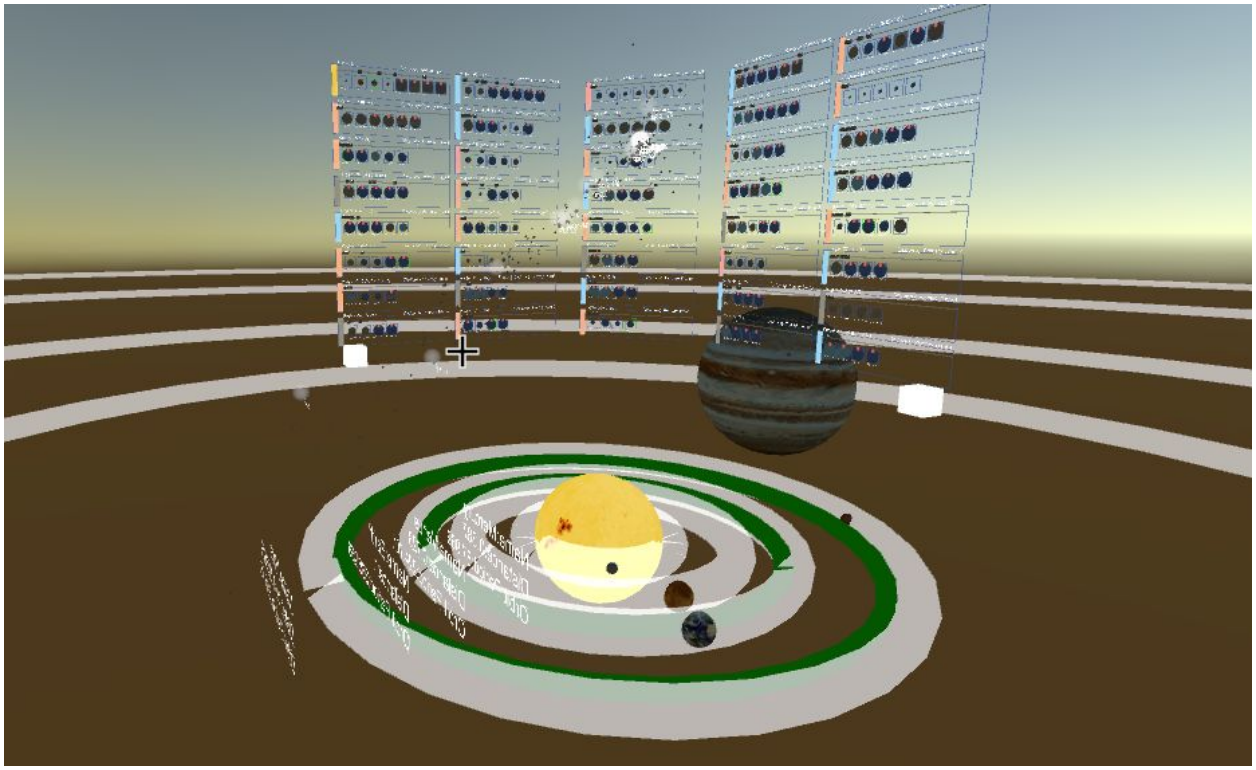
CS 491 Project 2
Group 5
UIC
Fall 2017

Team:
Dimitar Kirilov
Peter Hanula
Bartosz Kupiec

Project 2 CS 491,AR/VR, Here Comes the Sun



Description + Features:

Our application has the ability to interact with over 600 solar systems. This is done by allowing the user to view each solar system in three different ways:

1. Universe View: This feature shows all of the solar systems in our dataset along with where they are placed relative to ours. Our solar system is made slightly bigger and has a texture of the planet Earth so the user can find it easily and distinguish it from the other systems shown. The solar systems which are currently shown on the 2D view are highlighted in the Universe View and are labeled so that the user can tell which system is located where.
2. 3D View: This feature allows the user to examine a solar system in 3D view. The system's sun is shown in the middle, and its planets revolve around it in circular orbits. The user is able to change the rotational velocity as well as the orbital period of those planets through interaction. The user can also move the system relative to them in order to examine it from different angles.
3. 2D View: This feature shows a list of solar systems in a 2D panel. Each panel shows the system's distance from our own, the discovery method, and the planets of that system along with where they are located. Planets which are too far to be seen are slightly

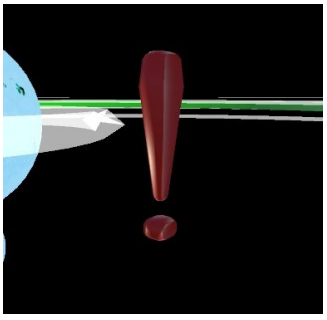
grayed out, planets which are too large to be viewed in the box have a red exclamation mark, and planets which are within a habitable zone have a green border. The user is able to scale the distance and the size of each planet so that they may compare everything and see planets into and out of view if they wish to. This view also textures planets which are similar to Earth the same way our planet is textured.

Interesting Features:

- 2D and 3D views depicting the solar systems in relation to each other.
    - Being able to see any view from 2D into 3D by selecting it
- Ability to search for planets using a virtual keyboard in the virtual reality environment, which go into the 2D view.
- Manipulating views by moving them, scaling, and making/removing them dynamically.
- All data overview which shows all the systems, and highlights the ones currently displayed.

Source Code & Assets Links:

- 3D Exclamation Mark - used when 3D planets are too big to be shown
    - https://www.models-resource.com/gamecube/starfoxadventures/model/20625/



- 2D Exclamation Mark - Used when 2D planets are too big to be shown
    - https://i.ytimg.com/vi/0UfhaE3CMHM/hqdefault.jpg

Github Repo: https://github.com/bkupie/cs491

How to Build:

Building from Unity:
From unity, the user can follow numerous guides on how to build from unity, and depending on your system make sure to use the one that you can use. The unity version used was Unity Version 5.5.0f3.Besides the source files used nothing extra is needed.

Running the executable:
To run the software, the user will need to be using a computer with and HTC Vive. Other head mounted displays such as the Oculus Rift are expected to not function properly, so please make sure an HTC vive is used. The system was developed along with SteamVR, which should be used when running, but may not be necessary but definitely highly recommended.

Download the file linked below, which includes an executable folder to get started. Then unzip the project in any way you please, and make sure that the executable is next to it's data folder. Then when ready, double click the executable, and get started exploring the solar systems. Enjoy!

Zipped Project:

https://uofi.box.com/s/ed1yt8652rrt3bw0b15ty5dl1k8o7hek

Zipped python script

https://uofi.box.com/s/metu00ejf2xyws4oowxmtub7yam7qilu

Data + Data Sources:

For our data, we combined three databases: NASA Exoplanet Archive, PHL Catalog, and this list of multi-planetary systems. We combined these CSV databases using a Python script, first by the "star + planet" key, and then by the "star" key (when merging the third database). After the merge, we used Excel to fill in any missing data. For example, the first database might have spectral ratings for some stars, while the second database might have spectral ratings for other stars. This step auto filled the relevant data.

This gave us 3600+ planets and 1400+ planet systems (600+ multi-planetary), with mostly complete data.

After this step, data was still potentially missing when all databases didn't have values for a specific field. We then ran the merged database through another Python script to calculate and derive as much missing values as possible, and then output the JSON files for our unity application. For example, correlating between planet mass and radius, or between spectral rating and star temperature. We also used distance, right ascension, and declination to calculate the xyz positions of stars in our application.

The final structures that our application reads in:

Example of system


```
{
        "ID": 0,
        "star": "Sun",
        "system": 1.0,
        "spectral": "G2V",
        "temperature": 5778.0,
        "distance": 0.0,
        "x": 0.0,
        "y": 0.0,
        "z": 0.0,
        "radius_Solar": 1.0,
        "mass_Solar": 1.0,
        "rotation_Period": 25.38,
        "luminosity": 1.0,
        "habitable_Inner": 0.95,
        "habitable_Outer": 1.4,
        "numPlanets": 8
}
```


Example of planet


```
{
        "ID": 0,
        "p_name": "Mercury",
        "p_discovery": "Naked Eye",
        "p_discoveryYear": "265 BC",
        "p_semiMajor": 0.387,
        "p_radius_Earth": 0.3829,
        "p_mass_Earth": 0.055,
        "p_rotation_Period": 58.646,
       "p_orbital_Period": 87.969,
       "p_temperature": 400.0
}
```