CMPS 335 Technical Report

Project: Alien Shoot

Jeff Cadena, Pedro Hernandez,
Lakhdeep Singh, Sabrina Smith
Concept

Alien Shoot is a first-person arcade shooter, similar to shooting gallery styled carnival games. The goal of Alien Shoot is to stop the impending alien take over, by shooting as many aliens as you can. The game is goes on for an unlimited time, as long as the player does not shoot other humans. If the player makes the mistake of three friendly casualties, the game is over.

Implementation

To begin the implementation of our project, we started with the Rainforest framework. The Rainforest framework provided some necessary components we needed to create Alien Shoot. It set up an OpenGL window and allowed for the usage of transparent images. We also used the Bigfoot structure as the framework for the objects on screen. We also used components from the Asteroids framework for inspiration on the shooting mechanisms of our game.
Methodology

As we began implementing our game, our team didn’t explicitly follow any one software engineering methodology. We started with the conception of our game and listed the different requirements we would need to complete. Then we constructed the design of our game and assigned each member a couple specific tasks to focus on implementing. Following our assignments, we began working on our specific tasks, testing and debugging along the way. Although we didn’t pick a specific methodology to adhere to, our project loosely followed the Waterfall methodology. Unfortunately, we didn’t produce as much work as we had initially hoped and not sticking to a certain methodology may have helped cause this issue. Our project may have been of better quality if we had made a conscious decision to stick by a certain methodology.

Menus - Sabrina

Our game has a main menu, pause menu, and game over screen in implementation. Each of these graphics were created on GIMP. When the game is started, the player is greeted with the main menu screen and instructed to press spacebar to enter into the game. Once the players presses space, they can enter the world of Alien Shoot. If the player needs a timeout, they can press P to pause the game and come back whenever they are ready. When these screens were first put into the game, they were not functioning properly. The menus showed, but the game was still running in the background. The issue was quickly resolved with a few simple if statements. Now Alien Shoot can be enjoyed as a game with start and pause menus
should be. The game over screen just lets the player know that they lost all of their lives
and must exit the game and restart.

![Game Over Screen](image1)

**HUD - Sabrina**

A minimal heads up display was created to assist a new player with the controls
of the game. The HUD shows the user how to move the crosshairs of the gun to aim
and how to fire the gun. It also lets the player know how to pause the game. The HUD shows the player's current score and how many lives they have left as well. The HUD is actually another image layered transparent on top of the screen, with informative text located toward the bottom. Having a HUD with controls on screen as the game starts makes it easier for the player to learn how to play, since they don’t have to pause the game or toggle a help screen to learn controls.

Images - Sabrina

Each image used in our game was found through Google's image search. Every image is licensed for reuse, as we didn’t want to take someone else’s work even though our game isn’t going public. Every photo had to be edited to fit our specific needs. There is a background image, as well as an image for the different tiered rows. The alien and human are their own images. The curtains and heads up display are images as well. The only components in our game that isn’t an actual image are the weapon, crosshairs, and bullet. Like many other groups, we had an issue with using cartoon style images and transparency. Since cartoon style images are typically outlined in black and
our alpha channel is black, we had to shop all black in our images to be a slightly less black. This was an easy and quick fix since we were photoshopping all our of images in the first place, but it wasn't until we had many images in game that we realized this fix needed to be made.

Aliens and Humans - Sabrina, Lakhdeep, Pedro

Both the aliens and humans in game were made from linked lists. The inspiration for the enemies and friendlies was a mix of Bigfoot and the raindrops in the rainforest framework. Each row is treated as a separate linked list. This means there are three lists for aliens and three for humans, six in total. Working with linked lists caused many memory leak issues, as expected. A lot of thinking and tweaking went into trying to perfect the lists. As the game runs now there should be no memory issues, but we did not get to test every scenario. It is possible that memory leaks still occur with these lists in our game. Each row of aliens and humans has a create function, as well as a delete function. There is a draw function for each list, so we can display the characters on screen, and there are check functions used for physics. A move function was created
In the create functions, a total count of all objects is kept track of to be used in the physics of our game.

**Physics - Sabrina, Jeff**

Based on the total count of each object, more objects of that type are created.

```java
if (alienCount < 15) {
    alienCount = createAliens1();
    alienCount = createAliens2();
    alienCount = createAliens3();
}
if (humanCount < 5) {
    humanCount = createHumans1();
    humanCount = createHumans2();
    humanCount = createHumans3();
}
```

This allows us to keep a steady amount of aliens and humans on the screen at once. The max limit for aliens is different than humans, because we want it to be harder for the player to kill humans and lose their lives. A future implementation of the game could put more humans on the screen as time goes on, making the game more challenging for the player. After the aliens and humans are created, the weapon functionality is checked. The bounds are checked on the bullet and the bullet is moved appropriately. Then a check is made on the aliens and the humans. The check function calls the move function for each row and the move functions check for collision from a bullet or if it has fallen off screen. Each object that is deleted must be kept track of. This was a problem originally creating segmentation faults, because many different objects would fall off the screen at once, and the program could not keep track of them all. The check function also updates the score and lives for the player. After each row is checked, the amount of enemies and friendlies on screen is calculated and more are spawned if need be.
Weapon - Jeff

The rear view of a weapon is displayed since Alien Shoot is a first-person shooter. The type of weapon displayed is a Glock pistol due to its widespread acceptance among law enforcement and military personnel alike, including elite special operations units. A rear view of a Glock model 17 pistol is displayed below in Figure 1. This pistol is equipped with the factory sights.

![Figure 1: Rear view of Glock pistol](image)

As such, the Alien Shoot pistol features the signature white U-shaped factory sights. The rear serrations seen in Figure 1 are also drawn on the Alien Shoot pistol. When the weapon is fired with the ‘f’ key, the special effect of muzzle flash is created as a bullet is discharged from the pistol. Once discharged, the bullet’s position begins at the end of the barrel of the pistol and travels to the red target crosshairs via a velocity vector. This vector is determined from the position coordinates of the target and the position coordinates of the bullet. If the bullet strikes an alien, the alien is deleted and
the game score is incremented. Otherwise, the bullet is deleted before it goes out of the bounds of the game.

The weapon, crosshairs, and bullets were all drawn using OpenGL. In addition, the creation of the weapon, crosshairs, and bullets was accomplished with Object-Oriented Programming so that the code would be modular and easy for our group of developers to trace and modify. Encapsulation was implemented to limit access to private data members. Furthermore, inheritance was used sparingly when creating the weapon.

**Target Crosshairs - Jeff**

The target crosshairs are displayed in red to provide a visible contrast to the green landscape, which makes it easier to track the target. Aiming is performed by moving the target crosshairs with the arrow keys to the desired position. To increase the difficulty of the game and to conform to real-life physics, the bullet will not land directly in the middle of the crosshairs every time. The motivation for this is that any shooter has to contend with trembling hands due to breathing and heart beat. Shooters also have to factor in gravity and air resistance when estimating a bullet’s trajectory. In addition, this deviation from the crosshairs discourages devious users from leaving the target in one spot and mashing the fire key.

**Sound - Pedro**

During the conception of our game, we intended on implementing some sound effects to make the game more appealing for the player. We wanted to have background music, sound effects for the aliens and humans, noises for the weapon
firing, etc. Unfortunately, this is one of the tasks we did not accomplish by the end of the quarter. Due to a late start and the complexity of OpenAl, we could not complete this task. If we continue to try to improve Alien Shoot in the future, this will be a major component we implement.

Conclusion

Although our group met up regularly, we didn’t accomplish as much as we had hoped for. This quarter taught us that it is tough working as a group on a project where all the responsibility falls on our shoulders to stay motivated. If we had explicitly selected a certain methodology to stick to, our project may have turned out differently. Also, our testing was limited due to a lagged start on the core functionalities of our game. Being said, we are proud of the work we produced during this Spring quarter and a lesson was learned by each of us. We hope our future software projects are even better.