Milestone II - GeoRacer - Group 3

Christos Ioannidis, Ludwig Leuschner, Pio Chibuzor Okongwu

Otto-Friedrich University of Bamberg

17th July 2019

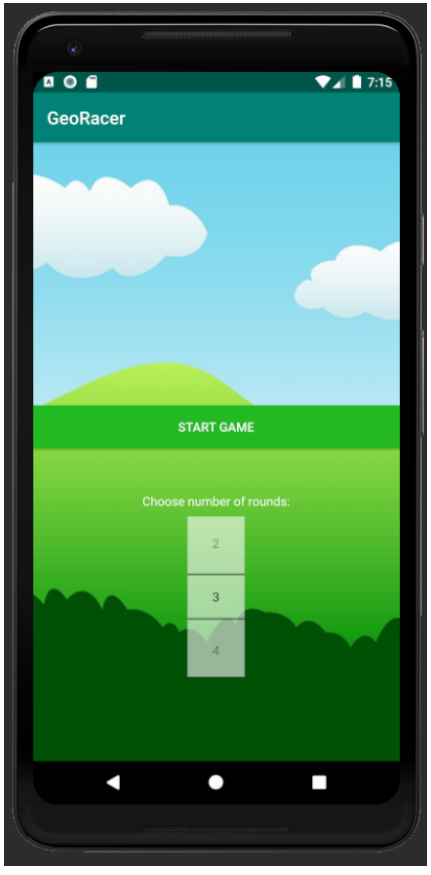
Application

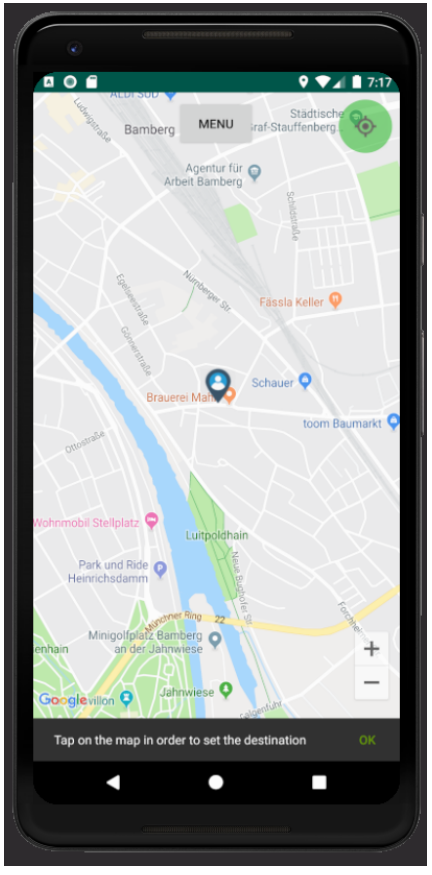
Class Diagram

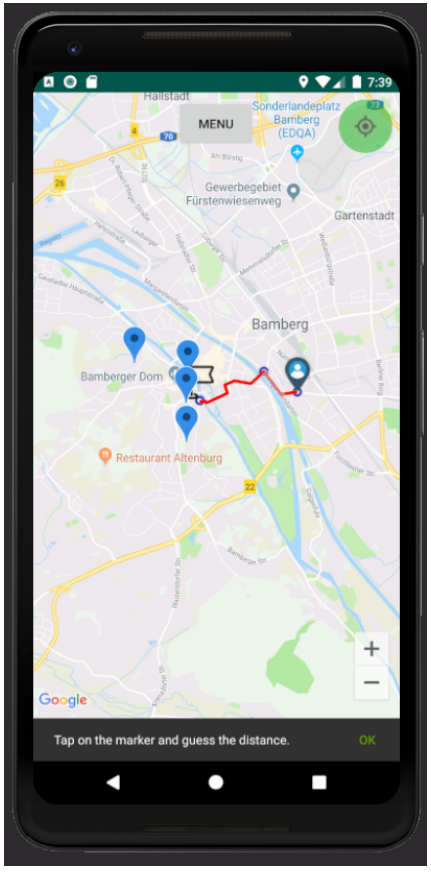
Lateration and Problematic Cases

Conclusion & Future Work

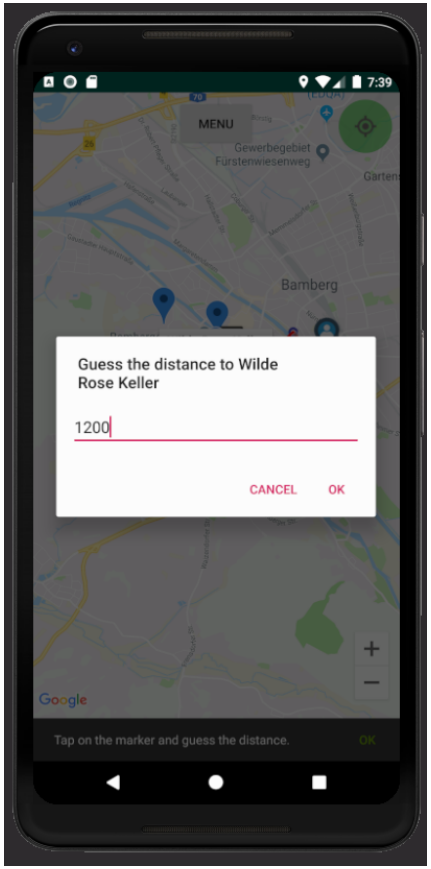
The Application

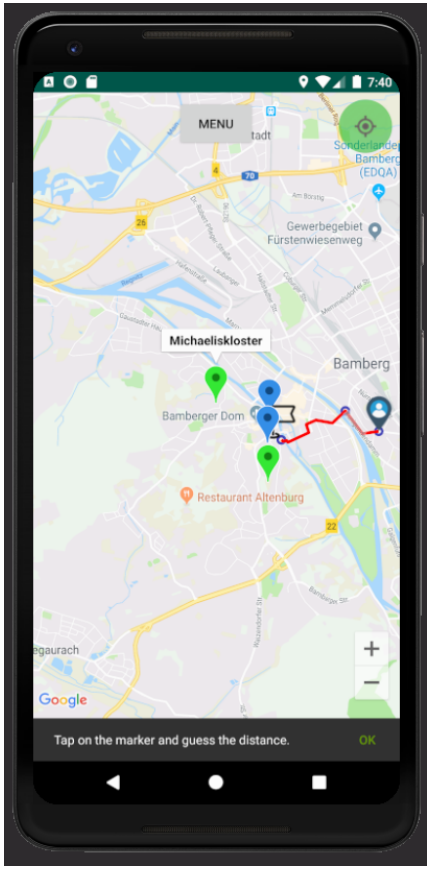


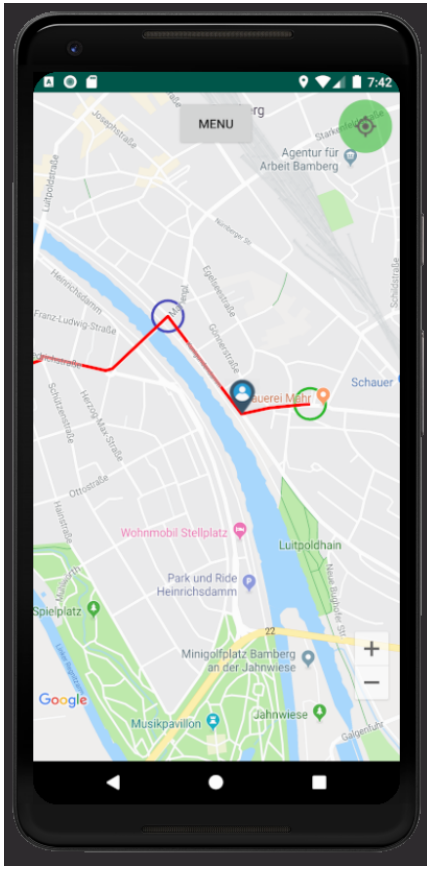




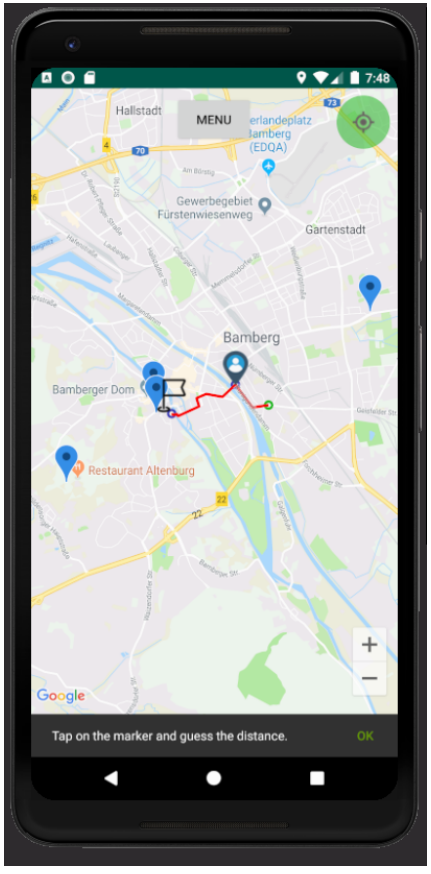
The Application

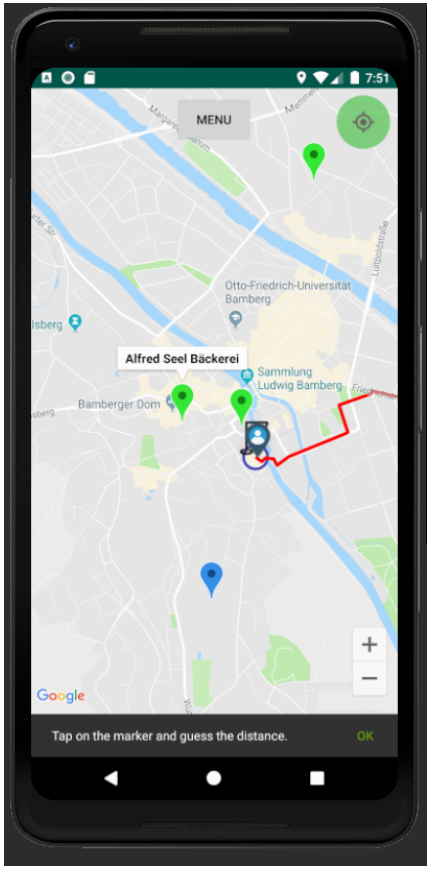


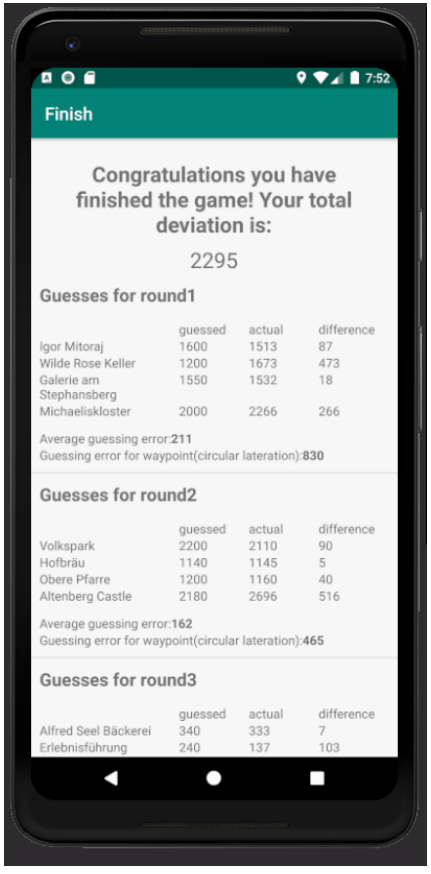




The Application







The classes

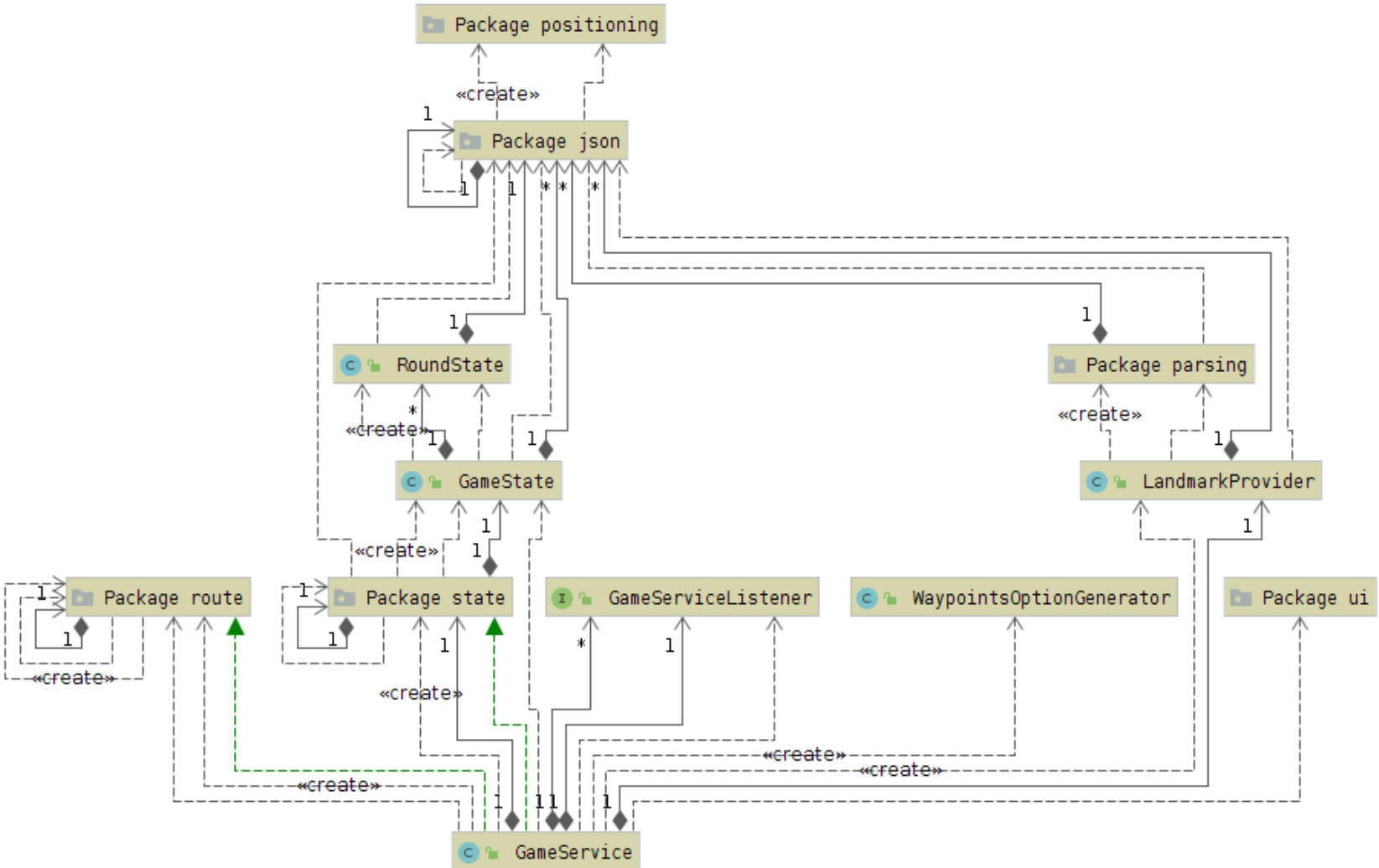


Figure: An overview of GeoRacer's sources

The Landmarks

- Stored in a JSON file
- Import with Gson library
- Map to model classes


```
[
    {
        "name": "Altes Rathaus",
        "position":{
            "latitude":49.8916433,
            "longitude":10.8845945
        }
    },
    {
        "name": "Bamberger Dom",
        "position":{
            "latitude":49.890778,
            "longitude":10.8803263
        }
    },
    {   "name": "Altenberg Castle",
        "position":{
            "latitude":49.8806604,
            "longitude":10.8670233

        }
    },
```


Figure: Landmarks.json

Circular Lateration

1. Call method with locations, guesses, and a starting position
2. Use cartesian coordinates (UTM) for implementing formualae
3. Iteratively approximate position until:
4. Correction vector less than threshold or MaxTries reached

Lateration Loop

```
try {
    while (vectorLength > THRESHOLD && counter <= MAXTRIES) {
        counter++;
        // 1. Residuals
        double[][] residuals = getResiduals(guesses, utmResult);
        // 2. Design Matrix
        double[][] designMatrixArray = calculateDesignMatrix(guesses, utmResult);
        // 3. Correction Vector
        SimpleMatrix correctionVector = calculateCorrectionVector(residuals, designMatrixArray);
        vectorLength = Math.sqrt(Math.pow(correctionVector.get( row: 0,  col: 0), 2) + Math.pow(correctionVector.get( row: 1,  col: 0), 2));
        // 4. Repeat until sufficient precision reached
        UTM utm = new UTM( zone: 32,  letter: 'U',  easting: utmResult.getEastling() + correctionVector.get( row: 0,  col: 0),  northing: utmResult.get
        utmResult = GeoLocation.fromUTM(utm.getEasting(), utm.getNorthing());
    }
} catch (SingularMatrixException e) {
    throw new DegradedMatrixException(e.getLocalizedMessage());
}
return utmResult;
```

Figure: The lateration loop

Circular Lateration - Problematic Cases

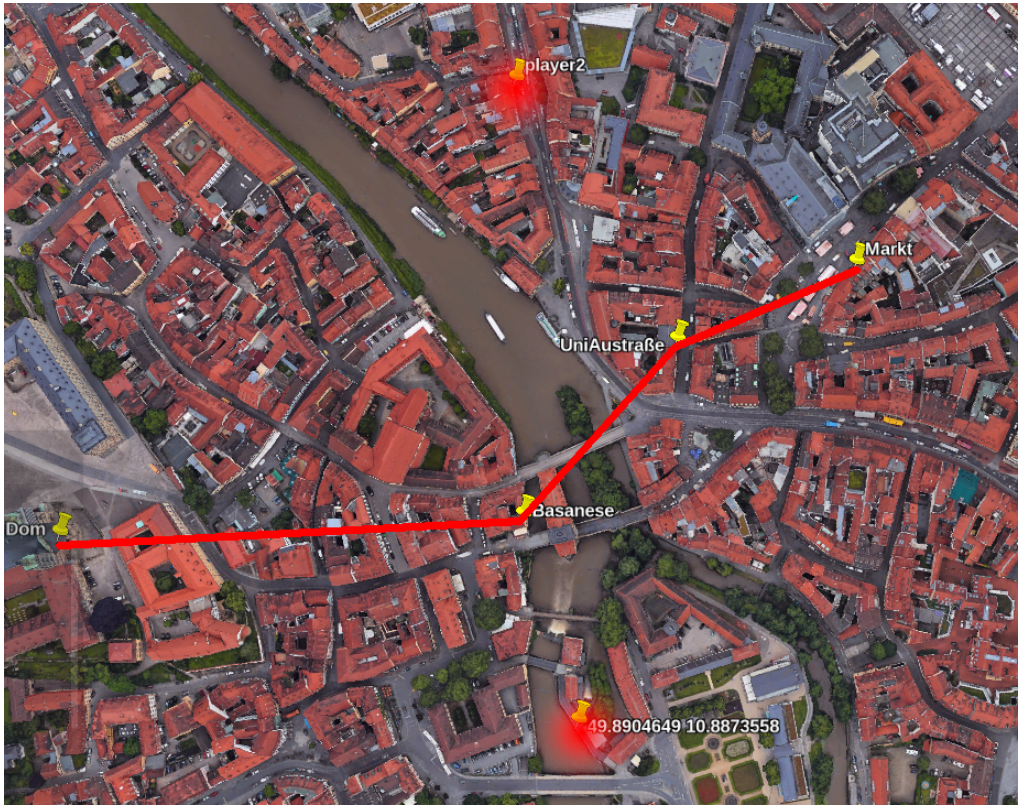


Figure: Almost colinear landmarks are problematic

Coordination of work

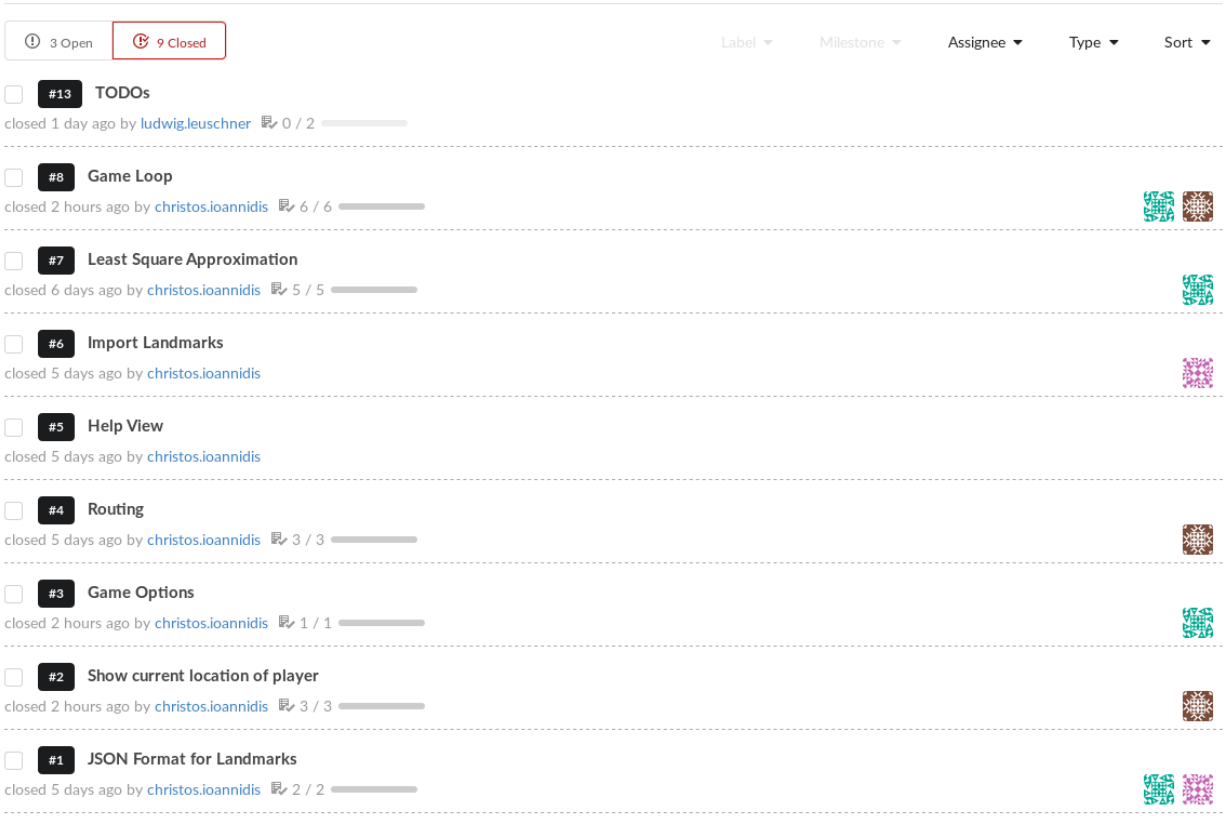


Figure: Issue Tracking with Gitea

Responsibilites

Task	Assignee
UI	Ludwig
State	Ludwig
Circular Lateration (Tests)	Christos
Landmarks	Pio
Scoring	All

Table: Seperation of tasks

Conclusion & Future Work

- Check for colinearity of prompted landmarks
- Idea: Use predefined combinations of landmarks instead of computing linearity
- This should minimize resource consumption and time on the device
- Optimize user experience under usability aspects
- Take advantage of lifecycle hooks (onPause, onResume) in order to pause the game in the background to minimize battery drain