CSC 305: Assignment 1 README

The following document outlines the usage, algorithms, data structures and extra features of my assignment 1 submission. Note that this has been tested in **QT 4.8.6 with QT Creator 3.2.**

# Usage

To use my application, start the program with Command + R. Click on the screen to add polygons. While adding vertices to a polygon, click the first vertex of the current polygon to close the polygon. Apply a matrix to all polygons by filling out the matrix fields and pushing that matrix to the stack. The buttons are labeled with their use. The buttons can be used in any order, and will do as they are labeled.

# Algorithms

There aren’t that many algorithms, given that most of this assignment was about mapping matrices to polygons in OpenGL through QT. I did implement a few basic LinAlg library functions such as Inverse to invert a matrix using the CoFactor method and VectorTransform to multiple any arbitrary QVector3D by any arbitrary QMatrix3x3.

# Data Structures

QVector: A dynamic array. I used this data type because of the convenience given with built-in class member function calls such as .last() which returns a reference to the element at the last index. Also nice because it is dynamic, which means I don’t have to worry about size.

QVector3D: An object that stores x, y, and z, being used to represent a point in the Cartesian plane, with z as the perspective value. I used QVector3D because it allowed for built-in class member function calls.

QMatrix3x3: A 3x3 matrix type, stores a 2D array equivalent to [3][3]. I used this type because it allowed for automatic matrix multiplication using an overloaded version of multiply.

QStack: A built-in stack type that allows me to add any number of elements to it. This is being used to keep track of the transformation matrices on the screen, since the program only ever needs the top element at any given time, and this data structure allows the user to input any number of matrices.

# Extra Features

Features that I included that were not listed as requirements include:

* The vertices of each polygon have their own colour set.
* After applying transformation matrices, the user is still able to add points where they click
* Users have the ability to clear polygons and clear the matrix stack independent of each other, without having to restart the application
* Users have the ability to apply or not apply the matrix stack, as well as the same freedom for the viewport scale
* The transformation matrices are being displayed so that the user can see the matrices that are being applied to their polygons. I did all of the formatting for this myself.