Practical Gaming 2023

# Dawid Jaworski

# T00196669

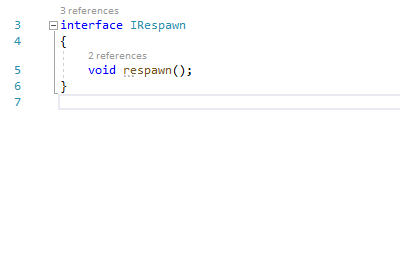
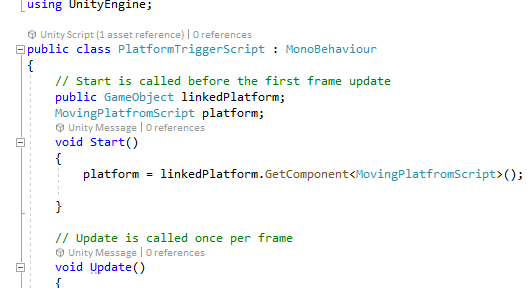
# Dummy Jumps

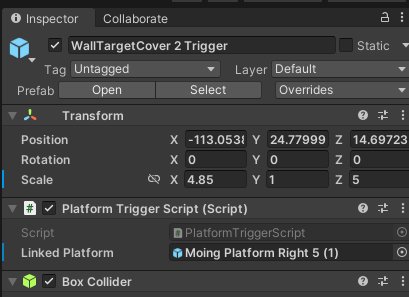
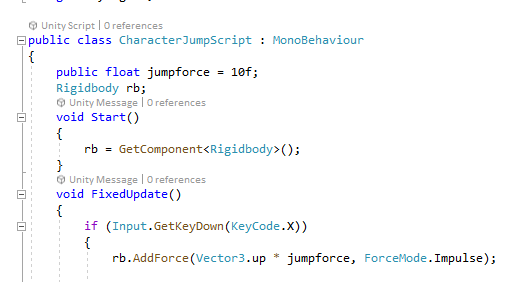
# Gameplay

My Game is a simple platformer. The player moves using WASD and jumps using SPACE the camera is operational using the mouse rotations on its axis. The player must make a series of jumps to unlock further platforms and progress to the end.

# Coding

Under each of the following headings, please describe the concept, why is it or isn’t it useful/needed, where do you implement in your project, you may provide screenshots or cut and past code segments etc..

* Frame Rate Independence
  + Frame Rate Independence is used in video games to makes sure the game works on different frame rates without any problems, some games might run on a specific frame rate and encounter bugs or glitches when the frame rate is changed.
  + In my game I use frame rate independence for the character movement using time.deltaTime to make sure the character will behave the same no matter the frame rate of the machine.
  + 
* Interfaces
  + Interfaces are important components they provide the means by which players can interact with the game world. The interface is the bridge between the game mechanics and the player, allowing the player to control and navigate the game world.
  + I use an interface in my game to respawn the player after failing so the player can keep going.
  + 
* Inheritance
  + Inheritance is a programming concept that allows programmers to write code once and reuse it for similar tasks, it can be by making a basic enemy script that would contain health, move speed, damage and reuse this code for different enemy types
  + My game uses Inheritance for platforms so that one basic movement script can do a few different movement methods.
  + 

* Case pattern
* Observer Pattern
  + The observer pattern is a method for different parts of the program to communicate together. There are 2 parts the observer and observable an observer is interested in what the observable does and updates different parts of the game such as health bars or scores in games it even controls NPCs in games so they can react to the players movements.
  + My game doesn’t use any observer patterns
* Polymorphism
  + Polymorphism is often used to create game objects with shared properties and behaviours, while still allowing for unique behaviours for each object. A good example would be enemies you can make a basic enemy object and just add unique values to the new items for example fast attacking enemies or area of effect enemies, both enemies would contain a basic speed, health and damage script but it would be different on how it works in the game.
  + My game uses polymorphism for triggers as they do similar things but are different at the same time.
  + 
* Communication between scripts/game objects
  + Communication between scripts and game objects are a very important way to make games more interactive for the player, you can make triggers or events that react to a player entering some area or certain dialogue to play upon walking into a room.
  + My game uses communication as triggers are involved in most of my platform areas so the player can keep advancing.
  + 
  + By linking platforms to triggers and platforms to platforms I am able to make stuff seem smoother as the player advances while giving a little bit of anticipation if the player will make the timed jump.
* Instantiation and Prefabs
  + Instantiation refers to the process of creating a new instance of a game object at runtime. When a game object is instantiated a copy of the data structure is created in memory which then the scripts can modify the object.
  + Prefabs are templates or blueprints for game objects that can create multiple instances of if the same object.
  + My game uses instantiation for the pillars, movement and triggers while the game launches.
  + In my game I use a prefab of the main character I also use a prefab for the wall and my inserted model.
* Magic Numbers
  + Magic numbers are hard coded values in the games code those values are usually never explained as to why they ware chosen those values can cause different kinds of bugs and glitches.
  + My game uses magic values for speed and height of the jump
  + 
  + Magic numbers should be avoided in coding in favour of named constraints or variables.
* Model Animation
  + Model Animation is the process of creating and manipulating animations for 3D models. To create animations for video games developers/animators would use a method called keyframe animation that captures a frame at which an action is to occur and the software would fill in the rest. Animation can also be done using inverse kinematics which simulate the way joints and limbs move in response to different actions. Mocap is another method which allows a person to capture animations from actions they perform themselves and apply those animations to the character.
* Self made models and or animations
  + My self made model is a Glock 19 the prefab for it came out nicely for the most part, the model doesn’t have any animations or interactions in the game it just floats and looks nicely.
  + 
* Interactions between objects/scripts
  + In my game the interactions between the objects and scripts is most apparent in the triggers to platforms interaction. Once you reach a certain point in the game you hit a trigger causing more platforms to appear and the player can move on.
* Propper code placement
  + Proper code placement refers to organizing code in a logical and efficient manner within the source code.
  + In video game development it is even more so important as games can involve complex interaction between multiple game objects, physics systems and user interfaces.
  + I believe my code is nicely organized as every script has its own function.
* Code repetition
  + Code repetition refers to copy and pasting / duplication of code
  + Duplication of code in video games can cause a lot of problems as there is a lot of complicated actions happening and duplicated code could interfere with some game objects.
* Feature 1
* Feature 2
* Feature 3