Dawid Rybinski
Student ID: 201556461

Model Behaviour (FSM)

For this project, I have opted to use the Finite State Machine (FSM) model to create a dynamic and responsive UFO AI. This new UFO model incorporates several behaviours to interact with both Player and Asteroid objects in a more engaging manner.

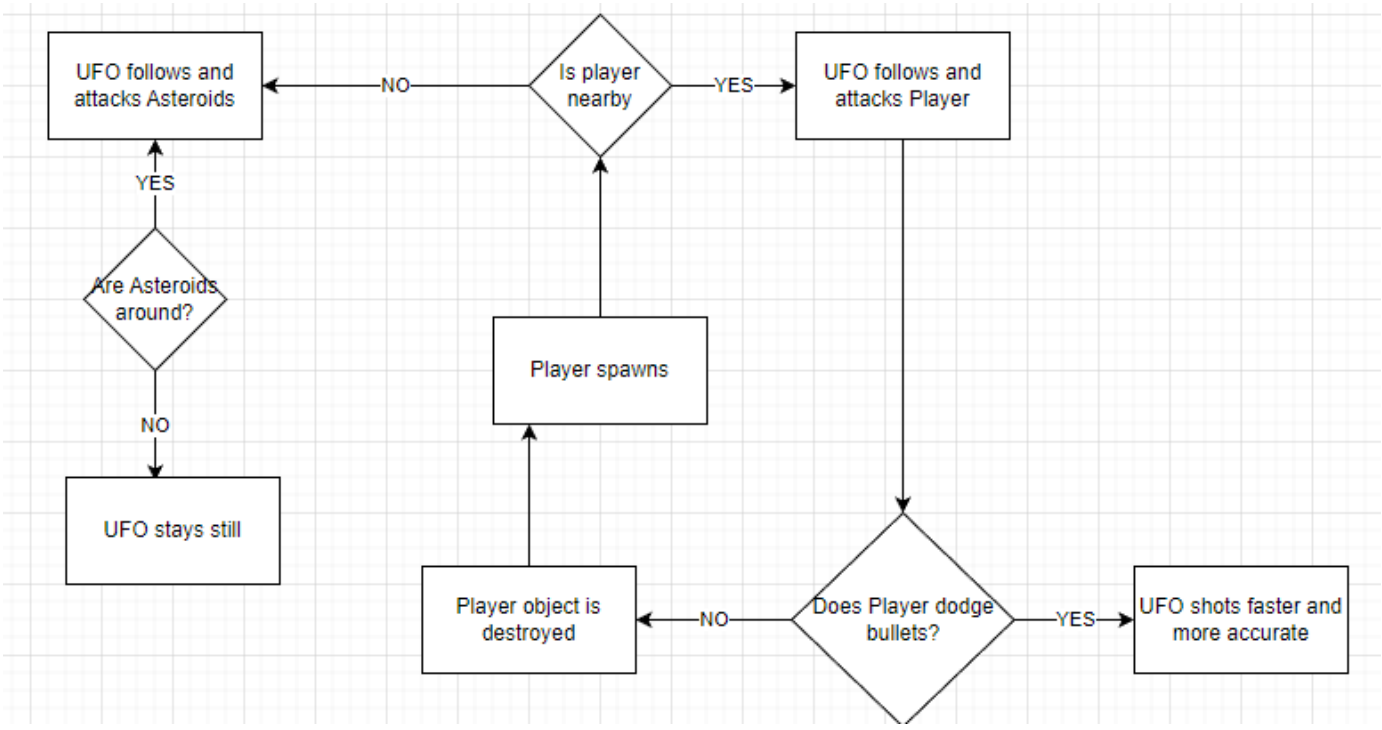
The UFO's primary target will always be the Player. Upon detecting the presence of a Player, the UFO will pursue and attempt to shoot them down. To make the UFO a more formidable adversary, it will adapt to the Player's movements by observing their dodging patterns. If the Player consistently dodges incoming bullets, the UFO will respond by increasing its bullet speed and accuracy. This adaptive behaviour resets one the UFO is destroyed.

In the event that the Player is eliminated, the UFO will shift its attention to nearby Asteroids, following and firing at them. However, if the Player reappears withing proximity, the UFO will promptly switch targets and pursuit of the Player.

In the unlikely scenario where neither the Player nor any Asteroids are present on the map, the UFO will simply remain stationary. This behaviour ensures that the UFO remains a persistent and adaptable threat, proving a more challenging and enjoyable gaming experience.

Graphical Representation

UFO Logic



Dawid Rybinski
Student ID: 201556461

Implementation

Class: AlienSpaceShip

MoveTowardsPlayer() – This method uses a pre-implemented method and Vector3 "Vector3.MoveTowards" where arguments are: transform and target position and speed * deltaTime.

Respawn() – UFO spawns at random location on the main camera. In order to do that I calculated the boundaries of the camera view for example, "float xMin = cameraPosition.x – /+ cameraWidth / 2f;", with previously assigned variables such as cameraPosition, cameraHeight and cameraWidth and did the same with Y.

Then I used these calculations to generate a random position withing the boundaries, and make sure that UFO will not spawn too close to middle as it is where Player spawns.

FindNearesAsteroid(float searchRange) – Here I used 2D physics to retrieve all colliders within a searchRange and then iterate through all colliders found.

While iterating the program is checking if any of colliders have tag "Asteroid", if yes it will calculate the distance between UFO and current asteroid. If the current asteroid is less than minimum distance, which was assigned previously, update the minimum distance and set the nearest asteroid to the current collider's GameObject.

ELSE return null.

MoveTowardsnearestAsteroid(GameObject asteroid) – Code is identical to **'MoveTowardPlayer()'** method but we pass argument asteroid in this case because we already have an access to Player through "target" variable but not to Asteroid yet.

IncrementSuccessfulDodgeCounter() – This method is public as it is connected to AlienBullet class. In class AlienSpaceShip we only increment an int and if it get bigger we set up Boolean to true and increase bullet speed and accuracy in **Shoot()** method.

ShootAtAsteroids() – Here we increment shooting timer by the time elapsed since the last frame and check if it reached or exceeded the shooting rate.

If yes then we reset shooting timer, calculate direction vector from UFO's position and shoot.

Shoot() – Similar to function above we increase shooting timer and first check if Boolean from **'IncrementSuccessfulDodgeCounter()'** is set to true, if yes we UFO shoots faster and more accurate.

We also check if shooting timer has exceed the shooting rate. If yes we reset shooting timer, calculate position of our target and in the same if statement we again check if boolen from method mentioned above is active. If yes we increase accuracy and after that UFO fires.

Dawid Rybinski
Student ID: 201556461

Class: AlienBullet

Fire() – a simple method that will add force to the object and it will be destroyed after max lifetime(10 seconds).

isOffScreen() – a method that checks if bullet went behind camera. By using Vector3 I got: 'Vector3 screenPoint = mainCamera.WorldToViewPoint(transform.position);' which allowed me to return X or Y and see if these coordinates are outside of camera.

Update() – in this method I just increment a public variable in AlienSpaceShip in order to make a Boolean there true and increase bullet speed and accuracy.

Additional notes

I decided to keep the feature where player's bullets push away UFO instead of only dealing damage, it will be easier for player to run away, however, each bullet takes 25 damage from UFO, after 4 bullets it will be destroyed and respawn after 10 seconds, it will give player some time to destroy asteroids and 150 additional points.

UFO can win the game as well by scoring 6000 points or more. UFO has similar score system to the one that player has but UFO gets 5 points for the biggest asteroid, 20 for middle size and 40 for smallest, it can also get 70 points for destroying the player. I also added UFO win screen.