UKRAINIAN CATHOLIC UNIVERSITY

BACHELOR THESIS

Basic Counter-Strike: Global Offensive demo viewer

Author:
Marc SOUMOUSSOU KODJOVI

Supervisor:
Matvii KOVTUN

A thesis submitted in fulfillment of the requirements for the degree of Bachelor of Science

in the

Department of Computer Sciences
Faculty of Applied Sciences

Lviv 2020

Declaration of Authorship

I, Marc SOUMOUSSOU KODJOVI, declare that this thesis titled, "Basic Counter-Strike: Global Offensive demo viewer" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

UKRAINIAN CATHOLIC UNIVERSITY

Faculty of Applied Sciences

Bachelor of Science

Basic Counter-Strike: Global Offensive demo viewer

by Marc SOUMOUSSOU KODJOVI

Abstract

The video game industry is growing fast our days. It has a lot of big companies which have at least one game which is their signature product in some genre. Gaming industry gives tons of working places as it created a lot of different types of job and generates a big amount of sales annually worldwide. One of the biggest and popular games played on different levels from amateur to pro is Counter-Strike: Global Offensive. That's what we are going to talk about.

Code for this project can be found here: Github repository

Acknowledgements

I want to thank my family and my friends especially Myroslav and Volodymyr who supported me throughout these years of studying and gave me a lot of important advices.

Contents

List of Figures

List of Abbreviations

CS:GO	**C**ounter-**S**trike: **G**lobal **O**ffensive
CS	**C**ounter-**S**trike
HL	**H**alf-**L**ife
CT	**C**ounter-**T**errorists
T	**T**errorists

Chapter 1

Introduction

Today Counter-Strike: Global Offensive (successor of Counter-Strike 1.6 and Counter-Strike: Source) is one of the most popular games in the world. It's a multiplayer FPS game which was developed and released by Valve in 2012.
In 2018 CS:GO became a F2P (free to play) game meaning that Valve are focusing on revenue from cosmetic items (weapons skins) and selling of special system that allows you to play competitive games with less cheaters thanks to Trust Factor.
There are a lot of maps that are played in CS:GO both casually by regular players and professionally by pros. But on tournaments pros play only active duty maps which at the moment are: Dust 2, Mirage, Inferno, Vertigo, Nuke, Train and Overpass. They can be seen here: CS:GO Maps. Sometimes the pool changes, for example when some older maps get reworked they might replace other maps that need to be reworked.

1.1 Weapons List and Equipment

Knives: Stock Knife (CT), Stock Knife (T)
Pistols: P2000 (CT), USP-S (CT), Glock-18 (T), P250, Five-Seven, Tec-9, CZ75-Auto, Dual Berettas, Desert Eagle and R8 Revolver
SMGs: MP9 (CT), MAC-10 (T), PP-Bizon, MP7, UMP-45, P90, MP5-SD
Rifles: FAMAS (CT), Galil AR (T), M4A4 (CT), M4A1-S (CT), AK-47 (T), AUG (CT), SG553 (T), SSG08, AWP, SCAR-20 (CT), G3SG1 (T)
Heavy (Shotguns + Machine guns): Nova, XM1014, MAG-7, Sawed-Off (T), M249, Negev
Grenades: HE grenade, Flashbang, Smoke grenade, Decoy grenade, Molotov (T), Incendiary Grenade (CT)
Gear: Kevlar Vest, Kevlar + Helmet, Zeus x27, Defuse Kit/Rescue Kit (CT)

More info about weapons can be found here: CS:GO Weapons and Equipment

1.2 Round Types

There are different round types in CS:GO. Let me tell about them.

1.2.1 Full Buy

Lets' start with Full Buy round. It's a round when the team has a good economy and can afford to buy Helmet + Kevlar and good weapons, also defuse kit for CT and grenades to maximize their chances to win a round.

1.2.2 Force Buy

Force Buy also called Half Buy is a round when each player in the team spends money on equipment so they can avoid Eco'ing next round. This is very risky because if the team loses the round their economy will be destroyed but on the other hand if they win the round there is a potential for the Force Buying team to stabilize the team's economy.

1.2.3 Anti-Eco

Anti-Eco is the round idea behind which is to shut down the enemy while they can't buy good weapons and maximize the amount of money you can earn in the round by purchasing high kill reward weapons like SMG's.

1.2.4 Eco-Round

Eco-Round or simply Eco is a round when the team tries to save as much money as possible for next rounds. On eco you want to buy some pistol but still have money for next round knowing what will be the loss bonus and if you're going to have the money to buy good weapon. Also you should analyze the economy of your teammates so that you don't eco and someone's in your team makes a full buy or force.

1.3 Ranks

There are 18 ranks in game:

- Silver I
- Silver II
- Silver III
- Silver IV
- Silver Elite
- Silver Elite Master
- Gold Nova I
- Gold Nova II
- Gold Nova III
- Gold Nova Master
- Master Guardian I
- Master Guardian II
- Master Guardian Elite
- Distinguished Master Guardian
- Legendary Eagle

- Legendary Eagle Master
- Supreme Master First Class
- The Global Elite

There are a lot of factors that impact player's rank. Detailed info can be found following this link: CS:GO Ranking System

1.4 Rules/Gameplay

There are 2 teams of 5 players in CS that are fighting each other: Ts or TTs (terrorists) and CTs (counter-terrorists). There also are different competitive game modes. In first one Ts should plant the bomb and CTs should prevent Ts from doing that or defuse it and in the second one Ts have to defend the hostages and prevents CTs from rescuing them.
Either team can win a round simply by eliminating the opposite team. All rounds in the game are 1:55 long. The game starts with pistol round and every player has 800$ to spend. There are weapon kills awards: knife – 1500$, CZ75-Auto (automatic pistol) – 100$, other pistols + assault rifles + auto-snipers – 300$, AWP – 100$, P90 SMG – 300$, other SMGs – 600$, machine guns – 300$, shotguns – 900$ and grenade – 300$. There is also weapon called Zeus x27 which doesn't give any reward at all. Also there is loss streak value: 1 round – 1900$, 2 rounds – 2400$, 3 rounds – 2900$ and 4 rounds streak which is the max gets you 3400$ for the round. You don't need to spend all the money because it transfers to the next round if you have something left. The pistol round win by elimination of opposite team will give you 3250$ + 300$ if the bomb was planted (only for Ts). If the round was won by bomb detonation (for Ts) or bomb defusal (for CTs) you get 3500$ instead of 3250$. If the Ts run out of time and don't plant the bomb or eliminate CTs they don't get any rewards but if they plant the bomb they get 800$ bonus added to the current loss streak value. Then teams need to build up their economy to be able to buy better weapons. The game goes up to 30th round and the team that gets to 16 round wins the game. When the game reaches 15th round it's called halftime and the teams change sides. So each team can win by taking 16 rounds first against the other one or they can tie the game with 15:15 score. But if both teams get 15 rounds in pro match they might play overtimes (need to check tournaments rules). Overtime is another 6 rounds. Overtime starts with teams staying on the same sides they finished the regular rounds and with maximum bank you can have. Overtime composes of 3 rounds, there also is side change after 3 rounds played. The first one to 19 wins unless they tie again at 18:18 where it goes into double overtime and so on.

More information about matchmaking rules can be found here: CS:GO MM Rules

1.5 Pro Scene

CS:GO has a professional scene which has a lot of big tournaments held by third-party organizations and the biggest tournament (Major) played every season of the year which is also sponsored by Valve where 24 teams fight for the title of world's best team and 1 000 000$ prize money which is distributed between 16 best of 24 teams on the tournament. There is HLTV top teams which is composed during the year where teams are distributed by the points they earned on big tournaments.

Usually professional CS:GO game are being streamed on twitch.tv (gaming streaming platform). But as professional CS started to grow up in popularity companies started to televise the game in cinemas and theatres. First such tournament was ELEAGUE Major 2017. It even was translated on US cable television.

1.6 CS:GO demos

When you play CS:GO your match is being recorded. After that you can watch the demo in any way you want, for example you can watch it from your POV or other player's POV or you can fly around the map to see the general situation at any moment. Also there is a demo ui that allows you to control the demo, for example stop, speed up, slow down, fast forward etc.
CS:GO demos are games replays saved in .dem files. DEM is a short name for demo. It is a demo file format used by Source Engine (was used to create CS:GO). DEM file contains match events which you can replay in-game. You can find your replays in CS:GO replays folder but if you want to replay pro team's matches demos you can get them on hltv.org site. To want to watch the demo that wasn't originally created on your PC (for example your friend's demo or pro team demo) you have to download the demo and put it into csgo/ folder and then open the game and write viewdemo demoname, where demoname is your demo file name without the .dem extension.

1.7 Problem

Sometimes the top 1 teams from HLTV top in certain periods of time are performing at such high level of play that no one can beat them. Not long ago the Danish pro team Astralis was ranked top 1 team in CS:GO and they had 30+ win streak on one of the competitive maps in CS. Analysts in general couldn't say why this team was performing so good. There most likely are patterns how the team plays and that's exactly what will be checked and analyzed in this project.

1.8 Motivation

I'm playing CS myself so I'm very motivated to learn some new aspects of the it in-game and outside it. I like to watch my demos and analyze them myself and improve in my playstyle in every way I possibly can. So that's why I want to make this project to improve even more than now by just watching demos and pro matches. Another reason for me to get this project done for player who likes CS:GO and everything about it to analyze a lot of data and find possible patterns in it. This project might help me and others to see something you can't normally see with your eyes in the replay.

1.9 Approach

So in this project I want to parse CS:GO demos and to analyze which positions the players were playing and where were they dying or getting a lot of kills to try understand why they were so dominant on the map for such a long period of time. In this project I'm going to use csgo-demoparser and use it to get the position/coordinates of killer and the victim, then apply the coordinates grid to the map's top view and

mark the data I will get from parser. Then connect the two marks and analyze this new data and compare with other regular and pro teams demos. As I already mentioned there are a lot of maps but the starting point will be analyzing one of the matches played by regular players then analyze one match of pro team on the same map and then analyze this team's matches on the same map for last 2 years and see what we get as a result and what conclusion can we make based on it.

Chapter 2

Background Information

2.1 Valve Corporation – CS series creator

Valve Corporation is an American video game developer, publisher and digital distribution company. Valve developed Steam and a lot of popular games including Counter-Strike series.
Valve was founded in 1996 by Gabe Newell and Mike Harrington. When Valve's debut product Half-Life was released it claimed commercial success. In 2003 Valve launched Steam which became popular and accounted for around half of digital PC games sales by 2011. By 2012 Valve became most profitable company per employee in United States.

FIGURE 2.1: Valve Corporation logo

2.1.1 Foundation and HL

Valve was founded by former Microsoft employees Gabe Newell and Mike Harington.

Interesting fact: the corporation was founded on Newell's wedding day.

"Fruitfly Ensemble" and "Rhino Scar" were considered as alternatives names for company. Valve modified Quake engine into their new one called GoldSrc. It was used to develop and release their first product Half-Life in 1998. According to IGN in 2014, the history of FPS genre "breaks down pretty cleanly into pre-Half-Life and post-Half-Life eras."
The company released software development kit (SDK) for GoldSrc engine. It was used by regular users to be able to develop mods (games modifications). One of them being Counter-Strike became one of the most popular. Valve decided to hire developers to create standalone Counter-Strike game.

2.1.2 Re-incorporation, Source and Steam

In 2003 Valve was re-incorporated in Bellevue, Washington. Then in 2010 the office moved to larger location and then in 2016 Valve started renting nine-floor Lincoln Square Complex, doubling the size of their offices.
After Half-Life's success Valve worked on mods using their new Source engine and started working on Half-Life 2 using this same engine. Source engine was also used to create Team Fortress 2 from Team Fortress Classic. Team Fortress 2 and Portal were developed by student teams hired by Valve.
Besides developing games Valve developed Steam which is a video game distribution service. The idea behind creation of Steam was that Valve had to maintain patches for games like Counter-Strike so that all players were up-to-date. Steam was introduced in 2002 and was created by Valve themselves as other developers didn't won't to help them. Originally you could only get Valve's games on Steam but later on they allowed third-party companies to sell their products and they would be taking cut of their revenues for content delivery. Steam became the most popular way gamers use to acquire digital games. Steam is accounting up to 70% of all digital sales.

2.1.3 Games

Valve is a developer and publisher of popular single-player games such as Half-Life and Portal and multiplayer games like Counter-Strike, Team Fortress 2, Dota 2, Artifact, Left 4 Dead and Left 4 Dead 2. Valve also released VR (virtual reality) game Half-Life: Alyx on March 23, 2020 which is a spin-off of Half-Life series.

2.1.4 Steam

Valve announced Steam at the 2002 GDC (Game Developer Conference). It was launched in 2003 and was used for patches and updates for Valve's online games. By July 2014 there were 3400 games available on Steam. And by January 2018 there were over 150 million registered accounts. Valve announced that Steam had reached over 67 million monthly and 33 million daily active users on the platform in August 2017.

2.1.5 Pipeline Project

Pipeline Project was announced as an intern project in July 2013 by Valve to teach a group of 10 high school students how to create video game content and see if it's possible to train them with as they have minimal work experience. It was stated that the company is not good at "teaching people straight out of school".

2.2 Source (game engine)

Source is a 3D game engine developed by Valve and written in C++ programming language. It is being developed since June 2004. One of Source's main features is its modular base and flexibility. Also Source engine is known for its lip sync speech, technology of emotion expression and physics system. Source engine is constantly incrementally updated instead of having a version numbering scheme.

FIGURE 2.2: Source engine logo

2.2.1 Source Name

Source's engine predecessor is GoldSrc engine (also developed by Valve) which was a modified version of Quake Engine. One of Valve's employees explained that when Valve were releasing the Half-Life game they forked off the code to be both /$Goldsrc and /$src because they didn't have time to check the code and make changes and they couldn't risk it. Then at E3 (Electronic Entertainment Expo) when Valve showed Half-Life 2 for the first time they were referring to the "Source" engine instead of "GoldSource" and that name stuck. Slowly Source started replacing GoldSrc in Valve's projects.

2.2.2 Modularity

Source has modular base. This allows different systems to be represented as separate modules which can be updated independently. This opposes "version jumps" of Source's competitors. With use of Steam (a video game digital distribution service by Valve) Valve can make updates automatically among a lot of users. There was though a break in compatibility chain when 2 Valve's projects introduced new versions of the engine that required from developers to upgrade the code and in some cases the content because it couldn't be used to run older game and mods. But still these cases required less work to update its version than competing engines.

2.2.3 Source SDK

Source SDK is a software development kit to develop assets for games used by Valve. It has command-line programs and GUI-based programs designed for special functions within asset pipeline and handling complex functions.
Hammer Editor, Model Viewer, and Face Poser create a package of Source SDK. Hammer Editor is used to create maps using SDK's rendering and compiling tools and the binary space partitioning method. The Model Viewer allows users to view and develop models. And Face Poser is the tool which is used to edit facial expressions, gestures, movements lip sync and preview how all these actions will look in the game's engine scene.

2.2.4 Academic Papers

Valve sometimes produce academic papers for events and publications explaining aspects of development of Source engine.

2.3 CS:GO demo parser

CS:GO demo parser (csgo-demoparser) is a library for parsing Counter-Strike: Global Offensive demo files.
It processes the file and events are emitted for which callbacks can be registered.
It's written in Python.

2.3.1 How does it work?

It parses .dem files created in CS:GO and gets messages from game events. The list of game events can be found here: CS:GO Events List

Chapter 3

Related Works

3.1 Moneyballing 32k rounds of Counter Strike by Mads Bønding

The author of this article didn't post any github repository and I didn't manage to find the code to it. But what I can tell is that as he said he used the data collected by skihikingkevin which can be found here: skihikingkevin's CS:GO data. So what he does in this work is analysis of such things:

3.1.1 Weapon of choice

Here Mads analyzes the data and visualizes which weapons do players from Gold Nova 1 rank to Legendary Eagle Master. First he only shows weapon's choices on normal buy rounds. And in the end we can see that most used weapons are AK47, M4A4 and AWP on all analyzed ranks. Then Mads investigates force rounds buys and we can see that on lower ranks players use SMG's like UMP, P90, MP7, AK-47 rifle and Desert Eagle as pistol. Higher ranks use AK-47, Desert Eagle, Five-Seven and UMP.

3.1.2 Utility Usage

In this part Mads shows that higher ranks players starting with DMG are using more grenades that lower ranks players. As he says and shows on the figure we can see that higher ranks use approximately 1 more grenade of each type except Molotov/Incendiary and Decoy than lower ranks players.
The next figure shows the round win percentage based on CTs and Ts nade amount difference per round. As we can clearly see the team that uses 5 more grenades more than the opposite team wins 75% of the rounds. But as Mads says this might not be a good way to show the utility usage and gives an example when on of your teammate or you throws 2 flashbangs and flashes his team twice and then the opponent would throw just one incendiary to stop your team from pushing the bombsite and that would count like a superior usage on your part (because the amount of flashes is greater than amount of incendiary).

3.1.3 Mechanichal Skill

Coming to the hitting of shots. Headshots almost always result in instant kill. All weapons can kill you with headshot if don't have a helmet but if you do there are some weapons that need 2 headshot hits to kill. After playing a bit of CS you would probably think higher ranks are hitting the shots better than low ranks. But that's

not really true. As we can see on Mads's figure high ranks have only 4% more headshot hits than low ranks. All other parts of body hits have pretty much the same percentage on any of observed ranks.

3.1.4 Map Insights

In this part we can observe which positions on the map are likely to be smoked by Ts for all those ranks which Mads is analyzing on Mirage map. So the common positions that are getting smoked are: Stairs, Jungle, CT, Window, Top mid, Short. We can see that all the smokes are focused on mid and A bombsite areas of the map with a very little amount of smokes on B bombsite (actual for all the ranks).

Next Mads shows where were the smokes thrown from and where they landed. We can see that most common smokes on MGE ranks are:

- From T spawn to Window and Stairs for Ts
- From CT to T ramp, from Stairs to Palace on A and from Kitchen Window and Car to B apps for CTs

3.1.5 Executes

Execute of bombsite is when the team decides to enter one of 2 bombsites with the help of utility (smokes, flashes and molotovs, sometimes HEs).
Mads shows that execute with 3 smokes on A is the most common tactic for Ts on mirage. The 3 smokes are Stairs, Jungle and CT. Just these 3 smokes cut off CTs vision and the Ts can enter the bombsite under cover for 15 seconds. After that time the smokes will fade and it will be very hard to enter the bombsite. From 146 instances of this triple smoke tactic that Mads analyzed 60% of the time the execute results in A site bomplant.

3.1.6 Effective use of HE grenades

Here Mads analyzed in which spots on the map HE deal the most damage. So he split 20k of HE nades into 25 clusters and then he checked which clusters have the highest hitrate (nades that do damage). So here are the results.

Most hitrate spots:

- Default/Ninja – 49.2% hitrate
- Tetris/T ramp – 43.9% hitrate
- Top mid – 42.8% hitrate
- Firebox – 42.6% hitrate
- T spawn – 49.2% hitrate

Most average damage spots:

- Apps – 38 avg. damage
- Palace – 30 avg. damage
- Tetris/T ramp – 30 avg. damage

- T ramp stairs – 30 avg. damage
- T spawn/T ramp – 29 avg. damage

Mads also gave the timings at which you should use the HEs but that's not really relevant info because throwing HEs is a situational thing.

3.1.7 Kills and Deaths

And in the final part of this article Mads shows where people kill and die the most on Mirage.
We can see on figures that on all ranks people kill and die on the same spots. Those spots are: Connector, Palace, Apps, Stairs, Jungle, Ramp, CT, Tetris and Window. And after that he gives more interesting ideas to explore.

3.2 Finding classic smokes by T-side on mirage

Here is the link for this project: billfreeman44's project This project was done by Bill Freeman in Python. He used K-means to find common nades on mirage.
He started by reading the data about grenades in matchmaking specifying smoke grenade for T side on Mirage map. Then he used game to image coordinate converter to get the positions of Ts and their smokes landing positions. Bill dropped old (pre-converted) data and plotted the result on Mirage image. Then he used the K-means algorithm to create 25 clusters of smoke throwers to show the most common positions for Ts to throw smoke from.
And in the end as well as Mads he gave some ideas what to explore more.

3.3 CS:GO stats trackers

There are 3 most popular sites to check your CS:GO stats when you search for "csgo stats" in Google:

- csgostats.gg
- csgo-stats.com
- tracekr.gg

They do pretty much the same. All 3 of these sites can tell:

- your K/D ratio
- actual hours in matches (not just in-game)
- HS
- deaths count
- winrate
- MVP count
- headshots count
- score (sum of points for all your games)

- bombs planted and defused count
- shots accuracy
- wins and losses count and a lot more statistics of this kind

3.4 Leetify

But there is one site I discovered recently. It's leetify.com.

3.4.1 How it works?

You give it match history sharing code, so it can analyze your matches. The good thing is you don't need to give a new code every time. After you played a new game it will be uploaded on the site automatically and you will be able to see all the info. After it loads and analyzes your matches it gives you the stats you actually need to become better. So this site gives you the ability to check every match separately.

You can basically see everything. For example:

- when the match was played, which map, what was the score
- you can download the game from the site (no need to start CS and getting a sharing code to do it)
- general stats like (KDA, HS kills count, HS%, players ranks, players steam profiles)
- activity breakdown (shost fired, dmg done, dmg/shot, time alive, thrown grenades count)
- aim group (HS accuracy, general accuracy (all shots), spray accuracy, counter-strafing %, crosshair placement, time to damage)
- utility group (flash assists, enemies flashed, teammates flashed, dmg per HE, unused utility on death, CT smokes that stopped rush %)

You can also check visualizations of your utility usage and aim for certain period of time which you will choose. Also the site will show you your stats compared to average stats on your rank so you can see the difference and know how to improve, it will even tell you how to improve and offer video links so it will be easier for you to understand. In the "Aim" tab the site will show you the spray patterns of different weapons and how does your spray pattern look compared to the original one. Here you will also find some stats related to aim, there will be the option to check details of any of those stats for all weapons and how to improve them if you have lower % compared to average for your rank. As a result it can be said that this is the most advanced CS:GO tracker site which can help player improve their gameplay for sure.

Chapter 4

Solution

4.1 Where this project's idea came from?

At the beginning of thesis I actually had a completely different idea for my project which was declined by Dean's Deputy and wasn't related to CS:GO and Python, but to web development. So it was quite hard time getting used to Python.

4.2 Start

First the goal was to collect from CS:GO demos and analyze it in the way that as result I would get some kind of patterns, parameters or dependencies in this data.
How did I try to collect data? So I used csgo-demoparser which has a limited lists of in-game events which return some messages/values. That means it is not possible to get all the data that is needed for this goal. For example data like what was the map played, start and end time, duration i.e. match metadata. Along with the demo file there is .dem.info file saved which might contains this information. So why not use it? Because if you want to analyze pro match which can be downloaded from hltv.org as .zip file contains only .dem file and nothing more. Besides that it's very hard to think of some features/factors which could describe why one team plays better than other that you wouldn't see from demo. So the conclusion is that there is not much point of moving in this direction.

4.3 Direction change and approach

Now as I showed what people did in Related Works I decided to not really analyze data, but to use it to create some kind of basic demo viewer. What I did was taking events which I could get use of and first analyzed what they return and how to work with that data.
After figuring out the events I thought of what I want to get as result. And knowing there are already a lot of different things done in CS:GO world I came to the idea of basic demo viewer.

4.4 Basic demo viewer and economy research

This idea attracted me because I didn't see anything like that on internet. Basic demo viewer is a simple system which allows a user to upload demo to the server and get round by round plots which user can change. Also there are some more plots which show demo by team side (T or CT) and team number (Team 1 and Team 2).

After that I thought it would be interesting to also research some every round economy and plot it to see round by round both sides money situation displaying it with levels from 1 to 5.
Steps to see demo by rounds:

1. Run server
2. Upload demo to server
3. Wait until server works with demo
4. See the results on page

4.5 Tools used

For this project implementation I used Python as programming language and Flask as web framework which is also written in Python. For plotting data Matplotlib library was used.

4.5.1 Python

This is interpreted object-oriented programming language. It has dynamic typing semantic and high-level data structures which makes it a good programming language for fast software development. Python is modular and allows reusage of code. As already mentioned it's a interpreted language which means it has the interpreter which executes code directly line by line and stops the script execution is case any error occurs. Also Python shows only one error even if there are multiple which allows easier debugging.

```
n = int(input('Type a number, and its factorial will be printed: '))

if n < 0:
    raise ValueError('You must enter a positive integer')

fact = 1
i = 2
while i <= n:
    fact *= i
    i += 1

print(fact)
```

FIGURE 4.1: Python code example (factorial function)

Python advantages:

- Clear and easy syntax
- Portability
- Possibility of using a command line interpreter for solving easy tasks
- Very useful for solving mathematics tasks

- Open source code which allows other users to modify it

Languages like ABC, Modula-3, C, C++, Smalltalk, Lisp, Fortran, Miranda, Java and Icon had influence on Python which collected some of the key features which it has now. Python is portable and works almost on every platform. Of course with the flow of time some platform stop to support newer Python versions but they can still use older versions. It has an interactive mode which allows even experienced programmers to test parts of code before using them in main program.

Python disadvantages:

- Slow code execution (compensates with general development time and more compact code than same code on Java or C++)
- No possibility to modify built-in classes
- GIL (Global Interpreter Lock)

Overall Python is very popular language which is used in a big amount of projects mostly for building extensions and apps integrations or as main language.

4.5.2 Matplotlib

It's a library mostly used for 2D data visualizations which can be saved as images. These images can be used in things like computer graphics, publications or web applications. It's built based on OOP principles.

```
import numpy as np
import matplotlib.pyplot as plt

# Fixing random state for reproducibility
np.random.seed(19680801)


N = 50
x = np.random.rand(N)
y = np.random.rand(N)
colors = np.random.rand(N)
area = (30 * np.random.rand(N))**2  # 0 to 15 point radii

plt.scatter(x, y, s=area, c=colors, alpha=0.5)
plt.show()
```

FIGURE 4.2: Matplotlib code example

Possibilities of Matplotlib:

- Line plot
- Scatter plot
- Bar plot
- Histogram

User can add axes labels and title to plots. Matplotlib also allows to create animated images.

Here are supported image formats:

- EPS
- EMF
- JPEG
- PDF
- PNG
- PostScript
- RGBA
- SVG
- SVGZ
- TIFF

Pylab interface allows users which worked with MATLAB before to easily use Matplotlib. This library has some advantages over MATLAB: SVG support, open source software and it's free.

4.5.3 Flask

Flask is a web framework written in Python as mentioned before. It's based on two components. First being Werkzeug (realization of software objects for request, response and utility functions) and second being Jinja2 (web template engine).
Flask was created by group of Python enthusiasts in 2004 being originally just an idea for April Fool's which became popular and made into serious project.

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello, World!'
```

FIGURE 4.3: Flask app code example

Flask's main features:

- Development server and debugger
- Jinja templating
- Secure cookies support
- Unicode based
- Detailed documentation
- Extensions support for extending features

In Python Developers Survey 2018 Flask was voted the most popular web framework.

4.6 Structure

Some files might look strange. That's because as I mentioned before I was writing it for analyzing data and figuring out some features/factors that would tell why one team plays better than another and as my time was limited I decided not to change some functions or code fragments. For example I would read multiple demos and collect data so I was reading the whole directory but now I have only one file at a time but I still use the code for reading the directory and adding all its files to list.

get_coordinates.py – is a Python script where all needed event's messages are added to list including attackers and victims positions. This data then is converted to Pandas Dataframe and saved as .csv file. Almost all the data was created manually meaning that for some crucial data there was no events so I had to think other ways to get it. It is worth mentioning that demo in-game coordinates and coordinates on image are different so there are two functions for converting X and Y coordinates for certain map.

```
154  def pointx_to_resolutionx(xinput, startX = -2486, endX = 2127, resX = 1024):
155      sizeX = endX - startX
156      if startX < 0:
157          xinput += startX * (-1.0)
158      else:
159          xinput += startX
160      xoutput = float((xinput / abs(sizeX)) * resX)
161      return xoutput
162
163  def pointy_to_resolutiony(yinput, startY = -1150, endY = 3455, resY = 1024):
164      sizeY = endY - startY
165      if startY < 0:
166          yinput += startY * (-1.0)
167      else:
168          yinput += startY
169      youtput = float((yinput / abs(sizeY)) * resY)
170      return resY - youtput - 10
171
```

FIGURE 4.4: In-game to on-image coordinates converting functions

One problem here is that those startX, startY, endX and endY are unique to each map in CS:GO. resX and resY are dimensions of image to which you want to convert the coordinates.

split_demos_to_images.py – Python script which reads .csv data about demo and plots it for different parameters.

```python
def plot_image_by_rounds(dataframe, csv_name):
    last_idx = 0

    for j in range(dataframe['round'].nunique()):
        fig, ax = plt.subplots(figsize = (20, 20))
        current_demo_round = dataframe.loc[dataframe['round'] == j + 1]

        plt.title("Round outcome: " + current_demo_round.winner_team.to_list()[0], fontsize = 30)

        ax.scatter(current_demo_round.att_map_x, current_demo_round.att_map_y, alpha = 1, c = 'b', s = 19**2, edgecolors='black')
        ax.scatter(current_demo_round.vic_map_x, current_demo_round.vic_map_y, alpha = 1, c = 'r', s = 19**2, edgecolors='black')

        for i in range(len(current_demo_round)):
            xy_a = current_demo_round.att_map_x[i + last_idx], current_demo_round.att_map_y[i + last_idx]
            xy_b = current_demo_round.vic_map_x[i + last_idx], current_demo_round.vic_map_y[i + last_idx]

            con = ConnectionPatch(xy_a, xy_b, coordsA = "data", coordsB = "data",
                                  arrowstyle="-|>", shrinkA=5, shrinkB=5,
                                  mutation_scale=20, fc="w")
            ax.add_artist(con)

        last_idx += len(current_demo_round)

        ax.imshow(im)

        plt.savefig('./static/images_by_rounds/{}_round_{}.jpg'.format(csv_name, j + 1))
        plt.close()
```

FIGURE 4.5: Function which plots demo round by round

Example of function which plots every round of the game with lines connecting attacker and victim respectively.
Right now it only works with Dust 2 map from CS:GO but in future I'm planning to do map choice when uploading demo.

server.py – Flask server which contains code that is responsible for file uploads and processing those files data in order to show the plots on server where you can choose different rounds and different plots in general (for example for sides or teams).

```python
@app.route('/demo_by_rounds')
def demo_dropdown():
    def sort_nicely(l):
        convert = lambda text: int(text) if text.isdigit() else text
        alphanum_key = lambda key: [convert(c) for c in re.split('([0-9]+)', key)]
        l.sort(key = alphanum_key)
        return l
    get_coordinates()
    res_images()
    round_numbers = return_rnd_numbers()
    images_names = []

    for file in os.listdir("./static/images_by_rounds"):
        if file.endswith(".jpg"):
            images_names.append(file)
    # images_names = os.listdir("./images_by_rounds")
    sorted_names = sort_nicely(images_names)
    return render_template('demo_viewer.html', round_nums = round_numbers, images = sorted_names)
```

FIGURE 4.6: Example of Flask server function

economy_research.py – Script which calculates buy level in each round for both T and CT sides.

```
8
9   def csv_to_df(list_of_csv):
10      for csv in list_of_csv:
11          df = pd.read_csv(csv, index_col = 0)
12      return df
13
14  def economy(dataframe):
15      t_buy_level = 0
16      ct_buy_level = 0
17      counter = 1
18      win_streak = 0
19      prev_winner = ''
20
21      for i in range(len(res_list[0])):
22          if len(res_list[0][i]) == 1:
23              rnd_winner = res_list[0][i][0].get("winner team")
24              if counter == 1 or counter == 16:
25                  win_streak = 0
26                  if rnd_winner:
27                      win_streak += 1
28                  ct_buy_level = 1
29                  t_buy_level = 1
30
31              if counter == 2 or counter == 17:
32                  if rnd_winner == "CTs win":
33                      if rnd_winner == prev_winner:
34                          win_streak += 1
35                          ct_buy_level = 2.5
36                          t_buy_level = 1.5
```

FIGURE 4.7: Example of economy calculating function

economy_to_plot.py – Script which reads .csv data about the demo and plots economy levels of both sides on separate plots.

```
13  def plot_t_economy(dataframe, csv_name):
14      # ax = plt.gca()
15      # dataframe.plot(kind='line', x='round', y='t_buy_level', ax=ax)
16      dataframe.set_index('round')['t_buy_level'].plot()
17      plt.xlabel("Round", labelpad=10)
18      plt.ylabel("Buy Level", labelpad=15)
19      plt.title("T economy level by rounds", y=1.02, fontsize=22)
20      plt.xticks(dataframe['round'].unique(), rotation=90)
21      plt.savefig('./static/t_economy_plot/{}_t_economy.jpg'.format(csv_name))
22      plt.close()
23
24  def plot_ct_economy(dataframe, csv_name):
25      # ax = plt.gca()
26      # dataframe.plot(kind='line', x='round', y='ct_buy_level', ax=ax)
27      dataframe.set_index('round')['ct_buy_level'].plot()
28      plt.xlabel("Round", labelpad=10)
29      plt.ylabel("Buy Level", labelpad=15)
30      plt.title("CT economy level by rounds", y=1.02, fontsize=22)
31      plt.xticks(dataframe['round'].unique(), rotation=90)
32      plt.savefig('./static/ct_economy_plot/{}_ct_economy.jpg'.format(csv_name))
33      plt.close()
```

FIGURE 4.8: Example of economy plotting function

4.7 Results

4.7.1 Demo results

Results which I achieved can be seen on the following figures.

FIGURE 4.9: Demo by rounds with select of any round of the game

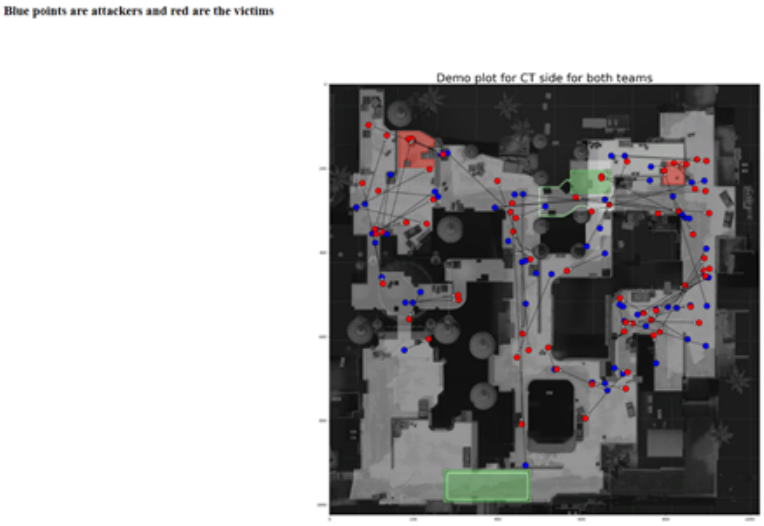

FIGURE 4.10: CT side demo view

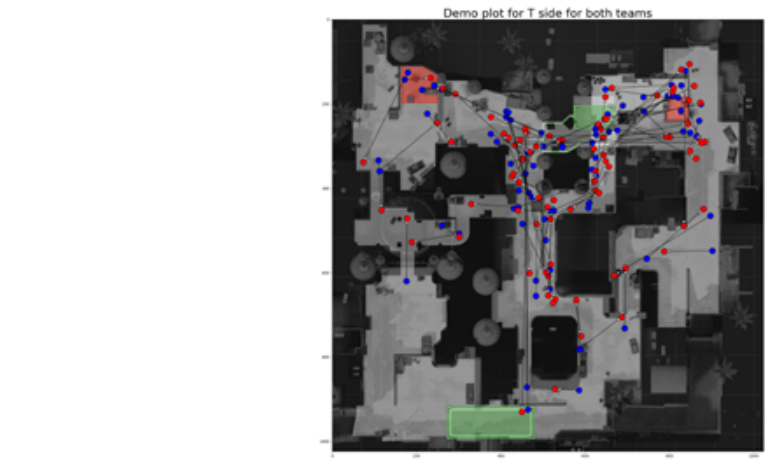

FIGURE 4.11: T side demo view

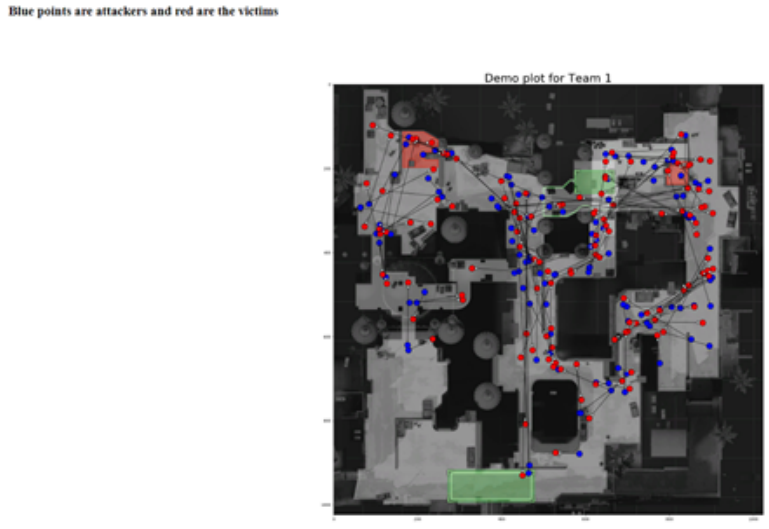

FIGURE 4.12: Team 1 demo view

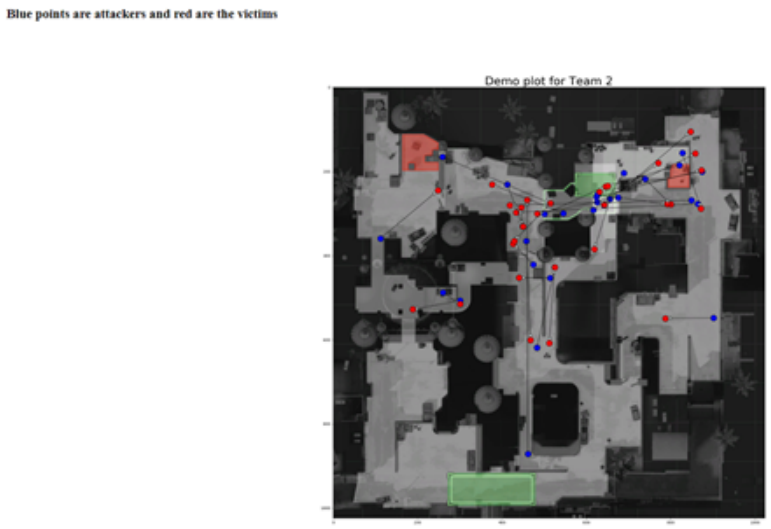

FIGURE 4.13: Team 2 demo view

4.7.2 Economy results

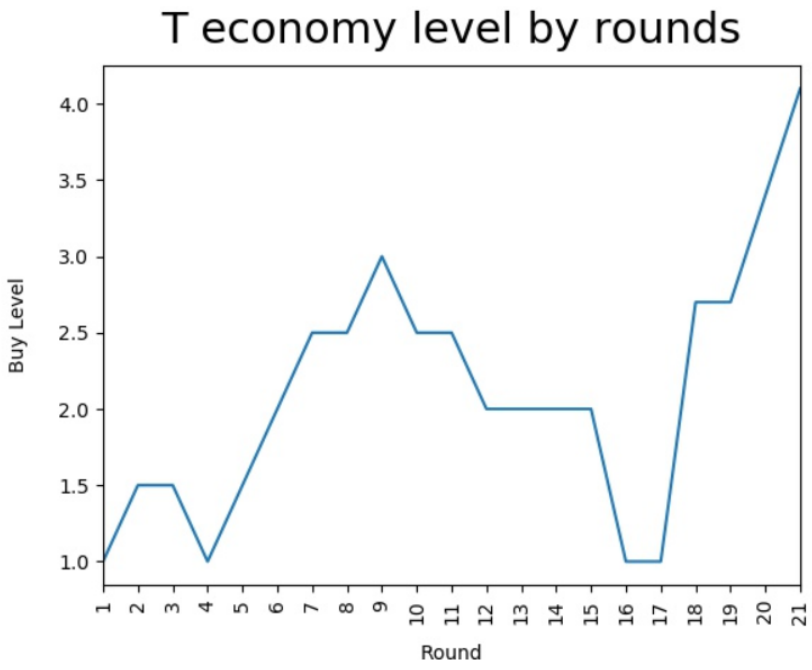

FIGURE 4.14: T buy levels example

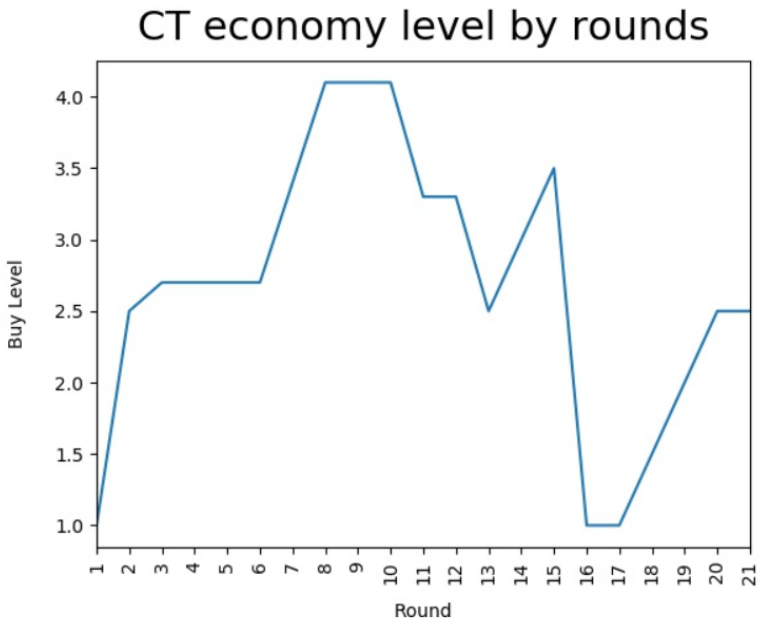

FIGURE 4.15: CT buy levels example

4.8 Conclusion

I would like to say that I got to my goal in this project. But I had to change direction and it's still a really interesting project for me as player and developer to improve this project and find ways to collect more data about the matches and use it to create something that might become popular in CS:GO pro scene and used by many pros and team coaches.

4.9 What are the problems and what to do next

The main problem as I mentioned is that there is for example no CS:GO API or something like this to help with data collection.

What are my next steps:

- To try collect the data I needed for my original idea with CS:GO positioning analysis
- Add map choice for demo
- Split map by spots and analyze K/D (kill/death) ratio on them
- Analyze demos with trade kills (it's the situation when you "trade" your teammate when he is killed by enemy) and without to see % difference in winrate
- How good are the players throwing different nades (especially smokes and flashes). See the impact. Do they land correctly?

There is for sure much more interesting stuff to do which I'm willing to realize with my teammates.

Bibliography

[1] Introduction to CS:GO, https://dotesports.com/counter-strike/news/beginners-in-depth-introduction-to-csgo-and-its-economy-system-7770

[2] Introduction to Python, https://www.python.org/about/gettingstarted/

[3] Flask documentation, https://flask.palletsprojects.com/en/1.1.x/

[4] Matplotlib documentation, https://matplotlib.org/3.2.1/contents.html

[5] Mads Bønding's CS:GO project, https://medium.com/@madsbnding/moneyballing-32-000-rounds-of-counter-strike-9cc4a9f493ef

[6] CS:GO positioning analysis by mdolr, https://www.kaggle.com/mdolres/cs-go-positionning-analysis/data

[7] Billfreeman44's finding classic smokes for T side on Mirage, https://www.kaggle.com/billfreeman44/finding-classic-smokes-by-t-side-on-mirage

[8] CS:GO Events list and descriptions, https://wiki.alliedmods.net/Counter-Strike:_Global_Offensive_Events

[9] CS:GO demoparser, https://github.com/ibm-dev-incubator/demoparser/tree/5e04604f421a0378b8408e32b5ce36adc7776b37

[10] Valve Corporation, https://en.wikipedia.org/wiki/Valve_Corporation

[11] Source Engine, https://en.wikipedia.org/wiki/Source_(game_engine)