Project Document – CS-A1123

1. Personal information

- Map Guessing Game,
- Tuukka Korpela,
- 947804,
- Master of Geoinformatics,
- 9.5.2024

2. General Description

The game has two modes, a game mode where the player can guess a European country from a map highlighted by the country's flag, and an interactive map which gives the player information about a country when the country is selected.

Additional hints (e.g. capital and population number) about the country are given if the player guesses the country wrong. My project's uniqueness comes from experimenting with audio. The national anthem is played for the player in the interactive map. I took some freedom in the implementation of the game, so I'm not sure if it hits the "hard" level described in the requirements. Strictly speaking, it doesn't even meet the medium level. You make the judgement.

3. Instructions for the User

Run the main.py module and you should be set. There are two options to choose from: the game mode, or the interactive map. Navigate between them with the arrow on the up left corner of the window. Unfortunately, I didn't have time to implement a "reset" button for the game, so if you want to start again, you need to just restart the program. The game mode is also quite long. There's not going to be any incredible victory screen in the end, so play the whole thing through only if you really want to go through (almost) all 39 of the European countries.

Be wary that the interactive mode does indeed play audio, so if you're in a lecture hall, mute your device before playing.

4. External Libraries

If it is considered as an external library, I used the random library to shuffle the countries to be guessed in my game and in testing. Otherwise no external libraries have been used.

5. Structure of the Program

The program is basically structured into four distinct parts, excluding testing and the main module that is only used for executing the program. The four parts are:

1. Main widget
   - The main widget is responsible for creating all the game objects from the other classes and navigating the GUI. It is the backbone of the program, so to speak.
2. Main game mode
   - Main game mode is the class that creates the game mode object. It is heavily dependent on the main widget's get_input() method that feeds it the player's guess for a country. The main game mode then checks the answers (method check_answer()) and keeps the game rolling by giving the correct amount of points per guess and changing the country to be guessed when needed (show_outlines()). For creating a unique gaming experience every time a new game is started, the shuffle_lists() method makes sure that the guessing order of the countries is always randomised.
3. Interactive map
   - The interactive map class takes care of the interactive map mode. It monitors the mouse clicks and highlights the clicked country, shows its info, and plays the national anthem. The

mouse click monitoring (with the method get_clicked_region()) is quite poorly executed, I think, but it works.

4. Countries information
   - A class that was created for being able to retrieve the needed country info in the main game mode and in the interactive map. A very useful supportive class for the other classes.

In addition to these four classes, there is the testing module and the main module. Testing is – as the name suggests – for testing some of the main functionalities of the program quickly. Main is just for executing the program. The “anthems” folder contains all the national anthems and the “GUI_items” folder contains all the pictures used.

6. Algorithms

I am a bit unsure what to write in this section. I would appreciate a definition of “an algorithm” in the instructions.

Everything I’ve done in my program runs with plus and minus calculations. The most “sophisticated” algorithm that I can think of in my program is the mousePressEvent() method which is connected to the get_clicked_region() method in the interactive map class. The mouse press event is entirely handled by PyQt. It records the position of the mouse press in x and y coordinates. In the get_clicked_region() method I use those coordinates to create “selection boxes” for each country and with those selection boxes I can determine which country info needs to be displayed. I think I have done it the hard and very manual way, but I couldn’t come up with a better one with the knowledge I have. In any case, it works surprisingly well.

7. Data Structures

Mutable dictionaries were used as the main data structure to store the country info in the Country class in the countries information module. Dictionaries are nice because I can access the values with a specific country key. I think this makes the code easier to read and understand.

I have also used mutable lists to shuffle the countries to be guessed in the main game mode. Mutable lists are good because I want to make every game unique, as was explained earlier. I used lists in the interactive map class to store the “selection boxes”, but this could’ve been done with an array, as well. Probably would’ve been better, even, to use an array because I always used x-coordinates first, and y-coordinates after.

8. Files

My program is quite heavy on image (.png) and audio (.mp3) files. Even though the player might think that I’ve used only one background map for the modes and only show certain parts of the map whenever a guess needs to be made, he/she might be surprised that every highlight is its own map and image. This is a bit data heavy and would be optimised (preferably only one image for the background) if the game would be developed further, but for now it was the only way I could fathom to do the business, and so I did the business.

9. Testing

I created tests for the three main classes: main widget (ensuring that the navigation works), main game mode (check that the points system works), and interactive map (check that the right info is displayed). The main navigation was tested with QTest for simulating mouse and button clicks, but the others were done with unittest. The unittests were implemented in a “positive” style where I checked that the assertions are true when the code works as intended. All the tests are OK. More time could’ve been used for testing the program more rigorously, but I think the main parts are tested with my implementation.

10. Flaws

The main two flaws of the program are that the game mode cannot be reset without rerunning the whole program, and that the number of countries to be guessed cannot be explicitly determined by the player. Correcting these two flaws would make the gaming experience much smoother and easier for the player.

One minor flaw is that checking the answers is extremely unforgiving. The player has to write the countries exactly as I've written them (I hope correctly) in the Country class. With more time, I would've implemented an "almost there" hint that would tell the player that all they have to do is to check the spelling of the country.

These are not game breaking flaws, however, so I'm quite happy with what I got.

11. 3 Best and 3 Worst Areas

Weaknesses:

- The weakest link is the "selection box" implementation for the interactive map. A lot of manual work. Not sure how to do it otherwise, however.
- Recording the scores would've made the game feel more like a game. Score tables could've been created which would've made competition possible.
- More rigorous testing would be beneficial.

Strengths:

- The game looks nice and adding the national anthems was a triumph for me. It has a finished feel to it.
- The game is quite intuitive to navigate through. By giving instructions and hints about e.g. the guesses left and where to write your guess help the player to play the game easily – I hope.
- I tried to be detailed in commenting my code. I hope that helps the reader to understand it better.

12. Changes to the Original Plan

No major changes had to be made to the original plan. I scratched the statistics idea, but otherwise I stayed with the plan, only with minor changes. All in all the project went mostly as planned and no major hiccups were encountered. I used less time than I originally planned for the project, but got a result that I was happy with anyway.

13. Realised Order and Schedule

I kept the ordering flexible and followed my issue tracking and the things I jotted down in the issues. Check the issues, and you'll get a good idea about on which dates I implemented certain parts for my game.

I started the work by creating the main stacked widget with which I could navigate between the different scenes of the game. To test the navigation, I created dummies ("scenes" displaying some text) that worked as my main game mode and the interactive map. When I was happy with the navigation and had the main frame working, I started with the main game mode. After that I moved onto creating the interactive map. When I got all the main pieces in place, I created some tests and started to fine-tune the different parts of the game, making some quality-of-life improvements.

14. Assessment

I am quite happy with my game. I managed to create something that works and looks nice, as well. Many improvements, which I have already touched upon in this document, would be needed to make it feel like an actual game, but for my first ever bigger programming project I think it's not too shabby. There are no significant shortcomings, unless you are literal with the requirements in A+. I have some extra that is not mentioned there and some things I've left out.

The structure of the program feels intuitive to me. One improvement could be dividing the main widget and main play widget clearer. They are perhaps a bit too intertwined at the moment. Adding more features to the game shouldn't be too difficult, because I've tried to make it clear how everything is structured. If another widget is needed for the main stacked widget, one should know about how the stacked widget works. That's a possible problem for expanding the game.

15. References

All the audio files were downloaded from the site https://nationalanthems.info/ which claims that the content is licensed under the Creative Commons license. The background map was downloaded from here: https://www.vecteezy.com/vector-art/5353681-doodle-freehand-drawing-of-europe-map. That is also free to use.

For coding I used the https://doc.qt.io/qtforpython-6/ website, Stack Overflow, A+, geeksforgeeks, and ChatGPT.

16. Attachments

I've recorded a quick screen recording of me playing the game. It can be found from the "document"-folder from my GitLab with the name "demo.mp4". It has no audio, which is an important part of my program, so try the game for yourself, as well.