Advanced Games Engine Creation

3D Transformations

In this practical you will

- Practice representing geometrical transformations as homogeneous matrices
- Practice transforming arrays of points using matrices
- Practice multiplying matrices to represent composite transformations
- Experiment with using OpenGL methods to position objects in a 3D scene

For the first part of this practical you will use pen and paper – not at SDL program! Your tutor will discuss the answers with the class as you work through the questions.

1. Consider a triangle with vertex coordinates (0, 0, 0), (1, 0, 0), (0, 1, 0). Sketch what it should look like in the x,y plane.

   Now sketch what the triangle should look like after each of the transformations below. Start with the original triangle each time, don't combine the transformations together. You don't need to do any calculations for this question, just use your knowledge of what effect each transformation should have.

   a) The identity transformation
   b) A scaling by a factor of 3 in all dimensions
   c) A scaling by a factor of 3 in x, 2 in y and 1.5 in the z direction
   d) A rotation of 30 degrees about the z axis
   e) A rotation of π radians about the z axis
   f) A translation of -3 units in the x direction, +2 units in the y direction, and 0 units in the z direction

2. Express each of the transformations of part 1 as a 3D homogeneous matrix.

3. Use multiplication to transform the vertices of original triangle of question 1 (0, 0, 0), (1, 0, 0), (0, 1, 0) by each transformation matrix of question 2. Do your answers correspond to the sketches you drew in question 1?

4. Now let's see what how to do this in a SDL/OpenGL program. Start with (a copy of) the OpenGL_SDL Base Project framework. Modify the GameScreenLevel1.cpp class so it is the same as appendix 1 of this document. You will also need to add the function prototype

   ```
   void DrawTriangle();
   ```

   to the list of private members in `GameScreenLevel1.h`

   Make the changes and run the program. You should see the original triangle in white, plus a red triangle which has undergone scaling transformation b)

Add code to the Render() method to render the triangle after it has undergone each of transformations c), d), e) and f). Display each triangle in a different colour.

Back to pen and paper

5. Sketch the result you would expect if the following transformations were applied to the triangle of question 1:

a) Scaling by a factor of 3 in x, 2 in y and 1.5 in the z direction, followed by a rotation of 30 degrees about the z axis.
b) Rotating by 30 degrees about the z axis, followed by scaling by a factor of 3 in x, 2 in y and 1.5 in the z direction
c) Scaling by a factor of 3 in x, 2 in y and 1.5 in the z direction, followed by a translation of -3 units in the x direction, +2 units in the y direction, and 0 units in the z direction
d) A translation of -3 units in the x direction, +2 units in the y direction, and 0 units in the z direction, followed by scaling by a factor of 3 in x, 2 in y and 1.5 in the z direction

6. For each of the cases of question 3, multiply the two transformation matrices together to generate a compound transformation matrix, and apply this matrix to the original triangle vertices. Are the results as you expect?

7. Now try programming the transformations of question 5/6. Here's transformation a) to get you started:

```
// draw a red triangle scaled by a factor 3, 2 and 1.5, then
// rotated 30 degrees about the z axis
      glColor3f(1.0, 0.0, 0.0);
      glPushMatrix();
      glRotatef(30, 0, 0, 1);
      glScalef(3, 2, 1.5);
      DrawTriangle();
      glPopMatrix();
```

Notice that the transformations are applied in the opposite order that they are specified.

8. On paper, sketch a rectangle with the vertices (-2, 0, 0), (1, 0, 0), (1, 2, 0), (-2, 2, 0). What would you expect to happen if you applied a simply scaling matrix (scale by a factor of 1.5 in all dimensions) to this rectangle? What about rotating by 30 degrees about the z axis? What if you wanted to apply both simple transformations – does the order matter?

What is the coordinates of the centre of this rectangle? Write down a series of 4 matrices that would have the effect of scaling the rectangle by a factor of

1.5 and rotating by 30 degrees about its centre. Sketch what the rectangle should look like after the matrices are applied.

Multiply the matrices together, then apply the resultant transformation matrix to the rectangle coordinates. Are the results as you expect?

9. Add the rectangle and its transformation of question 8 to your SDL program and verify it behaves as expected. Hint: use

```
glBegin(GL_QUADS);
```

and don't forget you need to specify 4 vertices.

10. Now with added teapots! Experiment with creating a 3D scene with multiple teapots and other GLUT primitives.

See the OpenGL reference pages https://www.opengl.org/sdk/docs/man2/ for documentation of all the OpenGL functions

Appendix 1: GameScreenLevel 1.cpp starting point (changes from base project in **bold**)

```
#include "GameScreenLevel1.h"
#include <time.h>
#include <windows.h>
#include <GL\gl.h>
#include <GL\glu.h>
#include "../gl/glut.h"
#include "Constants.h"

using namespace::std;

//-----------------------------------------------------------------------
------------------------------

GameScreenLevel1::GameScreenLevel1() : GameScreen()
{
	srand(time(NULL));

	glEnable(GL_DEPTH_TEST);

	glViewport(0, 0, SCREEN_WIDTH, SCREEN_HEIGHT);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();

	float aspect = (float)SCREEN_WIDTH / (float)SCREEN_HEIGHT;
	// not using perspective projection for this exercise
	//    gluPerspective(60.0f,aspect,0.1f,1000.0f);

	// use orthographic (flat) projection
	glOrtho(-5, 5, -5 / aspect, 5 / aspect, 0.1f, 1000.0f);

	glMatrixMode(GL_MODELVIEW);

	glEnable(GL_CULL_FACE);
	glEnable(GL_DEPTH_TEST);

	//clear background colour.
	glClearColor(1.0f, 1.0f, 1.0f, 1.0f);


	// specify line rather than filled polygons
		glPolygonMode(GL_FRONT, GL_LINE);
}

//-----------------------------------------------------------------------
------------------------------

GameScreenLevel1::~GameScreenLevel1()
{
}

void GameScreenLevel1::DrawTriangle() {
	glBegin(GL_TRIANGLES);
		glVertex3f(0.0f, 0.0f, 0.0f);
		glVertex3f(1.0f, 0.0f, 0.0f);
		glVertex3f(0.0f, 1.0f, 0.0f);
	glEnd();
}
```

```
//-------------------------------------------------------------------
-----------------------------

void GameScreenLevel1::Render()
{
	//Clear the screen.
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

	glLoadIdentity();
	gluLookAt(0.0f, 0.0f, 10.0f,
			0.0f, 0.0f, 0.0f,
			  0.0f, 1.0f, 0.0f);

	// delete the original drawing code here and add:

	// draw the first triangle
	glColor3f(0.0, 0.0, 0.0);
	DrawTriangle();

	//-------------------------------------------------------------
	//    Question 4
	//-------------------------------------------------------------

	// draw a red triangle scaled by a factor of 3 in all
dimensions
	glColor3f(1.0, 0.0, 0.0);
	glPushMatrix();
	glScalef(3, 3, 3);
	DrawTriangle();
	glPopMatrix();

	// draw a green triangle scaled by 3, 2 and 1.5
	// hint – use glScalef again

	// draw a blue triangle rotated by 30 degrees about the z axis
	// hint - use the glRotatef function, which has the prototype:
	// void glRotatef(float angle, float x, float y, float z)

	// draw a yellow triangle rotated by 180 degrees about the z
axis

	// draw a cyan triangle translated by (-3, 2, 0)
	// hint – use the glTranslatef() function, prototype
	// void glTranslatef(float x, float y, float z);

}

//-------------------------------------------------------------------
-----------------------------

void GameScreenLevel1::Update(float deltaTime, SDL_Event e)
{
	mCurrentTime += deltaTime;
}

//-------------------------------------------------------------------
	-------------------------------
```