Monash University
CSE3313 Computer Graphics

Tutorial 4 2007

11 September 2007

Objectives

- Understand the concept of backface removal;
- Understand how OpenGL uses backface removal to speed rendering;
- Hidden Surface Removal (HSR) using the depth buffer algorithm;
- Interactive Navigation of 3D space using Euler angles.

Solar System Program

On the CSE3313 web site, locate the *Sample Code* page http://www.csse.monash.edu.au/~jonmc/CSE3313/Resources/SampleCode/index.html. Download the program `solar.tar.gz`.

Untar the code and build it for your particular platform. The program is a simple model of a planet with moon, orbiting a sun. It uses hidden surface removal and lighting. Run the program and use the mouse to change the current viewpoint. Right-clicking brings up a menu which allows you to toggle various features.

Look over the code and get an understanding of how the program works. In particular make sure you understand how the hierarchical transformations are used to animate the orbit of the planet and moon.

Note also how depth buffering is enabled to remove hidden surfaces.

Exercise 1.1 (Backface Removal) *Add backface removal to the program. Allow the user to switch backface removal on and off using a right-click menu.*

See the lecture notes (Lecture 20) for details on Backface removal.

Mouse Navigation

Modify the program so that the solar system scene is displayed in two viewports:

- The first viewport should show a space ship view that the user pilots around using the mouse;
- The second viewport shows an orthographic view of the solar system with the sun, planet, moon and spaceship visible.

For the spaceship just use a cone pointing in the current direction of motion. Use the a and s keys to increase and decrease the speed of the ship. The ship should start with a speed of 0.

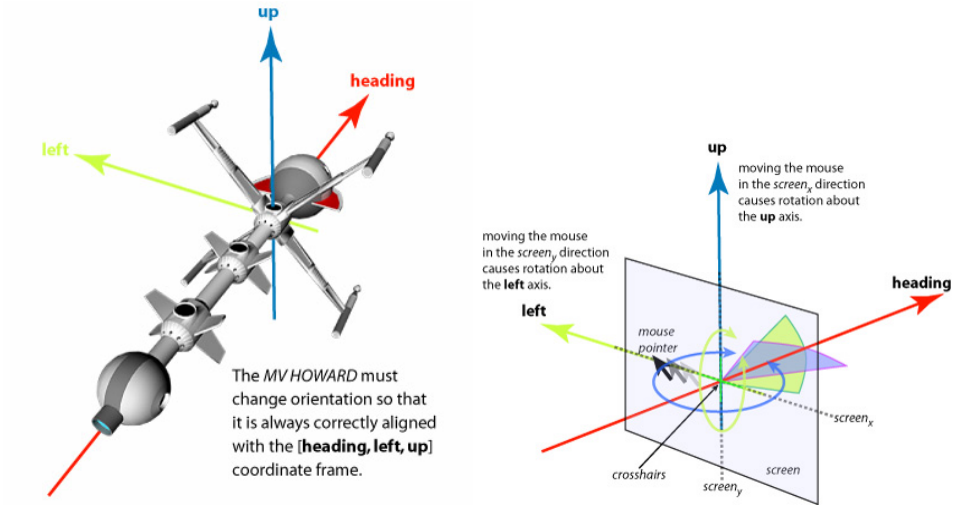

Figure 1: Mouse movement mapped to Euler rotations

The space ship needs to maintain its own co-ordinate system with **heading**, **left** and **up** unit vectors (see Fig. 1). At each time step the ship moves in the heading direction at the current speed.

Moving the mouse horizontally (in the x direction) rotates the **heading** vector about the **up** vector (yaw). The speed of rotation is proportional the the distance of the mouse from the centre of the viewport.

Moving the mouse vertically (in the y direction) rotates the **heading** vector about the **left** vector (pitch). The speed of rotation is proportional to the distance of the mouse from the centre of the viewport.

In both cases of mouse movement, the ship only stops turning when the mouse is at the centre of the viewport. You may find it helpful to mark the centre of the viewport with a small cross to aid navigation.

There is no need for the ship to implement a roll rotation.

You might wish to add a keyboard command to reset the ship position and orientation in case you get lost while flying around.

Hints

- The ship needs to maintain a current position and orientation.
- Orientation can be specified using an HLU orthogonal co-ordinate system.
- You will need to write functions that convert between HLU orientations and Euler angles
- You will also need a function that takes the ships position and orientation and converts it into an OpenGL transformation.
- At each time step, the new ship position equals the old position + the velocity $*dt$ (where dt is the time step).
- To draw to each viewport call a common function `drawScene` that draws all the objects in their correct world co-ordinates.