



Robotron: 2084 inspired Game
Written in PyGame with MVC architecture

John Montgomery
Candidate Number: 5199
Centre Name: Kimberley College
Centre Number: 15162
Qualification: AQA 7517 (A-Level Computer Science)

Supervisor: B. Harris

2020-2022

Contents

1	Abstract	3
2	Analysis	4
2.1	Introduction	4
2.2	About the game	4
2.3	More Modern 2D games	5
2.4	Robotron's Design	5
2.5	Robotron Step by Step	5
2.6	Objectives	6
2.7	What is MVC?	7
2.8	The Game	8
2.9	Limitations	8
2.10	Objectives	8
2.11	Design and Inspiration	9
3	Documented Design	11
3.1	MVC in practice	11
3.2	Boids	11
3.3	Database	12
3.4	The API	13
3.5	The Server Setup	13
3.6	Security	15
4	Technical Solution	16
4.1	Boids	16
5	Testing - TODO	18
6	Evaluation - TODO	19
7	Appendix & Bibliography	20
7.1	Appendix	20
7.2	Bibliography	20
7.3	Files and Listings	20
7.3.1	Website Code	21
7.3.2	Game Code	34

List of Figures

2.1	A diagram showing the MVC architecture	7
2.2	Screen from original game - https://arcadeblogger.com/2020/06/27/the-development-of-robotron/	
2.3	Advertising Material - https://arcadeblogger.com/2020/06/27/the-development-of-robotron/	
3.1	Class diagram	12
3.2	Class diagram	12
3.3	Class diagram	13
3.4	Class diagram of characters	13
3.5	Flowchart of MVC	14
3.6	How tokens are generated	14

Chapter 1

Abstract

This project is an implementation of the classic arcade game Robotron 2084. It will consist of 2 main parts:

The game code itself, which is pygame executed on the client machine, consisting of pygame, with an Model-View-Controller architecture driving it. This means the game consists of 3 discrete sections, the model controlling the games current state, the view controls what is displayed and the controller is what the user interacts with. The game changes through its various states with a finite state machine controlling it. A stack is also used which stores incoming events and future states.

There is also a website which displays the top 10 users and their scores. This data is sent from the game code (when the client is online) via an API, users must also login and bcrypt is used to secure thier password. A secure token is generated and stored to verify users are logged in.

This project is a modernised version of robotron 2084, it has 2 sections, the game code and the website. The game code, which is a remake of Robotron 2084, a classic arcade game, this is backed with an MVC architecture which is designed to separate out the functionalities of the game and communicates with the website via an API. The website will use a database backend to store all the scores and logins and will also be used to display the high scores of users.

Chapter 2

Analysis

2.1 Introduction

The goal of this project was to create a more modernised version of Robotron: 2084, the classic arcade game from the 80s. In order to make it a more modern version I will make a few additions, including improving the game itself and increase the competition aspect of it. I also needed to modernise the codebase itself, and couldn't build off the existing code written in the 80s.

As i was unable to access the code from the original game, nor find a viable method to emulate the code on my machine, i was forced to take my research from websites, videos and images, and whilst this isn't ideal, i was still able to gain a vast amount of information and replicate the game how i wanted to.

Whilst there is no specific target audience for the project, it could be played and enjoyed by anyone, ranging from my classmates to the people who played and enjoyed the original game from the 80s. Thanks to the wide range of possible users, there is a large market i can test this app with.

2.2 About the game

Robotron 2084 is a classic, top down, arcade game from the 80's, in which, a player (who is a mutant genetic super hero) attempts to save the last human family from swarms of killer robots. The game was a 2 stick shooter originally - this means 2 joysticks, one to move and one to shoot (this allows the 2 to occur individually and simultaneously). This was one of the first of its kind, and was largely considered a success. The game has a number of waves with varying number of robots, of different types, and varying numbers of humans.

In order to 'save' a human, the player simply has to touch them, this rescues the human and scores the player points. The more humans saved, the higher the points. The player, whilst a mutant, is still susceptible to damage - and whilst there is no 'health' the player can be killed rather easily. The robots simply have to touch the player (or the player accidentally collide with them) and the player 'dies'. You start the game with 3 lives and slowly progress through the waves - a new wave starts when all of the grunts (one of the robots, see table below) are killed, or when the player dies.

There are also various transition screens, including a boot screen, a testing screen and end screens. Along with these come a live counter, a score counter and flashing borders. The game is fast paced and bright, and graphically complex, with lots of colours and intense action. Making a game like this play as smoothly as they did at the time was truly an accomplishment.

The game even had sound! whilst it was only mono aural (no stereo) - this was still impressive, considering it was a game running on a 1MHz processor. They were able to develop this in a 2 man team over a period of 6 months. It was itself heavily influenced by a number of other games, including 'Berzerk' - a shooting game which players traverse a maze and shoot at enemies. However, this was a single stick shooter (with a button to fire) rather than a dual stick shooter.

The game had sequels but none were as successful as the original, and were never received as well. One even attempted a multiplayer system, where one would shoot and another would move, but this was not widely seen as a positive update.

2.3 More Modern 2D games

In order to bring this game into the 21st century, it will need more modernised features. For one, almost everyone plays games on computers, and as such, will need a 'dual stick' implementation. Almost every modern game uses WASD to move and arrows to shoot, but IJKL is also used to shoot. Other approaches to avoid seeming like a single stick shooter are using the mouse to control movement or shooting, or even ESDX and IJKM as move and shoot (this was the original set up for Robotron 2084 on apple products).

2.4 Robotron's Design

I mostly used video for my research into robotrons design - this felt like the best way due to how fast the game changes, with fast animations and colour changes, something that cannot be captured easily in photo screenshots. I also relied on websites, also linked, for more technical information and this is where i was also able to learn lots of the history behind the game.

- <https://www.youtube.com/watch?v=ccltMtkFBSI> - This video - impressive in itself for the gameplay, gave me more of an insight into the feel of the past paced nature of the game, and whilst the quality isnt perfect, it gives some idea of the general layout of the screen and game
- <https://www.youtube.com/watch?v=a0VA2Axxfdk> - This video was much higher quality screen capture of the game, this is what allowed me to see animation and character design as well as have a very good understanding of what layouts looked like, and the general interface.
- <https://arcadeblogger.com/2020/06/27/the-development-of-robotron/> This was possibly my most used resource. Having insight into the development and original views of the game was very important, and this website alone would probably have been enough to implement it.

Whilst i didn't want to build a carbon copy of the original, i did endeavour to create something as similar as i could without losing too much of the fast paced nature of the game.

2.5 Robotron Step by Step

Robotron starts up with a pattern of random pixels, followed by a screen indicating that tests were successful (or, as it may be, unsuccessful), before finally transitioning to the start screen. This screen has the words Robotron, along with credits to the developer. It is very bright, and flashing. The game also has a high score board displaying at the end of every run (and i believe

it would also display at times when the machine was not in game). When starting the game, it would show a level transition screen (bright rectangles would display from the centre of the screen moving out). A wave would then start. This would begin by displaying all character, but not allowing movement of enemies for a few moments (this is probably a design choice, a brief moment for the player to plan their first few movements, and analyse where they should head). The levels would then loop like this, with a transition showing between each.

2.6 Objectives

1 Game Objectives

- 1.1 Main (hero) character objectives
 - 1.1.1 Character can be displayed
 - 1.1.2 Character can move in all 8 directions
 - 1.1.3 Character faces in correct direction
 - 1.1.4 Characters movement is animated
 - 1.1.5 Character is bounded to window
 - 1.1.6 Character can shoot in 8 directions
 - 1.1.7 Player is invincible on load of level
- 1.2 Enemy character objectives (for each enemy type - Electrodes, Grunts and Hunks)
 - 1.2.1 Enemies can be displayed
 - 1.2.2 Enemies can move
 - 1.2.3 Enemy faces correct direction
 - 1.2.4 Enemies movement is animated
 - 1.2.5 Enemy can move
 - 1.2.6 Enemy kills player when touching
 - 1.2.7 Specific enemy functionality
 - 1.2.7.1 Electrodes are randomly spread around the page
 - 1.2.7.2 Use boids flocking algorithm to dictate movement of grunts
 - 1.2.7.3 Hunks slow down when shot
- 1.3 Menu objectives
 - 1.3.1 Logo is shown and animated
 - 1.3.2 Display static text
 - 1.3.3 Display animated text
 - 1.3.4 Display and allow input for options
 - 1.3.5 Allow for login and signup
 - 1.3.6 Show help menu
- 1.4 Decorations objectives
 - 1.4.1 Flashing border
 - 1.4.2 Random colour load screen
 - 1.4.3 "All tests" screen
 - 1.4.4 Inter—level animation

2 Website Objectives

- 2.1 Displays high score board of top 10 players
- 2.2 Website animated and looks like the high score board on original game
- 2.3 Have an error page - just incase
- 2.4 API Goals (a route to...)
 - 2.4.1 return top scores
 - 2.4.2 allow for sign up
 - 2.4.3 allow for log in
 - 2.4.4 generate tokens for login

2.4.5 upload scores and validate with token

2.7 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.

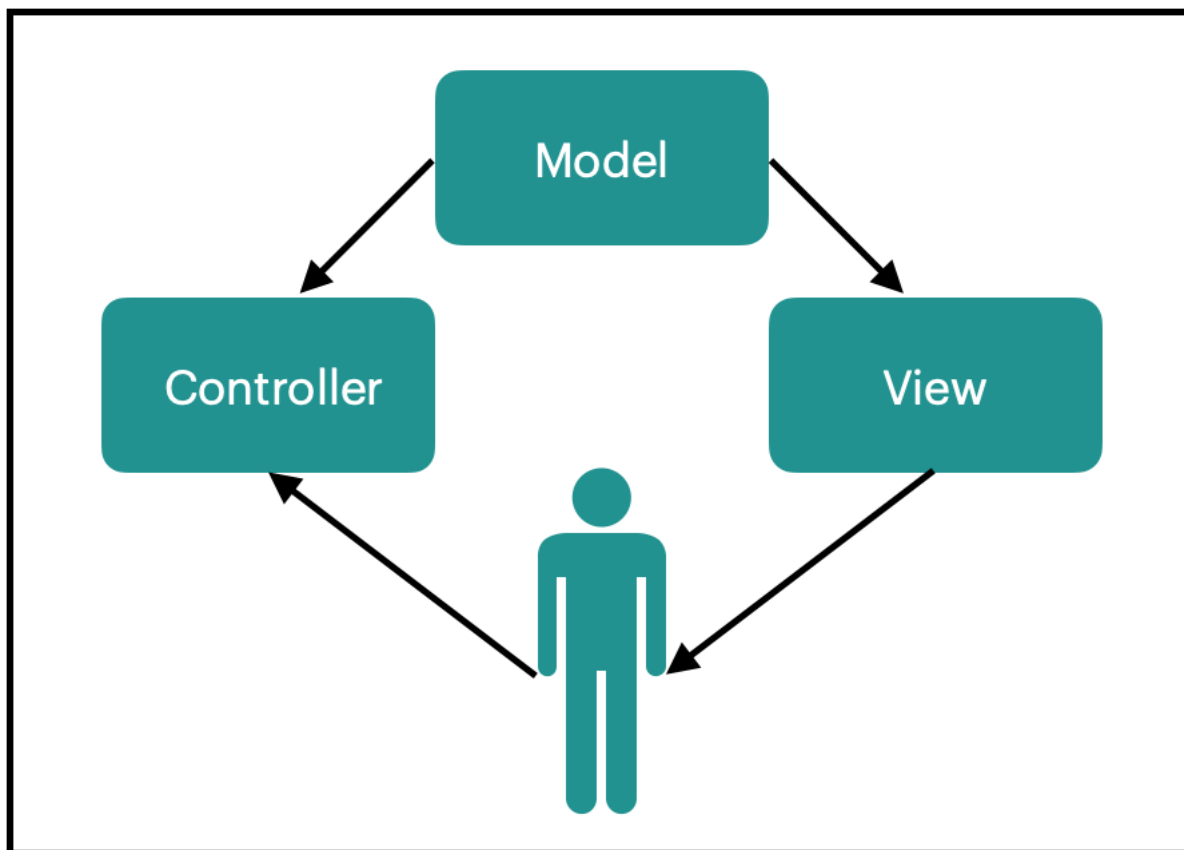


Figure 2.1: A diagram showing the MVC architecture

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.

2.8 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.

2.9 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.

2.10 Objectives

1 Game Objectives

- 1.1 Main (hero) character objectives
 - 1.1.1 Character can be displayed
 - 1.1.2 Character can move in all 8 directions
 - 1.1.3 Character faces in correct direction
 - 1.1.4 Characters movement is animated
 - 1.1.5 Character is bounded to window
 - 1.1.6 Character can shoot in 8 directions
 - 1.1.7 Player is invincible on load of level
- 1.2 Enemy character objectives
 - 1.2.1 Enemies can be displayed
 - 1.2.2 Enemies can move
 - 1.2.3 Enemy faces correct direction
 - 1.2.4 Enemies movement is animated
 - 1.2.5 Enemy can move
 - 1.2.6 Enemy kills player when touching
 - 1.2.7 Specific enemy functionality
 - 1.2.7.1 Electrodes are randomly spread around the page
 - 1.2.7.2 Use boids flocking algorithm to dictate movement of grunts
 - 1.2.7.3 Hunks slow down when shot
- 1.3 Menu objectives
 - 1.3.1 Logo is shown and animated
 - 1.3.2 Display static text
 - 1.3.3 Display animated text
 - 1.3.4 Display and allow input for options
 - 1.3.5 Allow for login and signup

- 1.3.6 Show help menu
- 1.4 Decorations objectives
 - 1.4.1 Flashing border
 - 1.4.2 Random colour load screen
 - 1.4.3 "All tests" screen
 - 1.4.4 Inter—level animation

2 Website Objectives

3

1. Create the API
2. Create login system
 - (a) Basic API sign up works
 - (b) GUI interactions with PyGame
3. High Scores
 - (a) Top 10
 - (b) Player Search
4. Create sounds with Game
5. Create scoring and score counter
6. Create a life counter
7. Automate testing on API and basic functions in PyGame

2.11 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>

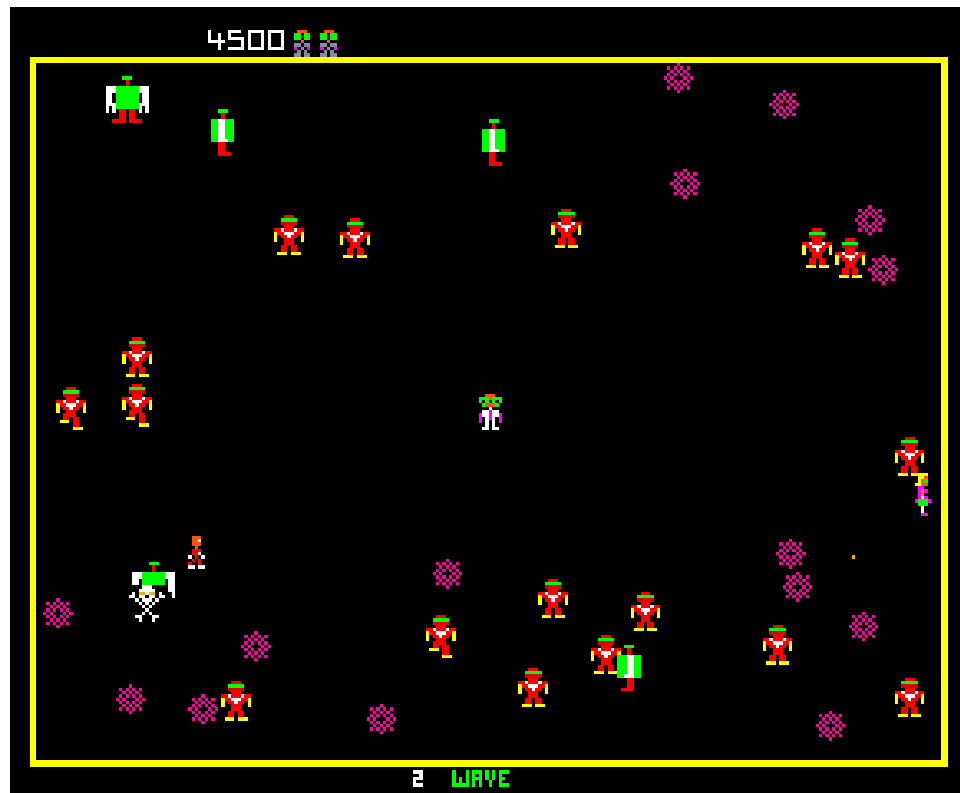


Figure 2.2: Screen from original game - <https://arcadeblogger.com/2020/06/27/the-development-of-robotron/>

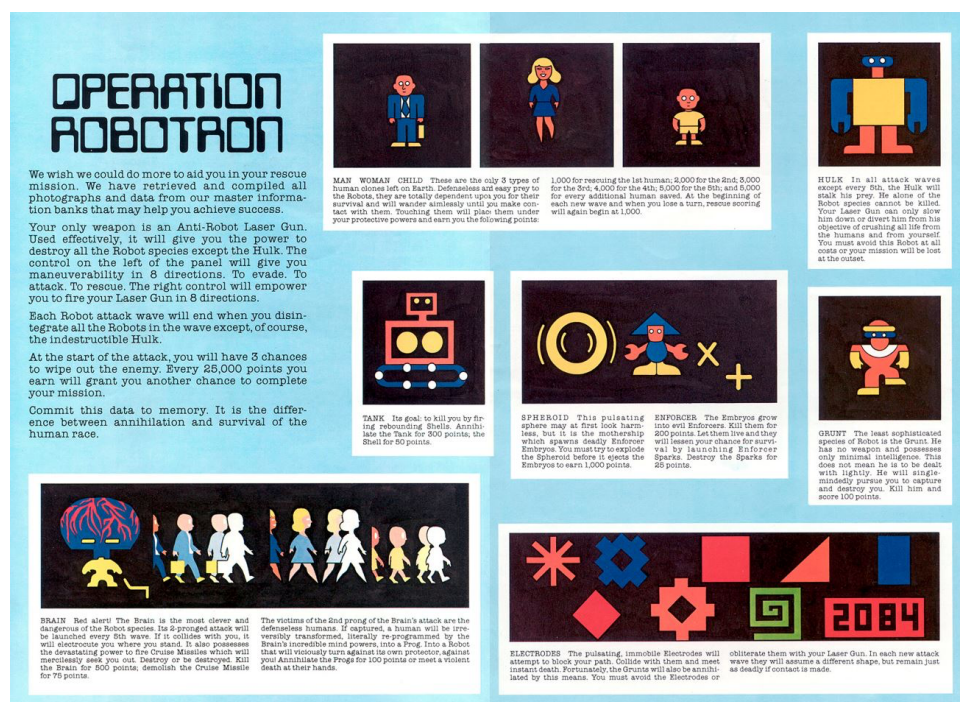


Figure 2.3: Advertising Material - <https://arcadeblogger.com/2020/06/27/the-development-of-robotron/>

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.

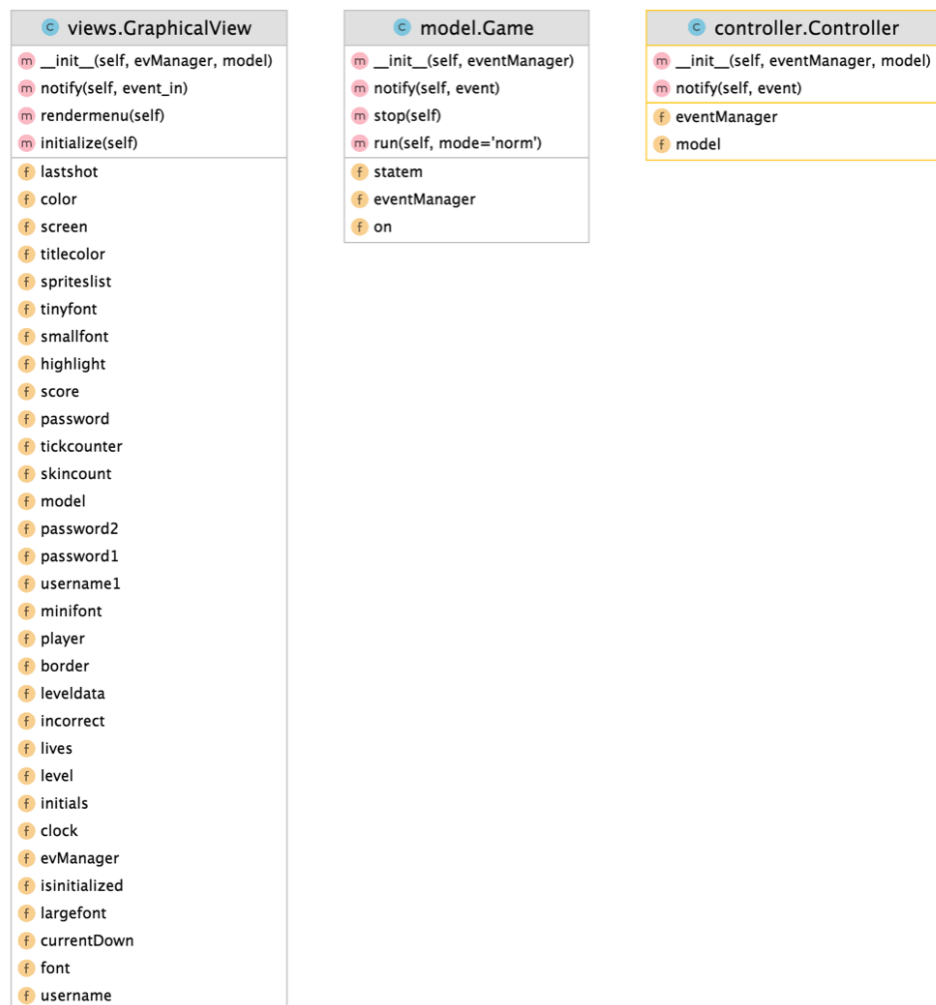


Figure 3.1: Class diagram

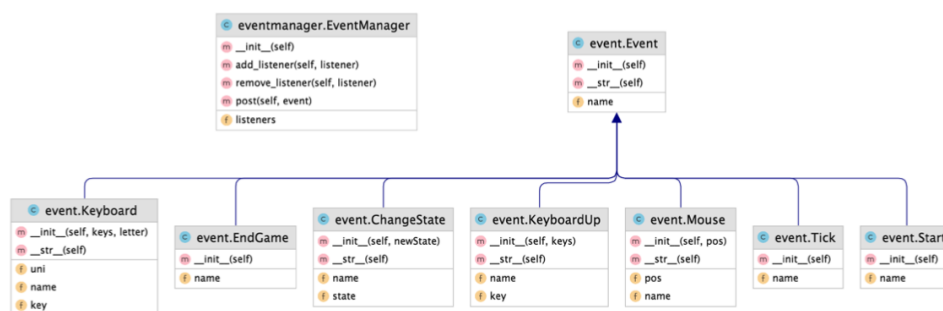


Figure 3.2: Class diagram

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

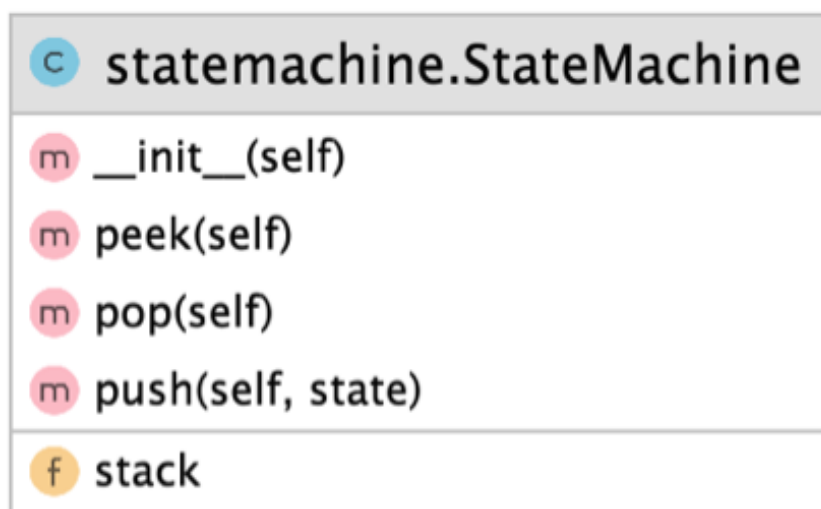


Figure 3.3: Class diagram

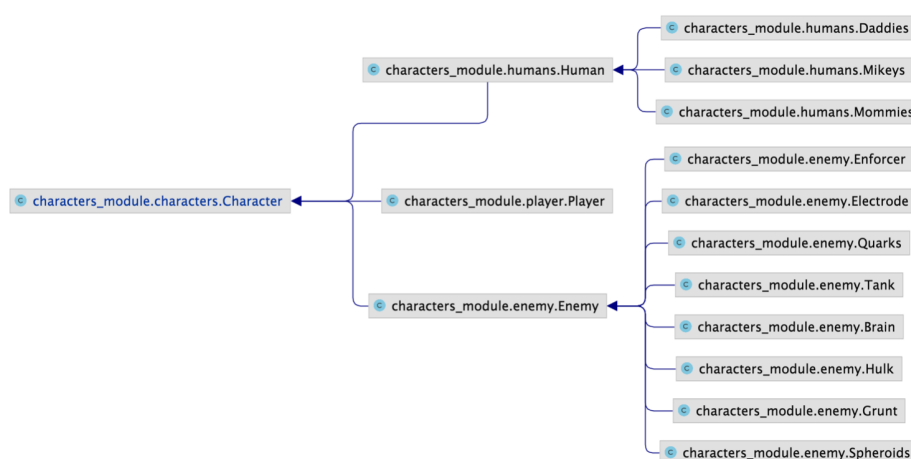


Figure 3.4: Class diagram of characters

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

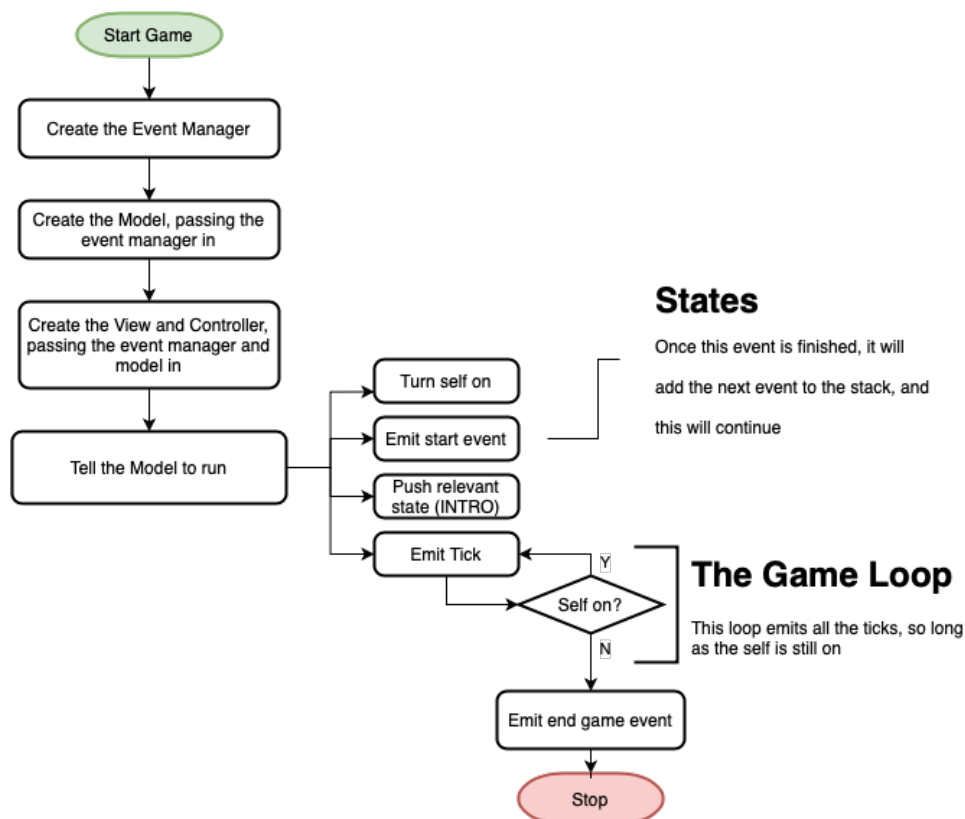


Figure 3.5: Flowchart of MVC

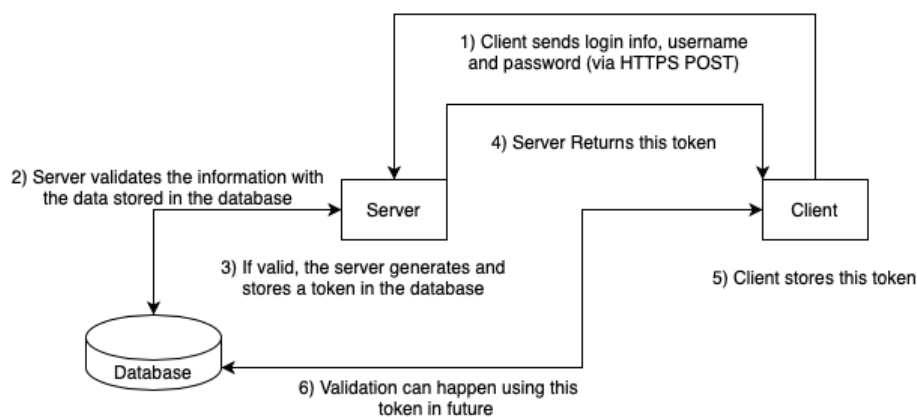


Figure 3.6: How tokens are generated

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

```

1      def boids(x, gruntlist, playerpos):
2
3          gruntlist = list(gruntlist)
4
5          xtot, ytot = 0,0
6          c1,c2 = 0,0
7          v1,v2 = 0,0
8
9          x1,y1 = x.rect[0], x.rect[1]
10         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)

```

1      for grunt in gruntlist:
2          x2,y2 = grunt.rect[0], grunt.rect[1]
3
4
5          xtot += x2
6          ytot += y2

```

If the boid is in sight then we update our values

```

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

```

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

```

1          v1 += grunt.vx
2          v2 += grunt.vy
3
4          p1 = (playerpos[0]-x1) /5
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 Bibliography

Bibliography

7.3 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

"Game Code/gameplay.py"	16
"Game Code/gameplay.py"	16
"Game Code/gameplay.py"	16
"Game Code/gameplay.py"	16
"Game Code/gameplay.py"	17
"Game Code/gameplay.py"	17
"Website Code/app.py"	21
"Website Code/templates/index.html"	25
"Website Code/templates/error.html"	30
"Website Code/static/css/styles.css"	32
"Game Code/main.py"	34
"Game Code/eventmanager.py"	34
"Game Code/statemachine.py"	35
"Game Code/model.py"	35
"Game Code/views.py"	36
"Game Code/controller.py"	38
"Game Code/event.py"	39
"Game Code/states.py"	40
"Game Code/menu.py"	41
"Game Code/gameplay.py"	51
"Game Code/APIinteractions.py"	55
"Game Code/characters_module/characters.py"	56
"Game Code/characters_module/enemy.py"	57
"Game Code/characters_module/humans.py"	60
"Game Code/characters_module/player.py"	61
"Game Code/characters_module/sprites.py"	62
"Game Code/constants/colors.py"	63
"Game Code/constants/const.py"	65
"Game Code/decorations/border.py"	65
"Game Code/objects/bullet.py"	66

7.3.1 Website Code

app.py

```

1
2 # Flask is used to handle the web requests
3 from flask import Flask, jsonify, request, render_template
4
5 # Sql alchemy handles all SQL interactions, but rather than using and overly
   relying on the ORM,
6 # Ill use raw SQL commands. The SQL server is running on RDS (AWS) with
   PostgreSQL

```

```

7 from sqlalchemy import create_engine
8 from sqlalchemy.orm import scoped_session, sessionmaker
9
10 # Allow CORS - so it will work from both the webserver and python
11 from flask_cors import CORS
12 from waitress import serve
13 # This is used to hash passwords and validate them - could of used a different
    tool, or built it myself, but
14 # But this is prebuilt and purpose designed
15 import bcrypt
16 import secrets
17 # This starts the App
18 app = Flask(__name__)
19 # Allow the cors to work
20 CORS(app)
21 # Gets the database URL, creates the connection
22
23 engine = create_engine(
24     'sqlite:///test.db',
25     connect_args={'check_same_thread': False}
26 )
27
28 db = scoped_session(sessionmaker(bind=engine))
29
30
31 db.execute('''
32 CREATE TABLE IF NOT EXISTS leaderboard (
33     id INTEGER UNIQUE PRIMARY KEY AUTOINCREMENT,
34     initials VARCHAR(255),
35     username VARCHAR(255) UNIQUE,
36     password VARCHAR(255)
37 )''')
38 db.commit()
39 db.execute('''
40 CREATE TABLE IF NOT EXISTS scores(
41     id INT,
42     scores INT
43 )
44 ''')
45 db.commit()
46 db.execute('''
47 CREATE TABLE IF NOT EXISTS tokens (
48     id INT,
49     token VARCHAR
50 )
51 ''')
52 db.commit()
53
54 @app.route('/test', methods=['GET'])
55 def test():
56     return render_template('error.html')
57
58 @app.route('/', methods=['GET'])
59 def index():
60     leaders = db.execute('''SELECT leaderboard.initials, scores
61 FROM leaderboard
62 LEFT JOIN scores
63 ON leaderboard.id = scores.id
64 ORDER BY scores DESC
65 LIMIT 10;''')
66     # ...so we convert it into a dictionary
67     a, d = [], {}
68     for lead in leaders:

```

```

69         a.append({"initials":lead[0],"scores":lead[1] })
70     return render_template('index.html', a=a)
71
72
73 @app.errorhandler(500)
74 def page_not_found(e):
75     # note that we set the 404 status explicitly
76     return render_template('error.html')
77
78
79 @app.route('/robo/leaderboard', methods=['GET'])
80 def leader():
81     """
82     This route fetches the top 10 results from the server, allowing the page to
83     display the leaderbaord
84     :return:
85     """
86     # This returns a Result Proxy object...
87     leaders = db.execute('''SELECT leaderboard.initials, scores
88 FROM leaderboard
89 LEFT JOIN scores
90 ON leaderboard.id = scores.id
91 ORDER BY scores DESC
92 LIMIT 10;''')
93     # ...so we convert it into a dictionary
94     a , d= [], {}
95     for lead in leaders:
96         for column, value in lead.items():
97             d = {**d, **{column: value}}
98     a.append(d)
99
100     return jsonify(a)
101
102 @app.route('/robo/user/<string:userid>', methods=['GET'])
103 def user(userid):
104     """
105     This returns a users high score, given their ID - this means that the API
106     will have to fetch the ID first
107     Could it have used the username? probably.
108     :param userid:
109     :return:
110     """
111     score = list(db.execute(f'''SELECT score
112 FROM leaderboard
113 LEFT JOIN scores
114 ON leaderboard.id = scores.id
115 WHERE leaderboard.id = {userid}
116 ORDER BY scores DESC
117 LIMIT 1;'''))[0][0]
118     return jsonify({'score': score})
119
120 @app.route('/robo/userid/<string:username>', methods=['GET'])
121 def useridget(username):
122
123     """
124     This is used to get the id of a user, from their username (which has to be
125     unique)
126     Returns a 0 if the username is not unique
127     :param username:
128     :return:
129     """

```



```

129     userid = list(db.execute(f"""SELECT leaderboard.id
130 FROM leaderboard
131 WHERE leaderboard.username = '{username}'
132 LIMIT 1;"""))
133     try:
134         print(userid)
135         return jsonify({'id': userid[0][0]})
136     except IndexError:
137         return jsonify({'id': 0})
138
139
140 @app.route('/login', methods=['POST'])
141 def login():
142     """
143     Used to login to the game, returns a token which is used to verify the
144     user.
145     :return:
146     """
147     userid = request.values.get('userid')
148     password = request.values.get('password')
149     print(userid)
150     hashed = list(db.execute(f'''SELECT password
151 FROM leaderboard
152 WHERE leaderboard.id = {userid}
153 LIMIT 1;'''))[0][0]
154     valid = bcrypt.checkpw(password.encode(), hashed.encode())
155     if not valid:
156         return jsonify({'message': 'password fail'})
157     else:
158         try:
159             token = list(db.execute(f'''SELECT token
160 FROM tokens
161 WHERE id = {userid}
162 LIMIT 1;'''))[0][0]
163             return jsonify({'token': token})
164         except:
165             token = secrets.token_urlsafe(30)
166             db.execute(f"""INSERT INTO tokens (id, token)
167 VALUES ('{userid}', '{token}');""")
168             db.commit()
169             return jsonify({'token': token})
170
171
172 @app.route('/robo/addscore', methods=['POST'])
173 def add():
174     """
175     Used to add scores to the database, uses a post request. Must provide a
176     password to add a score.
177     This might be slightly annoying, but adding in functionality for tokens and
178     storing them in python
179     feels like a lot of work, maybe I will, but I probably wont invest my time
180     there, I could always cache the
181     password inputted in the python code instead.
182     :return:
183     """
184     userid = request.values.get('userid')
185     score = int(request.values.get('score'))
186     token = request.values.get('token')
187
188     tokenDB = list(db.execute(f'''SELECT token
189 FROM tokens
190 WHERE id = {userid}

```

```

188     LIMIT 1;'''))[0][0]
189     valid = token == tokenDB
190
191     if not valid:
192         return jsonify({'message': 'password fail'})
193     try:
194         db.execute(f'''INSERT INTO scores (id, scores)
195         VALUES ({userid},{score});''')
196         db.commit()
197
198         return jsonify({'message': 'success'})
199     except:
200         return jsonify({'message': 'fail'})
201
202
203 @app.route('/robo/adduser', methods=['POST'])
204 def adduser():
205     """
206     This is the API used to add a user to the database, users provide a
207     username, initials and their password.
208     Password validation will be done client side, need to keep this app as
209     lightweight as possible.
210     :return:
211     """
212     username = request.values.get('username')
213     initials = request.values.get('initials')
214     password = request.values.get('password')
215
216     tostore = bcrypt.hashpw(password.encode(), bcrypt.gensalt()).decode()
217
218     db.execute(f"""INSERT INTO leaderboard (initials, username, password)
219     VALUES ('{initials}','{username}','{tostore}');""")
220     db.commit()
221
222     return jsonify({'message': 'success'})
223
224 if __name__ == '__main__':
225     app.run()

```

index.html

```

1 <!DOCTYPE html>
2 <html>
3
4 <head>
5     <meta charset="utf-8">
6     <meta name="viewport" content="width=device-width, initial-scale=1.0,
7     shrink-to-fit=no">
8     <title>Robotron</title>
9     <meta name="theme-color" content="rgb(55,55,55)">
10    <meta name="description" content="Robotron leaderboard for robotron by John
11    Montgomery - a pygame game.">
12    <link rel="icon" type="image/png" sizes="360x360" href="../static/img/icon.
13    png">
14    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter
15    -bootstrap/4.5.2/css/bootstrap.min.css">
16    <link rel="manifest" href="manifest.json">
17    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/aos
18    /2.2.0/aos.css">
19    <link rel="stylesheet" href="/static/css/styles.css">
20 </head>
21
22 <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
41 px;margin-left: 0px;height: 100%;width: 100%;border: 3px dotted #4d1c86;">
42
43         <div style="margin-top: 0px;margin-right: 0px;margin-bottom
44 : 0px;margin-left: 0px;height: 100%;width: 100%;border: 3px dotted #4d1c86;"
45 >
46
47         <div style="margin-top: 0px;margin-right: 0px;margin-
48 bottom: 0px;margin-left: 0px;height: 100%;width: 100%;border: 3px dotted
49 #451e8a;">
50
51         <div style="margin-top: 0px;margin-right: 0px;
52 margin-bottom: 0px;margin-left: 0px;height: 100%;width: 100%;border: 3px
53 dotted #451e8a;">
54
55         <h2 style="color: rgb(69,31,138);font-family:
56 Conv_robotron-2084;text-align: center;margin-top: 9px;">robotron heroes</h2>
57
58         <div class="container" style="padding-right: 50
59 px;padding-left: 50px;margin-top: 50px;">
60
61         <div class="row" style="margin-right: -15px
62 ;">
63
64             <div class="col">
65
66                 <h3 style="font-family:
67 Conv_robotron-2084;color: rgb(255,51,38);">1 > {{ a[0].initials }} - {{ a
68 [0].scores }}</h3>
69
70             </div>
71
72             <div class="col">
73
74                 <h3 style="font-family:
75 Conv_robotron-2084;color: rgb(255,51,38);">6 > {{ a[5].initials }} - {{ a
76 [5].scores }}</h3>
77
78             </div>
79
80         </div>
81
82         <div class="row" style="margin-right: -15px
83 ;">
84
85             <div class="col">
86
87                 <h3 style="font-family:
88 Conv_robotron-2084;color: rgb(255,51,38);">2 > {{ a[1].initials }} - {{ a
89 [1].scores }}</h3>
90
91             </div>
92
93             <div class="col" style="font-family:
94 Conv_robotron-2084;color: rgb(255,51,38);">
95
96                 <h3 style="font-family:
97 Conv_robotron-2084;color: rgb(255,51,38);">7 > {{ a[6].initials }} - {{ a
98 [6].scores }}</h3>
99
100            </div>

```

```

62         </div>
        <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
65 [2].scores }}</h3>
        </div>
66     <div class="col">
67         <h3 style="font-family:
Conv_robotron-2084;color: rgb(255,51,38);">8 > {{ a[7].initials }} - {{ a
68 [7].scores }}</h3>
        </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
73 [3].scores }}</h3>
        </div>
74     <div class="col">
75         <h3 style="font-family:
Conv_robotron-2084;color: rgb(255,51,38);">9 > {{ a[8].initials }} - {{ a
76 [8].scores }}</h3>
        </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
81 [4].scores }}</h3>
        </div>
82     <div class="col">
83         <h3 style="font-family:
Conv_robotron-2084;color: rgb(255,51,38);">10 > {{ a[9].initials }} - {{ a
84 [9].scores }}</h3>
        </div>

```

```

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

```

```

121     </div>
122
123     <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.5.1/jquery.min
124     .js"></script>
125     <script src="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap
126     /4.5.2/js/bootstrap.bundle.min.js"></script>
127     <script src="https://cdnjs.cloudflare.com/ajax/libs/aos/2.2.0/aos.js"></
128     script>
129     <script src="../../static/js/script.min.js"></script>
130 </body>
131 </html>

```

error.html

```

1 <!DOCTYPE html>
2 <html>
3
4 <head>
5     <meta charset="utf-8">
6     <meta name="viewport" content="width=device-width, initial-scale=1.0,
7     shrink-to-fit=no">
8     <title>Robotron</title>
9     <meta name="theme-color" content="rgb(194,1,0)">
10    <meta name="description" content="Robotron leaderboard for robotron by John
11    Montgomery - a pygame game.">
12    <link rel="icon" type="image/png" sizes="360x360" href="../../static/img/icon.
13    png">
14    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-
15    bootstrap/4.5.2/css/bootstrap.min.css">
16    <link rel="manifest" href="manifest.json">
17    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/aos
18    /2.2.0/aos.css">
19    <link rel="stylesheet" href="../../static/css/styles.css">
20 </head>
21
22 <body style="background: rgb(0,0,0);">
23     <div data-aos="zoom-out" data-aos-duration="2000" style="margin-right: 1%;
24     margin-bottom: 0;margin-left: 1%;height: 98vh;width: 98%;margin-top: 1vh;
25     border: 3px dotted #9f095c;">
26         <div style="margin-top: 0px;margin-right: 0px;margin-bottom: 0px;margin-
27         left: 0px;height: 100%;width: 100%;border: 3px dotted #9f095c;">
28             <div style="margin-top: 0px;margin-right: 0px;margin-bottom: 0px;
29             margin-left: 0px;height: 100%;width: 100%;border: 3px dotted #970b60;">
30                 <div style="margin-top: 0px;margin-right: 0px;margin-bottom: 0
31                 px;margin-left: 0px;height: 100%;width: 100%;border: 3px dotted #970b60;">
32                     <div style="margin-top: 0px;margin-right: 0px;margin-bottom
33                     : 0px;margin-left: 0px;height: 100%;width: 100%;border: 3px dotted #900c64;"
34                     >
35                         <div style="margin-top: 0px;margin-right: 0px;margin-in-
36                         bottom: 0px;margin-left: 0px;height: 100%;width: 100%;border: 3px dotted
37                         #900c64;">
38                             <div style="margin-top: 0px;margin-right: 0px;
39                             margin-bottom: 0px;margin-left: 0px;height: 100%;width: 100%;border: 3px
40                             dotted #880e68;">
41                                 <div style="margin-top: 0px;margin-right: 0px;
42                                 margin-bottom: 0px;margin-left: 0px;height: 100%;width: 100%;border: 3px
43                                 dotted #880e68;">
44                                     <div style="margin-top: 0px;margin-right: 0
45                                     px;margin-bottom: 0px;margin-left: 0px;height: 100%;width: 100%;border: 3px
46                                     dotted #81106b;">
47                                         <div style="margin-top: 0px;margin-in-
48                                         right: 0px;margin-bottom: 0px;margin-left: 0px;height: 100%;width: 100%;
49                                         border: 3px dotted #81106b;">

```

```

28         <div style="margin-top: 0px;margin-
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(../
4     fonts/robotron-2084.woff) format("woff"), url(../fonts/robotron-2084.ttf)
5     format("truetype"), url(../fonts/robotron-2084.svg) format("svg");
6     font-weight: 400;
7     font-style: normal;
8 }
9 #inputcmd,
10 body {
11     background-color: #333;
12     color: #0f0;
13     font-family: "andale mono", "monotype.com", monaco, "courier new", courier,
14     monospace;
15 }
16 #terminal-window {

```

```
14     padding: 10px;
15     display: block;
16     position: absolute;
17     width: 100%;
18     height: 100%;
19     top: 0;
20     left: 0;
21     background-color: #111;
22     overflow: hidden;
23 }
24 #terminal-window:before {
25     content: "";
26     z-index: 4010;
27     width: 100%;
28     height: 100%;
29     position: absolute;
30     top: 0;
31     left: 0;
32     background: linear-gradient(#444 50%, #111 50%);
33     background-size: 100% 4px;
34     background-repeat: repeat-y;
35     opacity: 0.14;
36     box-shadow: inset 0 0 1px 1px rgba(0, 0, 0, 0.8);
37     animation: 5s linear infinite pulse;
38 }
39 #cursor {
40     color: #0f0;
41     box-sizing: border-box;
42     border-left: 0.5em solid;
43 }
44 .blink {
45     animation: 6s steps(13, end) infinite typing, 1s step-end infinite blinking
46     ;
47 }
48 .scanlines {
49     z-index: 4100;
50 }
51 .hide {
52     display: none;
53 }
54 #inputcmd {
55     background-color: #111;
56     border: 1px;
57     font-size: 1em;
58     color: transparent;
59     text-shadow: 0 0 0 #0f0;
60 }
61 #inputcmd:focus {
62     outline: 0;
63 }
64 @keyframes pulse {
65     0% {
66         transform: scale(1.001);
67         opacity: 0.14;
68     }
69     8% {
70         transform: scale(1);
71         opacity: 0.13;
72     }
73     15% {
74         transform: scale(1.004);
75         opacity: 0.14;
```

```

76     }
77     30% {
78         transform: scale(1.002);
79         opacity: 0.11;
80     }
81     100% {
82         transform: scale(1);
83         opacity: 0.14;
84     }
85 }
86 @keyframes vline {
87     0% {
88         top: 0;
89     }
90     100% {
91         top: 100%;
92     }
93 }
94 @keyframes blinking {
95     from,
96     to {
97         border-color: transparent;
98     }
99     50% {
100         border-color: green;
101     }
102 }

```

7.3.2 Game Code

main.py

```

1 import sys
2 import controller, eventmanager
3
4 import model
5 import views
6
7
8 def run(mode):
9     evManager = eventmanager.EventManager()
10    gamemodel = model.Game(evManager)
11    graphics = views.GraphicalView(evManager, gamemodel)
12    keyboard = controller.Controller(evManager, gamemodel)
13
14    gamemodel.run(mode)
15
16
17
18 if __name__ == "__main__":
19     try:
20         if sys.argv[1].lower() == "test":
21             run("test")
22     except IndexError:
23         run(None)

```

eventmanager.py

```

1 from event import *
2
3
4 class EventManager:
5     """

```

```

6     Controls the flow of events between the M, V and C
7     """
8
9     def __init__(self):
10         """
11         Weak ref stops us needing to remove objects from the dict as they will
12         end up deleted when the objects instance is used. This will stop the dict
13         becoming bloated and stop me from needing to remember to remove items from
14         it.
15         """
16         self.listeners = []
17
18     def add_listener(self, listener):
19         """
20         This adds an object as a listener--in-place --aggressive --aggressive
21         """
22         self.listeners.append(listener)
23
24     def remove_listener(self, listener):
25         """
26         This is to stop objects listening, but due to the weak referencing it
27         doesnt end up used much
28         """
29
30         if listener in self.listeners:
31             del self.listeners[listener]
32
33     def post(self, event):
34         """
35         This will emit a message to all the objects in the listen dict
36         if it isn't a tick then we also print that event - mostly to debug
37         """
38         if not isinstance(event, Tick):
39             print(str(event))
40         for listener in self.listeners:
41             listener.notify(event)

```

statemachine.py

```

1 class StateMachine:
2     def __init__(self):
3         self.stack = []
4
5     def peek(self):
6         try:
7             return self.stack[-1]
8         except IndexError:
9             return None
10
11     def pop(self):
12         try:
13             self.stack = self.stack[1:]
14             return len(self.stack) > 0
15         except IndexError:
16             return None
17
18     def push(self, state):
19         self.stack.append(state)
20         return state

```

model.py

```

1 from event import *
2 from statemachine import StateMachine
3

```

```

4 from states import *
5
6
7 class Game:
8     def __init__(self, eventManager):
9         self.statem = StateMachine()
10        self.eventManager = eventManager
11        eventManager.add_listener(self)
12        self.on = False
13
14    def notify(self, event):
15        if isinstance(event, EndGame):
16            self.stop()
17        elif isinstance(event, ChangeState):
18            # pop request
19            if not event.state:
20                # false if no more states are left
21                if not self.statem.pop():
22                    self.eventManager.Post(EndGame())
23            else:
24                # push a new state on the stack
25                self.statem.push(event.state)
26
27    def stop(self):
28        self.on = False
29
30
31    def run(self, mode='norm'):
32        self.on = True
33        self.eventManager.post(Start())
34        if mode == 'test':
35            self.statem.push(STATE_TEST)
36        elif mode == 'light':
37            pass
38        else:
39            self.statem.push(STATE_INTRO1)
40        while self.on:
41            newTick = Tick()
42            self.eventManager.post(newTick)

```

views.py

```

1 import pygame
2
3 import menu
4 import testing
5 from characters_module.player import Player
6 from constants.const import *
7 from decorations.border import Border
8 from event import *
9 from states import *
10 import gameplay
11 from characters_module.humans import *
12 from characters_module.enemy import *
13
14 class GraphicalView(object):
15     """
16     Draws the model state onto the screen.
17     """
18
19     def __init__(self, evManager, model):
20         """
21         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
82             isinstance(event_in, KeyboardUp) or isinstance(event_in, Mouse):
83             currentstate = self.model.statem.peek()
84             if currentstate == STATE_TEST:
85                 testing.testing(event_in, self)

```

```

85         if currentstate == STATE_INTRO1:
86             menu.allopperational(self)
87         if currentstate == HOMESCREEN:
88             menu.home(self, event_in)
89         if currentstate == LOGIN:
90             menu.login(self, event_in)
91         if currentstate == PLAYGAME:
92             self.evManager.post(ChangeState(LOAD_LEVEL1))
93         if currentstate == ENDGAME:
94             menu.endgame(self, event_in)
95         if currentstate > 200:
96             gameplay.loadlevel(self, currentstate - 200)
97         if currentstate < 200:
98             gameplay.level(self, event_in)
99
100
101     def clearScreen(self):
102         self.screen.fill((0, 0, 0))
103
104     def initialize(self):
105         """
106         Set up the pygame graphical display and loads graphical resources.
107         """
108
109         result = pygame.init()
110         pygame.font.init()
111         pygame.display.set_caption(TITLE)
112         self.screen = pygame.display.set_mode(SCREENSIZE)
113         self.clock = pygame.time.Clock()
114         self.tinyfont = pygame.font.Font('font/robotron-2084.ttf', 10)
115         self.minifont = pygame.font.Font('font/robotron-2084.ttf', 18)
116         self.smallfont = pygame.font.Font('font/robotron-2084.ttf', 28)
117         self.font = pygame.font.Font('font/robotron-2084.ttf', 34)
118         self.largefont = pygame.font.Font('font/robotron-2084.ttf', 80)
119         self.isinitialized = True
120         self.player = Player()
121         self.lives = 3

```

controller.py

```

1
2 import pygame
3
4 from event import *
5
6
7 class Controller:
8     def __init__(self, eventManager, model):
9         self.eventManager = eventManager
10        eventManager.add_listener(self)
11        self.model = model
12
13    def notify(self, event):
14        if isinstance(event, Tick):
15
16            for event in pygame.event.get():
17
18                if event.type == pygame.QUIT:
19                    self.eventManager.post(EndGame())
20                if event.type == pygame.KEYDOWN:
21                    if event.key != pygame.K_ESCAPE:
22                        if event.key != pygame.K_BACKSPACE:
23                            self.eventManager.post(Keyboard(event.key, event.
unicode))

```

```

24         else:
25             self.eventManager.post(Keyboard(event.key, '
    backspace'))
26         else:
27             self.eventManager.post(EndGame())
28         if event.type == pygame.KEYUP:
29             self.eventManager.post(KeyboardUp(event.key))
30
31         if event.type == pygame.MOUSEBUTTONDOWN:
32             self.eventManager.post(Mouse(event.pos))

```

event.py

```

1
2 class Event:
3     """
4     A class which is a super for all other events the system might handle
5     """
6     def __init__(self):
7         self.name = 'Some event'
8
9     def __str__(self):
10         return self.name
11
12 class EndGame(Event):
13     """
14     This event is sent at the end of the game
15     """
16     def __init__(self):
17         self.name = 'End Game'
18
19
20 class Start(Event):
21     """
22     This event is sent at the start of the game
23     """
24     def __init__(self):
25         self.name = 'Start Game'
26
27
28 class Tick(Event):
29     """
30     A tick
31     """
32     def __init__(self):
33         self.name = 'Tick'
34
35 class Keyboard(Event):
36     """
37     Event for keyboard clicks
38     """
39     def __init__(self, keys, letter):
40         self.name = 'Keyboard'
41         self.key = keys
42         self.uni = letter
43     def __str__(self):
44         return f"Keypress - {self.uni}"
45
46 class KeyboardUp(Event):
47     """
48     Event for keyboard clicks
49     """
50     def __init__(self, keys):
51         self.name = 'Keyboard'

```



```

52         self.key = keys
53     def __str__(self):
54         return f"Key release - {self.key}"
55
56 class Mouse(Event):
57     """
58     Event for mouse clicks
59     """
60     def __init__(self, pos):
61         self.name = 'Mouse'
62         self.pos = pos
63     def __str__(self):
64         return f"Mouse - {self.pos}"
65
66 class ChangeState(Event):
67     def __init__(self, newState):
68         self.name = 'Change State'
69         self.state = newState
70     def __str__(self):
71         return str(self.state)

```

states.py

```

1 STATE_ = 1
2 STATE_TEST = 2
3 STATE_INTRO1 = 3
4 STATE_INTRO2 = 4
5 STATE_PLAY = 5
6 HOMESCREEN = 6
7 PLAYGAME = 7
8 HELP = 8
9 LOGIN = 9
10 START_SCREEN = 10
11 ENDGAME = 11
12
13 LEVEL1 = 101
14 LEVEL2 = 102
15 LEVEL3 = 103
16 LEVEL4 = 104
17 LEVEL5 = 105
18 LEVEL6 = 106
19 LEVEL7 = 107
20 LEVEL8 = 108
21 LEVEL9 = 109
22 LEVEL10 = 110
23 LEVEL11 = 111
24 LEVEL12 = 112
25 LEVEL13 = 113
26 LEVEL14 = 114
27 LEVEL15 = 115
28 LEVEL16 = 116
29 LEVEL17 = 117
30 LEVEL18 = 118
31 LEVEL19 = 119
32 LEVEL20 = 120
33 LEVEL21 = 121
34 LEVEL22 = 122
35 LEVEL23 = 123
36 LEVEL24 = 124
37 LEVEL25 = 125
38 LEVEL26 = 126
39 LEVEL27 = 127
40 LEVEL28 = 128
41 LEVEL29 = 129

```

```
42 LEVEL30 = 130
43 LEVEL31 = 131
44 LEVEL32 = 132
45 LEVEL33 = 133
46 LEVEL34 = 134
47 LEVEL35 = 135
48 LEVEL36 = 136
49 LEVEL37 = 137
50 LEVEL38 = 138
51 LEVEL39 = 139
52 LEVEL40 = 140
53
54 LOAD_LEVEL1 = 201
55 LOAD_LEVEL2 = 202
56 LOAD_LEVEL3 = 203
57 LOAD_LEVEL4 = 204
58 LOAD_LEVEL5 = 205
59 LOAD_LEVEL6 = 206
60 LOAD_LEVEL7 = 207
61 LOAD_LEVEL8 = 208
62 LOAD_LEVEL9 = 209
63 LOAD_LEVEL10 = 210
64 LOAD_LEVEL11 = 211
65 LOAD_LEVEL12 = 212
66 LOAD_LEVEL13 = 213
67 LOAD_LEVEL14 = 214
68 LOAD_LEVEL15 = 215
69 LOAD_LEVEL16 = 216
70 LOAD_LEVEL17 = 217
71 LOAD_LEVEL18 = 218
72 LOAD_LEVEL19 = 219
73 LOAD_LEVEL20 = 220
74 LOAD_LEVEL21 = 221
75 LOAD_LEVEL22 = 222
76 LOAD_LEVEL23 = 223
77 LOAD_LEVEL24 = 224
78 LOAD_LEVEL25 = 225
79 LOAD_LEVEL26 = 226
80 LOAD_LEVEL27 = 227
81 LOAD_LEVEL28 = 228
82 LOAD_LEVEL29 = 229
83 LOAD_LEVEL30 = 230
84 LOAD_LEVEL31 = 231
85 LOAD_LEVEL32 = 232
86 LOAD_LEVEL33 = 233
87 LOAD_LEVEL34 = 234
88 LOAD_LEVEL35 = 235
89 LOAD_LEVEL36 = 236
90 LOAD_LEVEL37 = 237
91 LOAD_LEVEL38 = 238
92 LOAD_LEVEL39 = 239
93 LOAD_LEVEL40 = 240
```

menu.py

```
1 import random
2 import webbrowser
3 from playsound import playsound
4 import pygame
5
6 from APIinteractions import *
7 from constants.colors import *
8 from constants.const import *
9 from event import *
```

```

10 from states import *
11
12 _circle_cache = {}
13 def _circlepoints(r):
14     r = int(round(r))
15     if r in _circle_cache:
16         return _circle_cache[r]
17     x, y, e = r, 0, 1 - r
18     _circle_cache[r] = points = []
19     while x >= y:
20         points.append((x, y))
21         y += 1
22         if e < 0:
23             e += 2 * y - 1
24         else:
25             x -= 1
26             e += 2 * (y - x) - 1
27     points += [(y, x) for x, y in points if x > y]
28     points += [(-x, y) for x, y in points if x]
29     points += [(x, -y) for x, y in points if y]
30     points.sort()
31     return points
32
33 def render(text, font, gfcolor=pygame.Color('dodgerblue'), ocolor=(255, 130,
45), opx=2):
34     textsurface = font.render(text, True, gfcolor).convert_alpha()
35     w = textsurface.get_width() + 2 * opx
36     h = font.get_height()
37
38     osurf = pygame.Surface((w, h + 2 * opx)).convert_alpha()
39     osurf.fill((0, 0, 0))
40
41     surf = osurf.copy()
42
43     osurf.blit(font.render(text, True, ocolor).convert_alpha(), (0, 0))
44
45     for dx, dy in _circlepoints(opx):
46         surf.blit(osurf, (dx + opx, dy + opx))
47
48     surf.blit(textsurface, (opx, opx))
49     return surf
50
51 def get_ran_col():
52     return random.choice(random_colors)
53
54 def randomStart(view):
55     for i in range(0, SCREENSIZE[0], 2):
56         for j in range(0, SCREENSIZE[1], 2):
57             col = get_ran_col()
58             rect = pygame.Rect((i, j), (2, 2))
59             pygame.draw.rect(view.screen, col, rect)
60
61 def allopperational(view):
62     view.tickcounter += 1
63     if view.tickcounter == 2:
64         playsound('audio/intro.mp3', block = False)
65     if view.tickcounter > 40:
66         view.evManager.post(ChangeState(HOMESCREEN))
67     elif view.tickcounter > 5:
68
69         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.edgocol = random.choice(edge)
116             for idx, letter in enumerate('ROBOTRON:'):
117                 image = render(letter, view.largefont, gfcolor=view.col, ocolor=
view.edgocol)
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
176 AS ' + text)
177                 position_font = (SCREENSIZE[0] - width) / 2
178                 view.screen.blit(somewords, (position_font + 6, 470))
179             except FileNotFoundError:
180                 somewords = view.smallfont.render(
181                     'ENTER for LOGIN',
182                     True,
183                     (22, 32, 221))
184                 width, _ = pygame.font.Font.size(view.smallfont, 'ENTER for
185 LOGIN')
186                 position_font = (SCREENSIZE[0] - width) / 2
187                 view.screen.blit(somewords, (position_font + 6, 470))
188
189         somewords = view.minifont.render(
190             '''Leaderboard Avaliable at - robo.johnmontgomery.tech''',
191             True,
192             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
262 =3)
263
264 pygame.draw.rect(view.screen, GREY, pygame.Rect(100, 150, 600, 40), width
265 =3)
266
267 pygame.draw.rect(view.screen, GREY, pygame.Rect(340, 200, 120, 40), width
268 =3)
269
270 pygame.draw.lines(view.screen, GREY, False, [(30,10),(10,25), (30, 40)],
271 width=5)
272
273 signup = view.smallfont.render(
274     'SIGN UP',
275     True,
276     (255, 255, 255))
277 width, _ = pygame.font.Font.size(view.smallfont, 'SIGN UP')
278 position_font = (SCREENSIZE[0] - width) / 2
279 view.screen.blit(signup, (position_font, 506))
280
281 pygame.draw.rect(view.screen, GREY, pygame.Rect(100, 300, 600, 40), width
282 =3)
283
284 pygame.draw.rect(view.screen, GREY, pygame.Rect(100, 350, 600, 40), width
285 =3)
286
287 pygame.draw.rect(view.screen, GREY, pygame.Rect(100, 400, 600, 40), width
288 =3)
289
290 pygame.draw.rect(view.screen, GREY, pygame.Rect(100, 450, 600, 40), width
291 =3)
292
293 pygame.draw.rect(view.screen, GREY, pygame.Rect(320, 500, 160, 40), width
294 =3)
295
296 if isinstance(event, Mouse):
297     if 100<event.pos[0]<700 and 100<event.pos[1]<140:
298         pygame.draw.rect(view.screen, WHITE, pygame.Rect(100,97,600,43),
299 width=5)
300         view.highlight = 'username'
301     elif 100<event.pos[0]<700 and 150<event.pos[1]<190:
302         pygame.draw.rect(view.screen, WHITE, pygame.Rect(100,147,600,43),
303 width=5)
304         view.highlight = 'password'
305     elif 100<event.pos[0]<700 and 300<event.pos[1]<340:
306         pygame.draw.rect(view.screen, WHITE, pygame.Rect(100, 297, 600, 43)
307 , width=5)
308         view.highlight = 'username1'
309     elif 100<event.pos[0]<700 and 350<event.pos[1]<390:
310         pygame.draw.rect(view.screen, WHITE, pygame.Rect(100, 347, 600, 43)
311 , 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
353             else:
354                 view.password2 = view.password2[:-1]

```



```

355         if len(view.password2)>40:
356             view.password2 = view.password2[:-1]
357
358         elif view.highlight == 'initials':
359             if event.uni != 'backspace':
360                 view.initials += event.uni
361             else:
362                 view.initials = view.initials[:-1]
363             if len(view.initials)>3:
364                 view.initials = view.initials[:-1]
365
366         else:
367             if event.uni != 'backspace':
368                 view.password += event.uni
369             else:
370                 view.password = view.password[:-1]
371             if len(view.password)>40:
372                 view.password = view.password[:-1]
373
374         if view.password2 != view.password1:
375             pygame.draw.rect(view.screen, RED, pygame.Rect(100, 347, 600, 43),
width=5)
376             pygame.draw.rect(view.screen, RED, pygame.Rect(100, 397, 600, 43),
width=5)
377
378         text = view.tinyfont.render(
379             'username',
380             True,
381             (255, 255, 255))
382         width, _ = pygame.font.Font.size(view.tinyfont, 'username')
383         position_font = (SCREENSIZE[0] - width) / 2
384         view.screen.blit(text, (position_font + 6, 301))
385
386         text = view.minifont.render(
387             view.username1,
388             True,
389             (255, 255, 255))
390         width, _ = pygame.font.Font.size(view.minifont, view.username1)
391         position_font = (SCREENSIZE[0] - width) / 2
392         view.screen.blit(text, (position_font + 6, 311))
393
394         text = view.tinyfont.render(
395             'password',
396             True,
397             (255, 255, 255))
398         width, _ = pygame.font.Font.size(view.tinyfont, 'password')
399         position_font = (SCREENSIZE[0] - width) / 2
400         view.screen.blit(text, (position_font + 6, 351))
401
402         text = view.minifont.render(
403             '*'*len(view.password1),
404             True,
405             (255, 255, 255))
406         width, _ = pygame.font.Font.size(view.minifont, '*'*len(view.password1))
407         position_font = (SCREENSIZE[0] - width) / 2
408         view.screen.blit(text, (position_font + 6, 361))
409
410         text = view.tinyfont.render(
411             'confirm password',
412             True,
413             (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.edgocol = random.choice(edge)
495             for idx, letter in enumerate('ROBOTRON:'):
496                 image = render(letter, view.largefont, gfcolor=view.col, ocolor=
view.edgocol)
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(
534                 str(view.score),
535                 True,

```

```

536         (246, 130, 20))
537     width, _ = pygame.font.Font.size(view.font, str(view.score))
538     position_font = (SCREENSIZE[0] - width) / 2
539     view.screen.blit(somewords, (position_font + 6, 450))
540
541     somewords = view.smallfont.render(
542         'SPACE for homescreen',
543         True,
544         (246, 130, 20))
545     width, _ = pygame.font.Font.size(view.smallfont, 'SPACE for
homescreen')
546     position_font = (SCREENSIZE[0] - width) / 2
547     view.screen.blit(somewords, (position_font + 6, 500))
548
549     somewords = view.smallfont.render(
550         '0 to open leaderboard',
551         True,
552         (246, 130, 20))
553     width, _ = pygame.font.Font.size(view.smallfont, '0 to open
leaderboard')
554     position_font = (SCREENSIZE[0] - width) / 2
555     view.screen.blit(somewords, (position_font + 6, 525))
556
557     if checkonline():
558         if isloggedin():
559             if addscore(view.score):
560                 somewords = view.smallfont.render(
561                     'Score added to leaderboard',
562                     True,
563                     (246, 130, 20))
564                 width, _ = pygame.font.Font.size(view.smallfont, 'Score
added to leaderboard')
565                 position_font = (SCREENSIZE[0] - width) / 2
566                 view.screen.blit(somewords, (position_font + 6, 550))
567             else:
568                 somewords = view.smallfont.render(
569                     'Enter to log in',
570                     True,
571                     (246, 130, 20))
572                 width, _ = pygame.font.Font.size(view.smallfont, 'Enter to
log in')
573                 position_font = (SCREENSIZE[0] - width) / 2
574                 view.screen.blit(somewords, (position_font + 6, 550))
575             else:
576                 somewords = view.smallfont.render(
577                     'OFFLINE',
578                     True,
579                     RED)
580                 width, _ = pygame.font.Font.size(view.smallfont, 'OFFLINE')
581                 position_font = (SCREENSIZE[0] - width) / 2
582                 view.screen.blit(somewords, (position_font + 6, 550))
583     pygame.display.flip()
584
585     view.clock.tick(TPS)

```

gameplay.py

```

1 import pygame
2 from constants.colors import *
3 from event import *
4 from constants.const import *
5 from objects.bullet import Bullet
6 from states import *
7 import csv

```

```

8 from characters_module.enemy import *
9 from characters_module.humans import *
10 from characters_module.player import *
11 from playsound import playsound
12 from math import sqrt
13 def loadlevel(view, level):
14     playsound('audio/change.mp3', block=False)
15     with open('levels/levels.csv') as f:
16         print(level)
17         csvreader = csv.reader(f, delimiter=',')
18         line = 0
19         for row in csvreader:
20             if line == 0:
21                 headers = row
22             if line == level:
23                 leveledata = row
24             line += 1
25
26     for header, count in zip(headers[1:], leveledata[1:]):
27         view.leveledata[header] = count
28
29     for char in view.leveledata.keys():
30         for _ in range(int(view.leveledata[char])):
31             newobject = eval(f"{char}()")
32             view.spriteslist.add(newobject)
33
34
35     r, g, b = 0, 102, 102
36     view.screen.fill(BLACK)
37     for i in range(60):
38         if r > 0 and b == 0:
39             r -= 17
40             g += 17
41         if g > 0 and r == 0:
42             g -= 17
43             b += 17
44         if b > 0 and g == 0:
45             b -= 17
46             r += 17
47         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)
48         view.clock.tick(TPS)
49         pygame.display.flip()
50
51     for i in range(60):
52         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)
53         view.clock.tick(TPS+4)
54         pygame.display.flip()
55
56     view.evManager.post(ChangeState(100+level))
57     return
58
59
60 def level(view, event):
61     player = view.player
62
63     if not view.isinitialized:
64         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]-
171 object.rect[1]<20 and not isinstance(object, Bullet):
172                     if isinstance(object, Grunt) or isinstance(object,
173 Electrode) or isinstance(object, Hulk):
174                         object.kill()
175                         item.kill()
176
177                     if isinstance(item, Electrode) or isinstance(item, Grunt) or
178 isinstance(item, Hulk):
179                         if -20<item.rect[0]-player.position[0]<20 and -20<item.rect[1]-
180 player.position[1]<20:
181                             view.lives -= 1
182                             if view.lives > 0:
183                                 view.evManager.post(ChangeState(view.model.statem.peek
184 () + 101))
185
186                             return
187                         else:
188                             view.evManager.post(ChangeState(ENDGAME))
189
190                     if isinstance(item, Electrode):
191                         for object in view.spriteslist:
192                             if -10 < item.rect[0] - object.rect[0] < 10 and -10 < item.
193 rect[1] - object.rect[1] < 10 and not isinstance(object, Electrode):
194                                 if isinstance(object, Grunt):

```

```

186         object.kill()
187         if isinstance(item, Mommies) or isinstance(item, Daddies) or
188         isinstance(item, Mikeys):
189             if -20 < item.rect[0] - player.position[0] < 20 and -20 < item.
190             rect[1] - player.position[1] < 20:
191                 score = item.die(view)
192                 view.score += score
193
194     gruntcount = sum(1 if isinstance(i, Grunt) else 0 for i in view.spriteslist
195 )
196     if gruntcount == 0:
197         view.evManager.post(ChangeState(view.model.statem.peek() + 101))
198         return
199
200     view.spriteslist.draw(view.screen)
201
202     if view.tickcounter > 30:
203         player.getskin(view.skincount)
204         view.screen.blit(player.getskin(view.skincount), player.position)
205
206     view.clock.tick(TPS)
207     # flip the display to show whatever we drew
208
209     pygame.display.flip()

```

APIinteractions.py

```

1 import requests
2
3 apiurl = 'http://127.0.0.1:5000'
4 from requests.adapters import HTTPAdapter
5 from requests.packages.urllib3.util.retry import Retry
6
7
8 session = requests.Session()
9 retry = Retry(connect=3, backoff_factor=0.4)
10 adapter = HTTPAdapter(max_retries=retry)
11 session.mount('http://', adapter)
12
13 def loginuser(username, password):
14     userid = session.get(apiurl+'/robo/userid/'+username).json().get('id')
15     if userid:
16         token = session.post(apiurl+'/login', params={
17             'userid': userid,
18             'password': password}).json().get('token')
19         if not token:
20             return False
21         with open('.token', 'w') as f:
22             f.write(token + '>' + str(userid) + '>' + username)
23
24         return True
25     return False
26
27
28
29 def signupuser(u,p1,p2,i):
30     try:
31         userid = session.get(apiurl + '/robo/userid/' + u).json().get('id')
32         if not userid:
33             if p1 == p2:
34                 session.post(apiurl + '/robo/adduser', params={

```



```

35         'username': u,
36         'password': p1,
37         'initials': i})
38     userid = session.get(apiurl + '/robo/userid/' + u).json().get('
id')
39     token = session.post(apiurl + '/login', params={
40         'userid': userid,
41         'password': p1}).json().get('token')
42
43     with open('.token', 'w') as f:
44         f.write(token + '>' + str(userid) + '>' + u)
45
46     return True
47 else:
48     return False
49 except:
50     False
51
52 def checkonline():
53     try:
54         requests.get(apiurl)
55         return True
56     except requests.exceptions.ConnectionError:
57         return False
58
59 def addscore(score):
60     with open('.token', 'r') as f:
61         token, id, _ = f.read().split('>')
62     result = session.post(apiurl + '/robo/addscore', params={
63         'userid': id,
64         'token': token,
65         'score': score}).json().get('message')
66     return True
67
68 def isloggedin():
69     try:
70         open('.token')
71         return True
72     except:
73         return False

```

characters_module

characters.py

```

1 from pygame import sprite, image, transform
2 from characters_module import sprites
3 from constants.const import *
4 from characters_module.sprites import stretech_image
5
6 class Character(sprite.Sprite):
7     """
8     This is a very basic character, from which all the other characters will
9     extend, this is never used directly,
10    and there will need to be lots of extra functions. This code mostly is
11    needed for the animation and directions
12    """
13    def __init__(self, sheetname, imagecount=12, scale=30):
14        """
15        This creates the character, mostly handles grabbing the spritesheet,
16        clipping the sprites and scaling them.
17        """
18        super().__init__()

```

```

16         self.sheetname = sheetname
17         self.spritesheet = image.load(self.sheetname).convert()
18         h,w = self.spritesheet.get_height(), self.spritesheet.get_width()/
imagecount
19         self.images = [transform.scale(sprite_item, (scale,scale)) for
sprite_item in
20             sprites.loadStrip((0, 0, w, h), imagecount, self.
spritesheet)]
21
22         self.direction = 'N'
23         self.position = (300,200)
24         self.moving = (0,0)
25         self.image = self.images[0]
26         self.rect = (300,200)
27
28     def setdir(self, mov, dir):
29         """
30         This sets the current direction (for the spirte animation) based on the
where the character is moving and facing
31         """
32         if dir:
33             if mov > 0:
34                 self.direction = 'E'
35             if mov < 0:
36                 self.direction = 'W'
37         else:
38             if mov > 0:
39                 self.direction = 'S'
40             if mov < 0:
41                 self.direction = 'N'
42
43
44     def onstart(self, view):
45         """
46         When the character is created, this places it onto the screen, adding
some stretch
47         """
48         view.screen.fill((0, 0, 0))
49         img, h = stretch_image(self.images[0], 30-view.tickcounter)
50         posX, posY = self.position
51         view.screen.blit(img, (posx, posY - h / 2))

```

enemy.py

```

1
2 from characters_module.characters import Character
3 from constants.const import *
4 import random
5 import pygame
6 class Enemy(Character):
7     """
8     This enemy is again, only used to extend from. It acts as a basic super
class which can easily be used to generate
9     the other classes for the enemies. Because the enemies need to update in
different ways, its not possible to have
10    them all exhibit the same behaviour here.
11    """
12    def __init__(self, sheetname, images=12):
13        Character.__init__(self, sheetname, images)
14        self.rect = (random.randint(50, SCREENSIZE[0]-50), random.randint(70,
SCREENSIZE[1]-50) )
15
16
17

```

```

18 class Grunt(Enemy):
19     """
20     This is the basic enemy, which is only able to move, and on colliding with
21     the player, it kills the player. If it
22     gets hit by a bullet, it dies
23     """
24     def __init__(self):
25         self.sheetname = 'sprites/grunt.png'
26         Enemy.__init__(self, self.sheetname)
27         self.vx = random.randint(-20,20)
28
29         self.vy = random.randint(-20, 20)
30     def update(self, count, movx,movy):
31         """
32         This is a pretty poorly executed AI. I think ill replace this with a
33         boids algorithm.
34         """
35
36         self.image = self.images[count]
37         position = self.rect
38         x = movx
39         y = movy
40         legnth = sqrt(x**2 + y**2)
41         adj = legnth / 3
42         newy = y / adj
43         newx = x / adj
44
45         newx = max(50,min(self.rect[0]+newx, SCREENSIZE[0]-50))
46         newy = max(50, min(self.rect[1]+newy, SCREENSIZE[1]-50))
47
48         self.rect = (newx, newy)
49
50 class Electrode(Enemy):
51     """
52     These are the static enemies
53     """
54     def __init__(self):
55         self.sheetname = 'sprites/electrode.png'
56         Enemy.__init__(self, self.sheetname,3)
57         self.image = random.choice(self.images)
58         self.image = pygame.transform.scale(self.image, (20,20))
59     def update(self, count, _):
60         return
61
62 class Hulk(Enemy):
63     """
64     These are like the grunts, but cant be killed. They only slow down when hit
65     .
66     """
67     def __init__(self):
68         self.sheetname = 'sprites/hulk.png'
69         Enemy.__init__(self, self.sheetname)
70         self.living = 0
71
72     def getskin(self, count):
73         """
74         This is overriding the base function. Hulks always face the same way
75         """
76         if self.velocity[0] < 0:
77             return self.images[:3][count]
78         elif self.velocity[0] > 0:
79             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 class Spheroids(Enemy):
124     def __init__(self):
125         self.sheetname = 'sprites/spheroids.png'
126         Enemy.__init__(self, self.sheetname, 8)
127
128 class Quarks(Enemy):
129     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
8     have the same movement and actions) so
9     they can all extend from a very very basic class.
10    """
11    def __init__(self, sheetname, images=12):
12        Character.__init__(self, sheetname, images, 60)
13        self.rect = self.image.get_rect()
14        self.rect.center = (random.randint(50, SCREENSIZE[0] - 50), random.
15        randint(70, SCREENSIZE[1] - 50))
16
17    def update(self, count, _):
18        if not self.living % 25 :
19            self.velocity = (random.choice((-4, 4, 0)), random.choice((-4, 4, 0))
20        )
21        flag = False
22        while not flag:
23            if (35+BORDER_W < self.rect[1] + self.velocity[1] < SCREENSIZE[1]-
24            BORDER_W*2-35):
25                self.rect = (self.rect[0], self.rect[1]+self.velocity[1])
26                flag = True
27            else:
28                self.velocity = (random.choice((-4, 4, 0)), random.choice((-4,
29            4, 0)))
30            if (BORDER_W-20 < self.rect[0] + self.velocity[0] < SCREENSIZE[0]-
31            BORDER_W*2 -20):
32                self.rect = (self.rect[0]+self.velocity[0], self.rect[1])
33            else:
34                self.velocity = (random.choice((-4, 4, 0)), random.choice((-4,
35            4, 0)))
36
37        self.living += 1
38
39        self.image = self.getskin(count)
40
41    def die(self, view):
42        value = self.value
43        somewords = view.minifont.render(
44            self.value,
45            True,
46            (246, 130, 20))

```

```

41         view.screen.blit(somewords, self.rect)
42         time.sleep(0.05)
43         self.kill()
44         return int(value)
45
46     def getskin(self, count):
47         if self.velocity[0] < 0:
48             return self.images[:3][count]
49         elif self.velocity[0] > 0:
50             return self.images[3:6][count]
51         elif self.velocity[1] < 0:
52             return self.images[6:9][count]
53         elif self.velocity[1] > 0:
54             return self.images[9:12][count]
55         else:
56             return self.images[0]
57
58
59
60 class Mommies(Human):
61     def __init__(self):
62         self.sheetname = 'sprites/mommies.png'
63         self.living = 0
64         self.velocity = (random.randint(-4, 4), random.randint(-4, 4))
65         self.value = '1000'
66         Character.__init__(self, self.sheetname)
67
68
69
70
71 class Daddies(Human):
72     def __init__(self):
73         self.sheetname = 'sprites/daddies.png'
74         self.living = 0
75         self.velocity = (random.randint(-4, 4), random.randint(-4, 4))
76         self.value = '1000'
77         Character.__init__(self, self.sheetname)
78
79
80
81 class Mikeys(Human):
82     """
83     These are the 'kids' - i made them move slower
84     """
85     def __init__(self):
86         self.sheetname = 'sprites/mikeys.png'
87         self.living = 0
88         self.velocity = (random.randint(-3, 3), random.randint(-3, 3))
89         self.value = '1000'
90         Character.__init__(self, self.sheetname)

```

player.py

```

1 from characters_module.characters import Character
2 from constants.const import *
3
4
5 class Player(Character):
6     """
7     Because most of the logic about whether a player is alive and the score is
8     handled by the Model, most of it
9     can be abstracted away. This class mostly handles the player screen logic,
10    and doesnt look ay the logic of whether
11    or not the player is alive.

```

```

10 """
11 def __init__(self):
12     self.sheetname = 'sprites/player.png'
13     Character.__init__(self, self.sheetname)
14     self.l_images = self.images[:3]
15     self.r_images = self.images[3:6]
16     self.f_images = self.images[6:9]
17     self.u_images = self.images[9:12]
18
19 def getskin(self, count):
20     if self.direction[0] == 'N':
21         return self.u_images[count]
22     elif self.direction[0] == 'S':
23         return self.f_images[count]
24     elif self.direction[0] == 'W':
25         return self.l_images[count]
26     elif self.direction[0] == 'E':
27         return self.r_images[count]
28
29 def movy(self, newMov):
30     if (25+BORDER_W < self.position[1] + newMov < SCREENSIZE[1]-BORDER_W
31 *2-30) :
32         self.position = (self.position[0], self.position[1] + newMov)
33         self.setdir(newMov, 0)
34
35 def movx(self, newMov):
36     if (BORDER_W-10 < self.position[0] + newMov < SCREENSIZE[0]-BORDER_W*2
37 -10):
38         self.position = (self.position[0]+newMov, self.position[1])
39         self.setdir(newMov, 1)

```

sprites.py

```

1 import pygame
2 from constants import colors as COLS
3
4
5 def getImage(sheet, rectangle):
6     """ Grab a single image out of a larger spritesheet
7         Pass in the x, y location of the sprite
8         and the width and height of the sprite. """
9     rect = pygame.Rect(rectangle)
10
11     # Create a new blank image
12     image = pygame.Surface(rect.size).convert()
13
14     # Copy the sprite from the large sheet onto the smaller image
15     image.blit(sheet, (0, 0), rect)
16
17     # Assuming black works as the transparent color
18     image.set_colorkey(COLS.BLACK)
19
20     # Return the image
21     return image
22
23
24
25 def stretch_image(imagenname, progression, rect=None):
26     """
27     This function is to stretch out an image, where the progression is a value
28     which defines how far along in the
29     process of the stretch it is (the stretch is non linear)
30
31     imagenname could be a string, or could be an instance of an image

```

```

31 :param imagenname:
32 :param progression:
33 :param rect:
34 :return:
35 """
36
37 if isinstance(imagenname, str):
38     sheet = pygame.image.load(imagenname).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(imagenname).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 imagenname not being an imagenname, but rather
52     # an object of type image already.
53     h, w = imagenname.get_height(), imagenname.get_width()
54     return pygame.transform.scale(imagenname, (w, h + progression ** 2)), h
55     + progression ** 2
56
57 # Load a whole bunch of images and return them as a list
58 def getImages(sheet, rects):
59     "Loads multiple images, supply a list of coordinates"
60     return [getImage(sheet, rect) for rect in rects]
61
62 # Load a whole strip of images
63 def loadStrip(rect, image_count, sheet):
64     "Loads a strip of images and returns them as a list"
65     tups = [(rect[0]+rect[2]*x, rect[1], rect[2], rect[3])
66             for x in range(image_count)]
67     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,
20     GREEN,
21     LIGHTGREY,

```



```
22     TEAL ,
23     YELLOW ,
24     RED
25 ]
26
27 random_colors = [
28     '#281ed5' ,
29     '#c79e32' ,
30     '#661b61' ,
31     '#918738' ,
32     '#a98996' ,
33     '#6b9362' ,
34     '#77cc12' ,
35     '#45e61a' ,
36     '#c1656b' ,
37     '#9e8dcf' ,
38     '#141110' ,
39     '#e537d8' ,
40     '#e6db9e' ,
41     '#f4ece7' ,
42     '#2b6b3c' ,
43     '#2c1873' ,
44     '#34179f' ,
45     '#f3e044' ,
46     '#9442ca' ,
47     '#b8268f' ,
48     '#dd250d' ,
49     '#25174d' ,
50     '#78c869' ,
51     '#d66d47' ,
52     '#ea5e97' ,
53     '#68250b' ,
54     '#ac5e27' ,
55     '#8e1c3d' ,
56     '#ed6209' ,
57     '#c32463' ,
58     '#6139cc' ,
59     '#6dc947' ,
60     '#c243d4' ,
61     '#6531a0' ,
62     '#d63b14' ,
63     '#a9409b' ,
64     '#3fb86a' ,
65     '#6b5d60' ,
66     '#b0cf25' ,
67     '#70c091' ,
68     '#934039' ,
69     '#923df4' ,
70     '#ae5d96' ,
71     '#ec9bd8' ,
72     '#cd2440' ,
73     '#c47415' ,
74     '#ec8312' ,
75     '#44120c' ,
76     '#f1e80d' ,
77     '#ed5d6a' ,
78     '#f1dd73' ,
79     '#e7622a' ,
80     '#986bca' ,
81     '#bbcbbd' ,
82     '#3ca138' ,
83     '#e7bde2' ,
84     '#ac232d' ,
```

```

85     '#101f9c',
86     '#326996',
87     '#4ca8b1',
88     '#821cbc',
89     '#94c5e1',
90     '#4c7cbc'
91 ]
92
93 title_colors = [
94     (249,52,242),
95     (0,0,0),
96     (32,28,208),
97     (249,36,4),
98     (255,255,255)
99 ]
100 edge = [
101     (180,46,38),
102     (249,46,0),
103     (255, 130, 45),
104     (255, 130, 45)
105 ]

```

const.py

```

1  """
2  These constants control game play. Most of these constants are adjustable and
3  will adapt automatically for the game.
4  """
5  from math import sqrt
6
7  VELOCITY = 6
8  DVELOCITY = sqrt(2*(VELOCITY**2))
9  SCREENSIZE = (800,600)
10 TPS = 28
11 TITLE = 'Robotron 2084'
12 PROJ_VELOCITY = 8
13 DPROJ_VELOCITY = sqrt(2*(PROJ_VELOCITY**2))
14 COOLDOWN = 5
15 BORDER_W = 10
16 BORDERSPEED = 15

```

decorations

border.py

```

1  import pygame
2  from constants.const import *
3  from constants.colors import *
4
5
6  class Border(pygame.sprite.Sprite):
7      """
8      This border class is drawn around the edge of the screen. it is a sprite
9      itself, so it cannot be killed
10     """
11
12     def __init__(self):
13         pygame.sprite.Sprite.__init__(self)
14         self.color = (255,0,0)
15         self.image = pygame.Surface((SCREENSIZE[0], SCREENSIZE[1]))
16         self.image.fill(BLACK)
17         self.image.set_colorkey(BLACK)

```

```

17         self.drawrect()
18         self.rect.center = (SCREENSIZE[0]/2, SCREENSIZE[1]/2)
19
20     def update(self, __, __):
21
22         r,g,b = self.color
23         if r>0 and b == 0:
24             r -= BORDERSPEED
25             g += BORDERSPEED
26         if g > 0 and r == 0:
27             g -= BORDERSPEED
28             b += BORDERSPEED
29         if b > 0 and g == 0:
30             b -= BORDERSPEED
31             r += BORDERSPEED
32         self.color = (r,g,b)
33         self.drawrect()
34
35     def drawrect(self):
36         self.lines = [
37             pygame.draw.line(self.image, self.color, [0, 30], [SCREENSIZE[0],
38             30], BORDER_W),
39             pygame.draw.line(self.image, self.color, [0, SCREENSIZE[1]], [
39             SCREENSIZE[0], SCREENSIZE[1]], BORDER_W*2),
40             pygame.draw.line(self.image, self.color, [0, 30], [0, SCREENSIZE
40             [1]], BORDER_W*2),
41             pygame.draw.line(self.image, self.color, [SCREENSIZE[0], 30], [
41             SCREENSIZE[0], SCREENSIZE[1]], BORDER_W*2)
42 ]
43
44         self.rect = self.image.get_rect()
45
46     def die(self):
47         return

```

objects

bullet.py

```

1 import pygame
2 from constants.colors import BULLETS, BLACK
3 from random import choice
4 from constants.const import PROJ_VELOCITY, DPROJ_VELOCITY, SCREENSIZE, BORDER_W
5 from playsound import playsound
6 class Bullet(pygame.sprite.Sprite):
7     """
8     A class for the bullets
9     """
10    def __init__(self, x, y, dir):
11        playsound('./audio/shoot.mp3', block=False)
12        pygame.sprite.Sprite.__init__(self)
13        self.color = choice(BULLETS)
14        self.dir = dir
15        self.movx = 0
16        self.movy = 0
17
18
19
20        v = PROJ_VELOCITY if len(dir)<1 else DPROJ_VELOCITY
21
22        if 'N' in dir:
23            self.movy = -v

```

```
24     elif 'S' in dir:
25         self.movy = v
26
27     if 'W' in dir:
28         self.movx = -v
29     elif 'E' in dir:
30         self.movx = v
31
32     kill = 0
33     if len(dir)==1 and (dir=='N' or dir=='S'):
34         rotation = 90
35     elif len(dir)==1 and (dir=='E' or dir=='W'):
36         rotation = 0
37     elif 'NE'==dir or 'SW'==dir or 'EN'==dir or 'WS'==dir:
38         rotation = 45
39     elif 'NW'==dir or 'WN'==dir or 'SE'==dir or 'ES'==dir:
40         rotation = 315
41     else:
42         kill = 1
43         rotation = 0
44
45     self.image = pygame.Surface([25, 8])
46     self.image.fill(BLACK)
47     self.image.set_colorkey(BLACK)
48
49     pygame.draw.rect(self.image, self.color, pygame.Rect(0, 0, 25, 5),
border_radius=3)
50     self.image = pygame.transform.rotate(self.image, rotation)
51
52     self.rect = self.image.get_rect()
53     self.rect.center = (x,y)
54
55     if kill:
56         self.kill()
57         del self
58
59
60
61     def update(self, _, __):
62         self.rect.center = self.rect.center[0] + self.movx, self.rect.center[1]
+ self.movy
63         if (not (SCREENSIZE[1]-BORDER_W-30 >= self.rect.y >= 30 + BORDER_W) )
or\
64             (not (SCREENSIZE[0]-BORDER_W*4 -5 >= self.rect.x >= BORDER_W*2
- 5)):
65             self.kill()
66             del self
67             return
```