Robotron: 2084 inspired Game

Written in PyGame with MVC architecture

John Montgomery
Candidate Number: 5199
Centre Name: Kimberley College Sixth Form
Centre Number: 15125
Qualification: AQA 7517 (A–Level Computer Science)

Supervisor: B. Harris

2020-2022

Contents

List of Figures

Chapter 1

Abstract

In essence the project is an implementation of Robotron in PyGame, using Model-View-Controller, with a Flask based high scores bored. The main content of the code is in the game itself, with flask acting only as an API. This allows for shared usage of the route by a static web page, and by the PyGame code itself. The webpage is simply served off as static, where JS is able to communicate with the API to retrieve the information needed. The database used is Postgres.

Chapter 2

Analysis

2.1 What is MVC?

Model-View-Controller plays a large part in the project, the diagram [Figure 2.1] shows the main way that MVC works. It isolates the components of the game into 3 main components. The View, which is the screen, or what the user will see. The controller, which is where the user interacts with the game, in this case it is the interaction with the keyboard. The model, which is the part the user never interacts with, and stores the state of the game and current information about it.

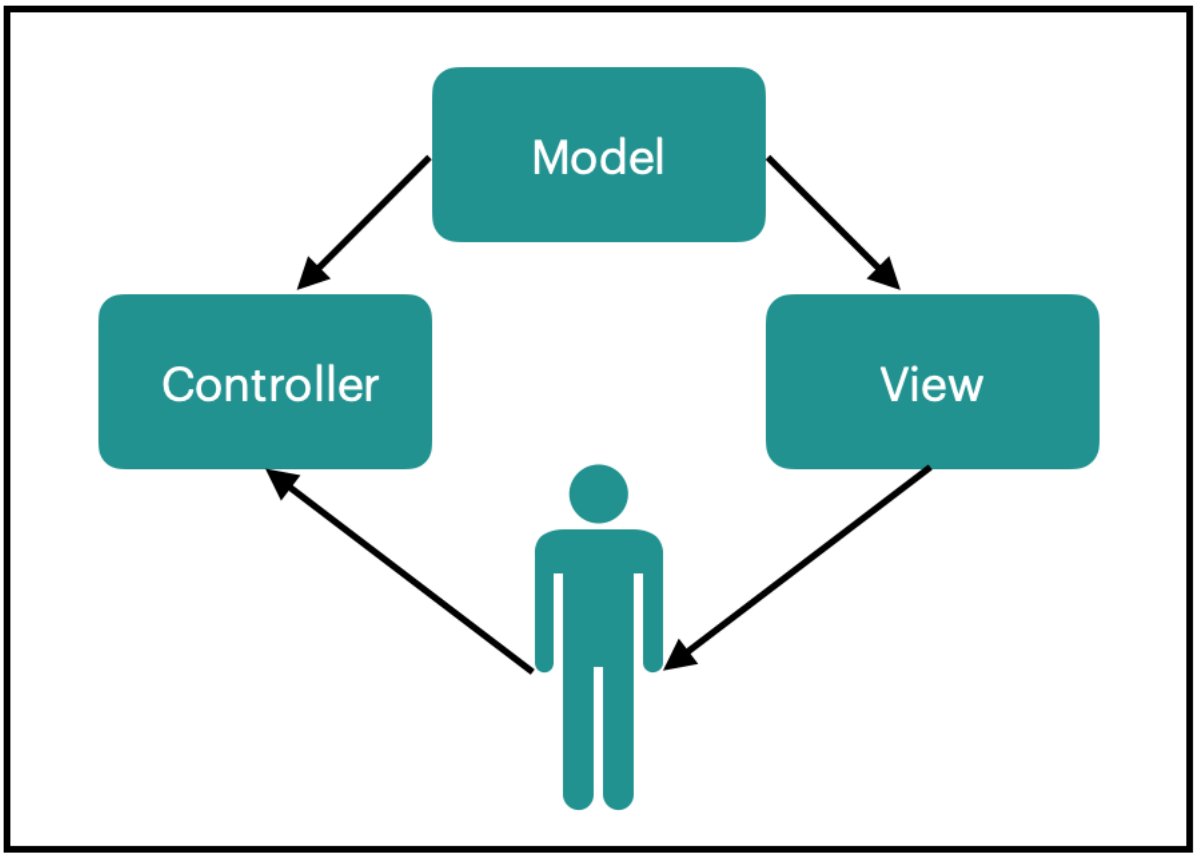

Figure 2.1: A diagram showing the MVC architecture

There are many benefits to this set up, for example, it will easily allow me to swap out what controller is used. If desired, it is much simpler to replace the keyboard as the human interface, and replace it with a game controller. Even more useful may be the ability to remove

the controller and view entirely, allowing for a streamlined game which an AI could learn how to play. This flexibility, along with ease of programming is what drew me to use MVC for the game.

Another important information is the way information travels between the 3 sections. This is done with events, and an event manager is responsible for maintaining the sending and receiving of events through the system. A similarly important section is the States, and state machine, which controls the current 'state' the game is in, that is to say what level is being played, or what screens should be shown, such as a loading or help screen.

2.2 The Game

"Robotron: 2084" was released in 1982 by Williams Electronics. It was revolutionary as a dual stick shooter, was high energy and loved by many. This is important to capture into the game, where I want it to have a similar feeling to the original game, with some modern twists.

The game is about a species of 'Robotrons' created by humans in the year 2084, after realising their failings and created an advanced species. The goal is to save the humans (Mommies, Daddies and Mikeys), whilst fighting the robots, which have many kinds. The most basic are electrodes, which are static obstacles that kill on contact, but can be shot by players. The other basic enemy is the grunt, which is simply a basic soldier, which kills on contact, but moves towards the player. There are some other robots that will be talked about and implemented later, but the details about them are less important.

2.3 Limitations

The dual stick shooter nature means the player uses one joystick to move, and one joystick to shoot. This is difficult to implement well with a keyboard, but a simple setup which I am using is having WASD to move, and IJKL to shoot. Holding 2 keys diagonally at the same time will result it movement in an angle, allowing for shooting in 8 directions, and moving in 8 too.

Robotron is a fast fast game, I had to slow it down slightly in order to make it more playable on my laptop, and so it does feel somewhat different to the original. However by slowing it as I have I have made it a much smoother game to play.

2.4 Objectives

1. Create basic playing ability
    (a) Player can move in 8 directions
    (b) Player can shoot in 8 directions
    (c) Players animation is correct for direction of travel
2. Create basic enemies
    (a) Enemy is spawned in random position
    (b) Enemy can move
    (c) Enemy is animated
    (d) Enemy kills players
3. Create Loading Screens
    (a) Fuzzy loading screen
    (b) 'All test' screen

   (c) Home Screen

4. Create levels and transitions
   (a) Player moves between levels
   (b) Level transitions
   (c) Player is invincible on load
5. Create the API
6. Create login system
   (a) Basic API sign up works
   (b) GUI interactions with PyGame
7. High Scores
   (a) Top 10
   (b) Player Search
8. Create sounds with Game
9. Create scoring and score counter
10. Create a life counter
11. Automate testing on API and basic functions in PyGame

2.5 Design and Inspiration

The design for all the game is heavily taken from the original game. I used many places to research this, but below is a selection of screenshots and videos which were used in the creation of the game.

- https://www.youtube.com/watch?v=ccltMtkFBSI
- https://www.youtube.com/watch?v=aOVA2Axxfdk

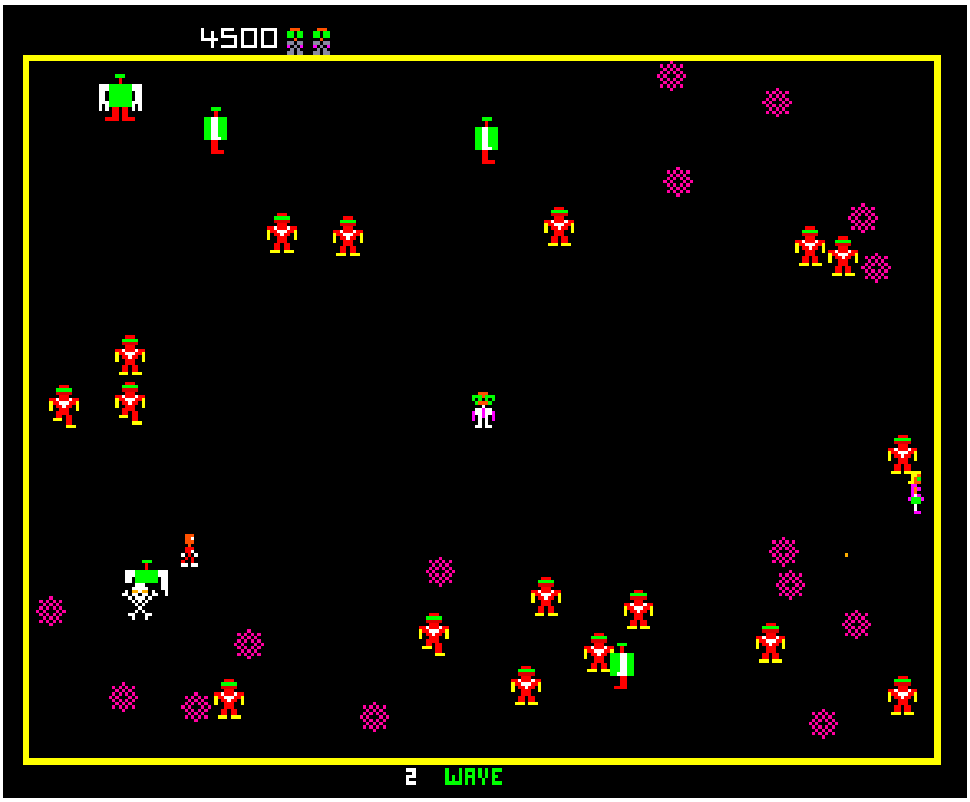

Figure 2.2: Screen from original game - https://arcadeblogger.com/2020/06/27/the-development-of-robotron/

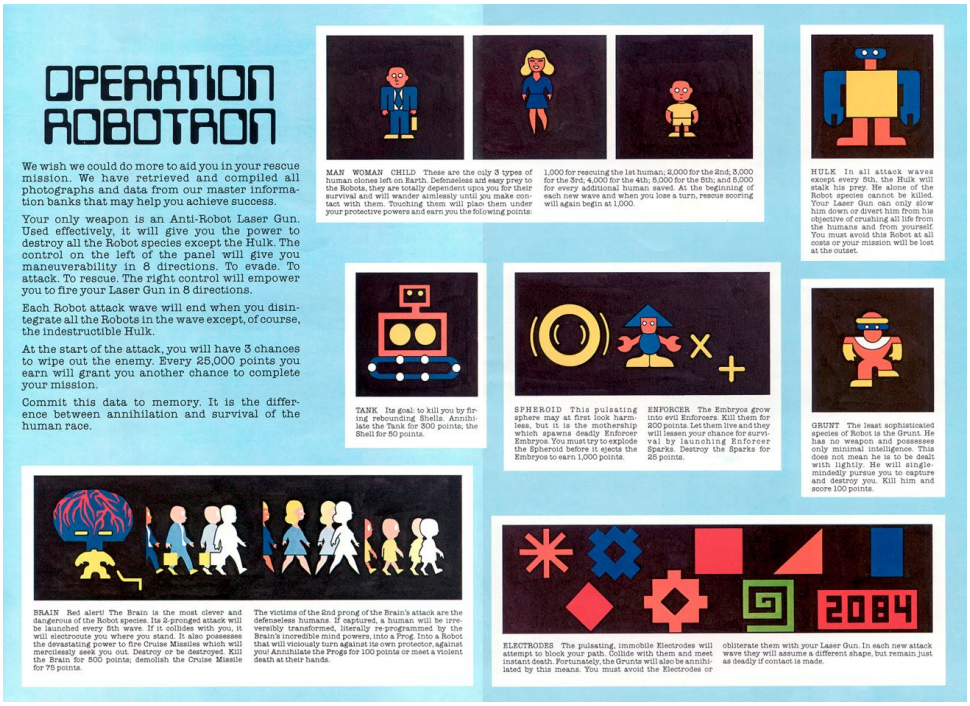

Figure 2.3: Advertising Material - https://arcadeblogger.com/2020/06/27/the-development-of-robotron/

Chapter 3

Documented Design

The main design aspect is the MVC architecture and how it forms the basis of the game. Fig 1, from the analysis section, gave a very brief, high level and non technical view of MVC. In this section I will go into more detail about my own implementation, and how it works in greater detail. This section also details the database on the web side, the API, the technical setup of the servers, the data structures and HCI designs.

3.1 MVC in practice

In the analysis section I gave a very high level overview of MVC, this part will detail further into my design on its implementation in python. The first main, basic components of MVC are of course, the model, the view, and the controller. Figure 3.1 shows the 3 classes diagrams for each of the implementations of these in python.

On top of these key features, there's also a range of other important cogs in the system. One of the most important, to allow for the communication between the M, V and C are Events, and an event manager. A Sample of events, and the event manager is given in Fig fig:events.

The other key class is the state machine. Each state is not given its own class, rather there is a constant number which is attributed to a given state. The states are used for the larger changes in the program and events are for the smaller interactions, and ticks.

In order to run through a basic idea of what happens when the program is run, I have created a step by step flowchart. This flowchart [Fig 9] is a gross oversimplification, but works as a high level description of what it is my code is doing when executed.

3.2 Boids

I have decided to implement a boids flocking algorithm into the game, this is a mathematical approach to natural flocking behaviour, and whilst this is not the 'AI' used by the robots in the original game (this was closed source, or at least, i have not found it), it does work quite well. Essentially there are 3 rules:

- move towards the centre of mass of the flock - match velocities with the flock - avoid collisions

in order to make them flock towards the player a 4th rule is added such that, in every iteration, the flock moves slightly closer to the player. This boids algorithm is much better than my original method, which essentially only implemented rule 4, and would get too close to the player and stack.

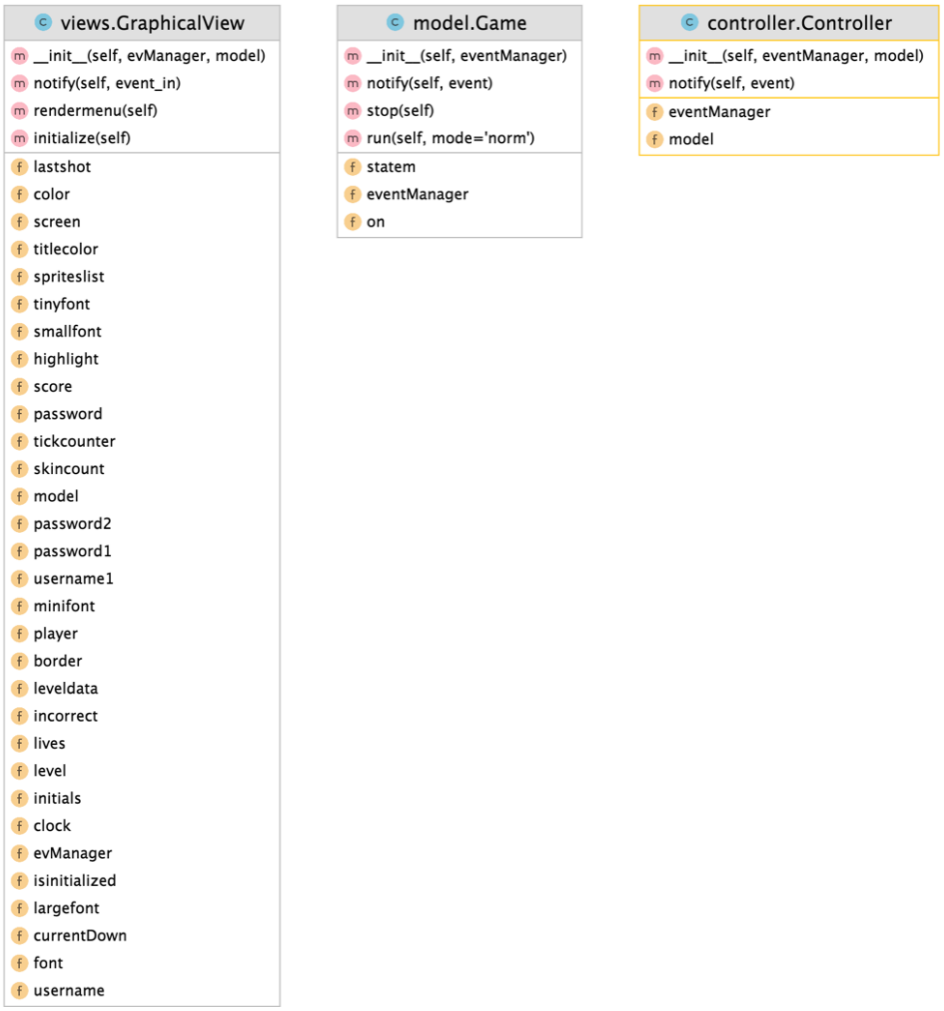

Figure 3.1: Class diagram

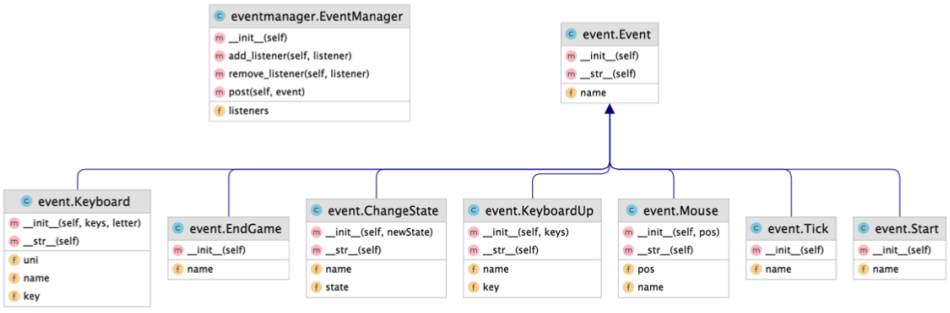

Figure 3.2: Class diagram

3.3 Database

This section will show the database design and set up, and explain some of the SQL used in the program. Fig 10 shows the database diagram.

[TODO - Database diagram]

There are 3 tables, scores, users and tokens. The scores database has 2 fields which store the users ID and their Score for a given game. The Users table stores the users info, such as emails, password hashes, etc, and then the tokens database is used to store validated tokens (with time limits) which are used to validate the GUI and avoids needing to login to the the

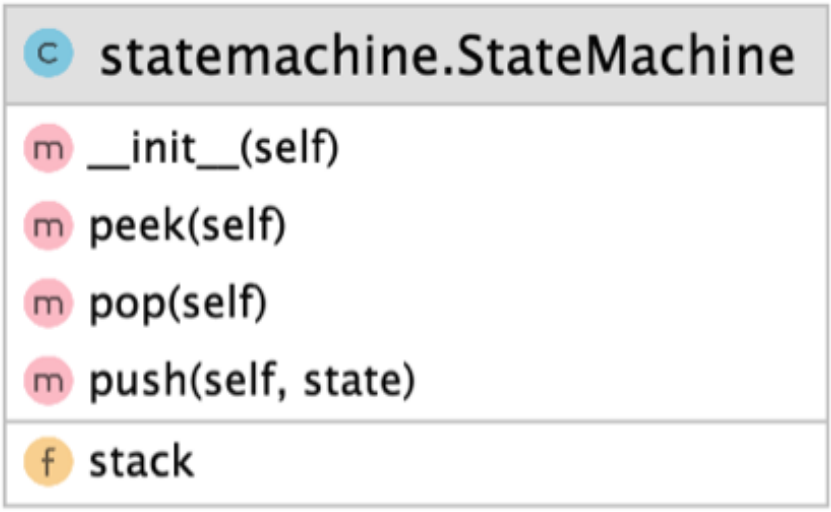

Figure 3.3: Class diagram

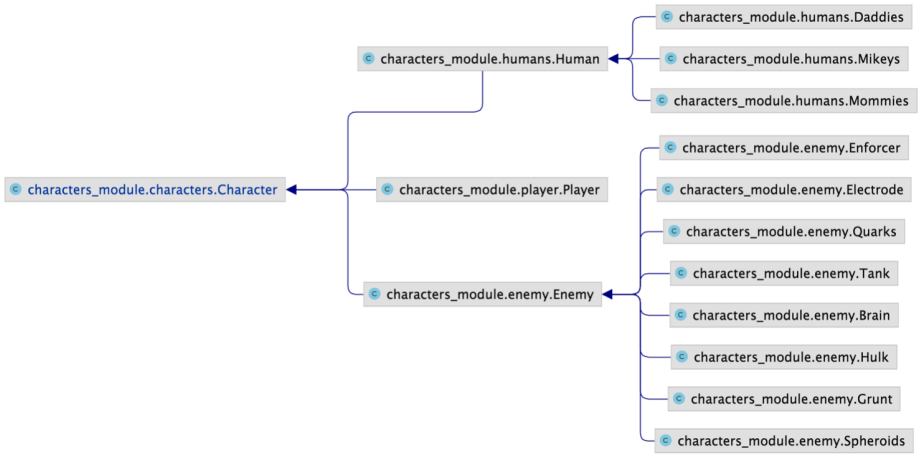

Figure 3.4: Class diagram of characters

program every time the game is run. Fig 11 shows the process of creating the tokens.

3.4 The API

The leaderboard contains only 6 routes, as these were all that are necessary, the details for the routes are detailed in the table below.

3.5 The Server Setup

Fig 12 shows the set up the server is in. All using AWS, there is an RDS Postgres database, and EC2 instance (this is the server running the actual flask) and then an S3 bucket to handle sending the static files. It may also be possible to use NGINX or Apache to serve and handle

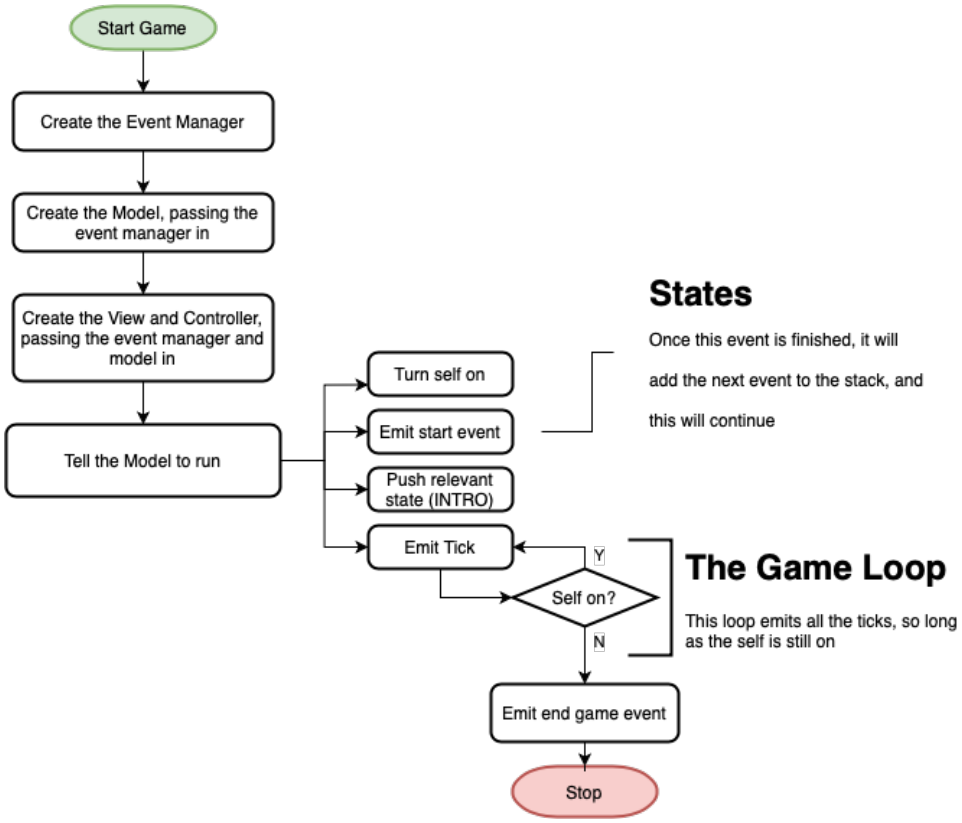

Figure 3.5: Flowchart of MVC

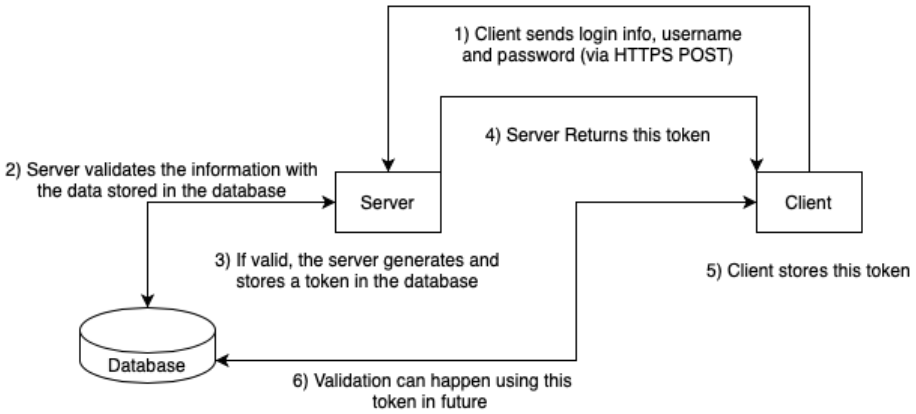

Figure 3.6: How tokens are generated

ROUTE	METHOD	DESCRIPTION
/leaderboard	GET	Returns JSON of top 10 users (initials + scores) in Database
/user/userid	GET	Returns JSON of top score
/username/userid	GET	Returns ID of given username
/login	POST	Logs in a user, sends token, or logs user in with token
/addscore	POST	Adds a score, given score and a token
/adduser	POST	Adds a user to the database

the API. This system may end up being better, so my current architecture could change.

3.6 Security

Because the database and client handles personal details like email and passwords, there needs to be a thought to security. First off, there is an enforcement of passwords and a strong policy. Users passwords will need to be 8 characters, with 1 special, and my plan is to check them against a list of common passwords (rocky.txt) using hashes. For this I will probably use MD5, or something even faster. However it is important to avoid these fast algorithms when hashing passwords for storage. As such, passwords will undergo key derivation through bcrypt, an algorithm which not only salts, but performs many rounds of hashing. I could implement a similar algorithm using the basic functions like SHA, but rolling your own crypto is never good, so its going to be done with bcrypt, as this is essentially the best option available, and more than secure enough.

To help further security, HTTPS is being used for all the sending and receiving of data, this avoids man in the middle attacks of the data as it gets sent over the internet.

Chapter 4

Technical Solution

Check listings (in the appendix) for a view of all the code. This code is commented to a high standard, but particularly vital sections will be outlined below.

4.1 Boids

Boids was talked about in design, here is the implementation:

First step is creating the function and and setting variables

```
        def boids(x, gruntlist, playerpos):

            gruntlist = list(gruntlist)

            xtot, ytot = 0,0
            c1,c2 = 0,0
            v1,v2 = 0,0

            x1,y1 = x.rect[0], x.rect[1]
            count = len(gruntlist)
```

Now we start looping through each grunt (each member of the flock), and checking if it is 'in view' of the current (x) grunt, to do this, calculate the distance between the points and check less than 60 (eg, a grunt has a sight radius of 60)

```
            for grunt in gruntlist:
                x2,y2 = grunt.rect[0], grunt.rect[1]


                xtot += x2
                ytot += y2
```

If the boid is in sight then we update our values

```
                if sqrt((x2-x1)**2 + (y2-y1)**2) < 60:
                    c1 = c1 - (x2 - x1)
                    c2 = c2 - (y2 - y1)
                    c1 += (playerpos[0] - x1) / 2
                    c2 += (playerpos[1] - y1) / 2
```

then update these values to reflect the centre of the flock etc

```
                v1 += grunt.vx
                v2 += grunt.vy

                p1 = (playerpos[0]-x1) /5
                p2 = (playerpos[1]-y1) /5
```

these last lines calculate and return the final v of the boid (given as $\Delta x, \Delta y$), which can be added to the current position for the new position.

```
1
2           xavg, yavg = xtot/count, ytot/count
3           vxavg, vyavg = v1/count, v2/count
4
5           return (xavg/100)+c1+(vxavg/20)+p1, (yavg/100)+c2+(vyavg/20)+p2
```

Now we use some functional type programming to efficiently find and update all the positions

```
1
2       gruntslist = list(filter(lambda x:isinstance(x, Grunt) , view.
    spriteslist))
3       f = lambda x:boids(x, gruntslist, player.position)
4
5       newPos = map(f, gruntslist)
6
7       newPos = list(newPos)
8       for i in range(len(newPos)):
9           item, mov = gruntslist[i],newPos[i]
10
11          item.update(view.skincount, mov[0],mov[1])
```

Chapter 5

Testing - TODO

Chapter 6

Evaluation - TODO

Chapter 7

Appendix & Bibliography

7.1 Appendix

Name	Server/Web/Game/Dev	Use
Flask	Server	Handles the API and web on server side
SQLalchemy	Server	Used to connect to the Postgres database
BCrypt	Server	Key derivation
Waitress	Server	WSGI server
PyGame	Game	Graphics and input handling
S3	Server	AWS static file hosting / serving
EC2	Server	AWS server to run flask app
Hetzner	Server	Alternative option to run flask and serve files
PyCharm	Dev	My IDE choice

7.2 Files and Listings

This section will outline the file structure of the project, see the file structure diagram of both the game and website code below

Game Code TODO INSERT DIR TREE

Website Code TODO INSERT DIR TREE

Listings

7.2.1 Website Code

app.py

```

# Flask is used to handle the web requests
from flask import Flask, jsonify, request, render_template

# Sql alchemy handles all SQL interactions, but rather than using and overly relying on the ORM,
# Ill use raw SQL commands. The SQL server is running on RDS (AWS) with PostgreSQL
```

```
from sqlalchemy import create_engine
from sqlalchemy.orm import scoped_session, sessionmaker

# Allow CORS - so it will work from both the webserver and python
from flask_cors import CORS
from waitress import serve
# This is used to hash passwords and validate them - could of used a different tool, or built it myself, but
# But this is prebuilt and purpose designed
import bcrypt
import secrets
# This starts the App
app = Flask(__name__)
# Allow the cors to work
CORS(app)
# Gets the database URL, creates the connection

engine = create_engine(
'sqlite:///test.db',
connect_args={'check_same_thread': False}
)

db = scoped_session(sessionmaker(bind=engine))


db.execute('''
CREATE TABLE IF NOT EXISTS leaderboard (
    id INTEGER UNIQUE PRIMARY KEY AUTOINCREMENT,
    initials VARCHAR(255),
    username VARCHAR(255) UNIQUE,
    password VARCHAR(255)
)''')
db.commit()
db.execute('''
CREATE TABLE IF NOT EXISTS scores(
    id INT,
    scores INT
)
''')
db.commit()
db.execute('''
CREATE TABLE IF NOT EXISTS tokens (
    id INT,
    token VARCHAR
    )
''')
db.commit()

@app.route('/test', methods=['GET'])
def test():
    return render_template('error.html')

@app.route('/', methods=['GET'])
def index():
    leaders = db.execute('''SELECT leaderboard.initials, scores
    FROM leaderboard
    LEFT JOIN scores
    ON leaderboard.id = scores.id
    ORDER BY scores DESC
    LIMIT 10;''')
    # ...so we convert it into a dictionary
    a, d = [], {}
    for lead in leaders:
```

```
        a.append({"initials":lead[0],"scores":lead[1] })
    return render_template('index.html', a=a)


@app.errorhandler(500)
def page_not_found(e):
    # note that we set the 404 status explicitly
    return render_template('error.html')


@app.route('/robo/leaderboard', methods=['GET'])
def leader():
    """
    This route fetches the top 10 results from the server, allowing the page to display the leaderbaord
    :return:
    """
    # This returns a Result Proxy object...
    leaders = db.execute('''SELECT leaderboard.initials, scores
FROM leaderboard
LEFT JOIN scores
ON leaderboard.id = scores.id
ORDER BY scores DESC
LIMIT 10;''')
    # ...so we convert it into a dictionary
    a , d= [], {}
    for lead in leaders:
        for column, value in lead.items():
            d = {**d, **{column: value}}
        a.append(d)

    return jsonify(a)


@app.route('/robo/user/<string:userid>', methods=['GET'])
def user(userid):
    """
    This returns a users high score, given their ID - this means that the API will have to fetch the ID first
    Could it have used the username? probably.
    :param userid:
    :return:
    """
    score = list(db.execute(f'''SELECT score
FROM leaderboard
LEFT JOIN scores
ON leaderboard.id = scores.id
WHERE leaderboard.id = {userid}
ORDER BY scores DESC
LIMIT 1;'''))[0][0]
    return jsonify({'score': score})


@app.route('/robo/userid/<string:username>', methods=['GET'])
def useridget(username):

    """
    This is used to get the id of a user, from their username (which has to be unique)
    Returns a 0 if the username is not unique
    :param username:
    :return:
    """
```

```
    userid = list(db.execute(f"""SELECT leaderboard.id
FROM leaderboard
WHERE leaderboard.username = '{username}'
LIMIT 1;"""))
    try:
        print(userid)
        return jsonify({'id': userid[0][0]})
    except IndexError:
        return jsonify({'id': 0})


@app.route('/login', methods=['POST'])
def login():
    """
    Used to login to the game, returns a  token which is used to verify the user.
    :return:
    """

    userid = request.values.get('userid')
    password = request.values.get('password')
    print(userid)
    hashed = list(db.execute(f'''SELECT password
        FROM leaderboard
        WHERE leaderboard.id = {userid}
        LIMIT 1;'''))[0][0]
    valid = bcrypt.checkpw(password.encode(), hashed.encode())
    if not valid:
        return jsonify({'message': 'password fail'})
    else:
        try:
            token = list(db.execute(f'''SELECT token
            FROM tokens
            WHERE id = {userid}
            LIMIT 1;'''))[0][0]
            return jsonify({'token': token})
        except:
            token = secrets.token_urlsafe(30)
            db.execute(f"""INSERT INTO tokens (id, token)
                VALUES ('{userid}','{token}');""")
            db.commit()
            return jsonify({'token': token})


@app.route('/robo/addscore', methods=['POST'])
def add():
    """
    Used to add scores to the database, uses a post request. Must provide a password to add a score.
    This might be slightly annoying, but adding in functionality for tokens and storing them in python
    feels like a lot of work, maybe I will, but I probably wont invest my time there, I could always cache the
    password inputted in the python code instead.
    :return:
    """
    userid = request.values.get('userid')
    score = int(request.values.get('score'))
    token = request.values.get('token')

    tokenDB = list(db.execute(f'''SELECT token
    FROM tokens
    WHERE id = {userid}
```

```
    LIMIT 1;'''))[0][0]
    valid = token == tokenDB

    if not valid:
        return jsonify({'message': 'password fail'})
    try:
        db.execute(f'''INSERT INTO scores (id, scores)
    VALUES ({userid},{score});''')
        db.commit()

        return jsonify({'message': 'success'})
    except:
        return jsonify({'message': 'fail'})


@app.route('/robo/adduser', methods=['POST'])
def adduser():
    """
    This is the API used to add a user to the database, users provide a username, initials and their password.
    Password validation will be done client side, need to keep this app as lightweight as possible.
    :return:
    """
    username = request.values.get('username')
    initials = request.values.get('initials')
    password = request.values.get('password')

    tostore = bcrypt.hashpw(password.encode(), bcrypt.gensalt()).decode()

    db.execute(f"""INSERT INTO leaderboard (initials, username, password)
    VALUES ('{initials}','{username}','{tostore}');""")
    db.commit()

    return jsonify({'message': 'success'})


if __name__ == '__main__':
    app.run()
```

index.html

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0, shrink-to-fit=no">
    <title>Robotron</title>
    <meta name="theme-color" content="rgb(55,55,55)">
    <meta name="description" content="Robotron leaderboard for robotron by John Montgomery - a pygame game.">
    <link rel="icon" type="image/png" sizes="360x360" href="../static/img/icon.png">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/4.5.2/css/bootstrap.min.css">
    <link rel="manifest" href="manifest.json">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/aos/2.2.0/aos.css">
    <link rel="stylesheet" href="/static/css/styles.css">
</head>

<body style="background: rgb(0,0,0);max-height: 100vh">
```

```
18      <div data-aos="zoom-out" data-aos-duration="2000" style="margin-right: 1%;
    margin-bottom: 0;margin-left: 1%;height: 98vh;width: 98%;margin-top: 1vh;
    border: 3px dotted #9f095c;">
19          <div style="margin-top: 0px;margin-right: 0px;margin-bottom: 0px;margin
    -left: 0px;height: 100%;width: 100%;border: 3px dotted #9f095c;">
20              <div style="margin-top: 0px;margin-right: 0px;margin-bottom: 0px;
    margin-left: 0px;height: 100%;width: 100%;border: 3px dotted #970b60;">
21                  <div style="margin-top: 0px;margin-right: 0px;margin-bottom: 0
    px;margin-left: 0px;height: 100%;width: 100%;border: 3px dotted #970b60;">
22                      <div style="margin-top: 0px;margin-right: 0px;margin-bottom
    : 0px;margin-left: 0px;height: 100%;width: 100%;border: 3px dotted #900c64;"
    >
23                          <div style="margin-top: 0px;margin-right: 0px;margin-
    bottom: 0px;margin-left: 0px;height: 100%;width: 100%;border: 3px dotted
    #900c64;">
24                              <div style="margin-top: 0px;margin-right: 0px;
    margin-bottom: 0px;margin-left: 0px;height: 100%;width: 100%;border: 3px
    dotted #880e68;">
25                                  <div style="margin-top: 0px;margin-right: 0px;
    margin-bottom: 0px;margin-left: 0px;height: 100%;width: 100%;border: 3px
    dotted #880e68;">
26                                      <div style="margin-top: 0px;margin-right: 0
    px;margin-bottom: 0px;margin-left: 0px;height: 100%;width: 100%;border: 3px
    dotted #81106b;">
27                                          <div style="margin-top: 0px;margin-
    right: 0px;margin-bottom: 0px;margin-left: 0px;height: 100%;width: 100%;
    border: 3px dotted #81106b;">
28                                              <div style="margin-top: 0px;margin-
    right: 0px;margin-bottom: 0px;margin-left: 0px;height: 100%;width: 100%;
    border: 3px dotted #7a126f;">
29                                                  <div style="margin-top: 0px;
    margin-right: 0px;margin-bottom: 0px;margin-left: 0px;height: 100%;width:
    100%;border: 3px dotted #7a126f;">
30                                                      <div style="margin-top: 0px
    ;margin-right: 0px;margin-bottom: 0px;margin-left: 0px;height: 100%;width:
    100%;border: 3px dotted #721473;">
31                                                          <div style="margin-top:
     0px;margin-right: 0px;margin-bottom: 0px;margin-left: 0px;height: 100%;
    width: 100%;border: 3px dotted #721473;">
32                                                              <div style="margin-
    top: 0px;margin-right: 0px;margin-bottom: 0px;margin-left: 0px;height: 100%;
    width: 100%;border: 3px dotted #6b1577;">
33                                                                  <div style="
    margin-top: 0px;margin-right: 0px;margin-bottom: 0px;margin-left: 0px;height
    : 100%;width: 100%;border: 3px dotted #6b1577;">
34                                                                      <div style=
    "margin-top: 0px;margin-right: 0px;margin-bottom: 0px;margin-left: 0px;
    height: 100%;width: 100%;border: 3px dotted #63177b;">
35                                                                          <div
    style="margin-top: 0px;margin-right: 0px;margin-bottom: 0px;margin-left: 0px
    ;height: 100%;width: 100%;border: 3px dotted #63177b;">
36                                                                              <
    div style="margin-top: 0px;margin-right: 0px;margin-bottom: 0px;margin-left:
     0px;height: 100%;width: 100%;border: 3px dotted #5b197e;">
37
     <div style="margin-top: 0px;margin-right: 0px;margin-bottom: 0px;margin-
    left: 0px;height: 100%;width: 100%;border: 3px dotted #5b197e;">
38
          <div style="margin-top: 0px;margin-right: 0px;margin-bottom: 0px;margin
    -left: 0px;height: 100%;width: 100%;border: 3px dotted #541b82;">
39
              <div style="margin-top: 0px;margin-right: 0px;margin-bottom: 0px;
    margin-left: 0px;height: 100%;width: 100%;border: 3px dotted #541b82;">
```

```
40
                <div style="margin-top: 0px;margin-right: 0px;margin-bottom: 0
    px;margin-left: 0px;height: 100%;width: 100%;border: 3px dotted #4d1c86;">
41
                    <div style="margin-top: 0px;margin-right: 0px;margin-bottom
    : 0px;margin-left: 0px;height: 100%;width: 100%;border: 3px dotted #4d1c86;"
    >
42
                        <div style="margin-top: 0px;margin-right: 0px;margin-
    bottom: 0px;margin-left: 0px;height: 100%;width: 100%;border: 3px dotted
    #451e8a;">
43
                            <div style="margin-top: 0px;margin-right: 0px;
    margin-bottom: 0px;margin-left: 0px;height: 100%;width: 100%;border: 3px
    dotted #451e8a;">
44
                                <h2 style="color: rgb(69,31,138);font-family:
    Conv_robotron-2084;text-align: center;margin-top: 9px;">robotron heroes</h2>
45
                                <div class="container" style="padding-right: 50
    px;padding-left: 50px;margin-top: 50px;">
46
                                    <div class="row" style="margin-right: -15px
    ;">
47
                                        <div class="col">
48
                                            <h3 style="font-family:
    Conv_robotron-2084;color: rgb(255,51,38);">1 > {{ a[0].initials }} - {{ a
    [0].scores }}</h3>
49
                                        </div>
50
                                        <div class="col">
51
                                            <h3 style="font-family:
    Conv_robotron-2084;color: rgb(255,51,38);">6 > {{ a[5].initials }} - {{ a
    [5].scores }}</h3>
52
                                        </div>
53
                                    </div>
54
                                    <div class="row" style="margin-right: -15px
    ;">
55
                                        <div class="col">
56
                                            <h3 style="font-family:
    Conv_robotron-2084;color: rgb(255,51,38);">2 > {{ a[1].initials }} - {{ a
    [1].scores }}</h3>
57
                                        </div>
58
                                        <div class="col" style="font-family:
    Conv_robotron-2084;color: rgb(255,51,38);">
59
                                            <h3 style="font-family:
    Conv_robotron-2084;color: rgb(255,51,38);">7 > {{ a[6].initials }} - {{ a
    [6].scores }}</h3>
60
                                        </div>
61
```

```
                                    </div>
62
                                    <div class="row" style="margin-right: -15px
    ;">
63
                                        <div class="col">
64
                                            <h3 style="font-family:
    Conv_robotron-2084;color: rgb(255,51,38);">3 > {{ a[2].initials }} - {{ a
    [2].scores }}</h3>
65
                                        </div>
66
                                        <div class="col">
67
                                            <h3 style="font-family:
    Conv_robotron-2084;color: rgb(255,51,38);">8 > {{ a[7].initials }} - {{ a
    [7].scores }}</h3>
68
                                        </div>
69
                                    </div>
70
                                    <div class="row" style="margin-right: -15px
    ;">
71
                                        <div class="col" style="font-family:
    Conv_robotron-2084;color: rgb(255,51,38);">
72
                                            <h3 style="font-family:
    Conv_robotron-2084;color: rgb(255,51,38);">4 > {{ a[3].initials }} - {{ a
    [3].scores }}</h3>
73
                                        </div>
74
                                        <div class="col">
75
                                            <h3 style="font-family:
    Conv_robotron-2084;color: rgb(255,51,38);">9 > {{ a[8].initials }} - {{ a
    [8].scores }}</h3>
76
                                        </div>
77
                                    </div>
78
                                    <div class="row" style="margin-right: -15px
    ;">
79
                                        <div class="col">
80
                                            <h3 style="font-family:
    Conv_robotron-2084;color: rgb(255,51,38);">5 > {{ a[4].initials }} - {{ a
    [4].scores }}</h3>
81
                                        </div>
82
                                        <div class="col">
83
                                            <h3 style="font-family:
    Conv_robotron-2084;color: rgb(255,51,38);">10 > {{ a[9].initials }} - {{ a
    [9].scores }}</h3>
84
                                        </div>
```

```
85                                      </div>
86
87                                  </div>
88
89                                  <div class="row" style="margin-top: 10%;">
90
91                                      <div class="col">
92
93                                          <h1></h1>
94
95                                          <h2 style="color: rgb(69,31,138);font-family: Conv_robotron-2084;text-align: center;margin-top: 9px;">play the game</h2>
96
97                                          <p style="font-family: Conv_robotron-2084;color: rgb(254,51,38);text-align: center;margin-top: 16px;font-size: 16px;">Get the game - <a href="#">Github</a></p>
98
99                                          <p style="font-family: Conv_robotron-2084;color: rgb(254,51,38);text-align: center;margin-top: 16px;font-size: 16px;">Original game info - <a href="#">here</a></p>
100
101                                     </div>
102
103                                 </div>
104
105                                 <h2 style="color: rgb(113,113,113);font-family: Conv_robotron-2084;text-align: center;margin-top: 50px;font-size: 12px;"> by John Montgomery</h2>
106
107                             </div>
108
109                         </div>
110
111                     </div>
112
113                 </div>
114
115             </div>
116
117         </div>
118
119     </div>
120 </div>
```

```
    </div>

    <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/4.5.2/js/bootstrap.bundle.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/aos/2.2.0/aos.js"></script>
    <script src="../static/js/script.min.js"></script>
</body>

</html>
```

error.html

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0, shrink-to-fit=no">
    <title>Robotron</title>
    <meta name="theme-color" content="rgb(194,1,0)">
    <meta name="description" content="Robotron leaderboard for robotron by John Montgomery - a pygame game.">
    <link rel="icon" type="image/png" sizes="360x360" href="../static/img/icon.png">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/4.5.2/css/bootstrap.min.css">
    <link rel="manifest" href="manifest.json">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/aos/2.2.0/aos.css">
    <link rel="stylesheet" href="../static/css/styles.css">
</head>

<body style="background: rgb(0,0,0);">
    <div data-aos="zoom-out" data-aos-duration="2000" style="margin-right: 1%;margin-bottom: 0;margin-left: 1%;height: 98vh;width: 98%;margin-top: 1vh;border: 3px dotted #9f095c;">
        <div style="margin-top: 0px;margin-right: 0px;margin-bottom: 0px;margin-left: 0px;height: 100%;width: 100%;border: 3px dotted #9f095c;">
            <div style="margin-top: 0px;margin-right: 0px;margin-bottom: 0px;margin-left: 0px;height: 100%;width: 100%;border: 3px dotted #970b60;">
                <div style="margin-top: 0px;margin-right: 0px;margin-bottom: 0px;margin-left: 0px;height: 100%;width: 100%;border: 3px dotted #970b60;">
                    <div style="margin-top: 0px;margin-right: 0px;margin-bottom: 0px;margin-left: 0px;height: 100%;width: 100%;border: 3px dotted #900c64;">
                        <div style="margin-top: 0px;margin-right: 0px;margin-bottom: 0px;margin-left: 0px;height: 100%;width: 100%;border: 3px dotted #900c64;">
                            <div style="margin-top: 0px;margin-right: 0px;margin-bottom: 0px;margin-left: 0px;height: 100%;width: 100%;border: 3px dotted #880e68;">
                                <div style="margin-top: 0px;margin-right: 0px;margin-bottom: 0px;margin-left: 0px;height: 100%;width: 100%;border: 3px dotted #880e68;">
                                    <div style="margin-top: 0px;margin-right: 0px;margin-bottom: 0px;margin-left: 0px;height: 100%;width: 100%;border: 3px dotted #81106b;">
                                        <div style="margin-top: 0px;margin-right: 0px;margin-bottom: 0px;margin-left: 0px;height: 100%;width: 100%;border: 3px dotted #81106b;">
```

```
28                                                  <div style="margin-top: 0px;margin-
    right: 0px;margin-bottom: 0px;margin-left: 0px;height: 100%;width: 100%;
    border: 3px dotted #7a126f;">
29                                                      <div style="margin-top: 0px;
    margin-right: 0px;margin-bottom: 0px;margin-left: 0px;height: 100%;width:
    100%;border: 3px dotted #7a126f;">
30                                                          <div style="margin-top: 0px
    ;margin-right: 0px;margin-bottom: 0px;margin-left: 0px;height: 100%;width:
    100%;border: 3px dotted #721473;">
31                                                              <div style="margin-top:
     0px;margin-right: 0px;margin-bottom: 0px;margin-left: 0px;height: 100%;
    width: 100%;border: 3px dotted #721473;">
32                                                                  <div style="margin-
    top: 0px;margin-right: 0px;margin-bottom: 0px;margin-left: 0px;height: 100%;
    width: 100%;border: 3px dotted #6b1577;">
33                                                                      <div style="
    margin-top: 0px;margin-right: 0px;margin-bottom: 0px;margin-left: 0px;height
    : 100%;width: 100%;border: 3px dotted #6b1577;">
34                                                                          <div style=
    "margin-top: 0px;margin-right: 0px;margin-bottom: 0px;margin-left: 0px;
    height: 100%;width: 100%;border: 3px dotted #63177b;">
35                                                                              <div
    style="margin-top: 0px;margin-right: 0px;margin-bottom: 0px;margin-left: 0px
    ;height: 100%;width: 100%;border: 3px dotted #63177b;">
36                                                                                  <
    div style="margin-top: 0px;margin-right: 0px;margin-bottom: 0px;margin-left:
     0px;height: 100%;width: 100%;border: 3px dotted #5b197e;">
37
     <div style="margin-top: 0px;margin-right: 0px;margin-bottom: 0px;margin-
    left: 0px;height: 100%;width: 100%;border: 3px dotted #5b197e;">
38
         <div style="margin-top: 0px;margin-right: 0px;margin-bottom: 0px;margin
    -left: 0px;height: 100%;width: 100%;border: 3px dotted #541b82;">
39
             <div style="margin-top: 0px;margin-right: 0px;margin-bottom: 0px;
    margin-left: 0px;height: 100%;width: 100%;border: 3px dotted #541b82;">
40
                 <div style="margin-top: 0px;margin-right: 0px;margin-bottom: 0
    px;margin-left: 0px;height: 100%;width: 100%;border: 3px dotted #4d1c86;">
41
                     <div style="margin-top: 0px;margin-right: 0px;margin-bottom
    : 0px;margin-left: 0px;height: 100%;width: 100%;border: 3px dotted #4d1c86;"
    >
42
                         <div style="margin-top: 0px;margin-right: 0px;margin-
    bottom: 0px;margin-left: 0px;height: 100%;width: 100%;border: 3px dotted
    #451e8a;">
43
                             <div style="margin-top: 0px;margin-right: 0px;
    margin-bottom: 0px;margin-left: 0px;height: 100%;width: 100%;border: 3px
    dotted #451e8a;">
44
                                 <h1 style="color: rgb(69,31,138);font-family:
    Conv_robotron-2084;text-align: center;margin-top: 9px;">UH OH</h1>
45
46
                                 <h2 style="color: rgb(69,31,138);font-family:
    Conv_robotron-2084;text-align: center;margin-top: 9px;">Something went wrong
     :(</h2>
47
48
                                 <h2 style="color: rgb(113,113,113);font-family:
     Conv_robotron-2084;text-align: center;margin-top: 50px;font-size: 12px;">&
```

```
    nbsp;by John Montgomery</h2>
49
                                </div>
50
                            </div>
51
                        </div>
52
                    </div>
53
                </div>
54
            </div>
55
        </div>
56
                                                                                </
    div>
57
                                                                            </div>
58
                                                                        </div>
59
                                                                    </div>
60
                                                                </div>
61
                                                            </div>
62
                                                        </div>
63
                                                    </div>
64
                                                </div>
65
                                            </div>
66
                                        </div>
67
                                    </div>
68
                                </div>
69
                            </div>
70
                        </div>
71
                    </div>
72
                </div>
73
            </div>
74
    </div>
75
    <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.5.1/jquery.min
    .js"></script>
76
    <script src="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap
    /4.5.2/js/bootstrap.bundle.min.js"></script>
77
    <script src="https://cdnjs.cloudflare.com/ajax/libs/aos/2.2.0/aos.js"></
    script>
78
    <script src="../static/js/script.min.js"></script>
79 </body>
80
81 </html>
```

styles.css

```
1  @font-face {
2      font-family: Conv_robotron-2084;
3      src: url(../fonts/robotron-2084.eot) format("embedded-opentype"), url(../
       fonts/robotron-2084.woff) format("woff"), url(../fonts/robotron-2084.ttf)
       format("truetype"), url(../fonts/robotron-2084.svg) format("svg");
4      font-weight: 400;
5      font-style: normal;
6  }
7  #inputcmd,
8  body {
9      background-color: #333;
10     color: #0f0;
11     font-family: "andale mono", "monotype.com", monaco, "courier new", courier,
        monospace;
12 }
13 #terminal-window {
```

```
    padding: 10px;
    display: block;
    position: absolute;
    width: 100%;
    height: 100%;
    top: 0;
    left: 0;
    background-color: #111;
    overflow: hidden;
}
#terminal-window:before {
    content: "";
    z-index: 4010;
    width: 100%;
    height: 100%;
    position: absolute;
    top: 0;
    left: 0;
    background: linear-gradient(#444 50%, #111 50%);
    background-size: 100% 4px;
    background-repeat: repeat-y;
    opacity: 0.14;
    box-shadow: inset 0 0 1px 1px rgba(0, 0, 0, 0.8);
    animation: 5s linear infinite pulse;
}
#cursor {
    color: #0f0;
    box-sizing: border-box;
    border-left: 0.5em solid;
}
.blink {
    animation: 6s steps(13, end) infinite typing, 1s step-end infinite blinking
    ;
}
.scanlines {
    z-index: 4100;
}
.hide {
    display: none;
}

#inputcmd {
    background-color: #111;
    border: 1px;
    font-size: 1em;
    color: transparent;
    text-shadow: 0 0 0 #0f0;
}
#inputcmd:focus {
    outline: 0;
}
@keyframes pulse {
    0% {
        transform: scale(1.001);
        opacity: 0.14;
    }
    8% {
        transform: scale(1);
        opacity: 0.13;
    }
    15% {
        transform: scale(1.004);
        opacity: 0.14;
```

```
    }
    30% {
        transform: scale(1.002);
        opacity: 0.11;
    }
    100% {
        transform: scale(1);
        opacity: 0.14;
    }
}
@keyframes vline {
    0% {
        top: 0;
    }
    100% {
        top: 100%;
    }
}
@keyframes blinking {
    from,
    to {
        border-color: transparent;
    }
    50% {
        border-color: green;
    }
}
```

7.2.2 Game Code

main.py

```
import sys
import controller, eventmanager

import model
import views


def run(mode):
    evManager = eventmanager.EventManager()
    gamemodel = model.Game(evManager)
    graphics = views.GraphicalView(evManager, gamemodel)
    keyboard = controller.Controller(evManager, gamemodel)

    gamemodel.run(mode)



if __name__ == "__main__":
    try:
        if sys.argv[1].lower() == "test":
            run("test")
    except IndexError:
        run(None)
```

eventmanager.py

```
from event import *


class EventManager:
    """
```

```
    Controls the flow of events between the M, V and C
    """

    def __init__(self):
        """
        Weak ref stops us needing to remove objects from the dict as they will
    end up deleted when the objects instance  is used. This will stop the dict
    becoming bloated and stop me from needing to remember to remove items from
    it.
        """
        self.listeners = []

    def add_listener(self, listener):
        """
        This adds an object as a listener--in-place --aggressive --aggressive
        """
        self.listeners.append(listener)

    def remove_listener(self, listener):
        """
        This is to stop objects listening, but due to the weak referencing it
    doesnt end up used much
        """

        if listener in self.listeners:
            del self.listeners[listener]

    def post(self, event):
        """
        This will emit a message to all the objects in the listen dict
        if it isn't a tick then we also print that event - mostly to debug
        """
        if not isinstance(event, Tick):
            print(str(event))
        for listener in self.listeners:
            listener.notify(event)
```

statemachine.py

```
class StateMachine:
    def __init__(self):
        self.stack = []

    def peek(self):
        try:
            return self.stack[-1]
        except IndexError:
            return None

    def pop(self):
        try:
            self.stack =  self.stack[1:]
            return len(self.stack) > 0
        except IndexError:
            return None

    def push(self, state):
        self.stack.append(state)
        return state
```

model.py

```
from event import *
from statemachine import StateMachine

```

```
from states import *


class Game:
    def __init__(self, eventManager):
        self.statem = StateMachine()
        self.eventManager = eventManager
        eventManager.add_listener(self)
        self.on = False

    def notify(self, event):
        if isinstance(event, EndGame):
            self.stop()
        elif isinstance(event, ChangeState):
            # pop request
            if not event.state:
                # false if no more states are left
                if not self.statem.pop():
                    self.eventManager.Post(EndGame())
            else:
                # push a new state on the stack
                self.statem.push(event.state)

    def stop(self):
        self.on = False


    def run(self, mode='norm'):
        self.on = True
        self.eventManager.post(Start())
        if mode == 'test':
            self.statem.push(STATE_TEST)
        elif mode == 'light':
            pass
        # TODO impliment levels with less characters, slower, etc
        else:
            self.statem.push(STATE_INTRO1)
        while self.on:
            newTick = Tick()
            self.eventManager.post(newTick)
```

views.py

```
import pygame

import menu
import testing
from characters_module.player import Player
from constants.const import *
from decorations.border import Border
from event import *
from states import *
import gameplay
from characters_module.humans import *
from characters_module.enemy import *

class GraphicalView(object):
    """
    Draws the model state onto the screen.
    """

    def __init__(self, evManager, model):
        """
        evManager (EventManager): Allows posting messages to the event queue.
```

```
22          model (GameEngine): a strong reference to the game Model.
23
24          Attributes:
25          isinitialized (bool): pygame is ready to draw.
26          screen (pygame.Surface): the screen surface.
27          clock (pygame.time.Clock): keeps the fps constant.
28          smallfont (pygame.Font): a small font.
29          """
30
31          self.evManager = evManager
32          self.model = model
33          evManager.add_listener(self)
34          self.isinitialized = False
35          self.screen = None
36          self.clock = None
37          self.minifont = None
38          self.smallfont = None
39          self.font = None
40          self.largefont = None
41          self.skincount = 0
42          self.player = None
43          self.currentDown = {
44              97: 0,
45              100: 0,
46              115: 0,
47              119: 0
48          }
49          self.spriteslist = pygame.sprite.Group()
50          self.border = Border()
51          self.spriteslist.add(self.border)
52          self.lastshot = 0
53          self.tickcounter = 0
54          self.titlecolor = (0,0,0)
55          self.color = (0,0,0)
56          self.username = ''
57          self.password = ''
58          self.highlight = None
59          self.username1 = ''
60          self.password1 = ''
61          self.password2 = ''
62          self.initials = ''
63          self.incorrect = False
64          self.level = 1
65          self.lives = 3
66          self.score = 0
67          self.leveldata = {}
68
69      def notify(self, event_in):
70          """
71          Receive events posted to the message queue.
72          """
73          if isinstance(event_in, Start):
74              self.initialize()
75          elif isinstance(event_in, ChangeState):
76              self.tickcounter = 0
77          elif isinstance(event_in, EndGame):
78              # shut down the pygame graphics
79              self.isinitialized = False
80              pygame.quit()
81          elif isinstance(event_in, Tick) or isinstance(event_in, Keyboard) or
    isinstance(event_in, KeyboardUp)  or isinstance(event_in, Mouse):
82              currentstate = self.model.statem.peek()
83              if currentstate == STATE_TEST:
```

```
                testing.testing(event_in, self)
            if currentstate == STATE_INTRO1:
                menu.allopperational(self)
            if currentstate == HOMESCREEN:
                menu.home(self, event_in)
            if currentstate == LOGIN:
                menu.login(self, event_in)
            if currentstate == PLAYGAME:
                self.evManager.post(ChangeState(LOAD_LEVEL1))
            if currentstate == ENDGAME:
                menu.endgame(self, event_in)
            if currentstate>200:
                gameplay.loadlevel(self, currentstate-200)
            if 99<currentstate<200:
                gameplay.level(self, event_in)


    def rendermenu(self):
        self.screen.fill((0, 0, 0))

    def initialize(self):
        """
        Set up the pygame graphical display and loads graphical resources.
        """

        result = pygame.init()
        pygame.font.init()
        pygame.display.set_caption(TITLE)
        self.screen = pygame.display.set_mode(SCREENSIZE)
        self.clock = pygame.time.Clock()
        self.tinyfont = pygame.font.Font('font/robotron-2084.ttf', 10)
        self.minifont = pygame.font.Font('font/robotron-2084.ttf', 18)
        self.smallfont = pygame.font.Font('font/robotron-2084.ttf', 28)
        self.font = pygame.font.Font('font/robotron-2084.ttf', 34)
        self.largefont = pygame.font.Font('font/robotron-2084.ttf', 80)
        self.isinitialized = True
        self.player = Player()
        self.lives = 3
```

controller.py

```

import pygame

from event import *


class Controller:
    def __init__(self, eventManager, model):
        self.eventManager = eventManager
        eventManager.add_listener(self)
        self.model = model

    def notify(self, event):
        if isinstance(event, Tick):

            for event in pygame.event.get():

                if event.type == pygame.QUIT:
                    self.eventManager.post(EndGame())
                if event.type == pygame.KEYDOWN:
                    if event.key != pygame.K_ESCAPE:
                        if event.key != pygame.K_BACKSPACE:
```

```
                            self.eventManager.post(Keyboard(event.key,event.
    unicode))
                        else:
                            self.eventManager.post(Keyboard(event.key, '
    backspace'))
                    else:
                        self.eventManager.post(EndGame())
                if event.type == pygame.KEYUP:
                    self.eventManager.post(KeyboardUp(event.key))

                if event.type == pygame.MOUSEBUTTONDOWN:
                    self.eventManager.post(Mouse(event.pos))
```

event.py

```

class Event:
    """
    A class which is a super for all other events the system might handle
    """
    def __init__(self):
        self.name = 'Some event'

    def __str__(self):
        return self.name

class EndGame(Event):
    """
    This event is sent at the end of the game
    """
    def __init__(self):
        self.name = 'End Game'


class Start(Event):
    """
    This event is sent at the start of the game
    """
    def __init__(self):
        self.name = 'Start Game'


class Tick(Event):
    """
    A tick
    """
    def __init__(self):
        self.name = 'Tick'

class Keyboard(Event):
    """
    Event for keyboard clicks
    """
    def __init__(self, keys, letter):
        self.name = 'Keyboard'
        self.key = keys
        self.uni = letter
    def __str__(self):
        return f"Keypress - {self.uni}"

class KeyboardUp(Event):
    """
    Event for keyboard clicks
    """
```

```
    def __init__(self, keys):
        self.name = 'Keyboard'
        self.key = keys
    def __str__(self):
        return f"Key release - {self.key}"

class Mouse(Event):
    """
    Event for mouse clicks
    """
    def __init__(self, pos):
        self.name = 'Mouse'
        self.pos = pos
    def __str__(self):
        return f"Mouse - {self.pos}"

class ChangeState(Event):
    def __init__(self, newState):
        self.name = 'Change State'
        self.state = newState
    def __str__(self):
        return str(self.state)
```

states.py

```
STATE_ = 1
STATE_TEST = 2
STATE_INTRO1 = 3
STATE_INTRO2 = 4
STATE_PLAY = 5
HOMESCREEN = 6
PLAYGAME = 7
HELP = 8
LOGIN = 9
START_SCREEN = 10
ENDGAME = 11

LEVEL1 = 101
LEVEL2 = 102
LEVEL3 = 103
LEVEL4 = 104
LEVEL5 = 105
LEVEL6 = 106
LEVEL7 = 107
LEVEL8 = 108
LEVEL9 = 109
LEVEL10 = 110
LEVEL11 = 111
LEVEL12 = 112
LEVEL13 = 113
LEVEL14 = 114
LEVEL15 = 115
LEVEL16 = 116
LEVEL17 = 117
LEVEL18 = 118
LEVEL19 = 119
LEVEL20 = 120
LEVEL21 = 121
LEVEL22 = 122
LEVEL23 = 123
LEVEL24 = 124
LEVEL25 = 125
LEVEL26 = 126
LEVEL27 = 127
```

```
LEVEL28 = 128
LEVEL29 = 129
LEVEL30 = 130
LEVEL31 = 131
LEVEL32 = 132
LEVEL33 = 133
LEVEL34 = 134
LEVEL35 = 135
LEVEL36 = 136
LEVEL37 = 137
LEVEL38 = 138
LEVEL39 = 139
LEVEL40 = 140

LOAD_LEVEL1 = 201
LOAD_LEVEL2 = 202
LOAD_LEVEL3 = 203
LOAD_LEVEL4 = 204
LOAD_LEVEL5 = 205
LOAD_LEVEL6 = 206
LOAD_LEVEL7 = 207
LOAD_LEVEL8 = 208
LOAD_LEVEL9 = 209
LOAD_LEVEL10 = 210
LOAD_LEVEL11 = 211
LOAD_LEVEL12 = 212
LOAD_LEVEL13 = 213
LOAD_LEVEL14 = 214
LOAD_LEVEL15 = 215
LOAD_LEVEL16 = 216
LOAD_LEVEL17 = 217
LOAD_LEVEL18 = 218
LOAD_LEVEL19 = 219
LOAD_LEVEL20 = 220
LOAD_LEVEL21 = 221
LOAD_LEVEL22 = 222
LOAD_LEVEL23 = 223
LOAD_LEVEL24 = 224
LOAD_LEVEL25 = 225
LOAD_LEVEL26 = 226
LOAD_LEVEL27 = 227
LOAD_LEVEL28 = 228
LOAD_LEVEL29 = 229
LOAD_LEVEL30 = 230
LOAD_LEVEL31 = 231
LOAD_LEVEL32 = 232
LOAD_LEVEL33 = 233
LOAD_LEVEL34 = 234
LOAD_LEVEL35 = 235
LOAD_LEVEL36 = 236
LOAD_LEVEL37 = 237
LOAD_LEVEL38 = 238
LOAD_LEVEL39 = 239
LOAD_LEVEL40 = 240
```

menu.py

```
import random
import webbrowser
from playsound import playsound
import pygame

from APIinteractions import *
from constants.colors import *
```

```
from constants.const import *
from event import *
from states import *

_circle_cache = {}
def _circlepoints(r):
    r = int(round(r))
    if r in _circle_cache:
        return _circle_cache[r]
    x, y, e = r, 0, 1 - r
    _circle_cache[r] = points = []
    while x >= y:
        points.append((x, y))
        y += 1
        if e < 0:
            e += 2 * y - 1
        else:
            x -= 1
            e += 2 * (y - x) - 1
    points += [(y, x) for x, y in points if x > y]
    points += [(-x, y) for x, y in points if x]
    points += [(x, -y) for x, y in points if y]
    points.sort()
    return points

def render(text, font, gfcolor=pygame.Color('dodgerblue'), ocolor=(255, 130,
    45), opx=2):
    textsurface = font.render(text, True, gfcolor).convert_alpha()
    w = textsurface.get_width() + 2 * opx
    h = font.get_height()

    osurf = pygame.Surface((w, h + 2 * opx)).convert_alpha()
    osurf.fill((0, 0, 0, 0))

    surf = osurf.copy()

    osurf.blit(font.render(text, True, ocolor).convert_alpha(), (0, 0))

    for dx, dy in _circlepoints(opx):
        surf.blit(osurf, (dx + opx, dy + opx))

    surf.blit(textsurface, (opx, opx))
    return surf

def get_ran_col():
    return random.choice(random_colors)

def randomStart(view):
    for i in range(0, SCREENSIZE[0], 2):
        for j in range(0, SCREENSIZE[1], 2):
            col = get_ran_col()
            rect = pygame.Rect((i, j), (2, 2))
            pygame.draw.rect(view.screen, col, rect)

def allopperational(view):
    view.tickcounter += 1
    if view.tickcounter == 2:
        playsound('audio/intro.mp3', block = False)
    if view.tickcounter > 40:
        view.evManager.post(ChangeState(HOMESCREEN))
    elif view.tickcounter > 5:

        view.screen.fill((10, 10, 10))
```

```
70          todisplay1 = '''Initial tests indicate:'''
71
72          todisplay2 = 'Operational'
73
74          somewords1 = view.font.render(
75              todisplay1,
76              True,
77              WHITE)
78
79          somewords2 = view.font.render(
80              todisplay2,
81              True,
82              WHITE)
83
84          width1, _ = pygame.font.Font.size(view.font, todisplay1)
85          position_font1 = (SCREENSIZE[0] - width1) / 2
86          view.screen.blit(somewords1, (position_font1, SCREENSIZE[1]/2-50))
87
88          width2, _ = pygame.font.Font.size(view.font, todisplay2)
89          position_font2 = (SCREENSIZE[0] - width2) / 2
90          view.screen.blit(somewords2, (position_font2, SCREENSIZE[1]/2+50))
91
92      else:
93          randomStart(view)
94      pygame.display.flip()
95
96      view.clock.tick(TPS)
97
98
99  def home(view, event):
100     view.screen.fill(BLACK)
101     view.tickcounter += 1
102     if isinstance(event, Keyboard):
103         if event.key == 32:
104             view.evManager.post(ChangeState(PLAYGAME))
105         if event.key == 104:
106             view.evManager.post(ChangeState(HELP))
107         if event.key == 13:
108             view.evManager.post(ChangeState(LOGIN))
109         if event.key == 111:
110             webbrowser.open('https://robo.johnmontgomery.tech', new=2)
111     else:
112         prog = list(range(40,0,-1))
113         if view.tickcounter % 10 == 1:
114             view.col = random.choice(title_colors)
115             view.edgecol = random.choice(edge)
116         for idx, letter in enumerate('ROBOTRON:'):
117             image = render(letter, view.largefont, gfcolor=view.col, ocolor=
    view.edgecol)
118             w,h = image.get_width(), image.get_height()
119             image = pygame.transform.scale(image, (w, 0 if view.tickcounter<idx
     else h+int(1.3**prog[view.tickcounter-idx if view.tickcounter- idx<40 else
    39])))
120             view.screen.blit(image , (88+idx*74,90-image.get_height()/2))
121
122         if 220 >= view.tickcounter > 40:
123             view.tickcounter += 2
124             image = pygame.image.load('sprites/2084.png')
125             w,h = image.get_width(), image.get_height()
126             image = pygame.transform.scale(image, (w, 180*h // (view.
    tickcounter - 40)))
127             view.screen.blit(image, (196, (100+ (180*h // (view.tickcounter -
    40)))/2 ))
```

```
128         if 220 < view.tickcounter:
129             image = pygame.image.load('sprites/2084.png')
130             view.screen.blit(image, (196,140))
131 
132             somewords = view.smallfont.render(
133                 'Created By:',
134                 True,
135                 (246, 130, 20))
136             width, _ = pygame.font.Font.size(view.smallfont, 'Created By:')
137             position_font = (SCREENSIZE[0] - width) / 2
138             view.screen.blit(somewords, (position_font + 6, 320))
139 
140             somewords = view.smallfont.render(
141                 'John Montgomery',
142                 True,
143                 (246, 130, 20))
144             width, _ = pygame.font.Font.size(view.smallfont, 'John Montgomery')
145             position_font = (SCREENSIZE[0] - width) / 2
146             view.screen.blit(somewords, (position_font + 6, 360))
147             if view.tickcounter % 5 == 0:
148                 if view.color == (0,0,0):
149                     view.color = (22, 32, 221)
150                 else:
151                     view.color = (0,0,0)
152             somewords = view.font.render(
153                 'SPACE to PLAY',
154                 True,
155                 view.color )
156             width, _ = pygame.font.Font.size(view.font, 'SPACE to PLAY')
157             position_font = (SCREENSIZE[0] - width) / 2
158             view.screen.blit(somewords, (position_font + 6, 400))
159 
160             somewords = view.smallfont.render(
161                 'H for HELP',
162                 True,
163                 (22, 32, 221))
164             width, _ = pygame.font.Font.size(view.smallfont, 'H for HELP')
165             position_font = (SCREENSIZE[0] - width) / 2
166             view.screen.blit(somewords, (position_font + 6, 440))
167 
168             try:
169                 with open('.token', 'r')as f:
170                     text = f.read().split('>')[2]
171                     somewords = view.smallfont.render(
172                         'LOGGED IN AS '+text,
173                         True,
174                         (22, 32, 221))
175                     width, _ = pygame.font.Font.size(view.smallfont, 'LOGGED IN
    AS '+text)
176                     position_font = (SCREENSIZE[0] - width) / 2
177                     view.screen.blit(somewords, (position_font + 6, 470))
178             except FileNotFoundError:
179                 somewords = view.smallfont.render(
180                     'ENTER for LOGIN',
181                     True,
182                     (22, 32, 221))
183                 width, _ = pygame.font.Font.size(view.smallfont, 'ENTER for
   LOGIN')
184                 position_font = (SCREENSIZE[0] - width) / 2
185                 view.screen.blit(somewords, (position_font + 6, 470))
186 
187             somewords = view.minifont.render(
188                 '''Leaderboard Avaliable at - robo.johnmontgomery.tech''',
```

```
189                 True,
190                 random.choice(title_colors))
191             width, _ = pygame.font.Font.size(view.minifont, 'Leaderboard
    Avaliable at - robo.johnmontgomery.tech')
192             position_font = (SCREENSIZE[0] - width) / 2
193             view.screen.blit(somewords, (position_font + 6, 510))
194 
195             somewords = view.minifont.render(
196                 '(Press o to open link)',
197                 True,
198                 (255,255,255))
199             width, _ = pygame.font.Font.size(view.minifont, '(Press o to open
    link)')
200             position_font = (SCREENSIZE[0] - width) / 2
201             view.screen.blit(somewords, (position_font + 6, 540))
202 
203             somewords = view.minifont.render(
204                 'ORIGIONAL GAME CREATED BY: WILLIAM ELECTRONICS INC.',
205                 True,
206                 (246, 130, 20))
207             width, _ = pygame.font.Font.size(view.minifont, 'ORIGIONAL GAME
    CREATED BY: WILLIAM ELECTRONICS INC.')
208             position_font = (SCREENSIZE[0] - width) / 2
209             view.screen.blit(somewords, (position_font + 6, 570))
210 
211     pygame.display.flip()
212 
213     view.clock.tick(TPS)
214 
215 def login(view, event):
216     view.screen.fill(BLACK)
217     if isinstance(event, Mouse):
218         if 340 < event.pos[0] < 460 and 200 < event.pos[1] < 240:
219             status = loginuser(view.username, view.password)
220             if status:
221                 view.evManager.post(ChangeState(HOMESCREEN))
222             else:
223                 view.incorrect = 250
224 
225         elif 320 < event.pos[0] < 480 and 500 < event.pos[1] < 540:
226             success = signupuser(view.username1, view.password1, view.password2
    , view.initials)
227             if success:
228                 view.evManager.post(ChangeState(HOMESCREEN))
229             else:
230                 view.incorrect = 550
231 
232         elif 0 < event.pos[0] < 50 and 0 < event.pos[1] < 50:
233             view.evManager.post(ChangeState(HOMESCREEN))
234 
235 
236     if view.incorrect:
237         somewords = view.smallfont.render(
238             'INCORRECT',
239             True,
240             (200,0,0))
241         width, _ = pygame.font.Font.size(view.smallfont, 'INCORRECT')
242         position_font = (SCREENSIZE[0] - width) / 2
243         view.screen.blit(somewords, (position_font + 6, view.incorrect))
244 
245     somewords = view.font.render(
246         'LOGIN + SIGN UP',
247         True,
```

```
248         (246, 130, 20))
249     width, _ = pygame.font.Font.size(view.font, 'LOGIN + SIGN UP')
250     position_font = (SCREENSIZE[0] - width) / 2
251     view.screen.blit(somewords, (position_font + 6, 20))
252 
253     logintext = view.smallfont.render(
254         'LOGIN',
255         True,
256         (255, 255, 255))
257     width, _ = pygame.font.Font.size(view.smallfont, 'LOGIN')
258     position_font = (SCREENSIZE[0] - width) / 2
259     view.screen.blit(logintext, (position_font, 206))
260 
261     pygame.draw.rect(view.screen, GREY, pygame.Rect(100, 100, 600, 40), width
    =3)
262 
263     pygame.draw.rect(view.screen, GREY, pygame.Rect(100, 150, 600, 40), width
    =3)
264 
265     pygame.draw.rect(view.screen, GREY, pygame.Rect(340, 200, 120, 40), width
    =3)
266 
267     pygame.draw.lines(view.screen, GREY, False, [(30,10),(10,25), (30, 40)],
    width=5)
268 
269     signup = view.smallfont.render(
270         'SIGN UP',
271         True,
272         (255, 255, 255))
273     width, _ = pygame.font.Font.size(view.smallfont, 'SIGN UP')
274     position_font = (SCREENSIZE[0] - width) / 2
275     view.screen.blit(signup, (position_font, 506))
276 
277     pygame.draw.rect(view.screen, GREY, pygame.Rect(100, 300, 600, 40), width
    =3)
278 
279     pygame.draw.rect(view.screen, GREY, pygame.Rect(100, 350, 600, 40), width
    =3)
280 
281     pygame.draw.rect(view.screen, GREY, pygame.Rect(100, 400, 600, 40), width
    =3)
282 
283     pygame.draw.rect(view.screen, GREY, pygame.Rect(100, 450, 600, 40), width
    =3)
284 
285     pygame.draw.rect(view.screen, GREY, pygame.Rect(320, 500, 160, 40), width
    =3)
286 
287 
288     if isinstance(event, Mouse):
289         if 100<event.pos[0]<700 and 100<event.pos[1]<140:
290             pygame.draw.rect(view.screen, WHITE, pygame.Rect(100,97,600,43),
    width=5)
291             view.highlight = 'username'
292         elif 100<event.pos[0]<700 and 150<event.pos[1]<190:
293             pygame.draw.rect(view.screen, WHITE, pygame.Rect(100,147,600,43),
    width=5)
294             view.highlight = 'password'
295         elif 100<event.pos[0]<700 and 300<event.pos[1]<340:
296             pygame.draw.rect(view.screen, WHITE, pygame.Rect(100, 297, 600, 43)
    , width=5)
297             view.highlight = 'username1'
298         elif 100<event.pos[0]<700 and 350<event.pos[1]<390:
```

```
299             pygame.draw.rect(view.screen, WHITE, pygame.Rect(100, 347, 600, 43)
    , width=5)
300             view.highlight = 'password1'
301         elif 100<event.pos[0]<700 and 400<event.pos[1]<440:
302             pygame.draw.rect(view.screen, WHITE, pygame.Rect(100, 397, 600, 43)
    , width=5)
303             view.highlight = 'password2'
304         elif 100<event.pos[0]<700 and 450<event.pos[1]<490:
305             pygame.draw.rect(view.screen, WHITE, pygame.Rect(100, 447, 600, 43)
    , width=5)
306             view.highlight = 'initials'
307         else:
308             view.highlight = None
309     else:
310         if view.highlight:
311             if view.highlight == 'username':
312                 pygame.draw.rect(view.screen, WHITE, pygame.Rect(100, 97, 600,
    43), width=5)
313             elif view.highlight == 'username1':
314                 pygame.draw.rect(view.screen, WHITE, pygame.Rect(100, 297, 600,
     43), width=5)
315             elif view.highlight == 'password1':
316                 pygame.draw.rect(view.screen, WHITE, pygame.Rect(100, 347, 600,
     43), width=5)
317             elif view.highlight == 'password2':
318                 pygame.draw.rect(view.screen, WHITE, pygame.Rect(100, 397, 600,
     43), width=5)
319             elif view.highlight == 'initials':
320                 pygame.draw.rect(view.screen, WHITE, pygame.Rect(100, 447, 600,
     43), width=5)
321             else:
322                 pygame.draw.rect(view.screen, WHITE, pygame.Rect(100, 147, 600,
     43), width=5)
323 
324     if isinstance(event, Keyboard):
325         if view.highlight:
326             if view.highlight == 'username':
327                 if event.uni != 'backspace' :
328                     view.username += event.uni
329                 else:
330                     view.username = view.username[:-1]
331                 if len(view.username)>40:
332                     view.username = view.username[:-1]
333 
334             elif view.highlight == 'username1':
335                 if event.uni != 'backspace' :
336                     view.username1 += event.uni
337                 else:
338                     view.username1 = view.username1[:-1]
339                 if len(view.username1)>40:
340                     view.username1 = view.username1[:-1]
341 
342             elif view.highlight == 'password1':
343                 if event.uni != 'backspace' :
344                     view.password1 += event.uni
345                 else:
346                     view.password1 = view.password1[:-1]
347                 if len(view.password1)>40:
348                     view.password1 = view.password1[:-1]
349 
350             elif view.highlight == 'password2':
351                 if event.uni != 'backspace' :
352                     view.password2 += event.uni
```

```
                else:
                    view.password2 = view.password2[:-1]
                if len(view.password2)>40:
                    view.password2 = view.password2[:-1]

            elif view.highlight == 'initials':
                if event.uni != 'backspace' :
                    view.initials += event.uni
                else:
                    view.initials = view.initials[:-1]
                if len(view.initials)>3:
                    view.initials = view.initials[:-1]

            else:
                if event.uni != 'backspace':
                    view.password += event.uni
                else:
                    view.password = view.password[:-1]
                if len(view.password)>40:
                    view.username = view.username[:-1]

    if view.password2 != view.password1:
        pygame.draw.rect(view.screen, RED, pygame.Rect(100, 347, 600, 43),
    width=5)
        pygame.draw.rect(view.screen, RED, pygame.Rect(100, 397, 600, 43),
    width=5)

    text = view.tinyfont.render(
        'username',
        True,
        (255, 255, 255))
    width, _ = pygame.font.Font.size(view.tinyfont, 'username')
    position_font = (SCREENSIZE[0] - width) / 2
    view.screen.blit(text, (position_font + 6, 301))

    text = view.minifont.render(
        view.username1,
        True,
        (255, 255, 255))
    width, _ = pygame.font.Font.size(view.minifont, view.username1)
    position_font = (SCREENSIZE[0] - width) / 2
    view.screen.blit(text, (position_font + 6, 311))

    text = view.tinyfont.render(
        'password',
        True,
        (255, 255, 255))
    width, _ = pygame.font.Font.size(view.tinyfont, 'password')
    position_font = (SCREENSIZE[0] - width) / 2
    view.screen.blit(text, (position_font + 6, 351))

    text = view.minifont.render(
        '*'*len(view.password1),
        True,
        (255, 255, 255))
    width, _ = pygame.font.Font.size(view.minifont, '*'*len(view.password1))
    position_font = (SCREENSIZE[0] - width) / 2
    view.screen.blit(text, (position_font + 6, 361))

    text = view.tinyfont.render(
        'confirm password',
        True,
        (255, 255, 255))
```

```
414     width, _ = pygame.font.Font.size(view.tinyfont, 'confirm password')
415     position_font = (SCREENSIZE[0] - width) / 2
416     view.screen.blit(text, (position_font + 6, 401))
417 
418     text = view.minifont.render(
419         '*'*len(view.password2),
420         True,
421         (255, 255, 255))
422     width, _ = pygame.font.Font.size(view.minifont, '*'*len(view.password2))
423     position_font = (SCREENSIZE[0] - width) / 2
424     view.screen.blit(text, (position_font + 6, 411))
425 
426     text = view.tinyfont.render(
427         'initials',
428         True,
429         (255, 255, 255))
430     width, _ = pygame.font.Font.size(view.tinyfont, 'initials')
431     position_font = (SCREENSIZE[0] - width) / 2
432     view.screen.blit(text, (position_font + 6, 451))
433 
434     text = view.minifont.render(
435         view.initials,
436         True,
437         (255, 255, 255))
438     width, _ = pygame.font.Font.size(view.minifont, view.initials)
439     position_font = (SCREENSIZE[0] - width) / 2
440     view.screen.blit(text, (position_font + 6, 461))
441 
442     text = view.tinyfont.render(
443         'username',
444         True,
445         (255, 255, 255))
446     width, _ = pygame.font.Font.size(view.tinyfont, 'username')
447     position_font = (SCREENSIZE[0] - width) / 2
448     view.screen.blit(text, (position_font + 6, 101))
449 
450     usernametext = view.minifont.render(
451         view.username,
452         True,
453         (255, 255, 255))
454     width, _ = pygame.font.Font.size(view.minifont, view.username)
455     position_font = (SCREENSIZE[0] - width) / 2
456     view.screen.blit(usernametext, (position_font + 6, 111))
457 
458     text = view.tinyfont.render(
459         'password',
460         True,
461         (255, 255, 255))
462     width, _ = pygame.font.Font.size(view.tinyfont, 'password')
463     position_font = (SCREENSIZE[0] - width) / 2
464     view.screen.blit(text, (position_font + 6, 151))
465 
466     passwordtext = view.minifont.render(
467         len(view.password) * '*',
468         True,
469         (255, 255, 255))
470     width, _ = pygame.font.Font.size(view.minifont, len(view.password) * '*')
471     position_font = (SCREENSIZE[0] - width) / 2
472     view.screen.blit(passwordtext, (position_font + 6, 161))
473 
474 
475     pygame.display.flip()
476 
```

```
477     view.clock.tick(TPS)
478
479
480 def endgame(view, event):
481     view.screen.fill(BLACK)
482     view.tickcounter += 1
483     if isinstance(event, Keyboard):
484         if event.key == 32:
485             view.evManager.post(ChangeState(HOMESCREEN))
486         if event.key == 111:
487             webbrowser.open('https://robo.johnmontgomery.tech', new=2)
488         if event.key == 13:
489             view.evManager.post(ChangeState(LOGIN))
490     else:
491         prog = list(range(40, 0, -1))
492         if view.tickcounter % 10 == 1:
493             view.col = random.choice(title_colors)
494             view.edgecol = random.choice(edge)
495         for idx, letter in enumerate('ROBOTRON:'):
496             image = render(letter, view.largefont, gfcolor=view.col, ocolor=
    view.edgecol)
497             w, h = image.get_width(), image.get_height()
498             image = pygame.transform.scale(image, (w, 0 if view.tickcounter <
    idx else h + int(
499                 1.3 ** prog[view.tickcounter - idx if view.tickcounter - idx <
    40 else 39])))
500             view.screen.blit(image, (88 + idx * 74, 90 - image.get_height() /
    2))
501         if 220 >= view.tickcounter > 40:
502             view.tickcounter += 2
503             image = pygame.image.load('sprites/2084.png')
504             w, h = image.get_width(), image.get_height()
505             image = pygame.transform.scale(image, (w, 180 * h // (view.
    tickcounter - 40)))
506             view.screen.blit(image, (196, (100 + (180 * h // (view.tickcounter
    - 40))) / 2))
507         if 220 < view.tickcounter:
508             image = pygame.image.load('sprites/2084.png')
509             view.screen.blit(image, (196, 140))
510
511             if view.tickcounter % 5 == 0:
512                 if view.color == (0, 0, 0):
513                     view.color = (22, 32, 221)
514                 else:
515                     view.color = (0, 0, 0)
516
517             somewords = view.font.render(
518                 'GAME OVER',
519                 True,
520                 view.color)
521             width, _ = pygame.font.Font.size(view.font, 'GAME OVER')
522             position_font = (SCREENSIZE[0] - width) / 2
523             view.screen.blit(somewords, (position_font + 6, 330))
524
525             somewords = view.font.render(
526                 'You scored:',
527                 True,
528                 (246, 130, 20))
529             width, _ = pygame.font.Font.size(view.font, 'You scored:')
530             position_font = (SCREENSIZE[0] - width) / 2
531             view.screen.blit(somewords, (position_font + 6, 400))
532
533             somewords = view.font.render(
```

```
                str(view.score),
                True,
                (246, 130, 20))
            width, _ = pygame.font.Font.size(view.font, str(view.score))
            position_font = (SCREENSIZE[0] - width) / 2
            view.screen.blit(somewords, (position_font + 6, 450))

            somewords = view.smallfont.render(
                'SPACE for homescreen',
                True,
                (246, 130, 20))
            width, _ = pygame.font.Font.size(view.smallfont, 'SPACE for
    homescreen')
            position_font = (SCREENSIZE[0] - width) / 2
            view.screen.blit(somewords, (position_font + 6, 500))

            somewords = view.smallfont.render(
                '0 to open leaderboard',
                True,
                (246, 130, 20))
            width, _ = pygame.font.Font.size(view.smallfont, '0 to open
    leaderboard')
            position_font = (SCREENSIZE[0] - width) / 2
            view.screen.blit(somewords, (position_font + 6, 525))

            if checkonline():
                if isloggedin():
                    if addscore(view.score):
                        somewords = view.smallfont.render(
                            'Score added to leaderboard',
                            True,
                            (246, 130, 20))
                        width, _ = pygame.font.Font.size(view.smallfont, 'Score
     added to leaderboard')
                        position_font = (SCREENSIZE[0] - width) / 2
                        view.screen.blit(somewords, (position_font + 6, 550))
                else:
                    somewords = view.smallfont.render(
                        'Enter to log in',
                        True,
                        (246, 130, 20))
                    width, _ = pygame.font.Font.size(view.smallfont, 'Enter to
    log in')
                    position_font = (SCREENSIZE[0] - width) / 2
                    view.screen.blit(somewords, (position_font + 6, 550))
            else:
                somewords = view.smallfont.render(
                    'OFFLINE',
                    True,
                    RED)
                width, _ = pygame.font.Font.size(view.smallfont, 'OFFLINE')
                position_font = (SCREENSIZE[0] - width) / 2
                view.screen.blit(somewords, (position_font + 6, 550))
    pygame.display.flip()

    view.clock.tick(TPS)
```

gameplay.py

```
import pygame
from constants.colors import *
from event import *
from constants.const import *
from objects.bullet import  Bullet
```

```
from states import *
import csv
from characters_module.enemy import *
from characters_module.humans import *
from characters_module.player import *
from playsound import playsound
from math import sqrt
def loadlevel(view, level):
    playsound('audio/change.mp3', block=False)
    with open ('levels/levels.csv') as f:
        print(level)
        csvreader = csv.reader(f, delimiter=',' )
        line = 0
        for row in csvreader:
            if line == 0:
                headers = row
            if line == level:
                leveldata = row
            line += 1

    for header,count in zip(headers[1:], leveldata[1:]):
        view.leveldata[header] = count

    for char in view.leveldata.keys():
        for _ in range(int(view.leveldata[char])):
            newobject = eval(f"{char}()")
            view.spriteslist.add(newobject)


    r, g, b = 0,102,102
    view.screen.fill(BLACK)
    for i in range(60):
        if r > 0 and b == 0:
            r -= 17
            g += 17
        if g > 0 and r == 0:
            g -= 17
            b += 17
        if b > 0 and g == 0:
            b -= 17
            r += 17
        pygame.draw.rect(view.screen, (r,g,b), pygame.Rect(200- (i*5 + 10),
    SCREENSIZE[1]/2 - i*7 + 10, (SCREENSIZE[0]- 2 * (200- (i*5 + 10))), i*14 +
    10), width=3)
        view.clock.tick(TPS)
        pygame.display.flip()

    for i in range(60):
        pygame.draw.rect(view.screen, (0,0,0), pygame.Rect(200- (i*5 + 10),
    SCREENSIZE[1]/2 - i*7 + 10, (SCREENSIZE[0]- 2 * (200- (i*5 + 10))), i*14 +
    10), width=3)
        view.clock.tick(TPS+4)
        pygame.display.flip()

    view.evManager.post(ChangeState(100+level))
    return


def level(view, event):
    player = view.player

    if not view.isinitialized:
        return
```

```
65
66      view.screen.fill(BLACK)
67
68      if view.tickcounter <= 30:
69          view.player.onstart(view)
70
71      view.tickcounter += 1
72      if isinstance(event, Keyboard):
73          view.currentDown[event.key] = 1
74
75      if isinstance(event, KeyboardUp):
76          view.currentDown[event.key] = 0
77
78      shoot = ''
79
80      v = VELOCITY if sum(view.currentDown.values()) > 1 else DVELOCITY
81      for key in view.currentDown.keys():
82
83          if view.currentDown[key]:
84              if key == 119:
85                  player.movy(-v)
86              if key == 115:
87                  player.movy(v)
88              if key == 97:
89                  player.movx(-v)
90              if key == 100:
91                  player.movx(v)
92              if len(shoot) < 2:
93                  if key == 105:
94                      shoot += 'N'
95                  if key == 107:
96                      shoot += 'S'
97                  if key == 106:
98                      shoot += 'W'
99                  if key == 108:
100                     shoot += 'E'
101
102     if shoot:
103         if view.lastshot == 0:
104             bullet = Bullet(player.position[0], player.position[1], shoot)
105             view.spriteslist.add(bullet)
106             view.lastshot += COOLDOWN
107         else:
108             view.lastshot -= 1
109
110
111     view.skincount += 1 if view.tickcounter % 2 == 0 else 0
112     if view.skincount > 2:
113         view.skincount = 0
114     playlist = [view.player.position]
115     for idx,item in enumerate(view.spriteslist):
116         if not isinstance(item, Grunt):
117             item.update(view.skincount, playlist[idx%len(playlist)])
118
119     if view.tickcounter > 50:
120
121         def boids(x, gruntlist, playerpos):
122
123             gruntlist = list(gruntlist)
124
125             xtot, ytot = 0,0
126             c1,c2 = 0,0
127             v1,v2 = 0,0
```

```
128
129             x1,y1 = x.rect[0], x.rect[1]
130             count = len(gruntlist)
131
132             for grunt in gruntlist:
133                 x2,y2 = grunt.rect[0], grunt.rect[1]
134
135
136                 xtot += x2
137                 ytot += y2
138
139                 if sqrt((x2-x1)**2 + (y2-y1)**2) < 60:
140                     c1 = c1 - (x2 - x1)
141                     c2 = c2 - (y2 - y1)
142                     c1 += (playerpos[0] - x1) / 2
143                     c2 += (playerpos[1] - y1) / 2
144
145                 v1 += grunt.vx
146                 v2 += grunt.vy
147
148                 p1 = (playerpos[0]-x1) /5
149                 p2 = (playerpos[1]-y1) /5
150
151             xavg, yavg = xtot/count, ytot/count
152             vxavg, vyavg = v1/count, v2/count
153
154             return (xavg/100)+c1+(vxavg/20)+p1, (yavg/100)+c2+(vyavg/20)+p2
155
156         gruntslist = list(filter(lambda x:isinstance(x, Grunt) , view.
    spriteslist))
157         f = lambda x:boids(x, gruntslist, player.position)
158
159         newPos = map(f, gruntslist)
160
161         newPos = list(newPos)
162         for i in range(len(newPos)):
163             item, mov = gruntslist[i],newPos[i]
164
165             item.update(view.skincount, mov[0],mov[1])
166
167         for item in view.spriteslist:
168             if isinstance(item, Bullet):
169                 for object in view.spriteslist:
170                     if -20<item.rect[0]-object.rect[0]<20 and -20<item.rect[1]-
    object.rect[1]<20  and not isinstance(object, Bullet):
171                         if isinstance(object, Grunt) or isinstance(object,
    Electrode) or isinstance(object, Hulk):
172                             object.kill()
173                         item.kill()
174             if isinstance(item, Electrode) or isinstance(item, Grunt) or
    isinstance(item, Hulk):
175                 if -20<item.rect[0]-player.position[0]<20 and -20<item.rect[1]-
    player.position[1]<20:
176                     view.lives -= 1
177                     if view.lives > 0:
178                         view.evManager.post(ChangeState(view.model.statem.peek
    () + 101))
179                         return
180                     else:
181                         view.evManager.post(ChangeState(ENDGAME))
182             if isinstance(item, Electrode):
183                 for object in view.spriteslist:
184                     if -10 < item.rect[0] - object.rect[0] < 10 and -10 < item.
```

```
    rect[1] - object.rect[1] < 10 and not isinstance(object, Electrode):
                        if isinstance(object, Grunt):
                            object.kill()
            if isinstance(item, Mommies) or isinstance(item, Daddies) or
    isinstance(item, Mikeys):
                if -20 < item.rect[0] - player.position[0] < 20 and -20 < item.
    rect[1] - player.position[1] < 20:
                    score = item.die(view)
                    view.score += score

    gruntcount = sum(1 if isinstance(i, Grunt) else 0 for i in view.spriteslist
    )
    if gruntcount == 0:
        view.evManager.post(ChangeState(view.model.statem.peek() + 101))
        return

    view.spriteslist.draw(view.screen)



    if view.tickcounter > 30:
        player.getskin(view.skincount)
        view.screen.blit(player.getskin(view.skincount), player.position)


    view.clock.tick(TPS)
    # flip the display to show whatever we drew

    pygame.display.flip()
```

APIinteractions.py

```
import requests

apiurl = 'http://127.0.0.1:5000'
from requests.adapters import HTTPAdapter
from requests.packages.urllib3.util.retry import Retry


session = requests.Session()
retry = Retry(connect=3, backoff_factor=0.4)
adapter = HTTPAdapter(max_retries=retry)
session.mount('http://', adapter)

def loginuser(username, password):
    userid = session.get(apiurl+'/robo/userid/'+username).json().get('id')
    if userid:
        token = session.post(apiurl+'/login', params={
            'userid': userid,
            'password': password}).json().get('token')
        if not token:
            return False
        with open('.token', 'w') as f:
            f.write(token + '>' + str(userid) + '>' + username)

        return True
    return False



def signupuser(u,p1,p2,i):
    try:
        userid = session.get(apiurl + '/robo/userid/' + u).json().get('id')
        if not userid:
```

```
            if p1 == p2:
                session.post(apiurl + '/robo/adduser', params={
                    'username': u,
                    'password': p1,
                    'initials': i})
                userid = session.get(apiurl + '/robo/userid/' + u).json().get('id')
                token = session.post(apiurl + '/login', params={
                    'userid': userid,
                    'password': p1}).json().get('token')

                with open('.token', 'w') as f:
                    f.write(token + '>' + str(userid) + '>' + u)

                return True
        else:
            return False
    except:
        False

def checkonline():
    try:
        requests.get(apiurl)
        return True
    except requests.exceptions.ConnectionError:
        return False

def addscore(score):
    with open('.token', 'r') as f:
        token, id, _ = f.read().split('>')
    result = session.post(apiurl+'/robo/addscore', params={
        'userid': id,
        'token': token,
        'score': score}).json().get('message')
    return True

def isloggedin():
    try:
        open('.token')
        return True
    except:
        return False
```

characters_module

characters.py

```
from pygame import sprite, image, transform
from characters_module import sprites
from constants.const import *
from characters_module.sprites import stretech_image

class Character(sprite.Sprite):
    """
    This is a very basic character, from which all the other characters will extend, this is never used directly,
    and there will need to be lots of extra functions. This code mostly is needed for the animation and directions
    """
    def __init__(self, sheetname, imagecount=12, scale=30):
        """
        This creates the character, mostly handles grabing the spritesheet, clipping the sprites and scaling them.
```

```
        """
        super().__init__()
        self.sheetname = sheetname
        self.spritesheet = image.load(self.sheetname).convert()
        h,w = self.spritesheet.get_height(), self.spritesheet.get_width()/imagecount
        self.images = [transform.scale(sprite_item, (scale,scale)) for sprite_item in
                       sprites.loadStrip((0, 0, w, h), imagecount, self.spritesheet)]

        self.direction = 'N'
        self.position  = (300,200)
        self.moving = (0,0)
        self.image = self.images[0]
        self.rect = (300,200)

    def setdir(self, mov, dir):
        """
        This sets the current direction (for the spirte animation) based on the where the character is moving and facing
        """
        if dir:
            if mov > 0:
                self.direction = 'E'
            if mov < 0:
                self.direction = 'W'
        else:
            if mov > 0:
                self.direction = 'S'
            if mov < 0:
                self.direction = 'N'


    def onstart(self, view):
        """
        When the character is created, this places it onto the screen, adding some stretch
        """
        view.screen.fill((0, 0, 0))
        img, h = stretech_image(self.images[0], 30-view.tickcounter)
        posx, posy = self.position
        view.screen.blit(img, (posx, posy - h / 2))
```

enemy.py

```

from characters_module.characters import Character
from constants.const import *
import random
import pygame
class Enemy(Character):
    """
    This enemy is again, only used to extend from. It acts as a basic super class which can easily be used to generate
    the other classes for the enemies. Because the enemies need to update in different ways, its not possible to have
    them all exhibit the same behaviour here.
    """
    def __init__(self,sheetname, images=12):
        Character.__init__(self, sheetname, images)
        self.rect = (random.randint(50,SCREENSIZE[0]-50),random.randint(70,SCREENSIZE[1]-50) )

```

```
16
17
18  class Grunt(Enemy):
19      """
20      This is the basic enemy, which is only able to move, and on colliding with
        the player, it kills the player. If it
21      gets hit by a bullet, it dies
22      """
23      def __init__(self):
24          self.sheetname = 'sprites/grunt.png'
25          Enemy.__init__(self, self.sheetname)
26          self.vx = random.randint(-20,20)
27
28          self.vy = random.randint(-20, 20)
29      def update(self, count, movx,movy):
30          """
31          This is a pretty poorly executed AI. I think ill replace this with a
        boids algorithm.
32          """
33
34          self.image = self.images[count]
35          position = self.rect
36          x = movx
37          y = movy
38          legnth = sqrt(x**2 + y**2)
39          adj = legnth / 3
40          newy = y / adj
41          newx = x / adj
42
43          newx = max(50,min(self.rect[0]+newx, SCREENSIZE[0]-50))
44          newy = max(50, min(self.rect[1]+newy, SCREENSIZE[1]-50))
45
46          self.rect = (newx, newy)
47
48  class Electrode(Enemy):
49      """
50      These are the static enemies
51      """
52      def __init__(self):
53          self.sheetname = 'sprites/electrode.png'
54          Enemy.__init__(self, self.sheetname,3)
55          self.image = random.choice(self.images)
56          self.image = pygame.transform.scale(self.image, (20,20))
57      def update(self, count, _):
58          return
59
60
61  class Hulk(Enemy):
62      """
63      These are like the grunts, but cant be killed. They only slow down when hit
        .
64      """
65      def __init__(self):
66          self.sheetname = 'sprites/hulk.png'
67          Enemy.__init__(self, self.sheetname)
68          self.living = 0
69
70      def getskin(self, count):
71          """
72          This is overriding the base function. Hulks always face the same way
73          """
74          if self.velocity[0] < 0:
75              return self.images[:3][count]
```

```
76          elif self.velocity[0] > 0:
77              return self.images[3:6][count]
78          elif self.velocity[1] < 0:
79              return self.images[6:9][count]
80          elif self.velocity[1] > 0:
81              return self.images[9:12][count]
82          else:
83              return self.images[0]
84
85      def update(self, count, _):
86          if not self.living % 25:
87              self.velocity = (random.choice((-3, 3, 0)), random.choice((-3, 3,
    0)))
88          flag = False
89          while not flag:
90              if (35 + BORDER_W < self.rect[1] + self.velocity[1] < SCREENSIZE[1]
     - BORDER_W * 2 - 35):
91                  self.rect = (self.rect[0], self.rect[1] + self.velocity[1])
92                  flag = True
93              else:
94                  self.velocity = (random.choice((-4, 4, 0)), random.choice((-4,
    4, 0)))
95              if (BORDER_W - 20 < self.rect[0] + self.velocity[0] < SCREENSIZE[0]
     - BORDER_W * 2 - 20):
96                  self.rect = (self.rect[0] + self.velocity[0], self.rect[1])
97              else:
98                  self.velocity = (random.choice((-4, 4, 0)), random.choice((-4,
    4, 0)))
99
100         self.living += 1
101
102         self.image = self.getskin(count)
103
104     def kill(self):
105         self.velocity = (random.choice((-2, 2, 0)), random.choice((-2, 2, 0)))
106         flag = False
107         while not flag:
108             if (35 + BORDER_W < self.rect[1] + self.velocity[1] < SCREENSIZE[1]
     - BORDER_W * 2 - 35):
109                 self.rect = (self.rect[0], self.rect[1] + self.velocity[1])
110                 flag = True
111             else:
112                 self.velocity = (random.choice((-3, 3, 0)), random.choice((-3,
    3, 0)))
113             if (BORDER_W - 20 < self.rect[0] + self.velocity[0] < SCREENSIZE[0]
     - BORDER_W * 2 - 20):
114                 self.rect = (self.rect[0] + self.velocity[0], self.rect[1])
115             else:
116                 self.velocity = (random.choice((-3, 3, 0)), random.choice((-3,
    3, 0)))
117
118 class Brain(Enemy):
119     def __init__(self):
120         self.sheetname = 'sprites/brain.png'
121         Enemy.__init__(self, self.sheetname)
122
123
124 class Spheroids(Enemy):
125     def __init__(self):
126         self.sheetname = 'sprites/spheroids.png'
127         Enemy.__init__(self, self.sheetname, 8)
128
129
```

```
130 class Quarks(Enemy):
131     def __init__(self):
132         self.sheetname = 'sprites/quark.png'
133         Enemy.__init__(self, self.sheetname, 8)
134 
135 
136 class Enforcer(Enemy):
137     def __init__(self):
138         self.sheetname = 'sprites/enforcer.png'
139         Enemy.__init__(self, self.sheetname, 6)
140         self.image = self.images[1]
141 
142 class Tank(Enemy):
143     def __init__(self):
144         self.sheetname = 'sprites/tank.png'
145         Enemy.__init__(self, self.sheetname, 4)
```

humans.py

```
1  from characters_module.characters import Character
2  import random
3  from constants.const import *
4  import time
5  class Human(Character):
6      """
7      The base Human class.  Because the behaviour is so similar here (they all
       have the same movement and actions) so
8      they can all extend from a very very basic class.
9      """
10     def __init__(self,sheetname, images=12):
11         Character.__init__(self, sheetname, images, 60)
12         self.rect = self.image.get_rect()
13         self.rect.center = (random.randint(50, SCREENSIZE[0] - 50), random.
       randint(70, SCREENSIZE[1] - 50))
14 
15 
16     def update(self,count, _):
17         if not self.living % 25 :
18             self.velocity = (random.choice((-4, 4,0)), random.choice((-4, 4,0))
       )
19         flag = False
20         while not flag:
21             if (35+BORDER_W < self.rect[1] + self.velocity[1] < SCREENSIZE[1]-
       BORDER_W*2-35):
22                 self.rect = (self.rect[0], self.rect[1]+self.velocity[1])
23                 flag = True
24             else:
25                 self.velocity = (random.choice((-4, 4, 0)), random.choice((-4,
       4, 0)))
26             if (BORDER_W-20 < self.rect[0] + self.velocity[0] < SCREENSIZE[0]-
       BORDER_W*2 -20):
27                 self.rect = (self.rect[0]+self.velocity[0], self.rect[1])
28             else:
29                 self.velocity = (random.choice((-4, 4, 0)), random.choice((-4,
       4, 0)))
30 
31         self.living += 1
32 
33         self.image = self.getskin(count)
34 
35     def die(self, view):
36         value = self.value
37         somewords = view.minifont.render(
38             self.value,
```

```
            True,
            (246, 130, 20))
        view.screen.blit(somewords, self.rect)
        time.sleep(0.05)
        self.kill()
        return int(value)

    def getskin(self, count):
        if self.velocity[0] < 0:
            return self.images[:3][count]
        elif self.velocity[0] > 0:
            return self.images[3:6][count]
        elif self.velocity[1] < 0:
            return self.images[6:9][count]
        elif self.velocity[1] > 0:
            return self.images[9:12][count]
        else:
            return self.images[0]



class Mommies(Human):
    def __init__(self):
        self.sheetname = 'sprites/mommies.png'
        self.living = 0
        self.velocity = (random.randint(-4, 4), random.randint(-4, 4))
        self.value = '1000'
        Character.__init__(self, self.sheetname)




class Daddies(Human):
    def __init__(self):
        self.sheetname = 'sprites/daddies.png'
        self.living = 0
        self.velocity = (random.randint(-4, 4), random.randint(-4, 4))
        self.value = '1000'
        Character.__init__(self, self.sheetname)



class Mikeys(Human):
    """
    These are the 'kids' - i made them move slower
    """
    def __init__(self):
        self.sheetname = 'sprites/mikeys.png'
        self.living = 0
        self.velocity = (random.randint(-3, 3), random.randint(-3, 3))
        self.value = '1000'
        Character.__init__(self, self.sheetname)
```

player.py

```
from characters_module.characters import Character
from constants.const import *


class Player(Character):
    """
    Because most of the logic about whether a player is alive and the score is handled by the Model, most of it
    can be abstracted away. This class mostly handles the player screen logic,
```

```
    and doesnt look ay the logic of whether
    or not the player is alive.
    """
    def __init__(self):
        self.sheetname = 'sprites/player.png'
        Character.__init__(self, self.sheetname)
        self.l_images = self.images[:3]
        self.r_images = self.images[3:6]
        self.f_images = self.images[6:9]
        self.u_images = self.images[9:12]

    def getskin(self, count):
        if self.direction[0] == 'N':
            return self.u_images[count]
        elif self.direction[0] == 'S':
            return self.f_images[count]
        elif self.direction[0] == 'W':
            return self.l_images[count]
        elif self.direction[0] == 'E':
            return self.r_images[count]

    def movy(self, newMov):
        if (25+BORDER_W < self.position[1] + newMov < SCREENSIZE[1]-BORDER_W
    *2-30) :
            self.position = (self.position[0], self.position[1] + newMov)
        self.setdir(newMov, 0)

    def movx(self, newMov):
        if (BORDER_W-10 < self.position[0] + newMov < SCREENSIZE[0]-BORDER_W*2
    -10):
            self.position =  (self.position[0]+newMov,self.position[1])
        self.setdir(newMov, 1)
```

sprites.py

```
import pygame
from constants import colors as COLS


def getImage(sheet, rectangle):
    """ Grab a single image out of a larger spritesheet
        Pass in the x, y location of the sprite
        and the width and height of the sprite. """
    rect = pygame.Rect(rectangle)

    # Create a new blank image
    image = pygame.Surface(rect.size).convert()

    # Copy the sprite from the large sheet onto the smaller image
    image.blit(sheet, (0, 0), rect)

    # Assuming black works as the transparent color
    image.set_colorkey(COLS.BLACK)

    # Return the image
    return image



def stretech_image(imagenmame, progression, rect=None):
    """
    This function is to strech out an image, where the progression is a value
    which defines how far along in the
    process of the strech it is (the stretch is non linear)
```

```
29
30      imagename could be a string, or could be an instance of an image
31      :param imagenmame:
32      :param progression:
33      :param rect:
34      :return:
35      """
36
37      if isinstance(imagenmame, str):
38          sheet = pygame.image.load(imagenmame).convert()
39          h, w = sheet.get_height(), sheet.get_width()
40          image = pygame.transform.scale(sheet, (w, h + progression ** 2))
41          image.set_colorkey(COLS.BLACK)
42          return image, h + progression ** 2
43
44      elif rect is not None:
45          sheet = pygame.image.load(imagenmame).convert()
46          image = getImage(sheet, rect)
47          h, w = image.get_height(), image.get_width()
48          image = pygame.transform.scale(image, (w, h + progression ** 2))
49          return image, h + progression ** 2
50      else:
51          # This condiiton handles imagename not being an imagename, but rather
    an object of type image already.
52          h, w = imagenmame.get_height(), imagenmame.get_width()
53          return pygame.transform.scale(imagenmame, (w, h + progression ** 2)), h
     + progression ** 2
54
55  # Load a whole bunch of images and return them as a list
56  def getImages(sheet, rects):
57      "Loads multiple images, supply a list of coordinates"
58      return [getImage(sheet, rect) for rect in rects]
59
60  # Load a whole strip of images
61  def loadStrip(rect, image_count, sheet):
62      "Loads a strip of images and returns them as a list"
63      tups = [(rect[0]+rect[2]*x, rect[1], rect[2], rect[3])
64              for x in range(image_count)]
65      return getImages(sheet, tups)
```

constants

colors.py

```
1   """
2   This is just a selection of constants as colours
3   """
4
5
6   BLACK = (0, 0, 0)
7   GREY = (61, 61, 61)
8   BROWN = (40, 28, 14)
9   PURPLE = (33, 19, 52)
10  GREEN = (23, 40, 19)
11  LIGHTGREY = (70, 70, 70)
12  TEAL = (18, 51, 54)
13  YELLOW = (85, 80, 52)
14  RED = (76, 14, 33)
15  WHITE = (255,255,255)
16  BULLETS = [
17      GREY,
18      BROWN,
19      PURPLE,
```

```
    GREEN,
    LIGHTGREY,
    TEAL,
    YELLOW,
    RED
]

random_colors = [
    '#281ed5',
    '#c79e32',
    '#661b61',
    '#918738',
    '#a98996',
    '#6b9362',
    '#77cc12',
    '#45e61a',
    '#c1656b',
    '#9e8dcb',
    '#141110',
    '#e537d8',
    '#e6db9e',
    '#f4ece7',
    '#2b6b3c',
    '#2c1873',
    '#34179f',
    '#f3e044',
    '#9442ca',
    '#b8268f',
    '#dd250d',
    '#25174d',
    '#78c869',
    '#d66d47',
    '#ea5e97',
    '#68250b',
    '#ac5e27',
    '#8e1c3d',
    '#ed6209',
    '#c32463',
    '#6139cc',
    '#6dc947',
    '#c243d4',
    '#6531a0',
    '#d63b14',
    '#a9409b',
    '#3fb86a',
    '#6b5d60',
    '#b0cf25',
    '#70c091',
    '#934039',
    '#923df4',
    '#ae5d96',
    '#ec9bd8',
    '#cd2440',
    '#c47415',
    '#ec8312',
    '#44120c',
    '#f1e80d',
    '#ed5d6a',
    '#f1dd73',
    '#e7622a',
    '#986bca',
    '#bbcbdb',
    '#3ca138',
```

```
    '#e7bde2',
    '#ac232d',
    '#101f9c',
    '#326996',
    '#4ca8b1',
    '#821cbc',
    '#94c5e1',
    '#4c7cbc'
]

title_colors = [
    (249,52,242),
    (0,0,0),
    (32,28,208),
    (249,36,4),
    (255,255,255)
]
edge = [
    (180,46,38),
    (249,46,0),
    (255, 130, 45),
    (255, 130, 45)
]
```

const.py

```
"""
These constants control game play. Most of these constants are adjustable and
    will adapt automatically for the game.
"""

from math import sqrt

VELOCITY = 6
DVELOCITY = sqrt(2*(VELOCITY**2))
SCREENSIZE = (800,600)
TPS = 28
TITLE = 'Robotron 2084'
PROJ_VELOCITY = 8
DPROJ_VELOCITY = sqrt(2*(PROJ_VELOCITY**2))
COOLDOWN = 5
BORDER_W = 10
BORDERSPEED = 15
```

decorations

border.py

```
import pygame
from constants.const import *
from constants.colors import *


class Border(pygame.sprite.Sprite):
    """
    This border class is drawn around the edge of the screen. it is a sprite
    itself, so it cannot be killed
    """

    def __init__(self):
        pygame.sprite.Sprite.__init__(self)
        self.color = (255,0,0)
        self.image = pygame.Surface((SCREENSIZE[0], SCREENSIZE[1]))
```

```
        self.image.fill(BLACK)
        self.image.set_colorkey(BLACK)

        self.drawrect()
        self.rect.center = (SCREENSIZE[0]/2, SCREENSIZE[1]/2)

    def update(self,_,__):

        r,g,b = self.color
        if r>0 and b == 0:
            r -= BORDERSPEED
            g += BORDERSPEED
        if g > 0 and r == 0:
            g -= BORDERSPEED
            b += BORDERSPEED
        if b > 0 and g == 0:
            b -= BORDERSPEED
            r += BORDERSPEED
        self.color = (r,g,b)
        self.drawrect()

    def drawrect(self):
        self.lines = [
            pygame.draw.line(self.image, self.color, [0, 30], [SCREENSIZE[0], 30], BORDER_W),
            pygame.draw.line(self.image, self.color, [0, SCREENSIZE[1]], [SCREENSIZE[0], SCREENSIZE[1]], BORDER_W*2),
            pygame.draw.line(self.image, self.color, [0, 30], [0, SCREENSIZE[1]], BORDER_W*2),
            pygame.draw.line(self.image, self.color, [SCREENSIZE[0], 30], [SCREENSIZE[0], SCREENSIZE[1]], BORDER_W*2)
]

        self.rect = self.image.get_rect()

    def die(self):
        return
```

objects

bullet.py

```
import pygame
from constants.colors import BULLETS, BLACK
from random import choice
from constants.const import PROJ_VELOCITY, DPROJ_VELOCITY, SCREENSIZE, BORDER_W
from playsound import playsound
class Bullet(pygame.sprite.Sprite):
    """
    A classs for the bullets
    """
    def __init__(self, x, y, dir):
        playsound('./audio/shoot.mp3', block=False)
        pygame.sprite.Sprite.__init__(self)
        self.color = choice(BULLETS)
        self.dir = dir
        self.movx = 0
        self.movy = 0



        v = PROJ_VELOCITY if len(dir)<1 else DPROJ_VELOCITY

```

```
        if 'N' in dir:
            self.movy = -v
        elif 'S' in dir:
            self.movy = v

        if 'W' in dir:
            self.movx = -v
        elif 'E' in dir:
            self.movx = v

        kill = 0
        if len(dir)==1 and (dir=='N' or dir=='S'):
            rotation = 90
        elif len(dir)==1 and (dir=='E' or dir=='W'):
            rotation = 0
        elif 'NE'==dir or 'SW'==dir or 'EN'==dir or 'WS'==dir:
            rotation = 45
        elif 'NW'==dir or 'WN'==dir or 'SE'==dir or 'ES'==dir:
            rotation = 315
        else:
            kill = 1
            rotation = 0

        self.image = pygame.Surface([25, 8])
        self.image.fill(BLACK)
        self.image.set_colorkey(BLACK)

        pygame.draw.rect(self.image, self.color, pygame.Rect(0, 0, 25, 5),
    border_radius=3)
        self.image = pygame.transform.rotate(self.image, rotation)

        self.rect = self.image.get_rect()
        self.rect.center = (x,y)

        if kill:
            self.kill()
            del self



    def update(self, _, __):
        self.rect.center = self.rect.center[0] + self.movx, self.rect.center[1]
     + self.movy
        if (not (SCREENSIZE[1]-BORDER_W-30 >= self.rect.y >= 30 + BORDER_W) )
    or\
                (not (SCREENSIZE[0]-BORDER_W*4 -5 >= self.rect.x >= BORDER_W*2
    - 5)):
            self.kill()
            del self
            return
```