Faculty of Science, Engineering, and Computing
School of Computer Science and Mathematics
Kingston University London

CI7500 - Assignment 02: Game/App Development

MSc GAME DEVELOPMENT(PROGRAMMING)

The Starship Simulator For Playstation 5

Prajwal Shetty Vijaykumar - K2371155

Supervisor: Prof. Vasileios Argyriou

Email: vasileios.argyriou@kingston.ac.uk

Release date: 2nd December 2024

Table Of Contents:

Technology, Algorithms and Tools:

The physics:

This project is a **true to scale simulation**, which attempts to represent the planet, rocket and their forces in close to real world scale and values.

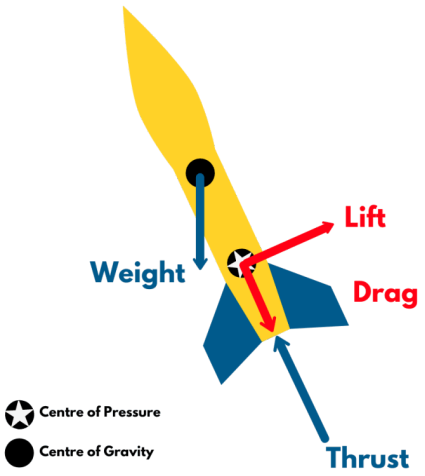
It uses Unreal's "Large World Coordinates" , which is now a default for v5.2+, where all the 3D math objects are now double-precision floats, and the default **WORLD_MAX size in Unreal is now 88 million kilometers** (Epic Games Developer, 2024), which easily enables planet scale simulations and with some optimisations can also enable inter planet simulations. But to set the scope to an achievable target with a limited time of under 1.5 months, this project only aims to simulate the starship's maneuvers at the scale of Mars.

The project also has the real martian surface of 144.4 million km² at its true scale based on the satellite data of the planet's high quality relief and height maps that are widely available to use from many research institutes. But the level of detail of the surface is extremely low as that is clearly out of the current project's scope and also requires many hours of optimisation work to render the planet at a higher quality efficiently. Nevertheless it's good to notice the potential of modern game and simulation tech and how comparatively easier it is now to simulate complex physics better.

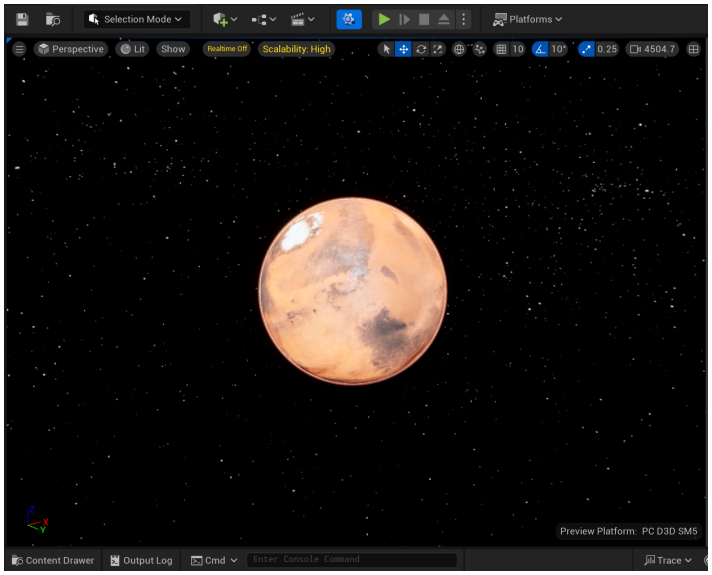
Torque, Gravity, Lift and Drag equations were used with minor modifications for the rocket behavior in Space and Martian Atmosphere.

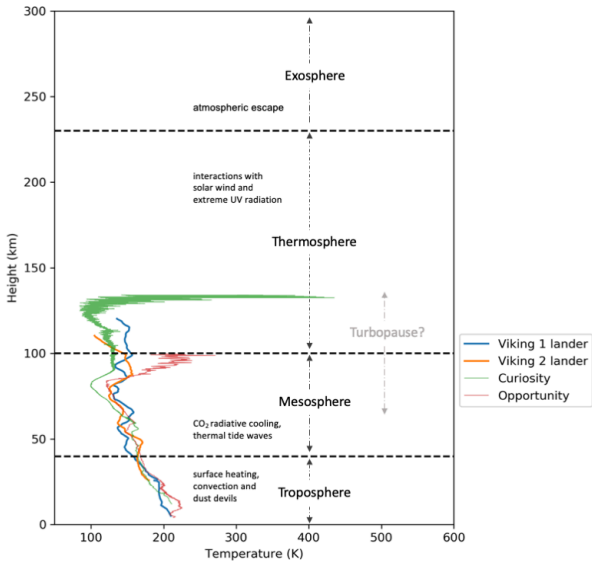
$$\tau = I\alpha. \qquad F = G\frac{m_1 m_2}{r^2}, \qquad L = C_l \cdot \frac{\rho \cdot V^2 \cdot A}{2} \qquad D = C_d \cdot \frac{\rho \cdot V^2 \cdot A}{2}$$

These forces are modeled to replicate real-world physics as accurately as possible. Initially the project targeted full accuracy, but eventually to keep the scope small and game fun there had to be some tweaks to the equations later on. More of the changes will be explained in the creative liberty section below.



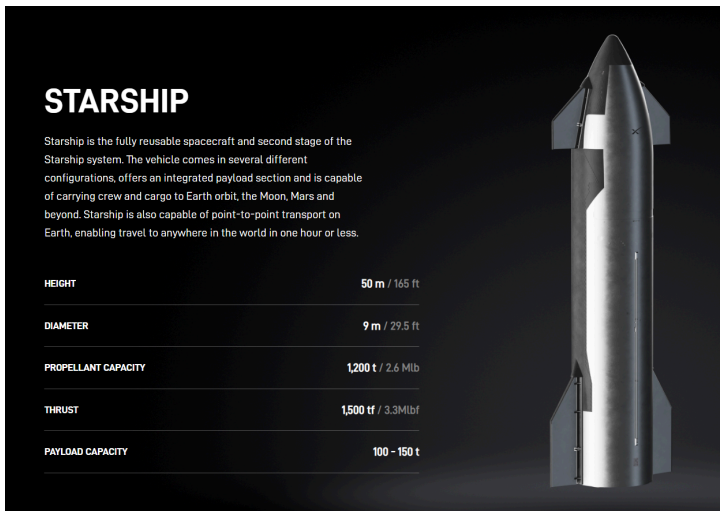
Recreating Mars in Unreal Engine 5:





Mars has a **planetary radius of 3,389.5 km**, which is approximately 338,900,000 unreal units, (Wikipedia Contributors, 2019) with the atmospheric pressure of 0.636 kPa i.e 0.00628 atm, 0.6% of Earth's 101.3 kPa (14.69 psi) (Xue et al., 2022) and a **surface gravity of 3.72076 m/s2** (0.3794 g0) and escape velocity of 5.027 km/s. The atmosphere implementation currently is very basic and doesn't fully replicate the martian atmosphere. However, close to actual Martian atmospheric drag values were used during rocket descent. In **Unreal Force is in cm/s², Distance is in cm, Mass is in Kilograms Angle is in Degrees and Force is in Newtons**. It was straightforward to convert any real world date to unreal metrics.

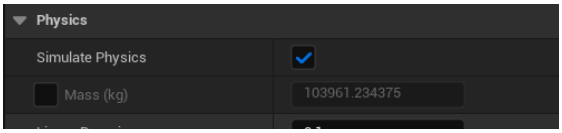
Starship:



Starship **(without the super heavy booster)** is ~50m tall, and 9 meters in diameter, with the Propellant Capacity of ~936,000kgs of cryogenic liquid oxygen and ~264,000kgs liquid methane. (SpaceX, 2024)

The **body weight and size** of the rocket was **exactly replicated** using the Unreal's physics material properties by modifying its density and other values. The thrust generated (1,500 tf / 3.3Mlbf) by the rocket was exactly replicated too based on real world numbers provided in the internet from SpaceX but later on decided to give more thrust to the rocket just to make the game more fun to play with.

There is no proper center of Mass values available on the starship from the internet so for the project it's the bounding center of its volume. Theoretically the center of gravity of a rocket will slowly move to the bottom of the rocket as it burns more fuel, as the engines are heavier compared to empty tanks. The game does **implement the change in mass of the rocket as it burns more fuel,** but it doesn't change the center of Mass. (SpaceX, 2024)

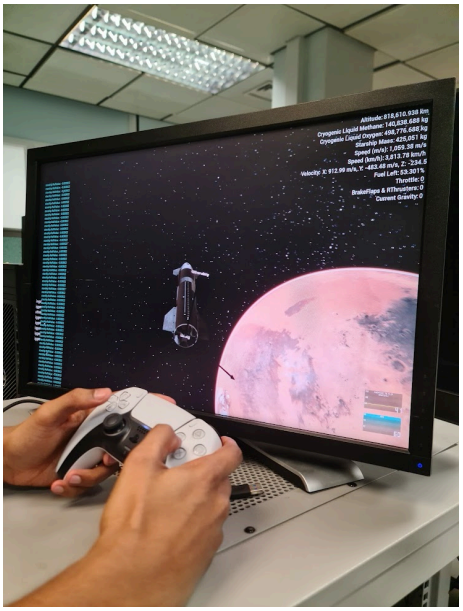


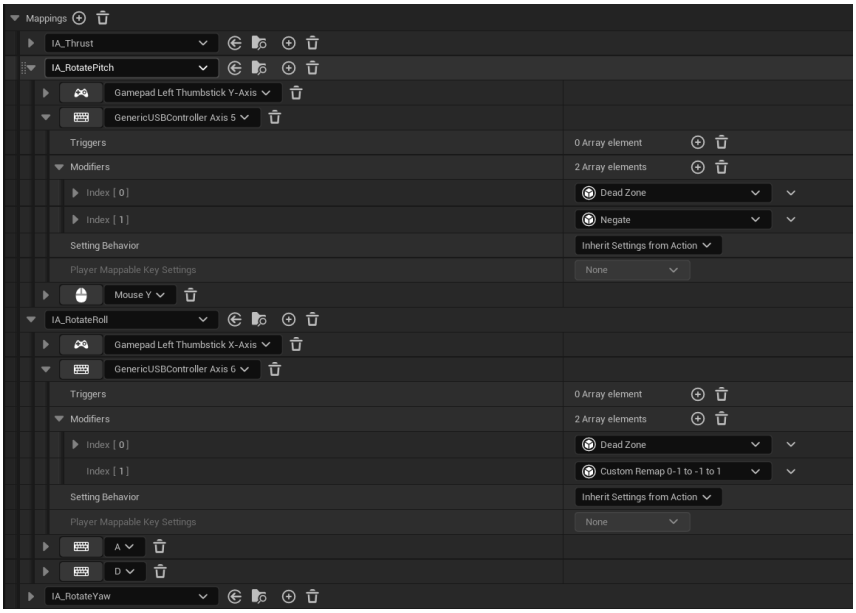
PS5 Devkit and Unreal:

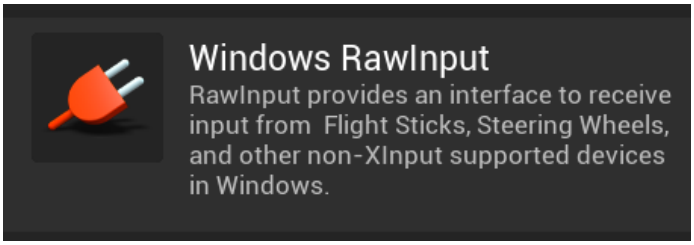
Input:

With Unreal's new Enhanced Input System (Epic Games Developer, 2024) the project can handle input from **four different devices simultaneously** for two different platforms. The image below shows part of the InputMappingContext file where Gamepad represents the actual DualSense Controller input in PlayStation 5 builds only, and GenericUSBContoller represents the DualSense input in windows builds and editors which is generated by the Unreal's Raw Input Plugin.

The DualSense is also supported in the windows build except for any code that uses the sony's proprietary API. Keyboard and mouse support in windows are very basic and used during the development phase mostly so not recommended for testing.



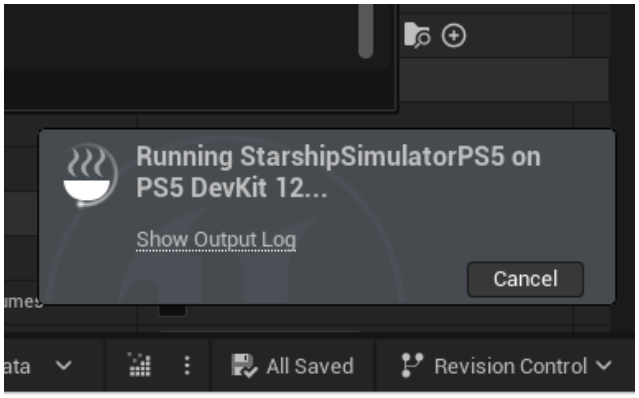


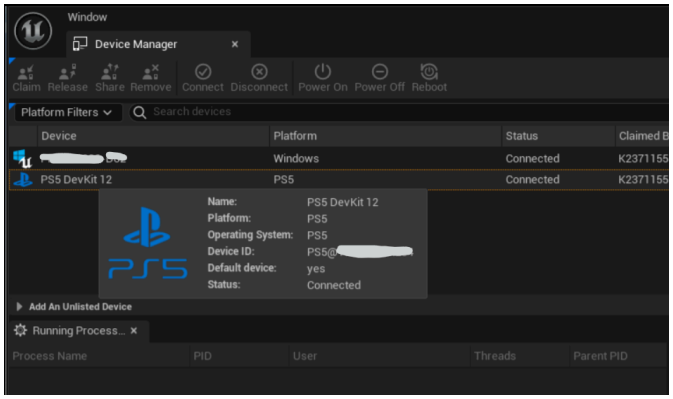


Building and Testing in PlayStation 5:

Testing the Starship Simulator on PlayStation 5 required the use of the PS5 devkit in combination with Unreal Engine 5.3. The project also briefly explores working with DualSense's API and Playstation API with platform specific code. There were initial attempts to integrate the DualSense's Gyro for rocket control but it only made the control's more complicated and had to be let go in a later phase of development.

The PlayStation dev kit also gave profiling and debug tools, which played an important role assessing the real-time performance of the project in the PS5 environment. The build process was slow but straightforward for most of the process except some debugging cases where it took multiple back and forths between Building and changing implementations. All playstation related API's could only be tested in the deployed build so testing them was time consuming.





Source code analysis:

The majority of the code base (about 95%+) is written in C++, including most of the UI update logic. Some intro to the files:

- **Starship Control:**
  - StarshipPawn.h defines the player-controlled pawn within Unreal, giving players direct control over the Starship; this handles the input and stores data references.
  - StarshipController.h, SInitHelpers aids the pawn script for additional logic like constructor Init and etc.
- **Physics:**

    - StarshipPhysics.h contains the implementation of the physics calculations, including forces for rocket movement, gravity etc.. This file plays a pivotal role in simulating the dynamics of a Martian landing in a realistic manner.
- **Data:**
    - FStarshipConfig.h, SCustomModifiers.h, StarshipInputConfig.h handles configuration settings for the Starship and user input, providing the underlying data structures necessary for customization.
- **UI:**
    - SGameHUD.h manages the in-game HUD giving players critical information regarding their Starship's status, mission progress, and environmental conditions.
- **Utility:**
    - SDualSense.h integrates the PlayStation 5 DualSense controller's unique functionalities, including adaptive triggers and haptic feedback.
    - SGlobalDefinitions.h and SInitHelper.h provide utility functions and global variables used across different components. SGlobalDefinitions also has a timer setup for other scripts to measure compute time.
    - StarshipVFXComponent.h manages the visual effects (VFX) associated with the Starship, such as engine thrusters, audio and other visual elements crucial for immersion.
- **Build Configuration:**
    - StarshipSimulatorPS5.Build.cs and StarshipSimulatorPS5Editor.Target.cs define the build configurations, ensuring the project compiles correctly for PlayStation 5, Windows and the Unreal Editor.
- **Main Game Files:**
    - StarshipSimulatorPS5.cpp, StarshipSimulatorPS5.h, and the game mode files (StarshipGameModeBase.cpp/.h) provide the basic entry points for game logic. These files tie together all other components to define the overall flow of the game.

Aims and Objectives:

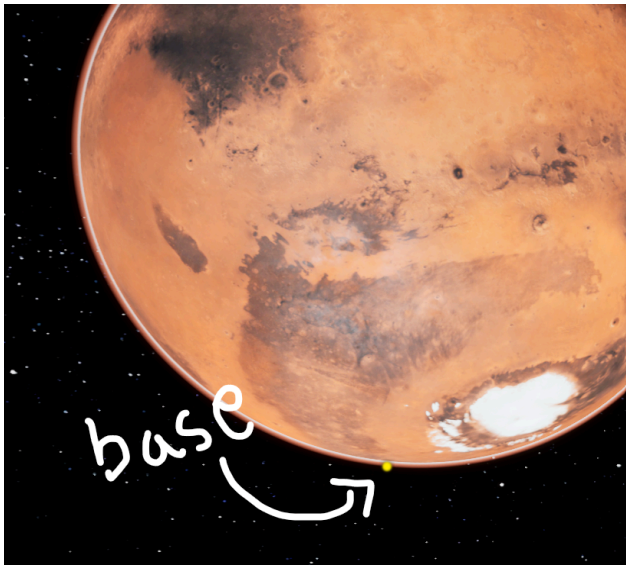
The primary goal of the Starship Simulator is to recreate the experience of navigating a spacecraft in Space and Martian conditions. The focus is on leveraging accurate physics and real-world data to provide an immersive experience. The project aims to achieve:

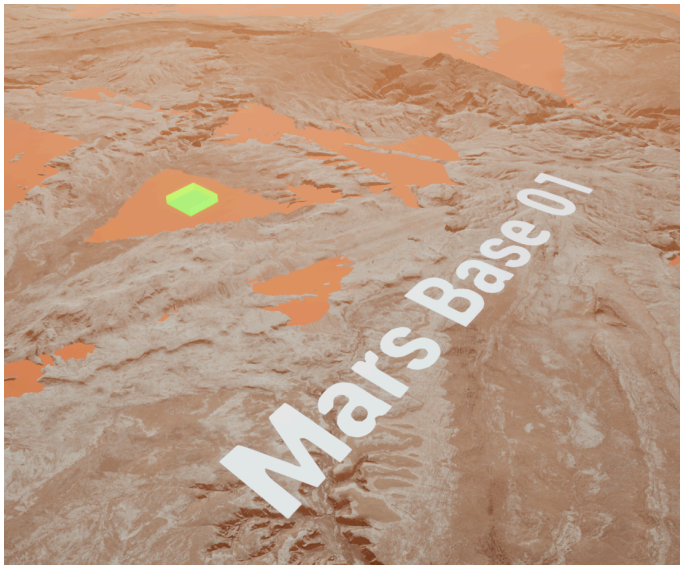
1. Exploring the crux of Rocket Physics!
2. Exploring the Playstation-5 development pipeline and Input.
3. True-to-scale planetary and vehicle simulation.
4. Realistic rendering of Mars and atmospheric conditions.

Plot/Goal:

So the game starts with starship around 7000 km away from Mars, The player gets an intro message that states the mission statement:

“Your mission is to land safely at the Martian Base 1 near the south pole of MARS. Landing on a low gravity planet like Mars requires complex maneuvers. There are no brakes in space, and as you get closer to the planet the Mars Gravity will start pulling the ship towards it. Spin the rocket around to use the main engine thrust to reverse. The boost button gives you 1000x the regular starship speed but also makes it incredibly hard to land as you might be going too fast to properly slow down near the landing area, but without the boost it takes longer to travel ~7000 kms to the surface. Follow the yellow marker on the planet to reach the base.“





Novel Contributions:

- One of the significant contributions is attempting a true-to-scale simulation within Unreal Engine, which is rare in gaming simulations due to the complexity and computational challenges.
- Fusing Unreal, Playstation 5 and Simulation together.

Creative Liberty:

Some changes were done half a through the project to keep the scope small and make the game more engaging, if all the numbers like thrust and fuel consumption are fully real, it would take hours and hours of gameplay for the mission to finish. To keep the game engaging and fast the rocket thrusts were increased 10x while decreasing its fuel consumption, the project also has a boost mode which gives an unrealistic 1000x more thrust on top of the 10x boost already all while keeping fuel usage at minimum.

The game also starts at ~7000 km from mars, so **realistically it should start at under 10% fuel** as the journey was 50 million kilometers long, **but it starts with 100%.** Also equations like gravity and torque are simplified. Mars' **atmosphere is only 11km**, in game it is **bumped up to 110 kms.**

Tech Stack:

- Unreal Engine 5.4
    - Windows Raw Input
    - Enhanced Input
- Rider 2024.1.6
- Audacity
- Playstation 5 Devkit

Game Evaluation with Qualitative and Quantitative Results

Qualitative Analysis

The qualitative analysis of the Starship Simulator focused on user experience, gameplay immersion, and feedback from playtesters, mostly my friends who had an interest in space. The simulation's realistic physics and integration of PlayStation 5's DualSense controller features were praised for adding to the game's immersion but also mentioned the complexity of maneuvering the ship during re-entry into the planet. Playtesters specifically highlighted the following aspects:

- **Immersion through Tactile Feedback**: The DualSenses adaptive triggers, thrust effect and audio feedback allowed players to feel the strain of the rocket's engines and the atmospheric pressure changes. This feature significantly enhanced the sensory experience of the gameplay, making the player feel more connected to the mission.
- **Atmospheric Visuals**: The visuals, combined with the realistic atmospheric effects, helped to create a highly believable Martian environment.
- **Learning Curve**: Most people noted the steep learning curve for the controls due to the realistic physics. This aspect was perceived both positively, as it added to the realism, and negatively, as it made the game less accessible to casual players. Future iterations might benefit from adding tutorial levels or simplifying certain mechanics, and unlocking the camera orbiting.
- **Engagement and Challenge**: Players found the core gameplay of navigating and landing on Mars engaging, particularly the challenges of managing fuel and landing. The difficulty level was seen as appropriate for a simulator, appealing to players who enjoy methodical problem-solving rather than fast-paced action.

Quantitative Analysis

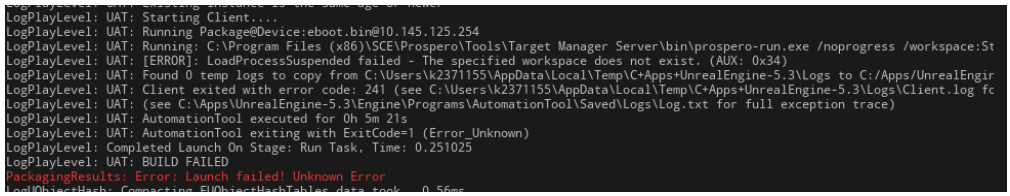
The quantitative analysis involved metrics gathered through playtesting sessions and performance evaluations:

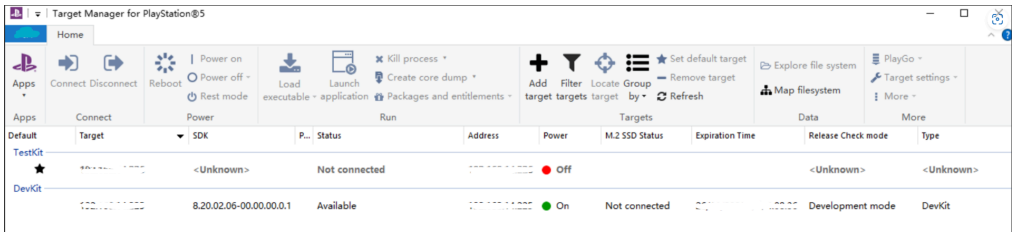
- **Frame Rate Consistency**: The game achieved an average frame rate of **60 fps** on the PlayStation 5, with rarely dropping to **55 fps**. This is because of the lack of details in planet surface and dust particles, which is actually a good thing for a smaller simulation project like this one.
- **Performance Metrics**: The CPU and GPU load was monitored using Unreal Engine's profiling tools.
- **User Playtesting Scores**: Playtesters were asked to rate various aspects of the game on a scale of 1 to 5:
    - **Immersion**: **5/5**
    - **Control Responsiveness**: **4/5**
    - **Visual Realism**: **4.5/5**
    - **Difficulty**: **2/5** (with some players finding it challenging but rewarding)
    - **Overall Satisfaction**: **4.4/5**
- **Average Playtime**: The average playtime recorded during playtesting sessions was **4.5 minutes.** Many playtesters also tried to land the rocket multiple times as they crashed every time.

Performance and Optimization:

Playstation Build Issues:

There were issues related to PS5 builds mostly due to build caching in the devkit's side. The Target Manager and Workspace Explorer apps for PS5 in windows helped to solve the problem.

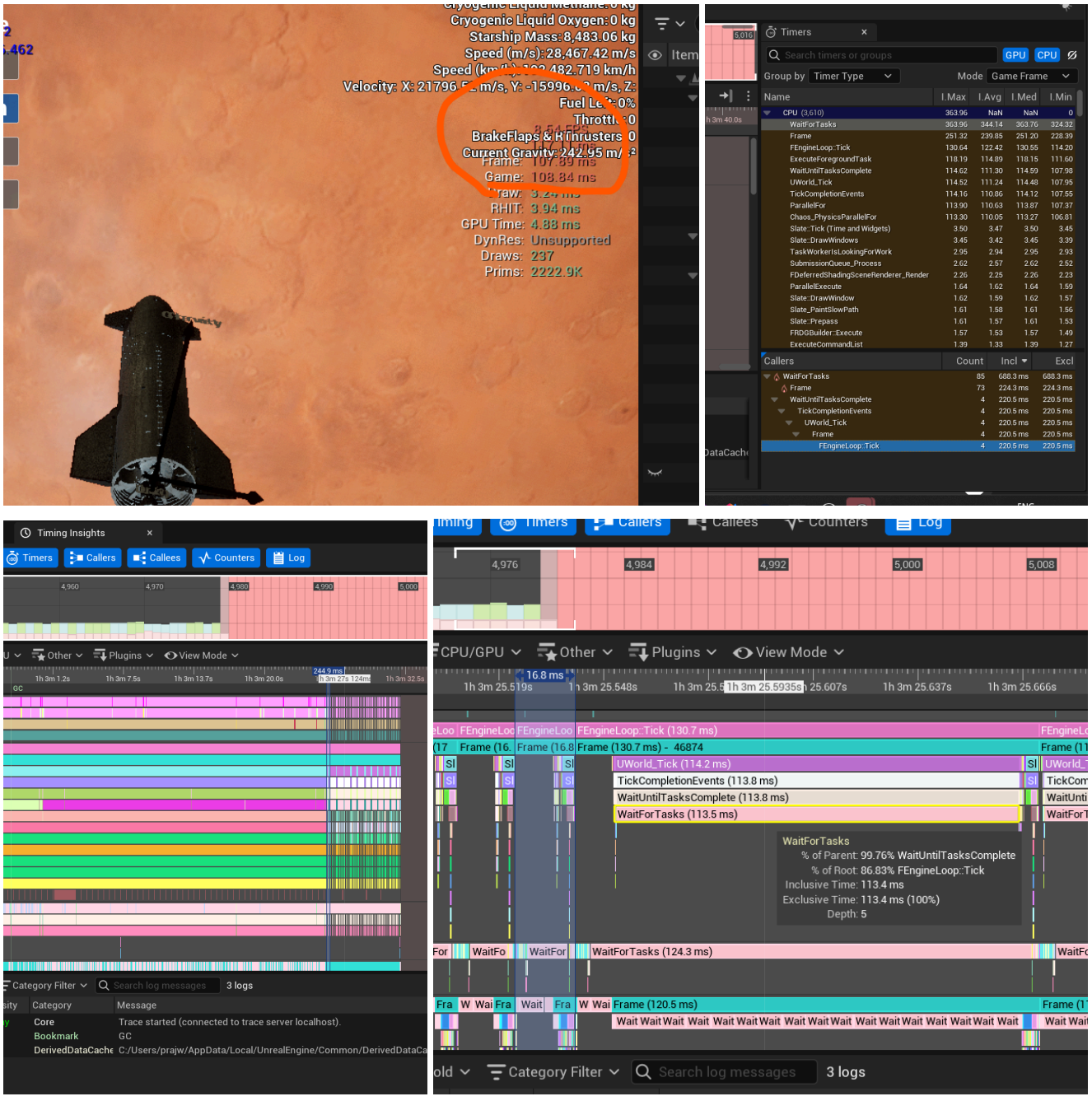




Fixing Performance Issues:

The project did run into performance bottlenecks when the ship got closer to the planet, it was quickly sorted by the help of unreal insights tool, which in turn pinpointed to the root cause.

In the project's case it was the physics collision calculations which was the bottleneck, was optimised by using simpler checks and replacing complex collisions with primitives.



Gen-AI:

There was a plan to integrate TARS-like AI from interstellar to the game, but given the current complexity of the project the idea was scrapped mid-way to focus on the physics simulation and ps5 part of the game. Nevertheless the idea was to give the LLM API constant rocket number the fuel altitude etc, and it will keep informing the user on any potential issues, like if they are going too fast, too low or about to run of fuel or even a space joke just like the movies!

Legal:

Here's a quick update on the legalities of the current project, as it uses some proprietary tools and trademarks. There is no intention to publish the project in its current state without removing any code or content that includes unlicensed names or code covered under NDAs.

Disclaimer:

- The project uses Sony's PlayStation API for DualSense-specific features which cannot be published in any public forums or repositories.
- The project is built for PlayStation 5 Dev Kits and does not have permission to be released on the consumer version of PlayStation 5.
- The report briefly mentions Interstellar's TARS-AI, to which it does not hold any rights.
- The project uses a 3D model of SpaceX's Starship and the name "Starship," neither of which are licensed for use.

References:

- Epic Games Developer. (2024). Georeferencing a Level in Unreal Engine | Unreal Engine 5.4 Documentation | Epic Developer Community. [online] Available at: https://dev.epicgames.com/documentation/en-us/unreal-engine/georeferencing-a-level-in-unreal-engine?application_version=5.4 [Accessed 27 Nov. 2024].
- Wikipedia Contributors (2019). Mars. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Mars.
- Epic Games Developer. (2024). Large World Coordinates in Unreal Engine 5 | Unreal Engine 5.0 Documentation | Epic Developer Community. [online] Available at: https://dev.epicgames.com/documentation/en-us/unreal-engine/large-world-coordinates-in-unreal-engine-5?application_version=5.0 [Accessed 27 Nov. 2024].
- Xue, X., Jia, H., Rong, W., Wang, Q. and Wen, C. (2022). Effect of Martian atmosphere on aerodynamic performance of supersonic parachute two-body systems. Chinese Journal of Aeronautics, [online] 35(4), pp.45–54. doi:https://doi.org/10.1016/j.cja.2021.05.006.

- Unreal Engine (2020). Learn How to Work With Geospatial Data in Unreal Engine | Webinar. [online] YouTube. Available at: https://www.youtube.com/watch?v=RKyyuAhnqP4 [Accessed 27 Nov. 2024].
- techarthub. (2024). Scale and Measurement Inside Unreal Engine - techarthub. [online] Available at: https://techarthub.com/scale-and-measurement-inside-unreal-engine/.
- SpaceX (2024). Starship. [online] SpaceX. Available at: https://www.spacex.com/vehicles/starship/.
- Epic Games Developer. (2024). Enhanced Input in Unreal Engine | Unreal Engine 5.5 Documentation | Epic Developer Community. [online] Available at: https://dev.epicgames.com/documentation/en-us/unreal-engine/enhanced-input-in-unreal-engine
- Desisto, A. (2021). Starship and its Belly Flop Maneuver. [online] Everyday Astronaut. Available at: https://everydayastronaut.com/starships-belly-flop-maneuver/.