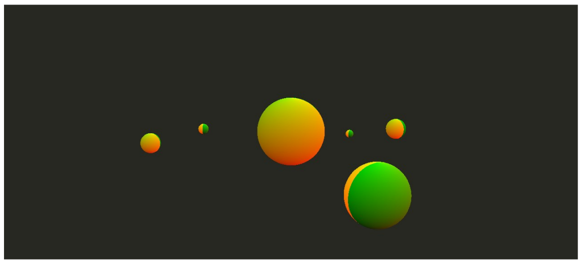
Computer Graphics

Assignment 3: Moving Planets (Solar System I)


(a) front view of the moving planets

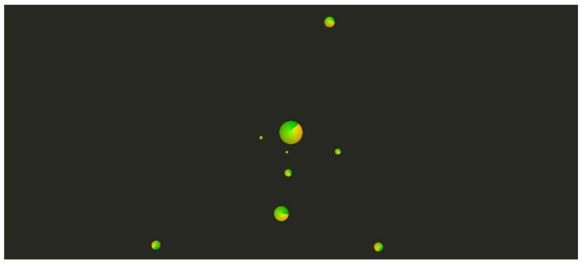

(b) top view of the moving planets

Figure 1: Example renderings of nine planets.

First, refer to the general instruction for assignment submission, provided separately on the course web. If somethings do not meet the general requirements, you will lose corresponding points or the entire credits for this assignment.

1. Objective

In the last assignment, you learned how to make a sphere using triangular approximation. In this assignment, you are expected to locate many planets in space and move them. To be more specific, you will learn how to apply transformations for 3D animation. The rendering result will be similar as shown in Figure 1.

2. Mandatory Requirements

If any of what is listed below is missing, you will lose 50 pt for each (up to 100 pt; equivalent to no submission).

- Use your own sphere. You may reuse the result of the last assignment.
- Instancing should be applied for a static vertex array object. In other words, you need to use only a single vertex buffer (or vertex array object) and it should stay constant.
- The animation should be implemented using uniform variables; do not change the content of your vertex buffers.
- The animation should use time stamps/ticks instead of a frame counter; this makes the speed of your animation consistent across different machines.
- Perspective view should be employed using virtual trackball.
- Visualize the planets as done in the last assignment.

3. Requirements

You may start from the last assignment or the "trackball" example for this assignment.

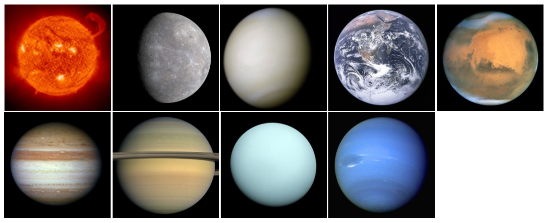


Figure 2: Nine planets in the solar system.

- (20 pt) There should be nine planets (9 instances of spheres): Sun, Mercury, Venus, Earth, Mars, Jupiter, Saturn, Uranus, and Neptune; Pluto is excluded (see Figure 2 for example).
- (40 pt) Eight planets should revolve around the Sun according to their revolution periods, and nine planets should be self-rotating according to their rotation periods.

  Note that you do not have to apply exact scales in size, revolution radii, and rotation timings; just scale them appropriately so that they are visualized well.
- (30 pt) Implement zooming and panning operations of the camera. Zooming uses both the right button and shift+left button of a mouse along the Y-axis in the screen. Likewise, panning uses both the middle button and ctrl+left button. Zooming and panning require being implemented in the host CPU application by extending trackball class given in the sample; do not use an additional matrix uniform (in shader) for zooming/panning.

  More specifically, the zooming moves the eye (or/and at) position along n axis, found for your view matrix. Similarly, the panning moves the eye and at positions along uv plane.

  Make sure not to scale fovy for zooming; in other words, do not change the projection parameters. Instead, move the eye position, while fixing the field of view. In addition, you may or may not move the at position; observe its effect for comparison.

4. What to Submit

- Source, project (or makefile), and executable files (90 pt)
- A PDF report file: YOURID-YOURNAME-A3.pdf (10 pt)
- Compress all the files into a single archive and rename it as YOURID-YOURNAME-A3.7z.
- Use i-campus to upload the file. You need to submit the file at the latest 23:59, the due date (see the course web page).