COMP 3271

Programming Assignment 1

Submission

- Deadline:

  11:59pm, Oct 17, 2019 HKT.

- Submission:

  Code.cpp

If you want to change any other files or implement the program without the template, please email me(wenhua00@hku.hk) before submission.

Outline

- Review of Homogeneous Coordinates and Transformation Matrix
- 2D Transformation in OpenGL
- Fractal Drawing – Programming Assignment
    - About the Template
    - About the Task
    - Submission

Outline

- Review of Homogeneous Coordinates and Transformation Matrix
- 2D Transformation in OpenGL
- Fractal Drawing – Programming Assignment
  - About the Template
  - About the Task
  - Submission

Review: Homogeneous Coordinates

- Common Transfer Matrix
  - Translation
  - Rotation
  - Scaling
- Add an extra dimension

Euclidean formulation

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & b \\ d & e \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} c \\ f \end{pmatrix}$$

$$\mathbf{p}' = \mathbf{M}_{2\times 2} \quad \mathbf{p} \quad + \quad \mathbf{T}$$

Homogeneous formulation

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$$\overline{\mathbf{p}}' = \mathbf{M}_{3\times 3} \quad \overline{\mathbf{p}}$$

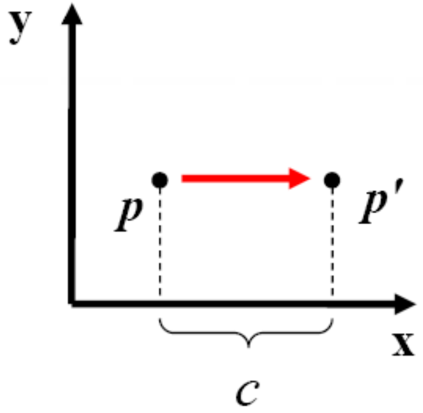
Review: 2D Transformation

- Translation Matrix

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & x_{offset} \\ 0 & 1 & y_{offset} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

- Example : Translate “c” units in x direction

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & c \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x+c \\ y \\ 1 \end{pmatrix}$$



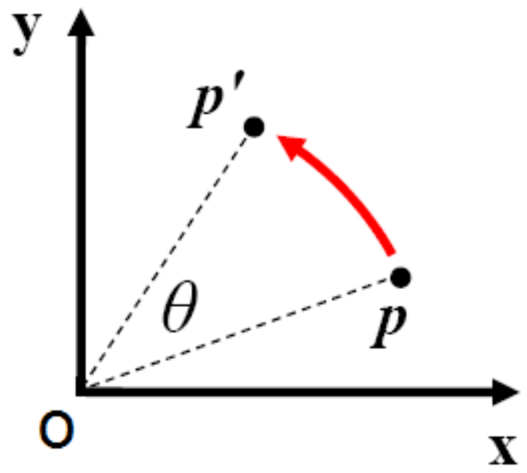
Review: 2D Transformation

- Rotation Matrix

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

- Example : Rotate θ around the origin point

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



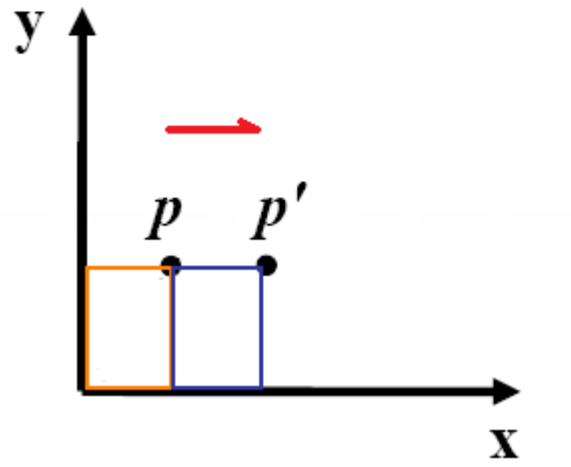
Review: 2D Transformation

- Scaling Matrix

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

- Example: Scale 2 in x coordinate

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} 2x \\ y \\ 1 \end{pmatrix}$$

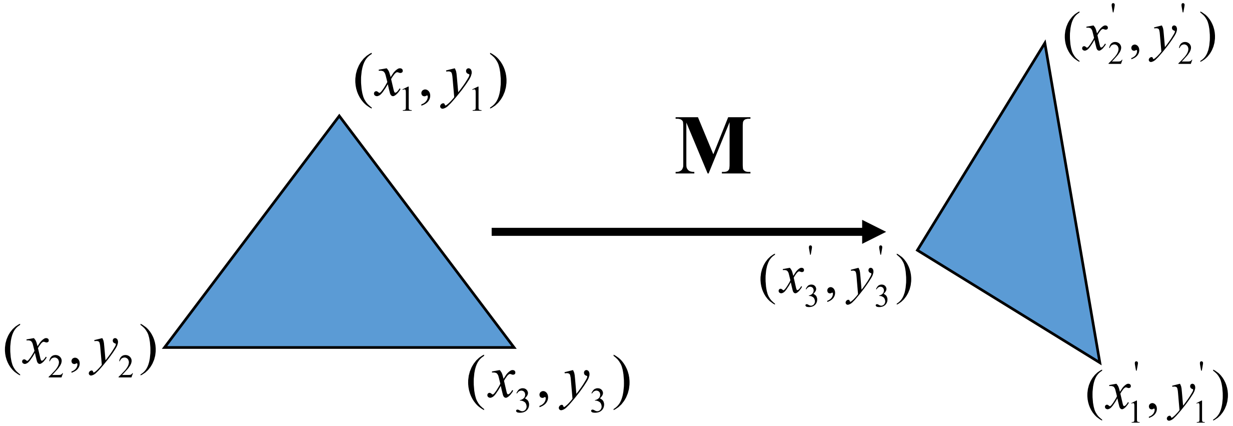


Review: Arbitrary Affine Transformation

$$\begin{pmatrix} x_1' & x_2' & x_3' \\ y_1' & y_2' & y_3' \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{pmatrix}$$

$$\mathbf{T'} = \mathbf{M} \quad \mathbf{T}$$

$$\mathbf{M} = \mathbf{T'} \quad \mathbf{T}^{-1}$$



Outline

- Review of Homogeneous Coordinates and Transformation Matrix
- 2D Transformation in OpenGL
- Fractal Drawing
  - About the Template
  - About the Task
  - Submission

Model View Matrix

$$\begin{pmatrix} x' \\ y' \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & 0 & c \\ d & e & 0 & f \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \\ 1 \end{pmatrix}$$

$$\overline{\mathbf{p}}\mathbf{'} = \mathbf{M} \; \overline{\mathbf{p}}$$

- In OpenGL we use Model View Matrix to represent the transformation matrix M.
- OpenGL provides some functions to modify matrix M.

Model View Matrix

- Transformation based on OpenGL

Translation Matrix

$$\begin{pmatrix} x' \\ y' \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & c \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \\ 1 \end{pmatrix}$$

Rotation Matrix

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

Scaling Matrix

$$\begin{pmatrix} x' \\ y' \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \\ 1 \end{pmatrix}$$

*OpenGL Functions

- void **glTranslate**{fd}(TYPEx, TYPE y, TYPEz);

  Translate an object by the given x, y, and z.

- void **glRotate**{fd}(TYPE angle, TYPE x, TYPE y, TYPE z);

  Rotate an object in a counterclockwise direction about the ray from the origin through the point (x, y, z). The angle parameter specifies the angle of rotation in degrees.

- void **glScale**{fd}(TYPEx, TYPE y, TYPEz);

  Stretch, shrink, or reflect an object along the axes. Each x, y, and z coordinate of every point in the object is multiplied by the corresponding argument x, y, or z.

OpenGL Functions

- void **glMultMatrix**(const GLdouble *m);

  GLdouble m[16];

  M=M*m, where M is the Model View Matrix.

- OpenGL Matrix is column major:
  - Example: Gldouble m[16] layout is

m[0]	m[4]	m[8]	m[12]
m[1]	m[5]	m[9]	m[13]
m[2]	m[6]	m[10]	m[14]
m[3]	m[7]	m[11]	m[15]

OpenGL Functions

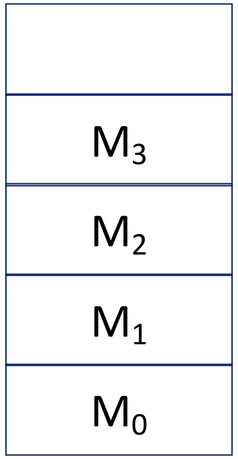
- void **glPushMatrix**(void);

  Put current model view matrix into matrix stack.

  Current matrix is unchanged.

- Void **glPopMatrix**(void);

  Pop matrix in stack to replace current matrix.



M_3

M_2

M_1

M_0

How can we draw fractal using these transformations?

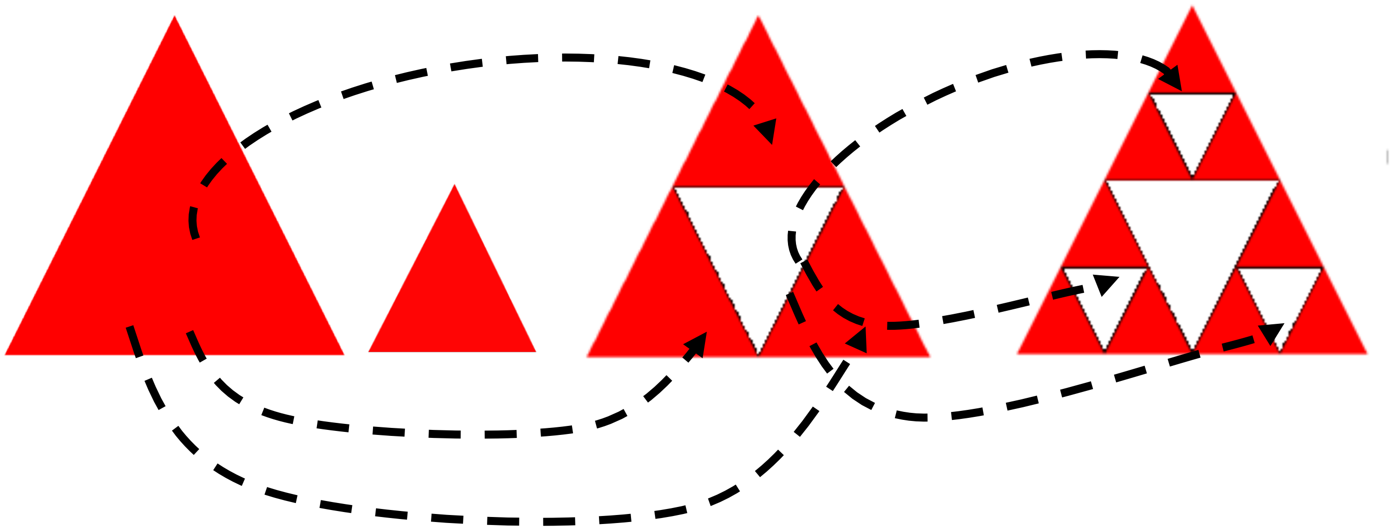
Iterated Function System

Iterated Function System(IFS)

- IFS is a method of constructing fractals, the resulting constructions are always self-similar.
- The fractal is made up of the union of several copies of itself, each copy being transformed by a function.

Iterated Function System - Example

- Sierpinski gasket triangle evolution
  - Start with any triangle in a plane
  - Shrink the triangle to ½ height and ½ width, make three copies, and position the three shrunken triangles so that each triangle touches the two other triangles at a corner
  - Repeat step 2 with each of the smaller triangles



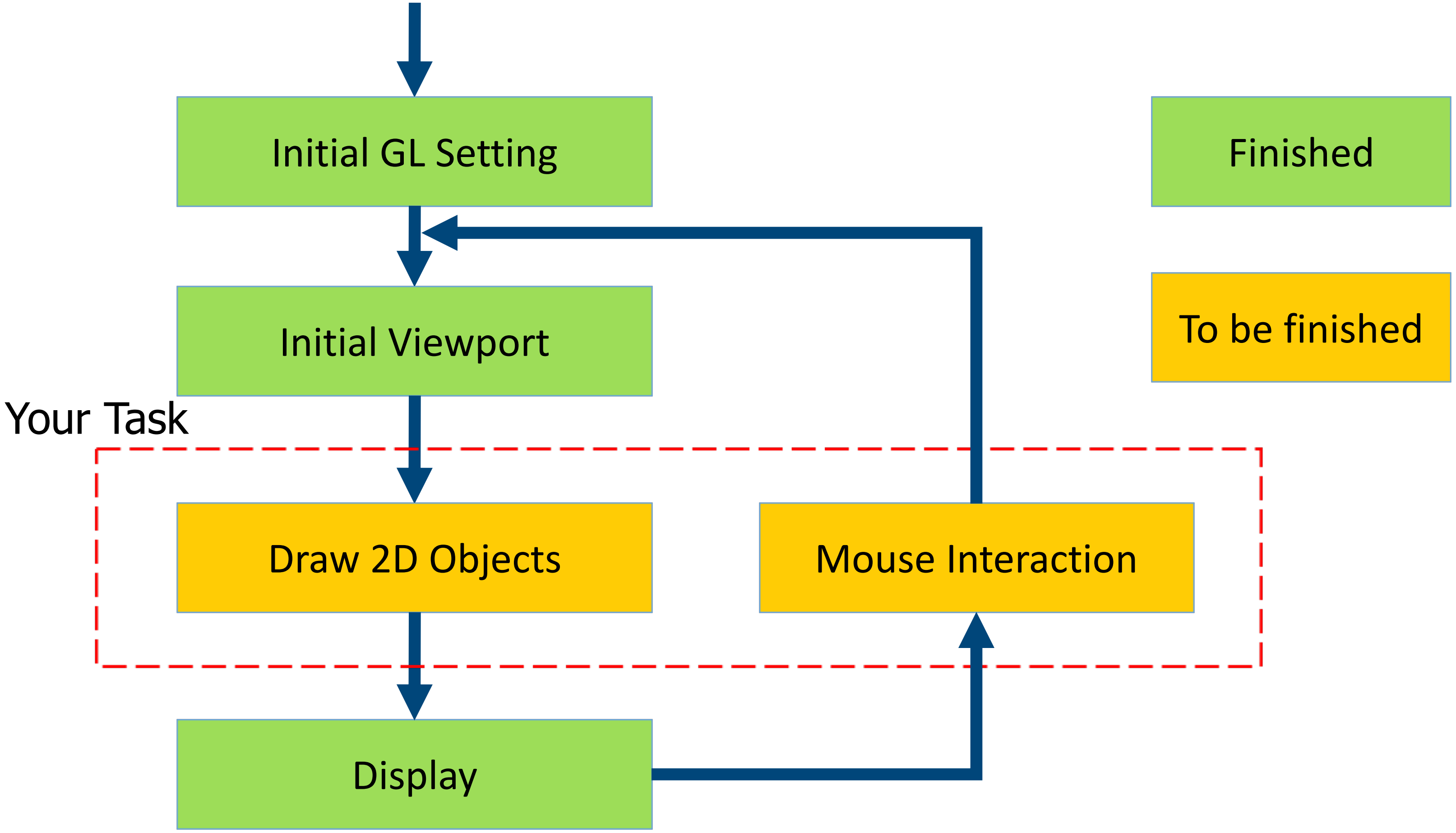
Outline

- Review of Homogeneous Coordinates and Transformation Matrix
- 2D Transformation in OpenGL
- Fractal Drawing – Programming Assignment
    - About the Template
    - About the Task
    - Submission

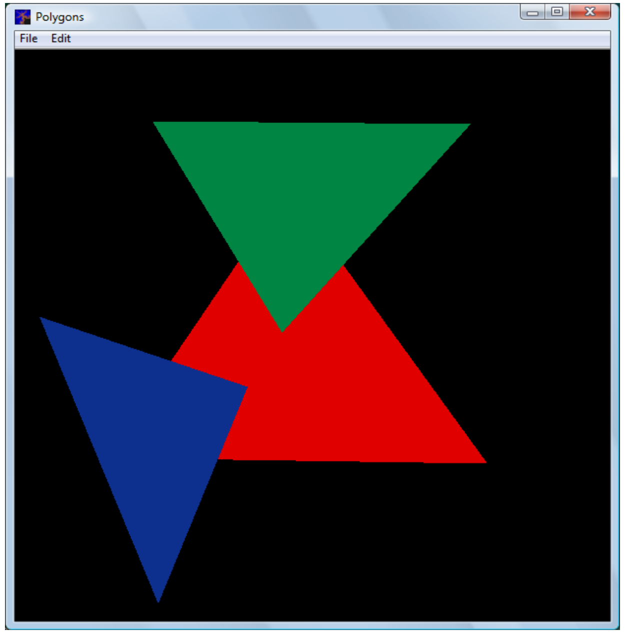
About the Template

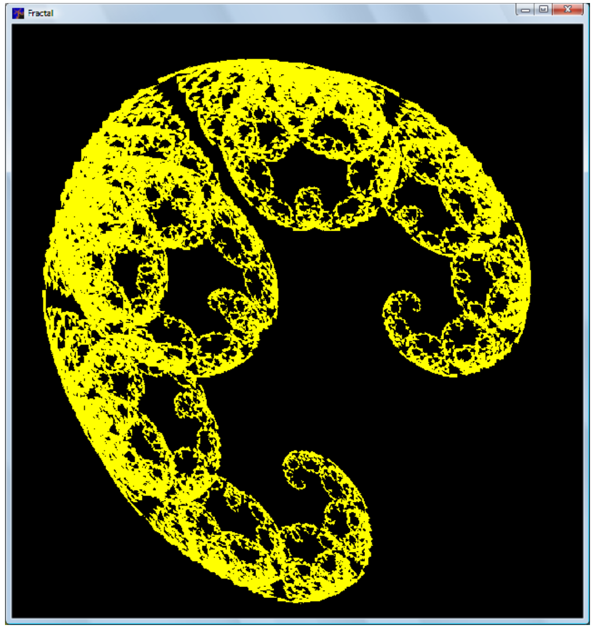
- MS-Windows: Use Visual Studio 2019, Interface based on MS Foundation Class (MFC). Written in C++.
- Download from the course webpage.
- Double-click the file D2CG.sln to open the project.
- A sample program D2CG.exe in the folder “bin”.
- Template includes:
    - An interface with two window for 2D objects rendering.
    - OpenGL init and projection setup.

About the Template



About the Template





Finished View

Outline

- Review of Homogeneous Coordinates and Transformation Matrix
- 2D Transformation in OpenGL
- Fractal Drawing – Programming Assignment
  - About the Template
  - About the Task
  - Submission

Your Task

The functions to fill in:

```
//For Mouse Interaction

void MouseInteraction(GLdouble m_x,GLdouble m_y)

//For Drawing 2D Objects

void DrawTriangles ()

void AffineMatricesCalculation()

void RecursiveFractal(int k)
```

The Data Structures to Use

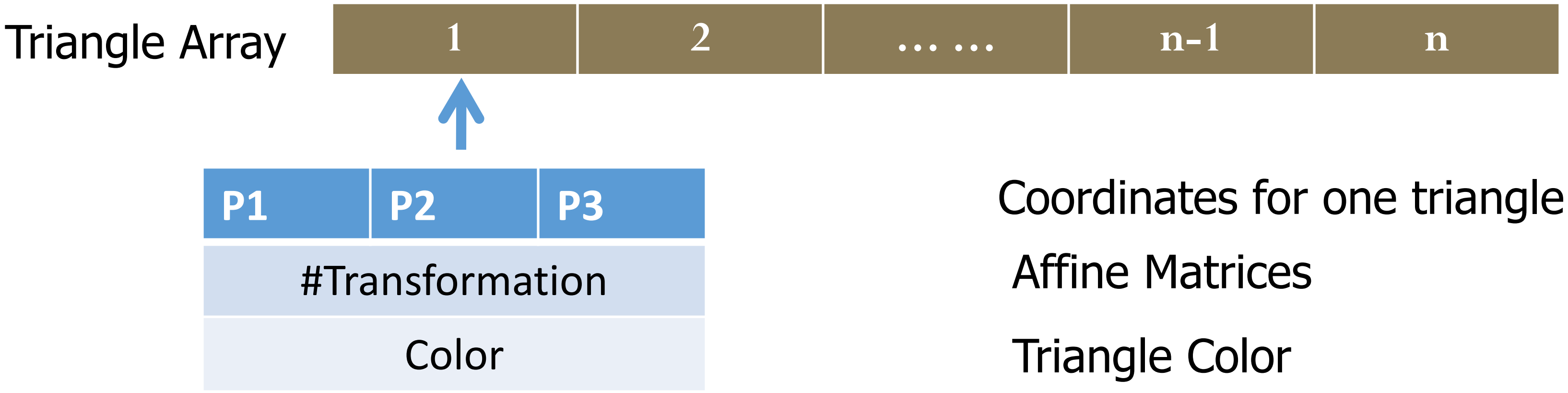
Data Structure Application

triangle_to_draw : stores vertices of a triangle

Triangles : list for triangles

color_array : list of different colors

affine_matrices : vector<D3Matrix>



Triangle Array

1	2	… …	n-1	n

P1	P2	P3
#Transformation		
Color		

Coordinates for one triangle

Affine Matrices

Triangle Color

The Data Structures to Use

- Vector Operations

```
#include  <vector>
//Define Stack
vector<Triangle>  TriangleStack;


//Usage:
vector::size()   //Get vector size.


TriangleStack[i]  //Get value of ith item.
```

```
typedef struct Triangle{
        GLdouble vertices[3][2];
        GLdouble matrix[3][3];
        int          color_index;
} ;
```

Your Task

Function : Mouse Interaction

void MouseInteraction(GLdouble m_x,GLdouble m_y)

- Called when left click happens
- m_x and m_y are the coordinates of clicked point in world coordinate system
- To-do: store points data for triangles
  - Store the points in *triangle_to_draw*
  - Once 3 new points are picked out, push this new triangle into *triangles.*

Your Task

DrawTriangles ()

- Called when the scene is rendered
- To-do: Render triangles using OpenGL
  - Extract data of triangles from *triangles*
  - Draw each triangle with different colors in *color_array.*
    - **glColor3ubv(color_array[i]);**

Your Task

Function: AffineMatricesCalculation()

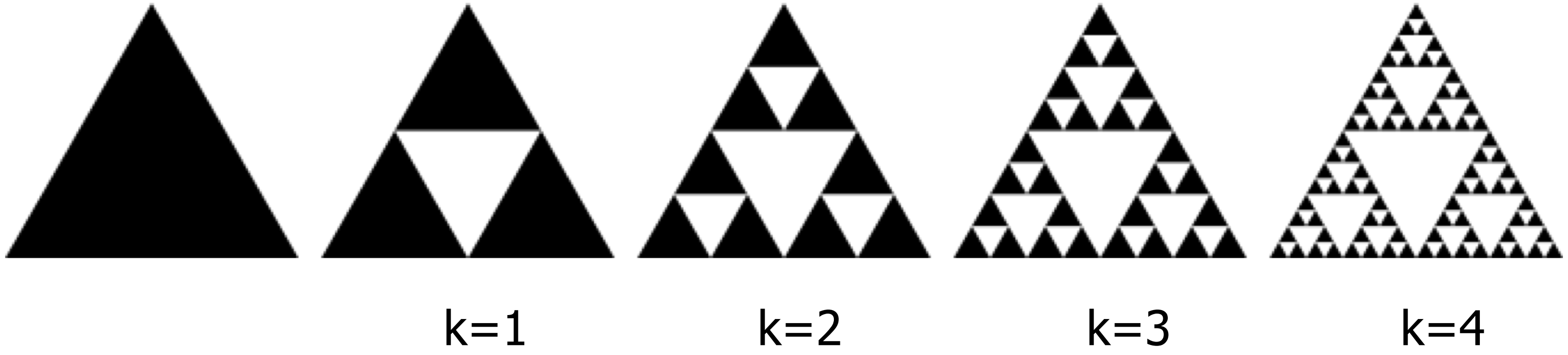
- Called after triangles are specified and before fractals are drawn
- To-do: Compute the affine transformation matrices
  - Compare the first triangle with subsequent ones to determine the affine transformations
  - Store each affine transformation matrix into ***affine_matrices***

Your Task

Function : RecursiveFractal()

void RecursiveFractal(int k)

- Called to draw the fractal
- K is the depth of recursion



Your Task

Pseudo code for RecursiveFractal()

```
Procedure: RecursiveFractal ( k )
//The parameter k above is the times of recursion.
Begin
    If ( k > 0 ) Then
        For each transformation matrix $M_i$
            Step (1): Push current modelview matrix into stack.
            Step (2): Multiply current matrix with $M_i$.
                // $M_i$ is transformation matrix from $T_0$ to $T_i$.
            Step (3): RecursiveFractal (k-1) to draw a fractal in $T_{k-1,i}$.
            Step (4): Pop matrix from matrix stack.
    Else
        Draw triangle T($P_1$, $P_2$, $P_3$) with current modelview matrix
END
```

Outline

- Review of Homogeneous Coordinates and Transformation Matrix
- 2D Transformation in OpenGL
- Fractal Drawing – Programming Assignment
    - About the Template
    - About the Task
    - Submission

Hand-in

- Submit the finished program source file(code.cpp)
- Ensure that your file can be compiled and run successfully.
- Submit your file through the Web-handin.

- Late Policy
  - 50% off for the delay of each working day.
  - Re-submission after deadline is treated as late submission.

- NO PLAGIARISM!

Deadline: 11:59pm, Oct 17, 2018 HKT

Some Useful Links

- OpenGL Official Site
  http://www.opengl.org
- OpenGL reference can be found in MSDN
  http://msdn.microsoft.com/library/default.asp?url=/library/en-us/opengl/apxb4_82lh.asp
  or inside VS.NET. (search for “opengl reference”)
- GLU reference
  http://pyopengl.sourceforge.net/documentation/manual/reference-GLU.html

Thank you!