Odd Semester (2021)



**BINUS UNIVERSITY**

**BINUS INTERNATIONAL**

**Student Information :**

|  |  |  |  |
| --- | --- | --- | --- |
| **First Name :** Fiqhy | **Last Name :** Bismadhika | | **Student ID :** 2101714824 |
| **Course Code :** COMP6502 | **Course Name :** Introduction to Programming | | |
| **Class :** L1BC  **Major :** Computer Science | **Lecturer(s) :** 1. Minaldi Loeis  2. Jude Ramirez | | |
| **Assignment’s Title :** Shoot ‘em Up  **Submission Pattern :**  **Due Date :** 7/11/2017 | | **Type of Assignment :** Final Project  **Submission Date :** 7/11/2017 | |

# Plagiarism/Cheating

Binus International seriously regards all forms of plagiarism, cheating and collusion as academic offenses which may result in severe penalties, including loss/drop of marks, course/class discontinuity and other possible penalties executed by the university. Please refer to the related course syllabus for further information.

# Declaration of Originality

By signing this assignment, I understand, accept and consent to Binus International terms and policy on plagiarism. Herewith I declare that the work contained in this assignment is my own work and has not been submitted for the use of assessment in another course or class, except where this has been notified and accepted in advance.

Signature of Student

Fiqhy Bismadhika

**“Shoot ‘em Up”**

**Python Language Program**

**Name :** Fiqhy Bismadhika

**ID :** 2101714824

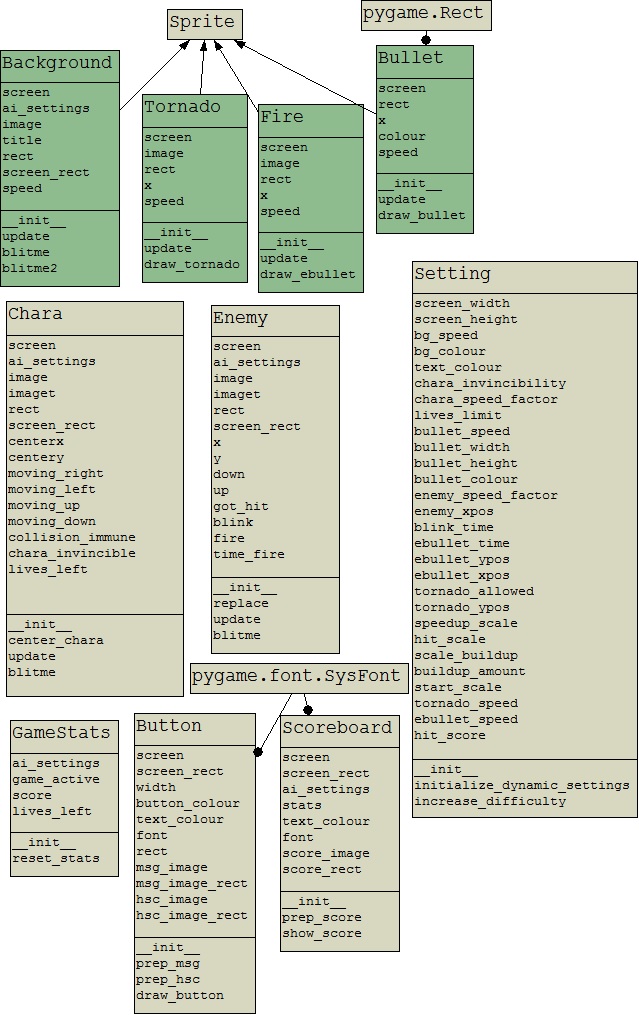
1. **Description**

**The function of this program:**

This program is made in python language version 3.6, PyGame module is used as the base program. Thus, a game is surely expected. The function of this program is to entertain the people who plays the game and used as a completion for the final project assignment which due before the mid term exam. The game that has been programmed is a simple 2D shoot ‘em up game and share several similiarities with the classic retro game Space War.

Shoot ‘em up is a subtype of game in which the player engages in a lone assault against a large number of enemies while dodging a barrage of attacks. The version of shoot ‘em up in this program does not contain multiple enemies, however it is switched into a single enemy in which the player can continuously attack to earn scores. The player is given three lives to keep earning scores, player lives can be lost due to the enemy’s attack and touching the enemy’s rectangle. Player’s high score will be displayed in the title screen. The overall of the game’s picture and music used are heavily based on an existing game called Cup head.

1. **Program’s Design**

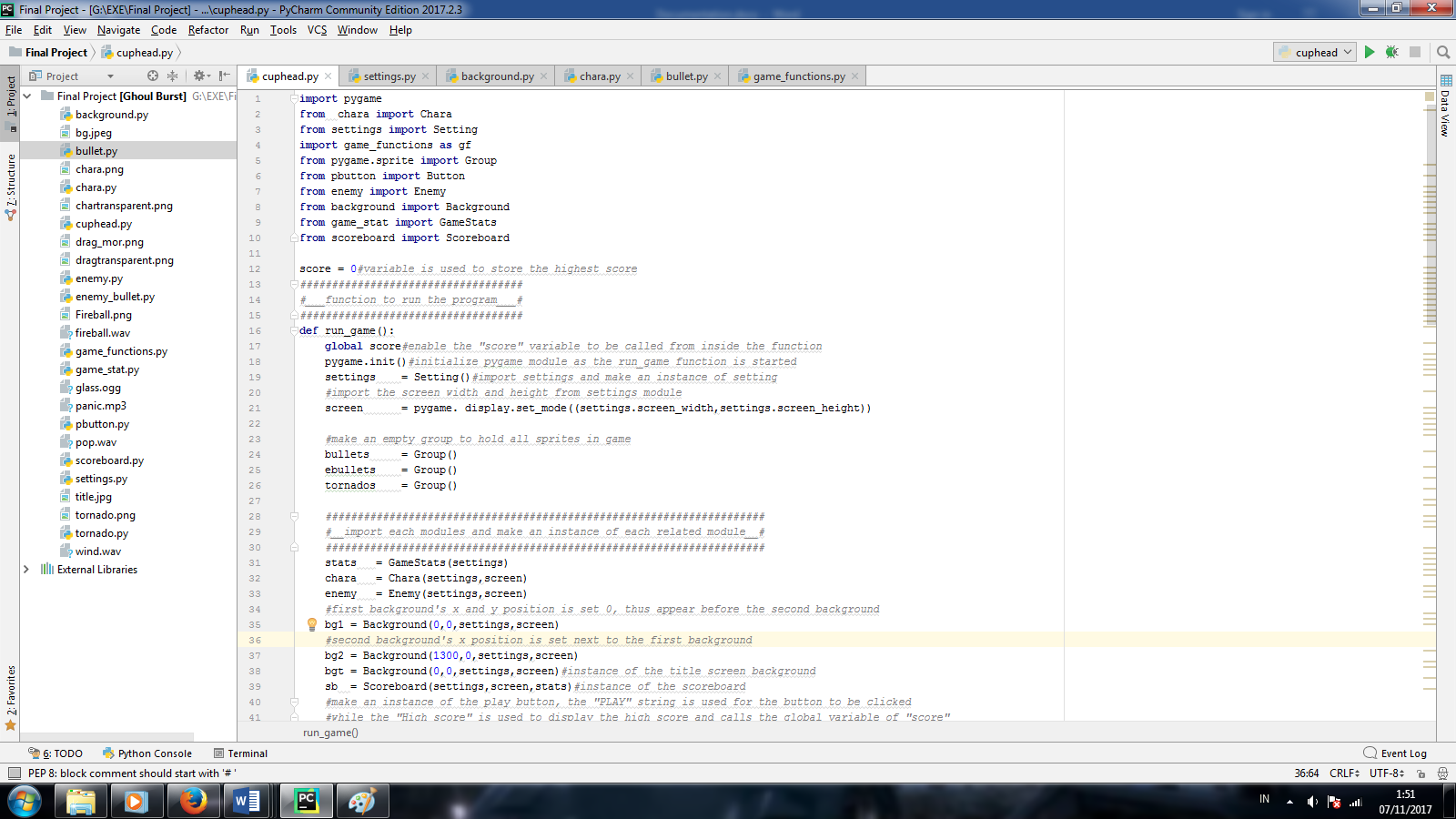


**Project’s UML Paragram**

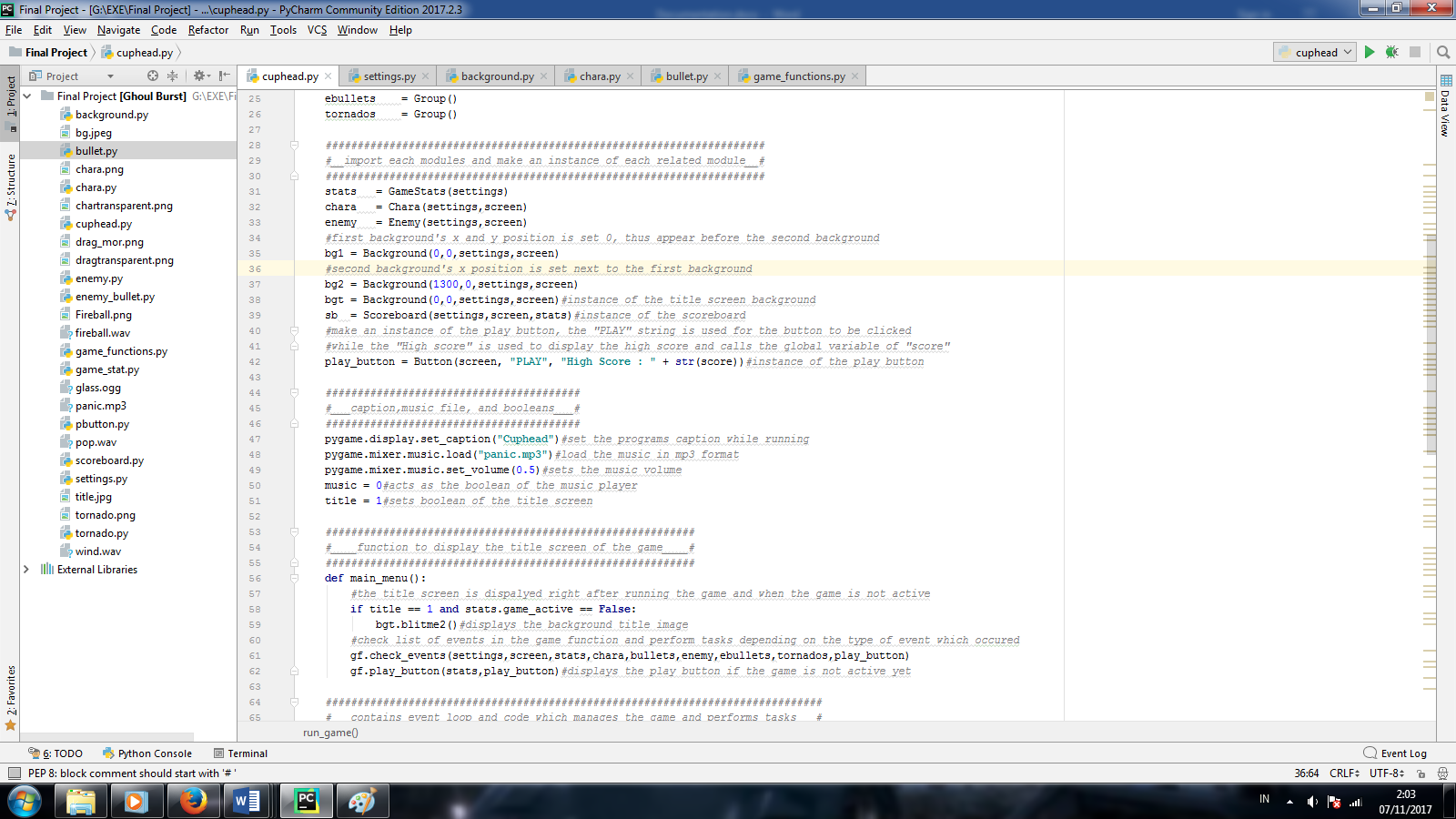
1. **Programming the Game**

The program or game that I had worked on for three weeks in total, relies on several modules. Each modules contains only a single class, this way the code will become easier to check whenever an error occured. There are twelve modules that I made during the proccess, it might sounds overwhelming to overview that many modules, but it is actually not.

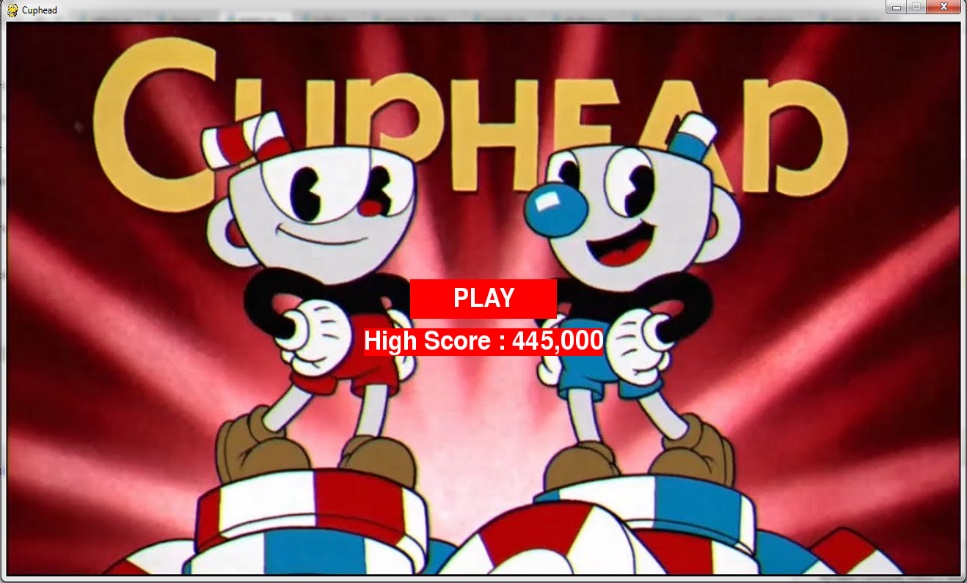
To make it simple these modules can be divided into four different type of handlings. The first type is the main module, it handles how the game works in sequence and only calls function from the second type of handling, which is called the game function. The game function contains many functions to make classes that were made works, and can be called by the main handling. The third type is class, class can be more than one and used to create object which will be used by the main function and contains function for the use of game function. The last but not least is the settings handling, the settings contains all sort of variables and values that will be used by all other three types of handling.

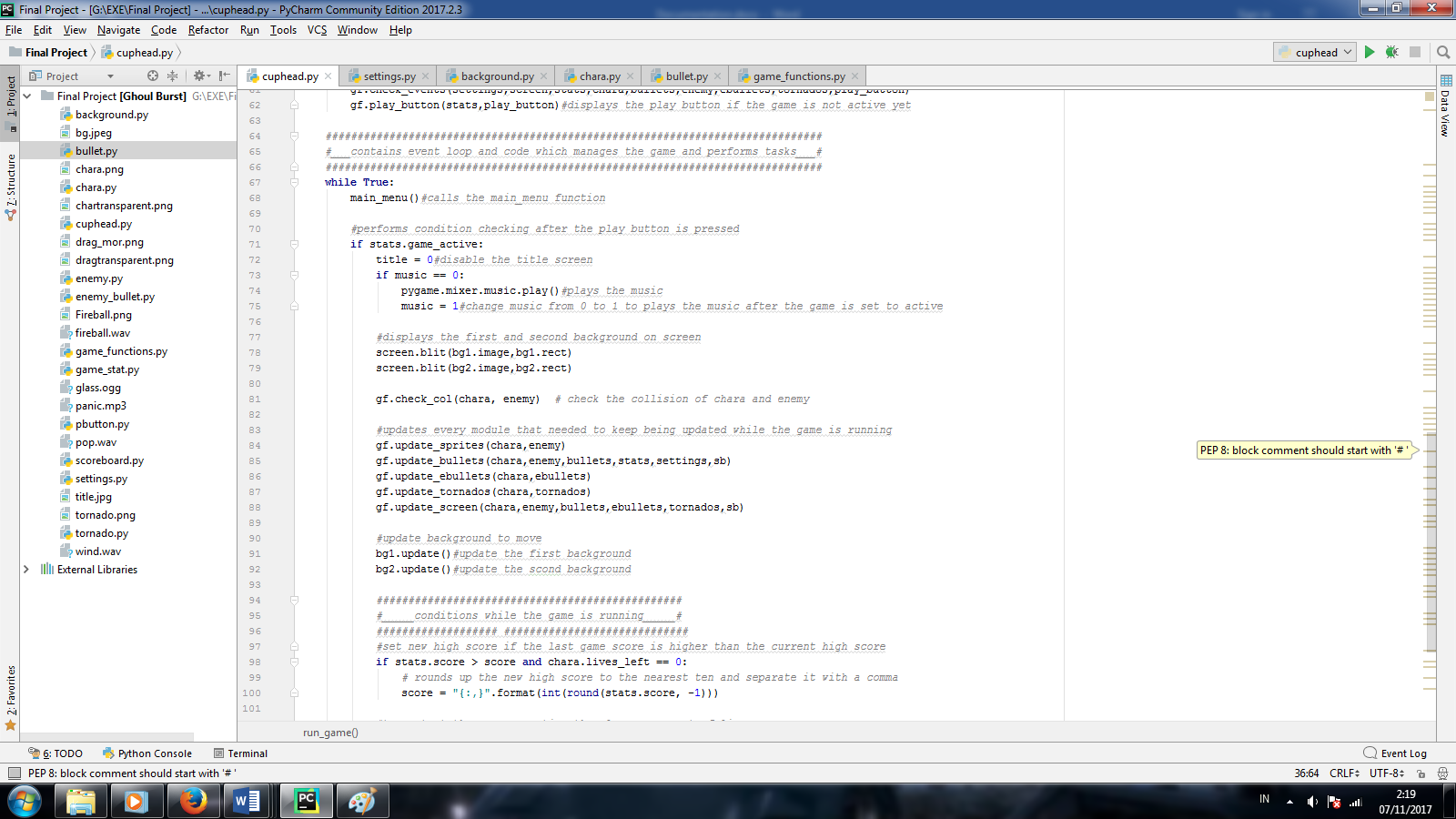
With the types already explained, We can go further to see the game’s code. The first part to be displayed is the main module. The main module import all other modules to be used while running the game.

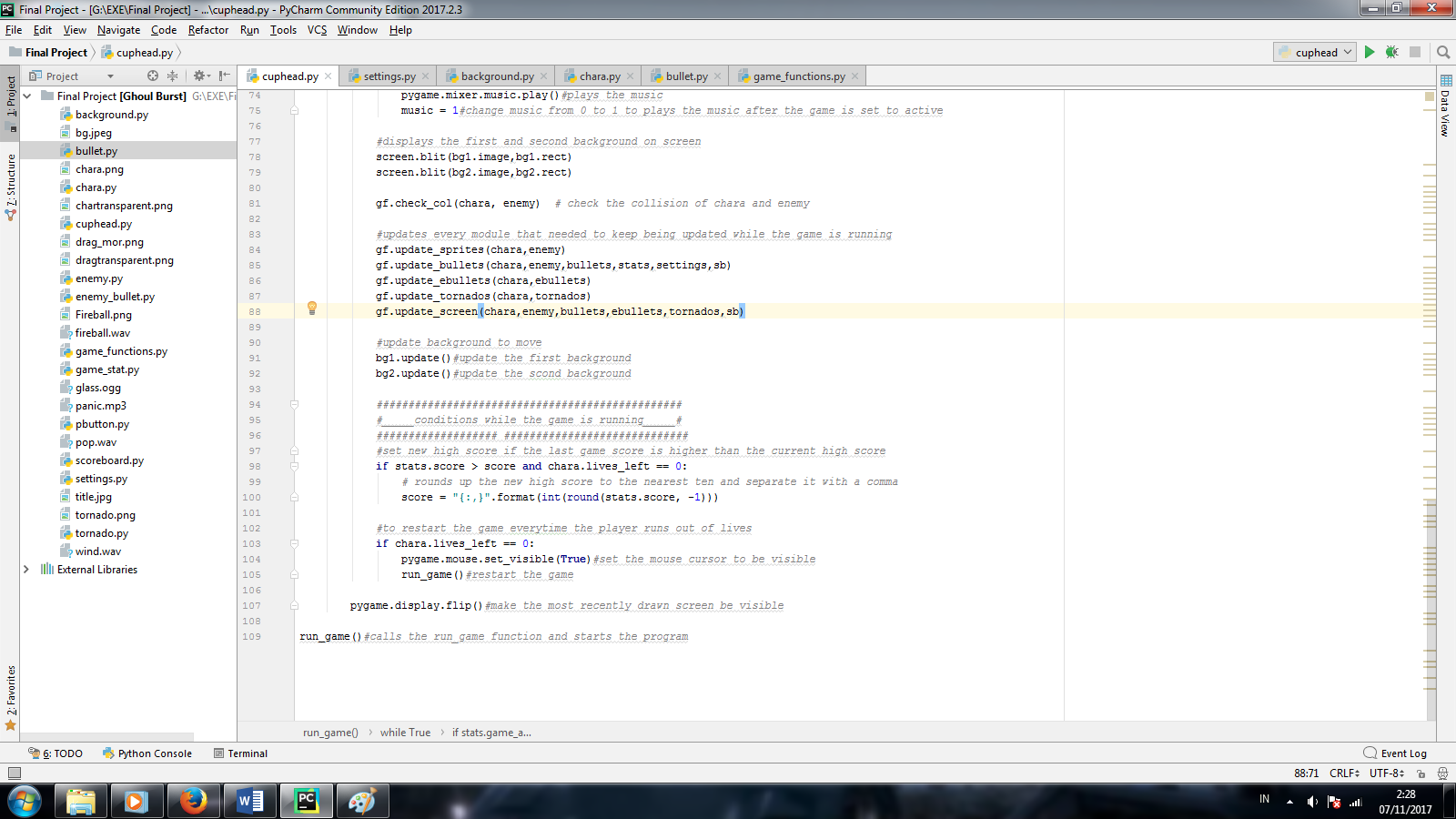
There is a “score” variable outside the function because it is used to store the high score and will not be lost when the program restart or rerun, and will be displayed in the title screen of the game.

The “score” variable is set to global so it can be called inside the function to take the value of high score. Pygame.init() is a function to initialize pygame module as the main function started. Screen is used to set the display resolution of the the pygame which it’s value is stored in the settings module

The class’ from other modules are made into objects or several instances inside the main function. Thus, the game can use these objects to be shown in the game.The program also plays music while the game is both running and active. To do so, the music file must be loaded with “pygame.mixer.music.load(‘file.format’)” then the music can be played with the code “pygame.mixer.music.play()” the music can only channel one music.play so if there is another code to do so, the music will replace the new one.

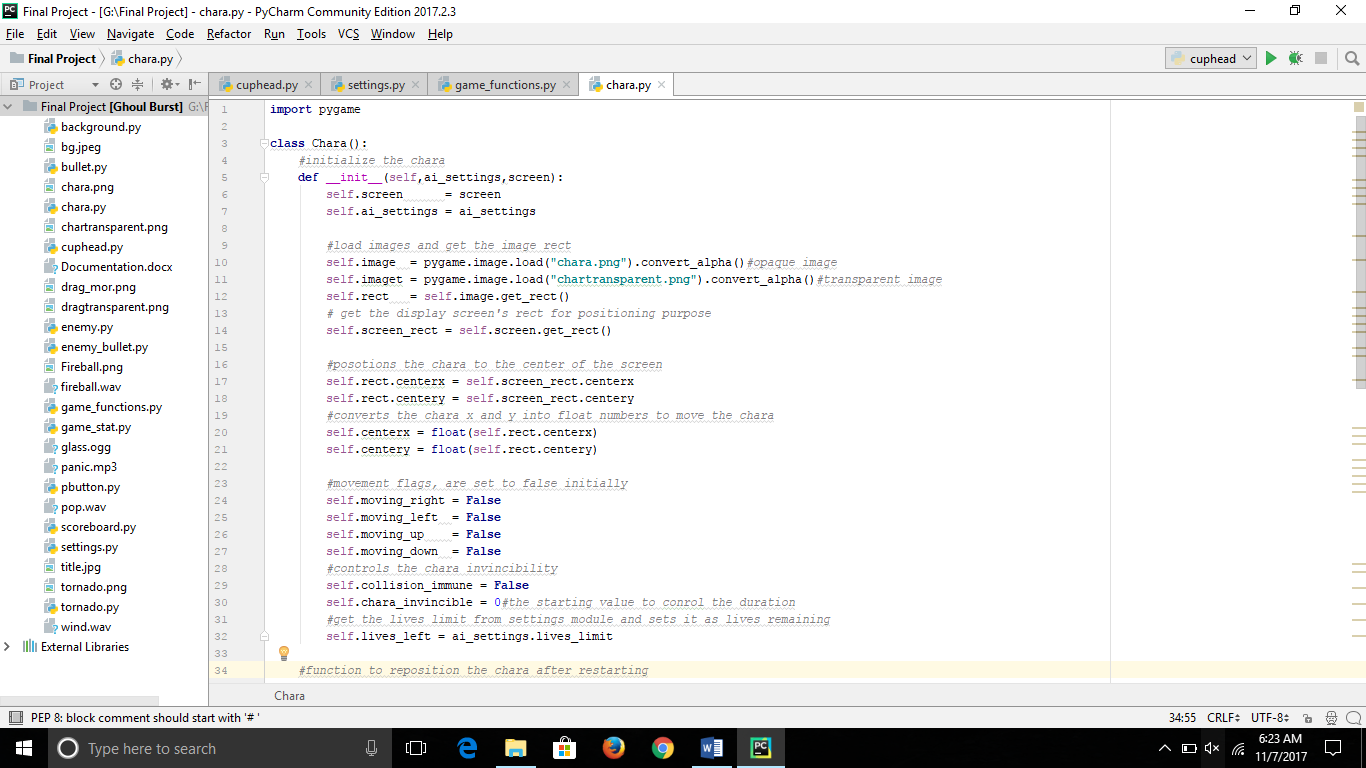
The game features title screen or main menu, in this state the game is not yet active and only shows the high scores, and the play button. The button can be left clicked to start the game and put the game state into active.

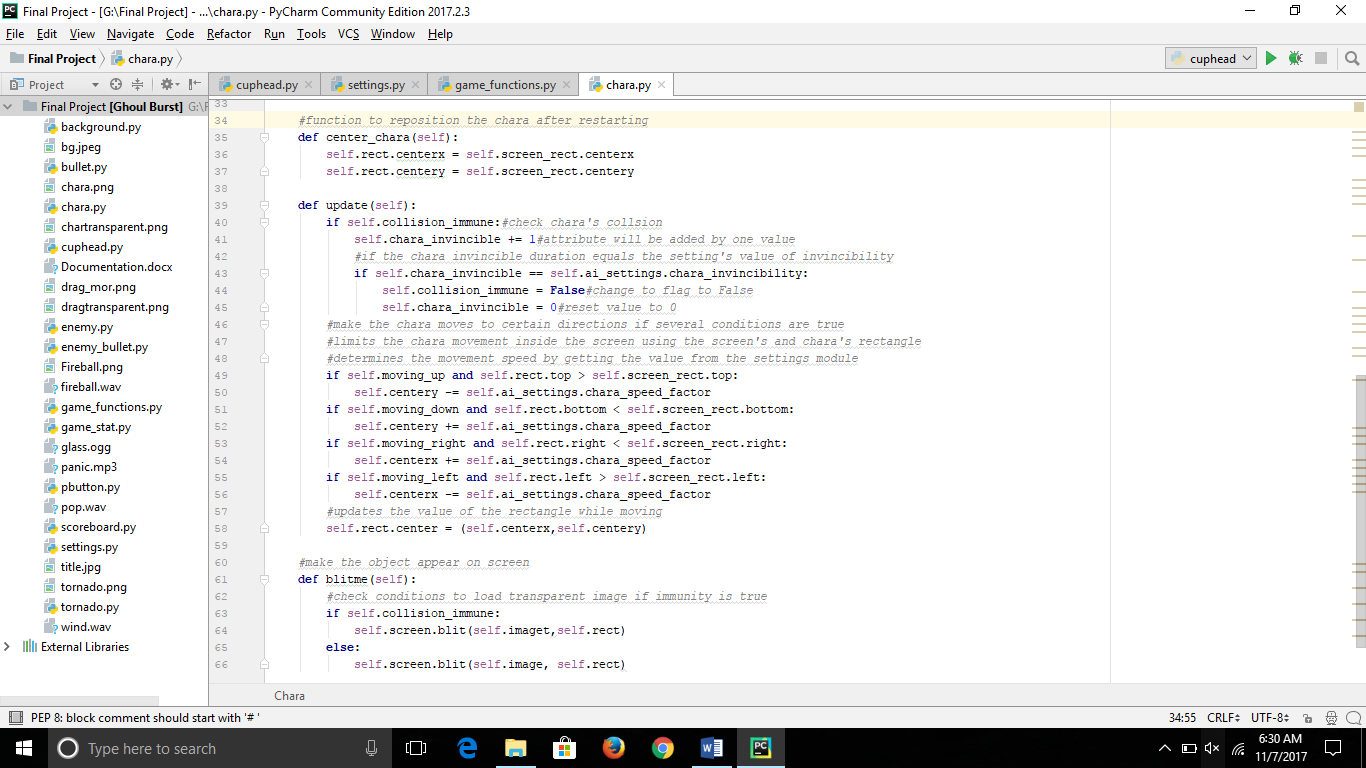
Inside the game main function, contains a “While True” loop. It is used to contains the event loops and codes which manages the game perform tasks. The loop checks the activity of the game, if it’s inactive then it will return to the main menu and vice versa. The loop keeps checking the updates required to make the games running and also display the updated objects on the screen by checking the events which pygame received.

At a certain point the game will end when the player doesn’t have anymore remaining life, and stops earning score. To ends the game, the program requires several condition to be checked. The game only checks the condition when the game state is running or True.

If the player lost all of their lives in the game, the game will stores the score in high score if only the last play score is higher than the current high score. The high score will be rounded to the nearest en and separated by a coma. After that, it will immidiately restart the game. Thus, returning the player to the title screen while showing the new high score. The “pygame.display.flip()” is to make the most recently drawn screen to be visible.

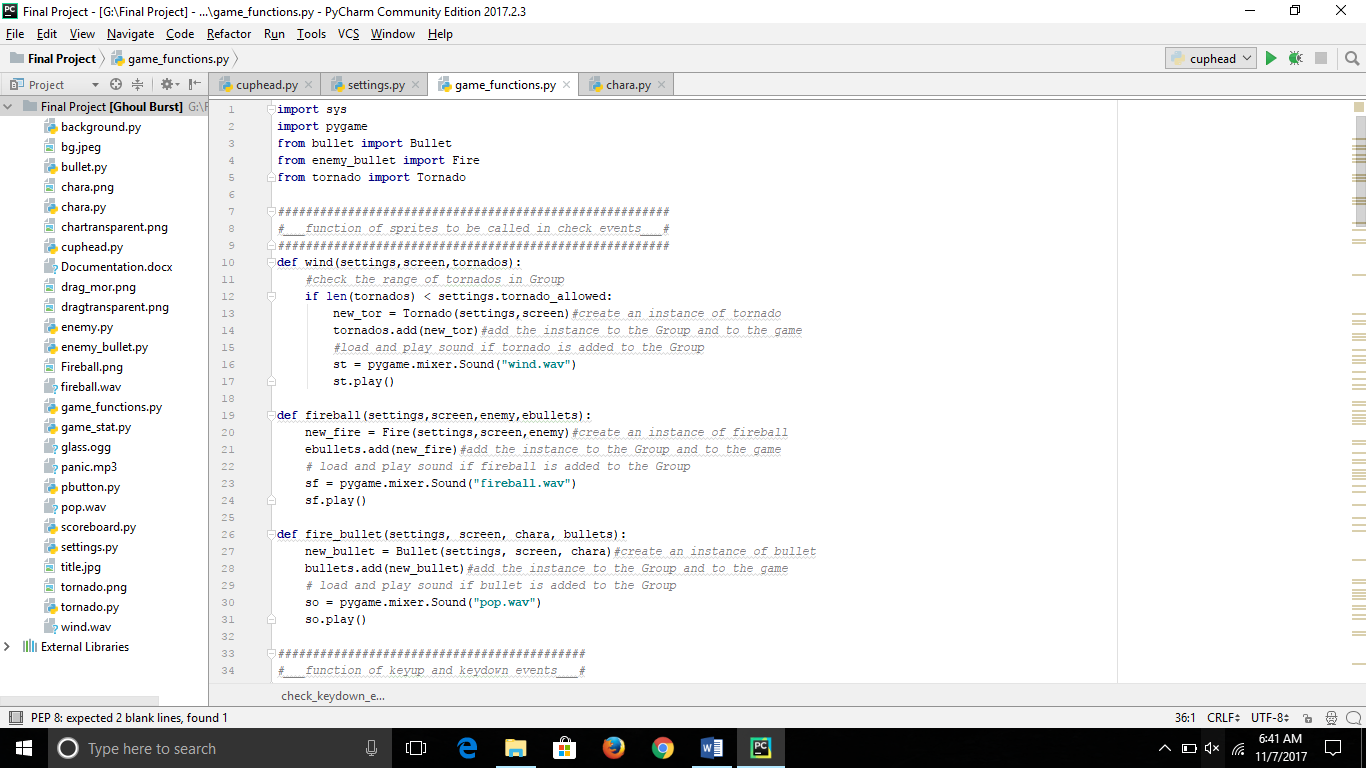
The picture above shows the in game display screen. There are several sprite classes in the code and each class controls each sprites functionality in the game. The player has chara class in which the player can move towards any direction while having the capabilities to shoot bullet which is created in different class, and contains the attributes for the function of shooting bullets to be used in the game function module.

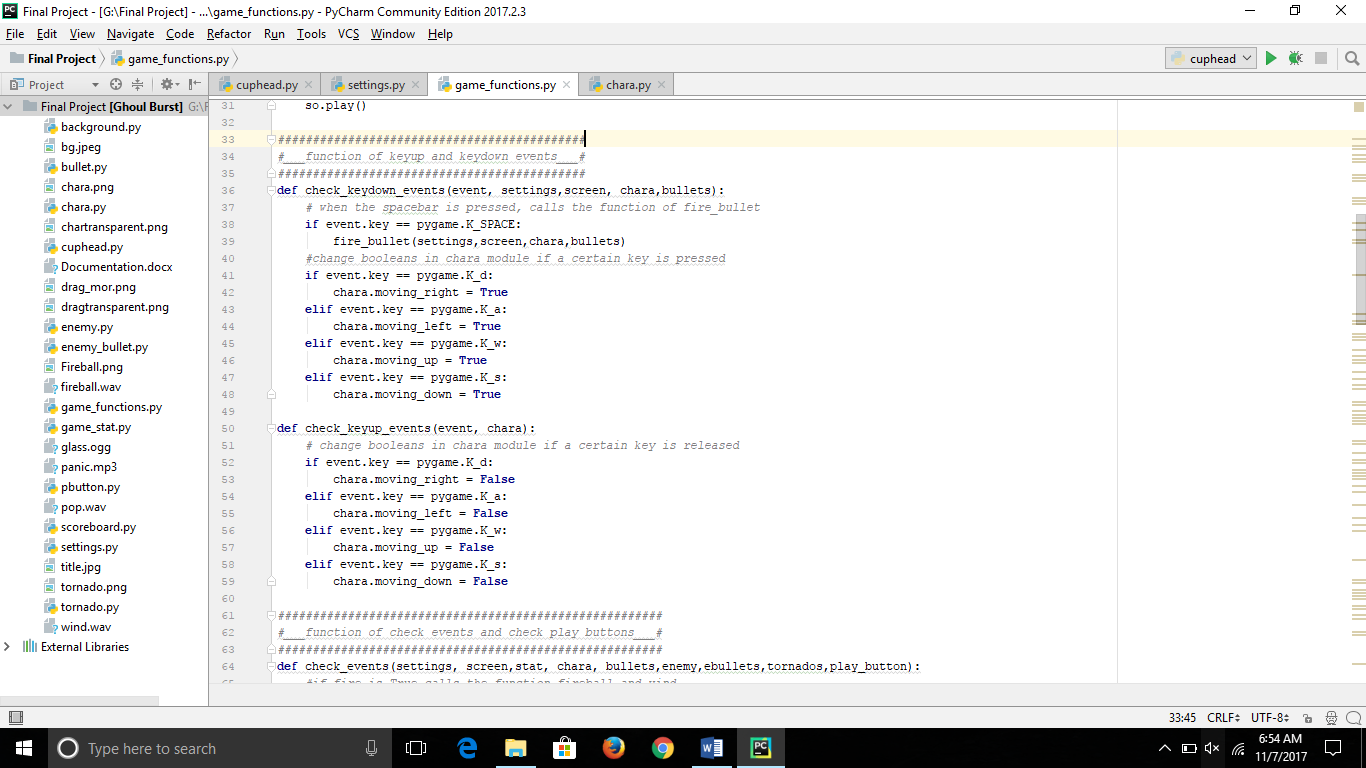
 The module besides is meant to show the example of the class type handling, the reason for this is because the classes shares the same functionality amongst other and also the vast amount of class the program has which mean going through the same thing over again.

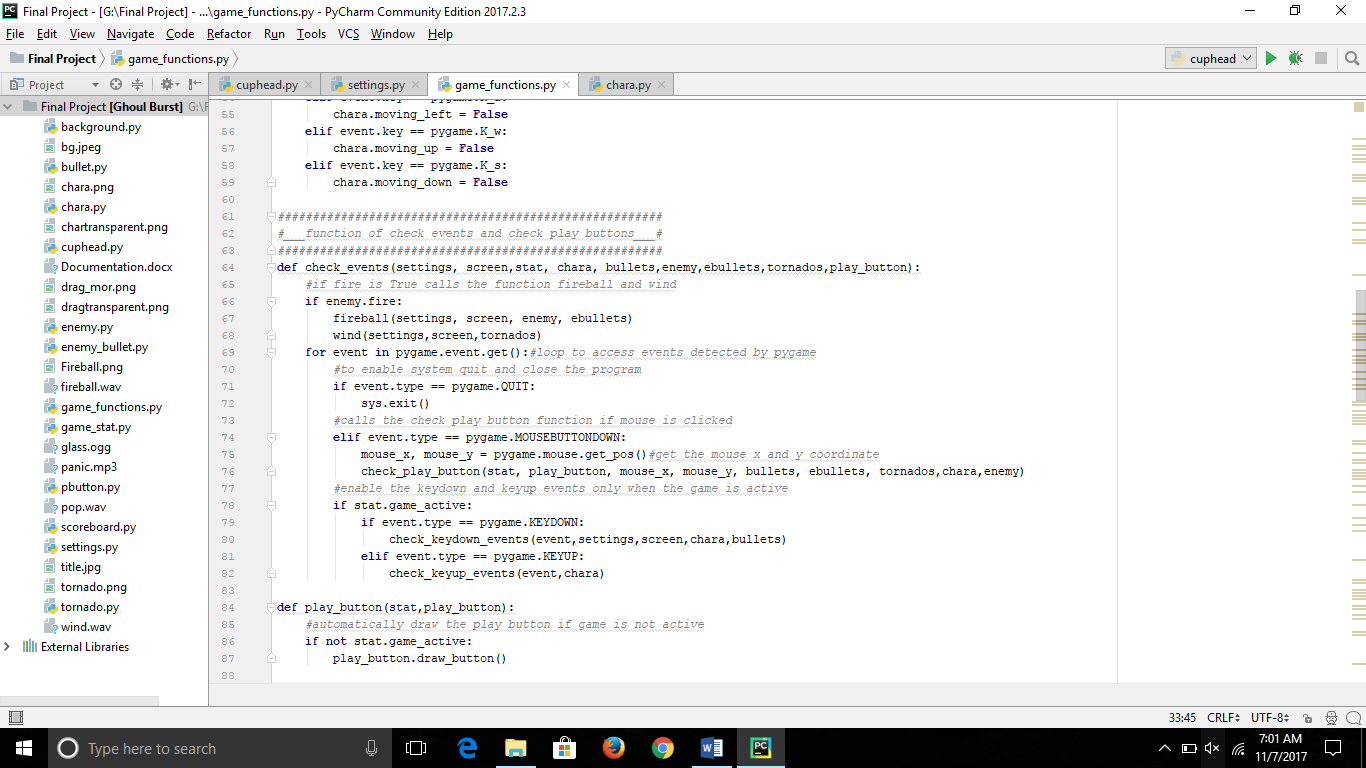
The chara class has it’s own attributes which will be used in the game function and also takes the attributes value from the settings module. The player starts the game in the center of the screen and be able to move to any directions within the screen boundaries, while using the booleans values set in the module.

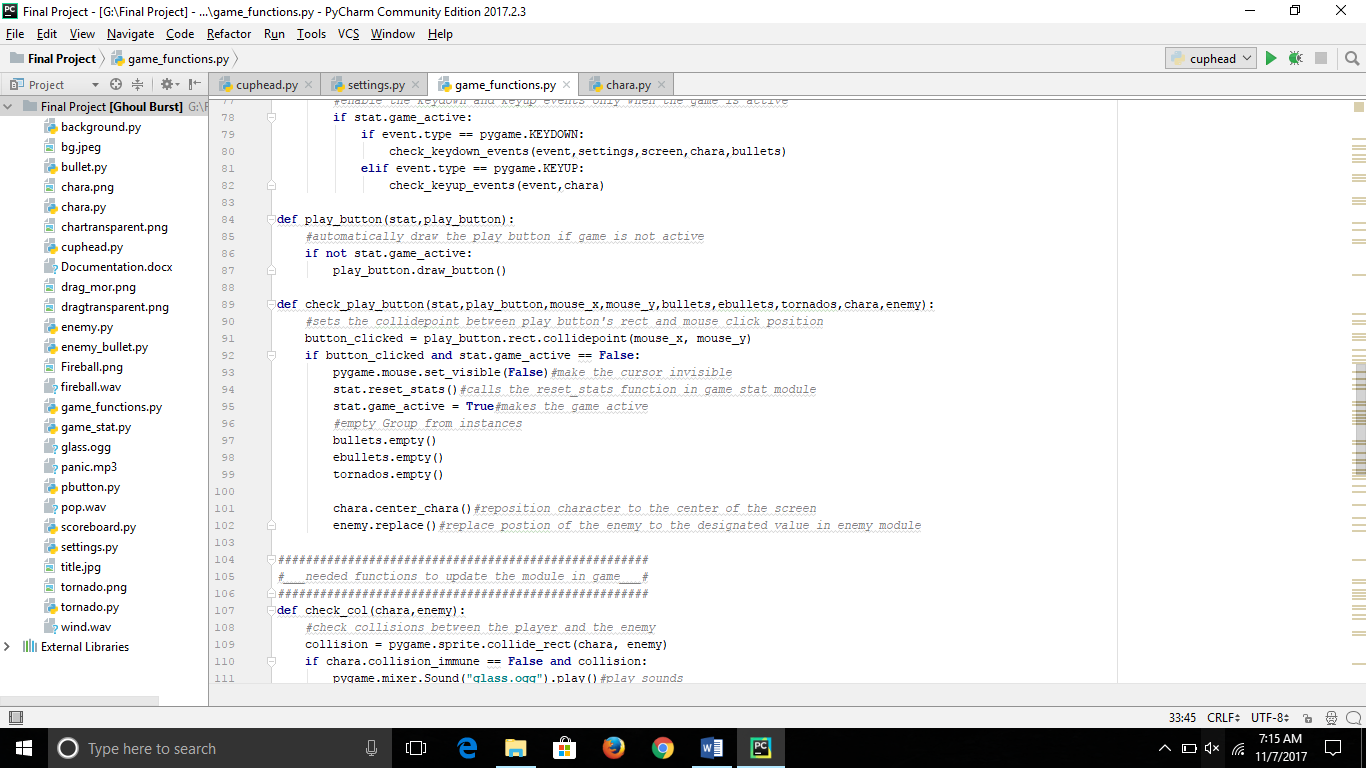
The player has limited lives which will lost when the player collides with other sprites in the game, everytime the player had a collision the player will goes invincible for a few seconds before it can take another collision. When the life reaches zero the game will restart back to the menu screen and the player stats, such as health, position, and score will reset back it’s original values.

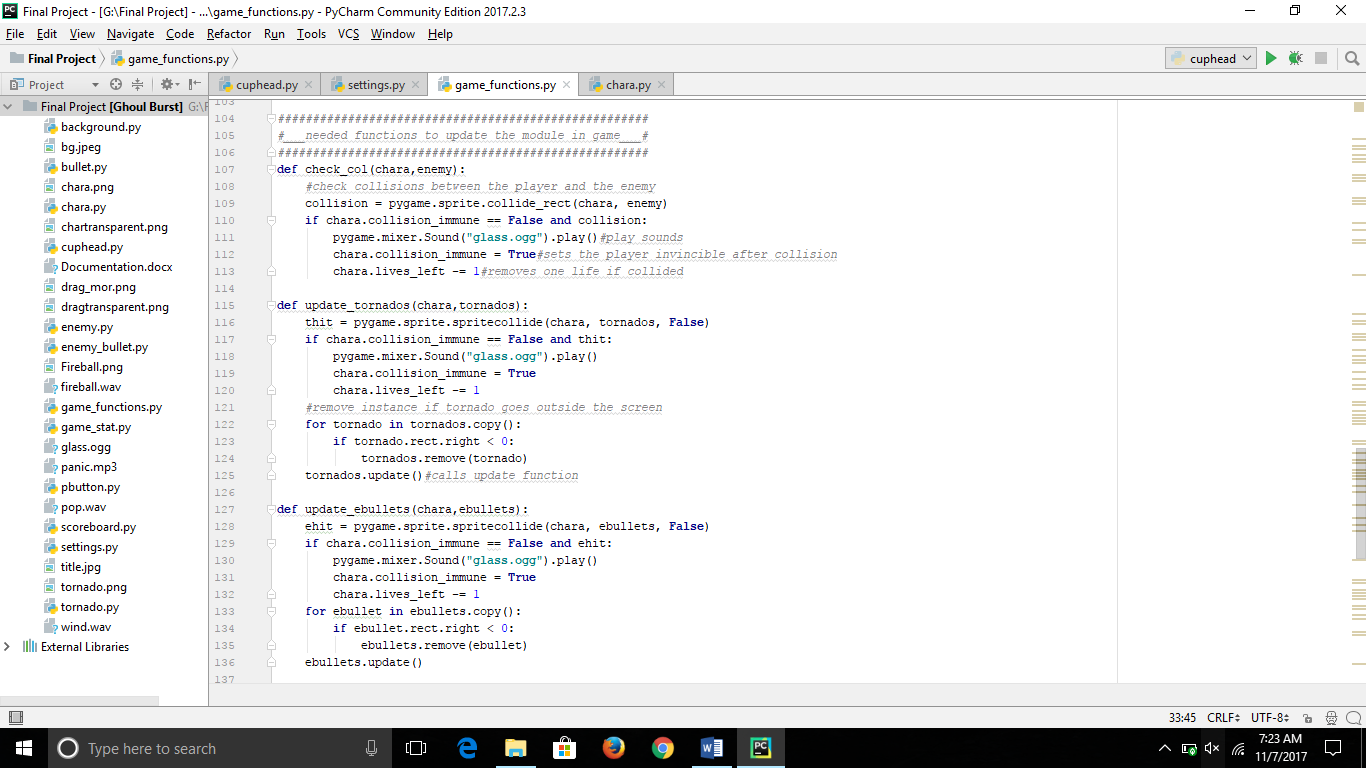
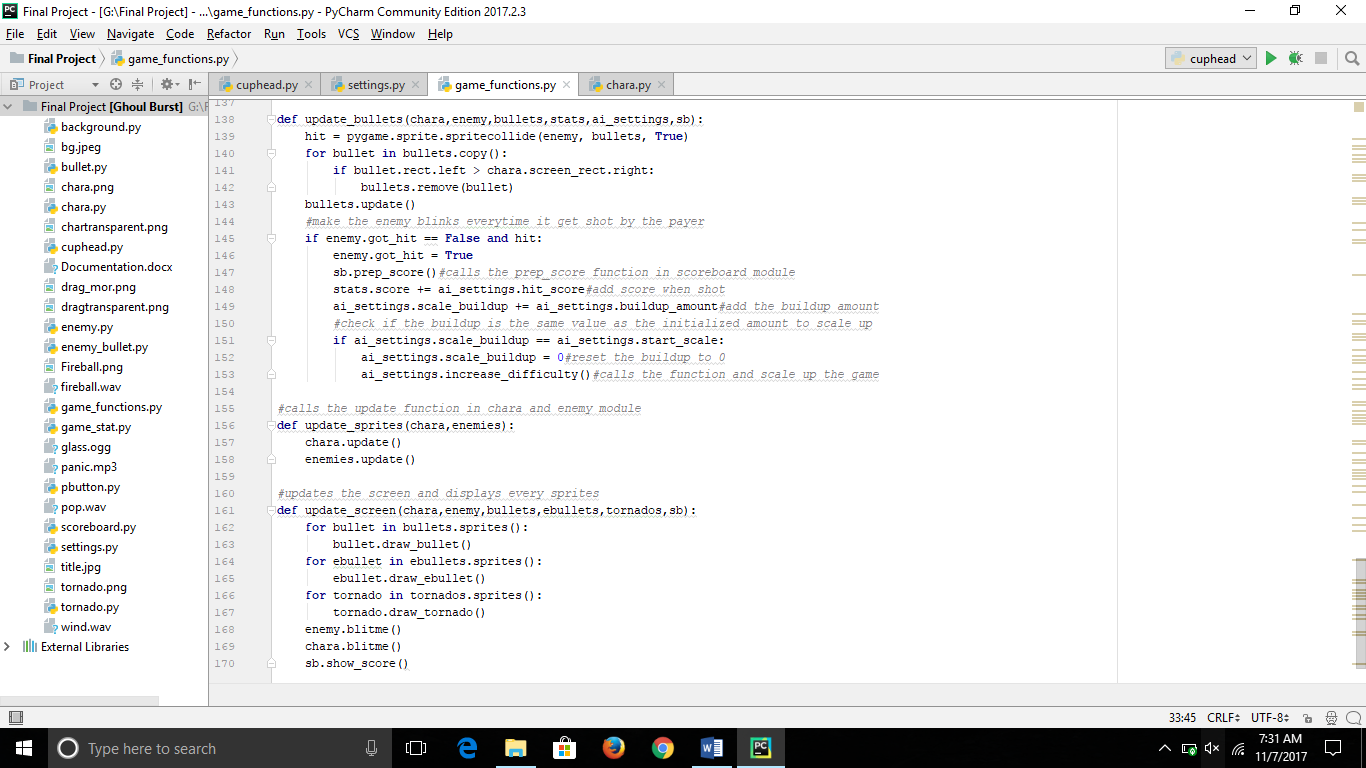
Each class has it’s own functionality but they are not assembled correctly just yet, thus we need a game function module. The module contains several functions in which each function does the core tasks that is crucial for the program to work both smoothly and correctly. Game functions imports all attributes from all the classes that are needed for the game, and also records or detects all events that are happening during the game.

This part of the game function manages the object sprites that are needed to be added while the game is running, unlike some other classes which already been a part of the game initially when the program starts. These object added itsel into the game when the trigger is active and object is below it’s limit number, also each object play different sounds if it’s added to the game.

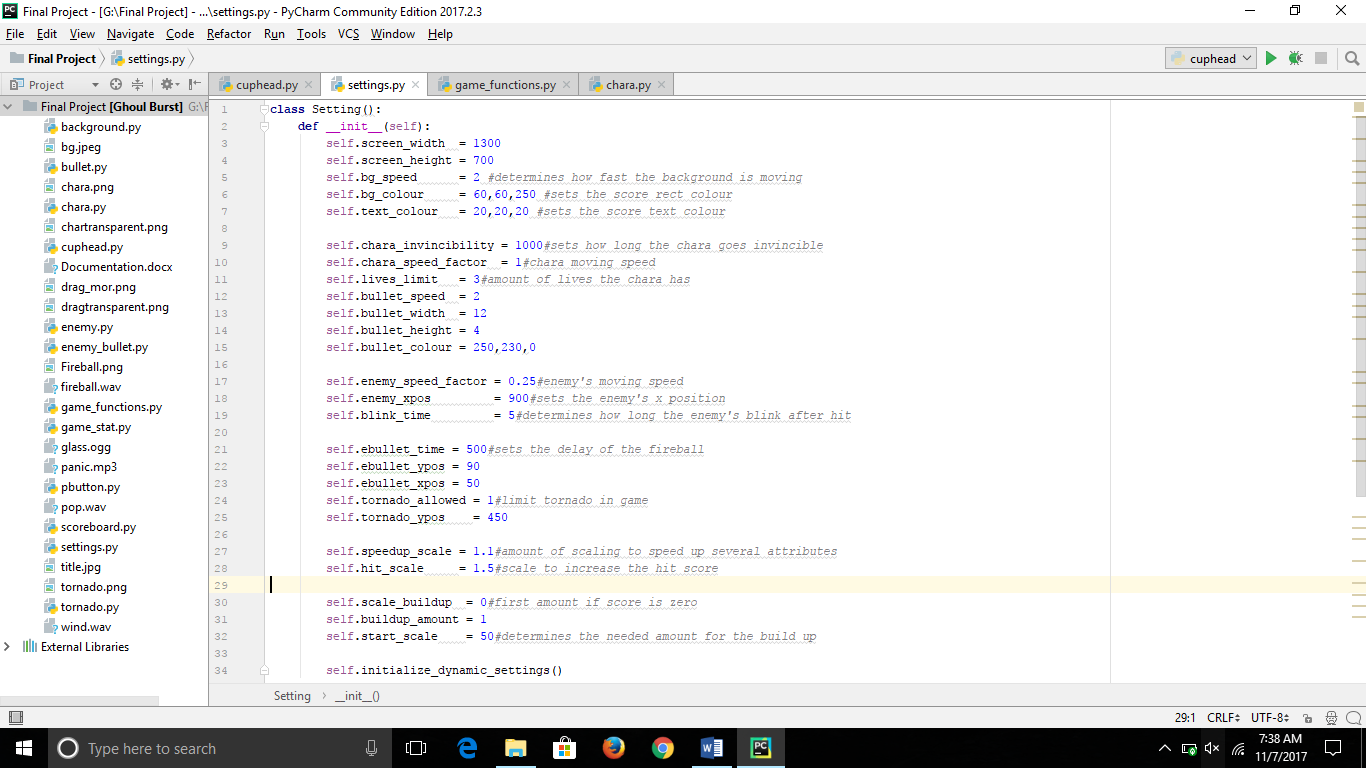
The game needs to respond to the user or player input, so we need a function that takes control of the events of the key in a form of booleans, in which the key will be pressed to respond and also the events that will happen after the key is released. The game uses WASD controls to move the player object around screen, and spacebar to start shooting bullets. Player stops moving when the key is released.

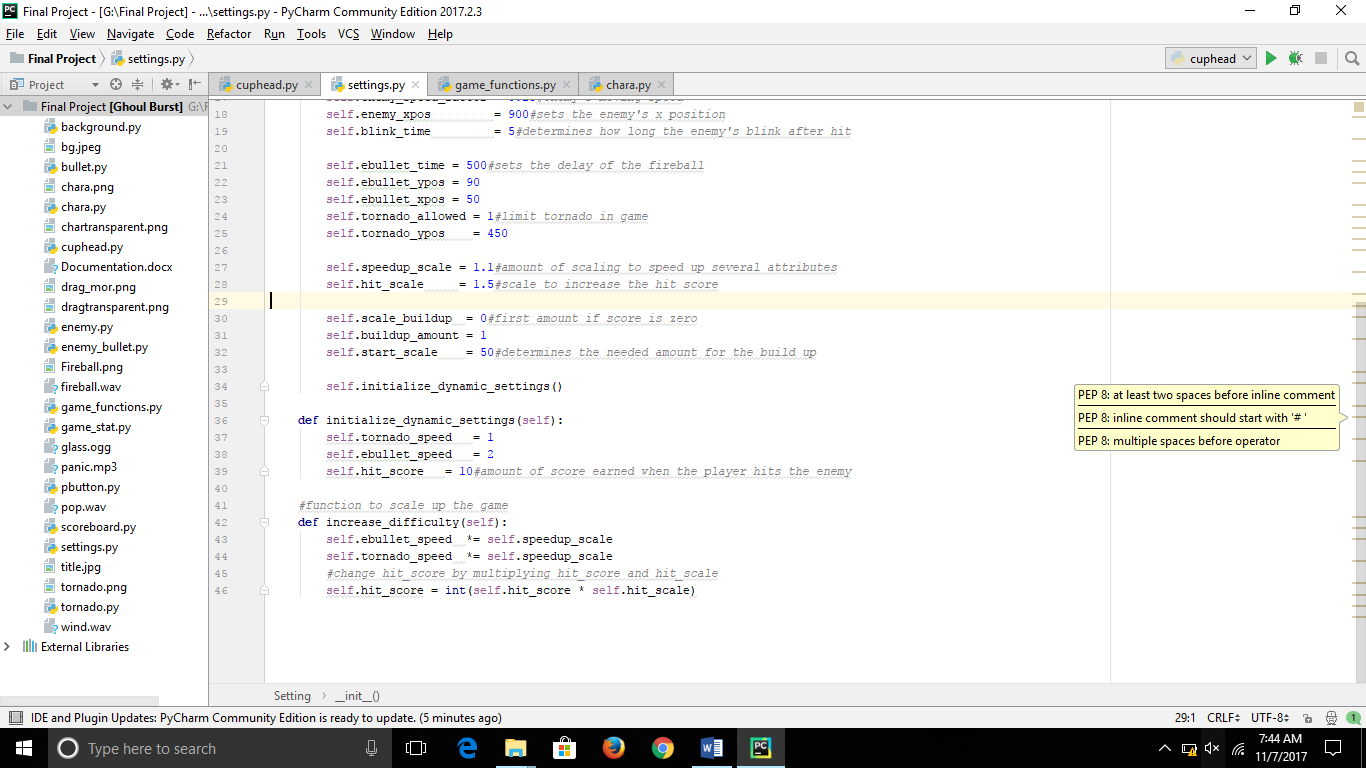
There will be several events other than those that will only respond to the player’s input, which are the other object sprites such as the enemy and it’s attacks to deal with the player. After several automatic triggers the game will spawn sprites that will act as the enemy and will make the player lost a life when collide. This function used as input for the player to starts the game after the play button in the main menu is pressed, and also enable the user to quit the game when the x system button is clicked manually.

The play button has it’s own function, starting with it getting the cursor’s x and y position in the game. While the cursor is within the button’s rectangle and left mouse clicked the game will change it’s state to active and the game will start while the main menu will be disappeared and will not respond to the mouse click anymore. The game will reset it’s stats back to the original values and also empties out the Group in the main folder while the sprites position being reset.

To check the collision of sprite objects in the game it needs it’s own function and defines what would happen if a collision happens during the game. And this part of function takes care of what happened if the sprite goes outside the screen rectangle, in this example, the bullets will simply remove itself, and each objects will trigger events that will occurred if a collision happens while it will keep being updated in the game. After the player’s bullets hit he enemy’s rectangle for a certain amount of times, the enemy object sprites will scale up which will make the game harder near the ending game.

Besides are functions to keep the object updated and display it in the display screen of the game.

All the classes and the game function modules needs a value that can be easily set and edited. Thus, we need the settings module which contains several attributes that has values to be called by other function from each different modules. all the values should be set to integers, not string because the program only needs integer value to be able to runs and works properly.

The settings module contains function of a dynamic settings from speed to the player scoring points and are used to scale several values after the trigger is active while the game is running.

**References :**

1. The program’s codes are based on the pygame section of python crash course
2. Several code functions learnt from : http:// [www.stackoverflow.com](http://www.stackoverflow.com)
3. The existing cuphead game for the images and music