Stefan Johansson
stefanj@cs.umu.se

UMEÅ UNIVERSITY
DEPARTMENT OF COMPUTING SCIENCE

3D Studio—Part 1
Transformations

Please, read the introduction for the project first (found under 'Files/Project Resources' on Canvas).

Purpose

The purpose of Part 1 of the project is to get familiar with graphics programming in OpenGL and to implement basic functionalities for later use. It also aims to give an understanding of affine transformations and the use of callback functions. Before you start on the project, you should have finished both the two exercises and Workshop 1.

Specification

Do Workshop 1 and then create a program that can do the following:

1. The program should somehow be able to update a vector or a matrix with the following three transformations (see Exercise – Matrix routines):

   $M = T(dx, dy, dz) \cdot M$ (translation)
   $M = S(sx, sy, sz) \cdot M$ (scaling)
   $M = R_{\{x,y,z\}}(angle) \cdot M$ (rotation)

2. The user should be able to open and read an OBJ-file (see Exercise – OBJ files) by pressing 'o' and enter the file name. The file name can be read from the terminal window (in part 2 of the project, we will add a GUI that handles the file opening).

   The loaded object (3D-model) must be normalized to fit into a unit cube (1.0×1.0×1.0). This can be done either by updating the vertex data (at reading) or by applying a scaling as a first step to the model matrix.

   When loading a new object, reset all transformations, check that it is inside the viewing volume, and that the camera is pointing towards it (the camera is introduced in part 2 of the project).

3. It should be possible to do (at least) the following operations on the object. Rotating the object around the x- and y-axis with ±10.0° using the arrow keys. Translating the object ±0.1 units along the x-axis using the keys *J* and *L*, and along the y-axis using the keys *I* and *K*. Possible extensions are to add rotation and translation around the z-axis and scaling.

The update of the model matrix should be done as described in point 1 above and the picture redrawn.

You may use all or parts of the code provided for Workshop 1.

Stefan Johansson
stefanj@cs.umu.se

UMEÅ UNIVERSITY
DEPARTMENT OF COMPUTING SCIENCE

Instructions

This is an individual assignment. The code should follow good programming practice and if you are using any libraries that are not pre-installed, include them in the separate subfolders `./lib` and `./include` and set appropriate parameters in the Makefile. No written report should be handed in.

The code **must** compile and run on the Linux computers at CS's computer labs.

Due date is at latest November 21, 2023, 12.00. Time for demonstration is provided November 17 and 21, in MIT.A.146 and MIT.A.156, if nothing else is agreed upon.

Tips and Tricks

Draw polygons

Basically we have two ways of drawing polygons. If you have done the triangulation you can do a draw in one call, otherwise you have to repeat your call for each polygon. In all cases, you need to think how the vertex data should be copied to the buffer with `glBufferData`.

Method 1 – Tringle list

This is a simple method, but a vertex may occur several times in the buffer (why?). This method is used in the code provided on Workshop 1.

```
glDrawArrays(GL_TRIANGLES, 0, N);
```

Method 2 – Indexed triangle mesh

This method is more elegant since we do not have to duplicate vertex data, but we need an additional vector of indices. This is the **recommended** method and should be used in the project.

```
glDrawElements(GL_TRIANGLES, count, GL_UNSIGNED_SHORT, indices);
```

Method 3 – multiple sets of triangles

It is also possible to draw multiple triangles in one call. It is similar to Method 1.

```
glMultiDrawArrays(GL_TRIANGLES, first, count, primcount);
```