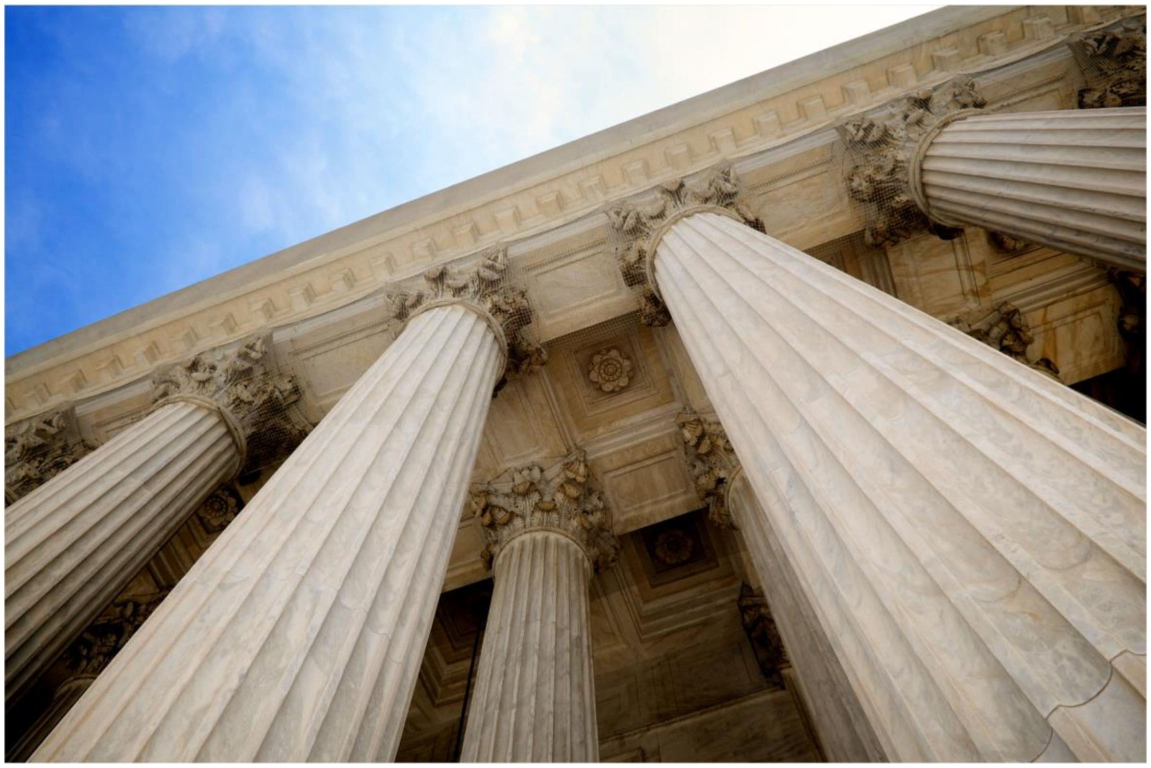

CS3305 Individual Report

TITANIUM GRAVITY WRANGLERS

Grzegorz Ikwanty | 114320191 | March, 2017

Introduction

My part of the project was the server side of the game. I was in charge to making the game run in accordance to the rules and handled all communication between the server and all clients. Messages ware processed by the server when received. I decided to use Python3 as the language to write the server in as it is quick and easy to write scripts. Most incoming messages were put into a queue and await processing. I chose to use queues as the built-in Python module for them is thread safe. This allowed me to run two threads simultaneously and both could alter the queue without interfering with one another.

My integrated development environment (IDE) of choice was PyCharm from JetBrains. I decided to use it for two main reasons. Firstly, it checks the code throughout writing it. JetBrains' IDE works especially well with scripting languages like Python as these languages don't compile. PyCharm catches a lot mistakes that would show up if Python was compiled and highlights them as color coded points on the sidebar. It also checks compliance with the PEP8 standard. Secondly, as a student I get the professional version of it for free.

In my group we decided to use transmission control protocol (TCP) as delivery of packets was had a higher priority than how quickly packets get sent. This decision lead to a biggest surprise I had to deal with when programming the server. TCP sends packets as a stream of data, which means sometimes messages would get added onto the previous messages. To overcome that I had to split the incoming messages before being able to convert each individual message form JavaScript Object Notation (JSON) into a Python3 dictionary.

Lessons Learned

There is many important lessons that I learned throughout the development of this project. The best first step to starting group projects is to set up a repository where all code would be stored. In my group we decided to use github.com. Due to working form multiple machines, I quickly learned the importance of remembering to push changes frequently and remembering to pull changes from the repository whenever coming back to the code. There were multiple instances where I forgot to push code and ended up redoing all changes that I already did once before on a different machine.

To send and receive messages to/from clients I was using the Python socket module. I have only briefly been introduced to sockets before this project. Major lesson about sockets that I learned is that TCP data is sent as a stream and therefore sometimes messages get added together. This meant that the JSON decoder wasn't able to load the message into a dictionary. Messages has to be split on a string like so, '}{' .

```
print("Combined JSON payloads received: " + data)
messages = data.split('}{')
messages[0] += '}'
messages[-1] = '{' + messages[-1]
for i, payload in enumerate(messages[1: -1], 1):
    messages[i] = '{' + payload + '}'
```

Only after doing that the messages ware ready to be processed in accordance to the command received in them.

Threading is one of the most difficult components of the project. I had to decide when each thread should be started and to what resources is should have access. There were three main threads running in parallel to the main program. Discovery thread allowed clients to find the server. The server was listening on a port for broadcasts and replying to them with the current state of the server. The service thread was used for accepting all new incoming TCP communication. The current game messages thread was used to enqueue messages form clients that are part of the ongoing game.

The project was developed in an agile style. For me this was the first project that I worked on that didn't have predefined final requirements. It was challenging at the start to write code that can be built up form small chunks so that it can be easily changed in accordance to new requirements. On top of that working as part of a team posed a new challenge of defining certain parts of the code before writing them so that other team members can assume their functionality and reference them in their parts of the project. To make this easier, I helped to develop an API in collaboration with Conor Twomey from my group and Ciaran Broderick and Tom Cordero for the Super Confused group. This API describes the format and information to be sent across the network between the client and server. This also ensured inter-operability between our groups.

Suggestions for the future years

Overall, the team project was an enjoyable and educational experience. However there is a couple of improvements that could be done to improve the module. First off, having a set specification of each phase of the project from the start would be a major improvement. Knowing things like the group sizes and whether choosing groups is up to ourselves of not would decrease some of pressures put on the group. With the uncertainty of that, I found myself being a part a group that was later disbanded in light of the possibility of groups being chosen at random, leading to having to finding a new group when a definite group size and ability to make own groups was given. Secondly, I didn't like the fact that more emphasis was put on documentation and presentation then the actual working product. I know that in most workplaces documentation is a huge part of any project but never bigger than the actual finished product. Given the 8 weeks to do this project we spent almost 6 weeks just clarifying what to do. All those meeting led to only one thing, that we should get rid of interoperability and just try to get our code working with one or two other groups. In my opinion making a game means making the whole program from start till the end and I wouldn't expect someone to write their own client to play my game. If the aim of this project was to imitate a company, split into groups and work on one collaborative project the interoperability would have worked flawlessly. It would have improved out communicative skills and showed us how to not only work as a team but as a team of teams trying to achieve one final goal.