PERCEPTION ULTRA:

USING VIRTUAL REALITY TECHNOLOGY TO VISUALIZE ASTRONOMICAL DATA

The text does not wrap; each field must be entered individually.

By

Ricardo B. Aviles

A capstone project submitted for
Graduation with University Honors

May 10, 2018

University Honors
University of California, Riverside

APPROVED

_____
Dr. Tim Labor
Department of Music and Department of Media and Cultural Studies

_____
Dr. Richard Cardullo, Howard H Hays Chair and Faculty Director, University Honors
Interim Vice Provost, Undergraduate Education

Abstract

The purpose of this research project is to create an immersive and new way for scientists to view and interact with data. Perception Ultra is the title of a virtual reality experience that was crafted from the latest game development software, the Unreal Engine. This experience is a space fly-through simulator where players encounter three types of visualized space objects: stars, gas, and dark matter. Location data for these objects was obtained through Dr. Laura Sales from the UCR Department of Physics and Astronomy. The data has been scaled down to make visualizing them in virtual reality easier, but the spatial relationships are accurate. Furthermore, a filtering system is in place to allow users to emphasize certain space objects against others which allows for a clearer view of spatial organization. Additionally, to make this experience more than just a learning tool there are game elements incorporated to make it more fun, immersive and interesting to a wide audience. Finally, the game design elements make this experience accessible to those with disabilities. Perception Ultra uses visual and vibrational cues to make the game playable without sound and it can be completely experienced in a seated position.

Acknowledgments

I would like to thank Dr. Tim Labor and Dr. Mario De Leo Winkler for mentoring me on this project. Furthermore, I would like to thank FIELDS interns Jessica Gonzalez, Elijah Marchese, and Angela Larias for their help. Special thanks to Dr. Bahram Mobasher and Dr. Laura Sales for providing the data and funding. Also, I'd like to thank the Sweeny Media Center for providing a space to build and test this project. Further acknowledgments:

- Fellowships and Internships in Extremely Large Data Sets (FIELDS)
- Ellustra simulation
- Erika Suderburg
- Michael Pazzani
- UCR Office of Developments
- Olaf Stapledon

Table of Contents

Introduction

New technology like virtual reality headsets have created new avenues for academic exploration. For this project, the goal was to find a way in which large sets of astronomical data could be visualized and manipulated. Many forms of digital media have been used to display astronomical data; however, one form that has not been utilized greatly is virtual reality. Using HTC Vive headsets and Unreal Engine game development software, the idea of creating a game that uses real data was born.

Background

The Unreal Engine utilizes a visual scripting system called *Blueprints* (see fig.6) that allow people with little knowledge of computer programming to create projects ("Blueprints Visual Scripting"). This project was built entirely from blueprints with some pre-programmed assets provided by the developers of the Unreal Engine called *Starter Content*. This content provides things like "materials to particle effects to primitive shapes and even a few props" ("Starter Content"). The data used in the project was provided by Dr. Laura Sales from the UCR Department of Physics and Astronomy. The data provided information on the location and ID associated with stars, gas and dark matter.

Important terminology regarding the Unreal Engine:

Actors: Anything that is placed into a level.

Pawn: An actor that a player can control.

Component: Additions to an existing actor.

Widget: A type of user-interface for things like menus and dialogue.

NPC: Non-playable character. These are characters that players see in the world and may be able to interact with.

Methodology

The process of designing and testing this project took place over the course of about 10 months. Much of the research for learning the Unreal Engine came from official documentation and some independent study. Information regarding astronomy came from Dr. Laura Sales and Dr. Mario De Leo Winkler of the Department of Physics and Astronomy at UCR. To all collaborate on the same project, a source control program called *Helix Visual Client* was used. This program allowed each member of the group to work on the same project. Changes and additions to the project were uploaded to a server mediated through the *Helix Visual Client* software, and everyone could download the changes onto their own workspaces. In addition, weekly meetings were held to showcase any updates, discuss ideas, and plan for future deadlines. The initial months of research consisted of getting comfortable with the Unreal Engine. Afterwards, efforts were made to create a comfortable locomotion mechanic and to import the real astronomical data into Unreal Engine. Testing was conducted many times per week using an HTC Vive headset. In addition, weekly tests were conducted by my mentor Dr. Tim Labor who attempted to break that game at every stage.

World Building and User Interface

In Perception Ultra there are three levels that were built from basic geometric shapes and customized with materials, art, props and particle systems to make them look interesting. Each level was fitted with an appropriate set of controls and movement mechanics such as teleportation and flying. User interface varied with each level, but consisted of on-screen instructions, directional arrows and 3-D menus.

Level Design to Accommodate Disabilities. Perception Ultra can be experienced completely seated allowing wheelchair play. Locomotion in each level is designed to reduce motion sickness. For example, in the Stapledon Institute level the player must move around using

teleportation instead of free movement. In the space level, the player must fly, but the mechanic is designed to reduce jerky movements, and the black backdrop of space aids in keeping the player balanced and comfortable. Additionally, everything in the game is guided by visual cues such as blue floor trails, directional arrows, and on-screen text (see figs 2A and 4). Also, there are vibrational cues that notify players if they have grabbed something. The visual and vibrational cues make the game accessible to the hearing impaired.

Locomotion

I lead the development of the space locomotion mechanic. The Unreal Engine only has a default teleportation mechanic that players can use to move around. For our space level, we wanted players to be able to freely move. Utilizing some ideas from the official unreal tutorials I created a mechanic that allowed players to fly through space. The design follows a simple six step formula:

1. A forward vector projects from the front of the motion controller
2. An input moves the player in the direction of the vector.
3. While input is made, the player moves towards a maximum speed.
4. Release of the input stops moving the player towards maximum speed.
5. Blueprints regarding deceleration allow the player to keep moving at the last known speed without input.
6. Another input makes the player come to a stop at a set deceleration.

Vector Based Locomotion. A pawn was created that would hold the gameplay logic for locomotion. A motion controller was attached as a component to the pawn. The first thing to accomplish was to get the "World Rotation" of the motion controller in the virtual world. I wanted to limit the world rotation to the Y and Z axes of the controller, so I used a built-in command from within Unreal Engine called a "Break Rotator" to define those parameters.

Furthermore, once the rotation of the controller was found I wanted a “Forward Vector” to project from front of the controller in relation to the defined axes. Then, I added an “Add Movement Input” command that corresponded to the direction of the vector with the target being the pawn.

Speed in Space. The rest of the locomotion blueprint script is manipulating maximum speed and deceleration. To set how fast the player could travel, I created a variable called “Maximum Speed” set at 22500 Unreal units per second. I also created a variable called “Minimum Deceleration” that would set the deceleration to zero once there was input so that the player would continue moving. If the button was held down, the pawn would accelerate towards maximum speed and set the deceleration to zero. As a result of the deceleration being set to zero, the pawn would continue at the last known speed once the button was released. This mechanic allowed players to cruise through space and look around comfortably. To stop the pawn, a button had to be pressed and it would set a “Maximum Deceleration” to 8000 (arbitrary number) which eased the player to a stop.

Spawning Data

Data we received from Laura Sales was an hdf5 file type, but was converted to a csv file so that the Unreal Engine could read it. Unreal Engine converts csv files into data tables. From these tables, Unreal Engine parses the data by mapping each row to an index. Then you can locate the data at each row by selecting the category or column. The data came in separate files for location of gas, stars and dark matter. The data is set to spawn around 30,000 space objects at the start of the level, at the same time (see fig.3B).

Visualizing the Data. The data sets we received regarding the locations of stars, dark matter and gases had to be visualized in a simple way. To do this, we made every object a sphere with a

glowing material and an ID number (see fig.4). The colors of the objects were not based on data and purely creative choice.

Filtering the Data. In the space level, to emphasize certain space objects against others, a filtering system was created. The basis of this filtering system was to highlight certain objects so that they are easier to see in the level (see fig.7). While players fly through space, they can bring up a small menu that allows them to select which data type they want to highlight (see fig.3A) The way this works is that a player selects an object type, like dark matter, which sends commands to the Unreal Engine to look at all the data for that object type. Blueprints are set up to alter the properties of that object type when it is selected such as changing its emissivity.

Dialogue

For players to interact with other characters in the game I created a dialogue system. This system had to be created with virtual reality in mind. Normally, dialogue is created with widgets, which are on-screen menus that allow players to select dialogue options or responses. In VR, however, players are using a headset and displaying a menu directly on the headset camera is not effective. One solution is to make the widgets 3-D (see fig. 1A).

Data Driven Dialogue. The dialogue in the game is all derived from a spread sheet that contains both the NPC dialogue and the player's possible responses. The rows of the spread sheet represent the actual text of the dialogue and the columns represent "screen numbers". These screen numbers simply label all the possible dialogue interactions that the player can have with an NPC (see fig.5). Next, blueprint logic was created to connect the data from the table to the widget. I had to create four different functions that dealt with: setting visibility of the widget, setting the text onto the widget, setting each group of dialogue choices to a screen number, and a dialogue exit function.

VR Compatibility. Once the blueprints were created, they had to be further altered to allow 3-D interaction for VR. It required creating an actor with a widget component that referenced the custom widget I made. Then, the motion controllers had to be edited with "Widget Interaction" components to allow the widget to work in VR. Furthermore, players needed a way to initiate dialogue. The simplest solution was to have players point their controllers at the NPC and press a button that would prompt the 3-D widgets. To accomplish this, in the motion controller blueprint I created a script that uses raytracing in conjunction with a right controller trigger input (see fig.6B). If the right trigger is pressed, an invisible line is projected from the controller to check if an NPC has intercepted the line. If true, a message is sent using an event dispatcher to the 3D widget blueprint to toggle its visibility.

Animations

Animations were created using the sequencer feature in Unreal Engine. The animations consisted of moving objects smoothly in response to certain player actions. For example, when a player finished talking to an NPC, the NPC would move out of the way, using animations, to reveal a point of interest.

Results

The result of the project is a small VR experience with game elements to make it more enjoyable and accessible to the public. The experience consists of three main levels: an options level, Stapledon Institute Level, and a space level. The options level is very basic, consisting of a black backdrop illuminated by stars of varying color, intensity and size. Players can select from a few choices such as: New Game, Load, Settings, Quit and Credits (see fig.1A). Additionally, players cannot move but are able to look in full 360 degrees of the environment (see fig.1B) The Stapledon Institute is a building built in 4-room dungeon style named after Science Fiction author Olaf Stapeldon whose work "Star Maker" inspired this project. The Stapledon Institute

consists of a starting area, receptionist area, a keycard room and a teleportation room. It is the players goal to explore and find commander Mario Bot to find out the details of their first mission. In this level, many elements are showcased such as locomotion through teleportation, dialogue, animations, art, particle systems, custom objects, and world building. Some of the assets used in the level were pre-made by the developers of the Unreal Engine such as: the teleportation mechanic and the ability to grab objects. When the player reaches the teleportation room, they can speak with commander Mario Bot and be sent to space. The space level is where the player views the real data. The level reads data from a spreadsheet then spawns it around an origin (0,0,0). The data is not scaled realistically, so objects are much closer for easier viewing. However, the distance between space objects is proportional and accurate. In the space level, there are two main interfaces: a pause menu and a space journal. The pause menu allows the player to stop the game and exit if they please. The space journal has a couple of functions. One is to filter the data and the other is to teleport to known locations in space. Filtering the data is a process that illuminates space objects for easier visibility (see fig.3A). The player's objective in space is to find the TRAPPIST-1 star in under 90 seconds. While searching for the star, players are prompted with on-screen warnings to hurry. Also, a green arrow is present to direct the player where to go. There are audio cues that get louder and louder to signify that the player is getting close to the TRAPPIST-1 star. Currently, the star is in a dummy location for the sake of the game. If they player is successful in their mission, they are congratulated and send back to the options level (see fig.4).

Figures

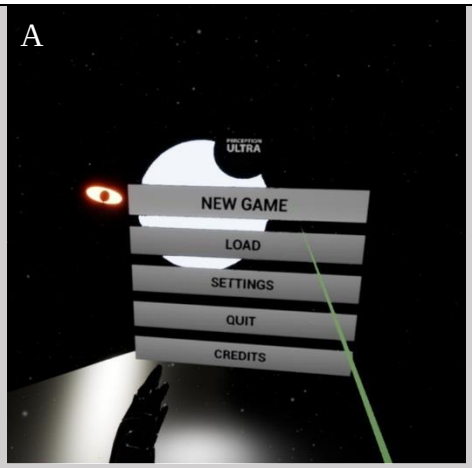
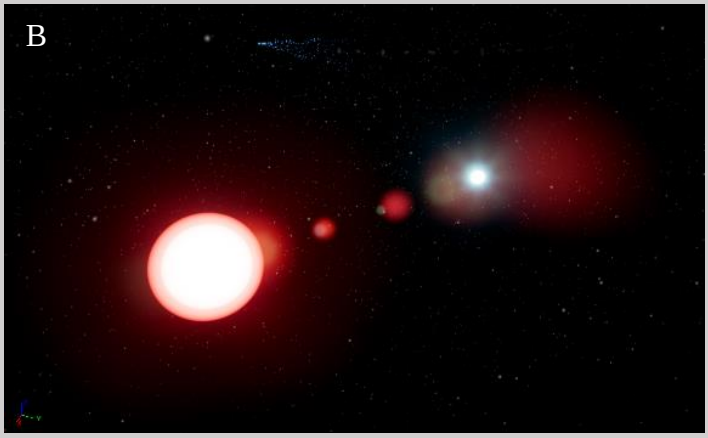
Options Level	A  B 

Figure 1.

A. A screenshot of the options menu from within the options level. Players can choose from a few options such as New Game and Credits.
B. A screenshot in-game of some of the objects that players can see in the options level. Objects such as stars of varying intenisty, color, and size. These objects are entirely creative and do not reflect any data.

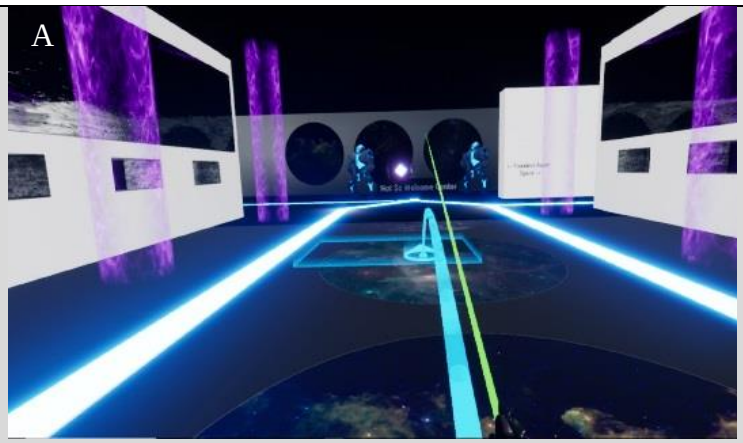
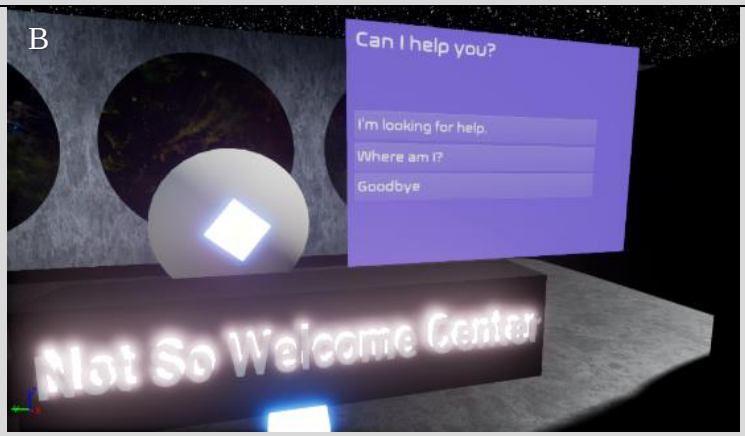
Stapledon Institute Level	 

Figure 2.

A. Screenshot taken in-game of the starting area of the Stapledon Institute level. The blue arch is the teleportation mechanic in action. The circular decals on the floor are space images created by Art Director Angela Lerias.
B. Screenshot taken in-game of the receptionist that the player can speak to for directions. The purple screen to the right is the 3-D dialogue widget that allows players to choose different dialogue options.

Space

Level

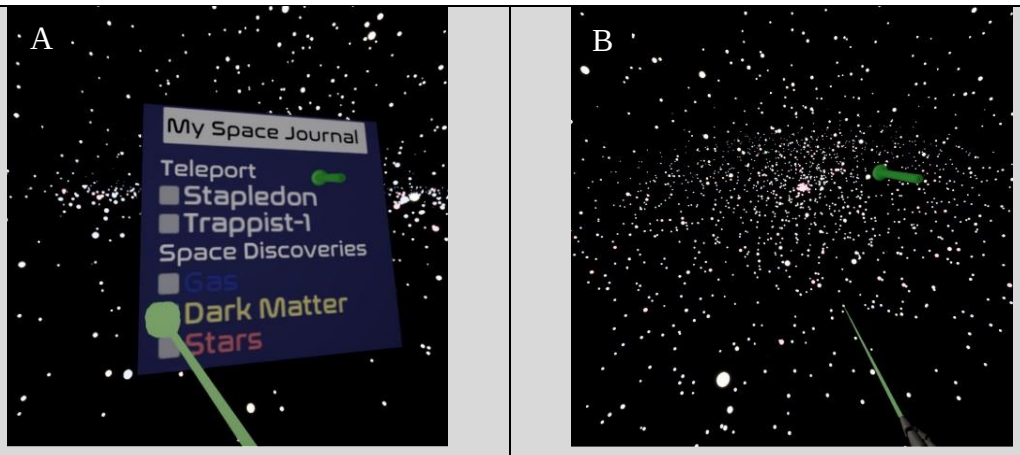


Figure 3.

In-game screenshots of the space level. A player is immersed in the darkness of space and the illumination of stars, gases and darkmatter.

A. Screenshot taken in-game of the filtering system. Players can open up a 3-D menu called "Space Journal" which contains the options to emphasize single or a combination of objects.
B. Screenshot taken in-game of the unemphasized objects. Their distribution resembles a disk witthin a sphere.

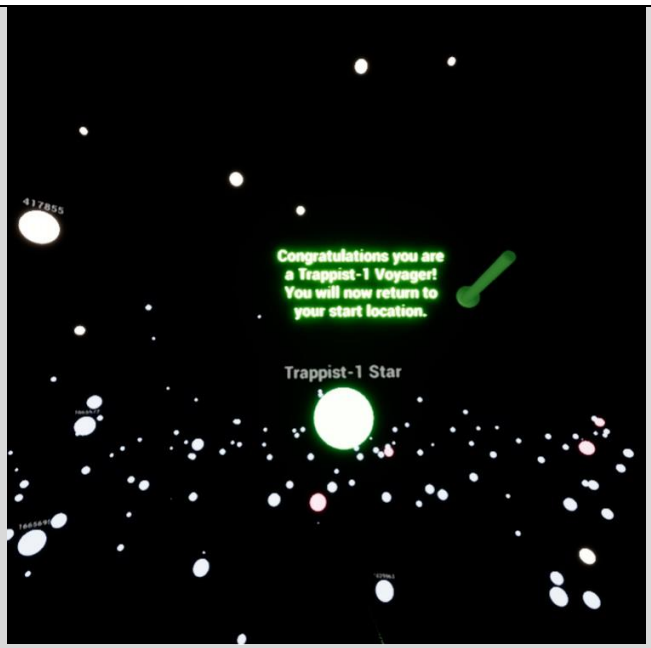


Figure 4.

In-game screenshot of what it looks like to find the TRAPPIST-1 star. Surrounding space objects are labeled with ID numbers that pop up when the player comes near them. Additionally, a green arrow that tells the player which direction to go is always present.

	A	B	C	D	E	F
1		1	2	3	4	5
2	Player	Hello.	I have to go.	I have questions.	Who am I?	Where am I?
3	NPC	Greetings!	How can I help you?	Ask away!	You are Player.	In the Unreal Engine.
4						

Figure 5.

A simple example of the dialogue spreadsheet. Rows contain the dialogue text and the columns represent the screen numbers. According to the screen numbers there are 5 possible dialogue interactions with only one possible response. In the full project, the largest dialogue had 16 possible dialogue interactions and three possible responses.

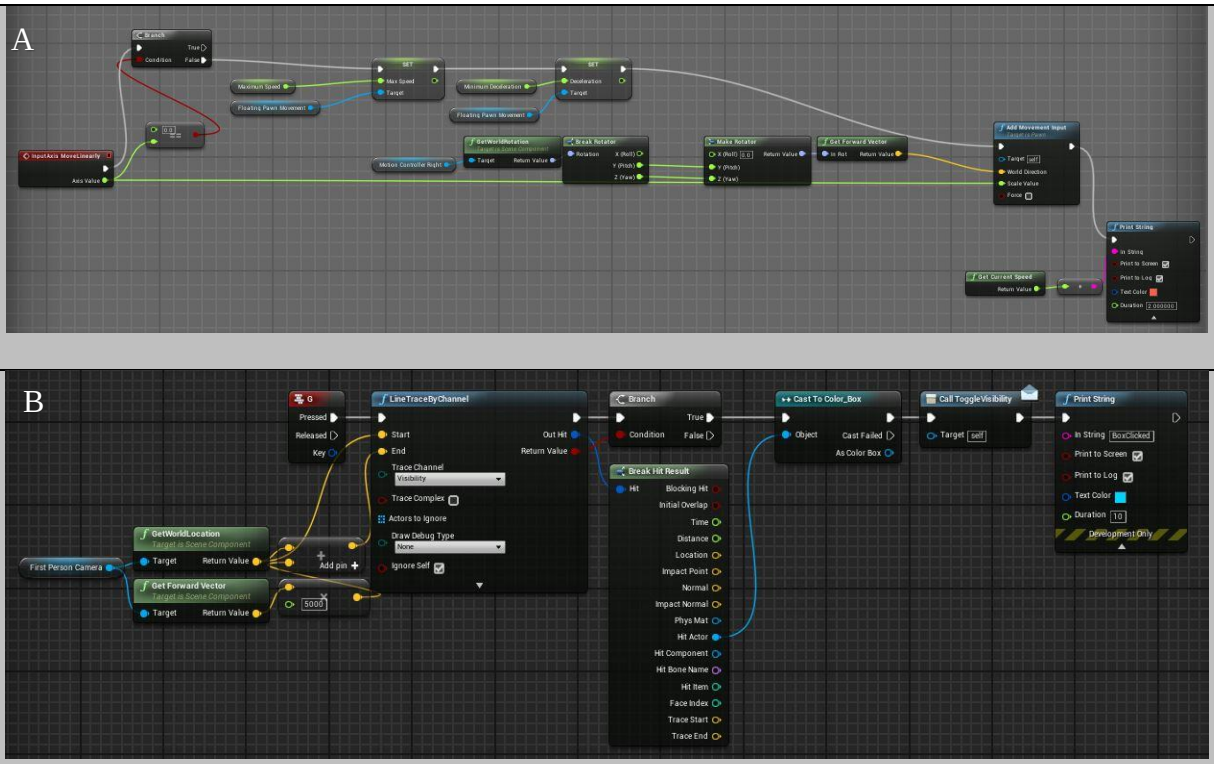


Figure 6.

These are some in-game examples of the blueprint system in the Unreal Engine. Blueprints are a way of visually scripting gameplay logic through a series of commands and nodes.

A. The blueprint for the flying locomotion mechanic.
B. The blueprint for the 3-D widget that allows players to initiate dialogue.

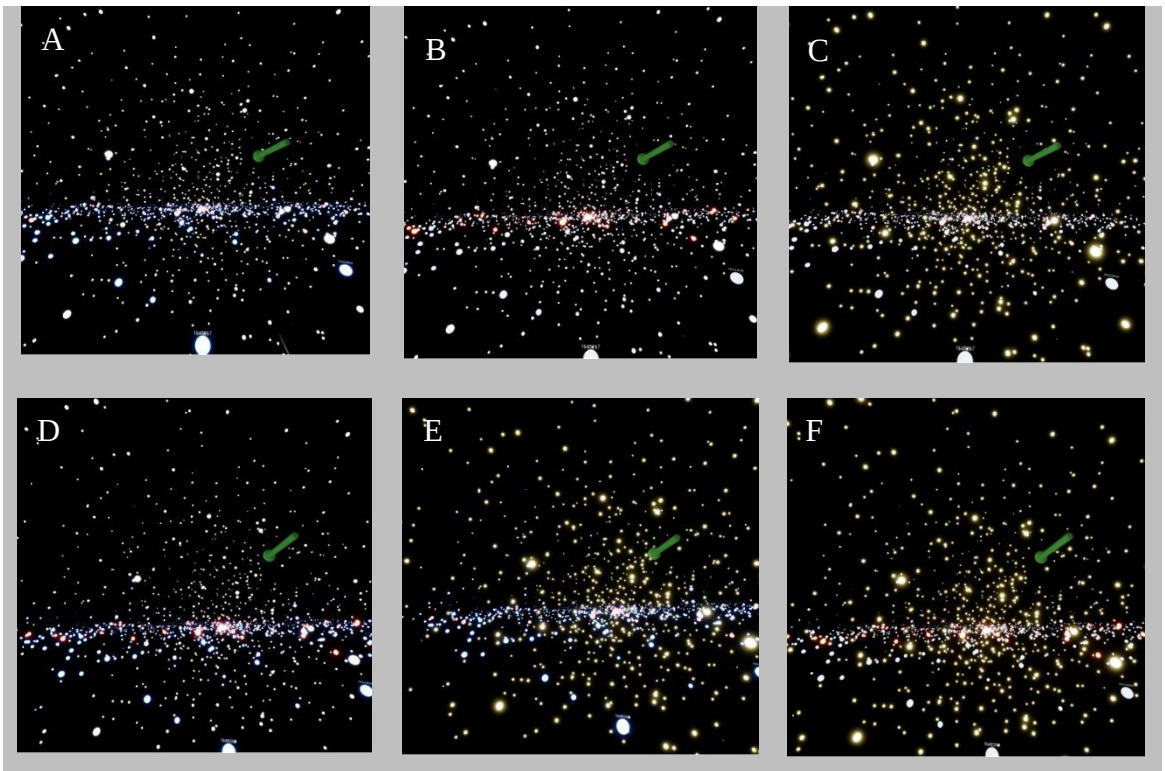


Figure. 7.

These images were taken in-game and show the distribution of the data filtered in various combinations.

A. Gas only (blue)
B. Stars only (red)
C. Dark matter only (yellow)
D. Gas and stars
E. Gas and dark matter
F. Dark matter and stars
G. Gas, stars and dark matter

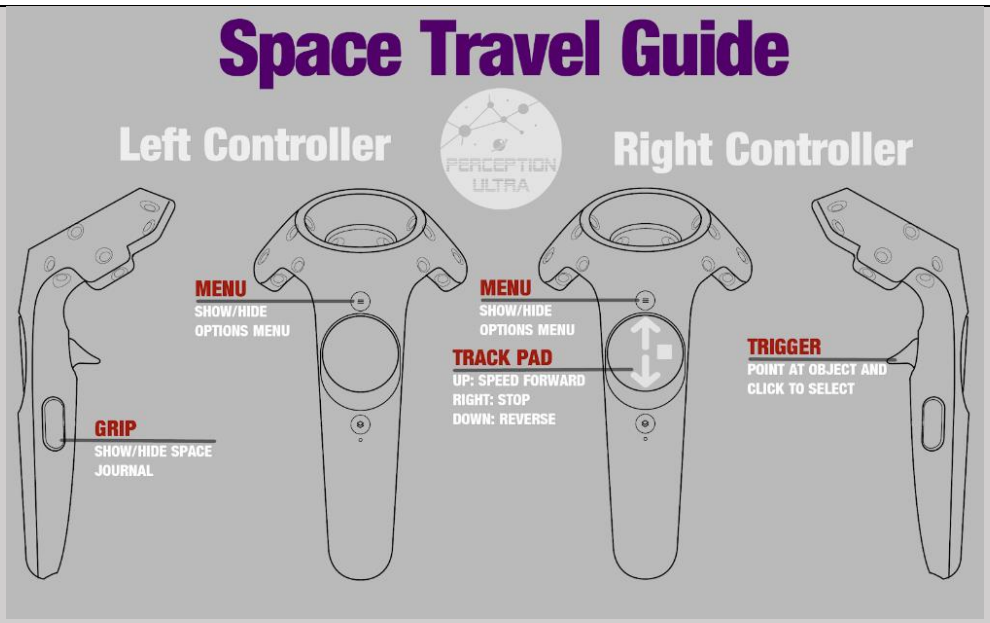


Figure 8.

A tutorial image created by Angela Lerias and updated by Jessica Gonzales. This image helps players get familiar with the controls for flying in the Space Level.

Discussion

At the result of this project it is important to speculate on the challenges that were faced, the data model, and future developments that would like to be achieved.

Challenges. There were many challenges that had to be overcome in this project. One of the biggest challenges was the fact that my team and I had no prior game development experience. We had to learn Unreal Engine from scratch with guidance from my mentor Dr. Labor. Learning level design was another challenge. Levels could not just be randomly constructed, but had to be built with possible player scenarios in mind. Dr. Labor helped us overcome this challenge by suggesting a dungeon-style layout, a classic game strategy, with basic rooms and objectives. Many mechanics and systems had to be built from scratch using the Unreal Engine blueprint system. For example, the flying locomotion mechanic had to be crafted in a way that was comfortable and, most importantly, in a way that reduced motion sickness; a very common issue

with VR (Jiwon, 2017). Additionally, the dialogue system had to be altered countless time to account for the complexities of VR. For example, the dialogue had to be strategically placed in the level so that players would not skip over it, a problem that is not commonly encountered in regular games. Also, just like the level design, the dialogue system had to be created to account for player action. Some questions I had to ask were "What if the player skips the dialogue?" or "What if they leave in the middle of a dialogue sequence?". To account for these possibilities, I had to figure out how to make different parts of the game design communicate with each other. An important source for coming up with solutions was the official Unreal Engine documentation website which had articles such as "Event Dispatchers" and "Using Raycasts (Tracing)". *YouTube* has always been an important source for learning. For communicating between blueprints, I had to master event dispatchers and the video *HTF do I? Event dispatchers in Unreal Engine 4* helped me tremendously. Another challenge was spawning the real data in the game. This part of the project was led by computer science majors Jessica Gonzalez and Elijah Marchese. The major challenge was converting the data into a readable format and being able to organize thousands of data entries within the Unreal Engine. Furthermore, we had to test the limits of the Unreal Engine. With our current data model of spawning the data all at once the Unreal Engine could handle around 30,000 space objects without crashing, any more would cause the program to become slow and error prone. Finally, we needed a way for this project to be useful. We put data into this game so what next? The filtering really made our project worthy of something. It created good way of viewing the data in terms of its size, shape and distribution. *The Data Model.* The representation of the data is simplistic, but with good reason. We could have spent much more time designing textures, materials and particles to make the space level seem more realistic; however, that would have taken a lot more development time. Our main

goal was to have a playable prototype with a usable data model. Representing each space object as a sphere in space made it easier to view the distribution of space objects. For example, it is clear when using our data model with the filtering system that stars and gases occupy a lot of their space along the same plane; like a disk. While dark matter surrounds everything like a sphere. In the future, there are plans to make more intricate and complicated designs.

Future Developments

- Educational Outreach
    - Presenting our game to the public could inspire youth to become interested in astronomy, Virtual Reality, and game development
    - One of our team members works with Riverside Unified School District. With this, we can easily set up events to schools all around Riverside County.
    - Our team could reach out to high school programs such as AVID or PUENTE to spread interest to college bound students.
- Creation of a buffer system for managing data and creating a continuous universe
    - There is currently no buffer system because all our data is spawned at the beginning of the game. Our data is static and does not change. In the future, we would like to develop an algorithm where the nearest data is spawned and the further data is de-spawned.
- Applicability to other fields
    - VR can be applied to other fields of learning besides astronomy such as medicine or therapy.
    - Other forms of data can make use of 3D visualization and manipulation in VR.
- Porting to mobile and other platforms

    - One goal of ours is to make this game more accessible to the public. We will pursue this by packaging this game into a mobile version. As of right now, Unreal Engine only supports Android devices on two mobile VR platforms: GearVR and Google VR. Oculus Rift is another possible platform to run our game on, however our focus is using the HTC Vive.
- More accessibility to people with vision impairment.
    - Currently the game can be played sitting down and there are visual cues like text and on-screen menus.
    - However, the current model is not adequate for the blind. Future developments hope to bring more complex audio design.

Conclusion

In conclusion, I hope that this project helps open new avenues in science. This type of technology is growing faster and better every day and can be used for a wide variety of research. Astronomy is not the only field that can benefit from virtual reality. I can see this technology being used in medicine, biology, chemistry and many other fields.

References

"Blueprints Visual Scripting." *Introduction to Blueprints*, Epic Games, docs.unrealengine.com/en-us/Engine/Blueprints.

"Event Dispatchers." *Introduction to Blueprints*, Epic Games, docs.unrealengine.com/en-us/Engine/Blueprints/UserGuide/EventDispatcher.

Gillon, Amaury Triaud Michael. *TRAPPIST-1*, www.trappist.one/.

Hiyosee. "Unreal Engine 4 Dialogue System Tutorial." *YouTube*, YouTube, 22 Mar. 2016, www.youtube.com/watch?v=w0pr5llbQZQ.

Jiwon, L., Mingyu, K., & Jinmo, K. (2017). A Study on Immersion and VR Sickness in Walking Interaction for Immersive Virtual Reality Applications. *Symmetry (20738994)*, *9*(5), 1-17. doi:10.3390/sym9050078

"Mitch's VR Lab." *YouTube*, YouTube, www.youtube.com/channel/UChvlNUgZKmEd-Gul_Tdv8Uw/videos.

"Motion Controller Component Setup." *Introduction to Blueprints*, Epic Games, docs.unrealengine.com/en-us/Platforms/VR/MotionController.

Stapledon, Olaf. *Star Maker*. Methuan, 1937.

"Starter Content." *Introduction to Blueprints*, Epic Games, docs.unrealengine.com/en-us/Engine/Content/Packs.

Taurian. "Trying to Call Event from Another Blueprint." *Unreal Engine Forums*, Epic Games, Mar. 2015, forums.unrealengine.com/development-discussion/blueprint-visual-scripting/36128-trying-to-call-event-from-another-blueprint.

Unrealty Simon. "Unreal Engine - [VR] Interactive Watch Menu." *YouTube*, YouTube, 17 Sept. 2017, www.youtube.com/watch?v=KQexLS2xjOs&t=228s.

"Using Raycasts (Tracing)." *Introduction to Blueprints*, Epic Games, docs.unrealengine.com/en-us/Gameplay/HowTo/UseRaycasts.

VR 360 TV. "Journey to Trappist-1." *YouTube*, YouTube, 14 July 2017, www.youtube.com/watch?v=YSLZeKCTDfY.

Wadstein, Mathew. "HTF Do I? Event Dispatchers in Unreal Engine 4." *YouTube*, YouTube, 18 Nov. 2015, www.youtube.com/watch?v=sEcoWGrF1Hg.