MID-TERM REPORT

PREPARED IN PARTIAL FULFILMENT OF

PRACTICE SCHOOL – 1

AT

TAMIL NADU SCIENCE AND TECHNOLOGY CENTRE (TNSTC), CHENNAI

A PRACTICE SCHOOL STATION OF

BIRLA INSTITUE OF TECHNOLOGY AND SCIENCE, PILANI

(A JOINT PROJECT BY STUDENTS FROM PILANI, GOA, AND HYDERABAD CAMPUSES)

STUDENT DETAILS

NAME OF STUDENT	STUDENT ID NUMBER
AMIT GUPTA	2020A7PS0105P
RAHUL CHORARIA	2020ABPS1048P
ARYAMAN AGGARWAL	2020B4AA1982P
PRATEEK UPADHYA	2020A7PS0056G
VENKATA NAGA SAI BHARATH THATHA	2020A4PS1904G
SRAVIKA LINGA	2020A7PS1310H
ANANYA KAPOOR	2020B3A70687H
VIRAJ MATHUR	2020B4A30956H
GADDAM SHRADDHA REDDY	2020B5A71957H
MAYANK SHARMA	2020B5AA2353H

ACKNOWLEDGEMENT

We want to use this opportunity to express our gratitude to everyone who supported us throughout this project.

We express our sincere gratitude to Tejasvi Alladi, our PS-1 instructor, for effectively guiding us throughout the course and ensuring all our doubts were cleared. He has been exceptional in carrying out the smooth functioning of our learning at the PS station.

We sincerely thank Mr. Lenin for guiding us from the course's inception. We sincerely acknowledge him for extending his valuable guidance and valuable critique of our work.

We would also like to extend my sincere gratitude to the Dean, Practice School Division, Birla Institute of Technology, Pilani Campus, and all other members of the Practice School Division for their generous and friendly attitude, and to the Associate Professor, Off-Campus Program, WILP at BITS Pilani for organizing all the insightful webinars by industry professionals for our better understanding of the sectoral world.

Finally, we would also like to thank our family members for their continuous support, which has been instrumental in the fulfillment of the project.

PRACTICE SCHOOL DETAILS

Station: TNSTC

Centre: Chennai, Tamil Nadu

Date of Start: 31 May 2022

Date of Submission: 27 June 2022

Title of Project: Interactive Space Game

Name of PS Mentor: Mr. Lenin

Name of PS Faculty: Mr.Tejasvi Alladi

Project Areas: Game Development, Gesture Control(Open CV)

About TNSTC: Tamil Nadu, Science and Technology Centre, was established in 1983 to design, develop, and fabricate Science and Technology, henceforth popularizing with the general public and students. It conducts various educational activities throughout Tamil Nadu by approaching educational institutes, research organizations, and philanthropists for their development programs.

INTRODUCTION

Context:

With technological advances and the ever-increasing diversity of gaming technologies, human interactive input device has become an important research topic. Rather than relying on conventional input devices such as a mouse, keyboard, joystick, or touchscreen, the human body could potentially serve as an input device directly in the form of a human interactive input interface.

It forms a new model of interaction where the movement of hands in the form of hand gestures is coordinated with virtual objects in the game. The project we have taken up aims to expose the general public to these advancements in human-computer interaction, sparking curiosity and spreading knowledge. The motives and agenda of our project stay true to the motto of TNSTC.

Problem statement:

Construct a working 3D Interactive Space game that users can interact with using hand motions and gestures.

Overview:

The project uses a low-cost Web camera mounted over the gaming screen display to provide an image feed to the hand tracking and gesture recognition system.

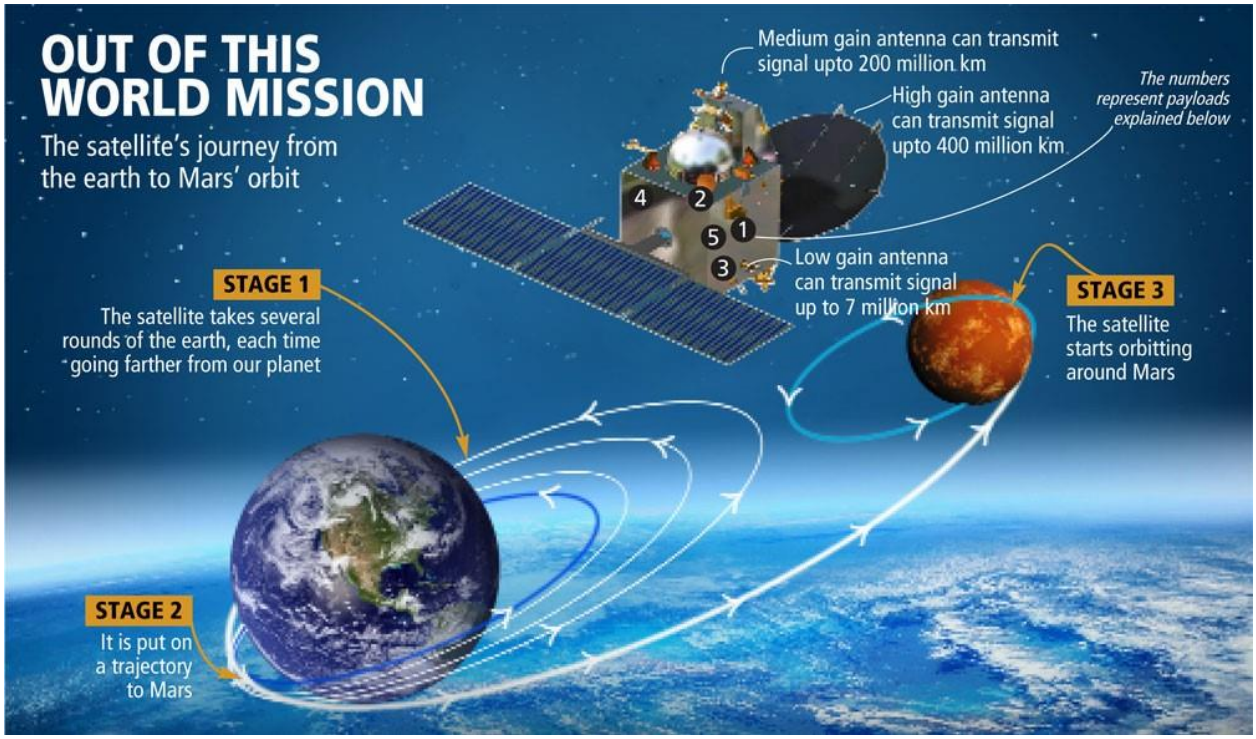
We hope to create an engaging game modelled after the Mars Orbiter Mission (MOM) launched by the Indian Space Research Organisation (ISRO) in 2013. The game is divided into three consequent levels for the user to get through before they successfully reach Martian orbit.

Via the abovementioned levels and information displayed at the start of the game, we intend for our project to be a fun and interactive learning experience. In the following chapters, we have provided a detailed description of our project and the tools used to implement it. We give an insight into the various features of every level and the technologies we have used.

METHODOLOGY

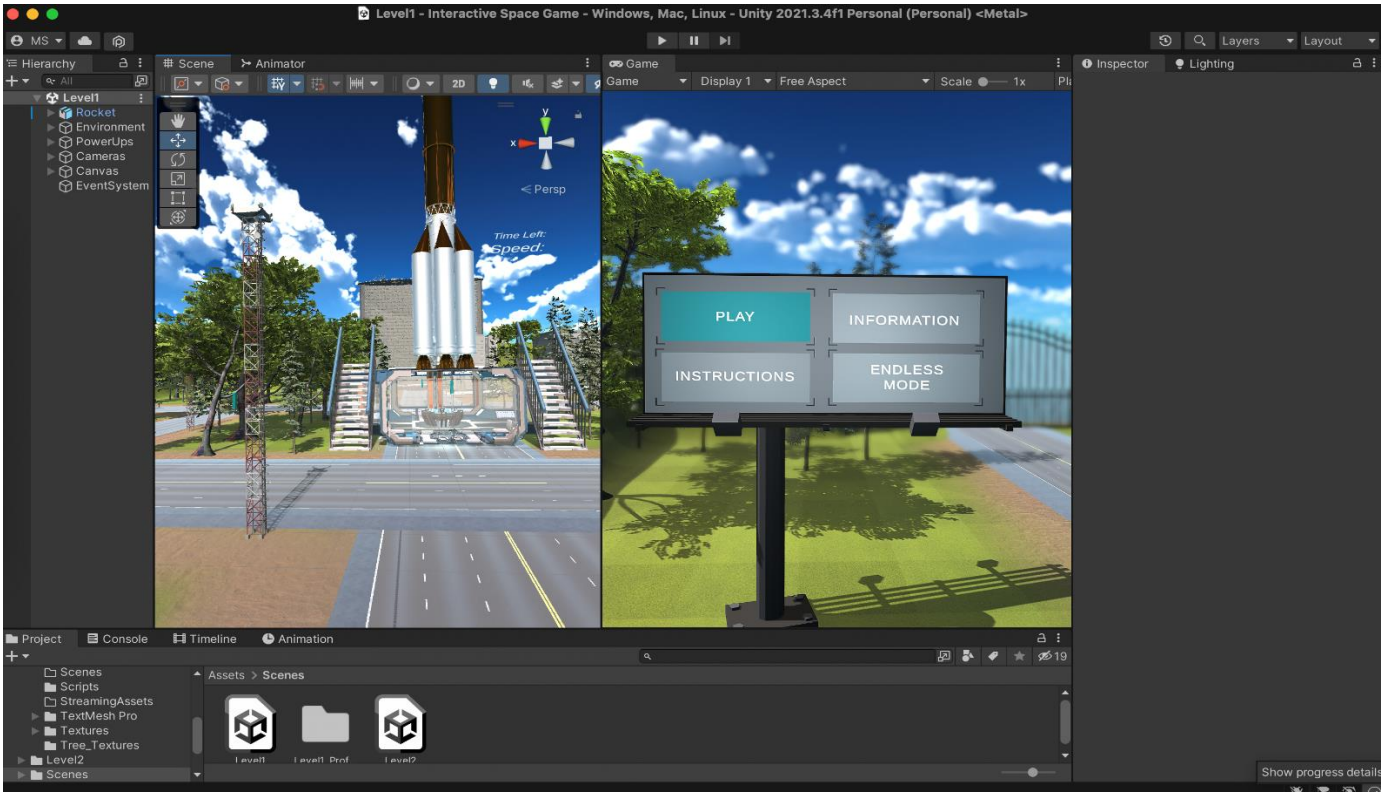
Project description:

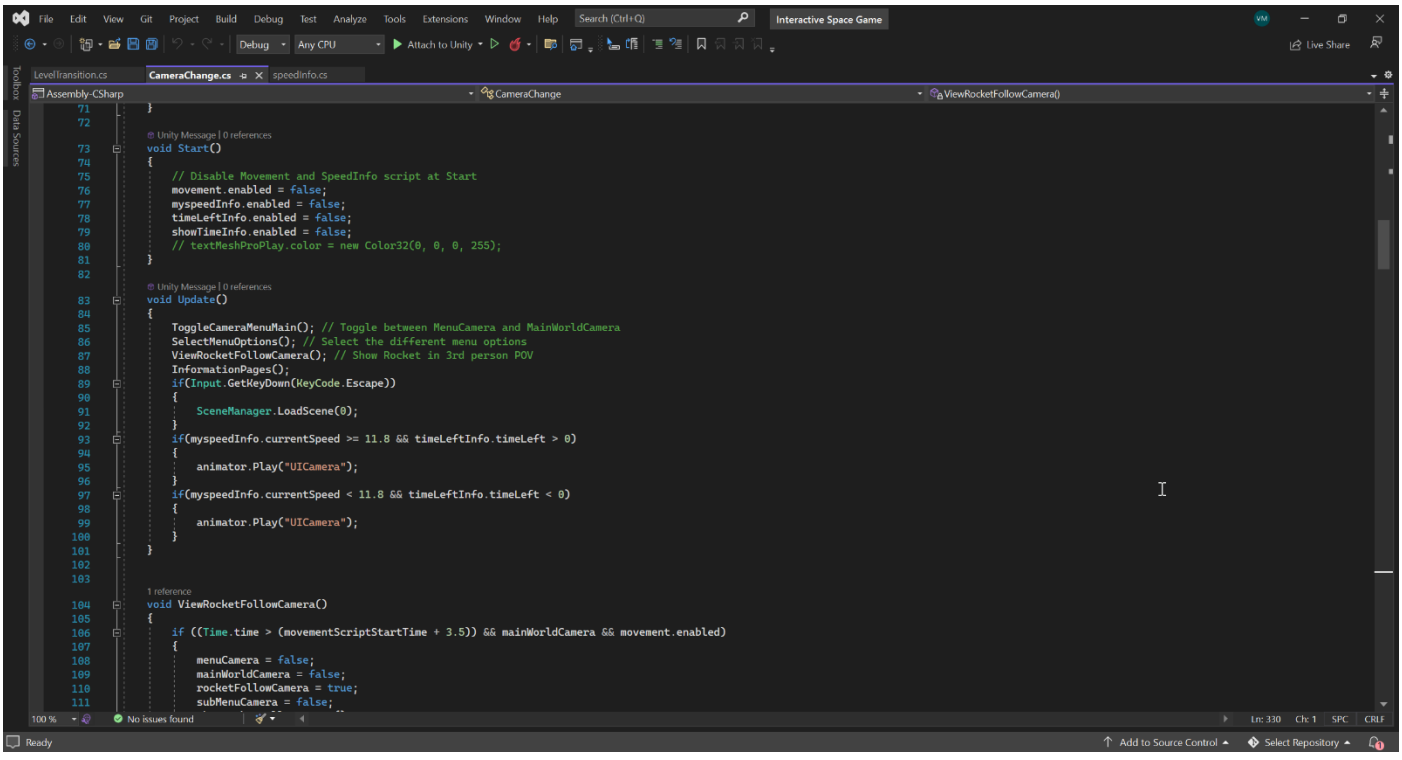
The game intends to capture the essence and provide an insight into the Mars Orbiter Mission. The Mars Orbiter Mission (MOM), also called *Mangalyaan*, has been a space probe orbiting Mars since 24 September 2014. It was launched on 5 November 2013 by the Indian Space Research Organisation (ISRO). It is India's first interplanetary mission, making it the fourth space agency to achieve Mars orbit, making India the first Asian nation to reach Martian orbit and the first nation in the world to do so on its maiden attempt. The game is created on the Unity game engine, which uses C# scripts to animate the game objects. The computer vision aspect will see the use of python and associated libraries as well as solutions like media pipe by google.



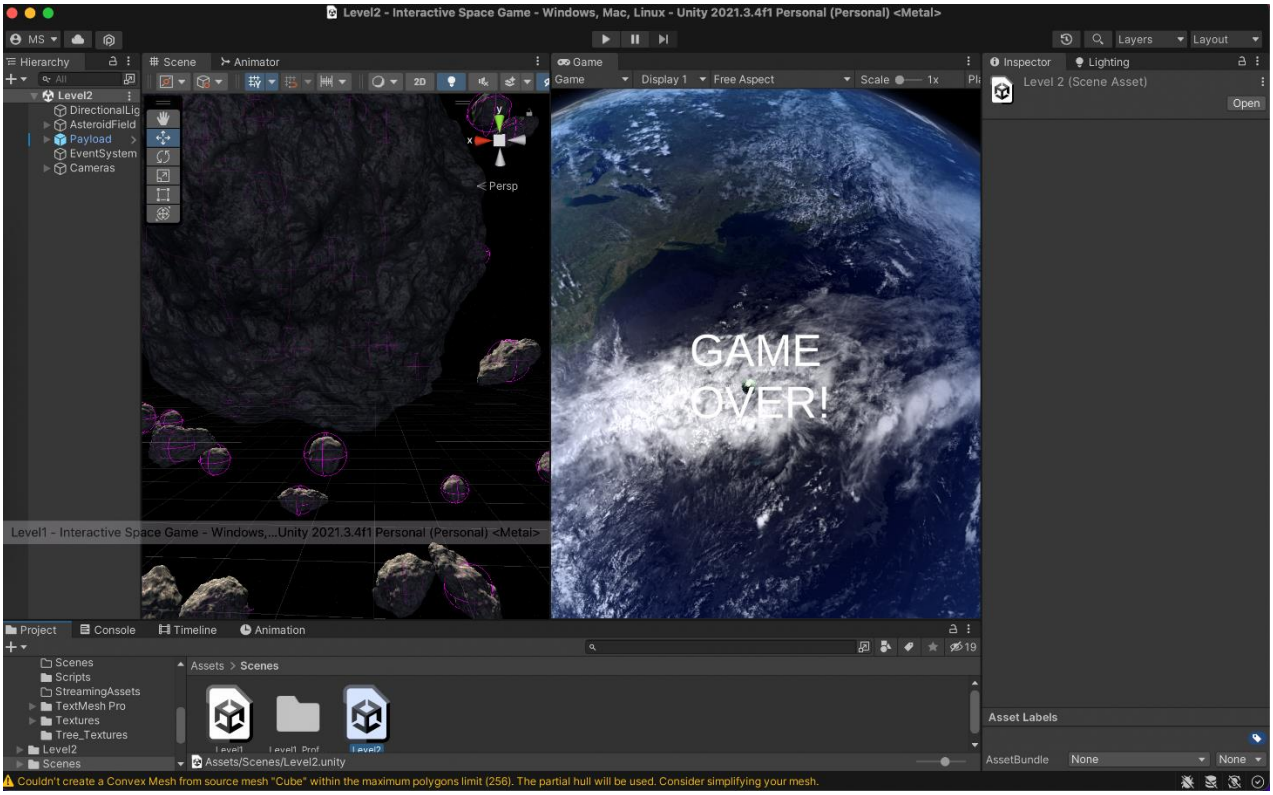
As every space mission begins at a launch site, **Level-1** involves the rocket launch process. Users must maneuver the rocket front, backward or sidewards to collect power-ups to achieve an escape velocity of 11.2 km/s. This level lasts for 90 seconds until the rocket escapes the Stratosphere and enters the Mesosphere of the earth's atmosphere. If the user cannot collect the powerups to reach the required velocity, the game ends, and the player loses. This level was constructed using a combination of self-made and free online assets. We have carefully selected skyboxes and game objects to make it as immersive as possible. The motion of the

rocket, the fire propulsion, and the camera's movement have been programmed using the C# language. Below is a snapshot of the level developed in the unity engine and a sample script.

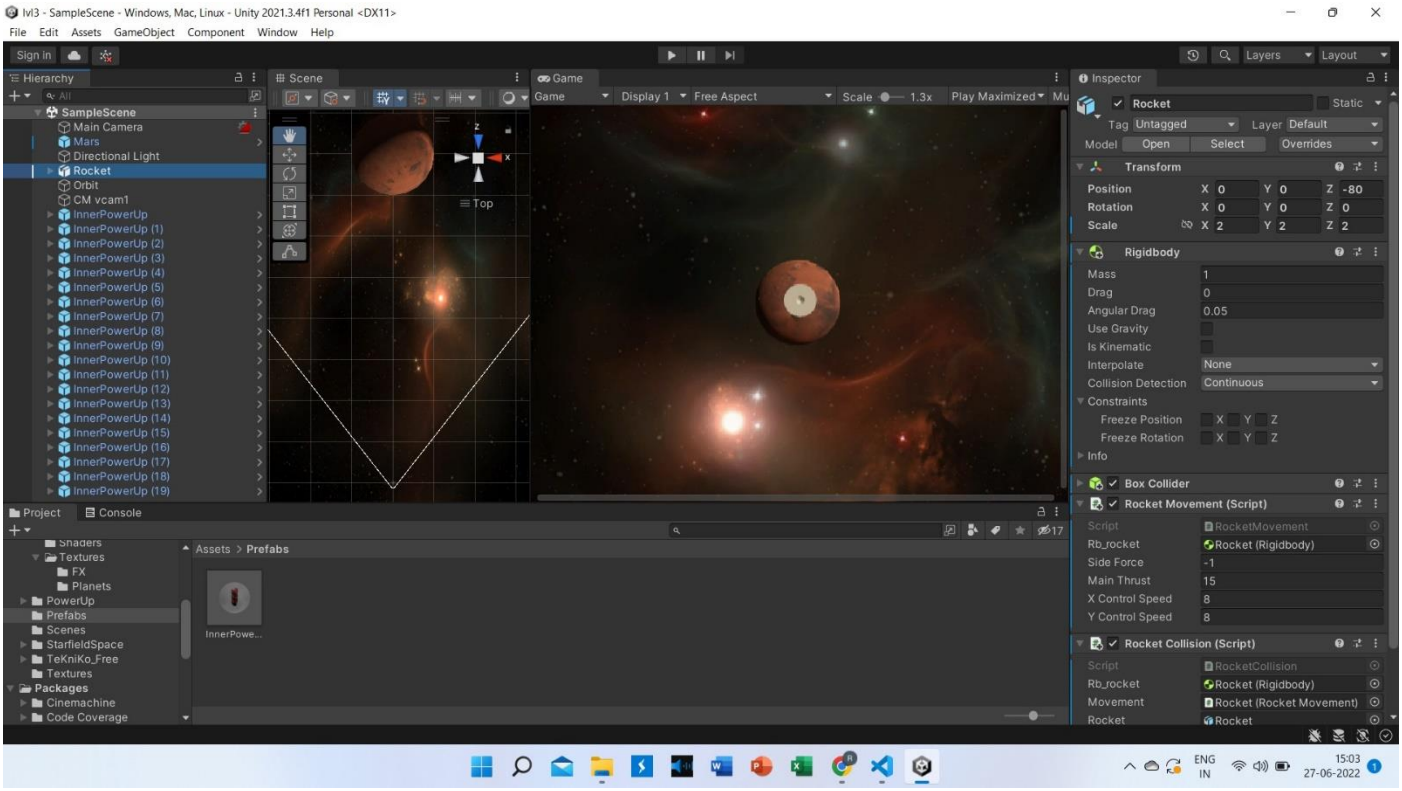




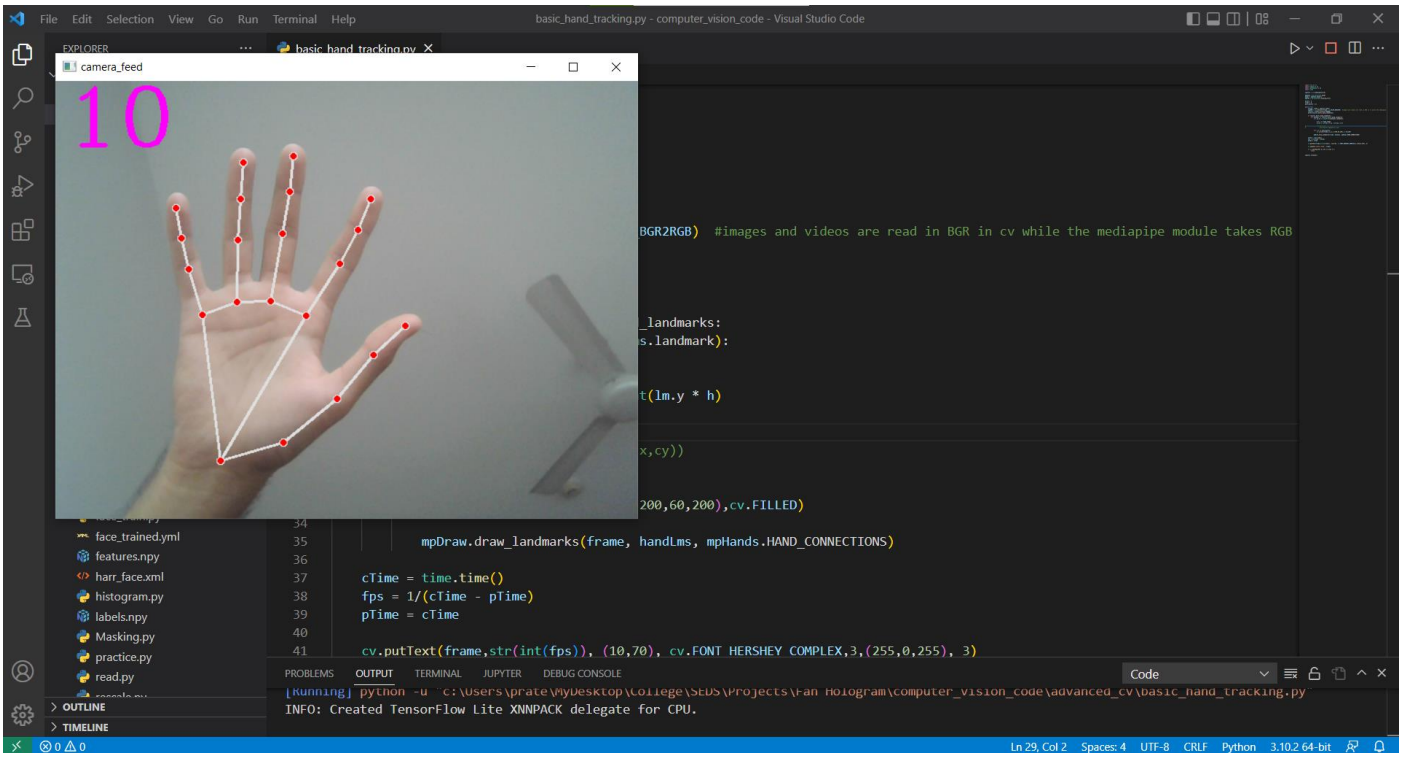
Level 2 begins when the rocket successfully crosses the earth's atmosphere and jettisons its boosters. Here, the player must maneuver the remaining booster and payload to dodge asteroids in the Main Asteroid Belt between Earth and Mars. For each collision with an asteroid, the player sustains 100 units of damage. If the player achieves damage of 1000 units on the damage meter, they are knocked out, and the game ends. This level lasts for 90 seconds. A procedure similar to level 1 is used. However, this level required its own set of custom assets and scripts to get it running successfully.



Level 3 begins once the player has successfully passed through the asteroid belt. As the orbiter hurtles towards the Red planet, it needs it make changes to its path and speed when it finally approaches Mars. This is essential to attain orbital velocity and orientation. To simulate this, this level requires the player to collect power-ups that reduce the speed. They are precisely placed such that when the player collects required the number of power-ups, the speed is sufficiently decreased.



The final step will involve using python and its associated libraries to construct a hand tracking system. The output of the hand tracking system will be fed into the game using unity assets specifically designed for integrations with OpenCV. We plan on using ready-made open-source solutions to help create a more robust and responsive human-computer interface.



TECHNOLOGIES AND SOLUTIONS:

We describe the software and languages actively used to convert this idea into a tangible product.

UNITY GAME ENGINE *(software)*

Unity is a cross-platform game engine developed by Unity Technologies, primarily used to create video games and simulations for computers, consoles, and mobile devices.

The language that's used in Unity is called **C#.** C# is a general-purpose, multi-paradigm programming language. C# encompasses static typing, strong typing, lexically scoped, imperative, declarative, functional, generic, object-oriented, and component-oriented programming disciplines. The engine can be used to create three-dimensional (3D) and two-dimensional (2D) games, interactive simulations and other experiences. The engine Indumachine have adopted the enginedeo gaming, such as film, automotive, architecture, engineering, construction, and the United States Armed Forces.

Python *(language)*

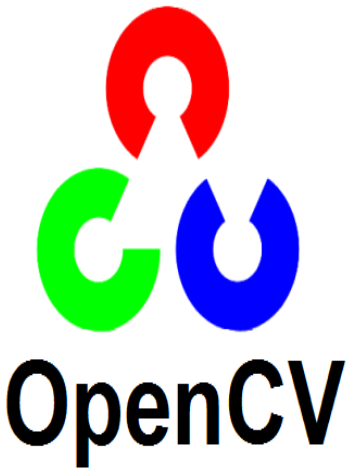
Python is a high-level, interpreted, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically-typed and garbage-collected. Once we understand the basic concepts of Python programming, we can use its libraries to make games with attractive graphics, suitable animation, and sound.

NumPy *(python library)*

NumPy stands for Numerical Python and was created in 2005 by Travis Oliphant and right now it is an open source project.It is Python library for scientific computing in Python. NumPy is used for working with arrays and also in the domain of linear algebra, fourier transform, and matrices. It is written in python, c and c++ programming languages and enhances the performance of arrays by more than 50x. Due to its powerful data structures and efficient calculations it is widely used for data science and machine learning.

OpenCV *(python library)*

OpenCV is a Python library which is designed to solve computer vision problems. OpenCV was originally developed in 1999 by Intel. Right now it is an open source project .It is written in c/c++ and is the most commonly used library for face detection, object tracking and landmark detection.



RESULTS AND TASKS COMPLETED

LEVEL 1:

- The assets for this level were taken from Unity's own assets and some free assets from the Asset Store to integrate the various props (game objects) into the level.
- Some assets that required further detailing were designed using the "Blender" 3-D modeling software.
- The terrain was made using Unity's own "Terrain Tools," and the roads were built using "Road Architec,", a fre,e open-source Unity add-on.
- The in-game light was handled by a single "Directional Light" element. We created globules that, when collected, increase the speed of the rocket. These globules will help the user win the game.

C# Scripts and their functions:

1. **CameraChange.cs:** this script handles all the possible events where a camera transition is required between any two in-game cameras.
2. **LevelTransition.cs:** handles the disabling of the Rocket and Power-Ups at the successful completion of the level.
3. **LevelTransitionUI.cs:** shows the level completion / level failure text.
4. **Movement.cs:** controls how the Rocket moves, handles Collison events, and controls the level audio.
5. **PlayColorChange.cs:** toggles between the colors in the Menu on selecting an option.
6. **SpeedInfo:** shows the live speed of the Rocket.
7. **TimeLeftInfo:** displays the time left for the level.

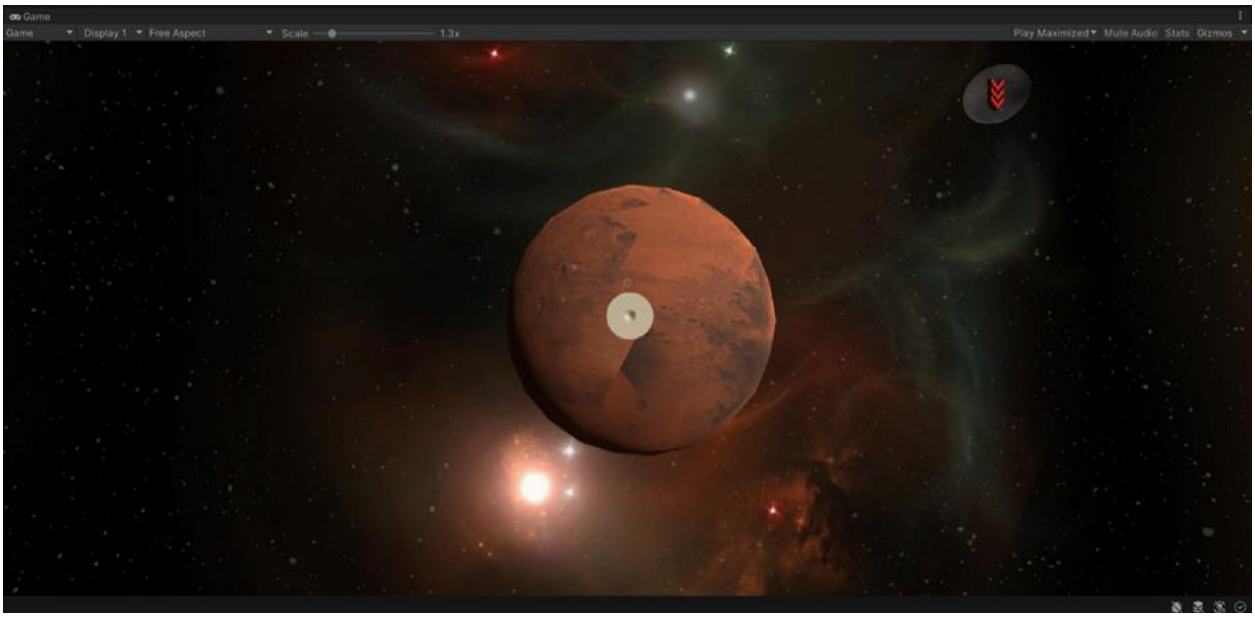
Level 2:

- Level 2 was on the slightly challenging side – a trade-off was faced between fact and function. Not a lot happened between exiting the earth's atmosphere and aligning the orbiter with mars excluding trajectory correction, acceleration, deceleration,n etc.
- So, we came up with a space debris field that symbolizes the cloud of space garbage accumulated over decades since the first space race. This is a real life problem threatening space travel in the future. However rendering space debris like satellite chunks/ used boosters would require a high-performance system and additional software like blender to construct it from scratch. Given the time limit and a realistic approximation of computing power available with the company we used space rocks as the obstacles.
- The implemented level is 60 seconds long, and the user is shown a damage meter beside the payload and booster. The damage count increases by 100 per collision, and when it reaches its maximum damage of 1000, the payload bursts player fails the level (also the case when the timer runs out).
- To create a more realistic effect, we implemented a script that randomly spawns asteroids in front of the user, The asteroids even rotate about their axis with very minute angular velocity.
- Finally we used the cinemachine package to create a smooth cinema-like the transition at the beginning of level2.

C# Scripts and their fucntions:

1. **CamerSwitch.cs** - switches between cameras based on different states
2. **Controls.cs** - controls how the payload moves.
3. **Movement.cs** - controls speed of the rocket and audio.
4. **OnCollision.cs** - handles all the collision events of the payload (with any asteroid or other space debris). It also handles the damage meter count.
5. **TimeLeft.cs** - shows the current time left.

Level-3



- This level renders of Mars along with the surrounding environment and its Aerocentric Orbit.
- We have scattered “Speed-downs” scattered the path of the Orbiter that decrease the speed of the Orbiter. This was used to simulate the trajectory correction as well as velocity changes necessary to obtain the orbital velocity of mars.
- We have implemented Three “State Driven Cameras” that activate when different states are called, showing different perspectives to the user.
- A Cinemachine “Main Camera” that acts as the central controlling camera for the state driven cameras along with a “Virtual Follow Camera” that follows the Orbiter as it moves is also implemented.

C# Scripts and their functions:

- **RocketCollision.cs:** specifies function calls for collision events of the Orbiter.
- **RocketMovement.cs:** controls how the Orbiter moves about.

FUTURE OBJECTIVES

A Space shooter mini-game:

We are yet to develop a “mini-game” which would be accessible to the player from the Game Menu at the start of the game.

The idea is to provide the user with a somewhat “fun” experience within the game so that those looking to just play around with, manoeuvring a Spaceship of sorts can do so.

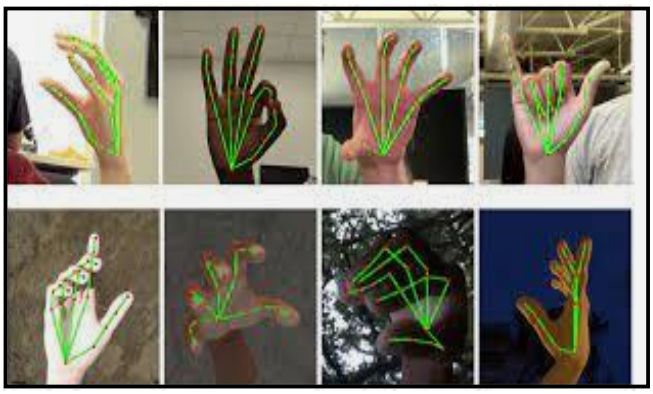
Layout and ideology of the mini-game planned:
The player would take control of a spaceship on a distant planet that has been invaded by enemy ships.
The player’s objective will be to move around the spaceship (not thrusting) and shoot down enemy personnel.
The player will be awarded a” score” which will also be updated live in the game which will be proportional to the number of enemy spaceships the player shoots down.

Hand-tracking model:

We are also to develop a hand-detection system model using the OpenCV library of Computer Vision and Python as the programming language.

The hand-detection system is currently prototyped to plot about 23 trackable “landmarks” on both the palms of the player which will return the real time position of the calculated palm centres.

These returned position values are to be integrated into Unity which, in turn, would simulate the movement of the Player model in the game in all the levels and the mini-game.



REFERENCES

https://www.scribbr.com/dissertation/introduction-structure/

https://ieeexplore.ieee.org/document/5219910

https://en.wikipedia.org/wiki/Mars_Orbiter_Mission#Mission_profile

https://tnstc.gov.in/

https://www.youtube.com/playlist?list=PLPV2KyIb3jR4CtEelGPsmPzlvP7ISPYzR

https://www.youtube.com/playlist?list=PLPV2KyIb3jR53Jce9hP7G5xC4O9AgnOuL

THANK YOU