St. Andrews Direct Entry

Oscar Ocieczek

November 2024

1 Introduction

Hello and welcome to my supporting document for 2025 direct entry into the Computer Science BSc. Here, you will find two projects which I feel show my potential and existing capability as a programmer, and the accompanying information which has been requested for both of them. I really enjoyed making this document, and I hope that you will enjoy reading it! I recommend watching the context videos for a demo of my projects and a further idea of my thoughts on my capabilities as a programmer. Let us begin.

2 Number Plate Recognition (Storage) System

2.1 Context video URL

https://youtu.be/SVGIzEQ1Xjc - please click!

2.2 Responses to questions

2.2.1 Programming languages used

This project is written in Python with a bit of SQL sprinkled in.

2.2.2 Main functionality

The purpose of this program is to be able to predict the contents of a number plate given a (script-generated) image of a number plate. This is accomplished through a convolutional neural network - written without the use of major libraries such as TensorFlow or PyTorch - which is trained on 'synthetic' license plate images generated with the help of the Pillow library.

2.2.3 Scale of project

- Number of classes: 7
- Number of functions: 39
- Number of lines of code: approximately 500

2.2.4 Use of object-oriented concepts

This is my first project where I have made a concerted effort to make use of OOP concepts where I feel that they are suitable and contribute to the structure of my program. For example, I have used encapsulation with my CNN to ensure that the logic of my neural network is fully self-contained and, perhaps most importantly, modular. I also make use of abstraction here, with the *segment_characters* function found in *preprocessing.py* being a great example of this - it is used in creating the dataset of characters for character-level OCR in *create_character_dataset* and it makes the code that little bit easier to read and reuse after this prototype. I have not made use of polymorphism or inheritance in this project as of yet, however I can see myself using these concepts when implementing different optimisers (other than SGD) alongside further UI elements for my final submission of this project for my coursework.

2.2.5 Algorithms and data structures

Probably the most interesting part of this project is my exploration of CNN's and the various algorithms and data structures that are applied to them. For example, the *convolve* method extracts a region of the input matrix and applies a filter across said matrix in order to extract potentially useful features about said region in the form of feature maps. This was quite challenging for me as this introduced me to new concepts such as working in more than 3 dimensions with NumPy, but I got there in the end.

There is also some interesting use of algorithms in the training of my model, with me calculating loss gradients with respect to the parameters of my model (weights and filters). This allows me to use SGD to adjust my parameters, thus essentially allowing it 'learn' the correct classifications of alphanumeric characters found on UK number plates.

In my training loop, I use mini-batch training as I have read, as part of my research for this project, that this strikes a fairly solid balance between efficiency and converging on a minimum loss for my model. I also have made use of the softmax activation function as the output layer for my model, and this needs the use of conversion of my labels to one-hot encoded vectors (1's or 0's).

All in all, I feel that my use of these algorithms without large machine learning libraries makes for an interesting exploration into the fundamentals of supervised learning - something that I was looking forward to after an introduction to the more high-level libraries which focus on abstracting away these mathematical details used in real-world environments.

2.3 Supporting images

These are some images that describe the structure of my project when it will be ready for a final submission as part of my A-level course. I felt it would be interesting to see what I should be capable of by the end of this academic year. Please zoom in!

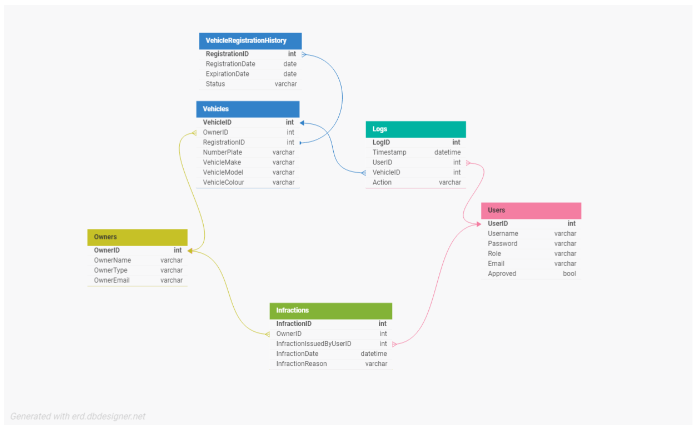

Figure 1: Future Database Structure

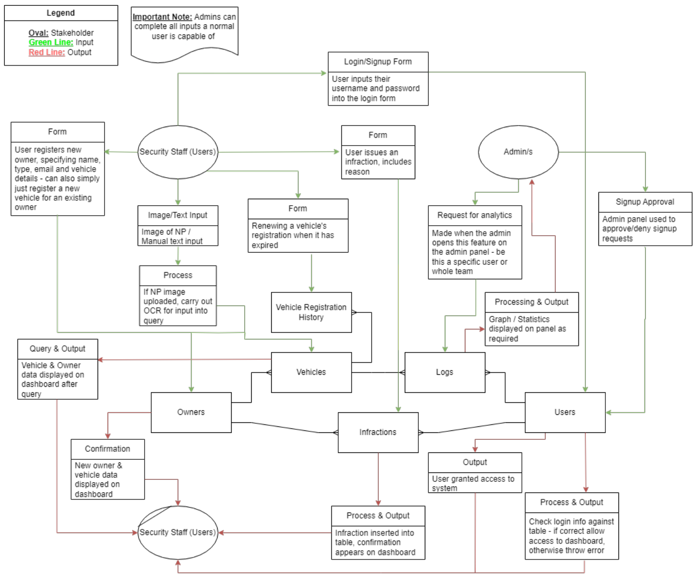
Legend
Oval: Stakeholder
Green Line: Input
Red Line: Output
Important Note: Admins can complete all inputs a normal user is capable of
Login/Signup Form
User inputs their username and password into the login form
Form
User registers new owner, specifying name, type, email and vehicle details - can also simply just register a new vehicle for an existing owner
Security Staff (Users)
Form
User issues an infraction, includes reason
Admin/s
Signup Approval
Admin panel used to approve/deny signup requests
Image/Text Input
Image of NP / Manual text input
Form
Renewing a vehicle's registration when it has expired
Request for analytics
Made when the admin opens this feature on the admin panel - be this a specific user or whole team
Process
If NP image uploaded, carry out OCR for input into query
Vehicle Registration History
Processing & Output
Graph / Statistics displayed on panel as required
Query & Output
Vehicle & Owner data displayed on dashboard after query
Vehicles
Logs
Owners
Infractions
Users
Confirmation
New owner & vehicle data displayed on dashboard
Output
User granted access to system
Process & Output
Check login info against table - if correct allow access to dashboard, otherwise throw error
Security Staff (Users)
Process & Output
Infraction inserted into table, confirmation appears on dashboard

Figure 2: Future Data Flow Diagram - Please forgive the mess...

3 Garry's Mod Programming

3.1 Context

I hate to disappoint, but there is no video for this section! The following code that you are going to see is from my adventures programming plugins for the game Garry's Mod, which runs off of the Source Engine. I spent quite a large amount of time in secondary school programming for game servers as a volunteer, contributing to the communities which I enjoyed playing on - this inadvertently gave me a lot of experience with event-driven programming, and this allowed me to build a rather intuitive understanding of asynchronous programming. I used this to develop a bot for the Discord chat platform in Python (this relies on a lot of asynchronous programming), however this code has been lost to a failed hard drive.

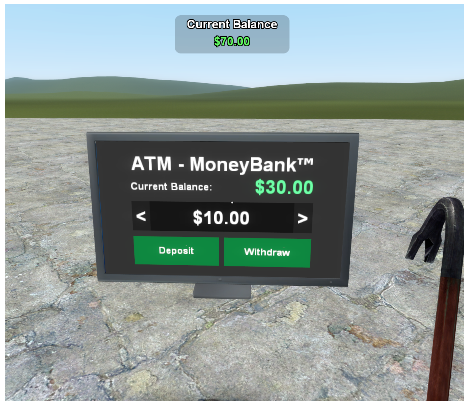

Figure 3: Banking plugin in action

3.2 Responses to questions

3.2.1 Programming languages used

Lua alongside extensive use of SQL.

3.2.2 Main functionality

The purpose of this plugin is to serve as a money deposit and withdrawal system for players on a server, who interact with ATM's (programmed under the entities folder) which provide an intuitive interface for the player to interact with and handle their funds.

3.2.3 Scale of project

Please note that that *imgui.lua* is not my code, it is a library designed to help with creating GUI's. It will not be included in the figures below.

- Number of functions: 35
- Number of lines of code: approximately 800

3.2.4 Use of object-oriented concepts

N/A - there is not much application of such concepts in programming plugins for servers on this game.

3.2.5 Algorithms and data structures

The management and tracking of players in this game is very important, and this often means that I make use of hashtables in my code in order to optimise potentially long operations such as linear searches of the server playerlist to see if someone is online - this cuts down on the amount of 'ticks' my code takes to run on the server-side. I ended up enjoying making sure that I was being as efficient as possible throughout my code, and this extended to the usage of networking messages - I was exposed to using compression techniques in order to make my messages as performance-friendly as possible.

Although this is not necessarily an algorithm, I thought it would be interesting to mention here regardless. I touched on event-driven programming earlier, and this script makes use of this concept throughout through the use of hooks - code which is executed only when a specific event is triggered in-game. Through being introduced to this style of programming first rather than having to learn it after traditional procedural paradigms, I found asynchronous programming much easier to grasp when moving on to programming with Python.

4 Conclusion

Thank you so much for taking the time to read all the way to the end of this (rather wordy) document! It has been a pleasure reflecting on my programming experience in detail. The idea of studying Computer Science at St. Andrews excites me a lot, and I hope that I have conveyed my eagerness to learn and current capability as a programmer to you. Thank you again, and bye for now!