High Altitude Balloon Simulation Project

Huanjia Liu
Colorado State University
Dr. Azer Yalin
sjmxx1414@gmail.com
April 6th, 2021

Abstract

This project is a virtual reality (VR) high-altitude balloon simulation built in Unity3d. In this simulation, users experience an immersive view of the entire flight of a high-altitude balloon based on data collected in a previous DemoSat project. The simulation pulls real satellite images of the balloon's location based on its latitude and longitude and updates over the course of the flight. This is therefore helpful to students and researchers by providing a new way to visualize the field of view and trajectory of high-altitude balloons. Importing the user's own data is supported to allow VR simulation of future missions.

1. Introduction

High-altitude balloons are crewed or uncrewed balloons, usually filled with helium or hydrogen, that are released into the stratosphere, generally attaining between 18 and 37 km (11 and 23 mi; 59,000 and 121,000 ft) above sea level. The most common type of high-altitude balloons are weather balloons. Other purposes include use as a platform for experiments in the upper atmosphere.[1]

However, due to weight limitations, balloons are not always allowed to carry cameras. Therefore, A simulation program was designed to help users intuitively experience the path and operation of the high-altitude balloon. In this simulation program, users can use our preset data, which is derived from DemoSaT launch data on July 30, 2019 , or import their own experimental data. The program will integrate the trajectory route based on the data, display the sensor data when the balloon is in each position, and call the satellite map along the way. Moreover, VR mode is added into simulation, from [2], immersive VR education allows for more intuitive access to information and absorption of knowledge. Thus, it can not only help users get a more physical experience in their real, but also contribute to education.

2. Methods

2.1. Data Processing (Python)

Before starting the project, data pre-processing is needed. Different high-altitude balloons use different kinds of sensors, though most common sensors save export data in raw text or *xlsx* format. Thus, we use Python to extract some standard parts, to eliminate differences and transfer the data into *JSON* format.

In the code developed for this project, four necessary data are set: time, longitude, latitude, and altitude, and two optional data: temperature and pressure. When the user inputs the column number of these data, the Python library “pandas”, a powerful data analysis tool, will read the raw data file. Then, the files are compared and, after processing, normalizing, and padding, the final *JSON* file is output. Normalizing and padding are two indispensable steps, the former unifies all data to the same order of magnitude for easy import into the simulation later. The latter fills in the incomplete parts of the data provided by the user and ensures that the simulation import and run steps are error-free.

A crucial consideration during the data processing step is that the function from pandas library *pd.read_excel* will sort column lists automatically. In order to make label and col consistent, we need to sort both label and col list manually in advance.

Readers can access the whole code In Appendix A. Instruction has been written inside the code. To generate *JSON* data, just running the code and following the guidance.

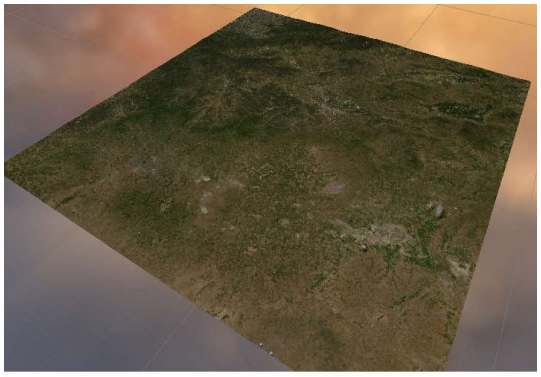

Figure 1. Online Maps v3 By Infinity Code (used for detail view)

2.2. Calculations

To simulate altitude balloon position in Unity3d, x and z-axis are set static, only changing the value of the y axis to achieve height simulation. As for the latitude and longitude, the changes will be applied on the Earth Terrain.

In more detail, in our project, the Earth's terrain is split into two parts. The first one is based on a Unity3D Plugin called *Online Maps v3* produced by Infinity Code (Figure 1), which is a plane like the figure. Users can input the center position (longitude and latitude) of the plane to change the Earth's true view, as shown by this plane. It is used to simulate the nearby detail view.

The second part is a 3D 64K Earth model (Figure 2.a), which is used as the overhead view. Here our Earth is set at static position. To guarantee the overhead view and nearby view in the same latitude and longitude, we need to apply a transformation based on Euler angles.

From the Kim and Chae spoke about the Euler angles in Cartesian coordinate system [3], the Euler angles (Figure 2.b) are three angles introduced by Leonhard Euler to describe the orientation of a rigid body with respect to a fixed coordinate system. They can also represent the orientation of a mobile frame of reference in physics or the orientation of a general basis in 3-dimensional linear algebra. For example, the plane above is our detail view and the sphere are our overhead view. Our destination is the redpoint. If we want to move to that place, we need to rotate the Earth at a certain angle in the green arrow's direction (Figure 2.d).

Thus, the problem is simplified into the relationship between position (latitude and longitude) and Euler angle.

The diagram (Figure 2.e) above is the solid geometry diagram to solve this relation problem. The angle α is longitude and the angle β is latitude. Point B is the radius of Earth, here earth is assumed to be a perfect sphere, and the radius is 1.

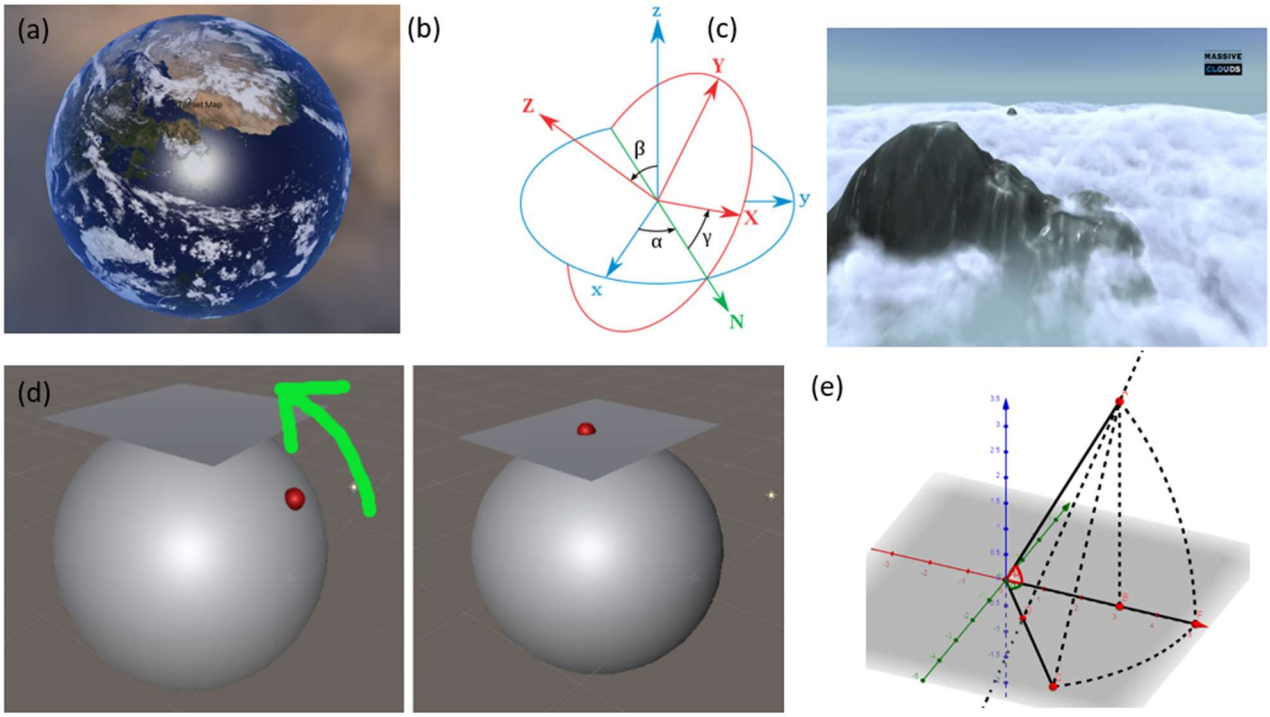

Figure 2. a) 3D 64K Earth Model (used for overhead view). b) Classic Euler angles geometrical definition (From WiKi). c) Massive Clouds powered by "MEWLIST" d) Combination of overhead view and detail view (overhead view rotates follow the green arrow to align the coordinates with detail view). e) The specific geometric diagram of the mathematical analysis part.

Arc AB is the length between the destination point and 0-degree latitude. Arc BC is the length between the destination point and 0-degree longitude. OA = OC = OB = 1. Then we can get the length of AB.

$$|AB|^2 = (Cos\beta - Cos\alpha)^2 + (Sin\beta)^2 + (Sin\alpha)^2 \quad (1)$$

Using the law of cosines to get the value of ∠AOC which is our first rotation.

$$Cos\angle AOC = \frac{OA^2+OB^2-|AB|^2}{2\cdot OA\cdot OB} = Cos\beta \cdot Cos\alpha \quad (2)$$

$$\angle AOC = \left(Cos(Cos\beta \cdot Cos\alpha)\right)^{-1} - \frac{\pi}{2} \quad (3)$$

Using Cos∠AOC we can generate point D and the length of AD.

$$AD = Sin\angle AOC = \left(1 - Cos^2\angle AOC\right)^{-\frac{1}{2}} \quad (4)$$

$$D = (Cos\beta \cdot Cos^2\alpha, 0, Sin\alpha \cdot Cos\alpha \cdot Cos\beta) \quad (5)$$

Then the third rotation angle γ can be computed by a dot product. Vector t is unit y-axis vector.

$$\overrightarrow{DA} \cdot \vec{t} = |AD| \cdot |t|$$
$$= (Cos\beta \cdot Cos^2\alpha - Cos\beta, -Sin\beta, Sin\alpha \cdot Cos\alpha \cdot Cos\beta) \cdot (0,1,0)$$
$$= Sin\angle AOC = \left(1 - Cos^2\angle AOC\right)^{-\frac{1}{2}} \cdot Cos\gamma \quad (7)$$

Thus, the angle γ is equal to:

$$\gamma = Sin^{-1}\frac{Si}{(1-(Cos\alpha\cdot Cos\beta)^2)^{-\frac{1}{2}}} \quad (8)$$

Finally, we can find the relationship between Euler angle and position is

$$\left(Cos^{-1}(Cos\beta \cdot Cos\alpha) - \frac{\pi}{2}, \alpha, Sin^{-1}\frac{Sin}{(1-(Co \cdot Cos\beta)^2)^{-\frac{1}{2}}}\right) \quad (9)$$

2.2. Unity3D Part

2.2.1. Scene

In this project, Unity3d was chosen as the development platform, which is a cross-platform game engine developed by Unity Technologies. The engine can be used to create three-dimensional, two-dimensional, virtual reality, and augmented reality games, as well as simulations and other experiences.[4] Our main scene consists of four essential parts, nearby detail terrain, overhead terrain, skybox (with simulated clouds), and balloon model. Detail terrain and overhead terrain has been introduced in *Calculations*, details about the skybox and balloon will be addressed in the following sections. The balloon model is the user's main point of view, execute the data transfer function and UI animation function.

Skybox is used to simulate the color of the sky changes as the altitude rises. As the altitude changes, the density of the atmosphere will also change, so the refractive index of the atmosphere to light will also change, thus changing the color of the sky.[5]

The cloud component is a package from Unity3d store called *"Massive Clouds - Screen Space Volumetric Clouds"* (Figure 2.c), which is produced by *mewlist*. With the help of clouds, we can realize the transition between nearby detail view and overlook view.

2.2.2. Data transfer

In Unity3d, the basic time unit is a frame, but from the *JSON* data, the basic unit is one second. Thus, the best way to transfer data is by combining speed and position information. There is a Unity 3d built-in function called *MoveTowards(),* which allow us to move an object from current position to target position at a specified speed.

The current position and target position can be used directly from *JSON* data, with the instantaneous speed approximately equal to the average speed. Because the entire DemoSat flight takes about 4 hours in total and the average time interval recorded by the sensors is 1s, slight speed changes in the process can be ignored on the premise of a fixed distance and time.

2.2.3. Import Local JSON data

As a simulation software, of course it needs to be able to import users' own data. After transferring the data to *JSON,* we can import them into simulation through "import" button (Figure 3.a) mentioned at last section.

If the user does not have data, but wants to experience the simulation, some default data was prepared inside, just press "normal" button or "VR" button. With the help of “VR” mode, future DemoSaT launch data will be able to be visualized. Future team can intuitively experience the path and operation of the high-altitude balloon, then performs a thorough analysis of the flight process.

2.2.4. VR Mode

Virtual reality (VR) is a simulated experience that can be similar to or completely different from the real world. A virtual reality headset typically includes two small high-resolution OLED or LCD monitors which provide separate images for each eye for stereoscopic graphics rendering a 3D virtual world.[6]

In order to make the user experience more immersive, VR function is added to this project. The VR system also is divided into two versions: PC and smartphone. As for the

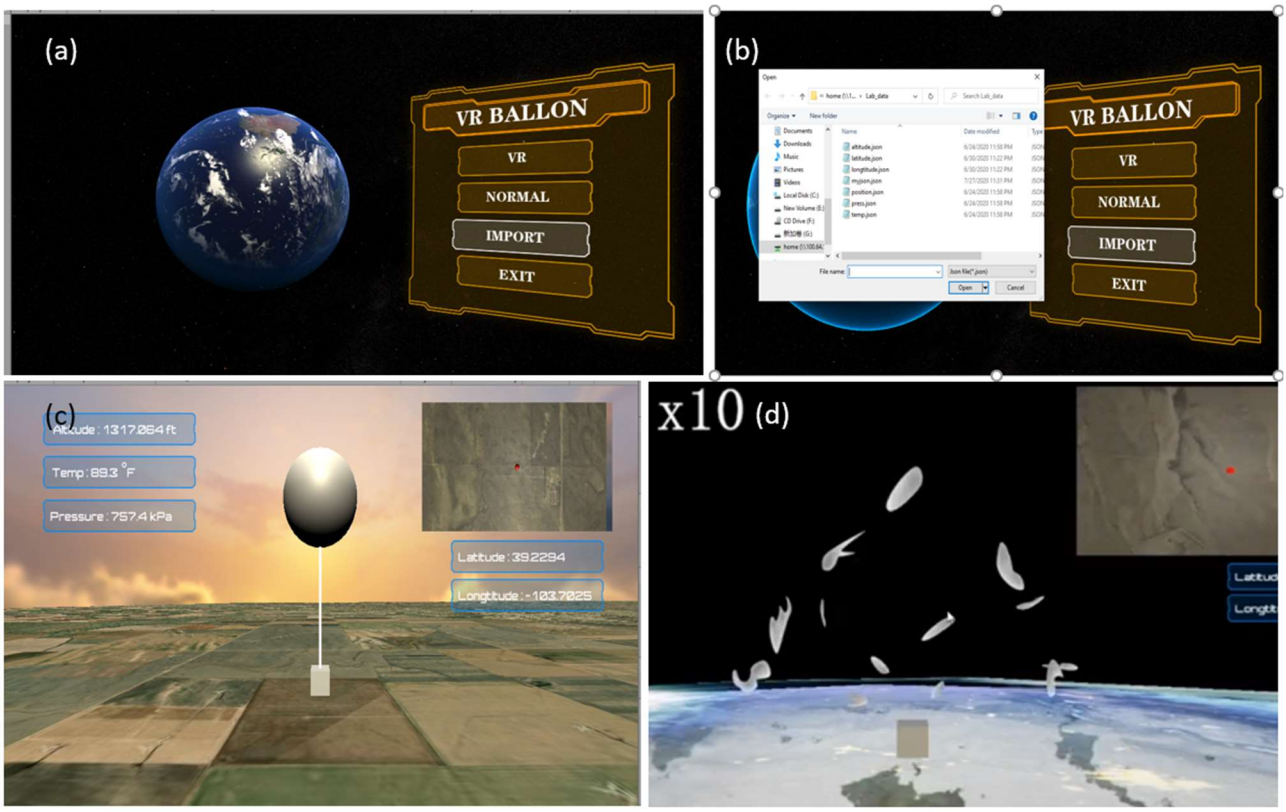

Figure 3: a) Operation Menu of this project simulation. b) UI view in normal mode. Importing local JSON data into simulation. c) UI view in normal mode. d) Balloon explosion view when reaching the highest point

PC part, the whole function is inherited by *Steam VR* (a Virtual Reality Platform developed by Valve as an extension of *Steam*) since their package is quite mature.

This version of VR requires special VR hardware to drive, such as *HTC VIVE*, *HTC COSMOS*. However, the price of these kinds of device is high, at the same time, it also needs a certain space to run. Thus, a more convenient and cheaper smartphone-based version was made.

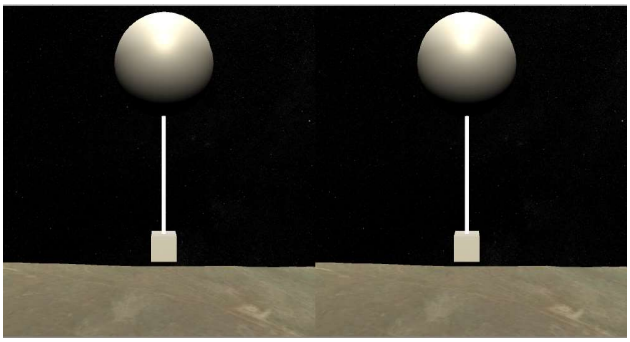

Figure 4: The view in VR mode, there is a slight difference in interpupillary distance between these two views.

In order to improve the fluency of the smartphone running this simulation, we eliminated most of the rendering and only provided images from two cameras. The interpupillary distance is maintained between the two cameras. Because of the pupillary distance of the human eye, there is a slight difference between the images viewed by the two eyes. Therefore, the camera view with such a parallax can simulate the effect of the human eye[7].

Smart phones have a higher resolution, so the screen can be clearly displayed when the phone is placed horizontally. Using with *Google Cardboard*, it can also help use enjoy the VR mode. *Google Cardboard* is a virtual reality (VR) platform developed by *Google*. Named for its fold-out cardboard viewer into which a smartphone is inserted, the platform is intended as a low-cost system to encourage interest and development in VR applications.[8]

2.2.5. UI and Animation

In this project, UI was used in two general areas: the operation menu and the information display during simulation.

The figure below (Figure 3.b) is our operation menu, including four function.

1) VR Button: A button loading user to VR mode.

2) Normal Button: A button loading user to normal simulation mode.

3) Import Button: A button helping user import their own *JSON* file.

4) Exit Button: A button closing the simulation.

Information display (Figure 3.c) consists of a mini map and numerical data. Numerical data present detail data which will update every second. Mini map gathers the position information and applies them into *Mapbox*'s satellite map API. During simulation, mini map will be consistent with detail view terrain.

Because of the difference in air pressure, the high altitude balloon will explode after reaching a certain altitude [9], An animation (Figure 3.d) was finally applied to the balloon explosion composed of particle special effects.

2.2.6. Speech Control

Since *Google Cardboard* is used and smartphone to build VR model, hands cannot directly touch the device buttons or screen for operation and speech recognition becomes the best choice. Here, Windows speech recognition API is used to achieve speech control. Below are the 6 keywords set to operate the balloon simulation in hands-free mode.

1) Play: Run or continue simulation.

2) Stop: Pause simulation.

3) Display text: Display altitude, temperature, pressure, latitude and longitude.

4) Display Mini: Display mini map.

5) Remove Text: Remove altitude, temperature, pressure, latitude and longitude.

6) Remove Mini: Remove mini map.

Users can perform specific operations by speaking these six keywords at runtime

3. Feature Development

In the future, this project will continue to be developed, adding support for various types of sensor data, as well as 360° video. Allowing users to easily access environmental data at various altitudes in different locations. A timeline system will also be considered, allowing users to drag the timeline to any point in time under the flight path. The development of this project will be useful for both research and education.

References

[1] "Research on Balloon to Float over 50km Altitude". Institute of Space and Astronautical Science, JAXA. Retrieved 2011-09-29.

[2] Burdea, Grigore C. “Teaching Virtual Reality: Why and How?” PRESENCE: Teleoperators & Virtual Environments, vol. 13, no. 4, Aug. 2004, pp. 463–483. EBSCOhost, doi:10.1162/1054746041944812.

[3] Kim, Phil, and Han Gil Chae. Rigid Body Dynamics for Beginners: Euler Angles & Quaternions. Createspace, 2013.

[4] Petty, Josh. “What Is Unity 3D & What Is It Used For?” Concept Art Empire, 14 Mar. 2019, conceptartempire.com/what-is-unity/.

[5] Lynch, David; William Charles Livingston (2001). “Color and light in nature”. Cambridge University Press. p. 31. ISBN 978-0-521-77504-5

[6] Schnipper, Matthew. "Seeing is Believing: The State of Virtual Reality". The Verge. Retrieved 7 March 2017.

[7] Naimark, Michael. “VR / AR Fundamentals - 2) Audiovisual Spatiality & Immersion.” Medium, Medium, 19 July 2019, michaelnaimark.medium.com/vr-ar-fundamentals-2-audiovisual-spatiality-immersion-298dc6bd6a0e.

[8] Pierce, David. “Google Cardboard Is VR's Gateway Drug.” Wired, Conde Nast, 3 June 2017, www.wired.com/2015/05/try-google-cardboard/.

[9] McNamara, Marilyn C., "An Analysis of Burst Altitude for Weather Balloons" (2016). Antonian Scholars Honors Program. 43. https://sophia.stkate.edu/shas_honors/43