**Kathmandu University**

**Dhulikhel, Kavre**

**Department of Computer Science and Engineering**



**Simultaneous Translation Rotation and Scaling**

COMP 342.

**Submitted by**

Sujal Paudel  
Roll no: 31

CE 3rd year 2nd semester

**Submitted to**

Mr. Dhiraj Shrestha

Assistant Professor

Department of Computer Science and Engineering

**Submission Date:**

27th June, 2018

#### Chapter1: Introduction

##### **1.1 Synopsis**

In this program the functionality of OpenGL is explored. This program enables to create 100 triangle, which is then translated, rotated and scaled simultaneously. The process of translation, rotation and the scaling is performed side by side with the help of concatenation of these individual events

##### **1.2 Objectives**

The objective of this program is to explore the feature that comes along with the OpenGL library, which in turn helps to create some fine art effects, in this case the simultaneous translation rotation and scaling.

#### Chapter2: Design and Implementation

**2.1 System Requirement Specification**

|  |  |
| --- | --- |
| * **Programming Languages** | C++ |
| * **Operating System** | Ubuntu 16.04 LTS |
| * **Graphics Library** | OpenGL |

##### **2.2. System Requirement Description**

The **OpenGL Utility Library** (**GLU**) is a computer [graphics library](https://en.wikipedia.org/wiki/Graphics_library) for [OpenGL](https://en.wikipedia.org/wiki/OpenGL).

It consists of a number of functions that use the base OpenGL library to provide higher-level drawing routines from the more primitive routines that OpenGL provides. It is usually distributed with the base OpenGL package. GLU is not implemented in the embedded version of the OpenGL package, [OpenGL ES](https://en.wikipedia.org/wiki/OpenGL_ES).

#### Chapter3: Methodology

##### **3.1 The Details**

The 100 triangles are scaled, rotated and translated slightly differently. The glm translate function takes as an input it likes to transform, in this case, it’s a default identity matrix. Here we translate in the x direction based on the sign value of time counter. The time counter is initially 0.01 and it is increasing every frame. In Y direction we don’t translate at all. In Z direction we translate based on which triangle we ate rendering. Rotate function takes the identity matrix, some angle of rotation and an angle of rotation we would like to rotate, scale function takes identity matrix and some vector value for how much to scale in x y z direction.

##### **3.2 Algorithm**

Step 1: Start

Step 2: Create 4 \* 4 matrices model view and projection.

Step 3: Create compile and link shader.

Step 4: Get the attributes uniform location of shader.

Step 5: Define view matrix for right handed coordinate system.

Step 6: Define projection matrix with perspective function.

Step 7: Send this value to the GPU binding them.

Step 8: For model transform draw 100 triangles, with translation rotation and scaling.

Step 9: Multiply these translation rotation and scaling matrices to get model matrices.

Step 10: Pass these model matrices to the GPU.

##### **3.3 Flow Chart**

Create 4 \* 4 matrices model view and projection.

Define view matrix, projection matrix

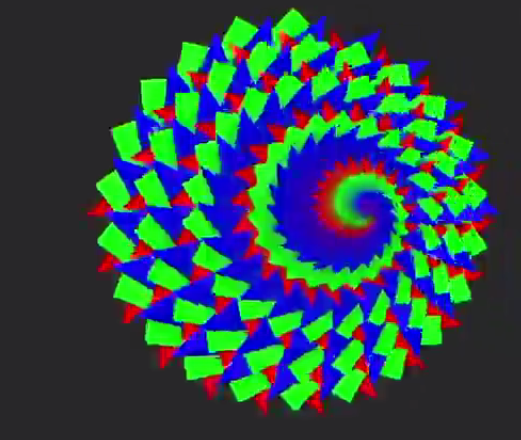
Multiply these translation rotation and scaling matrices

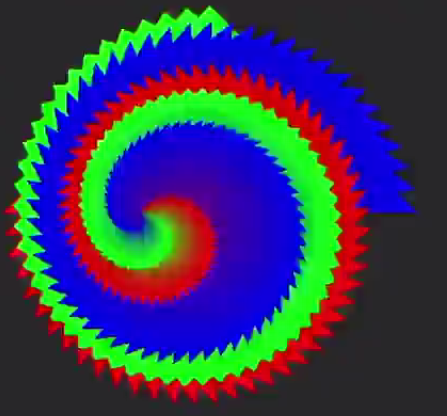
Model matrices to the GPU

Compute and result

Fig 3.3 Flow Chart

#### Chapter4: Results





Chapter5: Conclusion  
The program helped to make a better understanding of glm translate, glm rotate and glm scale along with glm perspective and glm look at to recreate the fix functionality pipeline. The matrices were generated and generated matrices were then sent into the GPU for processing.

**References:**

* ​ En.wikipedia.org. (2018).Available at:

https://en.wikipedia.org/wiki/OpenGL [Accessed 11 June

2018]

* OpenGL Mathematics

Available at:

https://glm.g-truc.net/0.9.9/index.html [Accessed

11 June 2018]