CG Assignment 06 21118

Problem definition: Write C++ program to draw 3D cube & perform following transformations on it using openGL
i) Scaling ii) Translation iii) Rotation about one axis

Objectives To learn to use functions & draw graphics & their transformations using openGL

Outcomes. To implement cube transformation in openGL using C++.

Hardware Requirement:
Processor: Intel core i5th 8th gen.
Memory: 8GB, 512GB SSD
Manufacturer: Acer Inc.

Software Requirements.
OS: Ubuntu 20.04 LTS on oracle virtual machine.
gcc version: 10.1.0.

Theory:
openGL is a cross-platform, hardware acceleration, language independence, standard industrial API for producing 2D & 3D graphical objects.

3 sets of libraries are used in openGL program:
① Core openGL (GL) eg. glColor, glVertex, glTranslate, glRotate
② OpenGL Utility library (GLU): eg glulookAt, gluPerspective.
③ OpenGL Utility Toolkit (GLUT): eg. glutCreateWindow, glutMouseFunc

The frame buffer:
The fragments that are produced by rasterization are given to the frame buffer where they are displayed. The frame buffer is a rectangular array of n bitplanes.

The bitplanes are arranged into several logic buffers - color buffer, depth buffer, stencil buffer & accumulation buffer

GL_QUADS:

This type of primitive is used to draw individual convex quadrilaterals. This has been used to draw the cube. we need to specify the co-ordinates for all the 6 faces of the cube to draw it.

GLUT_DOUBLE:

It refers to the double buffer mode. Here we have a front buffer & a back buffer. The front buffer is the one that is displayed while the drawing is done on the back buffer. Once the drawing on the back buffer is completed the buffers are swapped & in this way incomplete drawings on the screen are avoided.

- Algorithm

1) Start
2) Initialize glut in the main method
3) Specify display mode as GLUT_RGB & GLUT_DOUBLE
4) Create the window
5) Set the display function as display
6) Set the reshape function as reshape.
7) Clear the color to black.
8) Read the choice of the user to translate, scale & rotate
9) Call the functions on the display function
10) Display the transformation
11) Stop.

display()

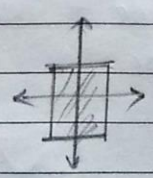
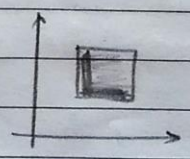
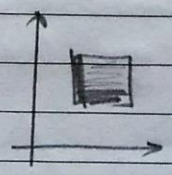
1) Start
2) Clear the color buffer & depth buffer

3) load identity to reset all transformations
4) Use inbuilt translate, rotate & scale function & pass the amount of it read from the user.
5) start drawing the cube
6) Specify the 4 vertices for each face of the cube with colors
7) Flush the buffer @ swap the buffer.
8) Stop

reshape()
1) Start
2) Invoke the viewport with depth & height of the window
3) change the matrix mode to PROJECTION_MATRIX
4) Set the perspective to view ∠ of 60°, aspect ratio of 1 the near frustum at 2 & the far frustum at 10.
5) Change the matrix mode back to the MODELVIEW MATRIX
6) Stop

init()
1) Start
2) Clear the color to black
3) Enable depth testing
4) Stop

4 <u>Test cases</u>

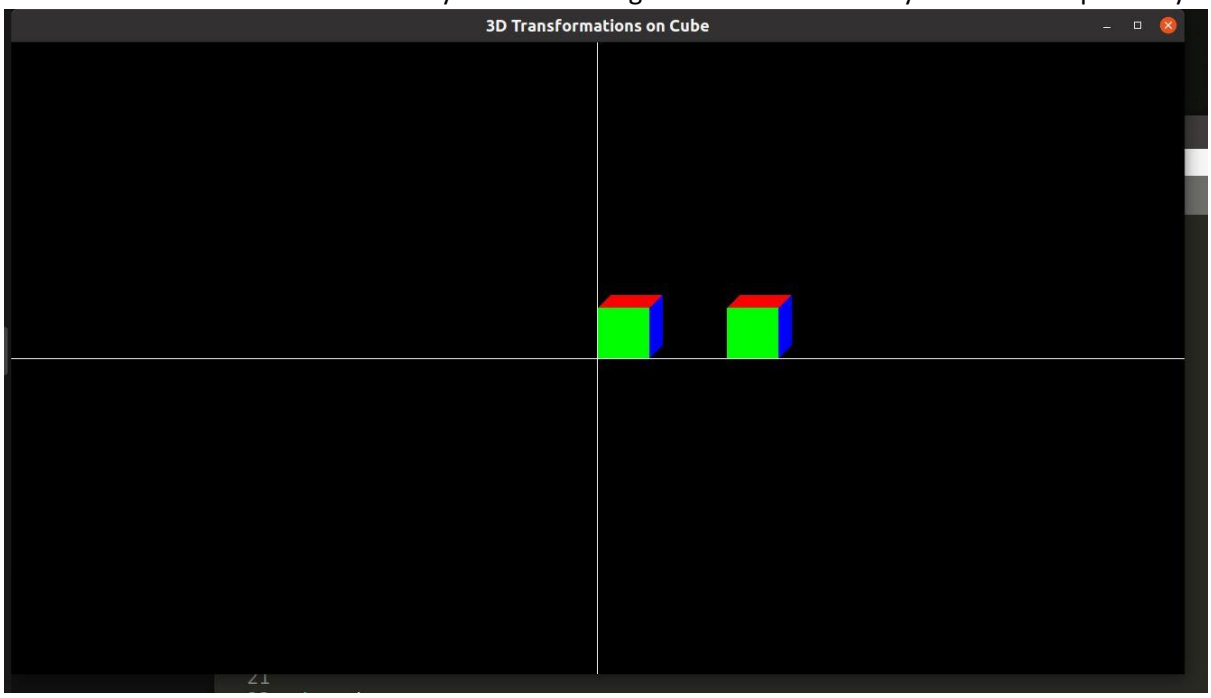
Test case description	Expected output	Actual output	Result
$t_x = 100$ $t_y = 100$ $t_z = 0$ $s_x = 1.0$ $s_y = 1.0$ $rotat = 0$ 			pass

Conclusion

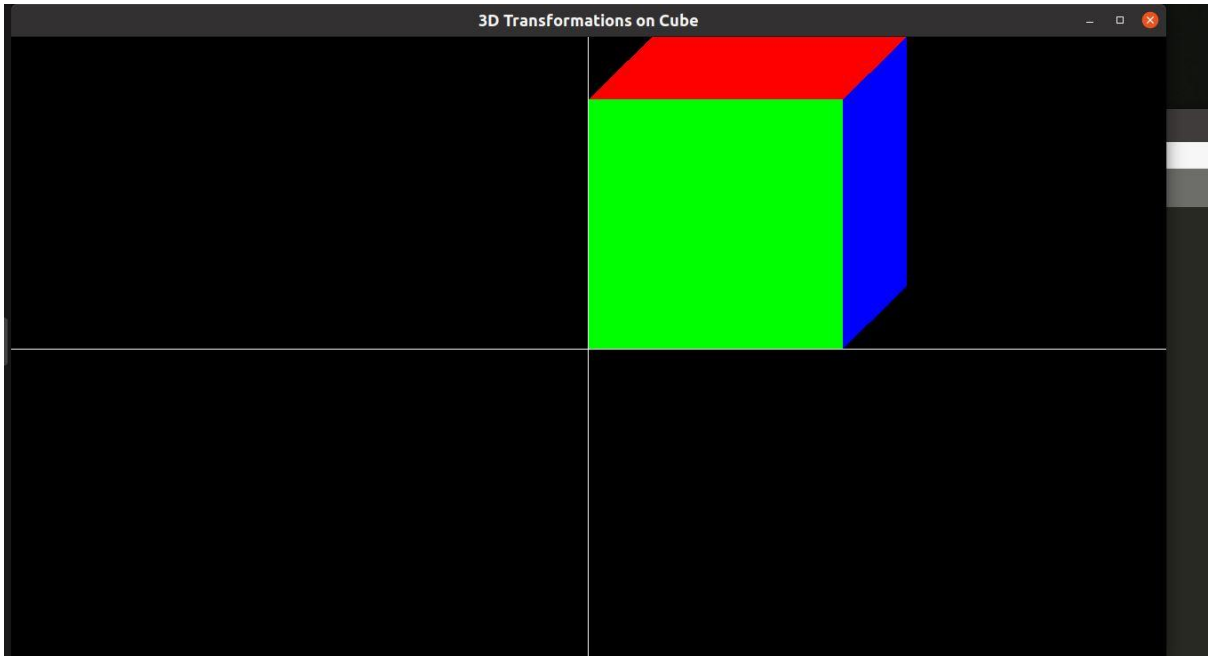
for translation, rotation & scaling glTranslatef() , glRotatef() & glScalef() functions are used. For rotation direction cosines of the vector which is the axis of rotation have been used to enable rotation of the cube about any arbitrary axis passing through the centre of the cube, which is the origin.

Testcases and Outputs Description:

Translation: The cube is translated by 100 units along +x axis and 0 units in y and z axis respectively.



Scaling: The cube is scaled by 5 units in all directions that is x, y and z.



Rotation: The cube is rotated by 180 degree along the z axis due to which the rotated cube is in the opposite quadrant of that original cube.