Space Defense:
Creating a 2D Game with 3D assets using Unity Game Engine

Jeevan Vase

Franz Kurfess, Faculty Advisor
California Polytechnic State University
San Luis Obispo, CA USA

May 20, 2019
## Contents

1 Abstract 3

2 Introduction 4

3 Background 4

4 Technology 4
   4.1 Unity Game Engine 4
   4.2 Version Control 4

5 Design 5
   5.1 Story 5
   5.2 Models/Meshes 5
      5.2.1 Enemy Destroyer 6
   5.3 Game Design 8
   5.4 Combat 10
   5.5 Sound 11
   5.6 Controls 11

6 Implementation 12
   6.1 Unity Visual Studio 12
   6.2 Organization 12

7 Analysis 14
   7.1 Play-Testing 14
      7.1.1 First Round 14
      7.1.2 Second Round 14

8 Future Work 14
   8.1 More Enemies 14
   8.2 More Abilities 15
   8.3 More Levels 15
   8.4 Cinematics 15

9 Conclusion 16

10 Citations 17
1 Abstract

The goal of this senior project was to learn and use the skills necessary for one person to make a complete game. The storyline of the game involves a young pilot who must protect his planet from both the enemy ships and asteroids speeding towards it. The 3D assets that were used were downloaded from the Unity Store. I used Unity Game Engine to create the logic for the game. This document explains the technologies that I used, design choices I made, feedback from player-testing, and work that I want to complete for the project in the future. The final game demo features levels, a start screen, and a combat system that allows players to gain additional ships to help defend their planet.
2 Introduction

Space Defense is an action packed 2D space-shooter game that focuses on the defense of a planet by a young pilot. Players defend worlds through a third-person perspective, by dodging plasma cannon shots and defeating all different kinds of enemies.

3 Background

I wanted to see how the general skills I learned in my undergraduate careers at Cal Poly could be applied to the challenge of learning how to make a game, and implementing what I learned in just one quarter. To showcase this, I wanted to design and implement an engaging and graphically stimulating video game. I definitely have the computer science background covered, but had to venture out of my major-specific talents to try and develop a more well-rounded and visually entertaining game, given the multiple roles typically required in the game development industry.

4 Technology

4.1 Unity Game Engine

To achieve anywhere near the level of gameplay I wanted with the amount of time I had, I needed to work in a game engine so that I could use some the frameworks already in place and focus more on the gameplay itself. I chose Unity after weighing the pros and cons between Unreal and Unity3D. After reading several comparative articles I found that Unity was definitively the more developed game engine, with more cross-platform support, not to mention superior documentation and tutorials. Also knowing that it had a multitude of free assets with which I could build my game cemented Unity as my choice.

4.2 Version Control

Given that I didn’t have to collaborate with anyone else ended up being a nice benefit to working alone. I was able to create the whole game in Unity and saved a copy of the working game as soon as I had the minimum functionality I wanted.
5 Design

5.1 Story

I wanted my game to have a background story, as this can be an essential element to any good game. The storyline of the game involves a young pilot who must protect the planet from both the enemy ships and asteroids speeding towards it. They have already completed their pilot training, but have never been required to go against such overwhelming odds. The longer they are able to last the more reinforcements are able to show up. As the invasion continues the enemy commander must continually send more ships, as the initial assumption that this planet would be a soft target is continued to be proven wrong.

I would have loved for my story to include cut scenes, but time would not allow me to produce these. A cutscene where my young pilot is simply manning his station, when suddenly he gets an alert! He sees the invasion coming towards him, and is the only one ready to take on the problem! He sounds the alarm and heads out to defend his planet. As the game progresses, it could have been cool for the first time a big enemy was hit for a cutscene to show the two medium sized spaceships spawn from behind it, showing that even as one enemy goes down, there are more to replace them. I would also want a cutscene showing when reinforcements arrive, to emphasize the help that is coming and that they will be flying in tandem.

5.2 Models/Meshes

Since I decided to make a 2D game with 3D assets, I needed to make sure I had good 3D assets to use. I will show you just two of the assets I used to give you an idea of the game style.
5.2.1 Enemy Destroyer

The image above is the final model and texture of the large Enemy Destroyer, rendered in Unity. When this model is hit, two medium sized ships will spawn in its place (not shown here).
5.2.2 Small Enemy Fighter

The image above is the final model and texture of the Small Enemy Fighter rendered in Unity. The lighting is slightly different from the in-game lighting, however the features should be almost equally visible.
5.3 Game Design

One of the most important aspects of a challenging and interesting game is the game design. Core game mechanics such as enemies attacking the player and the exchange of damage were play-tested constantly to ensure a fluid gameplay for essential parts of the game.

My game achieves meaningful play by engaging the player and requiring focus and strategy to progress through it. With no feedback loops, my game relies heavily on player skill. This makes the player value his or her progress in the game. While moving through the game, the player must strategize how they will play the game, whether it be primarily attacking or defending. In an attacking mindset, the player seeks out their enemies and tries to kill them immediately. A defensive player waits to dodges opponents’ projectiles and then counter. The method of playing can impact gameplay in a large way. The choices the player has adds to the meaningful play of the game because it provides the possibility of a unique experience every time the player plays the game. Also, during each of the levels, the enemies and asteroids that are spawned are randomized. This leaves uncertain as to what will be spawning on screen and where. This means that each time the game is played, a new scenario would be created. The difference between games is a unique factor in creating meaningful play.

Descriptive features are another unique factor included in my game. In order to make the game more meaningful, I needed to find a way to make the player feel like they were actually in the game. One of the best ways to engage more of the player’s senses was to add graphics and sounds. The meshes/models used for each of the objects were unique and effectively conveyed to the player what the actual objects looked like in real life. It was important that the types of enemies we used look similar to each other, because that kept the player immersed in the gameplay, since all the ships would be coming from the same fleet. The ability to keep the theme throughout the game was difficult but proved to be worth it because it made my game look complete and organized.

I added animations to my game to increase this aspect of meaningful play. When a plasma bolt hits an enemy or asteroid, the player sees an animation of that enemy or asteroid exploding. This gives the player a sense of satisfaction for striking down foes. I added sounds to the game including background music, a firing sound, and sounds for when the enemy or player is hit. This ensures the player that whenever an action occurs, it is working because he or she can hear it happening.
Another way I created meaningful play was through evaluative features. By adding these features, the player is not only engaged in the game, they are able to measure their progress. One way to measure progress is through score. My game has one main system of calculating score. It is by destroying an enemy or asteroid, which rewards you with a certain number of points. Each enemy is worth a different amount of points so killing certain enemies and dodging others is a strategic decision you must make. In addition to these points, new waves of enemies push players to completely eliminate prior waves as fast as they can, preventing them from hiding from the enemies the whole time. Score can be used comparatively with others to see who is the best. Adding a score adds a competitive level to the game. Another way to measure progress is through the progression of levels. My game has 3 levels currently, with a boss level coming soon, that contain differing challenges and level of difficulty. The player can know how far they made it through the game by comparing which level they are on. This furthers the competition with other players. For example, a player can brag to another player that they have gotten to a further stage, or even beaten the boss.

This differing difficulty of the game increases the meaningful play for the player. My game starts off fairly easy and gets increasingly harder as you move further into the game. This allows the player to become accustomed to the controls and gameplay early. Later on, they can focus on beating levels rather than how to control their spaceship. The levels increasing in difficulty makes the player ‘up their game’ in order to advance. However, I give the player multiple chances to beat the level by adding a restart system. Rather than starting the entire game over after you die, the player can restart at the beginning of the level that they just died on. This lets the player retry a level they are having a difficult time with. However, any wave progress the player has made is forfeited in order to retry a level, adding value to surviving the game. As the player gets used to the game, they are able to pass levels they were not able to previously.

Through all of these features, both descriptive and evaluative, my game achieves meaningful play. The game keeps the player engaged, as well as continually making the player think on their toes. Each of these features add to the overall replay value of the game. The strategy and decision making required creates a different outcome every time and this allows players to compete and compare their results with other players. The recurring challenge of the game is what makes our game replayable with meaningful play every time.
5.4 Combat

The main method of combat is a long range plasma bolt that results in you guessing it, a bolt of plasma hurtling towards your enemies. This attack becomes more effective if you kill enough enemies to be rewarded with an extra ship, as this will essentially double your attack power. There is a slight delay between firing. This was designed in order to prevent the player from simply spamming the attack button and encourage them to dodge enemies’ attacks between firing. The attack’s point of contact is based on a trigger collider and checks to see what kind of enemy it is before executing the game logic.

The images above highlight the combat scenario, as well as showing off some of the particle effects used in the explosions.
5.5 Sound

Because of the accelerated nature of this project, the priority of sound was relegated in relation to the models and the programming of the actual game. The sound effects I was able to implement ended up being shockingly good, and received great feedback on them despite the lack of time spent in the area. I found that adding sound effects significantly improved the both the gameplay and the atmosphere of my game. I used different sounds for the title screen and levels, as well as when the player was firing, and for all explosions.

5.6 Controls

In order to get the game playable as quickly as possible I kept the controls very simple. Future iterations could see them expanded upon, but for now this is what they are. You use either the arrow keys or WASD to move your pilot around the map. The spacebar is used to fire plasma bolts at any enemies. Enemies appear randomly throughout the level, and you must either destroy or avoid them. As you progress through the game, you can get extra reinforcements(life) as a reward for doing well. Your current score is displayed, as well as the current wave.
6 Implementation

6.1 Unity Visual Studio

All game logic was created through Unity’s Visual Studio. This allowed me to easily add functionality to objects in the game by writing scripts for each object, and attaching the scripts to the object.

6.2 Organization

At a high level, each level for Space Defense is centered around the player. As the player moves around in the level, they are looking to destroy enemies, which will add to their score at the end of the level.

Each level in Space Defense contains varying combinations of enemies and asteroids as well, hindering the players progress towards completing their objective. The player can attack enemies either with their plasma blaster to kill them and increase their score.

Game Objects

Player

The player game object contains all of the logic for the player movement and interaction with the world. In addition to the usual rigid body, sprite renderer, and collider, the following 2 scripts are attached to the player to make this happen.

- **PlayerController**: The main script for the player that controls the movement for the player, the keybindings for player actions, and visual aspects for the player.
- **VolumeControl**: A script that manages all the audio aspects for the player

Enemies

All different kinds of enemy game objects have an Enemy script attached to them that contains the enemies health, as well as on hit animations. Each specific enemy has a unique script attached to them that contains the logic for the unique behavior of it. This includes their movement patterns, attack types, and what the enemy can be affected by.
**GameController**

The GameController is an object that persists throughout the game, and keeps track of all variables and instances that need to carry over between levels. The object has a script attached to it that keeps track of all these variable, plus the state of the game, such as the current level, and if the player has lost or won. The GameController has the majority of the logic for the scoreboard and any extra lives that the player earns, as well as when to instantiate different kinds of asteroids and enemies.

**Major Issues**

One of the major issues I ran into was when I was creating multiple types of enemies, and they kept exploding when they ran into each other. After poring exhaustively through the scripts, I found that the problem was not logic related, but had to do with the prefab instances themselves. The prefabs that were exploding themselves had extra collider inside their meshes, deleting these extra colliders got rid of the problem.
7 Analysis

7.1 Play-Testing

7.1.1 First Round
For the first round of play-testing, I had implemented most of the first level. Since it was a basic level, I had a mix of play-testers, including both those who do not play computer games and those who do. This was to see if my first level needed difficulty changes to allow more of either segment to better enjoy the game. I got some good feedback from these playtesters, with both segments agreeing that the fast enemy I had was way too fast, and that there were too many asteroids spawning. Some more advanced players also wanted the option of more types of guns and abilities, something I definitely wanted to add. They also noticed that some of my enemy ships were colliding with themselves and exploding, which could be a feature, but was not the functionality I was looking for in my particular game. They really enjoyed the particle effect explosions I had put in, and found the controls to be both smooth and simple.

7.1.2 Second Round
For the second round of play-testing, both the first and second level were completely playable. I was not able to make all the changes suggested from the first round of playtesting, such as implementing more types of guns and abilities for the player, but was able to fix enemy collisions, ship speed, and asteroid over-spawning. Playtesters really enjoyed this round of testing, but found the second level to be very difficult as the wave number increased. I decided to kill two birds with one stone and incentivized players to go after enemy ships by giving players a bonus ship when they hit certain point totals each level.

8 Future Work
(If I had more time I would have liked to really develop the storyline more to better introduce and immerse the player into the game scenario).

8.1 More Enemies
Given the positive feedback I got just from having the unique enemies I have now, I would like to add even more enemies with individualized behavior to each of the levels. I
think it would really interesting if I could have the enemies gather data on player behavior and use it to target them more effectively, forcing the player to change their behavior or face losing the game. As new types of enemies are added more complex and interesting levels can be designed to integrate the new types of behavior. The new strategies that will have to be developed will be interesting, and potentially lend themselves to being solved by allowing players to have more abilities, increasing the overall state space of the game.

8.2 More Abilities

While players really enjoy the simple mechanics of the game as it stands now, some advanced players wanted to have some additional abilities unlock as the game progresses. I currently give players an extra ship in the game when they hit a certain point total, potentially making this secondary ship controllable by another player could be super fun, in addition to being inclusive and allowing for a greater kinship with fellow players.

8.3 More Levels

Given that scope in the beginning was fairly ambitious with at least three complete levels, I was happy to meet the original proposals, even if I felt that I could have done a lot more within each level. The next level I had planned was the final boss level that could have taken place by the enemies’ planet instead, to end the threat once and for all. I was able to implement the preliminary levels given the time I had.

In the future, I would like to add sub-levels that could be used to pick up extra bonuses, complete side missions, or act as intermediate levels between main levels. I would also like to add an endless level where players could compete to see who can last the longest.

8.4 Cinematics

I would like to add additional animations to both the player and enemy objects, such as barrel rolls or other evasive maneuvers that would be visually appealing. I would also really like to add cut scenes to the game to really help build out the story and more fully immerse the player in the gameplay, particularly at the beginning when the player is just finding out what they got themselves into.
9 Conclusion

Ultimately, I believe I've made a fun and exciting game that has a sufficient degree of challenge to be interesting, as well as a simple enough core mechanic that the majority of people could learn it quickly. There is definitely potential for replayability as players adopt new strategies to make it past increasingly difficult waves of enemies, and attempt to better their high scores. I definitely had to adjust my expectations and scope of the project as the quarter progressed and new challenges arose, however I learned so much about Unity and game development as a whole that I would certainly call this project a success.
10 Citations

*medium.com/@thinkwik/cryengine-vs-unreal-vs-unity-select-the-best-game-engine-eaca64c60e3e*.

*https://unity3d.com/learn/*

Downloaded free assets from the asset store in Unity
*https://assetstore.unity.com/*