Report: A3 Moving Planets

LEE SUNG JAE

2016314718

Mathematics Department

1. Planets Movement

There are two big parts to implement in this assignment, namely, planets movement and camera movement. And this paragraph is for planets movement. First, I used the unit sphere vertices and indices that I used in A2 assignment. Also, I did not make 8 sets of vertices and indices for each planet but only made only 1 set of vertices and indices. Then, for each planet, I set a translation matrix for its location, a scale matrix for its appearance size, a rotation matrix for its self-rotation, and a revolution matrix for its revolution around Sun. A revolution matrix is a new matrix that I made for adapting revolution. 7 planets revolve around z-axis on xy-plane and Sun is at the center of the coordinates. Hence, after adapting scale matrix, rotation matrix, and translate matrix like usual to make a model matrix, we can multiply by a revolution matrix to express the revolution around z-axis.

Model Matrix = Revolution Matrix * Translate Matrix * Rotation Matrix * Scale Matrix

For each planet as a structure, I let each have 7 attributes such that center(location), radius, rotation_theta(that saves how much angle it rotated from 0), revolution_theta(that saves how much angle it revolved from its original location on its orbit), rotation_speed, revolution_speed, and model_matrix(that converts the object to world space). I initialized the first 6 attributes when I created a planets vector(STL). Then, I updated the model_matrix attribute at each frame based on the machine time to avoid machine-dependent speed.

2. Camera Movement

Camera movement includes trackball movement, zooming, and panning. Trackball movement source code has been copied from Trackball lecture source code. Zooming and panning has been implemented by changing the main camera's view matrix. Firstly, zooming requires the camera to move along n-axis. Note that the view matrix looks like the following.

$$View\ \ Matrix = \begin{pmatrix} u_x & u_y & u_z & -u \cdot EYE \\ v_x & v_y & v_z & -v \cdot EYE \\ n_x & n_y & n_z & -n \cdot EYE \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Looking through the matrix above, we observe that moving along n-axis requires the (3,4) element of view matrix to change. Hence, in motion() callback function, when right mouse is clicked or when shift button and left mouse are clicked together, the (3,4) element of the view matrix is added by the vertical displacement of a mouse.

To implement panning, we know that we have to move the camera on uv-plane. By observing the view matrix above, we can say that the (1,4) and (2,4) elements are needed to change. Similar to the method of zooming, when middle mouse is clicked or when control button and left mouse are clicked together, the (2,4) element of view matrix is added by the horizontal displacement of a mouse and the (3,4) element is added by the vertical displacement of a mouse.

To sensor what keys and buttons are clicked by the user, in mouse() callback function, it checks what the user input and changes the boolean values such that right_button_clicked, shift_button_clicked, control_button_clicked, and middle_button_clicked.