Project Plan – Map Guessing Game

1. PERSONAL INFO
Tuukka Korpela – 947804 – Master's in Geoinformatics – 1st year – 23.2.2024

2. GENERAL DESCRIPTION OF THE PROJECT
The basic interface of the game would have an interactive map which gives the player information about a country when the country is selected. There would be a few game modes to make the application more intriguing: a guessing game where the player tries to guess a country based on the country's borders, and another one where the player guesses the country based on the country's flag. Additional hints (e.g. capital and population number) about the country will be given if the player guesses the country wrong.

I would like to make my project unique by experimenting with audio. It would be cool to have a game mode where a clip of the national anthem would be played for the player, and the player makes their guess based on that.

This implementation would adhere to the "hard" level given in the instructions but I might have to re-evaluate this depending on the amount of work the project ends up being.

3. USE CASE AND UI
The typical use case for the game would be that someone wants to relax a bit by playing a game, but wants to combine relaxation with gaining bits of geographical knowledge. Another use case would be when two (or more) friends want to have a little competition on who has the best grip on the world's countries.

The program communicates with the user with visual effects, such as the outlines of a country, and text. If the idea about national anthems can be realised, then audio has an important part to play, as well. The input data for the program could be, in its most simple form, a background map as an image and a programmed overlay of the country outlines. The country outlines can be drawn in e.g. Illustrator as .svg-files. I have to look into the possibilities for the data, but I assume .shp-files for creating the map can be useful. GeoJSON and Geopandas libraries for Python might be of use if the base map cannot be realised by only an image but needs more interactivity.

The user input is mainly clicks of the mouse. The player selects to just view the map or select a game mode, clicks countries in that game mode, and at some point exits the program. Dragging the map and zooming in and out with the mouse need to be supported. Hints about the countries pop up on the screen as a text prompt. This prompt needs to be closable with a mouse click.

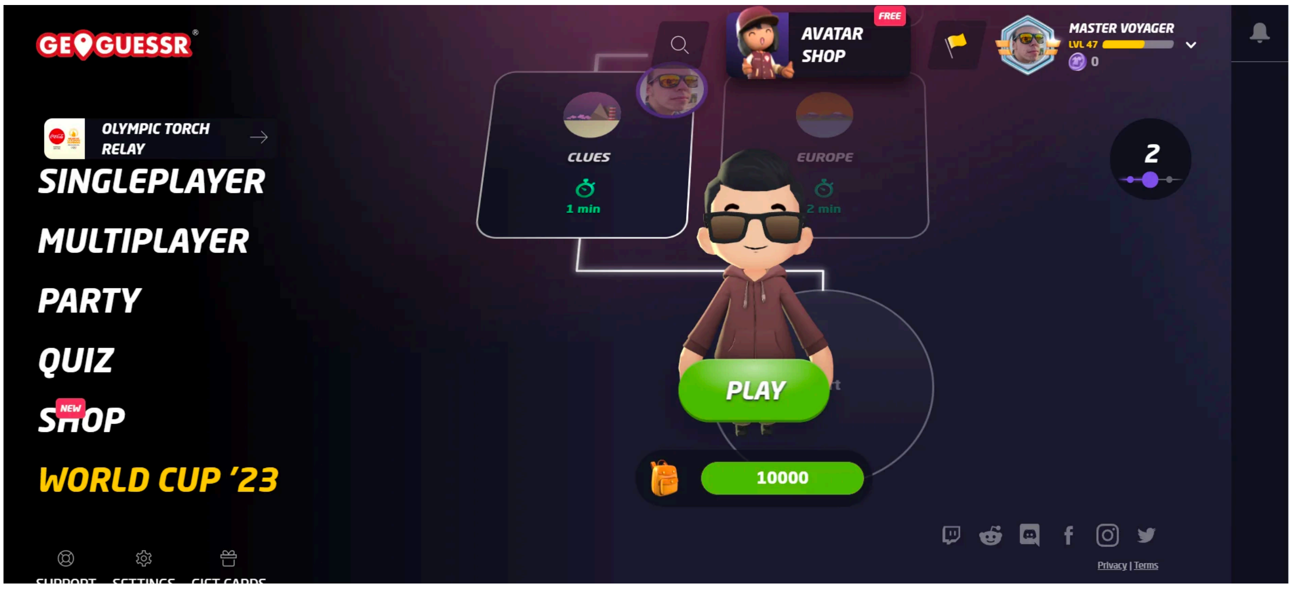

Figure 1. An example of the welcoming screen with different game modes (Geoguessr, 2024).

The most important windows for the UI are the welcoming screen (something similar to figure 1) in which to choose a view mode for only viewing countries or a game mode. Another window would be the actual map where the player can make guesses which country is to be guessed (something like figure 2). This window needs the zooming and dragging/panning functionalities.

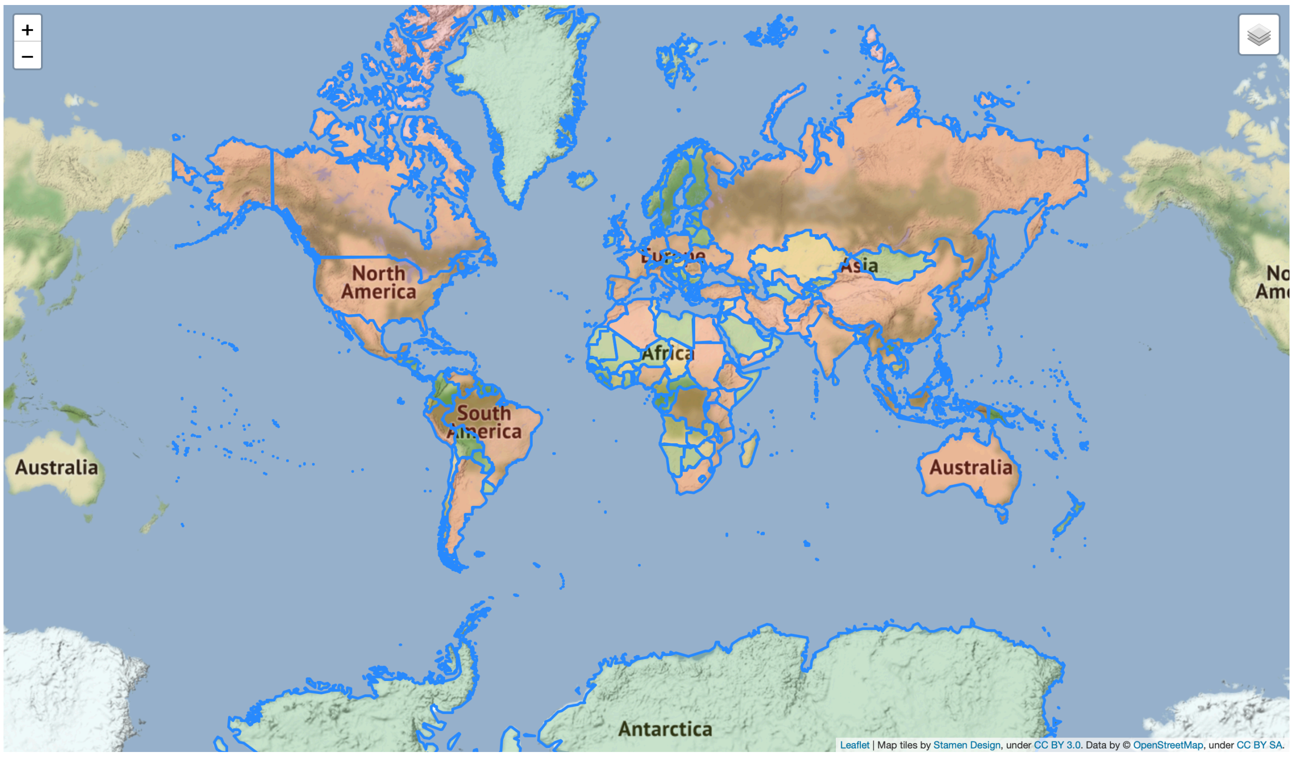
+
−
North America
Europe
Asia
Africa
South America
Australia
Australia
Antarctica
Leaflet | Map tiles by Stamen Design, under CC BY 3.0. Data by © OpenStreetMap, under CC BY SA.

Figure 2. An example of the game screen (Map Mania, 2024).

The user needs to also be able to access some kind of a record of results from previous runs to be able to have a sense of progress. This would also be nice to show the user in-game if they make a new record. The records table could just show the top three results. Every time a result is good enough for the top three, it replaces an existing result and the old result is deleted. This way the records table doesn't affect how well the program runs by recording all previous results.

4. PROGRAM'S STRUCTURE PLAN

I have tentatively divided the classes into three different categories (figure 3): Scene, Graphics, and Records. Scene class is the parent class for different windows/scenes that the player sees while in the program. The Graphics parent class gives an idea how different graphical elements are used in the different scenes. The Records class is still quite bare bones, but is there to illustrate that there needs to be some kind of manager class for keeping record of the inputs the user makes so that the hints for the correct country progress reasonably and that the score is recorded for the player to be able to see previous results.

The division is very coarse at this point and will need updating when the project evolves, but these are the classes that I thought might be the backbone of the program and give me a place to start from.

5. DATA STRUCTURES

A dynamic data structure is needed for the records scene and manager, in which results of the player are stored and deleted when certain conditions are met. Giving hints for the player could be done with a dictionary structure where the name of the country works as the key and the hints for that country are stored as the values for that specific key. The values for each key in the dictionary should be, however, ordered so that the hints get progressively easier. The dictionary for each country needs to be stored in a list that updates itself based on the inputs of the player: if a country is guessed, it should not appear as a possible guess again during the same game run.

The scenes and the different graphics objects could be stored in a multidimensional array from where a scene/graphic is fetched based on the player inputs and the progress the player has made.

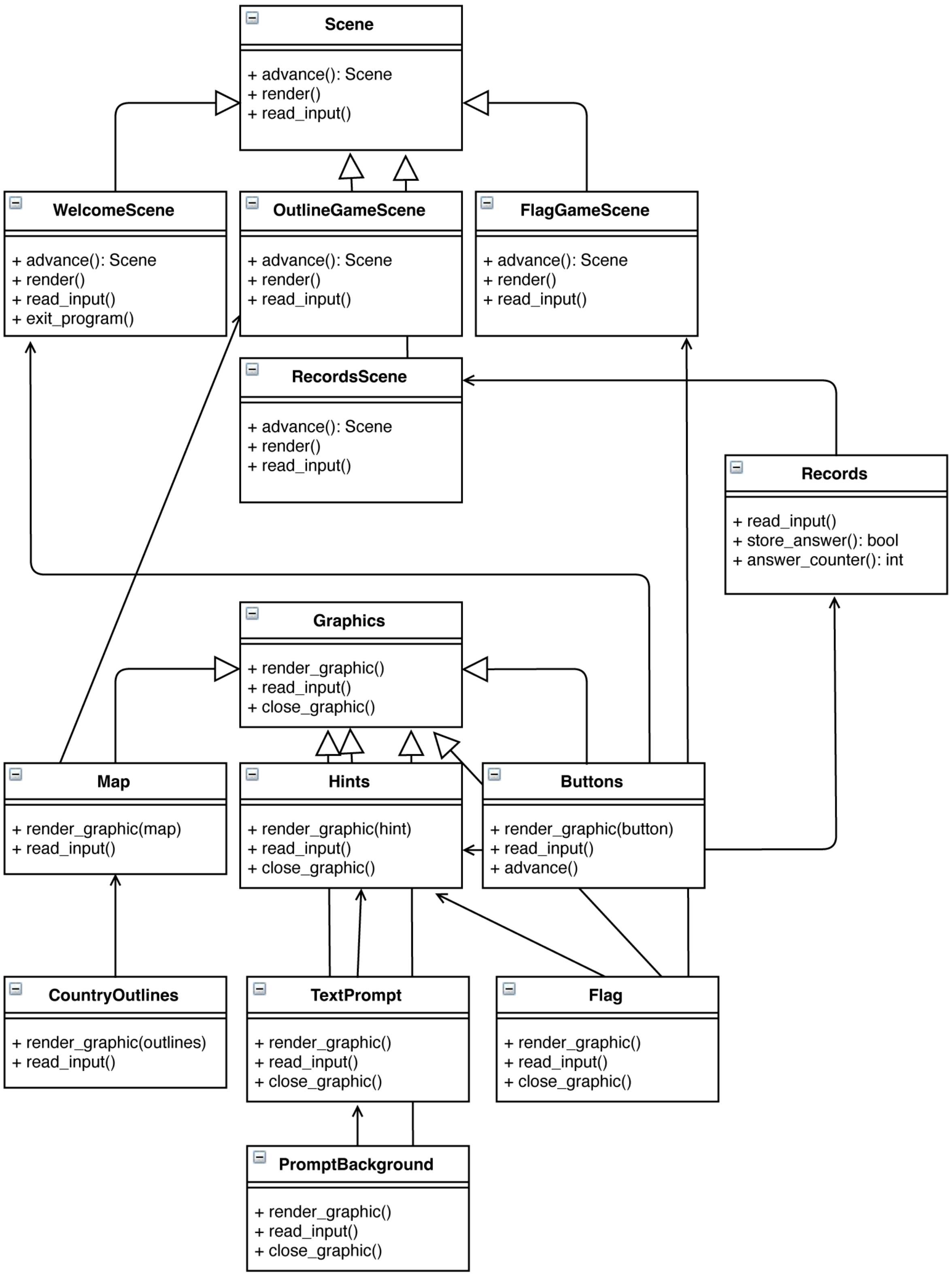

Figure 3. First version of the UML class chart describing the structure of the game.

6. FILES AND FILE FORMATS
The text prompts for the hints could be stored in a text file which is fetched every time a hint is needed. The outlines and flags for the guessing could be stored in separate image files.

7. ALGORITHMS
Simple addition and subtraction is needed for keeping record of the results and the guesses made by the player. Division and multiplication are involved when giving points to the player based on the number of hints it took for them to guess the country correctly. More guesses should lead to less points and vice versa.

8. TESTING PLAN
Testing the proper navigation of the program is important. I need to be sure that the player can switch between the scenes correctly and easily and not get stuck in one window. The player should be able to leave the game scene whenever they want, even in the middle of a hint prompt. It is also important to make sure that the player cannot affect the game score by switching scenes in the middle of a game: the game needs to restart every time a scene is switched.

Another crucial point to test is checking the player input and that it's recorded correctly. For example, no mouse clicks should be recorded as country guesses during a hint. If a country is clicked, no subsequent clicks should be recorded until the player is ready to make another guess.

Recording the results is also important to test. Is the correctly guessed map taken out of the possible guesses? Does the point system work correctly? Is the result of a game recorded properly into the records scene?

9. LIBRARIES AND OTHER TOOLS
As mentioned previously in section 3, I could utilise e.g. the Geopandas, Cartopy, Folium, etc. libraries for making the interactive map. However, I think that creating the interactive map is also possible with a good enough resolution image of the world map.

10. SCHEDULE
My thinking is that as a beginner of Python programming, implementing a full-blown program will take a lot of time and effort. Already shown by trying to create this plan, creating a program has a large number of unknowns to me that, I assume, only become understandable when encountered head-on. While still battling with the basic syntax of Python, I will try to allocate time for doing this project quite generously.

First, there is the time it takes to get an idea of the PyQt library and the GUI implementation, which is quite a big part of this project. Looking at the work-load for Round 6, it seems that getting the basics down with PyQt takes around 30 hours. Obviously, I hope that this will take some time off from the GUI implementation for my project by teaching me how to use the library, but I'm betting on putting about the same amount of work (on top of the round six training) on implementing something completely personal, so for the GUI I'm tentatively allocating 30 hours of work. In other words, I should've probably started with the GUI already last week. I'd say that a good place to start the implementation of the program is the scene navigation. In the first version I will only implement the scenes for the welcome scene, one game mode, and the records scene. It should be intuitive and gets me on my feet, even if the game modes don't work properly yet.

Developing the different game modes is the next obstacle. I will start with one game mode and get that working. With one mode I can test the functionality of my program. Then, god forbid, move on to programming another mode. I believe that the first game mode requires quite a bit of tinkering, so I'm putting a 30-hour tag on that one. The next one can be quite a bit faster because the basic components will probably be similar to the first mode.

The Records class can also prove to be tricky to implement in reality and is crucial for the program to run, so I need to allocate enough time for that as well. I'd throw a 10-hour mark on that one.

In addition to implementing the code, I need to be constantly recording my progress. This can seem like a non-factor but does take significant time if done correctly. Thus, I will put another 10 hours for providing readable and understandable documentation.

I understand that the first checkpoint for the project is the “3th [sic] Checkpoint” mentioned in the course materials on the 22nd of March. With quick maths (30h + 30h + 30h + 10h + 10h = 110h), I can conclude that if I start coding next week and put in 30 hours of work a week (for four weeks), I can submit a program with some functioning code on that deadline.

For the second deadline on 12th of April, testing the program is needed on top of finalising the code. This will take a lot of time, as well, but I hope to ease my workload then with doing as much as I can for the first deadline.

11. REFERENCES

- PyQt documentation: https://wiki.python.org/moin/PyQt
- If I use the mentioned external libraries for creating the interactive map, e.g. the course page for a course I’m currently finishing can be used: https://sustainability-gis.readthedocs.io/en/latest/?badge=latest
- This course’s A+ documentation will definitely be revised
- Python docs: https://docs.python.org/3/
- PyCharm docs: https://www.jetbrains.com/pycharm/features/tools.html