COMP222 – Principles of Video Game Designs and Implementation

Assignment 2 – Game AI option 2

Samuel Nuttall

201608203

The following document is my Documentation for assignment 2 of the COMP222 module. Given the 2 options available for the completion of this assignment I chose the latter of the 2, which was to implement Game AI mechanics into my previously implemented Asteroids game, which was the product of assignment 1. Please find: the short description and outline of my chosen AI behaviour model, the design description of my AI agent implementation and the description of my final implementation.

Upon completing this document, it has come to 4000+ words, and I have seen the specification says a 700-1000 word document, I really hope that this being in so much detail is not a problem and does not hinder the documentation grades for this assignment. Thank you!

1. *A short description of the behaviour model of your choice (e.g., FSM, Decision trees, etc.). You only need to write a couple of paragraphs to show your understanding of how the model works.*

Finite State Machine AI Behaviour Model

My chosen AI Behaviour model shall be the Finite State Machine model. This Finite State model represents that the behaviour of an agent can be in one out of a finite number of states at any given time. The agent can 'transition' between these states, and each of these states are associated with a set of actions that can be performed by the agent.

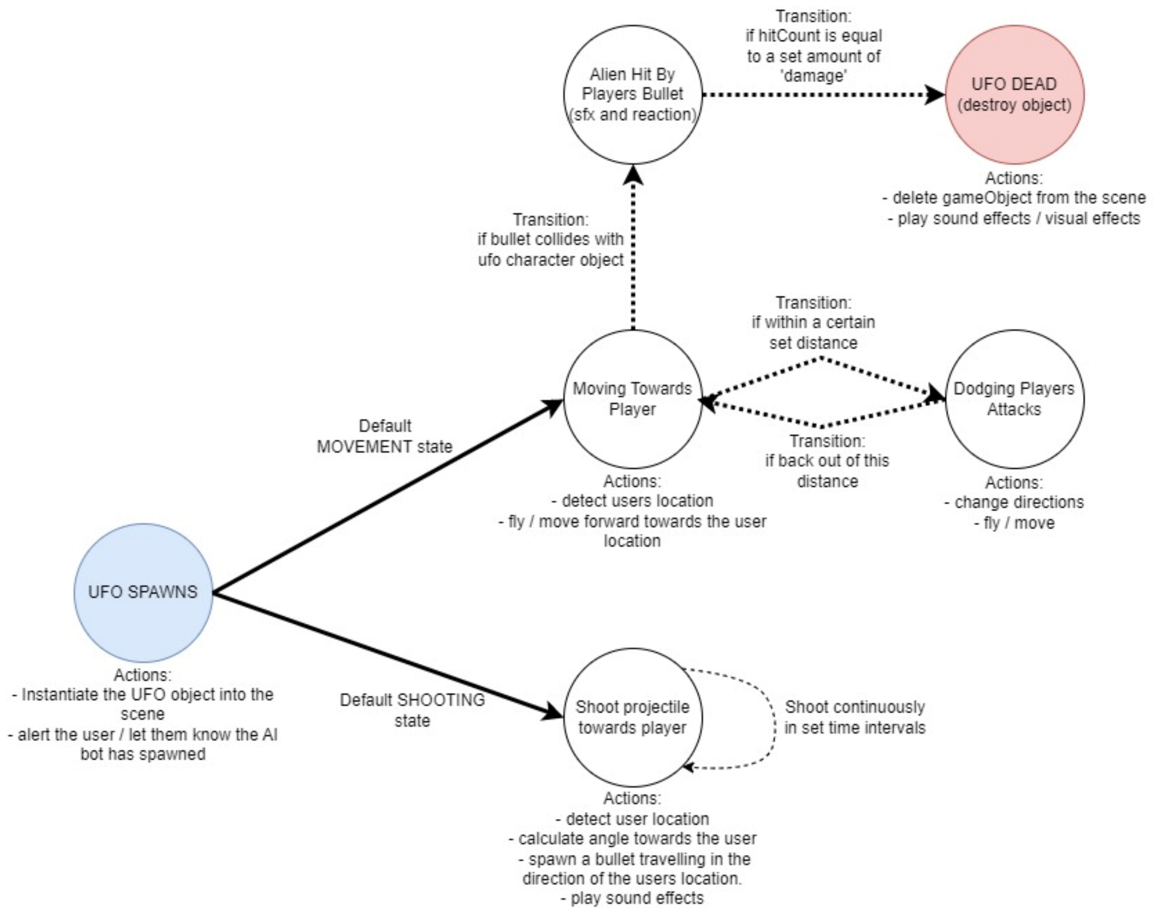
The states can each represent a different type of behaviour that the agent can have; some examples of these 'states' could be a moving state, a stopped or static state, a turning state or a shooting state.
The agent transitions between these states are based off conditions or rules. Some examples of these conditions or rules could be:
- Sensory data (such as the agent 'detecting' that it is close to something)
- Keyboard, mouse or controller input from the user.
- Conditional 'if then' statements in code.

The 'outputs' of the finite state machine are the representations of the agent's behaviour, the outputs can be things like movement, sound or visual effects.

Finite state machines are useful for controlling agent behaviour in structured and predictable ways, meaning that - in the context of a video game - it can be used to create characters and non-playable characters with specific behaviour sets. Ideally, we want a limit on the amount of states and transitions otherwise the FSM will become complex and they may not be able to handle more complex behaviours.

Design Description

UFO Ai Object Design (FSM)


Alien Hit By Players Bullet (sfx and reaction)
Transition: if hitCount is equal to a set amount of 'damage'
UFO DEAD (destroy object)
Actions: - delete gameObject from the scene - play sound effects / visual effects
Transition: if bullet collides with ufo character object
Transition: if within a certain set distance
Moving Towards Player
Dodging Players Attacks
Transition: if back out of this distance
Default MOVEMENT state
Actions: - detect users location - fly / move forward towards the user location
Actions: - change directions - fly / move
UFO SPAWNS
Actions: - Instantiate the UFO object into the scene - alert the user / let them know the AI bot has spawned
Default SHOOTING state
Shoot projectile towards player
Shoot continuously in set time intervals
Actions: - detect user location - calculate angle towards the user - spawn a bullet travelling in the direction of the users location. - play sound effects

Above find the Finite State Machine Graphical Representation / Design of my proposed UFO Ai Character

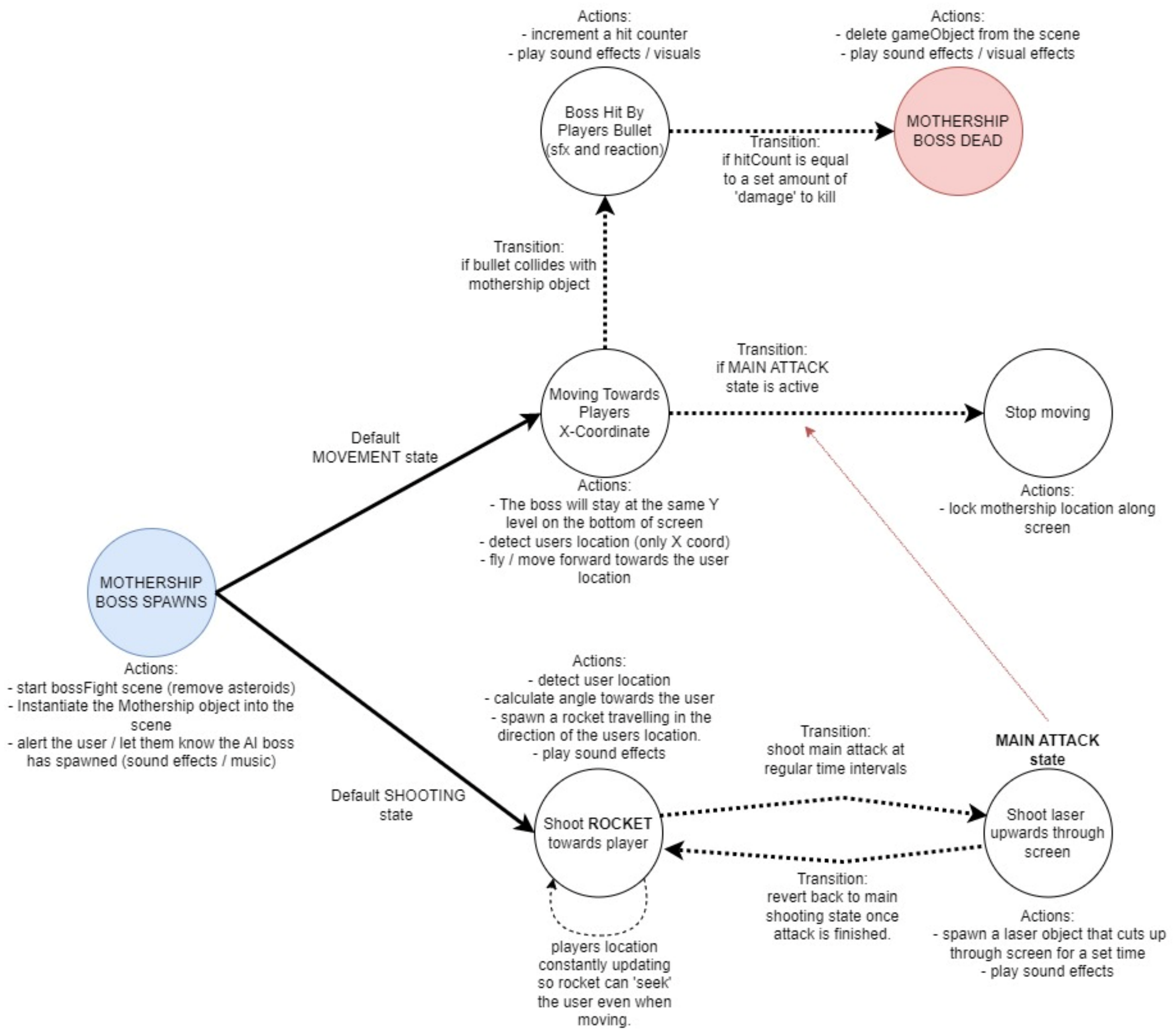
The UFO shall be spawned in at a random point within the screens bounds and should instantly have the default movement state active, thus should find the player object and 'seek' the go to their location, as the player object is moving the 'seeking' or following of the player location must be an updating function so that the players new location at any instant is passed to the alien object. I intend that the shooting mechanics of the alien shall be ran concurrently with the movement, so the state transition may be an instantaneous function call at the time of shooting. I want the shooting to be at regular intervals so it is like the alien is chasing you while shooting at you in predictable time periods, this should make the gameplay against the alien more fun and interactive as you can play against it and dodge its attacks / plan your attack back.

Instead of the UFO indefinitely flying into the user's hitbox, the alien should switch to a 'dodging' state whenever it reaches a close distance to the user, this state should allow the alien to roam freely within some bounds and 'dodge' the players attacks. Whenever the player then moves away and the distance between the alien and player is large enough, the default movement state shall be transitioned to, allowing it to follow the player again.

I think this will make the UFO mechanics more interesting as it sort of allows the alien to catch up to you then instead of the 'chase' it turns to more of a 'duel' between you and the alien, allowing for a different user gameplay experience in these different scenarios which could be more exciting as a gameplay mechanic.

My idea is that I can use the UFO and increment the difficulty of the game via the fights we have with these, as well as incrementing the difficulty of the asteroid mechanics. I intend to add a sort of swarm mode where more of the aliens spawn at any time, given that the aliens have 'alerted the others to come and help' (maybe when too many are killed or they have been following the player for a while). This incrementing difficulty would add more of a challenge as the game progresses which hopefully will mean that the user does not get bored of the playthrough, as this game with these added mechanics could turn into more of a storyline game with the aliens attacking you and such, as opposed to the asteroids just flying at you for the attempt at getting a new top score.

Mothership Ai Object Design (FSM)


Actions:
- increment a hit counter
- play sound effects / visuals
Boss Hit By Players Bullet (sfx and reaction)
Actions:
- delete gameObject from the scene
- play sound effects / visual effects
MOTHERSHIP BOSS DEAD
Transition: if hitCount is equal to a set amount of 'damage' to kill
Transition: if bullet collides with mothership object
Moving Towards Players X-Coordinate
Transition: if MAIN ATTACK state is active
Stop moving
Actions:
- lock mothership location along screen
Default MOVEMENT state
Actions:
- The boss will stay at the same Y level on the bottom of screen
- detect users location (only X coord)
- fly / move forward towards the user location
MOTHERSHIP BOSS SPAWNS
Actions:
- start bossFight scene (remove asteroids)
- Instantiate the Mothership object into the scene
- alert the user / let them know the AI boss has spawned (sound effects / music)
Actions:
- detect user location
- calculate angle towards the user
- spawn a rocket travelling in the direction of the users location.
- play sound effects
Transition: shoot main attack at regular time intervals
MAIN ATTACK state
Default SHOOTING state
Shoot ROCKET towards player
Shoot laser upwards through screen
Transition: revert back to main shooting state once attack is finished.
Actions:
- spawn a laser object that cuts up through screen for a set time
- play sound effects
players location constantly updating so rocket can 'seek' the user even when moving.

Above find the Finite State Machine Graphical Representation / Design of my proposed Mothership Ai Boss

The mothership shall be implemented as a sort of boss fight into the game, which will be started upon reaching a certain stage of gameplay, whether that be by reaching a certain score or killing a certain amount of the UFO alien enemies.

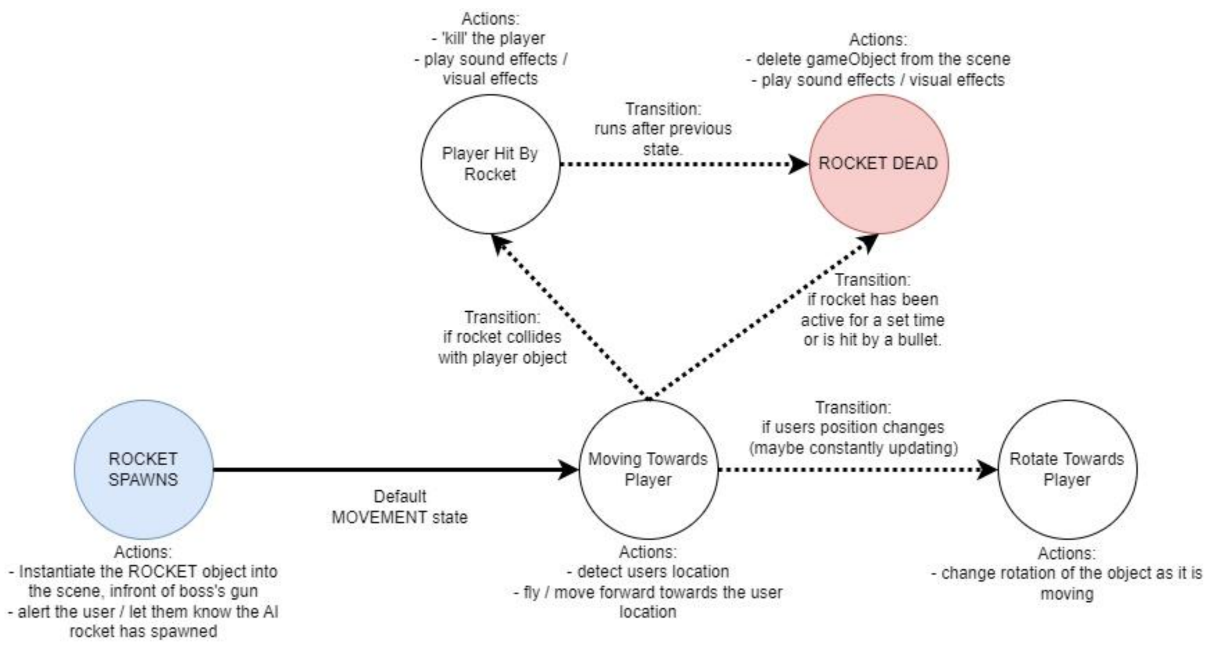
As seen in the diagram, when the mothership boss fight starts, all the asteroids and sprites on the screen will be removed to clear the scene for the boss. The boss shall then spawn, but it should only follow along the bottom of the screen, I will just lock the Y position of the alien but then let it move along the X-axis, constantly moving towards the players X coordinate so it is following them across the screen.

The mothership is intended to have 2 weapon mechanics, the laser beam which will just be a large beam that shoots directly up (so that if the player is directly above then they are obliterated) and the homing rockets which will be fired as the mothership is moving and provide another obstacle for the player to dodge, these will be fired and then locate the player and 'seek' to hit then by flying towards them, again the player shall have to fly away from them so they will be constantly updating the players location. They also will rotate to always look at the players direction. See below for more information about the rocket Ai idea, as there is a behaviour model laid out for this mechanic itself.

The primary attack, of the laser beam, should be carried out in a separate state from the 'moving' state, which could be the 'attack' state. In this state the mothership should lock its location along the bottom of the screen and charge up the shot, this will add an immersive element as the user will know that the ship is going to shoot so will have to dodge it. This will also be used against the mothership to win the boss fight, when it is stopped in the shooting state this should be when the player can open fire onto the mothership to do damage to it, damage which shall be displayed to the user in a sort of health bar, which will often be familiar with the user in boss fights in other games.

Overall, the mothership boss idea should definitely make the gameplay more interesting as it adds a totally new element to the asteroids game as to where it is no longer just a sandbox game about dodging asteroids but there is a kind of story that leads up to this big alien fight, which shall be exciting and add lots of features that the user should learn to fight and then beat at the end. It changes the difficulty of the game as it progresses to lead up to the final boss which is meant to be hard to beat for the player and puts to the test the movement mechanics they should have nailed up to now while dodging the asteroids and fighting the aliens.

Rocket Weapon Ai Object Design (FSM)


Actions:
- 'kill' the player
- play sound effects / visual effects
Player Hit By Rocket
Transition: runs after previous state.
Actions:
- delete gameObject from the scene
- play sound effects / visual effects
ROCKET DEAD
Transition: if rocket collides with player object
Transition: if rocket has been active for a set time or is hit by a bullet.
ROCKET SPAWNS
Default MOVEMENT state
Moving Towards Player
Transition: if users position changes (maybe constantly updating)
Rotate Towards Player
Actions:
- Instantiate the ROCKET object into the scene, infront of boss's gun
- alert the user / let them know the AI rocket has spawned
Actions:
- detect users location
- fly / move forward towards the user location
Actions:
- change rotation of the object as it is moving

Above find the Finite State Machine Graphical Representation / Design of my proposed Rocket Ai Weapon

As sort of explained in the mothership Ai design proposal, the rocket shall be a weapon that is utilized by the mothership boss in the main fight at the end. The rockets shall be spawned in whenever the mothership is shooting. They will have a default state that is to 'seek' the player and fly towards them, rotating towards them to look like they are realistically hunting the player down. The rockets should have sound effects to know when they have been launched and they should have a constant speed flying towards the user.

If the rocket collides with the player, it should 'explode' (with sound effects and an explosion visual effect to add depth to the feature) and it should kill the player, making them respawn and lose a life. The user should be able to shoot the rockets down before they are hit, and this should cause the rocket to explode in the same way it would if it was to hit them. Also, the rockets should have a set lifetime so that if they are not shot down / don't hit the player then they will explode on their own so that the screen isn't constantly crowded with rockets seeking to destroy the player as this would be hard and too distracting from the main gameplay.

Being a secondary weapon for the mothership I think that the different challenges the player has to face within the boss fight amplifies the gameplay experience as additional dangers add double the challenge in an already challenging environment.

Implementation Description

Alien UFO Ai Implementation

Creating Alien Prefab –

Much alike the asteroids to get start I made an alien prefab which can be instantiated to spawn in an alien. The prefab has 4 sprites to randomly choose from when instantiating and the typical rigidBody, Collider and sprite renderer. The alien prefab is linked to AlienAI.cs which is the script class which will control the behaviour of the Ai. The script class also contains sound effects and visual effects linked to the Alien, upon spawning there will be sound effects played and upon being hit / killed or shooting there will also be sound effects or particleSystem effects. The alien also doesn't collide with asteroids, it only collides with the boundary and the player and the players bullets.

Creating Alien Bullet Prefab –

The same as the players bullet, the alien bullet has a prefab. This prefab is linked to, and can be instantiated by the alienBullet.cs script class, which controls its behaviour. The game object is the same, with the sprite, collider and rigidbody and it behaves in the same way, travelling until it reaches a boundary or the player. Alien bullets do not collide with asteroids, as when I did have this implemented it caused too many splits and barriers and it made the alien Ai useless compared to how I wanted them to be when they just kept getting hit or getting in the way of asteroids.

Spawning the UFOs –

Based off my design I implemented the Alien prefab that spawns at a randomly chosen point picked within the screen's height and width bounds. It uses the class AlienSpawner which is the script for spawning, in this script there is the Spawn method which picks a random point on the screens X and Y axis and then instantiates an alien prefab on this point. The spawning method is called once the player reaches 50 points (this value of points that need to be achieved before aliens start to spawn can be changed if the player wants to). There is a UI pop up and sound effects alerting the player that 'aliens have found your location' and then after the time increment, which can be changed by the player if they insist, the alien spawns in. After the initial spawn, every time the alien is killed the spawn method is invoked with that same time increment, so that another will spawn a few seconds after the last one was killed.

Ai Tracking, Floating and Moving Toward Player Location (*Movement State*) –

In my design I proposed that the alien is constantly 'seeking' the player and moving towards their location. Upon instantiating the Alien Ai, the first thing the script will do is locate the 'player' game object, then it can use the player objects 'transform' properties to find its location on the screen relative to the alien. Simply using the inbuilt Vector2 method of MoveTowards, I set the alien to be moving towards the players location with in the update() function so that its constantly updating with every frame and always moving towards the players current location. There is a conditional in the update function that calculates and checks the distance between the player and the alien, and if its with the distance that I proposed in the design then a flyAround() function is called. This will add force to the alien object in random directions, but as soon as it moves out of range then it will continue following the player again.

Shooting (*Shooting State*) –

In order to achieve the shooting mechanic that I aimed for in the proposal I simply called a repeating invoke function when the alien prefab is spawned in, this function called the Shoot() method which instantiates a bullet, and adds force to the bullet in the direction of the player (which is found by normalising the Vector2 positions between the player and the alien Ai). The time between shots can be changed depending on the user but I have it set to 5 seconds in my build, so that 5 seconds after the alien spawns it shoots, and then every 5 seconds recurring after this. The bullets do not have any tracking behaviour but just move forward towards the players location, at the time of shooting, at a constant speed.

Taking Damage and Dying –

The alien can be shot by the player just like the asteroids. After testing with different values I found that a fun number of shots to kill the alien would be 3, so I made sure that when the bullet collided with the alien it just incremented a damage counter, which then would eventually call the alienDie() function when the hit counter was equal to the maximum damage it could take. Aliens give +25 points when killed. Every time the alien is collided with a bullet I set its velocity to 0, as I had a bug where the alien would get flung away by the momentum of the collision and then it would take a second for the alien to fully stop and then continue to follow the user as it should do. The hit counter on the aliens individual instantiation would be incremented with each hit, and then a conditional would check how many hits the alien has had. If the hits are less then the maximum damage then the alienHit sound effect would run. If the hits are equal to the maximum damage that can be taken by the alien, then the alienDead sound effect would run and the gameObject would be destroyed with an explosion particle effect.

Upon the boss battle starting all of the alien and asteroid instances on the map are destroyed. Upon player death all alien bullet instances are destroyed. Upon player death, aliens remain still and wait for the player to respawn. AlienBullet class is shared with the mothership laser weapon, see more on the mothership section.

All of the AlienAi mechanics described have been bug tested strenuously and fixed to allow for perfect functionality as I planned in my game design. This can all be tested by the marker and code can be checked for further detail into how I achieved the functionality that I wanted to with the alien Ai.

Mothership Ai Implementation

Creating Mothership Prefab –

The mothership prefab is created like other prefabs, in such that it can be instantiated at a point on the screen and behave with behaviours defined in the Mothership.cs script class. The sprite was edited by me to allow for good visual placement at the bottom of the screen and the prefab contains the default rigidBody, collider (of which I had to use a polygon to trace the mothership and make the hitbox as perfect a shape as possible, which can be tested when shooting at the mothership or crashing into it), and the sprite renderer. Upon instantiating the prefab, I set the mothership to have a kinematic rigidBody every time, this means that it cannot move, turn or rotate based off of external forces such as collisions, as I had some problems with it spinning out when colliding with bullets or the players ship.

Spawning the boss –

Spawning is controlled by the mothershipSpawn.cs script class. Within this class the Spawn() function is used to pick a random X axis coordinate within the screen bounds, and then uses a default Y axis coordinate of 90px alongside this to instantiate the prefab at the desired position, which as following my design will always be along the bottom of the screen and the mothership can not deviate from the 90px Y-axis coordinate as it follows the user along the screen. The Spawn function is called from the gameManager class when the player gets 300 points, after testing and playing the game I thought this was a good point value in order to get to the minigame, it doesn't take too long to get to and will be good for your testing to see how the minigame works. This amount of points to start the game can be changed to whatever is desired. Upon starting the mothership game, a UI pop up and boss sound effects will start to let the user know they are in the boss fight, and all of the alien and asteroid instances on the map will be destroyed and stop spawning just for the duration of the boss fight.

Movement Mechanics with Rocket Shooting (Movement State) –

If the mothership is in the movement state, as defined by a boolean value, then it will calculate a targetPosition based on the players transform.X coordinate. After the targetPosition is calculate the MoveTowards() function, that is also used for the alien movement, is used to move the mothership towards this point, with a set speed. Upon spawning the movement will start, and due to this code being in the update() function, the mothership will always be moving to the players current location. While in each instance of the movement state, 2 rockets are fired at regular intervals, of which there is about 1.5 seconds difference to allow them to not clip together and get in the way of each other. Read below section about the Rocket Ai to learn how the rockets work.

Mothership Shooting Main Weapon (*Shooting State*) –

The motherships shooting state is called based upon a Boolean value which defines whether it's moving or shooting. Upon the Shooting state being accessed, the mothership stops all movement along the X-Axis and starts the mothershipShoot() method after a set time period, which is chosen to be 1.25 seconds for my build as this was the most balanced and well-working value based off of testing and playing. The mothershipShoot() method is responsible for instantiating a laser beam (which shares the alienBullet.cs script class) that will be projected up from the mothership straight vertically through the map, as intended by my planned game design. If the player touches the beam then they are obliterated. A sound effect plays when the beam is shot and the beam gameObject is destroyed after a set amount of time, which again can be changed depending on player preference, but I have this set to 2 seconds as this is a good amount of time for the player to be able to shoot at the mothership to do damage while it shoots, which was the planned method of being able to destroy the mothership based off of my game design plans and ideas. After the shot is destroyed a method called enableMothership() is called which will re-transition to the movement state, perfectly matching the planned transition of 'when the shooting ends' in the game design Finite state machine. If the main weapon kills the player then the beam will be destroyed and the mothership will be enabled back into movement mode when they respawn.

<u>Mothership Taking Damage and Dying –</u>

As intended in the game design plan there is a health counter for the mothership, initially the mothership has a health value of 100. Similarly to the alien, each hit that the player bullet makes on the mothership adds 1 to a hit counter, which in turn decrements the motherships displayed health counter, in the user's UI. When the mothership hit counter is equal to 100 hits, the mothershipDead() function is ran. This function plays sound effects and particle explosions to mimic the mothership being defeated after removing the gameObject. Then, another UI pop up comes onto the screen, the player gets 100+ points and is indicated that the boss is beat, but the battle continues, from here on out the game continues in swarm mode with the sole purpose of killing more aliens and racking up points.

Again, all functionality of the mothership matches up to the original ideas I had about the design of my boss fight and boss attacks as described above in the Finite state machine design plans. All the functionality has again been strenuously bug fixed for any existing bugs that were found after many, many iterations of testing.

Mothership Rocket Weapon Ai Implementation

Creating Rockets Prefab –

Rockets are a prefab just like most other gameObjects that are spawned in at set times. They have a well defined collision box (so that the collisions are as accurate and fluid as possible), a rigidBody to allow movement and the sprite renderer to show my rocket sprite. The rockets are linked to the Rockets.cs script class which controls behaviour of these prefabs.

Instantiating Rockets and Moving Toward Player (during *Mothership Movement State*) –

Rockets are spawned at intervals of the motherships movement state as it is tracking the users position, this is described more in the mothership implementation description. The rockets spawn in and then find the players gameObject, then use the same Vector2 MoveTowards function that has been used for other objects to move towards the players function. A mathematic operation is also carried out in order to always update the angle that the rocket is facing so that it is always looking in the direction of the player. After a set amount of the time the missileExplode() function is ran, which will destroy it. See more about this below.

Rocket Collisions Ai / Rocket Exploding –

If a rocket collides with a player then, as defined in the game design plan FSM, the rocket explodes with the missileExplode() function, this starts an explosion particle effect on the rockets location, and plays an explosion sound effect. The collision will also 'kill' the player, when the player dies off a rocket explosion all other rockets in the map are exploded so they are not swarming the user when they spawn in, which was a bug that I had that I noticed needed fixing. If on the other hand, a rocket collides with a bullet, it will explode in the same way with the same function, but the player score will be incremented by 1, this can be changed if you want to make the player get more points per rocket shot. The rockets have a self-detonate timer if they don't hit the player in a set amount of time, so that there isn't a lot of rockets flying around the field at any one time, I set this to around 10 seconds in the build so that this is a fair and good value for testing.

Given that the Ai of the rockets is similar to the Ai of the alien in terms of following the user I did reuse some of my code but in turn created a brand new intelligent game object, which fits my design plans very well and has the function that I wanted to implement.

In this documentation I have tried to go as in depth as possible with how my implementation matched up with my initial plan, and all things considered, the final output I am very much happy with when looking back at the initial ideas I had while developing, many of the ideas I had originally weren't changed and I managed to implement over time.

Some other fixes to the base game that I made during the AI development

*not related to AI specifically

- Added fire rate to player weapon
- Improved difficulty settings in final build to make asteroid difficulty scale better
- Tidied up methods that had redundant code
- Bug fixed all previous submission even more
- Big fixed all new implementation
- Rescaled all of UI and screen objects so that there is more 'space' on the screen even though it's the same resolution, makes adding more elements to the game a lot more fluid and look better.