Extended Reality 3D Model Application in Space Exploration and Planetary Habitation

Mikhail Nikolaenko
Spring 2023

Background

Within the sphere of space exploration and discovery (SED) research exists a need for robust, interactive, and easy-to-use 3D visualization tools that assist in different aspects of space related science. Virtual reality, augmented reality, and mixed reality, conjointly known as extended reality, is a new emerging technology that focuses on creating immersive 3D experiences. With the rapid advancements in these extended reality technologies, SED can begin to incorporate these technologies to effectively visualize the different aspects of space science and develop further research. This research explores the virtual reality branch of extended reality and how it can be used to create an accurate, interactive, and multi-disciplinary 3D VR application with a multiple-use case and the specific intention to not be limited to a single field or area. Some proposed fields include, but are not limited to, astronomy, astrometry, early childhood – higher level science education, and aerospace. Some specific use cases include using the application as a learning tool for children to learn about the solar system in a classroom setting, ranging to using the application as an astronomy tool to view different metadata associated with celestial objects in a work or professional setting. The design choices that were chosen caters to the "interactive" aspect of this application, giving the user full control of the 3D environment to suite their specific needs. This includes functionality such as teleportation, scaling/time manipulation, as well as object highlighting. It is important to note that, although this application promises to reach useability within certain fields under the space science umbrella, there are other areas where this application needs more work to be used effectively; however, the current framework allows for this further development to occur with no issues with the concept of modules, which will be discussed later.

Prior work and research in this area remain in the early stages of development. After careful literature review and surveying sources related to this topic, multiple instances show the creation of similar 3D models of space related objects. One example involves NASA's "Eyes on exoplanets", powered by NASA's Exoplanet Archive, which presents a 3D universe model containing an approximate 1000 potential habitable exoplanets ("NASA's eyes: Eyes on exoplanets," 2018). There are a few things to consider: firstly, NASA's model contains data on purely exoplanetary objects. In this research, however, it was important to include data

involving multiple different types of celestial objects such as planets, moons, and stars which is vital for the wide variety of use-cases that this application will serve. This also means that in the example, proper or “complete” data is available for approximately 1000 objects (exoplanets), whereas in this research, proper data is available for millions of objects (planets, stars, moons, asteroids, satellites, etc.), given by databases publicly available. Secondly, NASA’s model is run through a webpage, which is useful for accessibility and low hardware specification requirements, but undesirable in other cases due to the (a) low fidelity design and (b) no spatial awareness, meaning there is no indication of scale and distance of these different objects. In this research, the application is a complete stand-alone, meaning the fidelity design can increase as hardware specifications increase, and because this application is taking advantage of virtual reality, scale, distance, and other spatial indicators can be properly identified. Compared to the NASA model which targets the single problem of exoplanetary data, this research targets the question of expandability: whether an application can serve as a basis for the solutions of many SED related problems.

Various frameworks were followed throughout the period of this research. As it involves creating software that requires high levels of accuracy and stability, it was important to follow standard software development protocols for the design of this research. A brief overview of this protocol is as follows: planning, analysis, design, implementation, testing, and maintenance. Throughout the entirety of the research period, different stages of the protocol were followed depending on the progress and integrity of the application. For example, if a design was failing, previous steps were repeated, such as planning, to ensure proper development of the software. Alongside this protocol, other frameworks were in place. The conceptual framework of key identified aspects of the application was incorporated to split the development into simpler steps. These steps include database analysis, software/hardware analysis, and interactive tools design. Although these aspects were split, they were developed in tandem. From this point forward, these steps will be known as “modules” of the conceptual framework, which helps distinguish their role in this research.

The significance of this research will be discussed in this next section through an example of where it can be applied. This example covers how this research can transform

current visual methods used in education and data analysis. It investigates the essence of visually assisted learning and data visualization by taking advantage of virtual reality. From the beginning, this research aimed to create a 3D virtual reality model and leverage virtual realities' capabilities to promote visualization of various areas of space science or SED. One of these areas, astronomical data analysis, is generally done by professionals in the field of astronomy, and existing databases of arbitrary celestial objects prove to be difficult to process for a non-professional individual trying to understand astronomy and astrometry concepts. In **Figure 1**, the raw metadata on various celestial objects given by the Gaia Data Release 3, a survey done by the European Space Agency, is shown.

Figure 1

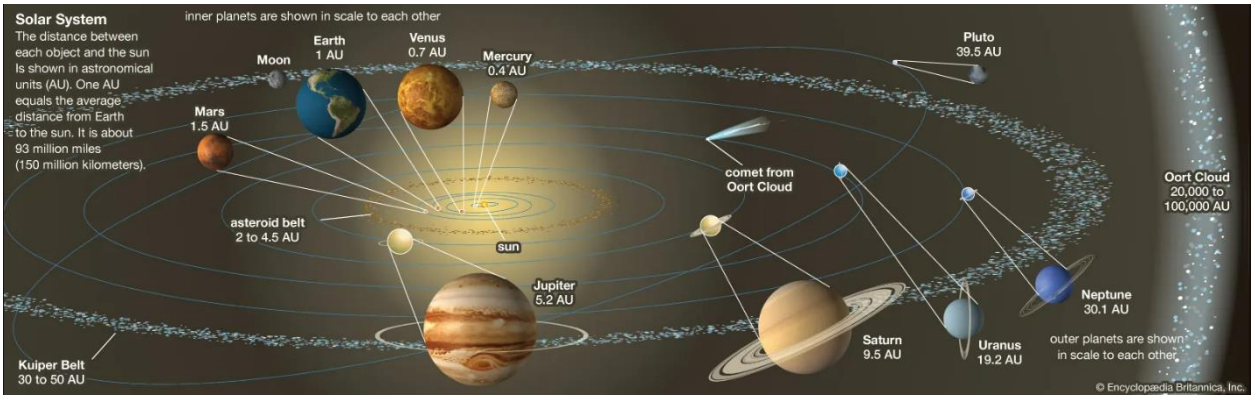
Gaia Data Release 3 Data, European Space Agency

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE
1	solution_id	source_id	random_index	ref_epoch	ra	ra_error	dec	dec_error	ra_dec_corr	astrom	ast	astri	astroi	astror	astron	astrom	astrometr	astrometr	astrometr	astrometr	astrometr	astrometr	astror	match	duplicate	scan_dire	scan_dire	scan_dire	scan_dire	scan_dire	scan_dire	scan_dire
2	1.64E+18	6.91E+18	644618244	2015	312.1157554	3.956163569	-3.74035	1.404508	0.99665	62	0	62	0	0	0		0.374397	3.284118	FALSE	1.235849	2.662341	2	9	FALSE	0.179991	0.180794	0.089429	0.942025	-55.6153	12.2722	-25.633
3	1.64E+18	6.91E+18	668015830	2015	312.1343319	4.045969091	-3.73963	1.437343	0.99665	62	0	62	0	0	0		0.432068	5.060065	FALSE	1.349939	2.566972	2	10	FALSE	0.179991	0.180794	0.08943	0.942025	-55.6164	12.27098	-25.6341
4	1.64E+18	6.91E+18	609250353	2015	312.138582	25.93889781	-3.73706	11.68324	-0.70815	25	0	25	0	0	0		1.58134	1.509319	FALSE	1.34948	0.121771	2	6	FALSE	0.691029	0.308073	0.848411	0.967013	51.66751	11.55506	12.92567
5	1.64E+18	6.91E+18	393393107	2015	312.1311733	24.81927117	-3.73526	11.17584	-0.7076	24	0	23	0	1	0		1.258626	1.049244	FALSE	1.164136	0.12126	2	6	FALSE	0.678125	0.253641	0.842835	0.965866	55.46267	8.501465	12.15571
6	1.64E+18	6.91E+18	241639886	2015	312.1346758	23.95391504	-3.73385	10.81131	-0.70745	25	0	25	0	0	0		1.077916	0.451322	FALSE	1.090901	0.091946	2	6	FALSE	0.691029	0.308073	0.848412	0.967013	51.66776	11.5553	12.92592
7	1.64E+18	6.91E+18	1083380421	2015	312.1281074	25.14685086	-3.73342	11.34067	-0.709	27	0	26	0	1	0		0.619861	0.287113	FALSE	1.047695	0.154572	2	6	FALSE	0.697708	0.332966	0.851243	0.967656	50.04232	12.59336	13.26761
8	1.64E+18	6.91E+18	991794988	2015	312.1124316	21.8538572	-3.73633	9.027788	-0.971	45	0	45	0	0	0			0	FALSE	1	0.04592	2	7	FALSE	0.130074	0.588884	0.40699	0.948593	119.7096	24.71952	-3.07636
9	1.64E+18	6.91E+18	451435156	2015	312.1067336	3.185126924	-3.7117	1.153372	0.98335	71	0	71	0	0	0		0.275343	0.335408	FALSE	1.029765	0.684084	2	9	FALSE	0.283846	0.12363	0.158696	0.926979	-54.3179	-13.6267	-43.0097
10	1.64E+18	6.91E+18	980057749	2015	312.1088187	24.87797005	-3.73645	11.22071	-0.70905	27	0	27	0	0	0			0	FALSE	1	0.160893	2	6	FALSE	0.704383	0.356351	0.854107	0.968296	48.56664	13.425	13.58268
11	1.64E+18	6.91E+18	609382341	2015	312.1248386	4.218940097	-3.72405	1.526005	0.99885	79	0	79	0	0	0		0.416117	49.05027	FALSE	2.726636	4.96576	2	10	FALSE	0.356403	0.162537	0.234846	0.924319	-54.1687	-32.6071	-47.4718
12	1.64E+18	6.91E+18	21568037	2015	312.122955	4.348196974	-3.72973	1.572403	0.9986	79	0	79	0	0	0		0.468832	107.589	FALSE	2.989277	4.117311	2	10	FALSE	0.365964	0.175772	0.257419	0.926684	-56.1868	-36.1545	-47.5403
13	1.64E+18	6.91E+18	772363589	2015	312.1257629	3.255037154	-3.72333	1.183533	0.98385	71	0	70	0	1	0		0.682951	2.391867	FALSE	1.206762	0.67239	2	9	FALSE	0.273653	0.120865	0.146727	0.930703	-54.6548	-10.4182	-42.5131
14	1.64E+18	6.91E+18	518116239	2015	312.1433811	2.950897397	-3.7266	1.080953	0.96435	62	0	62	0	0	0			0	FALSE	1	0.423741	2	9	FALSE	0.179992	0.180794	0.08943	0.942024	-55.617	12.27039	-25.6346
15	1.64E+18	6.91E+18	200796881	2015	312.1499471	3.205912033	-3.72236	1.167898	0.98425	62	0	62	0	0	0			0	FALSE	1	0.849971	2	9	FALSE	0.36655	0.198018	0.122331	0.921814	-31.4527	-37.3687	58.43854
16	1.64E+18	6.91E+18	346699479	2015	312.1425548	3.189075215	-3.71974	1.150169	0.9797	62	0	61	0	1	0		0.20585	0.167672	FALSE	1.015331	0.652856	2	9	FALSE	0.166556	0.196414	0.093674	0.941988	-55.4612	13.84176	-22.2863
17	1.64E+18	6.91E+18	822965385	2015	312.1533747	3.999442966	-3.7072	1.428178	0.99085	62	0	62	0	0	0		0.890556	17.25141	FALSE	2.045433	0.958729	2	10	FALSE	0.179992	0.180794	0.08943	0.942024	-55.6177	12.26974	-25.6351
18	1.64E+18	6.91E+18	669911155	2015	312.1355708	21.86309884	-3.71545	9.033389	-0.9706	43	0	43	0	0	0			0	FALSE	1	0.043236	2	8	FALSE	0.144535	0.570819	0.419506	0.948509	100.9386	24.24191	-0.99545
19	1.64E+18	6.91E+18	1065804604	2015	312.1317649	21.91959742	-3.70926	9.055514	-0.96975	41	0	41	0	0	0			0	FALSE	1	0.050853	2	7	FALSE	0.161858	0.550036	0.421022	0.950245	136.9144	24.99335	-5.13972
20	1.64E+18	6.91E+18	340485901	2015	312.133152	25.22884576	-3.70803	11.3763	-0.70905	27	0	27	0	0	0		0.60432	0.226551	FALSE	1.046984	0.189226	2	6	FALSE	0.704382	0.356352	0.854107	0.968296	48.56501	13.42333	13.58105
21	1.64E+18	6.91E+18	904055603	2015	312.1467259	4.978188083	-3.71137	6.757577	0.8818	35	0	35	0	0	0		1.076909	9.604273	FALSE	1.190646	0.211329	2	7	FALSE	0.538666	0.177173	0.450403	0.933193	17.95588	-14.5557	20.42931
22	1.64E+18	6.91E+18	270927225	2015	312.1467864	25.59407409	-3.70701	11.54443	-0.7092	27	0	27	0	0	0			0	-5.36E-16	FALSE	1	0.180802	2	6	FALSE	0.704382	0.356351	0.854107	0.968296	48.56411	13.42245	13.58015
23	1.64E+18	6.91E+18	1019462870	2015	312.0594665	24.09072374	-3.7663	10.87785	-0.7079	27	0	27	0	0	0		1.618732	1.035805	FALSE	1.206155	0.077965	2	5	FALSE	0.704383	0.356351	0.854106	0.968297	48.56997	13.42837	13.586
24	1.64E+18	6.91E+18	157936401	2015	312.0690109	2.65145935	-3.75694	1.179736	0.7405	45	0	44	0	1	0			0	8.05E-16	FALSE	1	0.075135	2	7	FALSE	0.203394	0.206154	0.210558	0.938462	11.25355	11.00731	8.275051
25	1.64E+18	6.91E+18	468420179	2015	312.066464	14.75689289	-3.75232	18.78226	0.98675	44	0	44	0	0	0		0.589283	1.700937	FALSE	1.158783	0.713022	2	7	FALSE	0.698623	0.189464	0.772645	0.988808	-91.7916	-48.7595	-45.5446
26	1.64E+18	6.91E+18	252634356	2015	312.0474497	3.715399972	-3.75798	1.343479	0.9969	71	0	71	0	0	0		0.484194	9.923437	FALSE	1.534219	2.549269	2	10	FALSE	0.441447	0.290636	0.22555	0.908576	-37.1372	-43.3974	-56.1322
27	1.64E+18	6.91E+18	810786160	2015	312.0460114	21.84225026	-3.75484	9.025407	-0.97045	36	0	35	0	1	0		1.506641	1.2437	FALSE	1.091057	0.047559	2	6	FALSE	0.288571	0.479564	0.516595	0.950554	67.22894	21.82511	5.547845
28	1.64E+18	6.91E+18	282940965	2015	312.0564258	4.170324317	-3.75063	1.512999	0.9967	71	0	71	0	0	0		0.568451	32.5803	FALSE	2.516543	2.56468	2	9	FALSE	0.40739	0.289051	0.396025	0.941494	-69.5294	-48.3842	-45.8855
29	1.64E+18	6.91E+18	64449962	2015	312.0583843	3.09602411	-3.74098	1.254902	0.8766	51	0	51	0	0	0		1.874051	6.897879	FALSE	1.668933	0.149312	2	8	FALSE	0.219118	0.084415	0.256897	0.957187	-106.632	3.51686	-31.4251
30	1.64E+18	6.91E+18	729998705	2015	312.0782289	3.753855081	-3.73758	1.363497	0.99845	81	0	81	0	0	0		0.421768	26.4885	FALSE	2.0913	4.501054	2	11	FALSE	0.49042	0.349183	0.311544	0.913311	-42.3372	-46.2906	-54.7721

For non-professional individuals, understanding the definitions of these figures would be time consuming. One of the applications that this research aims to accomplish is to translate this publicly available data of different celestial objects into a visual mirror that would help individuals understand the logic and scope of this data. Similarly, typical diagrams of the solar system tend to disregard scale and distance in the models, but in cases that they do indicate scale or distance, as shown in **Figure 2**, it is impossible to view the 2D model from any other angle or distance, which is helpful in forming the mental imagery of scale and distance between objects.

Figure 2

Solar System Diagram with Units



Note. From "Owen, T. Chant (2023, March 10). solar system. Encyclopedia Britannica. https://www.britannica.com/science/solar-system"

With the use of virtual reality, scale, distance, and awareness of any other spatial indicator is present and easier to grasp. This could be helpful in a wide variety of cases, such as learning about planets and their sizes in a classroom setting. One thing to add, this software acts as base level software, which can then be built upon and used in many problematic areas. For example, if an engineer required a prediction of how a rocket will be affected when flying by a celestial object, the software can be modified to handle orbital and gravitational predictions, accurately mapping this data in real time, where then the engineer can visual the prediction in 3D space. In conclusion, it is evident that virtual reality can assist in the visualization of these different 3D scenarios, which supports the significance of this research.

After understanding the scope of this research through the different discussions in this section, including the introduction, literature review, research frameworks, and significance, it's important to understand and summarize the question being answered by this research. With the vast amount of information and data publicly available, a need for an application that can provide a platform for the visualization of this information and data is necessary. This research plays a role in understanding whether a platform like this is viable. It looks to create introductory software that serves as a framework for other applications requiring visualization or interactive tools in the future. This framework consists of various modules such as importing data from existing databases, mapping this data into a 3D environment for virtual reality, then

creating interactable elements that assist in different applications mentioned earlier in this section. The different modules can be modified, or other modules can be added to cater towards a solution for the problem at hand. In the next section, we will look at the current modules in more detail.

Methods

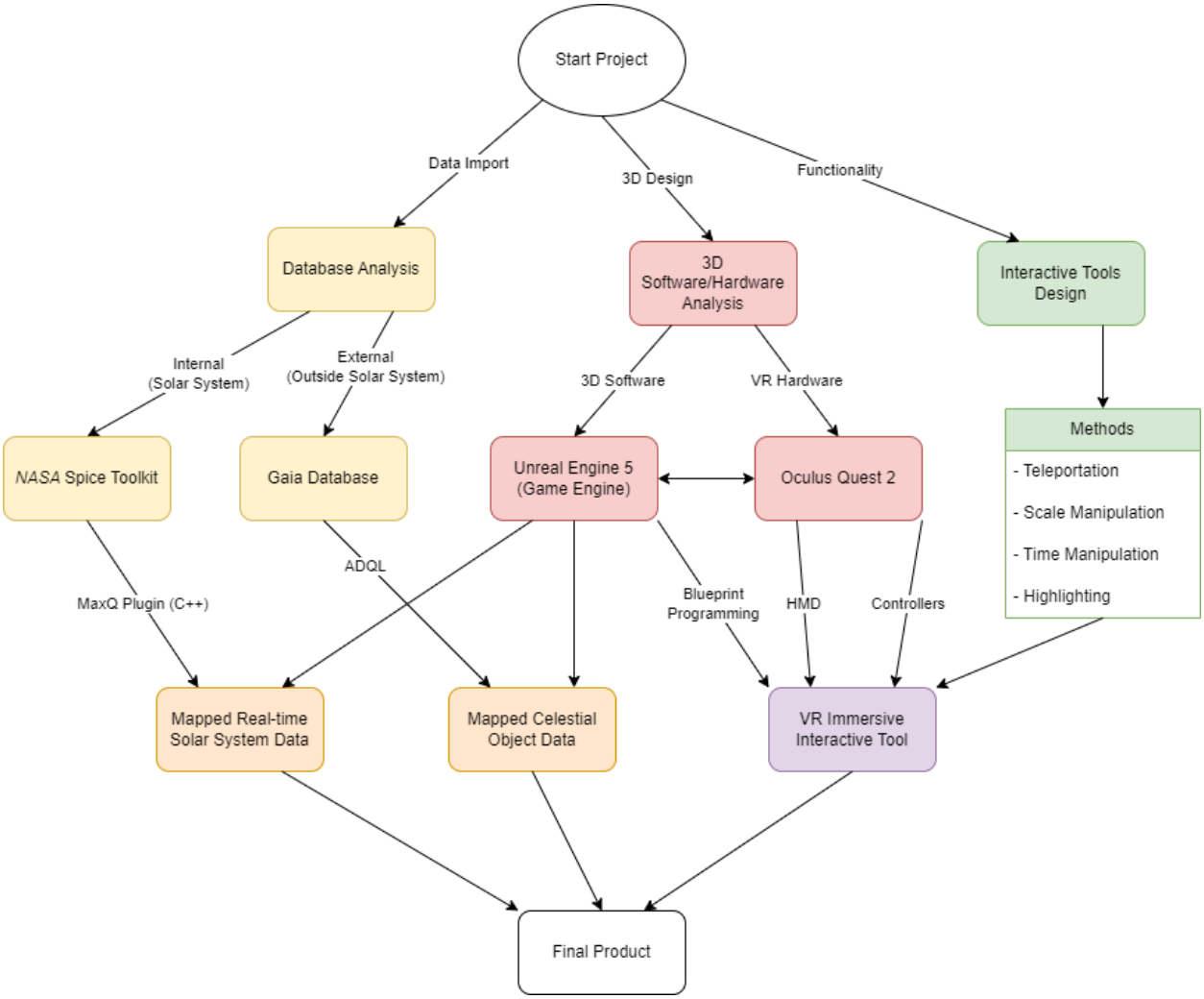
First and foremost, it is vital to discuss the main software involved in handling the bulk of the visual processing. Unreal Engine 5 (UE5) is an open-source software, known in the software development field as a "game engine" due to its original intention to assist in creating video game-like applications. As of now, it is being used for other applications as well, including architecture visualization, computer graphics, graphic design, cinematics, photography, environmental design, modeling, VR development as well as video game development. UE5 has many built in computer graphic features such as raytracing, real-time shadows and lighting, realistic textures, and much more. This allows a developer to solely focus on creating the application, rather than committing time to computer graphics, which is where it comes into this research. UE5 is built on the C++ programming language, but it hosts its own programming language known as Blueprinting. Blueprinting was the main language used throughout the research period, combined with more complex C++ programming for custom functionality. In this research specifically, UE5 served as the gateway between the data and virtual reality, because data was imported into UE5 which was then displayed in virtual reality through it. We will return to UE5 when we discuss the research process in more detail.

As stated in the previous section, the challenge of creating a framework design for this application is broken down into three main parts or modules, these being: importing the data, mapping this data into 3D, and creating interactive elements. These lay the groundwork for the application. It's important to understand that these modules are completely independent of each other, thus giving an easy ability to modify or incorporate new ideology and implementations into the framework. **Figure 3** describes these modules and how they relate to one another. We will discuss these modules in order, understanding their function, how they were implemented, software and hardware involved, and how to play a role in the overall goal

of the research. Throughout the discussion, you can refer to **Figure 3** to understand the overall idea and framework of this research.

Figure 3

Framework and It's Modules Diagram



Importing the data is the first module of the framework used in this application. Originally, there were two main goals in mind when considering the data involved in this application. These goals were to incorporate data on (a) objects in the solar system, and (b) external objects outside of the solar system. During this part, preliminary research was vital to have a good understanding of the current data we have on different celestial objects for both (a) and (b). This preliminary research included online searches, literature reviews, as well as discussions with professionals in the field of astrophysics who may be familiar with this data.

Starting with (a), after certain online research, it was chosen to use the NASA Spice Toolkit (NST). NST is an information system used by the planetary science and the engineering community to provide precision observation geometry with custom application programming interfaces (API) for a wide range of applications. This toolkit provides high-precision information on celestial objects in the solar system, making it perfectly applicable for (a). Incorporating this data inside of UE5 would prove to be simple, as a wrapper (conversion between two pieces of software) exists for UE5. MaxQ, the wrapper and a plugin for UE5, allows for NST integration inside of UE5, providing the same functionality as NST but in the C++ and Blueprinting languages. MaxQ was used to incorporate this data on (a) in real-time in UE5. NST has a wide variety of information on different celestial objects, but the main information that was needed for the goal of this module consisted of a few parts: position, orientation, and scale of these celestial objects (see Appendix A). This data is available in UE5 during runtime and was used to map the different objects, which in turn, allows the user to view this data in 3D and virtual reality at any point and time.

What is important to note here is that the main research goal is making an expandable visualization application that can be modified based on the use-case or problem. With the incorporation of NST, much of the calculations and information is already handled, and it takes some custom UE5 code to incorporate these functions. As an example, one use-case that NST can handle is complex orbital predictions. NST allows a user to input personal data on an object, such as a satellite, then calculate the ellipse of the orbit around an object the satellite may fly past at a certain period of time. This data is then available in UE5 during runtime, which can be seen visually in virtual reality. This example can be incorporated into this application with some simple coding and prior knowledge of the module or overall conceptual framework of the application, thus supporting one of the main goals of the research of making an expandible application.

For (b), discussions with professionals regarding this topic led to multiple conclusions. Dr. Bischoff and Dr. Bayliss, associate professors at the University of Cincinnati astrophysics department gave their expertise, opinion, and insight into these different databases that they believe should be incorporated into the application and would be of use. The main databases

that continually were of interest throughout the discussions included Gaia, SDSS, and the Dark Energy Survey. Gaia is an observatory mission conducted by the European Space Agency to create the largest 3D map of the milky way galaxy, which hosts information on nearly 2 billion celestial objects (1% of the milky way galaxy). The Sloan Digital Sky Survey, known as SDSS, is a survey consisting of information on distant stars and galaxies using spectral imaging and spectroscopy. Finally, the Dark Energy Survey is a survey mapping different celestial objects in hopes of understanding dark energy. For the goal of this module, it was decided to use Gaia, particularly Data Release 3 (DR3), to map different objects within the application, due to its vast range and consistency of data on celestial objects. What emerged from one of the discussions with the professors was the idea of merging two of these databases, thus providing information from both databases on any celestial object from within the application. In the astronomy community, this is known as cross matching an object between two databases. A complex algorithm is required to combine this information, so it was decided to focus on only one database, Gaia. Cross-matching could be a feature further expanded upon in the future, which supports the research goal of expandability, but was beyond the scope of the current module. Also, other databases could also be included in the module if required for a specific application. Now, with Gaia being the main database we would be working with, parsing this data, and extracting certain information was important to keep the scope of this limited due to several factors. Firstly, Gaia contains around 2 terabytes worth of data, which would be unnecessary for this application, and secondly, this would serve as a testing ground for what could be possible within our module. With that being said, to extract information from the Gaia database, the programming language Astronomical Data Query Language (ADQL) was used from within the Gaia archive. Approximately 1000 random objects were chosen with information regarding their positionality, specifically their parallax, inclination, and declination. Gaia provides a wide variety of data on these objects, however, only the positions were extracted for the purposes of testing this module. The positionality information of these celestial objects was then converted to be used in UE5, which then would be mapped (see Appendix B). As with (a), part (b) is mapped in real time in UE5 during the operation of the application, which is used as a visualization of the objects. A note is that the purpose of this research is to create an expandible or modifiable

framework, so this other data provided by Gaia can be incorporated in the future of this module in the overall framework if an application requires it.

The second module of the framework is mapping this imported data. With the data being readily available from the different databases, applying this information in UE5 is simple. What needs to be considered however, is that the data on objects inside of the solar system are separate from the data of the objects outside of the solar system (NST and Gaia respectively), and the importance of this lies with the type of data being measured. NST provides data given at any point of time that you require due to its prediction and calculation capabilities, whereas Gaia provides data from only one reference of time (when that data was collected). As a default, for this framework it was decided that NST maps the object the same moment in which the application exists, meaning celestial objects are shown exactly how they are now in real-time. In the next discussion of the module of the framework, time manipulation will further explain this concept for NST. For Gaia, because the data is from one reference of time, the data is mapped how it was captured (Gaia DR3 data was collected between 25 July 2014 (10:30 UTC) and 28 May 2017 (08:44 UTC) spanning 34 months). For NST data, the position, orientation, and scale are used to map the objects, and for Gaia, only the position is used to map the objects (see Appendix C). This module leaves many ways of expanding or modifying existing functionalities for any application, which ties back to one of the main goals of this research of expandability. For example, more Gaia data can be incorporated with proper optimization (limitations of this will be discussed in more detail later in this section), or scale for objects is different relative to each other, thus catering to an application that requires a visually appealing model.

Before moving onto the last module, it is important to talk about some of the hardware used throughout the period of this research. In recent years, virtual reality has seen many iterations of systems with notable companies including Valve, Meta, and Microsoft. In this particular research, it was decided to use the Meta Quest 2 for a multitude of reasons. Firstly, this system is the most accessible to individuals looking to get an entry-level virtual reality system. Secondly, it has easy compatibility with UE5 through the OpenXR platform. OpenXR is software responsible for connecting different virtual reality devices to platforms like Windows. Fortunately, UE5 has a plugin for OpenXR, making virtual reality development simple. The Meta

Quest 2 comes with a head mount display (HMD) which provides the virtual reality aspect to the user. The Quest 2 also has two controllers with different buttons serving different functions. This gives developers the ability to incorporate many interactive elements in their applications using these controllers, which is where this hardware comes into this research. For the last module of creating interactive elements, the development process consisted of programming different buttons on the Quest 2 to serve different functions, such as popping up of a menu or movement of the cursor.

The last module of the framework is creating interactive elements. This part of the framework caters to certain applications but can be modified or changed to cater to suit any application with specific needs. There were a few main interactions implemented which includes: teleportation, scale manipulation, time manipulation and highlighting. These interactions (besides highlighting) can be accessed through a simple menu as a part of this module. This menu appears when the user clicks the button “B” on their Quest 2 controller. Now, there will be a brief overview of each interactive function discussed previously.

Teleportation serves as a way of moving around the play space with a given model. With virtual reality, movement is restricted to a certain area, so teleportation is a way of bypassing this restriction. When the user chooses teleportation, they can choose from a list of celestial objects they want to teleport to. When the teleport occurs, the user is moved to the location of the celestial object.

Scale manipulation serves as a way of changing the scale of the given model. When the user chooses scale, they can choose whether they want “visual” or “realistic” scale. “Visual” scale refers to the celestial objects relative to each other are visually appealing, meaning the scale is not based off the real-life scale of these objects relative to each other to allow for better viewing of the model. In this example, the sun may appear smaller, with the planets appearing closer to the center of the system but at a size that is visible, and the Moon is at a comfortable orbit around the Earth. “Realistic” scale refers to the celestial objects relative to each other are realistically represented. This means what the user sees is exactly how the objects appear in real-life. This is useful in cases where you want to learn about the solar system and see the exact scale, distance, and other spatial indicators of the objects. Now in this case, generally the

planets appear small compared to the sun, and the distances are vast. The scale manipulation menu also contains a draggable slider, that the user can change with the cursor provided by the Quest 2 joystick button. This slider will dynamically change the scale of the entire model depending on the position of the slider. The exact size of the model is described in the scale menu. This gives the user the ability to change the scale of the model at will, which can be useful in cases where understanding the scale of the celestial objects is important.

Time manipulation serves as a way of changing the time of the solar system model. The NST time concept was mentioned earlier, however, it is important to expand on the understanding of time for this particular module. With each moment of time in the application, the NST data is updated to the specified or set time. As default, this time is set to now as stated in the previous module; however, the time could be set to any other time. For example, the time could be set to 4 months, 25 days, 3 hours, and 15 seconds into the future, and NST will calculate and predict the exact position, orientation, and scale of the planets at that point in time, the data will be mapped in real time, then this data could then be visualized through virtual reality. With that understanding, the time manipulation module allows for users to choose the point in time they want to view the model. They can choose days, months, or years to go into the future. A draggable slider is present that the user can change with the cursor provided by the Quest 2 joystick button. This slider will dynamically change the time of the entire model depending on the position of the slider, which updates the model in real time. There is also an indication of what time you are looking at in the time menu. An example where this could be useful is when a user wants to see the exact position of a planet 15 days into the future.

The last interactive function implemented in this module is called highlighting. Highlighting serves as a way of allowing the user to open optional information on various celestial objects in the model. When the user moves their controller over a desired object, a details panel appears above the object, giving the user additional information on the object such as the name, radius, position, and other relevant information. As soon as the user removes their controller from that area, the details panel becomes hidden. This is useful in cases where a user is learning various information about the different celestial objects in the solar system.

Now, it is important to mention that these interactive elements were chosen for applications as stated in the descriptions. If there is requirement for an application that needs other interactive elements to be included, this module can be changed or expanded upon to suit the need of that application.

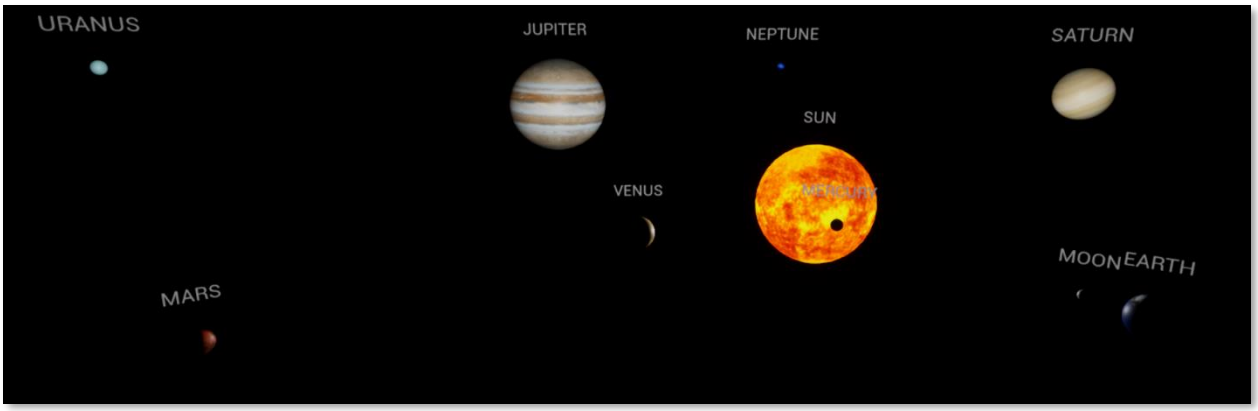
With the different modules of the framework of the application being discussed, it is important to understand the limitations of this application and its framework that were discovered or purposely implemented, as well as challenges faced during the research period. Firstly, incorporating more than a certain number of objects within the model causes undesired effects such as light flickering and performance drops in UE5. This restricts the current framework to a certain number of objects, which after testing proved to be approximately 1000-5000 objects. Now, in the future, it would be vital to optimize certain areas of the framework to support an increasing number of objects. Because of the large number of objects, it was also difficult to create interactive elements with the Gaia data. In the future, having various interactive tools like zoom in to the object, highlight, etc., that works with the Gaia data would open doors to many applications in different fields of space science and SED. The biggest challenge that was faced was, with the given time for conducting the research, the main modules discussed in this section were the only features that were feasible to create in this period of time. In the future, it would be important to expand upon the current system to cater to other applications in the different areas of space science and SED. Even so, with the current system, many use-cases in different areas of space science and SED can already be discussed, such as an educational learning tool or solar system visualization tool. Its good to mention that, although the modules discussed are the only ones incorporated in the current framework, they serve as the groundwork for other modules to be built upon. The goal of the research was to create an expandable 3D virtual reality model and application that serves as the basis for other applications that require other functionalities, and it is evident to state that that goal was accomplished.

Results

This section will discuss the results of this research. After the period of this research following the software development protocol and different conceptual frameworks, we have a fully interactive, accurate, easy-to use 3D virtual reality application for the Oculus Quest 2 that uses real publicly available data to map different celestial objects. An example of the solar system model created by the NST data is shown in **Figure 4**.

Figure 4

Solar System Model, SpaceXR



An example of the Gaia data mapped in the application is shown in **Figure 5**.

Figure 5

Imported Gaia Stars, SpaceXR

Here is an example of how the menu for the interactive elements looks like in **Figure 6.**

Figure 6

Interactive Elements

The current conceptual framework for the application is made up of different steps or "modules" which consist of the following: importing data, mapping the data, and interactive elements. The first module, importing data, implements various importation methods for databases containing data on celestial objects in the solar system, and outside of the solar system. The second module, mapping the data, implements ways of mapping this imported data into Unreal Engine 5, and bridges the gap between this data and virtual reality to create a visualization of this data. The last module, interactive elements, implements ways for the user to interact with the mapped model of different celestial objects within the solar system.

Conclusion

After understanding the methods, processes, and results, we can conclude this research. The current application, a fully interactive 3D virtual reality model, can assist in different problems faced by various areas of space science and SED, such as creating an educational learning tool in a classroom setting or solar system visualization tool in a work setting. This application also serves as a basic platform for other applications to build upon through the idea of "modules" or building blocks involved in creating the framework of the application. The current framework includes the following modules: importing data, mapping the data, and interactive elements. Each module can be modified or added upon to suit different space science or SED application's needs. One of the goals of the research was to create an expandable 3D virtual reality model and application that serves as the basis for other

applications that require other functionalities, and it is evident to state that that goal was accomplished.

Some of the limitations include restrictions on the number of objects permitted before software bugs and drops in performance in UE5 was visible. Optimization of the software can help increase the permitted number of objects and increase the overall performance of the application. Another limitation is, with the restricted time, other modules useful for various areas of space science or SED could not be developed. In the future, these modules can be added onto the current framework to assist in any visualization problem faced. One example of a problem like this goes as follows: an engineer requires a prediction of how a rocket will be affected when flying by a celestial object, so the software can be modified to handle orbital and gravitational predictions, accurately mapping this data in real time, where then the engineer can visual the prediction in 3D space. This example, along with many other problems faced by scientists, educators, engineers, and other fields under the space science and SED umbrella can use this software to create applications suitable for their particular needs to solve those problems.

Next Steps

This research set the basis for what this software can achieve. In the future, there will be several modifications and additions that would greatly benefit the capabilities of this application. Some necessary steps include optimizing the software to handle more objects. This opens the door to including more complex, higher fidelity, and visually stunning models, leveraging UE5's realistic computer graphical capabilities. Another necessary step is incorporating interactive elements with the Gaia data. At the moment, only the NST data can be interacted with, and translating this to other data will greatly expand the capabilities of this software. Also, one of the goals of the research, creating an expandable piece of software capable of handling modifications and changes, gives this software a plethora of directions it can go. The concept of modules gives any user, as long as they understand the modules and the framework of the application, the ability to create any solution for their problem. For example, creating an orbital prediction module that calculates and displays an object's orbit as it passes by a celestial object, which can then be visualized in virtual reality. The overall goal is to make

this software as accessible as possible, leveraging on the concept of modules, so polishing this concept will make this software go a long way. In the end, as long as this research inspires or benefits an individual, it has achieved its goal.

References

NASA. (2018, December 14). NASA's eyes: Eyes on exoplanets. NASA. Retrieved October 11, 2022, from https://eyes.nasa.gov/eyes-on-exoplanets.html

Owen, T. Chant (2023, March 10). solar system. Encyclopedia Britannica. https://www.britannica.com/science/solar-system

Acknowledgements

Project Advisor: Dr. Ming Tang, tangmg@ucmail.uc.edu

Astrophysics Discussions: Colin A Bischoff, bischocn@ucmail.uc.edu; Matthew Bayliss, baylismb@ucmail.uc.edu

Research Funding Source: University of Cincinnati Space Research Institute for Discovery and Exploration

Internal Solar System Data: Acton, C.H.; "Ancillary Data Services of NASA's Navigation and Ancillary Information Facility;" Planetary and Space Science, Vol. 44, No. 1, pp. 65-70, 1996. DOI 10.1016/0032-0633(95)00107-7

Charles Acton, Nathaniel Bachman, Boris Semenov, Edward Wright; A look toward the future in the handling of space science mission geometry; Planetary and Space Science (2017); DOI 10.1016/j.pss.2017.02.013

External Celestial Data: This work has made use of data from the European Space Agency (ESA) mission Gaia (https://www.cosmos.esa.int/gaia), processed by the Gaia Data Processing and Analysis Consortium (DPAC, https://www.cosmos.esa.int/web/gaia/dpac/consortium). Funding for the DPAC has been provided by national institutions, in particular the institutions participating in the Gaia Multilateral Agreement.

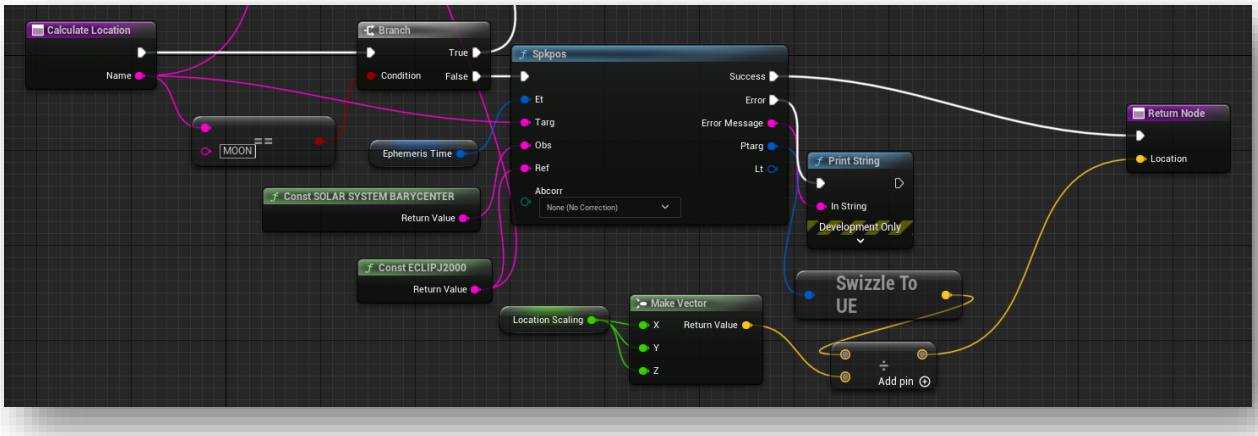
Appendix A

NST position, orientation, and scale code

NST functionality allows for precise calculations of different variables. In UE5 we use MaxQ's functions to access NST's capabilities. In this case, we wanted the position, orientation, and the scale of a particular celestial object. **Figure 1** demonstrates the code that calculates the location of the object. **Figure 2** demonstrates the code that calculates the size of the object. **Figure 3** demonstrates the code that calculates the rotation of the object. **Figure 4** demonstrates how the transform is calculated using the other functions.

Figure 1

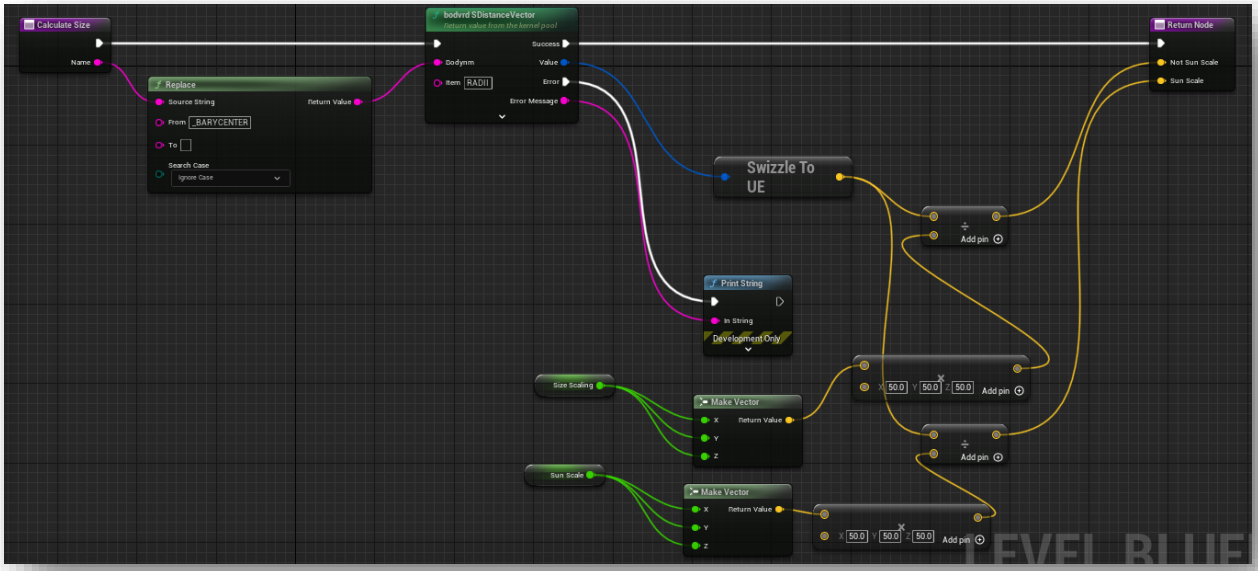
Location Blueprint Code



Note. The function starts with getting the name of the object, then uses the function "Spkpos" given by MaxQ to get the precise location information of that object. This data is converted to UE units, and scaled down to fit UE5's play area. It returns the location vector of that object.

Figure 2

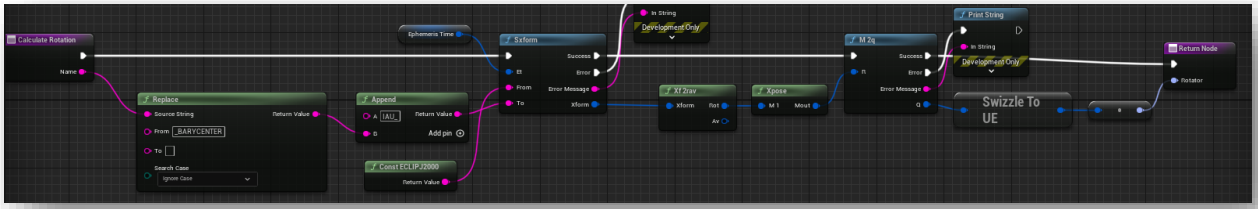
Size Blueprint Code



Note. The function starts with getting the name of the object, then uses the function “bodvrd SDistanceVector” given by MaxQ with the item parameter set to “RADII”. This gets the precise radius, or scale of the object. This data goes through multiple conversions and is scaled down to match the location. It returns the scale vector of that object.

Figure 3

Rotation Blueprint Code

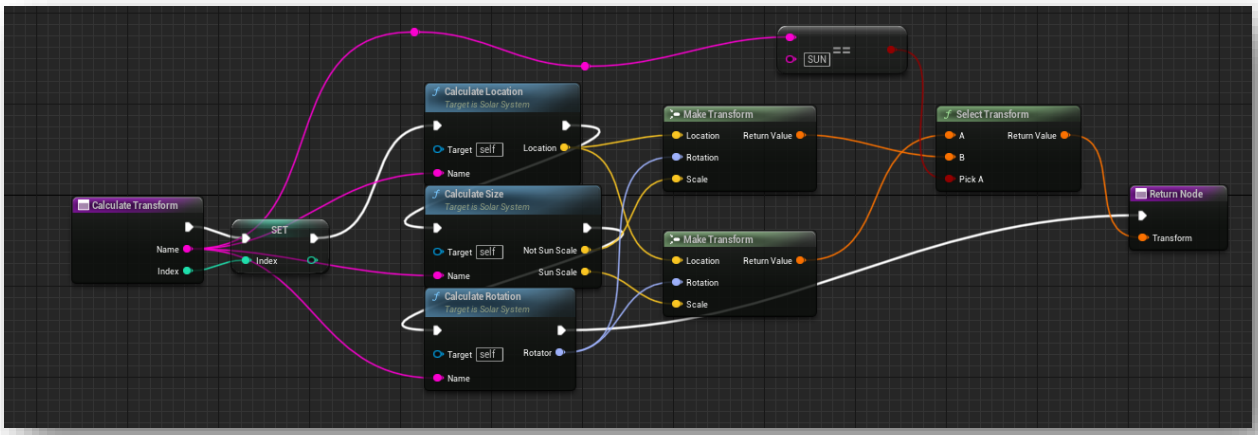


Note. The function starts with getting the name of the object, then uses the function “SxForm” to get the state transformation. This gives information on the rotation and angular velocity of an

object. This is then converted to a quaternion, then transformed into UE5's units, returning the rotator for that object.

Figure 4

Transform Blueprint Code



Note. The function starts with getting the name and index of a particular object. Prior to this function, objects were already initialized and given an index. The location, size, and rotation are calculated with the functions shown earlier, then the transform is created and returned. The object with the index is then given this transform to complete the mapping.

Appendix B

Gaia data extraction and import into UE5 code

To extract the data from the Gaia DR3 database, coding in ADQL is required. This language allows you to query up the specified data you require. In this case, we need the "source_id" (the object id), the "ra" (inclination), the "dec" (declination), and the "parallax". Different parameters were passed that prevented unwanted data from being extracted from the database as shown in **Figure 1**.

Figure 1

ADQL Gaia DR3 Code

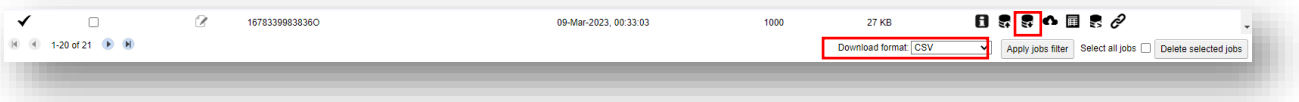
```
1 SELECT TOP 1000 source_id, ra, dec, parallax
2 FROM gaiadr3.gaia_source
3 WHERE parallax != 0
4 AND parallax > 0
```

Note. In the first line, we query the top 1000 objects with their associated "source_id", "ra", "dec", and "parallax". The second line indicates where we want to extract this data from, in our case DR3. The following two lines prevent unwanted data and outliers from being extracted.

Once we have this data queried, we can download this data into a .csv file as shown in **Figure 2**, where we can then manipulate the data.

Figure 2

CSV Download Query



Note. We can download the queried data by clicking the download button, and making sure the download format is set to "CSV".

Next we take this .csv file and calculate all of the needed values that we can use in UE5. First, we find the distance in parsecs which is calculated using $Distance = \frac{1}{Parallax}$. Then, using some trigonometry, we can calculate the X, Y, and Z associated with the inclination, declination, and distance. For X we use the formula $(Distance * \cos(Declination)) * \cos(Inclination)$. For Y we use the formula $(Distance * \cos(Declination)) * \sin(Inclination)$.

For Z we use the formula $(Distance * \sin(Declination)$. To convert from parsec to lightyear, we multiply the value by 3.262, and then from lightyear to kilometer by multiplying by 30856775812800. These calculations are and the associated values are shown in **Figure 3.**

Figure 3

CSV Data

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	source_id	ra_deg	dec_deg	parallax	distance_parsec	x1_parsec	y1_parsec	z1_parsec	x1_lightyear	y1_lightyear	z1_lightyear	x1_km	y1_km	z1_km
2	4038379047750904704	273.5826914	-35.72033019	3.831252876	0.261011223	0.099967586	0.027036158	0.239587952	0.326094267	0.088191947	0.781535898	3084677399737.02	834248666915.52	7392911709143.21
3	4038379047750906368	273.5842062	-35.71881712	0.459931003	2.174239167	0.835306766	0.227266559	1.994474486	2.724770669	0.741343515	6.505975774	25774873600083.30	7012713258265.48	61543052080987.50
4	4038379047750929664	273.5794017	-35.72154585	0.646441043	1.546931482	0.591332132	0.157839633	1.420707559	1.928925414	0.514872884	4.634348058	18246603021055.70	4870422177282.88	43838454651146.90
5	4038379047750930816	273.581518	-35.7195795	0.024809044	40.30788171	15.46966764	4.164286999	36.98747659	50.46205586	13.58390419	120.6531486	477344066410257.00	128496470340962.00	1141314273118900.00
6	4038379047750942080	273.5825531	-35.71791975	0.528860995	1.890856029	0.728263679	0.196850265	1.733845008	2.375596119	0.642125565	5.655802415	22471869061917.60	6074164502067.35	53500866691777.50
7	4038379047750946560	273.5778844	-35.72191376	0.379384942	2.635845258	1.007127791	0.267188339	2.421153491	3.285250853	0.871568362	7.897802688	31076716448484.10	8244570674089.60	74708990479676.90
8	4038379047750953344	273.5865756	-35.71414377	0.967435625	1.033660509	0.401126878	0.11015828	0.94626446	1.308475875	0.35933631	3.08671467	12377482132770.60	3399129354520.36	29198670311884.40
9	4038402927789291136	273.4244179	-35.75461548	0.536826902	1.862797852	0.676231217	0.07185242	1.73423309	2.205866229	0.234382593	5.657068339	20866315051577.30	2217134003277.32	53512841659757.70
10	4038402927789291264	273.4214969	-35.75656739	0.193622526	5.164688317	1.866115336	0.192771793	4.811907937	6.087268228	0.628821589	15.69644369	57582302577692.60	5948316004473.07	148479964451497.00
11	4038402927789294208	273.4172453	-35.75894453	0.142965512	6.994693959	2.513000372	0.248802122	6.522934172	8.197407213	0.811592523	21.27781127	77543089094079.90	7677231309919.47	201276717399739.00
12	4038402927789295104	273.42043	-35.75603372	0.567124281	1.763281937	0.638054442	0.065223803	1.64249675	2.08133359	0.212760044	5.357824399	19688302877731.70	2012596256285.51	50682153993892.90
13	4038402927789307008	273.4247821	-35.75149239	0.153680206	6.507018876	2.380881886	0.253855632	6.05047547	7.766436714	0.82807707	19.73665098	73466338606838.90	7833166313933.32	186698165131691.00
14	4038402927789307392	273.4166838	-35.75678331	0.462632374	2.161543496	0.780959643	0.076876972	2.014066099	2.547490354	0.250772681	6.569883616	24097896610229.60	2372175478772.78	62147586095816.30
15	4038402927789307776	273.4224458	-35.75285771	0.12067219	8.286913479	3.022415866	0.315117942	7.709645778	9.859120554	1.027914727	25.14886453	93262008781467.00	9723523688716.30	237894811363051.00
16	4038402927789312384	273.4196646	-35.75405218	0.32352626	3.090939203	1.124236548	0.114053356	2.876975699	3.667259621	0.372042047	9.384694731	34690315134021.90	3519318831421.00	88774194174032.50
17	4038402927789314048	273.4111104	-35.75940235	0.086260667	11.5927692	4.162484407	0.386339786	10.81280551	13.57802414	1.260240382	35.27137156	128440848169520.00	11921200163339.00	333648315435431.00

This csv file is then imported into UE5 as a data table, parsed and consequently mapped.

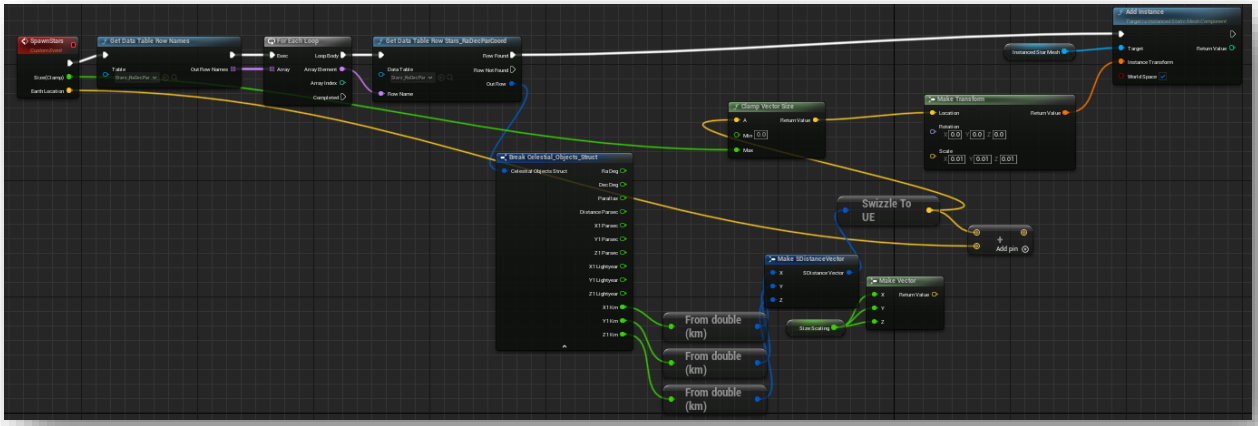
Appendix C

Mapping data code

For this particular part of the process, we must take the imported data from both the NST and the Gaia data, and properly map this data into UE5 units. For NST, we have the transform available from calculating it previously. Generally, the distance and the scale of the transform is manipulated to be smaller, due to the size restrictions UE5 has in place. This manipulation of the transform is what allows users to manipulate the scale during the interaction. For the Gaia data, this data is imported as a data table, which is then parsed and mapped. **Figure 1** demonstrates this process.

Figure 1

Gaia Celestial Object Map Function



Note. The function starts by parsing the data table in a for loop, and getting each column of each row. The kilometer value is then used but is scaled down and clamped to a certain value range. The clamp is in place to prevent extremely large values from positioning the star far enough away where the user can't see it. This value is then converted to a transform and applied to an object. The "Add Instance" function is important when dealing with a large number of objects. In this case, we have over 1000 objects, and rather than create a new mesh for each object, we re-use the same mesh, but create an instance. This technique allows for

many more objects to be created. Each time an instance is created, we apply this transform to that instance.