**North East Scotland College**

**Irvyn Hall Games**

Python/Tkinter benchmark game

Author: Irvyn John Hall

**CONTENTS**

**1 Introduction**

**1 Reaction Game**

**2 X and O’s**

**2 Number matching**

**3 Bug Fixing & Submitting**

**3 Complications in development**

**3 Future versions**

**3 Conclusion**

**Introduction**

When planning out my GU project I had already decided on doing a human benchmark type of game so that I can use it myself on a day-to-day basis to see if I can track improvements with my video game skills. I managed to do some research on some of the constructs I knew I was going to employ as I had been carrying out another task on one of my own projects out-with college.

I chose to do the program in python as I had thought out everything and really considered trying to use another programming language but I’m most familiar with python and for everything I needed python had available ALTHOUGH I should of used C++ for the reaction game as it is much more accurate at measuring small time frames as it is a compiled language and python is an interpreted language.

[python proof](https://stackoverflow.com/questions/1133857/how-accurate-is-pythons-time-sleep)

[c++ proof](https://stackoverflow.com/questions/13397571/precise-thread-sleep-needed-max-1ms-error)

The obvious draw back of running a C++ reaction time would be the CPU usage IF I chose to use spinlock which I also read about [here](https://blat-blatnik.github.io/computerBear/making-accurate-sleep-function/)

TLDR: I stuck with python despite the MINOR inaccuracy

I also knew that I would be inheriting the same tkinter buttons for all the multiple games and I had a very brief idea of how to do this due to a youtube tutorial I watched on tkinter which can be found: ([here](https://www.youtube.com/watch?v=yQSEXcf6s2I&list=PLCC34OHNcOtoC6GglhF3ncJ5rLwQrLGnV))

**Reaction Game**

I was aware that for the reaction game if the main program crashed or threw and exception the whole reaction count-down function should be ran on a separate thread and that it will allow for the users score to still be saved/counted despite the main program crashing.

For saving scores I chose to write to json files as once again out with college I recently had to make a script to go through json data and remove data from a json file so it was fresh in my memory. I understand that in a real world example saving data to a json file is really unsecure unless I sanitize all data – [example](https://www.acunetix.com/blog/web-security-zone/what-are-json-injections/#:~:text=JSON%20Injection%20Prevention,before%20serializing%20it%20to%20JSON.)

I ultimately chose a json file as 1. it was fresh in my memory and 2. it will give me marks for unfamiliar constructs.

Back to the saving scores function it was obviously the most complex part of the reactiongame - I had to make around 2 changes because when I blackbox tested I found errors with the logic of it – the first iteration of the function would only save the first score then only the scores HIGHER than the previous score would be saved which is not actually a better score (obviously all errors are fixed now)

I also made use of another unfamiliar construct by running the light\_runner on a daemon thread – this again like I explained in the introduction allows the score /game to be counted even if the main program hands or throws an exception.

**X and O’s**

The biggest issue I ran into with the X and O’s was the function to determine weather there was a win or not obviously I started with multiple huge if statements to check all the types of patterns but when it came to the end of creating the program I went back through my code and refactored everything – I ended up with an array of all possible win patterns that will check if the matrix (X and O’s game tiles) had any of the possible win patterns in it which is A LOT cleaner than the 19 if statements I had originally.

I called the X and O’s a “matrix” as when I was doing a previous college assignment (quidditch game) when I was researching for help I came across [a document](https://www.geeksforgeeks.org/array-vs-matrix-in-r-programming/) which refers to these arrays/arenas as matrix’s in game development.

**Number Matching**

This is the simplest of the games it really did not require much thought to program, and I did not have any issues with it during testing. I obviously used the usual function to create the base buttons - I have a counter for each choice where if the user picks a number and the next button is the same number then it will save until a wrong answer is given – I did this to make the game harder and so you cannot simply just spam buttons though the game – you must remember each individual button pairing. Again, like the X and O’s doing the quidditch game in class allowed for this to be easy as I used similar setters and getters in both to generate the matrices.

**Bug Fixing & Submitting**

I adhered to my ghantt chart and the bugs I found were all able to fix and I managed to hand in my completed project with around 1 – 2 days in advance. I managed to get estimated a mid-range B which is completely fine with me I don’t feel as if there are any remaining bugs I did both whitebox and blackbox testing as I had one of my friends blackbox test my program from a .exe – I might be wrong but from the lecturer feedback and feedback from my friend there doesn’t seem to be anything.

**Complications in development**

Along with my out of college work I realized I had too broad of scope for the game and I mentioned doing a word memory based game which would of read words from a .txt file and show them and get the user to input weather it’s a new word or a repeated word but I realized that in essence the number matching game will train / benchmark the same exact skill (memory) so I decided to scrap that idea and shorten my development scope – this wouldn’t of really effected my mark by much either as I don’t think there would have been marks picked up from the word guessing game as there would have been no new constructs used.

**Future Versions**

I could 100% improve the graphics of the program as its very basic tkinter buttons for the time being and I could easily add more games ontop of the program which I planned for – it is evident by the way I created the button class to allow it to have multiple inheritance.

Also like mentioned above I could recreate the game (s) in c++ to allow for more accurate reaction time calculation from a compiled language rather than an interpreted language.

**Conclusion**

In total I’m happy with the way my GU turned out I have some regrets about the choices of games I feel as if the X and O’s took more time than it should of and probably didn’t allow me to make use of any new concepts however the reaction time game is without a doubt my favorite because of one learning how to use a daemon thread in python and two just how easy it is to save and update scores / usernames with json files (I also understand the security risk like I assessed above) This project has made me more confident in my knowledge and ability to program in general.