Senior Design Final Report

JPL VR TREK

Version 1.0 - 05/06/2018

Team Members:
Ari Jasko, Ayush Singh, Bryan Lopez, Enrique Guardado, Fabio Carrasco, Ly Jacky Nhiayi, Justin Vuong, Lucca Andrade, Rizwan Vazifdar, Ruben Heredia

Faculty Advisor:
Dr. David Krum

Liaisons:
Eddie Arevalo, Richard Kim, Emily Law

Table of Contents

1. Introduction:

1.1. Background:

Established in the 1930s, Jet Propulsion Laboratory (JPL) has played a crucial role in NASA's space exploration missions. JPL is primarily focused on the development of space exploration technology and unmanned spacecraft missions. It has a rich history of significant contributions to the field of space exploration, including the development of the Mars Exploration Rover Mission, the Galileo mission to Jupiter, and the Voyager mission to the outer planets of our solar system.

Recently, JPL has partnered with California State University Los Angeles (CSULA) to create an innovative new project called the JPL Trek VR Room. This application is designed to provide users with an immersive and interactive experience of the celestial bodies in our solar system. By accessing scientific data from JPL's extensive database, users can explore and interact with these objects in a completely new and innovative way.

Using open standards for VR headsets, the JPL Trek VR Room provides users with an experience that is not only educational but also highly engaging. Through the use of VR technology, users can gain ground-level perspectives of the celestial bodies and a 3D stereoscopic view that enhances their understanding and visualization of the unique features of each object. The project represents an exciting new direction in space exploration and education, bringing the wonders of the universe directly to users' fingertips.

1.2. Design Principles:

The design principles behind the JPL Trek VR Room project are centered around creating an immersive and interactive experience for users that allows them to explore and learn about the celestial bodies in our solar system.

The JPL Trek VR Room provides users with a 3D stereoscopic view of the celestial bodies, which enhances their understanding and visualization of the unique features of each object. This level of detail provides users with a more comprehensive and detailed understanding of the objects they are exploring. The application also utilizes open standards for VR headsets, which ensures that the project is accessible to a wide range of users. This means that anyone with a VR headset that supports these standards can use the application, regardless of the model of their device. Additionally, the project incorporates scientific data from the Jet Propulsion Laboratory database, which provides users with an accurate and up-to-date representation of the celestial bodies they are exploring. This data is displayed in a visually comprehensive and interactive way that allows users to learn about these objects in a completely innovative way. Lastly, the project has a strong educational focus, providing users with an opportunity to learn about the insights behind the celestial bodies in our solar system. This makes it an excellent resource for students, teachers, scientists and anyone who is interested in space exploration and astronomy.

1.3. Design Benefits:

Incorporating design principles to achieve simplicity and efficiency, the latest iteration of VR Trek draws upon multiple user interface designs from previous project builds. These principles have been thoughtfully applied to both the main room environment and the OpenXR controller hand models.

The main room environment has been crafted with a focus on simplicity, utilizing carefully selected lighting and materials that complement one another to create an aesthetically pleasing and easily comprehensible game scene. Given that the primary focus of the scene is the globe, the use of simple and complementary assets is crucial in ensuring that the user's attention is not diverted from the game's central theme.

Similarly, the OpenXR controller hand models have been designed with a focus on simplicity and comprehensibility. The hand models serve as a tracked representation of the user's controller location, with a simple hand posture and blue material employed to enhance their clarity in all situations.

Overall, the latest version of VR Trek serves as a testament to the application of design principles to create a user interface that is both visually appealing and highly functional.

1.4. Achievements:

Our team has achieved several critical milestones throughout the project. We have successfully reconstructed multiple user interfaces, developed new C# scripts, and established API connections.

Our first significant accomplishment was the precise reconstruction of the user interface. We encountered several challenges, such as dealing with diverse user interfaces and applying a shader that complies with universal standards to the Unity project. However, we overcame these obstacles by starting from scratch with an empty VR project instead of importing the previous assets with the Universal Render Pipeline.

The second accomplishment was the creation of new C# scripts. While creating these scripts, we faced several challenges. Our biggest challenge was avoiding compiler errors when applying OpenXR assets/scripts to the project files. However, we overcame this challenge by ensuring that the C# scripts conformed to the project by removing any inconsistencies.

Lastly, we successfully achieved our goal of integrating the JPL API into the project. However, we encountered a challenge when attempting to implement both OpenXR and the JPL API simultaneously while updating the assets. To overcome this obstacle, we created game objects separately and utilized a script that translates SteamVR functions to ensure compatibility with OpenXR.

2. Related Technologies:

2.1. Existing Solutions:

Our main objective when updating VR Trek was to incorporate OpenXR into the previous project build, which was written in SteamVR and had to be redone for compatibility with OpenXR. OpenXR is an open API standard that works across a wide range of VR devices, ensuring high cross compatibility. However, rewriting the code was a major challenge due to a severe lack of documentation, and we had to spend a significant amount of time understanding the original SteamVR app.

We tested the VR Trek app using the Oculus MetaQuest 2 headset and hand tracking device, which allowed users to navigate the terrain, interact with celestial objects, and adjust the UI inside the virtual game world. The MetaQuest 2 features a high-resolution display, a powerful processor, and advanced motion tracking sensors, while the hand tracking system uses infrared sensors to detect and track users' hand movements.

To track changes to the source code, we used Plastic SCM, a version control system that provided invaluable branching, differing, and merging tools for seamless collaboration on the codebase from any location. Branching tools allowed each member to have an individual copy of the source code to modify without interfering with or changing other members' work, while merging allowed us to integrate changes back to the original source code.

We used Unity as the game engine to develop the VR trek app, which is widely used in the gaming industry and provides a variety of useful tools to create an immersive VR experience. These tools allowed us to create a seamless UI with pre-built buttons, sliders, text-boxes, and other components to integrate into the VR world.

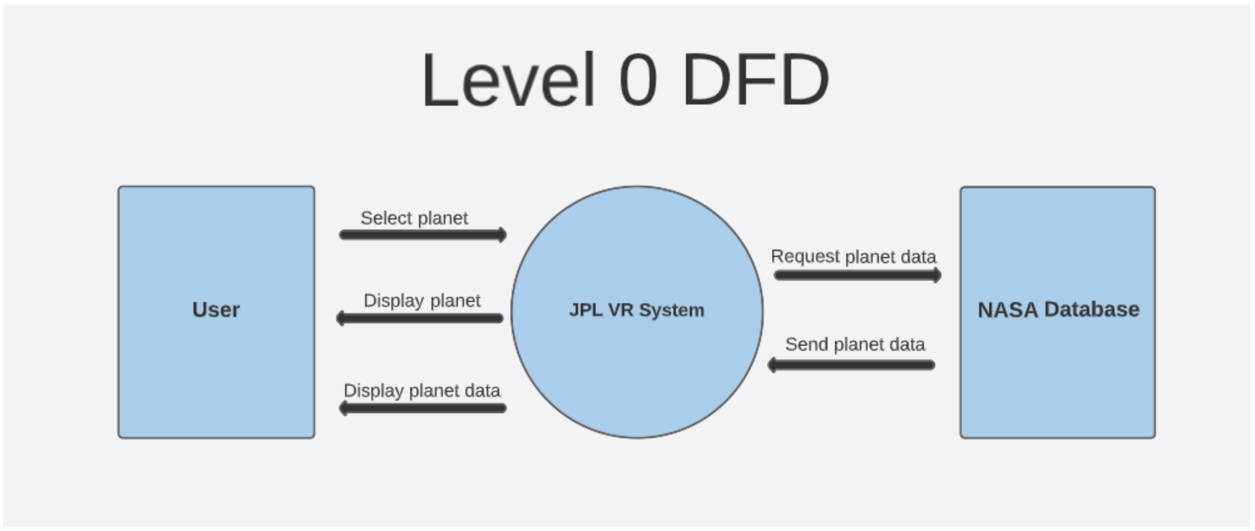
The language used to code the VR trek game world was C#, which is an object-oriented programming language highly utilized in the gaming industry, with tools for creating complex scripts that allow the game-world to function. We used Visual Studio Code as the IDE for testing this code, and its expansive debugging capabilities, extensions, and plugins were critical to creating functional software.

2.2. Reused Products:

As our primary goal was to update VR Trek, we confidently leveraged various products from the previous project. We effectively reused two key types of products, namely scripts and game assets, to replicate the previous build. The scripts remained untouched as they efficiently managed object generation, API connections, and input management. Notably, scripts such as "TerrainModelManager.cs" expertly controlled the generation and texture application of the globe. Furthermore, we opted to retain the game assets to ensure accurate reconstruction of user interfaces and player environments. Among these assets were game object models and materials, which proved essential to our success.

3. System architecture:

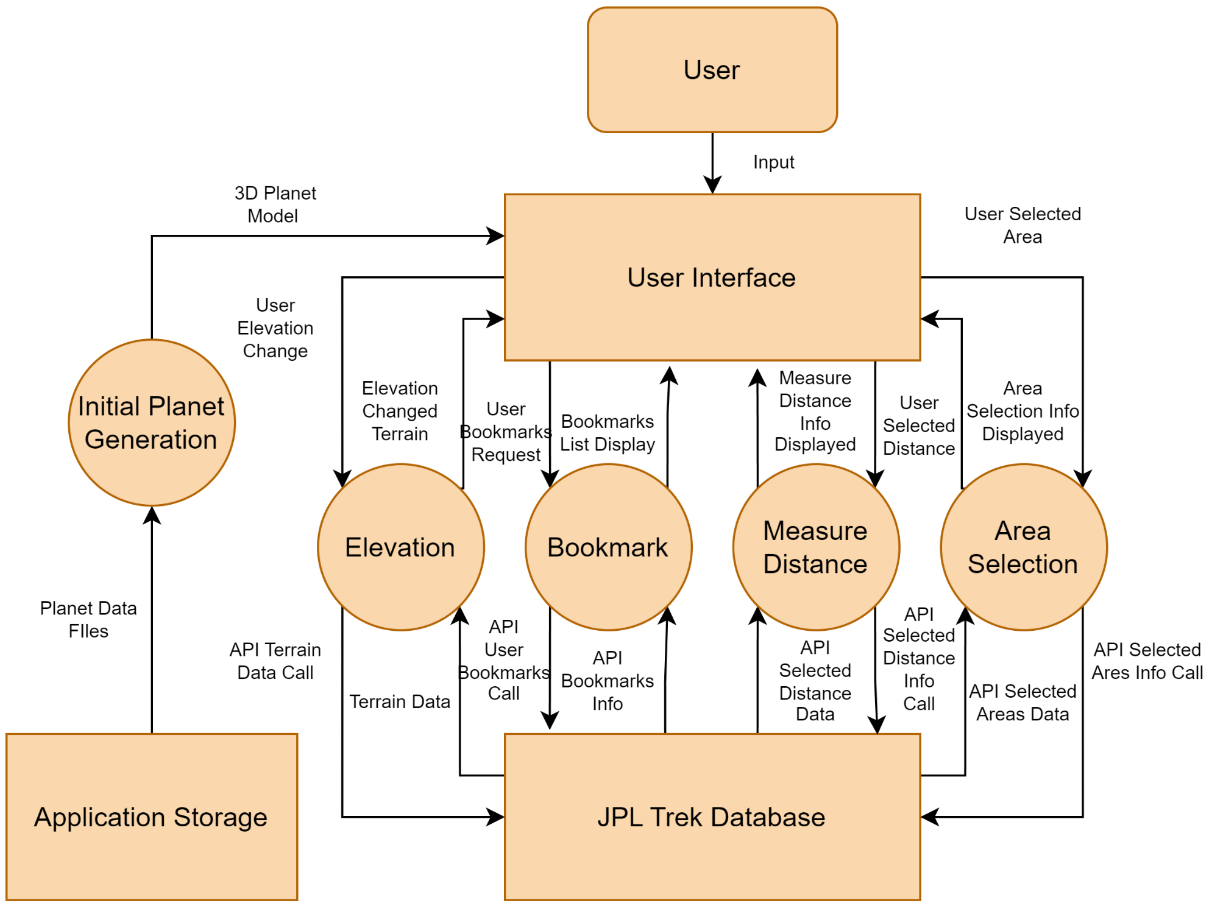
3.1. Overview:



- **The User:** The user will be able to interact with the planet and gather information about it through the user interface.
- **JPL VR System:** The objective of the project was to create the VR system which incorporates the functionalities of the previous VR Trek project with the OpenXR Standard. The JPL VR system's purpose is to display the functions that the user wants to see in virtual reality, such as seeing the terrain of the planet, seeing different planets, or seeing subsets of the planet.
- **Nasa Database:** The JPL API allowed us to pull information straight from the JPL Database. We were able to obtain different types of tiff images to recreate Mars through virtual reality.

3.2. Data Flow:

Here is an overview of the Framework as a system, and how it connects to the App (Implementation Activity), incidentally, this is also our DFD level 1:



There are six major modules in this system. They are described in more details in section 6. Here is a brief overview of them:

3.2.1. TerrainModelManager: This class is a manager which consists of many separate services in the application. This class handles all of the requests to create a new globe or refresh the current globe. This class is responsible for adding all the textures on the globe and even creating the height exaggeration.

3.2.2. XRInteractions: All of the XRInteraction classes are custom classes that make the controllers interact with the objects that we generate through code.

3.2.3. XRController: The XRControllers classes determine how the Virtual reality controllers will affect the Game objects inside the virtual reality environment.

3.2.4. User Interface: This is an interface that will allow the user to receive and see different parts of the planet displayed. The user will be able to exaggerate the height of the planet and he will be able to see different subsets of the planet.

3.2.5. Https: This is an interface used to manage all of the API calls being made to JPL's public API. It is there that we are able to gather the different textures of Mars and the Moon

3.3. Implementation:

The project was split into four sections: The globe Generation, the XR Controls, and finally the User interface and the control panel.

3.3.1. Globe Generation: The generation of the globe involves the utilization of the supplied Digital Elevation Model (DEM) located at TerrainConstants. This DEM is passed to the GlobeTerrainModel class, where its mesh is created based on the information contained in the DEM file. A tessellated plane is formed and transformed into the shape of a globe, with the inclusion of a Mars texture applied to the object. Additionally, various essential components, such as the physics mesh and the XRInteractable capabilities, are added to the game object within this script.

3.3.2. XRControls: The XR controls are implemented through a sequence of delegate actions and events. The XRInteractable object, which is the target of interaction, dispatches a OnButtonPressed function by means of delegation within the XRController class. In this class, the buttons of generic XR Controllers are mapped to corresponding function events. The button assignments are accomplished via the CustomController class utilizing the Unity Input System - Input Actions.

3.3.3. User Interface/User Experience: The Virtual Reality (VR) headset and controllers are connected to an object known as a VR rig. The XR rig, a Unity asset, is made available for free on the asset store once the OpenXR library is employed. The buttons on the controllers are mapped to specific input actions, while the movement of the hands is individually tracked through the utilization of 'XR Controller (Action-based)' scripts, also provided by the OpenXR library. Similarly, the headset is equipped with built-in scripts that assign and sync the main camera to the player's headset tracking. Player movement can be configured using prebuilt scripts, but in this particular case, they were modified to align with a more realistic and constrained movement profile within the globe room.

3.3.4. Control Panel: Within the application, we have two control panels that utilize the Zen Fulcrum embedded browser as both a display and a web function manager. The first control panel takes the form of a table positioned directly in front of the globe. Within this browser, the functions responsible for directly modifying the variables of the globe are located. These functions encompass tasks such as adjusting terrain exaggeration, toggling materials, and displaying navigation lines. The scripts responsible for these functionalities can be found in the TerrainControlPanel and ZFBrowser FunctionSet scripts.

The second control panel is accessed by pressing the assigned menu button, which is currently mapped to the right hand's primary button. The majority of the functions available on this panel require the utilization of the JPL API and can be located within the ControllerModal class.

4. Conclusions:

4.1. Results:

We created a globe mesh from scratch using computer graphics techniques, specifically triangulation. The user will be able to interact with the globe in an intuitive way and will be able to gather information about the globe via the User interface.

By utilizing JPL's API, we obtained TIFF images that served as the foundation for constructing our globe. This ensured the accuracy and reliability of the globe's appearance. Additionally, we modularized the application through the implementation of various classes, promoting code reusability and efficiency. We also incorporated additional features, such as texture removal, terrain height exaggeration, and globe rotation, enhancing the application's functionality and providing users with more customization options.

Furthermore, we utilized the Asset ZFBrowser to implement an angular application inside the unity application. This allowed us to build the control panel.

To enhance user interaction, we successfully converted the Steam VR controls to conform to the Open XR Standard. .This seamless integration improved the overall user experience, making it more intuitive and accessible.

4.2. Future:

The JPL VR Trek holds promise to in-depth scientific exploration and an immersive user experience. As the app currently is, we believe there is still room for improvements that can be implemented to further enhance the app. This section will serve to provide potential advancements to the app as well as practical tools that we believe can be added to further the user experience.

Mars and other Planets:

While the current version has focus only on Mars, we would hope to see other celestial bodies within the globe view, such as the moon or any other planet from the solar system. Expanding the globe view to show the textures of other celestial bodies such as the moon or Jupiter will help complete the XR experience for the users.

Control Panel Functions:

As other celestial bodies get added to the globe, the user will require a place to select which planet they would like to view on the globe.

As further advancements are made on the app, the control panel will have to update concurrently to keep up with all the functions.

Ruler Tool:

Adding a function that measures the distance on the celestial bodies would add a good way to analyze certain aspects of the planet. Users will be able to see the surface of a planet, and when used with the terrain exaggeration, will be able to get a detailed look of that section while in first person. The toll helps assist in determining the scale of the planets in comparison to other landmarks on the planet.

5. References:

Angular Developer Website: https://angular.io/docs

ZFBrowser documentation: https://zenfulcrum.com/browser/docs/Readme.html

Unity Documentation OpenXR: https://docs.unity3d.com/Manual/com.unity.xr.openxr.html

OpenXR Documentation:
https://registry.khronos.org/OpenXR/specs/1.0/html/xrspec.html