Practical Gaming 2024

# Name of Student: John O’Sullivan

# T Number: T00234079

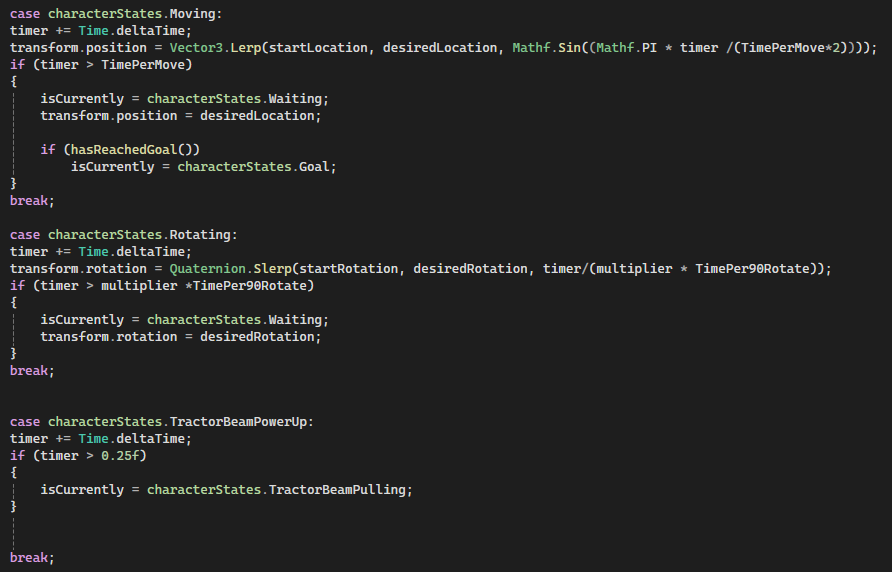
# Name of Project: Tractor Beam

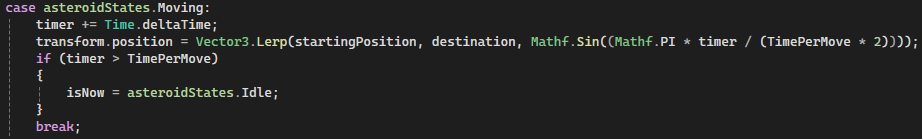
# Gameplay

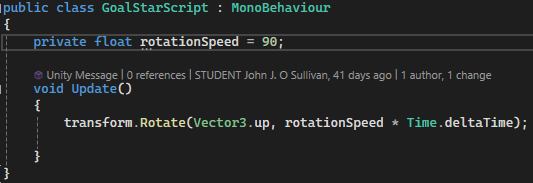
The object of my game Tractor Beam is to guide a U.F.O character to the star at the end of each level within the time allowed. For the purpose of explaining this game, I will refer to the main character as Teebie (this name is not used or referenced anywhere in gameplay). To reach the goal of each level, which is represented by a rotating star, you need to use the tractor beam Teebie is equipped with in order to move asteroids out of the way and create a path to the star. The tractor beam can only be used if there’s an asteroid two spaces away from Teebie with nothing separating them. Teebie must also be facing the asteroid with their propellers, the tractor beam will not work properly if Teebie is facing the asteroid from the side or with their antenna. Teebie is controlled using WASD to move with the arrow keys being used for rotation and the space bar activating the tractor beam. Teebie cannot perform another action while in the middle of another one. For instance, Teebie cannot move while using the tractor beam. Each level has a different time limit with the longest being thirty seconds and the shortest being only fifteen seconds. Technically speaking, every level has exactly 0.99 seconds over their starting time. This was done to make the maximum time display for longer giving the player a greater sense of how much time there is to beat each level. The only possible way to lose a level is for time to run out. The timer is displayed in the top right corner of the screen at all times during gameplay. During gameplay, there are buttons in the bottom right corner of the screen, The “Restart” button restarts the current level from the beginning if the player gets stuck and/or is running low on time. The “Main Menu” button returns the player to the title screen of the game but will erase all their current progress. Upon completing or failing a level an appropriate fanfare will play along with the text “Level Complete!” or “Time’s Up!” depending on the outcome. After the text disappears, buttons will appear in the middle of the screen. The “Play Again” button functions identically to the previously mention “Restart” button. In addition, there is another button to return to the title screen, also labelled as “Main Menu”. If the player is successful in completing the current level, a third button will appear labelled as “Next Level”. This button allows the player to progress to the subsequent level in the game. If the player runs out of time during the level, the “Next Level” button does not appear. Once all levels are completed, the game will progress to a credits scene from which the player can return to the main menu.

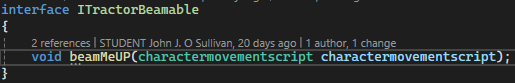
# Coding

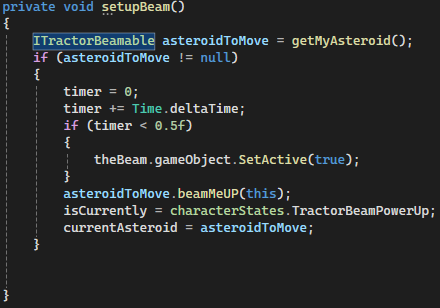
* Frame Rate Independence
  + To ensure frame rate independence in my game, I have several instances of Time.deltaTime in my code used for the moving objects.

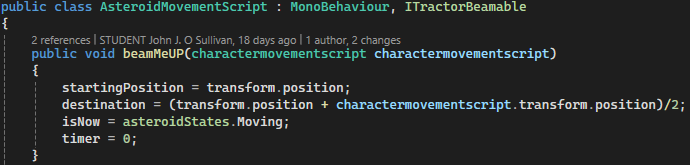






* Interfaces
  + I have an interface called ITractorBeamable which, as the name implies, makes it so that any object it is applied to can be pulled by the Teebie’s tractor beam.

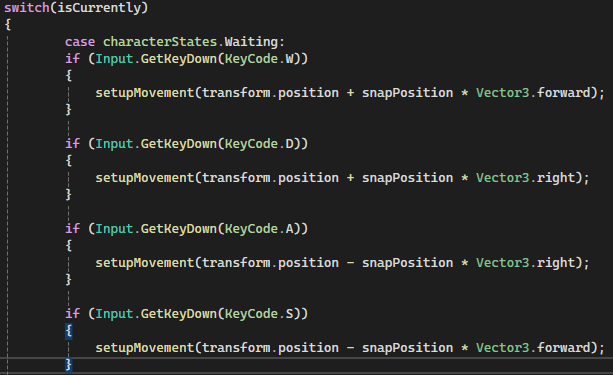


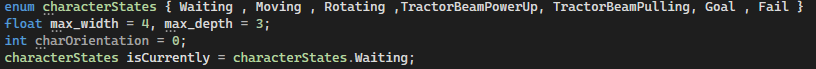


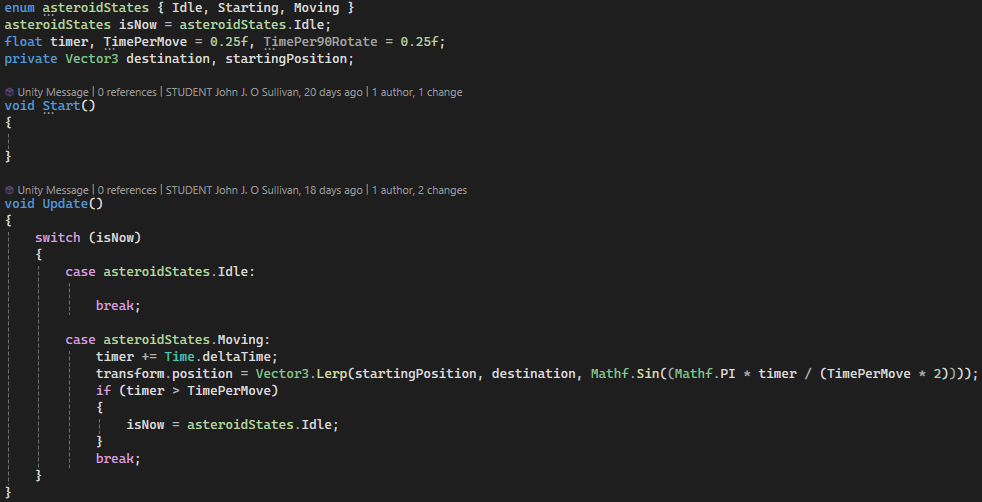
* Inheritance
  + The script on the Goal Star inherits from the MakeMeAppear script to make the blocks appear.



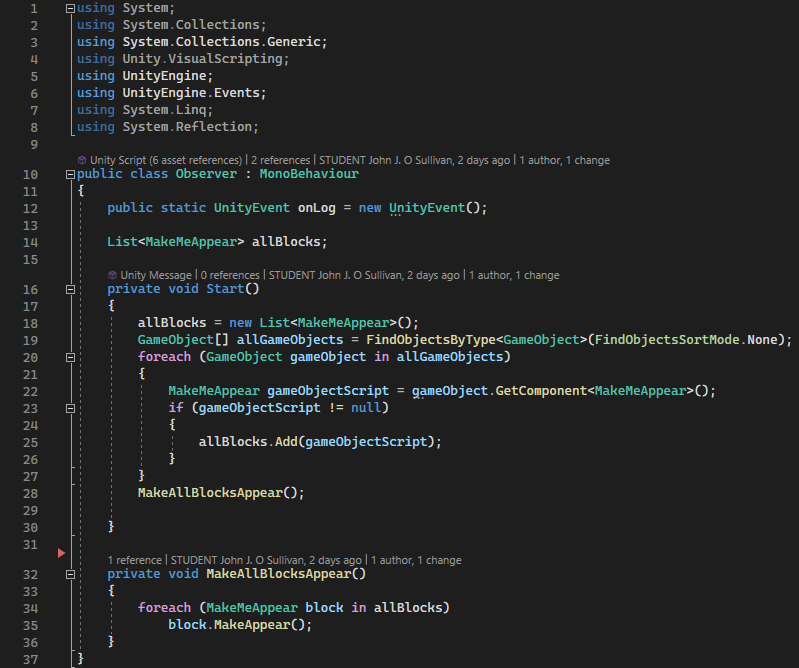
* Case pattern
  + Both Teebie and the asteroids have states they switch between depending on what is currently happening in the game.





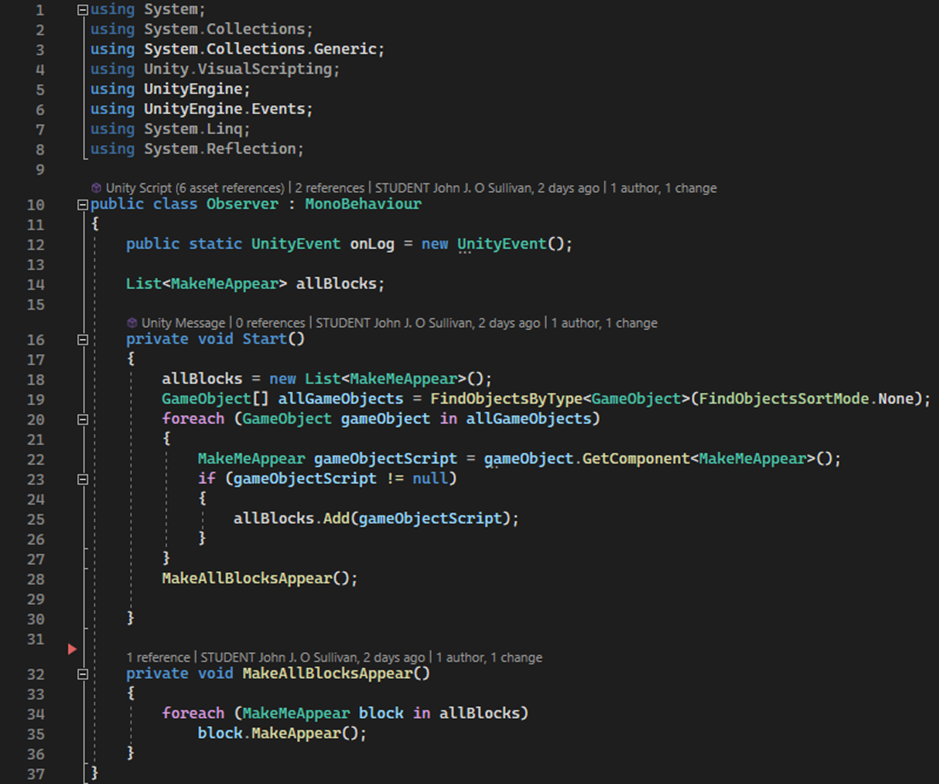


* Observer Pattern
  + I have an observer patter that searches for GameObjects in the scene when called

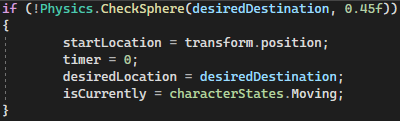


* Polymorphism
  + The script on the Goal Star inherits from the MakeMeAppear script which itself is used by the Observer Class

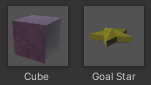




* Communication between scripts/game objects
  + Teebie communicates with the script in order to detect if there is an object they are supposed to collide with through use of a check sphere (i.e. Collision Detection)



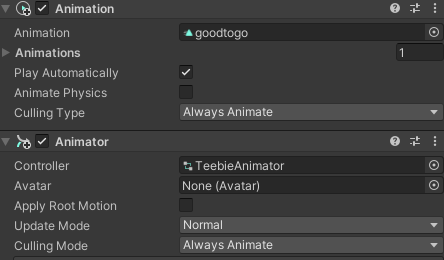
* Instantiation and Prefabs
  + I have prefabs for both the Goal Star and the cubes that make up each level



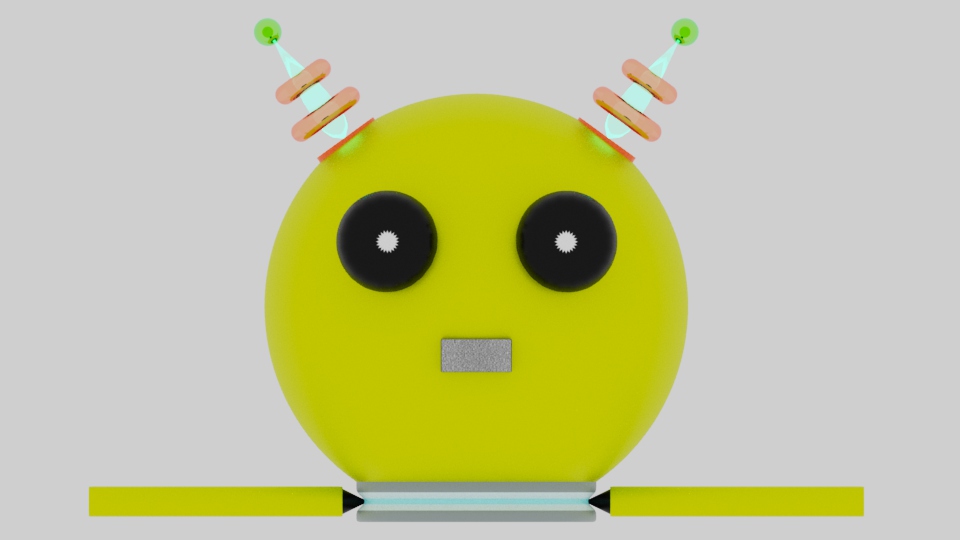
* Magic Numbers
  + I have several variables in my code used to store numeric values. Some examples are below



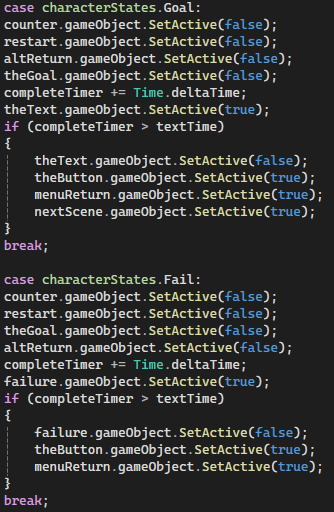
* Model Animation
  + Teebie has an animation for spinning their propellors while moving activated using an animator



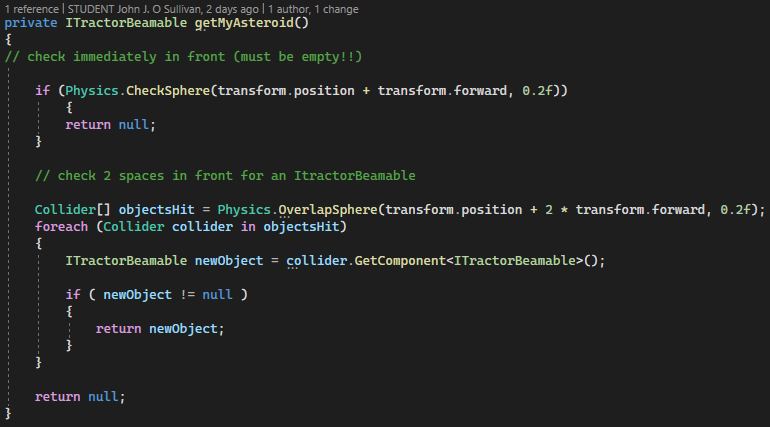
* Self made models and or animations
  + Below is a render of the 3D Model of Teebie I made for this game



* Interactions between objects/scripts
  + When Teebie reaches the goal, certain buttons on the UI are set to being either active or inactive



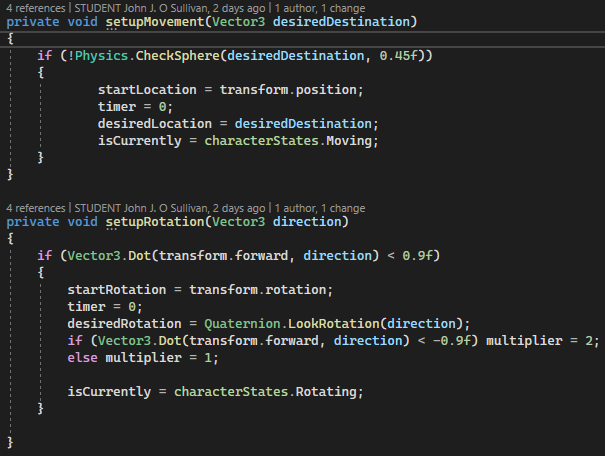
* Proper code placement
  + Teebie’s movement script creates an interface that the asteroid movement script uses in order to move when the tractor beam is activated



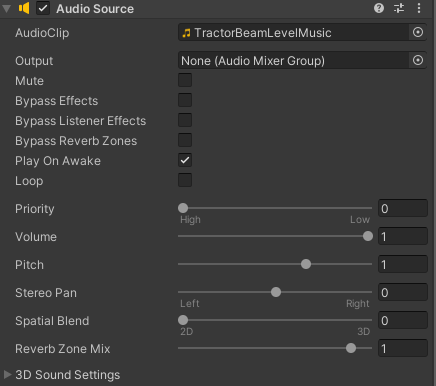


* Code repetition
  + The code that enables Teebie to move and rotate are both called numerous times using methods

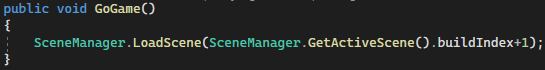




* Feature 1
  + I have several audio sources throughout the game to play music, fanfares and sound effects



* Feature 2
  + I have a script called gotothegame which has a method that progresses the player to the next scene



* Feature 3
  + I have a quit button on the main menu which as the name implies, closes the application.

