a2view-170050077_170050081

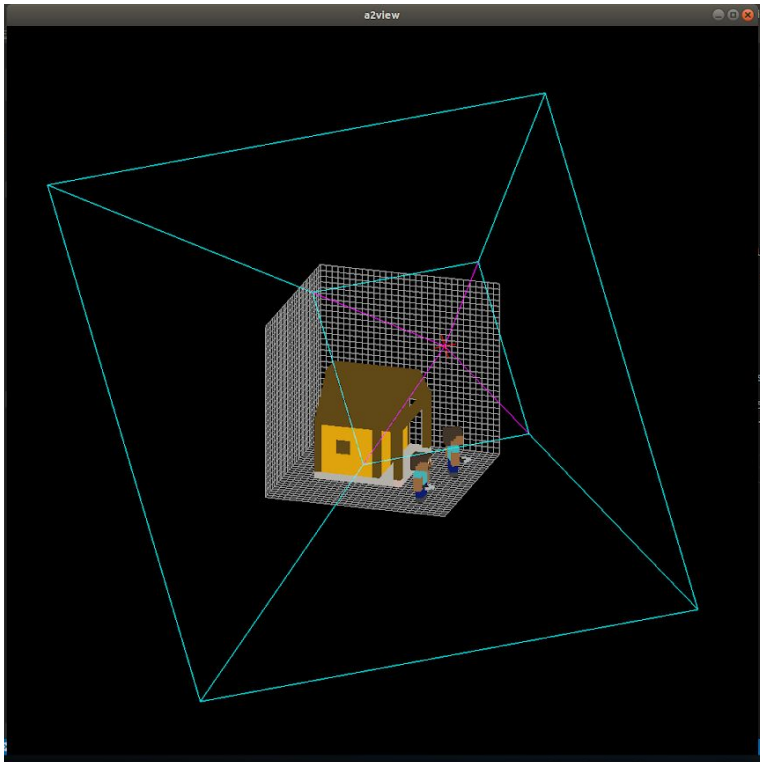
170050077 -- Nama N V S S Hari Krishna
170050081 -- Sailendra Bathi Babu

Note: Most of the boilerplate code like .glsl files and most of the .cpp and .hpp files are similar to that pf Assignment 1.

Code Walkthrough:
The main function sets up the OpenGL context, loads the shaders then calls the **initBuffersGL** function and renders using the **renderGL** function until the user terminates the window.

The **initBufferGL** function calls the functions corresponding to creating the points for grid, frustum and another function for parsing the **myscene.scn** file and initializing the necessary variables. Next, it sets up **vaos** and **vbos**. Separate **vaos** and **vbos** are used for each model loaded from the scene, the grid and the frustum.
The **renderGL** function applies the necessary transformations according to the current mode and then draws them.

WCS:
This is the default mode of the program. Here the grid, models loaded from the scene and the frustum are shown according to the world coordinate system.

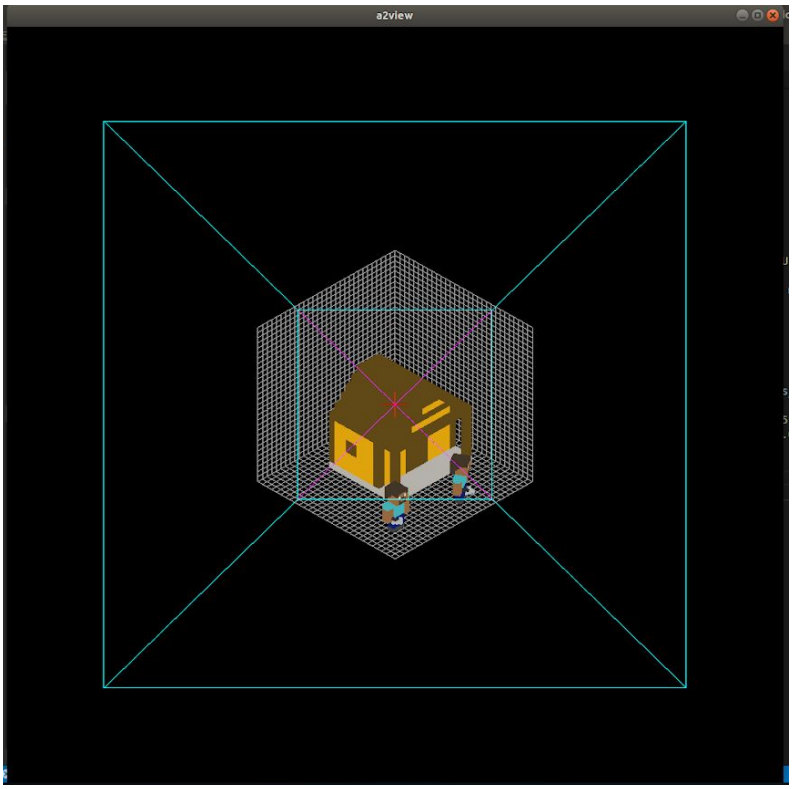


WCS

In the above picture, the red cross denotes the eye, the region between near and far planes are drawn in cyan and the projectors between the eye and near plane are drawn in magenta. This part uses the same transformations used by Assignment 1 to display the models, grid and frustum.

VCS:

Here a WCS to VCS transformation is applied on top of what is done in **WCS** which is essentially a transformation that maps **eye** to the origin and uses **u**,**v**,**n** unit vectors as the coordinate basis. The transformation matrix is named as WCStoVCS and initialized in the **initBuffersGL** (Same as the one in lectures). We later multiply with this transformation matrix in **renderGL** according to the current mode.

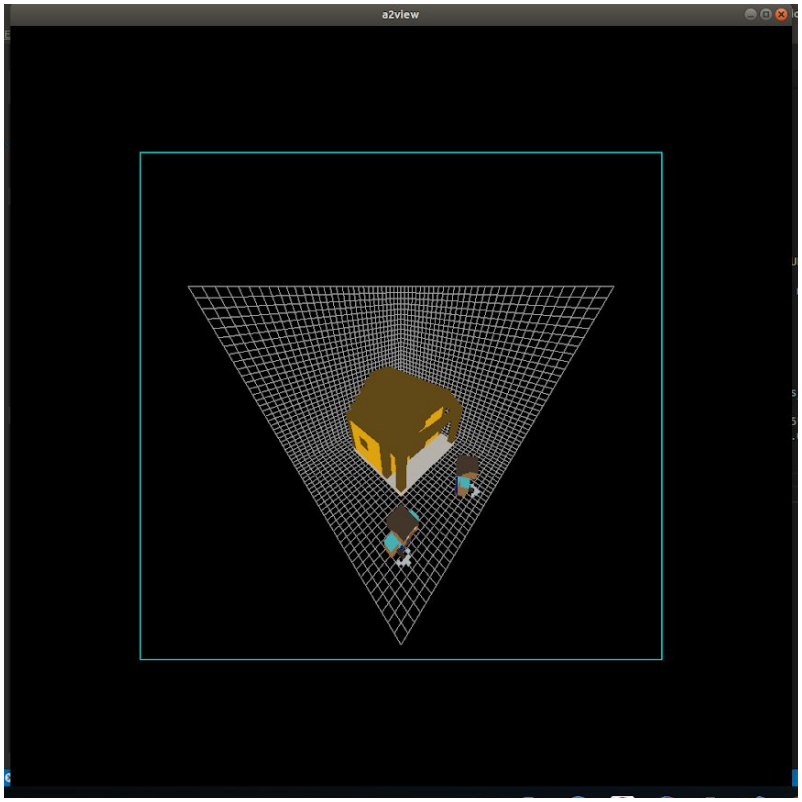


VCS

CCS:

Here a VCS to CCS transformation is applied on top of what is done in **VCS** which is basically a transformation that converts the view frustum to a unit cube via Shearing, Scaling, and Normalization. The transformation matrix is named VCStoWCS and initialized in the

initBuffersGL (Same as the one in lectures**)**. We later multiply with this transformation matrix in **renderGL** according to the current mode. Another subtle change was that the ortho matrix used for the visualization is also changed to accommodate the fact that all the points are now in the unit cube. (Initially, the grid points span from 0 to GRID_SIZE)



CCS/NDCS

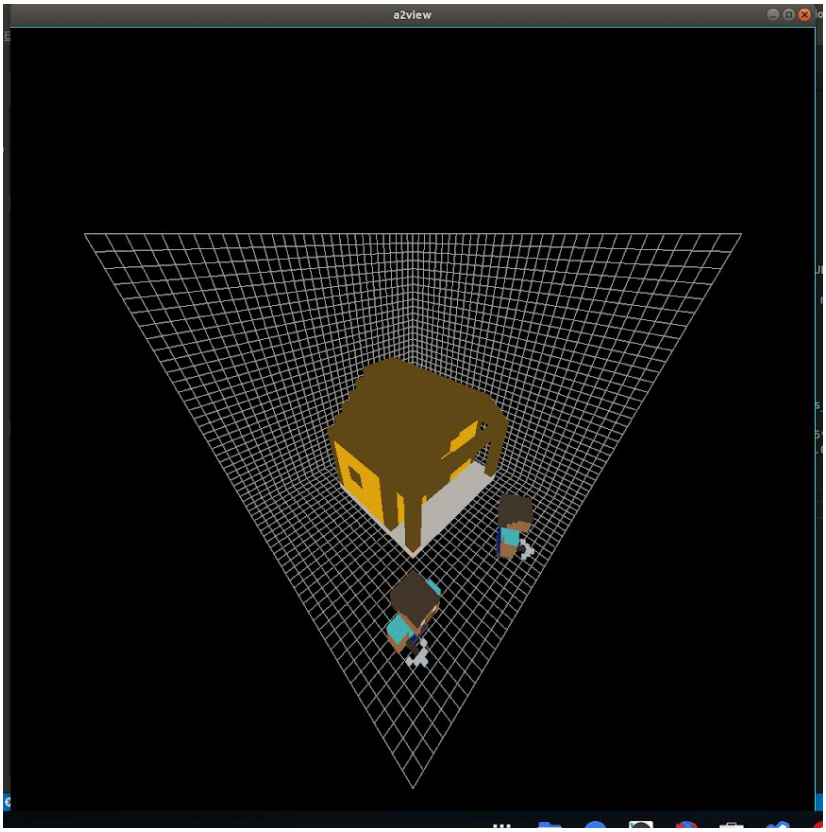
NDCS:

Here the CCS is followed by a perspective divide with the **w** coordinate. There is no change in the view of the model. It is the same as that of CCS.

DCS:

Here we need to map the points in NDCS to Device or Window Coordinates. To simulate this the ortho matrix which is used for the simulation is changed so that it displays the region inside the transformed frustum (i.e unit cube).

The corners of the unit square region are mapped to the corners of the window.



DCS