**Programming Journey – Escaperoom ‘Where is my Emma?’**

**What additional things (libraries, paradigms, etc.) did you learn during the project?**

The main new aspect we learned is the Pygame library. We learned how to display a frame and ‘blit’ images on it. We learned how to handle userinputs or so called ‘pygame.events’ (Keyboardinputs as well as mouseclicks). Further because of the high interactivity we learned the use of threads in python. The concept of threads was already known (‘Einführung in die Softwareentwicklung’), so we fastly came to the decision to use them for handling simultaneous processes.

With a (for us) new app ‘Poti poti’ we learned to design pixel-images and display them. The pixel-look helped a lot for identifying the placement of the many ‘buttons’. With paint we could reduce all images to our wanted resolution in pixel\*pixel. For design aspects we also learned to use/load a .tff font and display it on our game.

We learned how to handle blocking user-input (with blocking while-loop) grafically. Global variables. You need to state at the beginning ‘global x’ if you want to change x in this function. Access is possible by only referring to x.

[For the scientificier part of the program I learned to connect matplotlib with pygame and display plots there]

**What challenges did you face?**

One challenge we faced, was an updating problem. We learned that if file ‘a.py’ has a global variable that has been changed, file ’b.py’ that has an import statement at the beginning: ‘from a import \*’ the variable is not automatically updated in file b.py ’ after the change.

We approached this problem by printing this variable in both files and recognized asychnority.

First we tried to import it again (after the change was made) and it worked. Later we solved that problem more elegant by implementing a getter-function (modularity and visibility reasons).

Another problem with the import was the meaningfull connection between several .py files. This was (spätestens) clear as we learned that you cannot import file b.py in a.py and vice versa (simultaneously) which is logical because then you could write everthing to one file. We approached this by splitting the program concepts in files startscreen, endscreen and door1, door2 and door3 at the beginning. After implementing and some redundancy we implemented display\_components which holds functions and status variables that are used by every file (or the majority of files). The ‘bigger’ function handle\_userinput was put in one extra file.

A further but quickly solved challenge was the simultaneous gameplay an time displaying. Because (as already stated in the beginning) the concept was known, we came to use threads. One (probably the main one, like in java) handles the gameplay and user-input and the other one the display of the time. First (naive) approach was that everything is handled by the main process but even after setting the fps to 60, one second was incremented every three (or so) because that thread had so much other to handle inbetween.

**What further additions could be made?**

I think the first obvious addition could be increasing the complexity of the rooms task (like in real escaperoomgames) or add levels/new rooms. If that is not wanted, one could also add a look-around function which lets you look around in the room from different perspectives (also like in real escaperoom games). We already implemented some zooming functions so equally we could let the user ‘turn around’ to look at other things in the rooms.

We also have implemented one task, were the key to the next door is collected but neither acknowleged or displayed, so here could an itembox be added that displays it. Furthermore we had the idea of an Avatar which is seen in the loading screen or the final\_words room, but we did not really embedded it. Here is room to improvement.