PlanetTXT: A Text-based Planetary System Simulation Interface for Astronomy Edutainment

Wanwan Li
Email:wanwan@usf.edu
University of South Florida
Tampa, Florida, USA

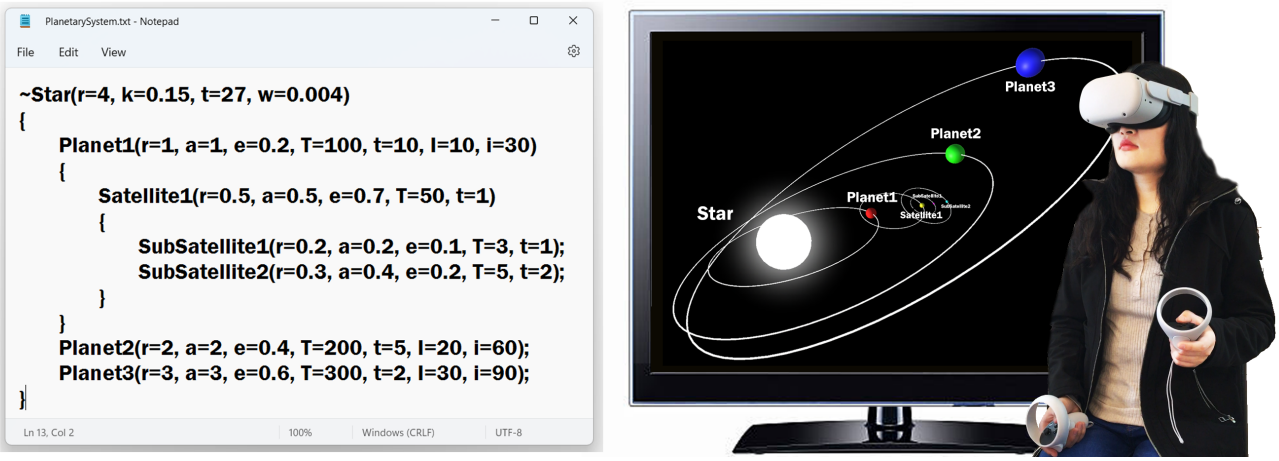


Figure 1: This figure shows an example that applies our proposed text-based user interface, PlanetTXT, to the interactive planetary system simulation in virtual reality for astronomy edutainment. Given the user input of C programming language-like scripts to describe the astronomic parameters for an arbitrary planetary system as shown on the left, the 3D virtual planetary system simulator is recursively constructed from the script and automatically loaded in the Unity Editor as shown in the right figure. After connecting with the SteamVR plugin, this planetary system simulator is loaded into the VR interactive platform. A user wearing Occulus Quest 2 can view this planetary system in an immersive virtual environment.

ABSTRACT

Planetary systems represent a set of gravitationally bounded non-stellar objects in or out of orbit around a star or star system. However, existing works for simulating exoplanetary systems other than the solar system are still limited. In this paper, we propose PlanetTXT, a plain text-based planetary system simulation interface for astronomy edutainment purposes. Using PlanetTXT, users can write simple scripts like C programming language to describe the astronomic parameters for an arbitrary planetary system. After putting this script into the interpreter implemented by us, the 3D virtual planetary system simulator is recursively constructed and automatically loaded in the Unity 3D Editor so that the user can view this planetary system simulation in an immersive virtual environment. In the end, we systematically test the capability of the PlanetTXT interface through a group of experiments, the results show that the PlanetTXT interface is good at simulating different planetary systems such as the solar systems, the Kepler systems, and some other exoplanetary systems.

IC4E 2023, February 01–04, 2023, Shenzhen, China

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

CCS CONCEPTS

• **Computing methodologies** → **Graphics systems and interfaces**; **planetary system simulation**; • **Human-centered computing** → **Systems and tools for interaction design**.

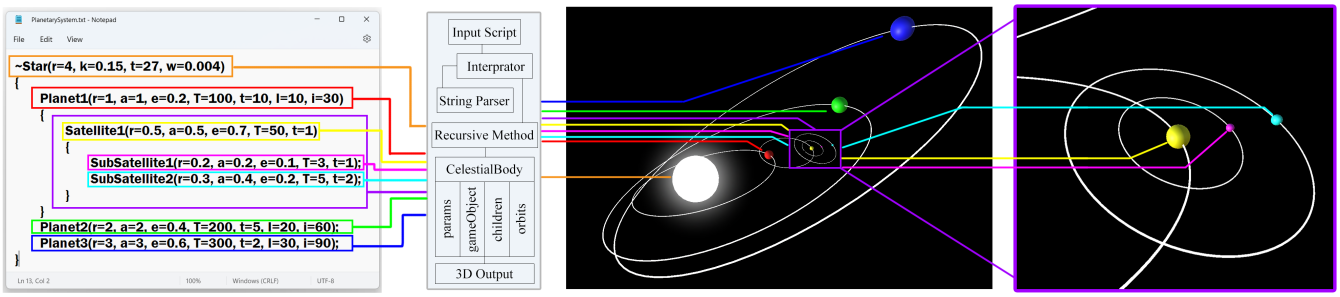
KEYWORDS

planetary system, physics simulations, text-based interface

ACM Reference Format:
Wanwan Li. 2018. PlanetTXT: A Text-based Planetary System Simulation Interface for Astronomy Edutainment. In *Proceedings of In 2023 14th International Conference on E-Education, E-Business, E-Management, and E-Learning (IC4E 2023).* ACM, New York, NY, USA, 7 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

The planetary systems, typically referring to the generalized versions of the solar system, represent a set of gravitationally bounded non-stellar objects in or out of orbit around a star or star system. The solar system itself is a member of the planetary systems. The term



(a) Input Script (b) Interprator (c) Generated Planetary System (d) Generated Planetary Subsystem

Figure 2: Overview of our approach.

exoplanetary system is sometimes used in reference to planetary systems other than the solar system. With the rapid progression of astronomy research, more and more exoplanetary systems are found. As a consequence, visualizing and simulating planetary systems become important for astronomy edutainment. Especially, simulating the solar system, which is the most familiar planetary system where we are living in, is extremely important for science education and attracts growing interest from researchers to propose different types of solar system simulators on different platforms.

Nowadays, researchers are closely following up with solar system simulation-related research projects for e-education and e-learning purposes. In 2017, Majgaard et al. [2] proposed a spatial visualization tool in augmented reality for solar system simulations with physical prototypes. Yu et al. [16] explored the learning effectiveness of the scale of the solar system using digital planetarium visualizations. In 2018, Wei et al. [8] designed and implemented the virtual simulation teaching platform for solar systems. Prima et al. [5] applied the PhET-based solar system simulation approach to improve students' understanding and motivation in learning solar systems. In 2020, Saputro et al. [6] discovers the effectiveness of using virtual reality media in physics education of solar systems towards cognitive learning outcomes. Zahara et al. [17] discovered that learning with the solar system scope application can enhance middle school students' learning efficiency.

Although plenty of successful work has been explored in simulating the solar system for immersive education and entertainment purposes, the work for simulating other exoplanetary systems is still limited. Also, there have been no efforts put into exploring a generalized planetary system simulation 3D modeling tool to automatically generate the different simulation scenarios with low input demands. Especially, traditional ways of modeling realistic planetary systems usually need to take into consideration all of the astronomical parameters and demand lots of manual effort from the simulation designer. Therefore, we propose PlanetTXT, a plain text-based planetary system simulation interface for astronomy edutainment purposes. Using PlanetTXT, users can write simple scripts like C programming language to describe the astronomic parameters for an arbitrary planetary system. After putting this script into the interpreter implemented by us, the 3D virtual planetary system simulator is recursively constructed and automatically loaded in the Unity 3D Editor so that the user can view this planetary system simulation in an immersive virtual environment. Figure 1 shows an example that applies our proposed PlanetTXT interface to the interactive planetary system simulation in virtual reality for astronomy edutainment. More specifically, in this example, the simulation is generated for an arbitrarily defined planetary system that does not exist in the real world and has shown the flexibility and scalability of applying our proposed interface for different education or entertainment purposes for astronomy science. Main contributions of our work include:

- We propose a clear and easy-to-learn grammar that can describe arbitrarily planetary systems with recursive features so as to help users efficiently encode specific or generalized planetary systems with given astronomic parameters.
- We design a recursively enumerable interpreter that can accurately extract the geometrical information within the given script so as to generate and simulate realistic planetary systems obeying the physics laws in the Unity 3D Editor.
- We conduct experiments to test our purposed interface in simulating different types of discovered planetary systems and conduct a preliminary study on deploying our simulator on virtual reality through the SteamVR plugin.

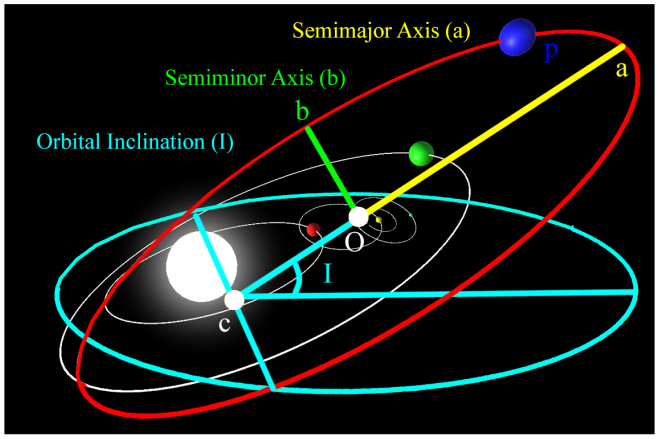
2 OVERVIEW

Figure 2 shows the overview of our approach. Given the plain text as the input script (a), we implement an interpreter (b) using C# scripts to automatically convert the input script into the CelestialBody data structures that stores the information necessary for the planetary system simulation. During the interpretation process, a data structure called StringParser is implemented for parsing the input script text word by word. We propose a method "$s' = NextWord(s,c)$" which is mathematically defined through the operator $\curvearrowright$ that represents the operation $s' = s \curvearrowright c$ which takes the input of current string s with begin index at i and output a substring $s' \in s$, s.t. $s_{i+|s'|+1} = c$. After the interpretation process through a recursive algorithm (for more details please refer to Section 3), the 3D planetary system (c) is automatically generated given the astronomic parameters specified in the plain text input script. Different colors in this figure correspond to different planets in this system. For example, the red box represents Planet1 and the blue box represents Planet3, etc. More specifically, a planetary subsystem is found recursively in this input script which is shown in (d) as highlighted in the purple box. In this planetary subsystem, the center satellite (star) is Satellite1 and the subsatellites (planets) are Subsatellite1 and Subsatellite2. This example shows a proof of concept for our proposed approach that doesn't exist in real world.

Algorithm 1 Create Planetary System: Recursive Algorithm

```
1:  procedure CreatePlanetarySystem(s, i)
2:      name ← (s ⌢′ (′)
3:      param ← (s ⌢′)′)
4:      celestialBody ← CreateCelestialBody(param)
5:      next ← s_{i←i+1}
6:      if next =';' then
7:          return celestialBody
8:      else
9:          if next =' {' then
10:             while s_{i←i+1} ≠'}' do
11:                 i ← i − 1
12:                 childSystem ← CreatePlanetarySystem(s, i)
13:                 celestialBody. AddChild(childSystem)
        return celestialBody
```

(a)



(b)

Figure 3: Technical Approach.

3 TECHNICAL APPROACH

In order to create the planetary system simulation from plain text input, we first define a grammar for the input script that contains celestial body name followed by fundamental astronomical parameters and a code block that defines the planetary subsystem or a semicolon to skip the planetary subsystem definition. More specifically, those astronomical parameters are celestial body's radius r in Earth Radius ($R_\oplus$), orbital semimajor axis a in Astronomical Unit (AU), orbital eccentricity $e \in [0, 1]$, orbital inclination $I \in [0°, 360°]$, equator inclination $i \in [0°, 360°]$, orbital period T (days), rotation period t (days). Generally speaking, these seven astronomical parameters are enough to generate realistic planetary system simulation and animation. Where the orbits geometry can be calculated through the following parametric equation:

$$\mathbf{p}(t) = \begin{pmatrix} x(\omega(t)) \\ y(\omega(t)) \\ z(\omega(t)) \end{pmatrix} = \begin{pmatrix} a[\cos(\omega(t)) + e]\cos(\theta) \\ a[\cos(\omega(t)) + e]\sin(\theta) \\ a\sqrt{1-e^2}\sin(\omega(t)) \end{pmatrix} \quad (1)$$

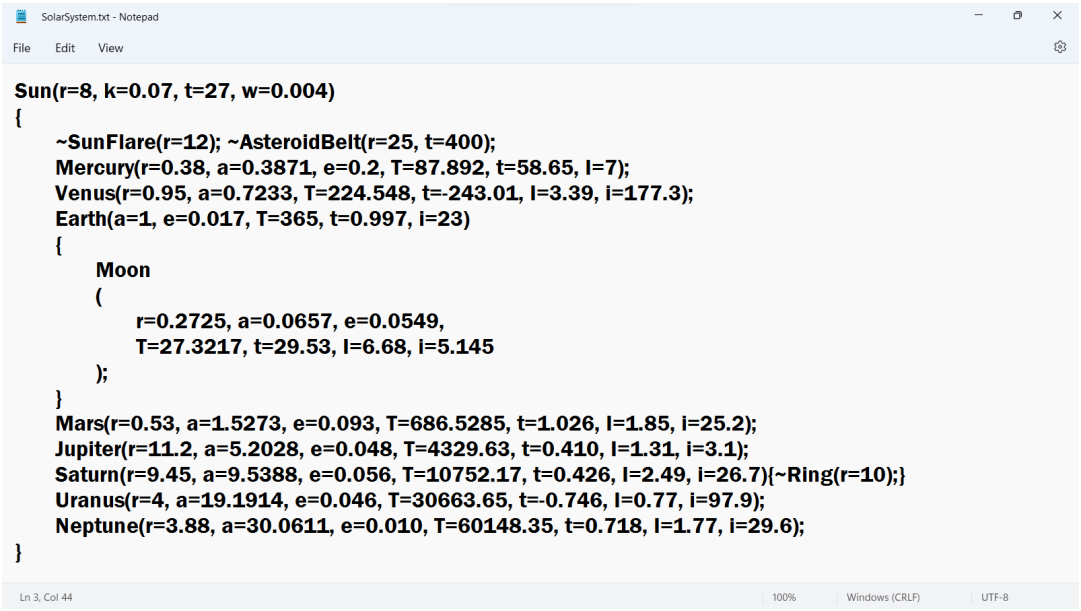
where $\theta = \pi \cdot I/180°$. As shown in Figure 3(b), the relation between semimajor axis a, semiminor axis b, and focal length c are $b = a\sqrt{1-e^2}$ and $c = ae$. In order to satisfy the Kepler's second laws of planetary motion [14] which states that a line segment joining a planet and the star sweeps out equal areas during equal intervals of time, the revolution angle $\omega(t)$ is calculated with integration:

$$\omega(t) = \frac{2\pi}{T}\int_0^t \frac{ab}{||\mathbf{p}(t)||^2}\mathrm{d}t \quad (2)$$

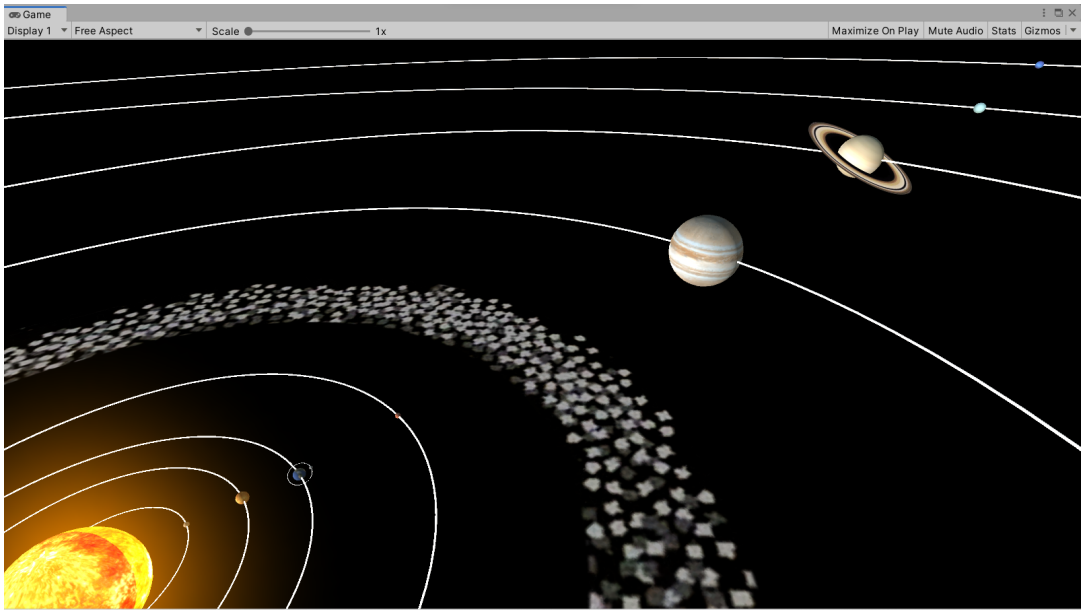
Due to the recursive structure of natural planetary systems, we define a C programming language-like recursive grammar for the input script. This way, a star can have a planetary system as its child system, similarly, a planet can have a satellite system as its child system and a satellite system can have a subsatellite system as its child system, and so forth. Figure 3(b) shows the algorithm that we proposed to interpret this recursive coding structure given arbitrary input plain text script s. We implement a data structure called StringParser which is for parsing the input script text word by word. For simplicity, we define an operator $\curvearrowright$ that represents the operation $s' = s \curvearrowright c$ which takes the input of current string s with begin index at i and output a substring $s' \in s$, s.t. $s_{i+|s'|+1} = c$. This operation extracts a substring from a string s starting at i and before a specific character c. Then, using this operation, we extract the celestial body name and the astronomical parameters. Given this information, our C# scripts load the material with the same name and generate the sphere geometry along with the orbit geometry calculated from the astronomical parameters. For a special case where the celestial body name is starting with a character of ~, the C# will directly load the prefab with exactly the same name but without ~. For the next step, we test the input string whether is followed by a code block that defines the planetary subsystem or followed by a semicolon. If a semicolon is detected, we skip the planetary subsystem definition and return to the generated celestial body itself; Otherwise, we generate the planetary subsystem recursively by calling this algorithm itself. Through these steps, after scanning the last line of the script, the whole planetary system is constructed automatically.

4 EXPERIMENT RESULT

In order to test our proposed interface, we have conducted a series of experiments to use PlanetTXT to generate and simulate different planetary systems. Figure 4 shows the experiment result of testing PlanetTXT on simulating the solar system. Figure 4 (a) shows the input of the user's plain text that contains the astronomic information of the solar system and Figure 4 (b) shows the output of the solar system simulator rendered in Unity3D Editor. In this example, as shown in the script, the solar system contains eight planets (e.g., Mercury, Venus, Earth, etc.), the SunFlare prefab which is showing the glow effect of the sun, and the AsteroidBelts prefab which is representing the asteroids between Mars and Jupiter. Both SunFlare and AsteroidBelts are manually designed prefab objects in Unity3D. The texture materials for different planets are downloaded from the website resource [1]. Orbital data for the planets as defined in the script is referenced from the website resource [7]. In this script, Earth has a satellite system that contains a Moon whose orbital data



(a) user's plain text input.

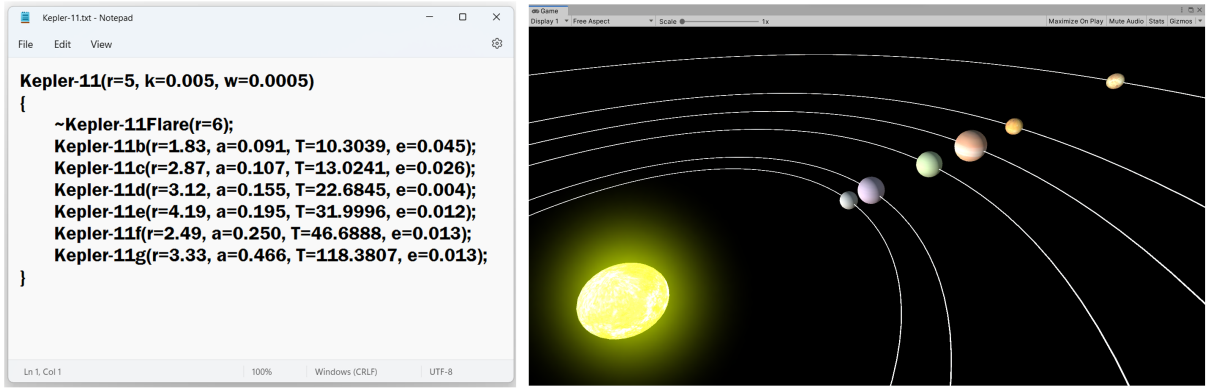


(b) solar system simulator output.

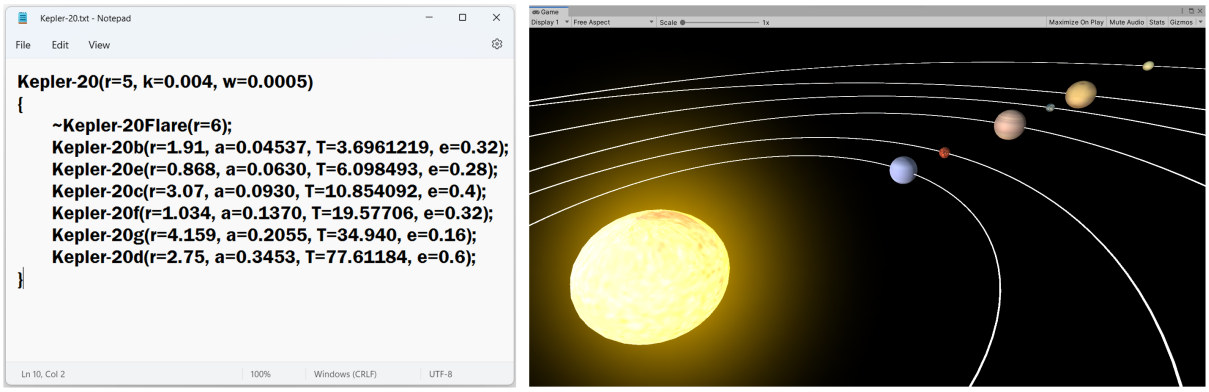
Figure 4: Solar System. This figure shows the experiment result of testing PlanetTXT on simulating the solar system. (a) shows the input of user's plain text that contains the astronomic information of the solar system and (b) shows the output of the solar system simulator rendered in Unity3D Editor.

is referenced from the website resource posted by NASA [4]. Especially, Saturn contains a child system that contains a Ring prefab which is manually designed given the ring texture. Both the prefab AsteroidBelts and prefab Ring are made from the quads rendered with the Sprite shader.

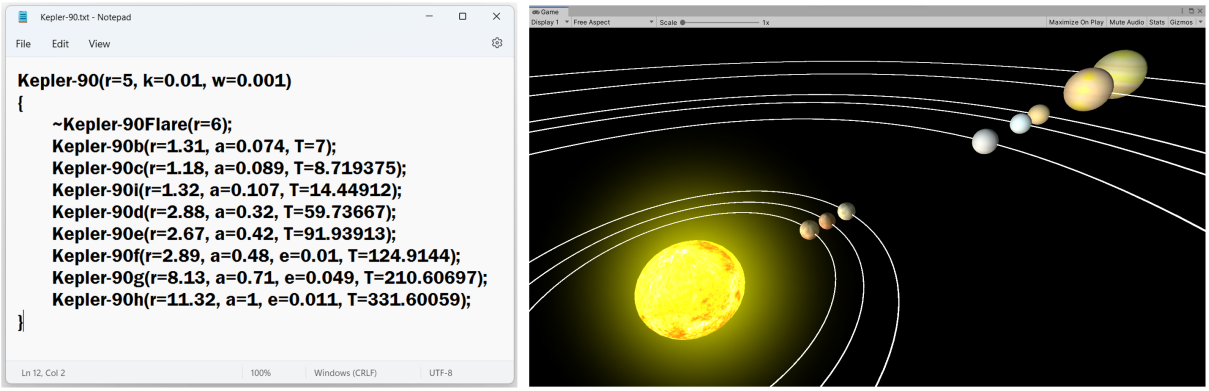
Besides the solar system, we also test our proposed interface on simulating other exoplanetary systems. There are lots of exoplanetary systems discovered by NASA's Kepler which are called Kepler planetary systems. NASA's Kepler [3] is the 10th in a series of low-cost, low-development-time, and highly focused discovery-class science missions, it was designed to discover Earth-like planets orbiting other stars in our region of the Milky Way. There are several famous Kepler planetary systems such as Kepler-11, Kepler-20, Kepler-90, etc. Figure 5 shows the experiment result of testing PlanetTXT on simulating these Kepler planetary systems. Figure 5 (a) shows the simulation of the Kepler-11 planetary system whose



(a) Kepler-11 planetary system.



(b) Kepler-20 planetary system.

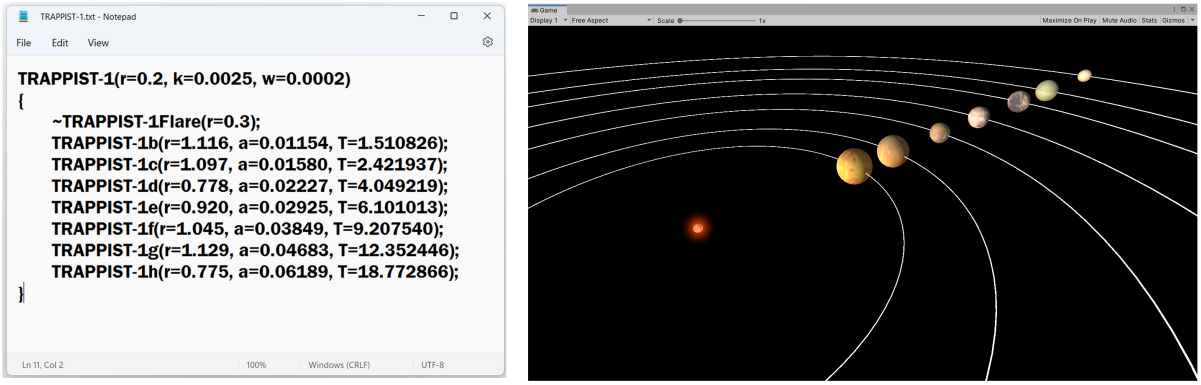


(c) Kepler-90 planetary system.

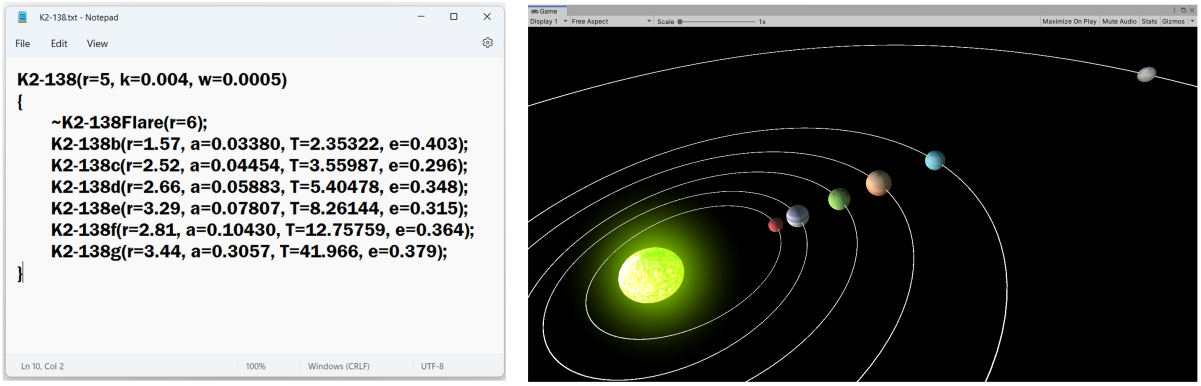
Figure 5: Kepler Planetary Systems. This figure shows the experiment result of testing PlanetTXT on simulating some Kepler planetary systems. (a) shows the simulation of the Kepler-11 planetary system, (b) shows the simulation of the Kepler-20 planetary system, and (c) shows the simulation of the Kepler-90 planetary system.

orbital data is referenced from the website resource [11], Figure 5 (b) shows the simulation of the Kepler-20 planetary system whose orbital data is referenced from the website resource [12], and Figure 5 (c) shows the simulation of the Kepler-90 planetary system whose orbital data is referenced from the website resource [13].

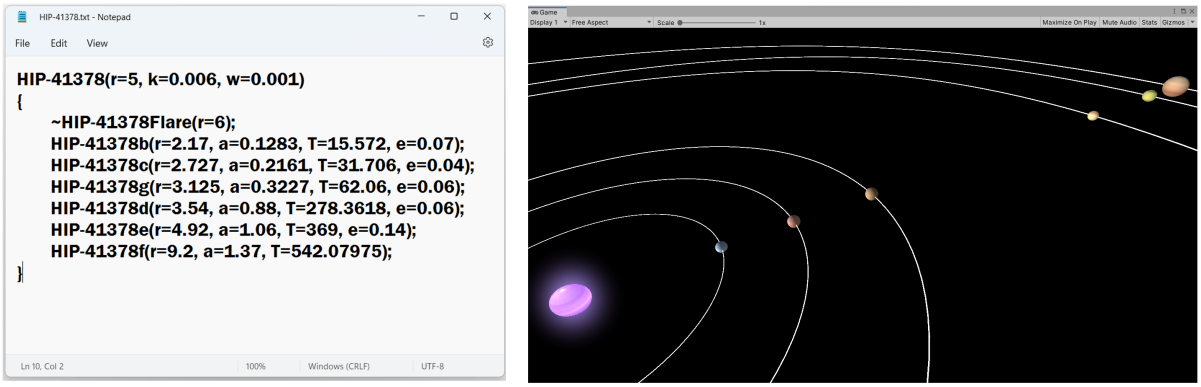
Figure 6 shows the experiment result of testing PlanetTXT on simulating some other planetary systems. Figure 6 (a) shows the simulation of the TRAPPIST-1 planetary system [15]. TRAPPIST-1 is an ultra-cool red dwarf star in the constellation Aquarius with a planetary system of seven known planets. Figure 6 (b) shows the simulation of the K2-138 planetary system [10]. K2-138 is notable for its large number of planets, all found through the efforts of citizen scientists. They are designated K2-138b, c, d, e, f, and g in order from their host star. Figure 6 (c) shows the simulation of the



(a) TRAPPIST-1 planetary system.



(b) K2-138 planetary system.



(c) HIP 41378 planetary system.

Figure 6: Other Planetary Systems. This figure shows the experiment result of testing PlanetTXT on simulating some other planetary systems. (a) shows the simulation of the TRAPPIST-1 planetary system, (b) shows the simulation of the K2-138 planetary system, and (c) shows the simulation of the HIP 41378 planetary system.

HIP 41378 planetary system [9]. In 2016, the K2 Kepler mission discovered five planets around HIP 41378, with sizes ranging from 2 times the size of Earth to the size of Jupiter, out to about 1 AU for the outermost planet. All of these simulation results show a coincidence between our simulations and the ones posted by NASA such as the orbit shape, planet sizes, revolution behavior, etc.

User Study. We tested our PlanetTXT interface in the virtual reality platform through a preliminary user study. After connecting PlanetTXT interface with the SteamVR plugin, this planetary system simulator is loaded into the VR interactive platform automatically. Figure 7 shows the user study process. During this study, a user wearing Occulus Quest 2 is sitting on a chair and observing different

planetary systems as simulated in Section 4 in a black universe-like immersive virtual environment. In this example, the user is view the solar system automatically generate with our approach. After the study, the user claimed that this VR planetary simulator is immersive, impressive, and has a potentially high value for astronomy edutainment. Video of this user study can be found through this URL link: https://youtu.be/R470Df1NmBE.

5 CONCLUSION

In this paper, we propose PlanetTXT, a plain text-based planetary system simulation interface for astronomy edutainment purposes. Different from the traditional stimulation interface that is created manually which demands lots of manual effort from the program designer, instead, using PlanetTXT, users can simply write plain text scripts to describe the astronomic parameters for an arbitrary planetary system. After putting this script into the interpreter implemented by us, the 3D virtual planetary system simulator is recursively constructed and automatically loaded in the Unity 3D Editor. In the end, we systematically test the capability of the PlanetTXT interface through a group of experiments, the results show that the PlanetTXT interface is good at simulating different planetary systems. After connecting with the SteamVR plugin, this planetary system simulator is loaded into the VR interactive platform. A user wearing Occulus Quest 2 can view this planetary system in an immersive virtual environment.

Future Work. In future work, we will test our PlanetTXT interface to generate more types of exoplanetary systems and let students of different ages try this interface to simulate their own planetary systems. More systematical user studies will be conducted to gather feedback from more users. From another perspective, we will test the learning curve of our text-based interface for users who have no coding experience. This may lead to another discovery that our PlanetTXT can not only help students learn about the universe, but also inspire the students to do coding simply by tuning the parameters in these planetary scripts. In that way, our proposed PlanetTXT interface may inspire text-based programmable scientific toy designs for children's creativity education.

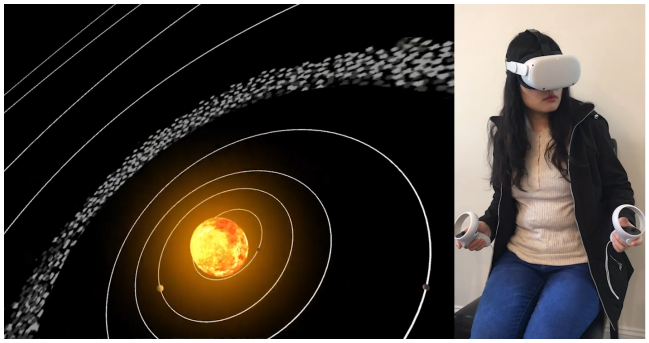


Figure 7: Preliminary User Study Result.

REFERENCES

[1] INOVE. [n.d.]. Solar Textures. https://www.solarsystemscope.com/textures/.
[2] Gunver Majgaard, Lasse Larsen, Patricia Lyk, and Morten Lyk. 2017. Seeing the unseen—Spatial visualization of the solar system with physical prototypes and augmented reality. *International Journal of Designs for Learning* 8, 2 (2017).
[3] NASA. [n.d.]. Kepler. https://solarsystem.nasa.gov/missions/kepler/in-depth/.
[4] NASA. [n.d.]. Moon Fact Sheet. https://nssdc.gsfc.nasa.gov/planetary/factsheet/moonfact.html.
[5] EkaCahya Prima, Aldia Ridwani Putri, and Nuryani Rustaman. 2018. Learning Solar System Using PhET Simulation to Improve Students' Understanding and Motivation. *Journal of Science Learning* 1, 2 (2018), 60–70.
[6] Sigit Dwi Saputro and Agung Setyawan. 2020. The Effectiveness Use of Virtual Reality Media in Physics Education of Solar System Towards Cognitive Learning Outcomes. *JPI (Jurnal Pendidikan Indonesia)* 9, 3 (2020), 389–400.
[7] Windows To Universe. [n.d.]. Orbital Data for the Planets & Dwarf Planets. https://windows2universe.org/our_solar_system/planets_orbits_table.html.
[8] Li Wei and Huimin Huang. 2018. Design and implementation of virtual simulation teaching system of solar system. In *2018 International Symposium on Educational Technology (ISET)*. IEEE, 29–32.
[9] Wikipedia. [n.d.]. HIP 41378. https://en.wikipedia.org/wiki/HIP_41378.
[10] Wikipedia. [n.d.]. K2-138. https://en.wikipedia.org/wiki/K2-138.
[11] Wikipedia. [n.d.]. Kepler-11. https://en.wikipedia.org/wiki/Kepler-11.
[12] Wikipedia. [n.d.]. Kepler-20. https://en.wikipedia.org/wiki/Kepler-20.
[13] Wikipedia. [n.d.]. Kepler-90. https://en.wikipedia.org/wiki/Kepler-90.
[14] Wikipedia. [n.d.]. Kepler's laws of planetary motion. https://en.wikipedia.org/wiki/Keplers_laws_of_planetary_motion.
[15] Wikipedia. [n.d.]. TRAPPIST-1. https://en.wikipedia.org/wiki/TRAPPIST-1.
[16] Ka Chun Yu, Kamran Sahami, and James Dove. 2017. Learning about the scale of the solar system using digital planetarium visualizations. *American Journal of Physics* 85, 7 (2017), 550–556.
[17] Atika Zahara, Selly Feranie, Nanang Winarno, and Nurhadi Siswantoro. 2020. Discovery Learning with the Solar System Scope Application to Enhance Learning in Middle School Students. *Journal of Science Learning* 3, 3 (2020), 174–184.