

- Einfacher TCP Socket Listener
- Empfängt Nachrichten vom Client
  - Login
  - Logout
  - UserList
  - Request
- Versendet Antworten an Client

Tetroid Client
aktiver Aktuer

- Spielmechanik
- P2P Kommunikation
- Sensorsteuerung

Tetroid Client

Bluetooth:

Pro

- kostenlos
- überall verfügbar
- weit verbreitet
- gute Dokumentation

Contra

- erst ab Android 2.0
- somit nur auf modernsten Handys testbar
- inoffizielle Bibliotheken nicht zu gebrauchen
- kleine Reichweite

Internet:

Pro

- große Reichweite
- gewohnte Socket Programmierung
- wird ohnehin für Kommunikation mit Server verwendet

Contra

- Kostenfaktor
- Empfang

Bluetooth

Seit API Level 5 (Android 2.0) stehen uns 8 Klassen zur Verfügung:

BluetoothAdapter: repräsentiert das lokale BT Gerät
BluetoothDevice: repräsentiert ein externes BT-Gerät
BluetoothServerSocket: hört auf eingehende Verbindungen
BluetoothSocket: initiiert eine Verbindung

BluetoothClass: grundsätzliche Charakteristika von BT-Geräten

Bluetooth

Um Bluetooth nutzen zu können, muss man die Berechtigungen in der Android-Manifest XML anpassen:

permission.BLUETOOTH

```
<manifest ...>
<uses-permission android:name=\android.permission.BLUETOOTH\ />
</manifest>
```

Möchte man einen DeviceScan durchführen, so braucht man ADMIN:

permission.BLUETOOTH_ADMIN

```
<manifest ...>
<uses-permission android:name=\android.permission.BLUETOOTH_ADMIN\ />
</manifest>
```


Bluetooth

Beispiel Programm zum Suchen & Anzeigen von BT-Geräten

```
BluetoothAdapter mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
if (!mBluetoothAdapter.isEnabled()) {
    Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
    startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);
}

private final BroadcastReceiver mReceiver = new BroadcastReceiver() {
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();
        if (BluetoothDevice.ACTION_FOUND.equals(action)) {
            BluetoothDevice device = intent.getParcelableExtra(
                                                BluetoothDevice.EXTRA_DEVICE);
            mArrayAdapter.add(device.getName() + "\n" + device.getAddress());
        }
    }
};
IntentFilter filter = new IntentFilter(BluetoothDevice.ACTION_FOUND);
registerReceiver(mReceiver, filter);
```

Bluetooth
ein Fazit

Vorteile

- Wider Erwartungen (theoretisch) sehr leicht zu handhaben
- Top Dokumentation & Guides

Nachteile

- Emulator unterstützt (noch) kein Bluetooth
- Leider sind die genannten Funktionalitäten erst ab Android 2.0 so verfügbar.
- Möchte man dennoch Bluetooth vor Version 2.0 nutzen, so muss man ggf. die inoffizielle Bluetooth API verwenden (http://code.google.com/p/android-bluetooth/)

Übersicht

Es gibt 2 Klassen, mit deren Hilfe man auf die Android Sensoren bezug nehmen kann:

- SensorManager
  - ist ein SystemService und verwaltet sämtliche dem Android bekannte Sensoren
- SensorEventListener
  - sollte von einer Klasse implementiert werden, die bestimmte Sensorenereignisse verarbeiten soll

Übersicht

Tetroid bietet eine Sensorunterstützung zur Spielsteuerung an.

Es handelt sich um Beschleunigungssensoren entlang der X, Y und der Z Achse.

Codebeispiel

Codebeispiel

```
Public class TetroidSensor implements SensorEventListener{

  Public TetroidSensor(Activity a){
    SensorManager sm = (SensorManager)a.getSystemService(
                                          Activity.SENSOR_SERVICE);
    sm.registerListener(sm.getSensorList(Sensor.TYPE_ACCELEROMETER));
  }

  Public synchronized onSensorChanged(SensorEvent  event){
    // verarbeite event
    // wichtig! Verzögerung einbauen!
  }
}
```

Fazit

Vorteile

- funktioniert gut und präzise
- mit wenig Code erreicht man schnelle Ergebnisse

Nachteile

- wenig Dokumentation
- erfordert grundlegendes Verständnis von Listenern und Events

HH-Plan

Motivation

- Fahrplanauskunft für die öffentlichen Verkehrsmittel Hamburgs über das Android
- Warum nicht einfach mit geofox?
  - weniger Aufwand für den Benutzer
  - weniger Traffic (spart ggf. Kosten)
  - kürzere Wartezeit als über den Browser
- Android stellt eine gute Plattform dar

Anforderungen an das App

- Fahrplanauskunft
- Bring me Home
- GPS-Koodinaten auswertung

Zugriff auf Services

- um Zugriff auf bestimmte Services zu erhalten (z.B. Internet, GPS), braucht man ersteinmal eine Berechtigung dazu
- dem Benutzer werden die Berechtigungen bei der Installation von Apps angezeigt
- Berechtigungen werden im Android Manifest festgelegt

Internet-Permission

```
<uses-permission android:name="android.permission.INTERNET"/>
```


Verwendete Klassen

- android.location.Location
  - repräsentiert eine geographische Position
  - es werden u.a. Breiten- & Längengrad, Uhrzeit, Geschwindigkeit und Genauigkeit gespeichert
- android.location.LocationManager
  - ist ein Systemservice, auf den man zugreifen kann
  - bietet eine Vielzahl von Methoden
- android.location.LocationListener
  - wird beim LocationManager angemeldet
  - ruft onLocationChanged auf, wenn sich die Geoposition verändert
- android.location.Geocoder
  - eine Klasse, die zum ,,geocoding" benutzt wird
  - als geocoding bezeichnet man den Prozess eine Adresse in eine geographische Position umzuwandeln.
  - reverse geocoding

android.location.Location

Initialisierung

```
context = >Aktuelle Activity<
lm = (LocationManager)context.getSystemService(Context.LOCATION_SERVICE);
latitude = lm.getLastKnownLocation(LocationManager.GPS_PROVIDER).getLatitude();
longitude = lm.getLastKnownLocation(LocationManager.GPS_PROVIDER).getLongitude();
lm.requestLocationUpdates(LocationManager.GPS_PROVIDER,5000,0,>LocationListener<);
```

LocationListener

```
public void onLocationChanged(Location location) {
  latitude = location.getLatitude();
  longitude = location.getLongitude();
}
```

Location → AddressList → Address → Straße

```
gc = new Geocoder(_context);
addresslist = gc.getFromLocation(latitude, longitude, >maxResults<);
String addressNearToLocation = addresslist.get(0).getAddressLine(0);
```


Dialogs

- Dialogs bieten eine Art GUI, welche auf die Activity ,,gelegt" werden kann
- können aufgerufen werden, ohne die aktuelle Activity zu beenden
- eigenen sich z.B. um:
  - Fehlermeldungen/Warnungen/Nachrichten auszugeben
  - Ladebalken anzuzeigen
  - Anfragen an den Benutzer zu stellen
  - Auswahlmöglichkeiten auszugeben
  - alles was du programmieren kannst!

Wie Dialogs erzeugt werden

Aufruf aus der Activity

```
static final int MEIN_DIALOG_ID = 0;

protected Dialog onCreateDialog(int id) {
  Dialog dialog;
  switch(id) {
  case MEIN_DIALOG_ID:
    dialog = new Dialog(getApplicationContext());
    dialog.setContentView(R.layout.MeinDialogLayout);
    dialog.setTitle("Mein Dialog");
    // passe die Anzeige auf meinem Dialog an
    break;
  default:
    dialog = null;
  }
  return dialog;
}

public void zeigeMeinenDialog() {
  showDialog(MEIN_DIALOG_ID);
}
```


Wie Dialogs erzeugt werden

Erzeugung eines PickTimeDialogs

```
int minutes = _datetime.getMinutes();
int hours = _datetime.getHours();
result = new TimePickerDialog(>context<, >OnTimeSetListener<, hours, minutes, true);
```

OnTimeSetListener

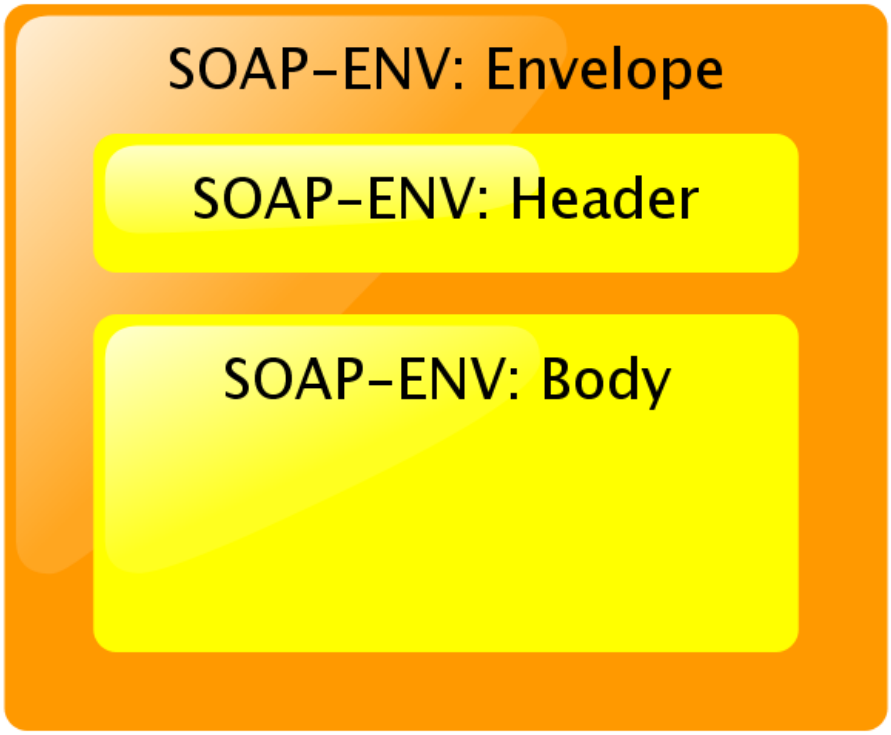
```
public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
  >context<.setHours(hourOfDay);
  >context<.setMinutes(minute);
  >context<.updateTime();
}
```

Einsatz

- Überall dort, wo der direkte Zugriff auf Datenbanken wenig sinnvoll ist
- Eliminiert Kompatibilitätsprobleme
- Sicherheitsaspekte

So kann der (partielle) Zugriff auf eine Datenbank ermöglicht werden, ohne dass dem Anwenderprogramm der direkte Zugang gestattet werden muss. Über die SOAP-Schnittstelle kann die Menge der ausführbaren Methoden reglementiert und definiert werden.

Struktur



Anfrage

SOAP-Anfrage

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body>
<m:GetPrice xmlns:m="http://www.w3schools.com/prices">
<m:Item>Apples</m:Item>
   </m:GetPrice>
</soap:Body>

</soap:Envelope>
```


Antwort

SOAP-Antwort

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body>
<m:GetPriceResponse xmlns:m="http://www.w3schools.com/prices">
   <m:Price>1.90</m:Price>
</m:GetPriceResponse>
</soap:Body>

</soap:Envelope>
```

Fazit

Vorteile

- Universelles Nachrichtenformat
- Sicherheit
- Kontrolle über Daten
- Einfache Handhabung

Nachteile

- Viel Overhead durch XML
- Zusätzlicher Programmieraufwand durch XML Parser

eine subjektive Meinung

Zusammenfassung

Was wir gelernt haben:

- eine Technische Realisierungsmöglichkeit von Tetris kennengelernt
- wie man die Sensoren anspricht und was man zu beachten hat
- wie man auf den GPS-Service zugreift, was man dabei beachten muss und wie man diese Daten verwerten kann
- was Dialogs sind und wie / wozu man diese einsetzt
- Grundlagen über SOAP-Services

Fragen?

Quellenangabe

- http://developer.android.com