Authoring Tool Detailed plan

Aimable Tuyishime
Aime Cesaire Musangamfura
Felix Uwizeye
Nebiyou Yismaw

October 10, 2018

Table of Content

1. Introduction

In this lab, we are required to write an authoring tool that allows us to instantiate different 2D and 3D objects, situate these objects at different positions and orientations, map images to the objects and generate an equirectangular stereo pair image.

Three different groups will be working on different pieces of the authoring tool. Our team will be working on the main program of the tool. The tasks of the main program are to:

- Create a black image of size 4096 x 4096. Objects and images will be written to this image.
- Define parameters for the desired objects. These parameters include the size and location of objects.
- Instantiate the desired objects using the specified parameters.
- Compute a transformation matrix that transforms world coordinates to camera coordinates.
- Loop through each object, perform an equirectangular projection and write pixel values to the output image.
- Handle object occlusion by giving priority to pixel values from objects that are closer to the camera.
- Optionally fill the gaps that might appear in the objects, which arises due to the technique used to compute pixel values.

2. Architectural overview

The high-level architectural overview is shown in figure 1.

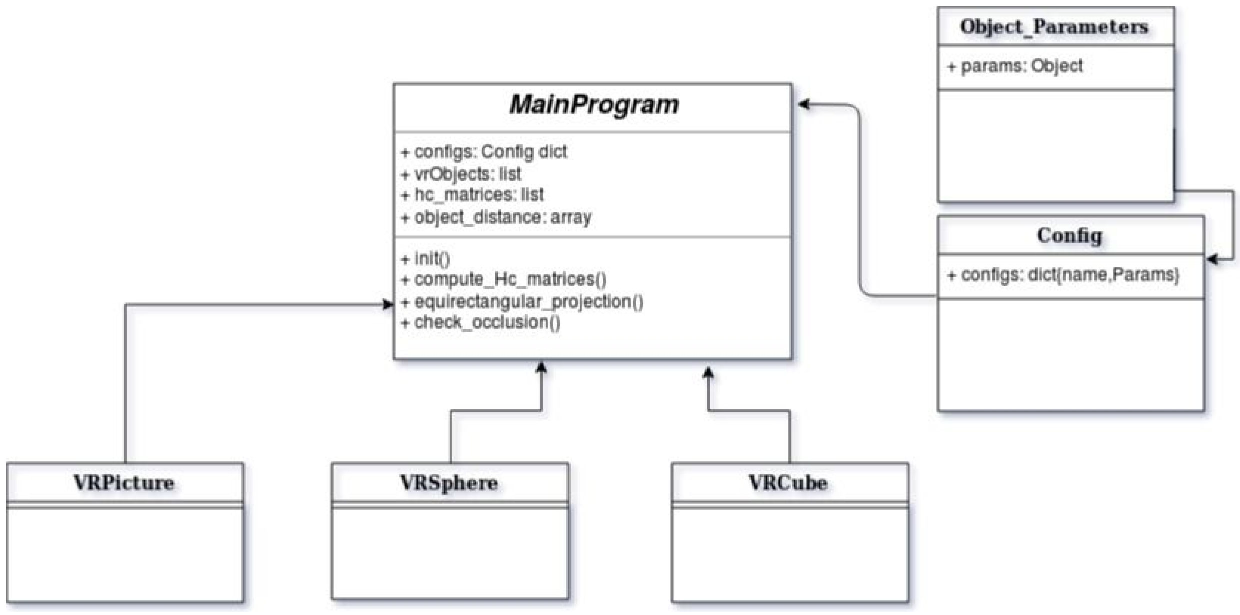

Figure 1. Architectural overview diagram.

Each object will have its own class implementation and will be implemented by different teams. These classes are VRPicture, VRSphere, and VRcube for the 2-D picture, the sphere, and the cube objects respectively. Our main program will import and use these classes. The main program also loads object configurations from a python dictionary and use it to instantiate the objects. The details of the methods in the main program are discussed below.

Objects Configuration and Instantiation

To configure objects that are going to be used in the authoring tool, a python dictionary object will be used as it is easy to export into the main program and easier to implement. Using the command line to pass object configurations would be tedious for the user. Using a text file to hold configurations would require a separate parser to extract the information. Hence, using a python configuration object would be optimal.

A python configuration object will allow a user to define the initial state of the main program. A configuration object will allow users to change both the characteristics and behaviors of virtual objects. The configuration object will have a definition as the following one:

```
authoring_tool = {
        “out_image_size” : (4096, 4096),             # Size of the output of an image in pixels
        “objects”: {
                “sphere”:  {
                        “radius”: 10,                       # in units of focal length
                        “size”: (128, 128),                 # in pixels
                        “location”: (23, 65),               # in pixels
                        “tait-bryan-angles”: [0, 30, 69]    # in degrees

                },
        }
}
```

Computing transform matrices for each object

In order to project objects to the left and right camera coordinates, transform matrices will be used. Each object will have its own transformation matrix, which depends on its parameters.
The object parameters and the corresponding transform matrices will be passed to a function that will perform transformation of the object’s parameters for each camera view and return corresponding H matrix.
The parameters to be passed in are the object’s tait Bryan angles and their center of location space location.
The object's coordinates will be converted to the World coordinate systems in order to track their position which will be aligned with the intended camera's coordinates, whether left or right.
Given the coordinates of point p on a given object, as the object rotates the point p will rotate around the stationary elemental axes.

The rotation matrix R will be given as $\mathbf{R_{comp} = R_y(\phi)R_x(\theta)R_z(\Psi)}$

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix} \quad R_y(\phi) = \begin{bmatrix} \cos(\phi) & 0 & \sin(\phi) \\ 0 & 1 & 0 \\ -\sin(\phi) & 0 & \cos(\phi) \end{bmatrix} \quad R_z(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

This matrix will give the coordinates of $\underline{p'}$ in the world coordinate system
$\underline{P'} = \mathbf{R}_{\mathbf{comp}}\underline{P} + \underline{t}$, where $\underline{t}$ is the origin of the body coordinate system in the world’s coordinate system.

The world to body conversion will be performed using the transpose of the $\mathbf{R}_{\mathbf{comp}}$
$\underline{P} = \mathbf{R}^{\mathbf{T}}_{\mathbf{comp}}\underline{P'} - \mathbf{R}^{\mathbf{T}}_{\mathbf{comp}}t$

Equilateral projection and Occlusion handling

After getting the transform matrices for each camera, we transform the world coordinates to left-right camera coordinates, perform equilateral projection and write the pixel values. The world coordinates will be converted to polar coordinates, which will be used to fill pixel values.

In order to handle object occlusion, we will track the viewing distance of the object whose pixel value is written in every image coordinates. Initially, the Occlusion distance, d, is at -1. We will create a lookup table that will map the output image coordinates with the corresponding viewing distance. The lookup table will have the same dimension as the image.

Eg: consider a 5x5 image, we will initialize the lookup table for that image to be as follows

-1	-1	-1	-1	-1
-1	-1	-1	-1	-1
-1	-1	-1	-1	-1
-1	-1	-1	-1	-1
-1	-1	-1	-1	-1

The lookup table will help us to compare the current values of the object distance in the output image coordinates with the viewing distance of the object whose pixels we want to write at these coordinates.

While we are looping through the image coordinates and writing our objects to the output image, we are going through the following steps:

- Check if there is an object at coordinate (x,y) of the output image by checking the value of the occlusion distance in the lookup table at that specific coordinate.
- If there is no object, meaning occlusion distance is -1, we will write the object and update the value of the occlusion distance with the objects viewing distance at that coordinate.
- If there is an object, meaning occlusion distance is different from -1, we will compare the viewing distance of the object with the current occlusion distance we have at that coordinate in the lookup table,
- If this object's distance is less than the value in the lookup table, we will overwrite the pixel value with this object's pixel value and update the lookup table.
    - Eg: For the 5x5 image we created, if we write an object at coordinates (0,0), (1,1), (2,2), (3,3) and (4,4) with a viewing distance of 0.5, our lookup table becomes

0.5	-1	-1	-1	-1
-1	0.5	-1	-1	-1
-1	-1	0.5	-1	-1
-1	-1	-1	0.5	-1
-1	-1	-1	-1	0.5

- If the object's distance is greater than or equal the value of the lookup table, we will not overwrite.

3. Schedule

Timeline		Activities
Start date	End date	
October 3, 2018	October 10, 2018	Detailed implementation plan of the lab, division of tasks and other related discussions.
October 11, 2018	October 12, 2018	Team members will works on and implement their part.
October 12, 2018	October 13, 2018	The initial merging of individual features and unit testing.
October 13, 2018	October 15, 2018	Integrated testing and finalizing the work.
October 15, 2018	October 16, 2018	Exchange our tested program with other groups.
October 16, 2018	October 19, 2018	Writing the final report and finalizing the program documentation.

4. Individual responsibilities

Team member	Responsibilities
Aimable Tuyishime	Implement parameter definition and instantiation of different objects.
Aime Cesaire MUSANGAMFURA	Handle object occlusion by tracking the distance of the object that is written in every image coordinate.
Felix Uwizeye	Implement transform matrix computation for each object.
Nebiyou Yismaw	Implement equirectangular projection and write pixel values to the output image.

5. Testing mechanism

After implementing all the required functionalities, we will provide a testing mechanism. These tests will assert the correctness of our implementation. Individual tests for each functionality and an integrated test for the main class will be provided. These tests are:

- Test instantiation and configuration
- Test transform matrices
- Test equilateral projection
- Test occlusion
- Test main

6. Conclusion

Our goal is to design the authoring tool that will be used to manipulate objects in a VR environment. It will provide a way to position and rotate objects, hence author our 3D space. We will provide an output image of size 4096x4096 containing 3-D objects, which can be viewed in 3D-360 mode in the Oculus Go.