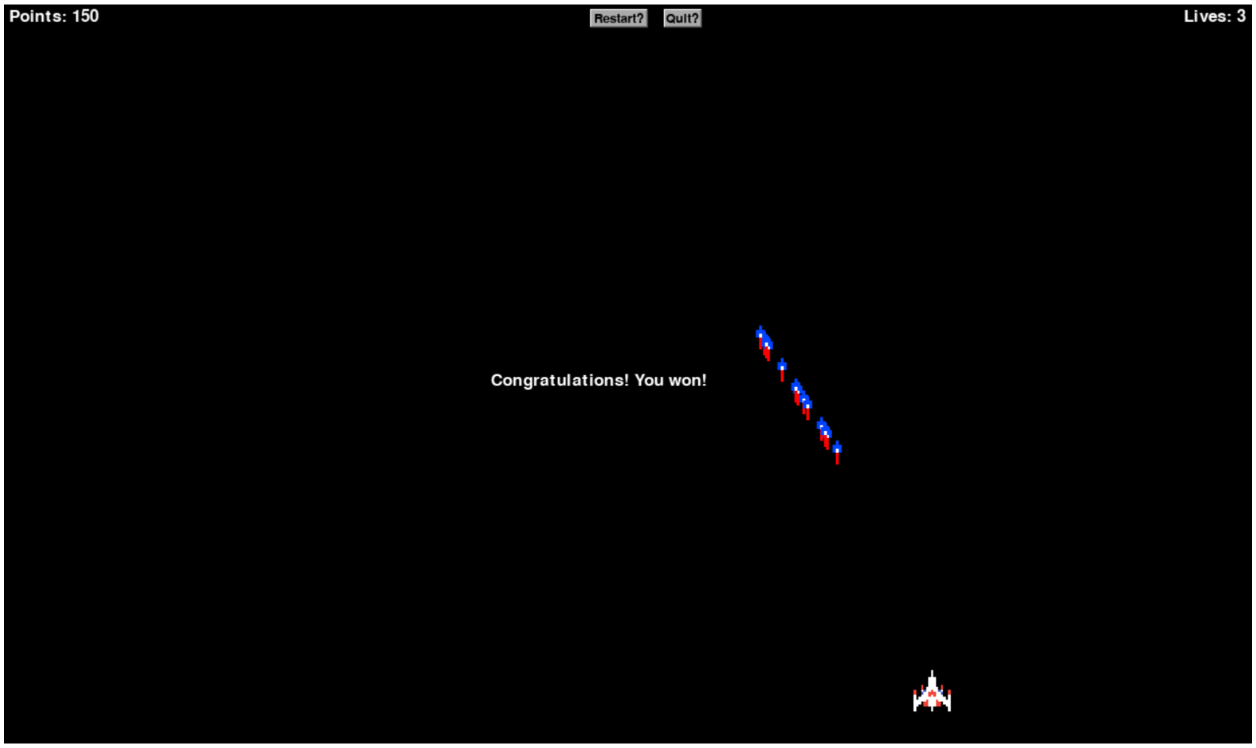
Aidan Butcher

2022-12-11

CS 1450 - Intro to OOP

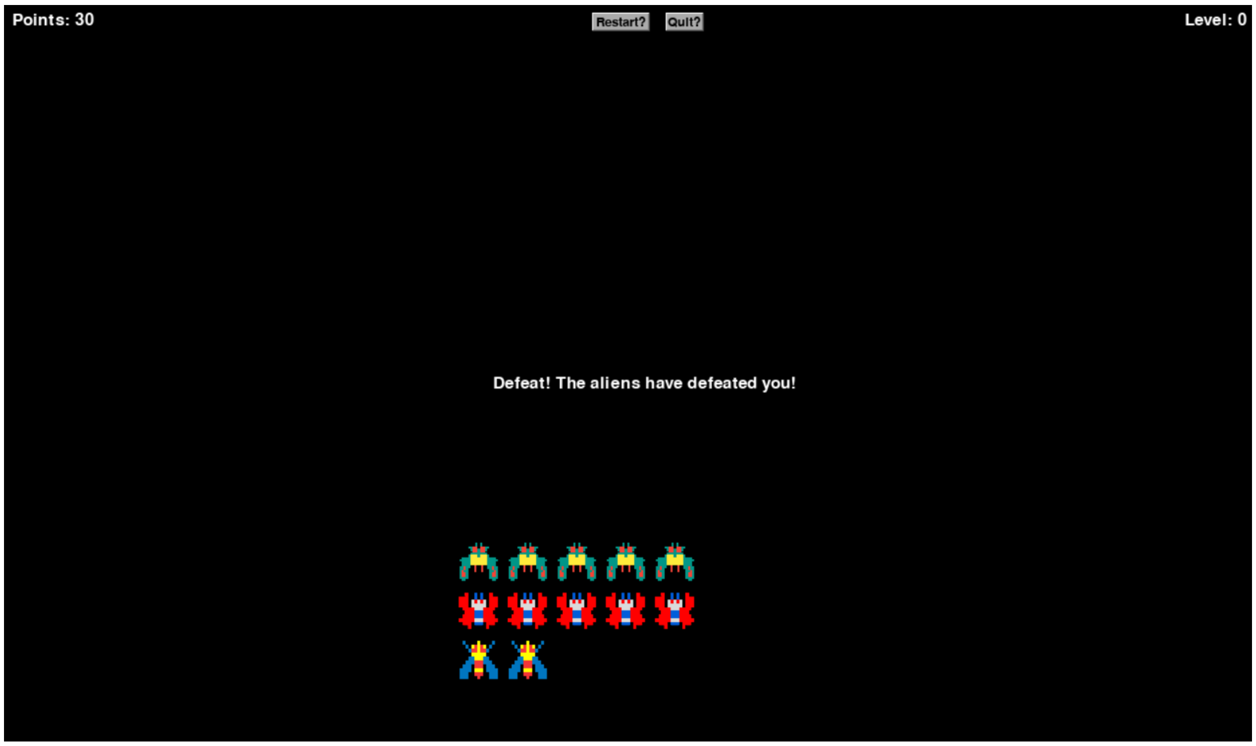
Final Report

For my final project, I decided to make a Space Invaders like game because it is a very rudimentary game for introductory purposes. I was hoping to make it more like Galaga where the aliens had more complex movement patterns and could even fire projectiles, but I ran out of time in making that. The game starts with a set of 15 aliens spawning and moving back and forth while moving downwards. Their goal is to crash into the player in order to destroy their ship and the player tries to fire missiles at the aliens in order to destroy them. The player gets 10 points for every alien they destroy, when the player destroys all the aliens they will have beaten the game. Some of the aliens have more health than others, I was actually planning on having a boss fight after the initial wave of aliens were defeated, but I ran out of time. Here is an example of the victory screen:



Victory text appearing after all the aliens are destroyed.

Here is an example of the defeat screen:

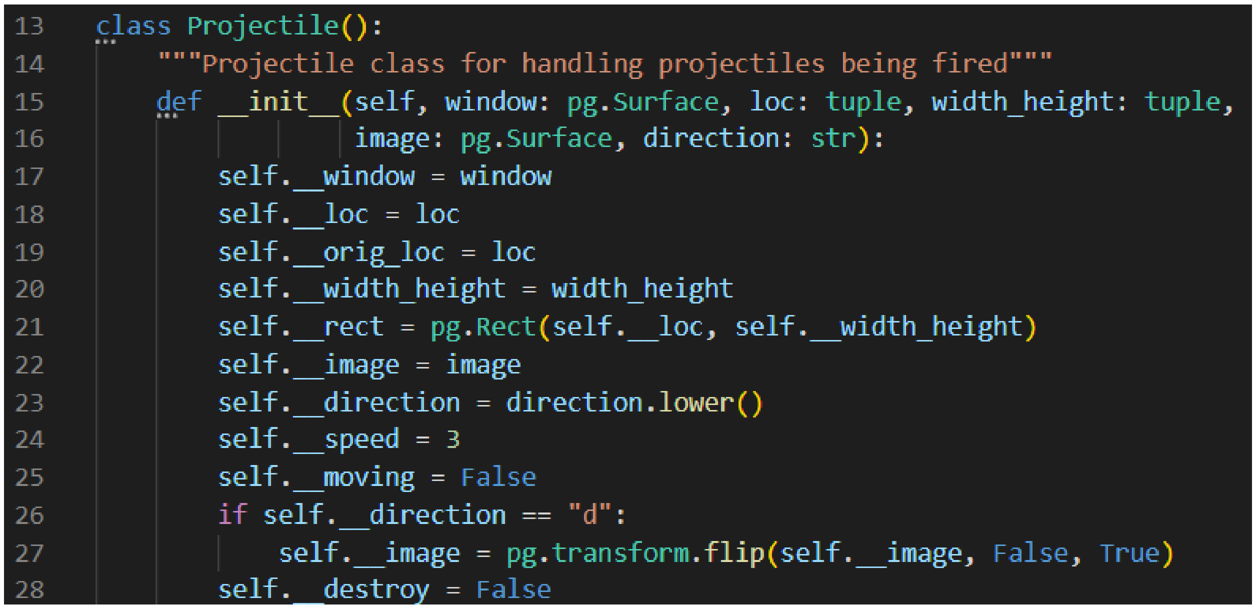


Defeat! The aliens have defeated you!

Defeat text appearing after the aliens destroyed the player's ship.

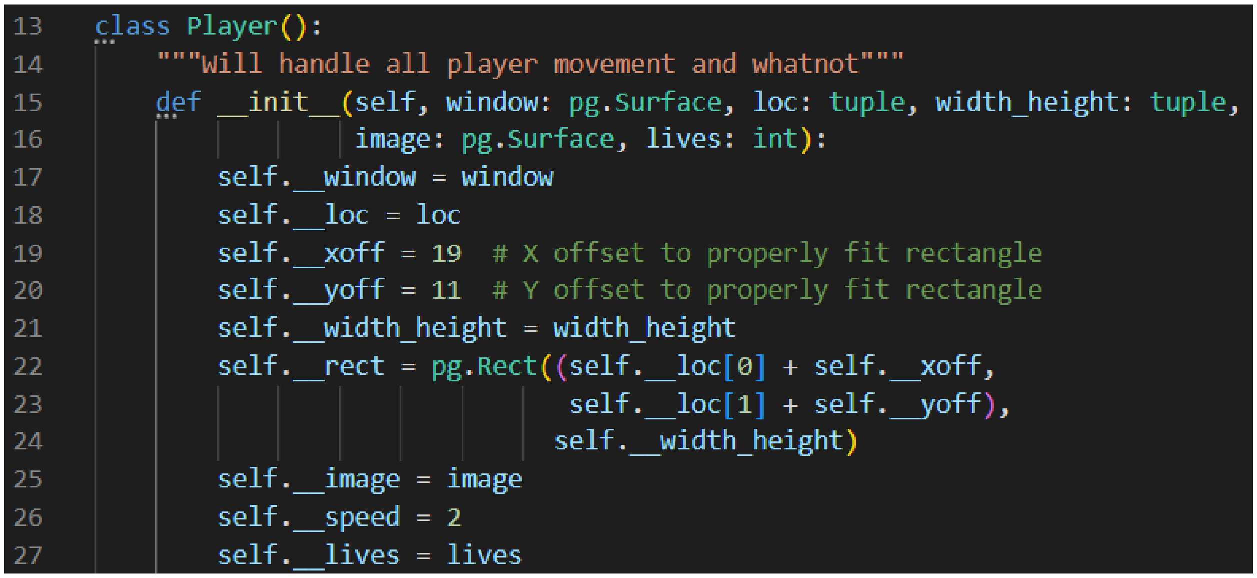
I have also included a restart button and quit button in case the player wants to close the game or restart after they lost.

As for complexity, I had several uses of Encapsulation in the classes. In projectiles.py and player.py, for the classes I used Python's name mangling to try and privatize the variables. Here is a picture of their code:



```python
class Projectile():
    """Projectile class for handling projectiles being fired"""
    def __init__(self, window: pg.Surface, loc: tuple, width_height: tuple,
                 image: pg.Surface, direction: str):
        self.__window = window
        self.__loc = loc
        self.__orig_loc = loc
        self.__width_height = width_height
        self.__rect = pg.Rect(self.__loc, self.__width_height)
        self.__image = image
        self.__direction = direction.lower()
        self.__speed = 3
        self.__moving = False
        if self.__direction == "d":
            self.__image = pg.transform.flip(self.__image, False, True)
        self.__destroy = False
```

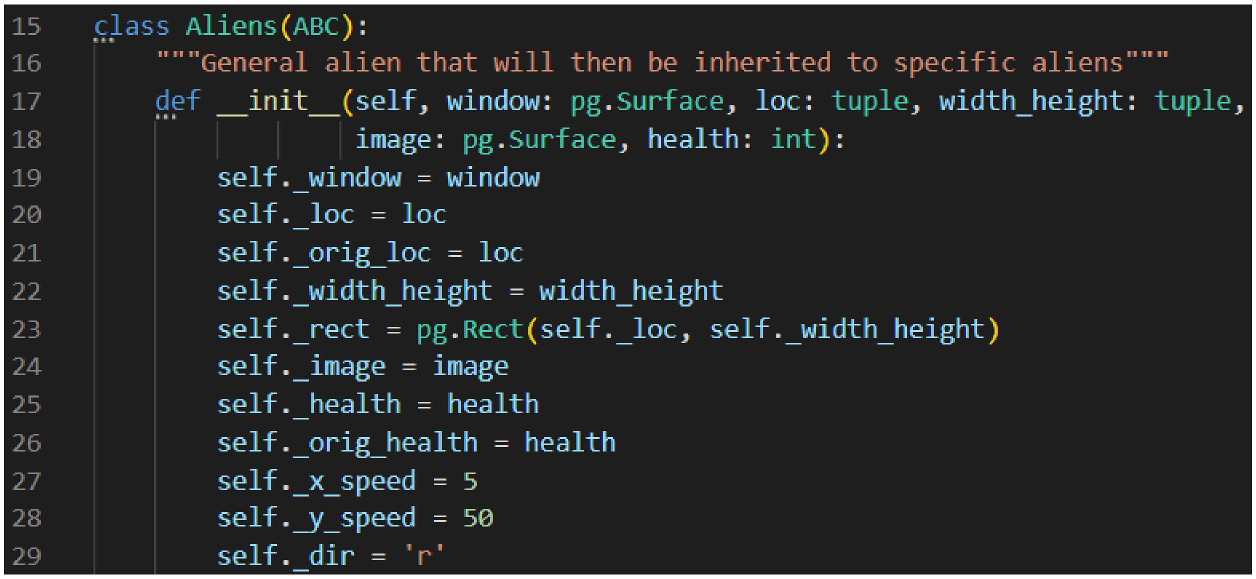
Projectile code with private attributes in projectiles.py.



```python
class Player():
    """Will handle all player movement and whatnot"""
    def __init__(self, window: pg.Surface, loc: tuple, width_height: tuple,
                 image: pg.Surface, lives: int):
        self.__window = window
        self.__loc = loc
        self.__xoff = 19  # X offset to properly fit rectangle
        self.__yoff = 11  # Y offset to properly fit rectangle
        self.__width_height = width_height
        self.__rect = pg.Rect((self.__loc[0] + self.__xoff,
                               self.__loc[1] + self.__yoff),
                              self.__width_height)
        self.__image = image
        self.__speed = 2
        self.__lives = lives
```

Player code with private attributes in player.py.

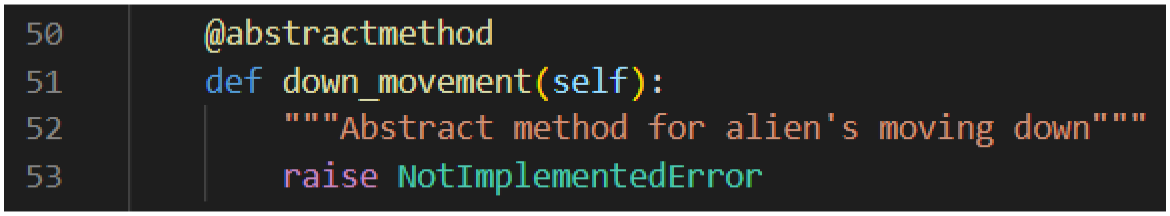
As for the aliens, since I used inheritance from an Abstract Class, I learned that private attributes don’t mesh well with inheritance and used protected attributes. So they have an underscore before their name instead of two underscores. Here is a picture of that code:



```python
class Aliens(ABC):
    """General alien that will then be inherited to specific aliens"""
    def __init__(self, window: pg.Surface, loc: tuple, width_height: tuple,
                 image: pg.Surface, health: int):
        self._window = window
        self._loc = loc
        self._orig_loc = loc
        self._width_height = width_height
        self._rect = pg.Rect(self._loc, self._width_height)
        self._image = image
        self._health = health
        self._orig_health = health
        self._x_speed = 5
        self._y_speed = 50
        self._dir = 'r'
```

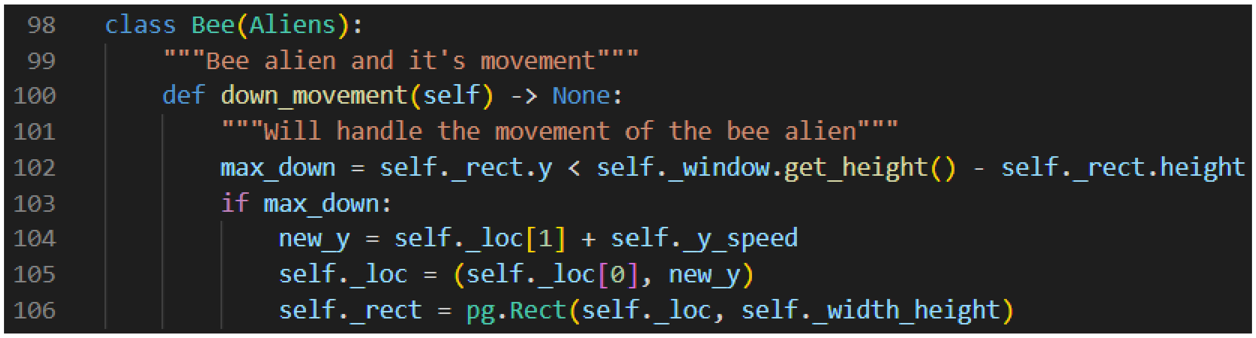
Aliens abstract class with protected attributes in aliens.py.

I also used an abstract method with Aliens.down_movement() because I originally planned on having more complex movement for each alien, but was unable to implement it. So it currently has similar methods in the down_movement() method for each alien subclass. Here is a picture of the code:



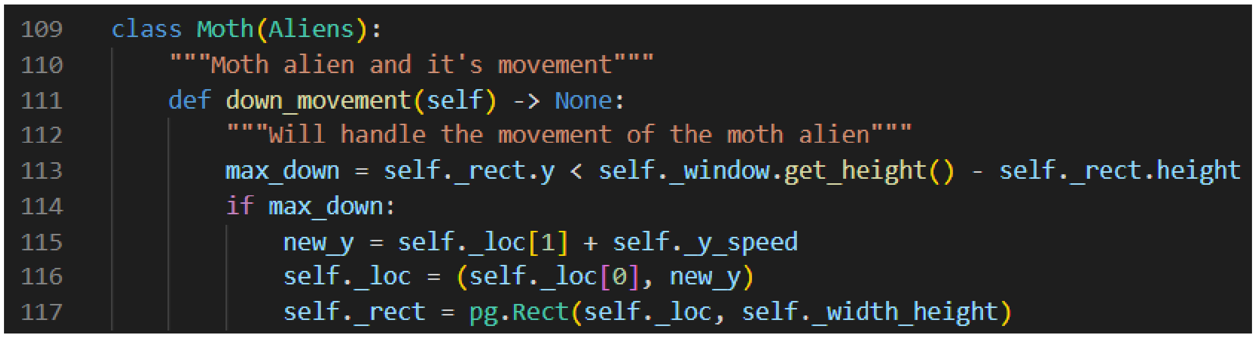
```python
    @abstractmethod
    def down_movement(self):
        """Abstract method for alien's moving down"""
        raise NotImplementedError
```

Picture of abstract method decorator for an abstract method in aliens.py.



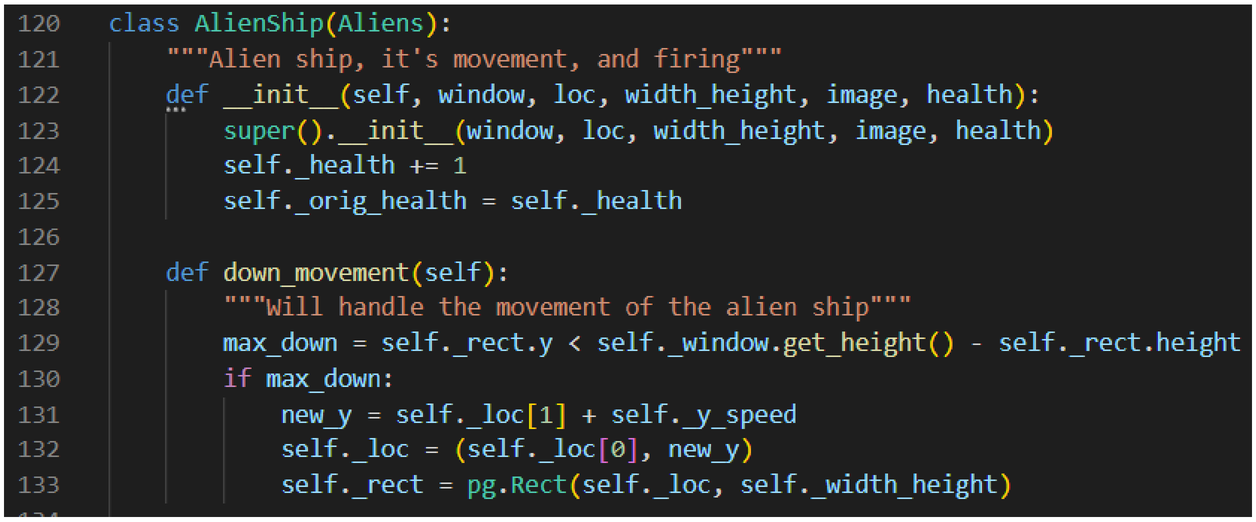
```python
class Bee(Aliens):
    """Bee alien and it's movement"""
    def down_movement(self) -> None:
        """Will handle the movement of the bee alien"""
        max_down = self._rect.y < self._window.get_height() - self._rect.height
        if max_down:
            new_y = self._loc[1] + self._y_speed
            self._loc = (self._loc[0], new_y)
            self._rect = pg.Rect(self._loc, self._width_height)
```

Bee subclass with down_movement implemented in aliens.py.



```python
class Moth(Aliens):
    """Moth alien and it's movement"""
    def down_movement(self) -> None:
        """Will handle the movement of the moth alien"""
        max_down = self._rect.y < self._window.get_height() - self._rect.height
        if max_down:
            new_y = self._loc[1] + self._y_speed
            self._loc = (self._loc[0], new_y)
            self._rect = pg.Rect(self._loc, self._width_height)
```

Moth subclass with down_movement implemented in aliens.py.

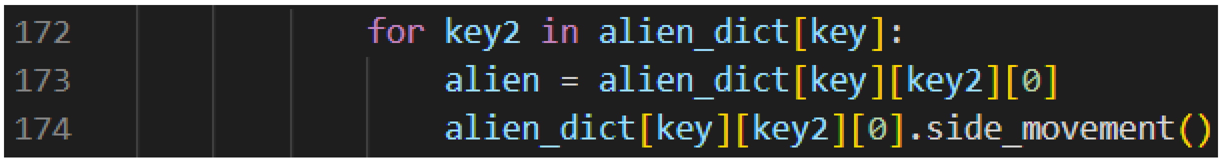


```python
class AlienShip(Aliens):
    """Alien ship, it's movement, and firing"""
    def __init__(self, window, loc, width_height, image, health):
        super().__init__(window, loc, width_height, image, health)
        self._health += 1
        self._orig_health = self._health

    def down_movement(self):
        """Will handle the movement of the alien ship"""
        max_down = self._rect.y < self._window.get_height() - self._rect.height
        if max_down:
            new_y = self._loc[1] + self._y_speed
            self._loc = (self._loc[0], new_y)
            self._rect = pg.Rect(self._loc, self._width_height)
```

AlienShip subclass with overridden init and down_movement implemented in aliens.py.

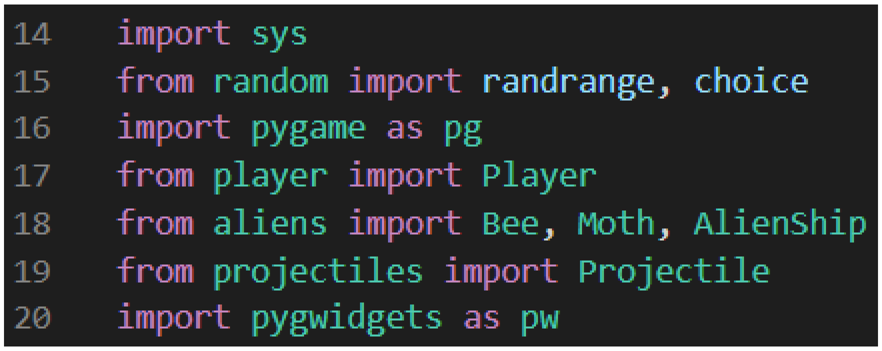
I also used polymorphism here by having all subclasses of Alien have their own down_movement method that were all named down_movement. I did use this in a nested dictionary in order to move the aliens around the screen while the game is running. Here is a picture of that code:



```python
                for key2 in alien_dict[key]:
                    alien = alien_dict[key][key2][0]
                    alien_dict[key][key2][0].side_movement()
```

Encapsulation being used to get the bugs to move to the side in main.py.

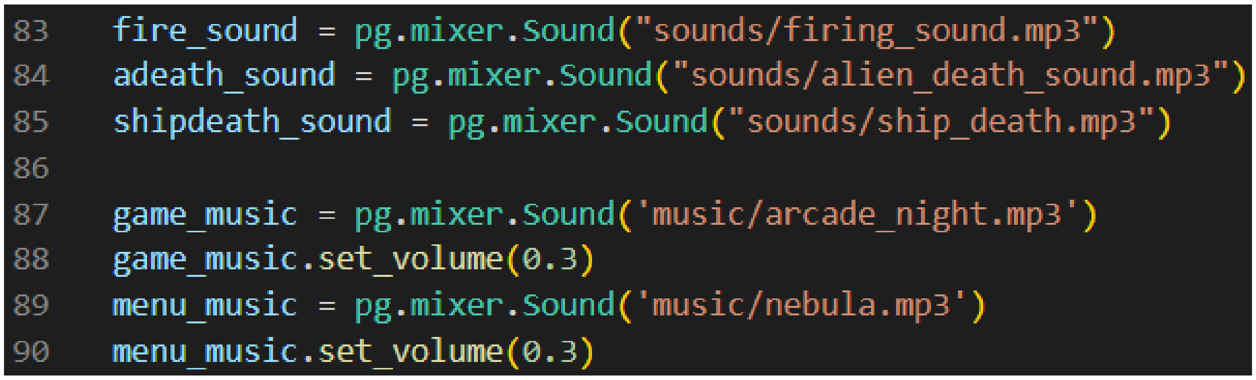
I also used pygwidgets and the random module for the creation of this game and its UI. I also had the class made in different files and imported them into the main.py file. Here are those imports in the code:



```python
import sys
from random import randrange, choice
import pygame as pg
from player import Player
from aliens import Bee, Moth, AlienShip
from projectiles import Projectile
import pygwidgets as pw
```

Imports that were used to make the game in main.py.

I also used multiple sounds, many of which were actually from Galaga. The music I found were actually songs I found online that I thought would work well as game and menu music. Here is that in the code:

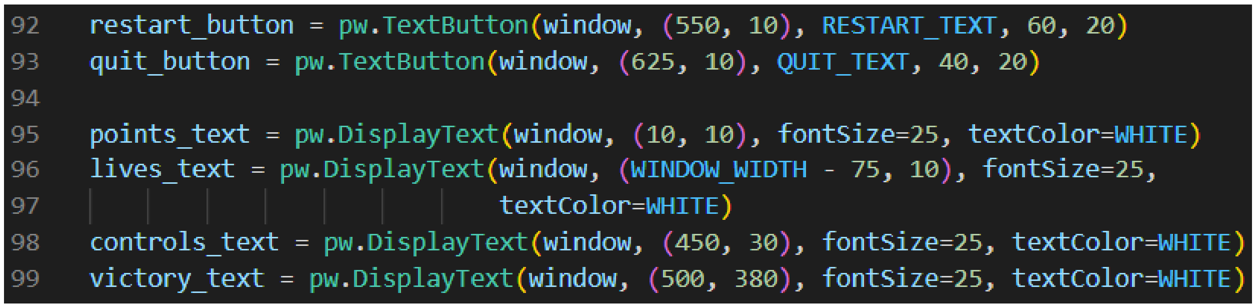


```python
fire_sound = pg.mixer.Sound("sounds/firing_sound.mp3")
adeath_sound = pg.mixer.Sound("sounds/alien_death_sound.mp3")
shipdeath_sound = pg.mixer.Sound("sounds/ship_death.mp3")

game_music = pg.mixer.Sound('music/arcade_night.mp3')
game_music.set_volume(0.3)
menu_music = pg.mixer.Sound('music/nebula.mp3')
menu_music.set_volume(0.3)
```

Sounds and music used for the game in main.py.

I also used pygwidgets to create a restart and quit button along with 3 different texts for points, lives, and a victory/defeat message. Here are those in the code:

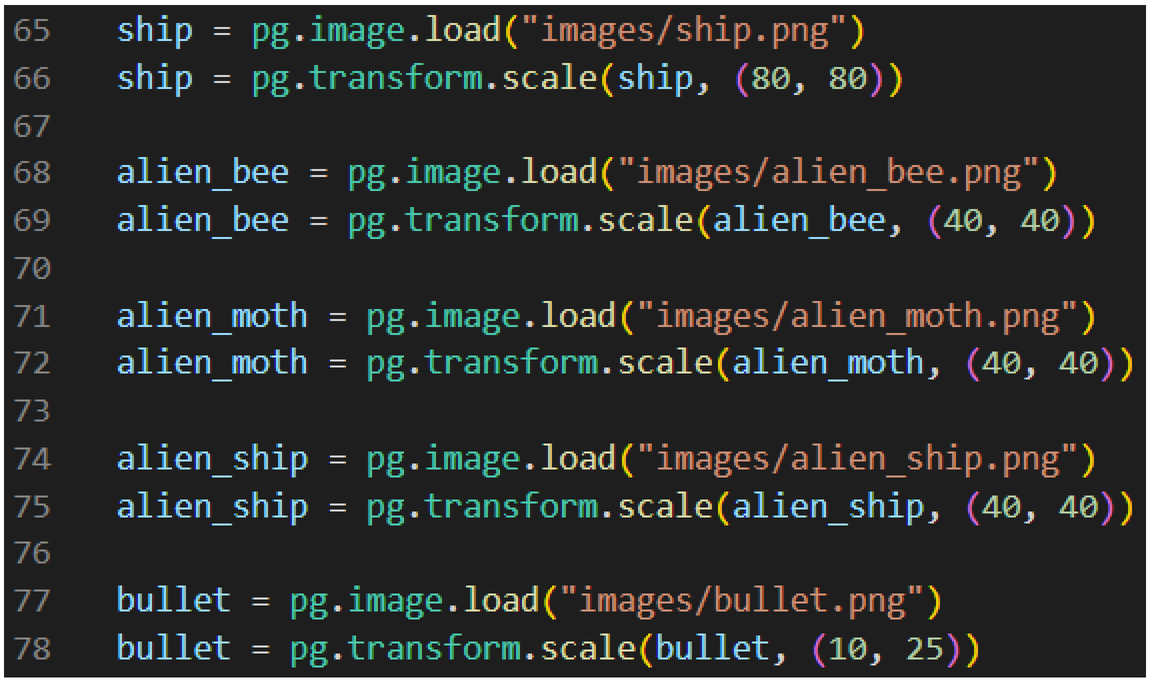


```python
restart_button = pw.TextButton(window, (550, 10), RESTART_TEXT, 60, 20)
quit_button = pw.TextButton(window, (625, 10), QUIT_TEXT, 40, 20)

points_text = pw.DisplayText(window, (10, 10), fontSize=25, textColor=WHITE)
lives_text = pw.DisplayText(window, (WINDOW_WIDTH - 75, 10), fontSize=25,
                            textColor=WHITE)
controls_text = pw.DisplayText(window, (450, 30), fontSize=25, textColor=WHITE)
victory_text = pw.DisplayText(window, (500, 380), fontSize=25, textColor=WHITE)
```

The TextButton and DisplayText classes I used from pygwidgets for text in main.py.

I used several images I found online, here are those in the code:


```
ship = pg.image.load("images/ship.png")
ship = pg.transform.scale(ship, (80, 80))

alien_bee = pg.image.load("images/alien_bee.png")
alien_bee = pg.transform.scale(alien_bee, (40, 40))

alien_moth = pg.image.load("images/alien_moth.png")
alien_moth = pg.transform.scale(alien_moth, (40, 40))

alien_ship = pg.image.load("images/alien_ship.png")
alien_ship = pg.transform.scale(alien_ship, (40, 40))

bullet = pg.image.load("images/bullet.png")
bullet = pg.transform.scale(bullet, (10, 25))
```

The images I loaded and scaled for the game in main.py.

I didn't really manage to get a background to work because I couldn't find one that would work with the screen. I wanted to have an animated background, but ran out of time before I could attempt to do so.

I would like to note some bugs the game has that I currently don't know how to deal with. First there is a small shooting bug where since the game is tied to the frame rate, pressing the fire button 1 time will most likely shoot multiple projectiles. I'm not entirely sure how to prevent this from happening, but I figured it wasn't a big enough issue to deal with it. Secondly, when the Restart button is pushed, sometimes certain aliens don't get properly reset and will instead move in the direction they were last moving which causes them to end up moving opposite of the group. I think this is because going through the nested dictionary takes more time than it takes for the game to go through the loop again. I'm not entirely sure how to fix this one outside of maybe handling the aliens differently instead of a nested dictionary. Finally, sometimes the projectiles will travel halfway up the screen and then vanish, I'm not sure why this is occurring. I think it is either there are some phantom hitboxes that they are hitting or there is something wrong with some of my math being used to move the projectiles. Those are the main two bugs.