**Introduction to Programming and Program Design Methods Final Project**

**Student Information**

Name: Nathanael Setiawan

Student ID: 2440042866

Binusian ID: BN123727116

Binusian Email: nathanel.setiawan001@binus.ac.id

Phone Number: 081296030494

Class: B1BC

**Lecturer Information**

Name: Ida Bagus Kerthyayana

Lecturer ID: D5757

**Table of contents:**

**Introduction3**

**Goal4**

**Methods5**

**Introduction:**

As a new BINUSian attending the international computer science course, one of the first courses the class of 2024 learnt is Introduction to Programming. In this course, students learn all about Python from its theoretical concepts all the way to the practical coding. As a way to test the student’s abilities and understanding of Python so far, the lecturer, in this instance Mr.Ida Bagus Kerthyayana, gave the students freedom in creating something in python with the concepts learnt thus far as well as enforcing further learning by requiring students to use some external libraries and modules that are not taught in the class. With this premise, I have decided I would like to create some kind of video game or at the very least a demo version of one.

The video game I have attempted to make is a simple First Person Shooter (FPS), utilizing a relatively recently developed (2019), open source game engine called the Ursina Engine (https://www.ursinaengine.org/index.html) for Python. Created by, Petter Amland, this game engine is designed to help creating video games by really simplifying code and creating some prebuilt ready to use functions and classes for things such as the player character. Although the engine helps when creating a game, the documentation of Ursina Engine is not comprehensive for me and there are little to no tutorials in terms of systems I planned on creating, so I still had to test and tweak my code constantly to create some systems. And as this is the first time I’ve encountered the Ursina Engine as well as the first time of creating a big project, I was pretty excited to start.

The IDE I used during the project is Pycharm Community Edition 2020.2.1 and as this project’s nature is open source, all progress will be uploaded to GitHub with this link: <https://github.com/Nathanael126/3D-Shooter-PDM-Final-Project>

**Goals:**

**Project Purpose:**

Primarily for the learning experience of the developer, it has the secondary purpose of entertaining any audience that fits within the nature of this project.

**Targeted Audience:**

The main audience of this project will be the lecturers; audience is not a big variable in the development of this project as the project nature serves as a learning tool rather for the developer rather than a practical application.

**Project Aim:**

The aim of this project is for the developer of the project to learn on how to create a new game through a library not taught directly to them, as well as demonstrate the competency by creating a functional experience for the user.

**Objectives in-game:**

Shooting the targets and gaining points is the main objective in-game.

**Specification set by the developer:**

-Functioning gun

-Functioning targets

-Functioning player

**Additional vision (not mandatory) set by the developer:**

-Multiple levels

-Multiple guns

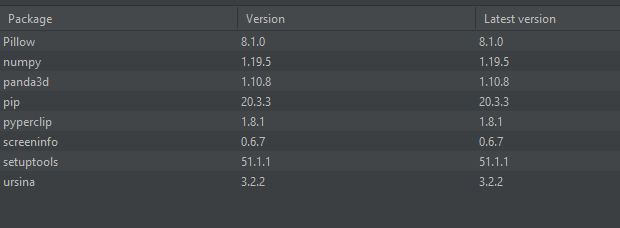
-AI

**Methods:**

**Overview:**

The game I created has been titled”*3D Shooter”*. It is a simple shooting game demo which currently only has 1 level as a testing ground which is a shooting range. The game is written entirely in Python code, and mainly utilizes the Ursina Engine for most of its code.

All of the libraries present in the project files of *“3D Shooter”* are listed in the following screenshot:

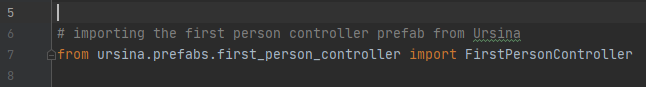


The package utilized the most is the Ursina engine, most if not all the code utilizes functions and methods that are recognized by the Ursina engine. The engine is based on panda3d, so that package is installed by default. pip and setuptools are packages immediately included from the IDE Pycharm, and when first installing ursina, Pycharm automatically downloaded Pillow, numpy, pyperclip, and screeninfo. Although I do not understand where they are utilized, as I did not reference them directly in my project, in the documentation these are all libraries that are considered dependencies for ursina.

**Prefab objects:**

Prefab (prefabricated) are objects/methods that have been created and included in the Ursina engine, ready to be used immediately with very little code to create these objects.

**First Person Controller:**

The only prefab to be considered a prefab inside the documentation of Ursina itself, the prefab creates an object, which is assigned to the Player variable. The prefab makes the camera’s parent to be the player object, and implements (WASD) movement controls as well as collision detection. The code needed to assign the player object to the variable is 2 lines long, the first line is for importing the prefab, and the second is assigning the variable, shown below.

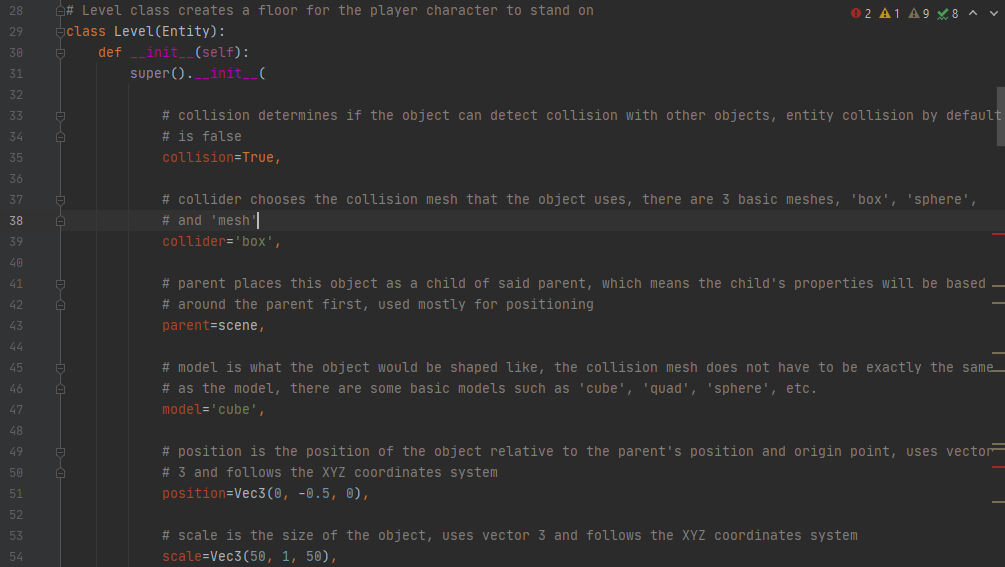


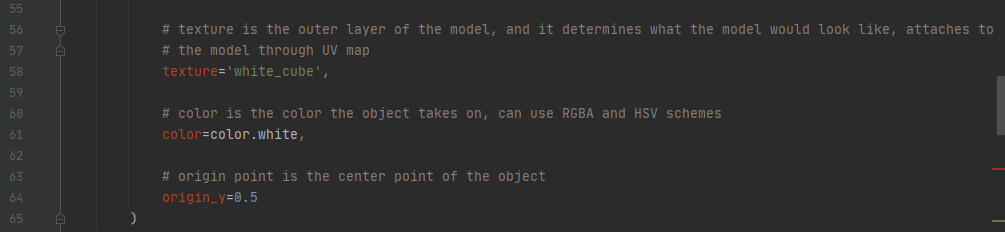
**Classes:**

Classes are utilized a lot, as Python is an OOP language, naturally libraries made for it relies on OOP concepts. Ursina is no different, as all objects in-game are just instances of classes.

**Level:**

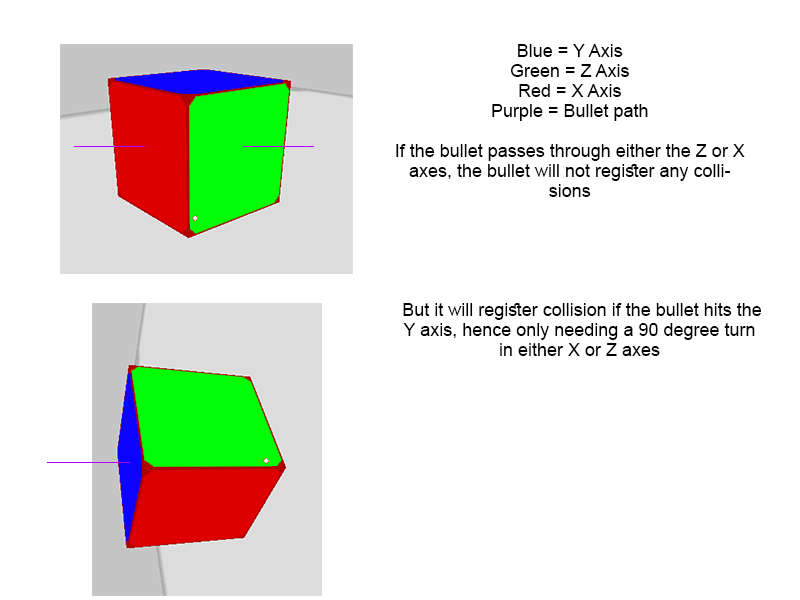
This class is used to just create a floor for the player to stand in; the code is shown below here.



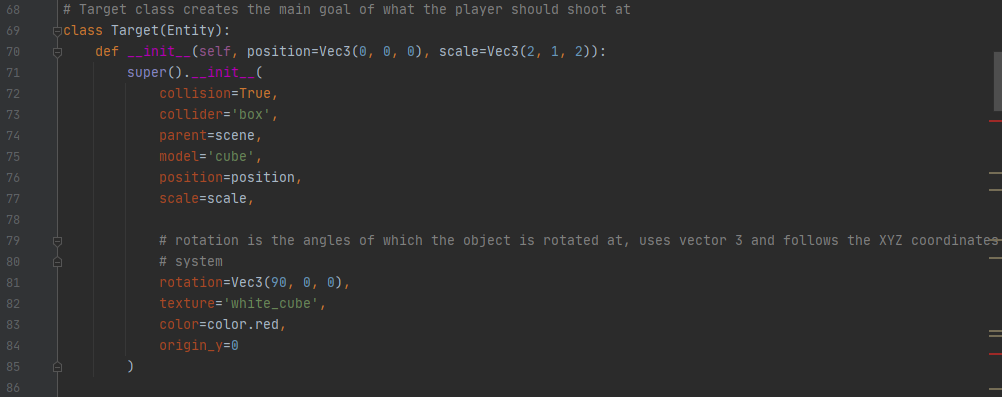


**Target:**

The target class is the class I spent the most time on. This is because the collision in Ursina is buggy, and there are little tutorials on how to use collision detectors such as raycast() and intersects(). The first collision detection method I tried was raycasting, but raycasts interact with their own models causing fake hits, and at the time, I did not comprehend how to know what entity it hits. The next method is intersects(), through a sample project I reverse engineered on how they work, but for some unknown reason, there is a bug where intersects() will not detect any collision from the X and Y axis of the Target class. Below is a visual on what the bug looks like.

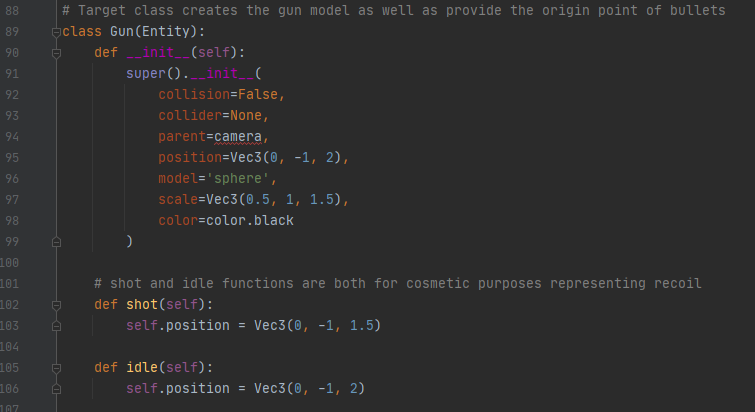


This bug cause a lot of issues for me, and with inconsistent collision detection from the target class as well as just learning how hit detection works through the API documentation, it took a significant portion of time, and the target is reduced to become a basic cuboid object, shown in the code below.



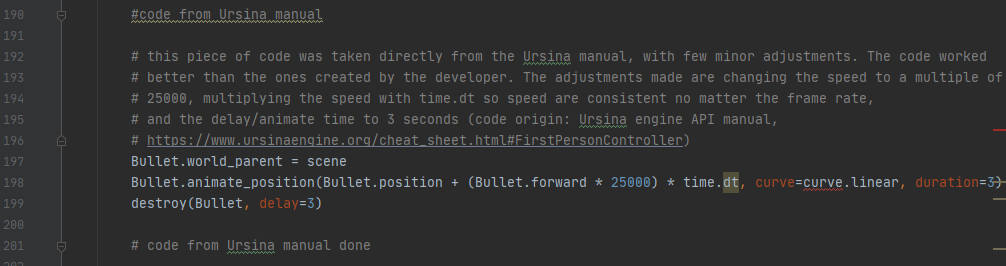
**Gun:**

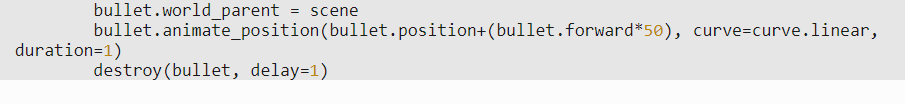
The gun is just a stretched sphere model with no collisions, meant to act as a visual of a gun and the origin point of bullets. The parent of the gun is the camera so it will follow the camera no matter what. Code is shown below.



**Bullet:**

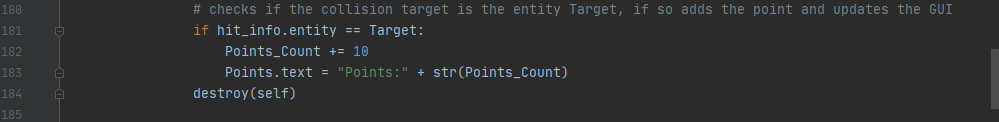
The bullet is a small cuboid with collision detection. The primary action of a bullet contains codes which are unoriginal with minor changes. This is because I could not find any other feasible way of creating the bullet behavior going forward while getting shot out of the gun. The code in particular is shown below as well as the original source code.

****

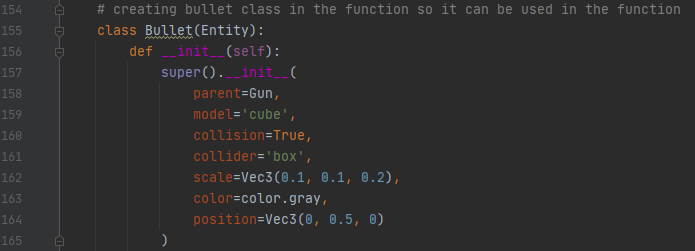
****

Link for the original source code: <https://www.ursinaengine.org/cheat_sheet.html#FirstPersonController>

The collision detection, just like the target, was difficult to program in, but not as difficult as the target, the bullet has better collision detection and because it was moving, there was a higher chance of any collision happening, the collision code can be seen below.

****

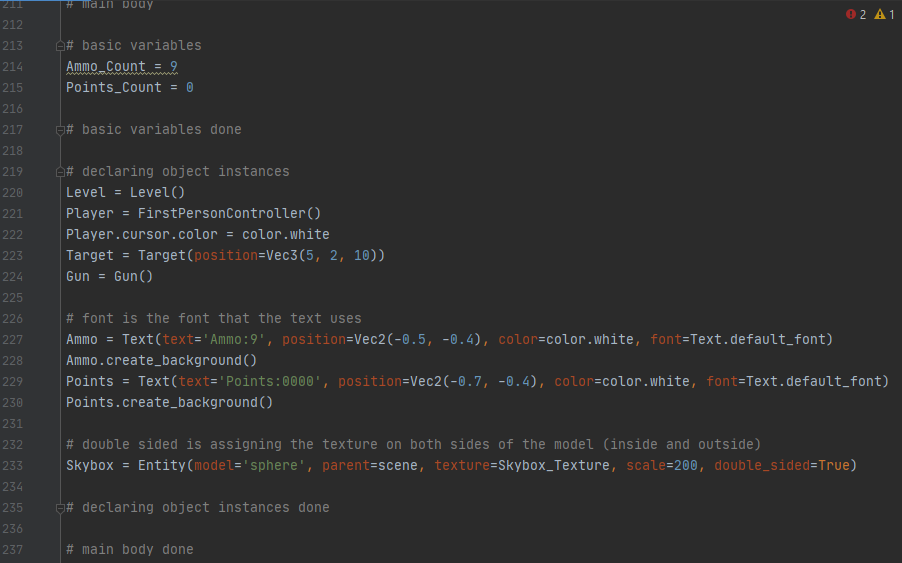
The rest of the class is shown below.

****

**Miscellaneous:**

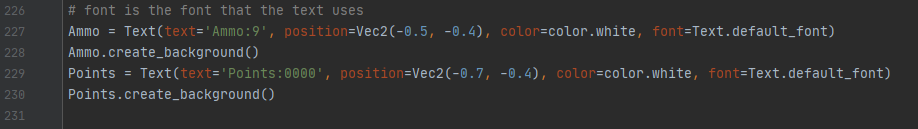
**Main body:**

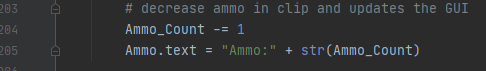
The main body of the class is used to create instances of the many different classes, and to just generally run the game as well as creating smaller things that do not really need a class such as a skybox which does not interact much. The code is shown below.

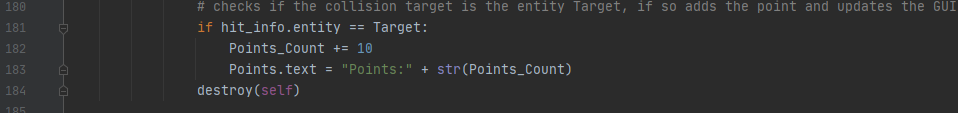
****

**GUI:**

The GUI is the final visual aspect of the game; it is created in the main body and it is updated when the variables change for that GUI. Code is shown below.

****

****

****