**Alien Invasion Game**

Timothy Ferguson  
Zhijin Jiang  
Brian Nordemo  
Matthew Yarab  
Adam Zuckerman

Submitted August 18, 2023

1. Abstract

This report includes information about Group B’s Alien Invasion game created for Professor Nicholas Houston’s CYBR 110 course at Norwich University. For this project, we followed the guide in Chapters 12-14 in the book Python Crash Course, 2D edition, by Eric Matthes. This was the culmination of the 8-week curriculum of learning the Python programming language.

2. Requirements/Specifications

In order to run our program, the user will need to have Pygame on their system. The various .py files we created must be located in the same directory/folder for the program to run correctly. See the “How to setup the program” attachment.

3. Design and Architecture

**alien\_invasion.py**

This is the main file to run the program. All other files are imported to this and part of the game function. Within this file, we begin by importing pygame, sys, sleep function from time, and then all of our functions from the various files. This includes Settings, GameStats, Scoreboard, Button, Ship, Bullet, and Alien functions that will be described later in this report. The first thing our alien\_invasion file does is that it displays the game window in full screen with the game title at the top. We began with a smaller window and eventually made the game display in full screen. After the window and background, the settings, scoreboard, ship, bullets, aliens, and finally, the Play button is loaded that needs to be pressed to start a game. During the initial phases, the game would immediately begin upon launching from the player’s terminal. However, the settings were eventually changed to have the game load in an inactive state. Once the Play button is pressed, the main game loop begins continuously loading player inputs, player ship position, ship bullets, and enemy aliens to include their movement patterns and detects collisions with the aliens, bullets, and player ship. Overall, the gameplay is the player's ship can only have three bullets on screen at a time, and the player has three lives/ships to use; once they are all destroyed, the game ends. A player loses a ship if an alien drops onto the ship or if an alien reaches the ground. Scoring and game level are recorded throughout the gameplay, and when a new game is started, the high score is checked, and all settings are reset for the next game.

**alien.py**

This file is for how the alien will look and the creation of the fleet. This file also includes the movement pattern of the aliens. The file begins by importing pygame as well as a function called Sprite. This also calls on the ai\_game so that it can utilize the screen and settings. We utilized the recommended alien image from the book and saved it in an images folder, and the file was named alien.bmp. The image type, .bmp, is what pygame is able to process and utilize for our use case in this game. We also set the images rectangle attributes. Within the game, all the images and bullets have a rectangle around the outside of them to have the game be able to detect collisions. This will be explained more in a later file. Now once the alien image is loaded onto the screen and has its perimeter defined, we utilize the check\_edges function to check if the alien is at the edge of the player screen. We also use the update function to move the aliens across the screen and update their position based on the settings of alien speed.

**bullet.py**

This file determines the size and behavior of the bullets that will fire from the ship. For the player bullets, we needed to import pygame as well as the Sprite capabilities that we loaded into the alien.py file. Once we are able to load the alien invasion screen and settings, we define the bullet rectangle, again to help with collision detection and have them begin at the middle top of the player ship. This gives the look that the bullets are coming out of the player ship. We then update the bullet based on the bullet speed setting that will be discussed in a later file to travel from the player ship upward toward the aliens.

**button.py**

This file is the start button to begin the game, and upon running out of lives will start a new game. This begins by importing pygame.font to utilize text on our button. It defines the button color as well as the text color and size. It then draws the blank button in the center of the screen with the defined colors and text on it. A rectangle is also defined around the button to be able to react to a player clicking the button.

**game\_stats.py**

This file defines the scoring and level as well as reset the stats upon starting a new game. This file, although very short, allows us to start the game in an inactive state while ensuring the high score is not reset with each game. As mentioned, it also resets the stats when a new game begins by setting the player score to 0, the level to 1, and the player lives back to whatever the settings defined.

**scoreboard.py**

This file starts with importing the Ship class from ship.py to create images of the ship to represent the players' lives. The next step is running the functions which prepare the player's current score, high score, level, and ship lives and displays them as their initial values. The last remaining functions serve to draw the score when called and to check if there is a new high score and update it if there is.

**settings.py**

This file mainly serves to hold the values that determine the size and color of the screen, ship, bullets, and aliens. It also has the speed values for the ship, bullets, and aliens. It contains two functions, one of which is called while the game is ongoing to change values such as the direction the alien fleet is moving. The other serves to increase the speed and point values of the aliens.

**ship.py**

This file manages the player's ship image and position. It will start with drawing the ship in the center of the screen and storing the values of the rectangle representing the ship’s location. The file has three functions, one of which updates the ship’s position based on the player’s input, one draws the ship at its current location, and the last one moves the ship back to its starting position.

**images-🡪alien.bmp ; ship.bmp**

In the images folder, we utilized two images ship.bmp for our player ship and alien.bmp for our alien fleet that we were supposed to shoot.

**\_\_pycache\_\_**

This folder is created by the Python interpreter when it imports a module. It contains the compiled bytecode of the module, which can be used to speed up subsequent imports of the same module.

4. Testing / Quality Assurance

Here we talked about and tested adjusting the speeds and sizes of bullets, and we tried to get a challenging but fun user experience. While testing, during various steps of the build, we changed the bullet width to essentially take up the whole screen to quickly eliminate the aliens to ensure the fleet would continue to spawn. This has given us some ideas for future implementation that will be mentioned in section 7, Future work, at the end of this report.

5. Project Management

Aug 2: Adam Zuckerman sends initial email to all group B members on Norwich.edu accounts

Aug 3: Initial meeting planned for Friday, Aug 4 via discord. Adam Zucerman, Matt Yarab, and Tim Ferguson are only members that replied able to meet. No comms with Brian Nordemo or Zhijin Jiang

Aug 4: Adam, Tim, and Matt talk through wavetops of project, Adam created github site to share files for group work <https://github.com/AMZ58/cyber101_final>

Aug 5 – 10: Collaboration and testing of code compiled and complete, still unable to reach other members of group B

Aug 13-18: Software report generated

Aug 18: Final project submitted

6. Conclusion

We have successfully created the alien invasion game as directed. We have learned, viewed, and experimented with game settings and have a better understanding of how to utilize classes and functions to write concise and efficient code in a complex program. The ability to import functions from separate files to keep the main game loop short was very well displayed. By the end of the project, it made sense why you would want to import functions so that in a separate file you can adjust things (like alien speed, player speed, or the various other settings) and not have to redefine functions in each file. This helped with testing and saved a lot of time from typing. With a better understanding of game development, or any other real-world tasks, we believe we can find use cases to implement our knowledge of the python programming language to improve efficiency and meet assignments/goals/etc.

7. Future Work

Some things that have already been discussed for future work on this project is exploring ways to change the game background color as the level changes. We have also discussed finding and implementing different-looking aliens (size, color, shape) and adjusting scoring based on different attributes. Additionally, we would like to explore a way to have the aliens shoot bullets down at the player at higher levels to increase the difficulty of the game. One final thought is to have power-ups incorporated into the player where maybe the bullets do not disappear after hitting one alien but act more like a laser and continue through destroying all aliens in the bullet path. Or perhaps a power-up that allows an unlimited number of bullets on screen at a time for a certain time and maybe a final power-up that changes the width of the bullets to essentially wipe out a whole row of aliens.