# DESCRIPTION

Open all files into your python IDE. Navigate to the ‘main.py’ file and run this. The ‘main.py’ file sets up the screen display surface (height/width of the game screen). Level layout defined by ‘settings.py’ is mapped to the screen surface and the frame rate is set to how many milliseconds since previously called. These variables are called by the other python files of the game. The surface background of the main menu is set to a background image (‘5.png’). A caption of the game title is displayed on the screen followed by four buttons which have been scaled and positioned. Each button image has been defined in an event loop so that when clicked, an action follows. The first three buttons lead to other pages, and the fourth mutes or unmutes the game’s music. Music and the ability to mute and unmute it is defined in ‘main.py’.

The first button leads to the instructions page ( defined in ‘main.py’). This new screen contains an image which has been overlayed with text and rescaled to the size of the page. There is a back button on this page, which takes the player back to the main menu.

A game method is defined on ‘main.py’. This determines whether the music in the main game is muted or not. The event loop of this method indicates that if a player is dead and the return key is pressed, the level is reloaded, and the player is redirected to the main menu again. Code in the event loop specifies that if the escape button is hit whilst in the main game, the player will be redirected to the main menu, although the game does not restart until the player closes the game. The quit button closes the whole game window.

All other python files pertain to the main game. The premise of the game is that the player must navigate the screen using the arrow buttons on their keypad. As defined in ‘level.py’, the game is side scrolling, so every defined attribute of the environment moves forward as the player reaches the end of the screen. The player can move left, right, jump up and face either left or right when shooting as defined in the ‘player.py’ file. The object of the game is for the player to avoid being shot at by enemies and kill the enemies first. Both enemies and the player can shoot to kill one another when in a certain range (500 pixels) from each other. The projectile attributes are defined in ‘projectile.py’. Within ‘level.py’ a method is defined so that when the bullet from the opposing side hits a character, that bullet stops existing and deducts health from the player/enemy health bar, turning it from green (healthy) to red (eventually dead). The rate at which bullets can be fired is also controlled in the ‘level.py’ file. The different enemies shoot bullets of different widths, making it harder to avoid.

Through the course of the game, the player can collect three different items: health, dmg boost and jump boost. The health item improves the players health, so that they have a longer life. The dmg boost changes the colour of the bullet to purple, thus increasing the power of the bullet by double. The jump boost increases the height that the player can jump. These items are needed to defeat the boss character. If killed during the game, a screen will show saying ‘you died’, the score obtained and to press ‘enter’. When enter is pressed by the player, they will return to the main menu. If the player survives, they will be able to teleport to a new room. The teleportation function is defined in ‘main.py’ in the event loop. If the Boolean function in the event loop for teleporting the character is true, the teleported player attributes are equal to the player before teleportation. The player will be teleported once they have killed the red monster (‘Big Enemy’). When teleported, a new level loads (the boss room). If the player kills the boss, a screen appears showing that the player has defeated the boss, their score, and to ‘press enter’ to return to the main menu. The game is reset.