Simulating Solar and Lunar Eclipses

Documentation

Ridwan Haque

University of Georgia

Physics and Astronomy

Inseok Song and Nandana Weliweriya

4/21/24

Documentation

OVERVIEW

The Purpose of this project is to create an accurate solar system simulation that can be integrated into astronomy courses for visualizing astronomical concepts

GOALS

Determine which software is better suited for the development of this Project

Attempt to create a planetary simulation of the Solar System

BLENDER 3.0 SYSTEM SPECIFICATIONS

64-bit quad-core CPU with SSE2 Support

8 GB RAM

Full HD Display

Mouse, TrackPad, or Pen & Tablet,

Graphics Card with 2GB RAM, OpenGL 4.3

Less than 10 years old (generalizing)

UNREAL ENGINE 5.2.1 SYSTEM SPECIFICATIONS

Intel i5 core processor 5th generation and above

256 GB Storage SSD or Hard Drive

8 GB RAM

NVIDIA GeForce GTX 1650

Hard Drive to save file on a source other than the computer

Blender 3D hardware requirements: What you need to get started with Blender. CG Cookie. (n.d.). https://cgcookie.com/posts/blender-3d-hardware-requirements-what-you-need-to-get-started-with-blender

---

SOFTWARE COMPARISON

Blender

Blender is a robust 3D software package which is a free and open-source one for creating VR content. It includes Blender 2.80, the latest version, which is user-friendly and integrated with quality-level features. This programme enables developers to create virtual solid reality apps, high-quality animation content and visual effects. Blender is the ideal software development tool for game creators as it makes product testing and exporting simple.

Unreal Engine

The potent Unreal Engine offers a full set of developer and supporting VR tools. Gaming, film, architecture, automotive and transportation, broadcasting, and AR/VR simulation benefit greatly from the Unreal Engine. Innovative aesthetics, a rich entertainment experience, and immersive virtual worlds can all be created without restrictions.

PROCEDURE

- Download any version of Blender following version 3.6.2

- Open and Save a new project file using the file menu
- Download the official Earth, Moon, and Sun models from NASA. Make sure the file format is .gltf
- Save the planets in a folder in your file explorer where you will remember it.
- Import all three models into your 3D space using the import option menu.
- If done correctly, all of the 3D objects should be spawned where the world cursor is which is directly in the middle of the 3D space. In order to properly select each planet to move individually, you can use the Outliner tab located on the top right of your screen to easily select which object you want to manipulate.
- The Outliner window allows you to rename objects. If the NASA files that you import do not already have names, rename the objects by clicking on that object on the Outliner Window and double click and type to rename.
- Next hold “shift + A” to open the Add Menu and hover over the “Curves” tab. Next click the “circle” option. If done correctly this will spawn a circle into your 3D space
- You can easily manipulate the size of the circle by selecting it with “left mouse button” and holding down the “S” hotkey. For reference, to scale objects in 3D space, you must press “S” and move your mouse to manipulate the sizing. To rotate an object, you must press the “R” hotkey and move your mouse accordingly. Lastly, to move an object or transform/translate an object in 3D space, you must press “G” and move your mouse accordingly. For the purposes of this project we will be restricting transformation of objects to the X and Y axis. To restrict movement to the X and Y axis, click the object you want to move and hold “shift Z” and move your mouse accordingly.

- Use the Scale Hotkey to increase the size of the circle and ensure that the circle lays flat relative to the X and Y axis
- Next spawn another circle and scale it so that it is bigger than the first circle and ensure it lays flat.
- Next you will select the Earth Model in the 3D space and then navigate to the Properties Window where you will find the object constraints tab. On this tab you will select a constraint called “Follow Path”
- Afterwards a new menu will open and you will click the “Target button” and move your cursor to either the 3D space or the Outliner window to select the first circle.
- Congratulations! Now next you will animate this sequence
- The distance from the nearest points of the moon and the Earth is 363,300 kilometers. Blender uses “meters in measurement” so we will convert 363,300 kilometers into 3.633 gigameters and so each meter in Blender is 1 gigameter.
- The Earth rotates from west to east. When viewed from the North Star, Polaris, the Earth appears to rotate counterclockwise. This type of rotation is called prograde motion.
- Rotate earth 360degrees 365 times

Solar System DATA

https://phys.libretexts.org/Bookshelves/Astronomy__Cosmology/Big_Ideas_in_Cosmology_(Coble_et_al.)/08%3A_Dark_Matter/8.02%3A_Velocities_Mass_and_Gravity-_the_Solar_System

TASKS

- A brief 4-second video or animated gif showcasing your current simulation.
- A screencast video lasting approximately 2 minutes, demonstrating your simulation in action. Feel free to provide commentary as you showcase its features and functionality.
- Two paragraphs detailing your background, the project or simulation you're currently working on, and your motivations for being involved in the project.

Credit for 3D models

NASA Visualization Technology Applications And Development (VTAD)

REFERENCES

https://www.jpl.nasa.gov/edu/learn/video/solar-system-size-and-distance/

https://www.jpl.nasa.gov/edu/pdfs/scaless_reference.pdf

https://www.nasa.gov/wp-content/uploads/2015/01/yoss_act1.pdf

One astronomical unit (about 150 million kilometres; 93 million miles) is defined as the mean distance between the centres of the Sun and the Earth.

https://science.nasa.gov/sun/facts/

Sun diameter (1.4Million KM) (Blender scale 14.000 meters)

scaling down the Sun from 14 million km to 14 meters, that's a scale factor of 1 meter = 1 million km. Here are the diameters of the planets in the solar system at this scale:

Mercury: 4.879 km1 -> 4.879 mm

Venus: 12.104 km1 -> 12.104 mm

Earth: 12.756 km1 -> 12.756 mm

Mars: 6.792 km1 -> 6.792 mm

Jupiter: 139.820 km1 -> 139.820 mm

Saturn: 116.460 km1 -> 116.460 mm

Uranus: 50.724 km1 -> 50.724 mm

Neptune: 49.244 km1 -> 49.244 mm

CONVERSION

Mercury: 4.879 mm -> 0.004879 meters

Venus: 12.104 mm -> 0.012104 meters

Earth: 12.756 mm -> 0.012756 meters

Mars: 6.792 mm -> 0.006792 meters

Jupiter: 139.820 mm -> 0.13982 meters

Saturn: 116.460 mm -> 0.11646 meters

Uranus: 50.724 mm -> 0.050724 meters

Neptune: 49.244 mm -> 0.049244 meters

Moon: 3.475 km -> 0.003475 meters

The distance between the moon and Earth in astronomical units (AU) is 0.002473 scaling down such that 1 meter represents 1 million km, then 0.002473 AU (which is approximately 369,955,534 meters1) would be:

0.002473 AU = 369.955534 km

0.002473 AU would be approximately 0.36996 meters

UNREAL ENGINE SOLAR SYSTEM DOCUMENTATION

Download and install Unreal Engine 5.2.1

Documentation for Unreal EngineSetting up the Solar System (Mainly focused on the Sun, Earth and Moon)

Create a project in 3rd person

Click File, New Level, and set it to empty open-world

Click Window, Env. light mixer, create atmospheric light 1 and set intensity to 5

Create a new visual effect with PostProcessVolume and check the box for Infinite extent (Unbounded)

To create the sun

You can either import an already-made sun, import a premade model from Blender, or create your own

Add a mesh UV sphere

Increase segments to 128, rings to 64, and dimensions to (20, 20, 20) meters (X, Y, Z)

Right-click and show smooth

Scale the sun by 695,500 and set the location to 0

Set the y rotation to 7 - 7.25 degrees

Now the sun is scaled correctly with an accurate tilt

In order to make it more realistic you can add a slight wobble to the sun (a barycenter) due to the large planets exerting a gravitational force on the sun

Note: The barycenter of the sun does not follow a single circular or elliptical orbit due to the orbits of the other planets

Could have time to add solar radiation that shows in the simulation

There is a website called Solar Textures that we will use and it allows you to download unique textures to use for planets and stars

Apply the textures to the sphere to make the mesh sphere look like the sun

To create the Earth repeat the same steps to create a UV Mesh Sphere and add textures from Solar Textures

Place the earth in the correct position (still in progress)

Select the Earth model and in details and in the Details panel, set the Earth to have the Sun as its parent. This ensures that Earth orbits the Sun.

Could add a potential Day-Night cycle

Having a moving system is very heavy on the system, there is constant lagging, stutters, and crashes

Repeat the same steps for the moon and set earth as its parent

Can further be optimized by the blueprints

CONTINUED…

RESEARCH DOCUMENTATION 4/21/2024

TASKS **Completed(green) Work in Progress(yellow) Planned(Red)**

- **Simulate Penumbra and Umbra**

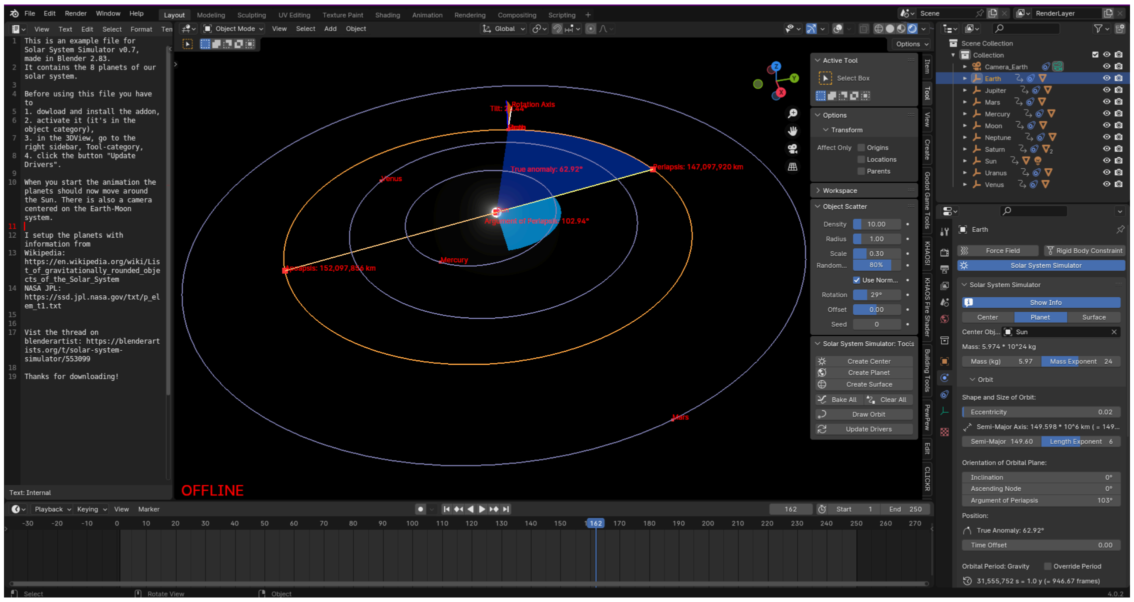
[Where is radius parameter for sun lamp in newer version of blender 2.8?]

For simulating Penumbra and Umbra I am experimenting with different methods such as using a cone or Blender's lighting mechanics to show lighting outlines to allow users to see lines of totality that the moon creates on the Earth.

- **Simulate accurate earth elliptical orbit**

Currently researching but plan to implement python script into blender

-GitHub - markus-ebke/SolarSystemSimulator: Blender Add-on for simulating solar systems



```
This is an example file for
Solar System Simulator v0.7,
made in Blender 2.83.
It contains the 8 planets of our
solar system.

Before using this file you have
to
1. dowload and install the addon,
2. activate it (it's in the
object category),
3. in the 3DView, go to the
right sidebar, Tool-category,
4. click the button "Update
Drivers".

When you start the animation the
planets should now move around
the Sun. There is also a camera
centered on the Earth-Moon
system.

I setup the planets with
information from
Wikipedia:
https://en.wikipedia.org/wiki/Lis
t_of_gravitationally_rounded_obje
cts_of_the_Solar_System
NASA JPL:
https://ssd.jpl.nasa.gov/txt/p_el
em_t1.txt


Vist the thread on
blenderartist: https://blenderart
ists.org/t/solar-system-
simulator/553099

Thanks for downloading!
```

Text: Internal

Density 10.00
Radius 1.00
Scale 0.30
Random... 80%
Use Norm...
Rotation 29°
Offset 0.00
Seed 0

Earth
Center Obj... Sun
Mass: 5.974 * 10^24 kg
Mass (kg) 5.97 Mass Exponent 24
Orbit
Shape and Size of Orbit:
Eccentricity 0.02
Semi-Major Axis: 149.598 * 10^6 km (= 149...
Semi-Major 149.60 Length Exponent 6
Orientation of Orbital Plane:
Inclination 0°
Ascending Node 0°
Argument of Periapsis 103°
Position:
True Anomaly: 62.92°
Time Offset 0.00
Orbital Period: Gravity Override Period
31,555,752 s = 1.0 y (= 946.67 frames)

OrbitalElementsBlenderSim Video

Dr. Song instructed us to test a solar system simulation github file and after downloading a python script addon and launching the project file I have much more tools to utilize for constructing accurate orbital mechanics in Blender. The video I have linked contains a simulation of the creator's project file that I have opened in my Blender application.

- **Latitude and longitude geopositioning**

  Currently researching but plan to implement python script into blender. I intend to contact the creator (Landon Ferguson) on how to utilize his coordinate system for my simulation.

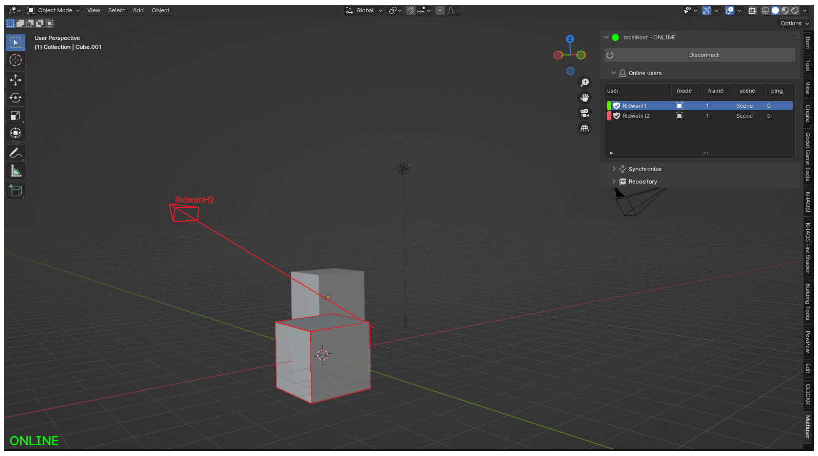
  -Script for Placing Mesh on Sphere with Provided Latitude and Longitude - Blender Stack Exchange

  -Geographic Coordinates to a 3D Sphere | Landon Ferguson

- **Multi user support**

  Swann Martinez / multi-user · GitLab

  Still needs further testing but I was able to host a server for multi user support. This would be a great way to collaboratively work together especially during the summer when we are doing remote work.



localhost - ONLINE

Disconnect

Online users

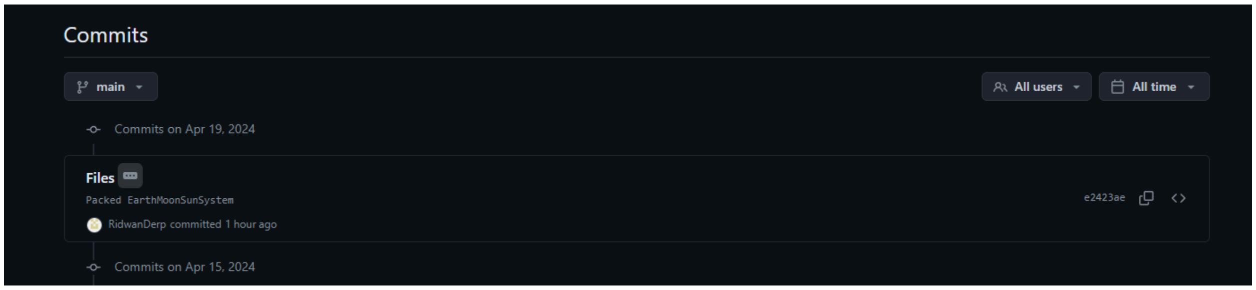
user	mode	frame	scene	ping
RidwanH		1	Scene	0
RidwanH2		1	Scene	0

Synchronize

Repository

- **Upload Blender large file system in Github**

Per Dr. Song's request I have learned to use Git LFS file upload for uploading large project files. I am also able to pack files so that anyone can access the current state of the blender script. This was done by downloading the github desktop app and uploading files into the repository in the file explorer. Repository can be accessed through VS Code as well.


Commits

main

All users

All time

Commits on Apr 19, 2024

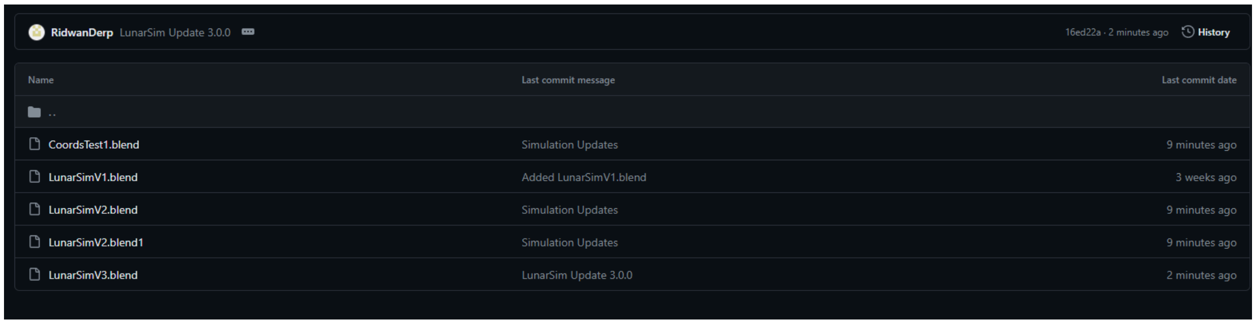
Files

Packed EarthMoonSunSystem

RidwanDerp committed 1 hour ago

e2423ae

Commits on Apr 15, 2024


RidwanDerp LunarSim Update 3.0.0 16ed22a · 2 minutes ago History

Name	Last commit message	Last commit date
..		
CoordsTest1.blend	Simulation Updates	9 minutes ago
LunarSimV1.blend	Added LunarSimV1.blend	3 weeks ago
LunarSimV2.blend	Simulation Updates	9 minutes ago
LunarSimV2.blend1	Simulation Updates	9 minutes ago
LunarSimV3.blend	LunarSim Update 3.0.0	2 minutes ago

I have also uploaded testing files for geolocation coordinate addon testing as well as the current state of the simulation which is version 3.0.0

- **Blender Camera improvements:**

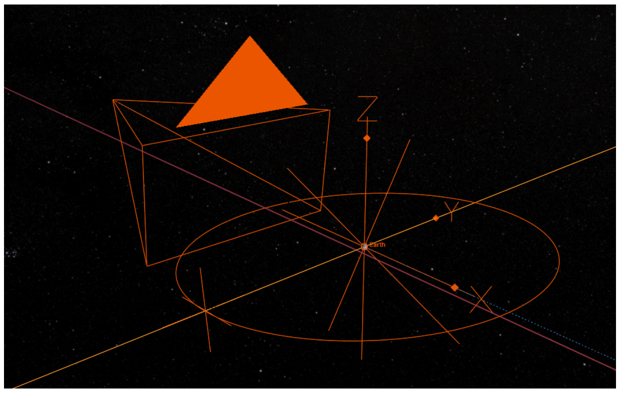
To see the eclipse (three moving objects blocking each other), we need to be able to fine-control the camera. One idea of doing it is:

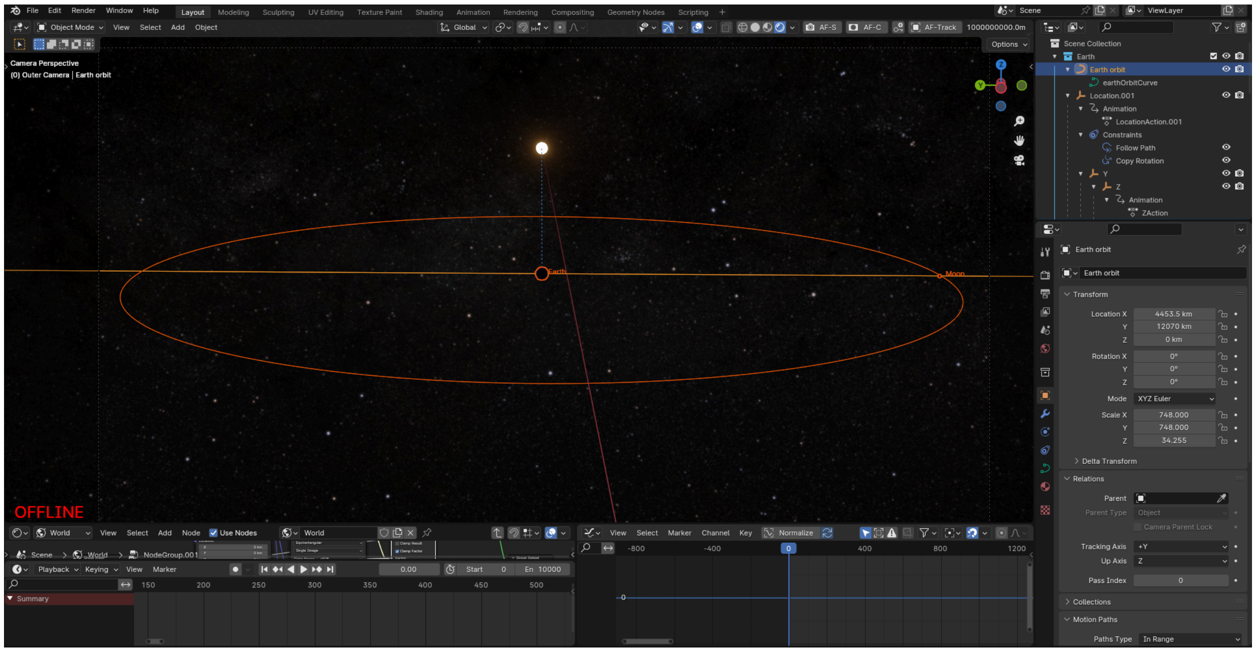
-Place a camera on a path that is slightly larger than Earth Orbit.

-The camera moves along with the Earth at the same orbital speed.

-Then, lock the camera to Earth.

-Or even lock the viewport to the Camera view (and this step may not be necessary).




File Edit Render Window Help
Layout Modeling Sculpting UV Editing Texture Paint Shading Animation Rendering Compositing Geometry Nodes Scripting
Object Mode View Select Add Object
Global
AF-S AF-C AF-Track 1000000000.0m
Options
Camera Perspective
(0) Outer Camera | Earth orbit
Earth
Moon
OFFLINE
World View Select Add Node Use Nodes World
Scene World NodeGroup.001
Playback Keying View Marker
Start 0 En 10000
Summary
View Select Marker Channel Key Normalize

Scene ViewLayer

Scene Collection
- Earth
  - Earth orbit
    - earthOrbitCurve
  - Location.001
    - Animation
      - LocationAction.001
    - Constraints
      - Follow Path
      - Copy Rotation
    - Y
      - Z
        - Animation
          - ZAction

Earth orbit

Transform

Location X	4453.5 km
Y	12070 km
Z	0 km
Rotation X	0°
Y	0°
Z	0°
Mode	XYZ Euler
Scale X	748.000
Y	748.000
Z	34.255

Delta Transform

Relations

Parent	
Parent Type	Object
	Camera Parent Lock
Tracking Axis	+Y
Up Axis	Z
Pass Index	0

Collections

Motion Paths

Paths Type	In Range

CameraCoMovingWithEarth Video Video link of the simulation

- **Make 2 paragraphs detailing background/project role**

This description is for the project website as per Dr. Song's request

Hi, My name is Ridwan Haque, and I am a first year computer engineering student attending the University of Georgia. I was first made aware of the research being done by the astronomy department when my Physics 1211 professor Dr. Weliweriya introduced the research to our class which is when I became interested in joining. The objective of our project aims to enhance the quality of instruction in astronomy courses at UGA, making complex astronomical physics more comprehensible and entertaining for students. The overarching goal is to provide students, particularly those facing challenges in grasping astronomical concepts, with a visually immersive learning experience. Currently students are restricted to learning astronomy through traditional methods of instruction that would involve 2D images in textbooks which do not provide an effective and captivating learning experience.

Working as a 3D design and simulation specialist for this project, my responsibility involves the creation of an accurate astronomical simulation of the Earth Moon and Sun system to replicate and predict solar eclipses to be viewed with VR tools in real time. The python scripted user interface I intend to code will allow astronomy students to select any location on Earth and view an accurate simulation of the virtual night sky. In addition, I am also working on calculating estimates for potential lines of totality by simulating umbra and penumbra. I am currently exploring various 3D software options such as Blender, Unity, and Unreal Engine to assess their capability in accurately rendering solar simulations, compatibility with VR, and effectiveness in producing software. At this point in the project I have utilized NASA's VTAD 3D resources and have scripted an orbital simulation of the Earth, Moon, and Sun system in Blender's 3D space. I have also imported an accurate skybox using NASA's official deep star maps. I am tasked with replication of accurate orbital velocities, planetary sizes, and accurate distances between celestial bodies. In addition, I have written detailed documentation of our progress which include sequential instructions so students and researchers can replicate our work. I've also uploaded simulation updates on our project github and have updates of our simulation posted on our project webpage.

- **Create a video of the Blender UI walkthrough for documentation**

Right now I have documented the process for the development of the current state of the blender simulation and intend to write more detailed explanations on recent developments and addons. In the future I will record comprehensive videos on how to construct the simulation. I also really want to learn how to update README files on github to add even more user friendly descriptive documentation.