



HIMMELSKÖRPER KOMPASS

Modul ÜK335

Mike Krenn

Inhaltsverzeichnis

Aufgabenstellung	2
Bewertung	2
Ziel	2
Mock-ups	3
Der Hauptbildschirm:	3
Settings Bildschirm	4
About Bildschirm	5
Projektidee	6
Erste Schritte	6
Tests	8
Reflexion	10
Quellen	10

Aufgabenstellung

In einer Einzel- oder Gruppenarbeit entwerfen, planen und realisieren die Lernenden eine native Smartphone App, welche mindestens aus drei Screens besteht.

In einem ersten Schritt erarbeiten die Lernenden am Computer eine Vorlage (Mock-up) für ihre App. Als Kriterien gelten die Best Practices Vorgaben für Android oder iOS, sowie ergonomische Standards (z.B. EN-9241-110). Hier sollen auch die verschiedenen Bildschirm Größen, Ausrichtungen und Gerätetypen berücksichtigt werden. Der Mock-up muss bezüglich Machbarkeit und Komplexität vom Dozenten abgenommen werden. Danach programmieren die Lernenden ihre App basierend auf den abgenommenen Mock-ups. Dabei müssen die Lernenden beweisen, dass sie die benötigten Kompetenzen zur Programmierung einer nativen App erlangt haben. Dazu wird einerseits die Qualität des Programmcodes, andererseits die Projektdokumentation, welche während der Realisierung geführt wird, bewertet.

Die Projektdokumentation muss mindestens die erarbeiteten Mock-ups, einen Teils zur technischen Realisierung, so wie einen Teils zum Testing (Testplan, Testergebnisse) enthalten.

Bewertung

- Können die Lernenden genügend genau erklären wie die Ressource funktioniert und wie man sie einbindet? 20 - 40%
- Entspricht die Verwendung der Ressource den gängigen Standards? 10 -20%
- Werden Fehler korrekt behandelt? z.B. gibt es ein Errorhandling auf Programmebene und werden allenfalls dem Benutzer sinnvolle Rückmeldungen gegeben? 10 - 20%
- Werden die Daten der Ressource effizient verarbeitet? z.B. werden die Daten in sinnvollen Zuständen (LifeCycle) gespeichert? 10 - 30%
- Wie verhält sich die Komponente bezüglich Laufzeitverhalten der APP und Datenkonsistenz? z.B. werden wichtige Daten in einer Datenbank oder Datei gespeichert? 5 - 20%
- Ist der Code selbsterklärend und wurde er falls nötig an den entsprechenden Stellen kommentiert? 15 - 30%

Ziel

Das Ziel meiner Applikation ist es die Position eines Himmelskörpers anzugeben, es soll jeweils in die Richtung des Himmelskörpers zeigen. Um dies zu erfüllen brauche ich Folgende Dinge:

- GPS, es wird die Position des Users benötigt, Latitude, Longitude.
- Die Position der Himmelskörper.
- Die Geräte Rotation und Neigung.

Mock-ups

Der Hauptbildschirm:

Per Dropdown können gewisse Himmelskörper ausgewählt werden.

Oben rechts kommt man zu den Einstellungen.

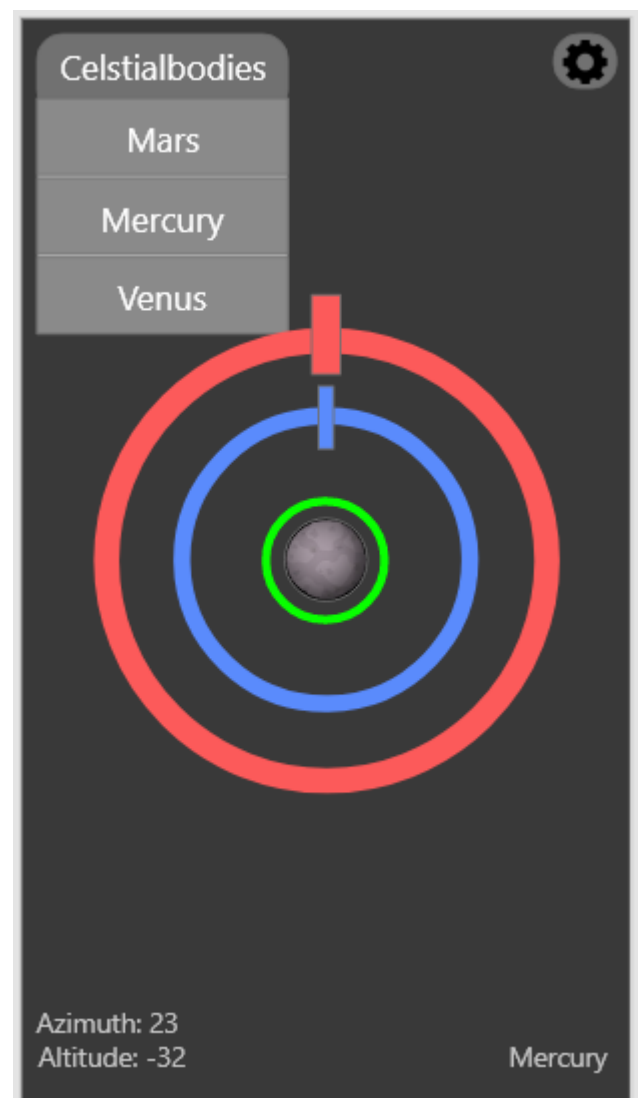
Der rote Kreis dreht sich jeweils zu dem ausgewählten Himmelskörper und zeigt auf diesen, egal wie die Rotation des Gerätes ist.

Das selbe Prinzip passiert auch mit dem grünen Kreis nur bewegt sich dieser nach unten und nach oben dies wird mit der Neigung des Gerätes dargestellt ist dieser Grüne Kreis in der Mitte des Himmelskörpers hat man die richtige Neigung.

Unten Links werden jeweils die Azimuth und Altitude Winkel angegeben, diese zeigen wo der ausgewählte Himmelskörper sich befindet.

Azimuth auch Z-Achsen Rotation, Yaw Rotation.
Altitude auch Y-Achsen Rotation, Pitch Rotation.

Unten Rechts wird der angegebene Himmelskörper angezeigt, sowie in der Mitte ist ein Bild das den Himmelskörper illustriert.



Settings Bildschirm

Open Links geht man zurück zum Hauptfenster.

Oben Rechts ist der About screen.

Ich bin mir nicht sicher was für Settings oder ob es überhaupt welche Settings geben wird. Da es eine sehr einfache App ist.

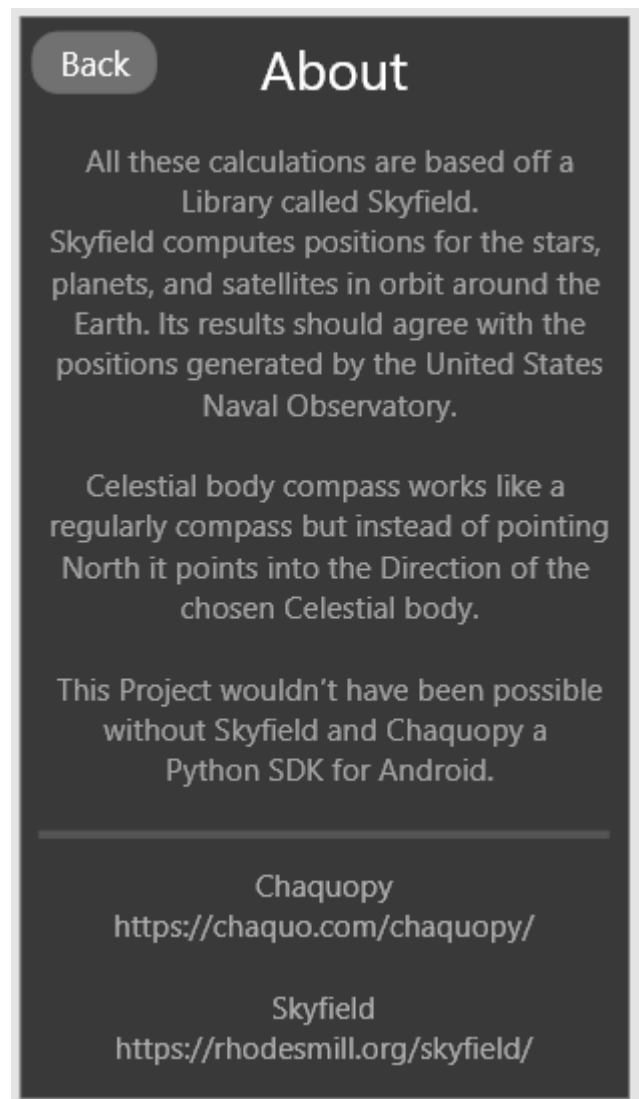
Unten ist ein Meldekasten Users können, Probleme melden die mir dann Zugesendet werden per E-Mail.



About Bildschirm

Open Links geht man zurück zu den Settings.

Der About screen ist ein kurzer Text, über die App und welche Libraries benutzt wurden.



Projektidee

Ich habe mich dazu entschieden einen Himmelskörper Kompass zu machen, der im Grundprinzip wie ein Kompass funktioniert allerdings zeigt er nicht nach Norden sondern, zudem ausgewählten Himmelskörper. Durch das GPS wird die Position herausgefunden, welche einen Bestandteil der genauen Ausrechnung liefert.

Die Himmelskörper Positionen werden anhand einer Python Bibliothek berechnet. Ich habe zuerst versucht eine Java Library zu finden, aber ohne Erfolg ebenfalls habe ich versucht die Berechnungen selbst zu erstellen, aber auch das scheiterte. Dadurch das ich eine Python Library und Python Files in meiner Android App benutze brauche ich eine SDK, die hilft Android mit Python zu verbinden. Dazu verwende ich Chaquopy.

Mit den zwei Positionen kann man nun mit zwei Sensoren die Ausrichtung des Gerätes ermitteln, dadurch wird ein Kreis immer auf die Richtung zeigen in dem sich der Himmelskörper befindet, egal wie die Ausrichtung des Gerätes ist.

Weil der Kompass die Hauptfunktion meiner App ist wusste ich nicht was ich für zwei weitere Screens ich erstellen soll, deshalb sehen und fühlen sich die beiden anderen Screens mager an.

Erste Schritte

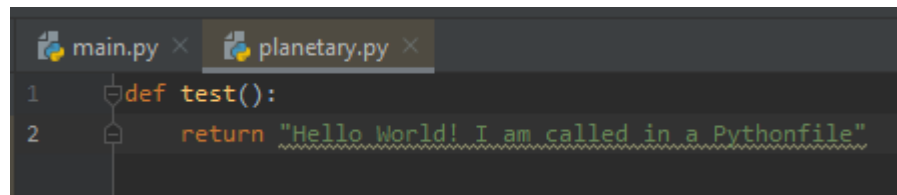
Die ersten Schritte die ich gemacht habe war das Herausfinden der Library oder API die mir Planetare Positionen ermitteln kann, dies war aber wesentlich schwieriger als ich gedacht habe, wie oben schon erklärt musste ich eine Python Library benutzen, dazu brauchte ich eine Python SDK.

Hier wird die Python SDK geladen und eine Klasse wird definiert, nun wird eine Methode aufgerufen die dann, einen Wert zurück liefert und in dem obj speichert.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    TextView textView = findViewById(R.id.myTextView);

    if (! Python.isStarted()) {
        Python.start(new AndroidPlatform( context: this));
    }
    Python py = Python.getInstance();
    PyObject pyf = py.getModule( s: "planetary"); //Python file name
    PyObject obj = pyf.callAttr( key: "test"); //methode name
    textView.setText(obj.toString());
}
```

Hier ist die Python
File und die Methode.



```
main.py x planetary.py x
1 def test():
2     return "Hello World! I am called in a Pythonfile"
```

Die angegebene Methode wird nun aufgerufen
und returnt einen einfach String.
Dieser wird in Java dann in einen TextView
gespeichert.



Tests

Die ersten Tests die ich durchgeführt habe, war das überprüfen der Werte, da die Library Daten von *United States Naval Observatory* Stammen war ich mir sicher das diese Daten Korrekt sind Dennoch habe Ich jeden einzelnen Himmelskörper ausprobiert, ich habe die Tests selber geschrieben und nicht automatisiert, weil sich diese Daten jede Minute oder öfters ändern.

Die Daten zur Überprüfung stammen von: <https://theskylive.com/planetarium>

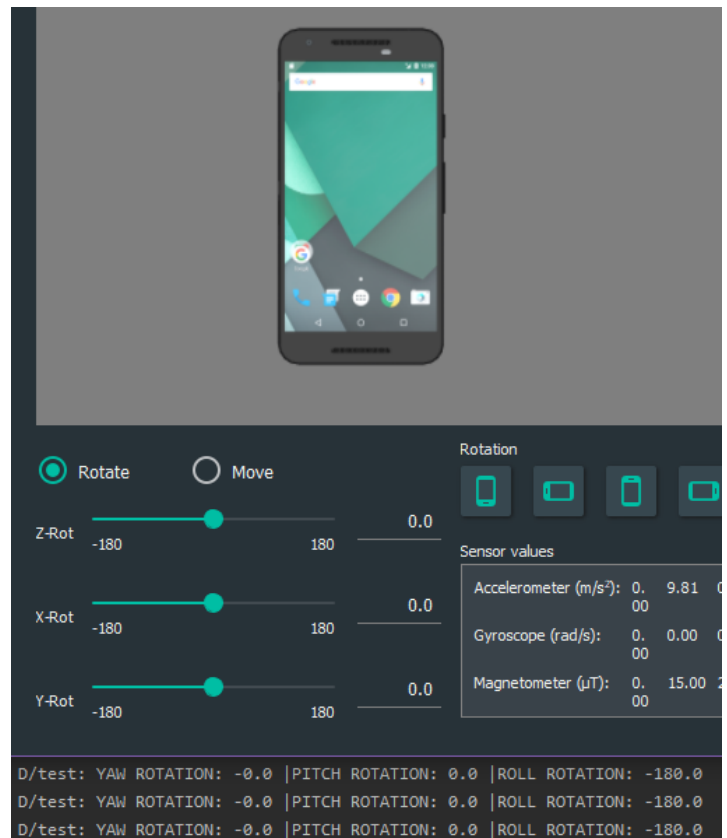
	Webseite Altitude	Webseite Azimuth	App Altitude	App Azimuth
Sun	28,7	92,7	28	92
Moon	26,5	188,6	26	189
Mercury	22,8	80,3	23	81
Mars	29	191	29	192
Jupiter	10,7	222,2	10	223
Saturn	13,6	218	12	220
Uranus	36,5	111,4	37	112
Neptune	36,8	172,7	37	173
Pluto	7	225,2	7	225

Da ich keine Minute und Sekunde im grad angebe sind meine Daten immer aufgerundet. Durch diese Test bin ich mir sicher das diese Daten präzise sind die Library die ich benutze und die Daten für die Himmelskörper haben ein „Ablaufs Datum“ Die Daten werden nur bis ins Jahr 2050 genau angezeigt.

Da es keinen Sensor gibt der die Momentane Rotation des Gerätes feststellt muss man anhand von 2 Sensoren: *Accelerometer und Magnetic_field* Sensor die Rotation berechnen, dieses Resultat sehen wir Unten in der Konsole. Im oberen Bereich ist die Emulation des Gerätes.

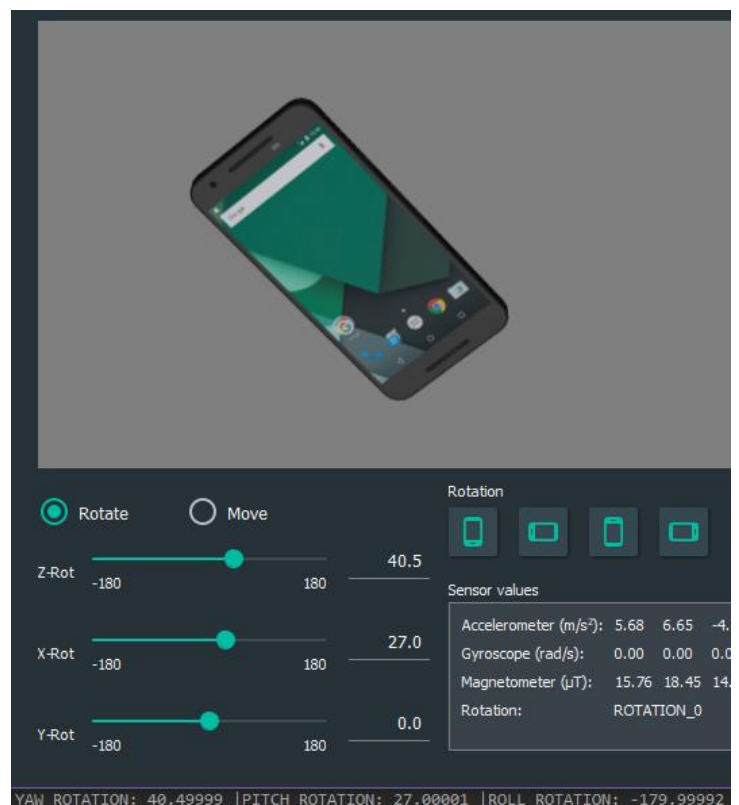
Die YAW Rotation ist die Z-Rotation
Die Pitch Rotation ist die X-Rotation
Die Roll Rotation ist die Y-Rotation

Roll ist -180 weil, das Gerät auf dem Rücken liegt.



Die Daten stimmen überein wenn man das Gerät dreht.

Somit habe ich eine Funktionierende Erkennung der Rotation des Gerätes.



Reflexion

Der Anfang dieses Modules war für mich am schwersten. Eine Idee für eine App zu finden war nicht einfach, dennoch habe ich am ersten Tag bereits eine App Idee gehabt.

Am zweiten Tag versuchte ich eine Library oder eine API zu finden die das liefert was ich benötige oder auch selbst an die Daten zu kommen mit Mathematik, aber all dies scheiterte und der zweite Tag war für nichts. Am nächsten Tag gab ich mir maximal noch 1 Stunde Zeit um etwas passendes zu finden, wenn nicht dann überlege ich mir ein neues Projekt.

Zum Glück fand ich was, mit der Hilfe von Skyfield eine Python Library.

Nachdem alles funktioniert hatte und ich die Daten die ich brauche bekam, war ich voller Elan ich war froh das ich dieses Projekt umsetzen konnte. Die nächsten paar Tage gingen schnell vorbei und ich war stets voller Konzentration. Nachdem ich das Hauptziel erreicht habe ging es an die weiteren Screens und dies fiel mir schwer, weil ich nicht wusste was genau ich für weitere Screens machen kann für meine App, deshalb sehen diese auch nicht wirklich interessant aus. Das Design von meiner App ist auch nicht wirklich allzu schön, aber es ist Okay, es ist relative schwierig ein schönes UI mit Android Studio zu erstellen.

Im Ganzen war dieses Projekt sehr interessant und ich habe einiges über ein Handy gelernt dessen Sensoren und wie das ganze aufgebaut ist (Layers). Was ich schade fand ist das ich kein Android hatte und ich nur mit dem Emulator testen konnte.

Quellen

Skyfield Documentation <https://rhodesmill.org/skyfield/>

Android Python SDK Chaquopy <https://chaquo.com/chaquopy/>

Himmelskörper Berechnungen <https://aa.quae.nl/en/reken/hemelpositie.html>

Android Dokumentation <https://developer.android.com/docs>

TheSkylive testen, überprüfen <https://theskylive.com/>

Stack Overflow Yaw und pitch Rotation <https://stackoverflow.com/questions/35303111/get-euler-yaw-angle-of-an-android-device>