

Assignment 2

CSE471: Computer Graphics

Fall 2022

Computer Science and Engineering

Instructor: Ilwoo Lyu

Objectives

- The purpose of this assignment is to learn:
 - 3D geometric transformations
 - 3D coordinate systems (local \leftrightarrow world \leftrightarrow screen)
 - Camera view
 - Orthographic/perspective projection
- Please read the entire slides carefully before you start your work – Do NOT start coding without reading the whole slides!

Introduction

OpenGL View

- (old) OpenGL pipeline is fixed
 - Model view
 - Local to world to camera coordinates
 - `glMatrixMode(GL_MODELVIEW)`
 - Projection
 - Orthographic/perspective projection
 - `glMatrixMode(GL_PROJECTION)`
- Although modern OpenGL deprecates the fixed pipeline, we will use this approach to understand the computer graphics system better.

OpenGL View

- Matrices

- GL_MODELVIEW and GL_PROJECTION maintain their own matrices
- It is common to switch view before you draw something
- For example, in your display function:

Def display():

```
glMatrixMode(GL_PROJECTION)      # projection matrix is loaded
glLoadIdentity()
# do some projection
glMatrixMode(GL_MODELVIEW)       # view matrix is loaded
glLoadIdentity()
# do some geometric transformations
```

OpenGL View

- From GL_MODELVIEW and GL_PROJECTION, you can read their current state (matrix) using
 - `glGetDoublev(GL_MODELVIEW_MATRIX)`
 - `glGetDoublev(GL_PROJECTION_MATRIX)`
- Remember OpenGL is a state language, so the state matrices can be obtained as needed

OpenGL Geometric Transformations

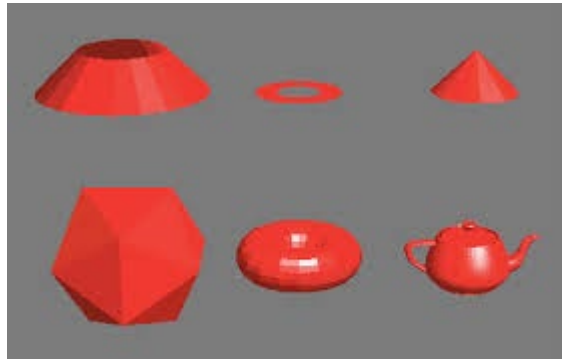
- There are two popular ways to define geometric transformations
 - Define matrices as we've done in assignment 1
 - Use OpenGL functions:
 - `glTranslatef()`
 - `glScalef()`
 - `glRotatef()`
 - These functions create and multiply a new matrix to the current matrix (projection or view matrix)
- The two approaches above are equivalent

OpenGL Projection Functions

- `glOrtho`
 - Parallel (orthographic) projection
- `glFrustum`
 - Perspective projection
- `gluPerspective`
 - Perspective projection using FoV
 - The `glu` library needs to be imported

GLUT Solid Shapes

- GLUT provides predefined 3D objects
 - glutSolidTeapot
 - glutSolidSphere
 - glutSolidTorus ...



Example Solid Shapes in GLUT

Assignment 2

Task 1

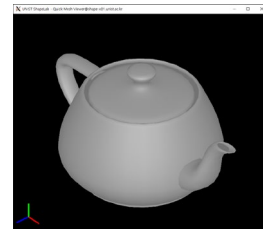
- Implement a camera view function
- You will need to:
 - Convert coordinate systems from world to camera
 - Implement a function
 - Arguments: "center of projection (px,py,pz)", "at (ax,ay,az)", "up (ux, uy, uz)"
 - Output: a camera view matrix (world to camera)

Task 2

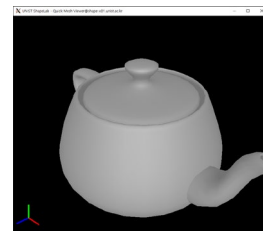
- Implement virtual trackball
- You will need to:
 - Update camera pose
 - Use $\min(\text{window_width}, \text{window_height})/2$ for the radius of your trackball
 - If $x^2 + y^2 > r^2$, scale down x and y to be $x^2 + y^2 = r^2$ (we covered this strategy in the lecture)
 - Implement a function
 - Arguments: "window coordinates (x, y) – starting point", "window coordinates (x, y) – destination point",
 - Output: a camera view matrix (rotation matrix)

Task 3

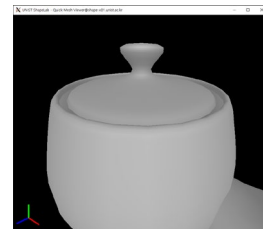
- Implement orthographic/perspective projection
- You will need to:
 - Interactively update projection (FoV)
 - Update the distance to the projection plane accordingly
 - Please review how FoV was defined in the lecture notes
 - Note: if you simply change FoV without distance update, you will see wrong scale in your distortion



Orthographic



Fov=45



FoV=90

Task 4

You can bind different keys/mouse behaviors, but if so, you must describe how to operate your tool

- Implement user interface
 - Rotation of the scene: drag left button
 - Rotation should appear real time; you should show the change of the scene interactively
 - Scaling/zoom of the scene: mouse wheel (if you don't have a wheel, use keys "+" and "-")
 - Wheel up (or +): scale up (zoom in)
 - Wheel down (or -): scale down (zoom out)
 - Translation/panning of the scene: drag right button
 - Your scene moves along right button movement
 - Fov change: "down" and "up" keys
 - down: reduce by 5 degree, up: increase by 5 degree
 - Min FoV: 0 (i.e., orthographic projection), Max FoV: 90; do not exceed these numbers!

Task 4

You can bind different keys/mouse behaviors, but if so, you must describe how to operate your tool

- Implement user interface
 - Press "c"
 - To reset translation/panning (translation = (0, 0, 0))
 - Press "t"
 - To reset viewpoint (rotation = identity)
 - Press "r"
 - To reset scale/zoom (scale = 1)
 - Press "0"
 - To reset projection (degree = 0, orthographic)
 - Press "esc"
 - To exit

Task 4

You can bind different keys/mouse behaviors, but if so, you must describe how to operate your tool

- Implement a “resize” function to interactively change window size
 - You can bind a resize event handler to `glutReshapeFunc`
 - You may want to adjust the projection matrix depending on your window size because the clipping volume will be affected
 - You could also customize viewport to handle the aspect ratio as needed
 - Your trackball’s radius should be synchronized with resize
 - Your 3D object should maintain its original ratio and fit (proper scale) to the screen
 - See also:
 - <https://www.opengl.org/resources/libraries/glut/spec3/node48.html>
 - <https://registry.khronos.org/OpenGL-Refpages/gl2.1/xhtml/gluPerspective.xml>
 - <https://registry.khronos.org/OpenGL-Refpages/gl2.1/xhtml/glOrtho.xml>
 - <https://registry.khronos.org/OpenGL-Refpages/gl4/html/glViewport.xhtml>

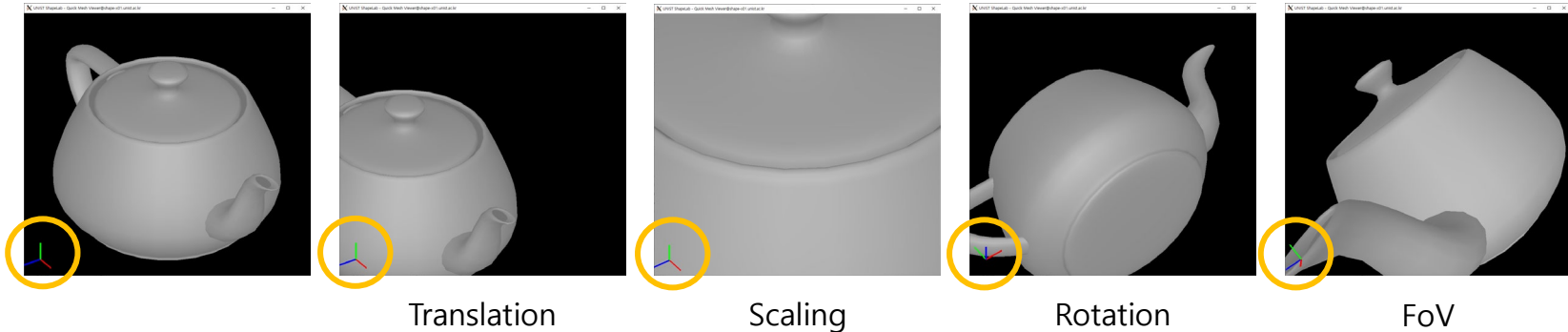
Task 5

- Configure a default environment
 - Use 3D object: GLUT teapot with size of 0.5
 - <https://www.opengl.org/resources/libraries/glut/spec3/node89.html>
 - Feel free to use test other objects, but make a "teapot" version for your submission
 - Set your camera to look at (0, 0, 0) from (0, 0, 1) and up=(0, 1, 0)
 - Use a sufficient clipping volume of projection that can fully contain the teapot
 - Set orthographic projection

Bonus (extra credit)

- There will be an extra credit if you can display the current axes of your scene at the bottom regardless of scaling/translation and projection types
- Submit a brief description of how you implement

Axes are drawn in orthographic projection independent of geometric transformations except for rotation



Be Aware

- Please DO (you will otherwise lose your credits):
 - Provide sufficient comments on what you implement; also, put Task # in your code
 - Provide user instructions to run your program
 - Do not make multiple files for your implementation – a single source file will be enough
- Feel free to use functions implemented in your first assignment that you might think useful

Submission

- What to submit
 - Your implementation and user manual
- Where to submit
 - Blackboard
- When to submit by
 - November 6th by midnight, 14 days including now (no extension – start as soon as possible)

Questions?

- Use Blackboard (Discussions → Course Board)
- Email/call the instructor
- Useful resources:
 - <https://www.opengl.org/>
 - <http://www.songho.ca/opengl/index.html>
 - <http://pyopengl.sourceforge.net/context/tutorials/index.html>