CS488 – Assignment 2

Alexander Klen

ayklen 20372654

June 6, 2014

Manual

Program Description

This program displays a wireframe cube and two gnomons using only 2d drawing operations. It is intended to show how matrix stacks for mapping between different coordinate spaces work, and how to implement a perspective view.

Controls

Click and drag sideways using the left, middle, and right mouse buttons to change view parameters. Their effects depend on the view mode:

- View rotation mode: rotate sight vector about x, y, and z axis
- View translate mode: translate eyepoint along x, y, and z axis
- Perspective mode: Change FOV, translate near plane along z axis, and translate far plane along z axis
- Model rotation mode: Rotate box about its x, y, and z axis
- Model translate mode: Translate box along its x, y, and z axis
- Model scale mode: Scale box along its x, y, and z axis
- Viewport mode: Click and drag only the left mouse button to set the viewport rectangle. Note that the aspect ratio of the viewport is that of the window, not the viewport

Menu Items

Under the Application menu you'll find the menu items “Reset (R)”, and “Quit (Q)”. The letters in brackets are the keyboard shortcuts for the menu item. “Reset” will reset the mode, viewport, and all translations, rotations, and scales.

Under the Mode menu, you'll find all of the modes mentioned in the Controls section with their shortcut keys listed.

Interface

The current interaction mode is displayed in a textual label at the bottom of the screen. Next to it the field-of-view (fov) is displayed in degrees. Also displayed are the near and far plane values. Note that you cannot make the near plane farther than the far plane, and you can't make either plane negative. The field-of-view is bound to the range [5, 160] degrees.

Additional Documentation

For clipping, see the function “homogeneousClip”, viewer.cpp:212. I implemented clipping in non-normalized homogeneous coordinates against the viewing volume planes $w = n$, $w = f$, $w + x = 0$, $w - x = 0$, $w + y = 0$, $w - y = 0$. This function is called on each line segment (pair of 4D points), and it either reports that the line is entirely invisible, or it shortens the line by setting its endpoints to the proper intersection points with the 6 viewing volume planes.

I implemented a basic scene graph in scene.hpp and scene.cpp. An instance is built at viewer.cpp:60, it is described there at what node specific matrices are stored. The matrices used in my pipeline are:

- a model scaling matrix that is applied to the cube's points
- a model translation and rotation matrix that acts on the cube and the modeling gnomon (transforming them from model coordinates to world coordinates)
- a viewing matrix which transforms the cube and both gnomons from world coordinates to viewing coordinates
- a perspective matrix which transforms everything from viewing coordinates to NDC
- A screen matrix which is applied after clipping and transforms everything from NDC to screen coordinates (this is created based off of the current viewport and window aspect ratio)

All of these except for the screen matrix are stored at seperate nodes in the scene graph (held in Viewer) and are concatenated using node's recursive method. Modifications are made to each by pre and post multiplying matrices onto the existing ones. The screen matrix is created each frame based off of the current viewport and window sizes (even while the viewport or window are being dynamically resized).

Assumptions

- I can use a projection transform before clipping to all 6 planes in non-normalized homogeneous coordinates, avoiding any dividing by zero.
- View translate mode should move the eyepoint in the mouse direction for x, therefore making the cube move in the opposite direction.
- Dragging/resizing the viewport should dynamically draw the box before you let go of the mouse button.
- When you decrease the size of the window, the viewport doesn't have to be shrunken down to fit inside.
- Change the size of the window changes the aspect ratio of the viewport but doesn't change the viewport's size (you can change its size manually).
- Model translations are done according to the model coordinates (applied after model rotations). This means that translating the model and then rotating the model coordinates will result in the cube swinging around the world coordinate axis as its coordinate system is rotated.