# Implementation of Translation, Scaling and Rotational Transformation using OpenGL

Tarun Kumar
UE143103
Computer Science Engineering
University Institute of Engineering and Technology
Panjab University
Chandigarh, 160030
Email: tarunk.1996@gmail.com

***Abstract*—Building an OpenGL application to draw various 2D shapes using lines, circles and ellipses. Provide an option to user, which lets user to translate(i.e scaling, translation and rotation) the drawn shapes. FreeGlut library is used in development of this application.**

## I. Transformations

Transformation means changing some graphics into something else by applying rules. We can have various types of transformations such as translation, scaling up or down, rotation, shearing, etc. When a transformation takes place on a 2D plane, it is called 2D transformation.

Transformations play an important role in computer graphics to reposition the graphics on the screen and change their size or orientation.

The three basic Transformation are as follows:

- Translation
- Scaling
- Rotation

### *A. Translation*

A translation moves an object to a different position on the screen. To translate a point in 2D can be done by adding translation coordinate

$(t_x, t_y)$ to the original coordinate $(X, Y)$ to get the new coordinate $(X', Y')$.

$$X = X + t_x$$

$$Y = Y + t_y$$

Let $P(x, y)$ be the coordinate of one of the points of the object, and suppose this is to be translated by $(t_x, t_y)$, then the new coordinates of point $P$, which becomes $P(x', y')$ can be calulated by :

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

which is basically

**P'=T.P**

Here the coordinates have been represented in homogeneous form and the net result becomes

$$x' = x + t_x$$

$$y' = y + t_y$$

### *B. Scaling*

To change the size of an object, scaling transformation is used. In the scaling process, it will either expand or compress the dimensions of the object. Scaling can be achieved by multiplying the original coordinates of the object with the scaling factor to get the desired result. Assume that the original coordinates are $(X, Y)$, the scaling factors are $(S_x, S_y)$, and the produced coordinates are $(X, Y)$.

$$X' = X.S_x$$

$$Y' = Y.S_y$$

$$\begin{bmatrix} X' \\ Y' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

hence

**P' = P . S**

### *C. Rotation*

In rotation, to rotate the object at particular angle (theta) from its origin. Consider a point $P(x, y)$, which makes an angle $\phi$ with positive direction of x-axis. Then

$$x = rcos\phi$$

$$y = rsin\phi$$

$P'(X', Y')$

$$X' = rcos(\phi + \theta) = rcos\phi cos\theta - rsin\phi sin\theta$$

$$Y' = rsin(\phi + \theta) = rcos\phi sin\theta + rsin\phi cos\theta$$

hence,

$$X' = xcos\theta - ysin\theta$$

$$Y' = xsin\theta + ycos\theta$$

The matrix representation is given by

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

**P'=R.P**

## II. About FreeGlut

FreeGLUT is a free-software/open-source alternative to the OpenGL Utility Toolkit (GLUT) library. GLUT was originally written by Mark Kilgard to support the sample programs in the second edition OpenGL 'RedBook'. Since then, GLUT has been used in a wide variety of practical applications because it is simple, widely available and highly portable.

GLUT (and hence FreeGLUT) takes care of all the system-specific chores required for creating windows, initializing OpenGL contexts, and handling input events, to allow for trully portable OpenGL programs.

FreeGLUT is released under the X-Consortium license.

## III. Some FreeGlut Functions

Following are functions used in developing the application with brief intro about it.

- **Creating A Window**
  The glutInitWindowPosition and glutInitWindowSize functions specify a desired position and size for windows that freeglut will create in the system. This glutCreateWindow("Line Drawing Algorithm") will create a Glut Window with a Title Line Drawing Algorithm on it.
- **Setting Background Color**
  This will set the background color of the screen to white.

```
glClearColor(1, 1, 1, 0);
glClear(GL_COLOR_BUFFER_BIT);
glFlush();
```

- Plotting X and Y axis in the window. Set all the pixel along X-axis to black color and similar to that in Y-axis. (Remember according to FreeGlut co-ordinate system (0,0) will lie at the top-left corner of the window, So it should be shifted to the center of the screen first). In order to create X and Y - axis, you can either use inbuilt line drawing function or can set each pixel along that line to color black. Both will gives the same result.
- **Showing Text**
  To display some text at any co-ordinate in the OpenGL window. It can be done by using the following code:

```
drawStrokeText(*string, x, y, z)
{
 char *c;
 glPushMatrix();
 glTranslatef(x, y+8,z);
 glScalef(0.09f,0.08f,z);
 for (c=string; *c != '\0'; c++)
 {
  glutStrokeCharacter(GLUT_STROKE_ROMAN,*c);
 }
 glPopMatrix();
}
```

  In the above function drawStrokeText accepts the string, x, y and z co-ordinates. It will print the string as in character by character format. $GLUT_STROKE_ROMAN$ is the text format that will be displayed on the screen.
- **Creating a Button**
  FreeGlut library as such doesnt include any provision for creating a clickable buttons and associating an action when a button is clicked. So in order to create a button, Create a button look alike box which will perform some action whenever there will a left mouse click on it. Following code will create a box in the glut screen of 80x25 size with a background color of red. GL_QUADS is used to create a quadrilateral in the OpenGL window.

```
glBegin(GL_QUADS);
glColor3f(1,0, 0);
glVertex2i(0,0);
glVertex2i(80,0);
glVertex2i(0,25);
glVertex2i(80,25);
glEnd();
drawStrokeText("Button",5,0,0);
```

  Now, to make this box a fully functional button. Mouse Left click event must be introduced in it.

```
mouse(int btn, int state, int x, int y)
  if(btn == GLUT_LEFT_BUTTON
&& state == GLUT_DOWN)
//Do some event
```

- **OpenGL Color Functions** To set the color display mode glut initialize display mode is used :

```
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB)
```

  Setting the color of a pixel can be done by 3 methods: Floating point values

```
glColor3f(0.0,1.0,0.0)
```

  An array specification

```
glColor3fv(colorArray)
```

  Integer Values

```
glColor3i(0,255,0)
```

- **Get properties of certain pixel** To read the seed pixel and extract the properties out of it is can be done by using $glReadPixels()$ function. Syntax of $glReadPixels()$

```
glReadPixels(X,Y,W,H,format,type,data);
```

  Here, $X, Y$ specifies the window coordinates of the first pixel that is read from the frame buffer. $W, H$ specifies the dimensions of the pixel rectangle, width and height of one correspond to a single pixel. $format$ specifies the format of the pixel data. $type$ specifies the data type of the pixel data. $data$ returns the pixel data(can be used to return the color of the seed pixel).

# OUTPUTS :

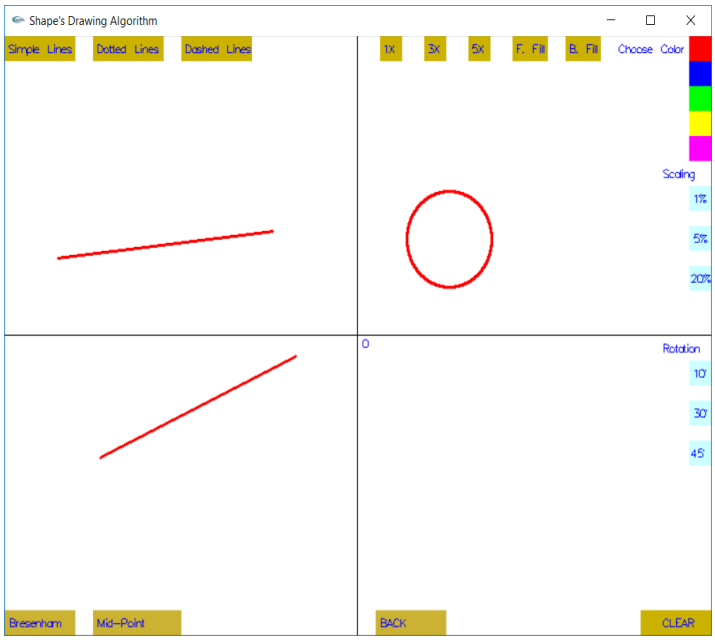


Fig 1: Sample Figure

2 Simple Lines and
1 Circle.


Fig 2: Translation
Translation of a Line and
a Circle using Right Click

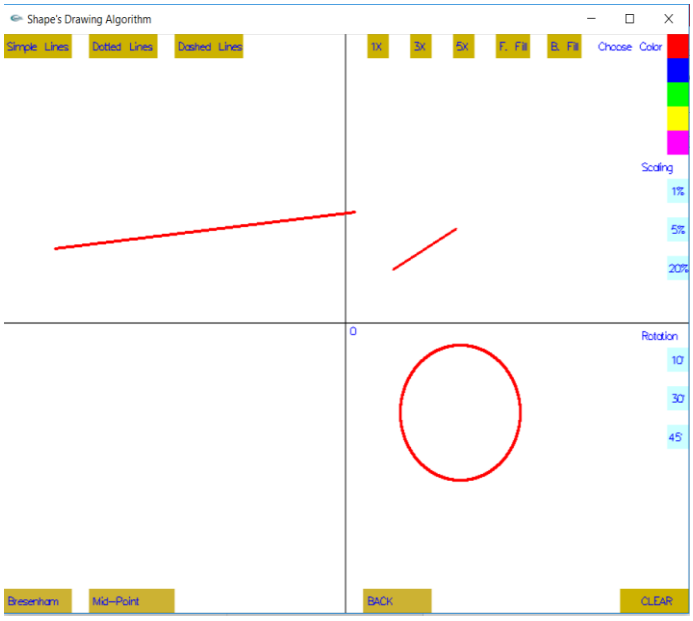


Fig 3: Scaling

Scaling up and down of all
the shapes using
keyboard input.

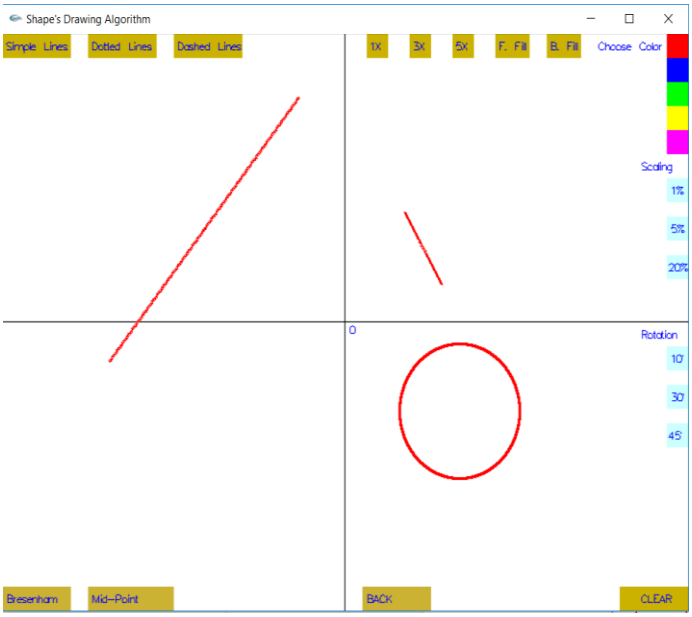


Fig 4: Rotation

Rotation of 2 lines using keyboard as
input for clockwise and anticlockwise
rotation.