NRTSViewer.cpp

Creates a Viewer object g_viewer, onto which a scene graph will be attached and then rendered. It calls an event traversal method of the Viewer class

In order to render the view, we have to point a camera towards the scene. By default, Viewer:run() will create osgGA::TrackballManipulator in order to control the Camera

To set own manipulator, call Viewer::setCameraManipulator(). In this case, there is no need to write Viewer::run(); instead you have to iteratively update the view changes per frame

NRTSKeySwitchMatrixManipulator.h -> m_viewer <- VSimApp.cpp

VSimApp.cpp -> main crux of the code

- ➔ Creates objects of each of the menu classes in the default constructor
- ➔ InitViewer
  - sets up several arguments for the viewer and then sets the camera manipulators.
  - A call is made to NRTSkeySwitchMatrixManipulator using the viewer object m_viewer in order to set the manipulators.
  - Using 3 manipulators here – NRTS (NRTSManipulator()) , USIM (USIMManipulator()) , Terrain (ObjectManipulator()) *All these are custom-made*.
  - Finally, set the CameraManipulator to be from one selected by the KeySwitchManipulator.
  - Set up something called as stste manipulators (I believe this is not being used at the moment)
  - A few handlers are initialized and set to their appropriate values. These include handlers that influence how the window size changes, printing statistics on the screen, using the LOD scale to increase and decrease something, etc.
  - In the end, the camera matrices are set by using the setProjectionMatrixAsPerspective with parameters such as fovy?, zNear?, zFar? And aspectRatio
- ➔ There are functions to either retrieve or set the values of fovy, zNear, zFar and aspectRatio

Notes about Camera and viewing:

Initial camera frame is centred at the origin with the view direction aligned with the negative Z axis of the world frame. Theefore, we apply transformations such as translation or rotation to it in order to move the camera frame relative to the world frame i.e. in order to get it to “classical viewing”. From thereon, each distance is computed relative from the viewer to the object

Refer: http://info.ee.surrey.ac.uk/Teaching/Courses/eem.cgi/lectures_pdf/opengl3.pdf

Zfar, znear are distances to plane from the centre of projection; projection plane is orthogonal to the z-axis. Fovy is the angle between the top and bottom planes and is computed as

$$fovy = 2\tan^{-1}\left(\frac{h}{2d}\right)$$

$$aspect = \frac{w}{h}$$

In VSim, the default setup is a perspective projection to view objects within a 55 degrees field of view at a distance of 1.0 (ZNear) to 5000.0 (ZFar) units with an aspect ratio of 16.0/10.0