Universidad Nacional de San Antonio Abad del Cusco
Departamento Académico de Informática
COMPUTACIÓN GRAFICA I
Práctica Nº 6

TRANSFORMACION 2D
ESCALAMIENTO Y REFLEXIÓN

1. **OBJETIVO.**
   - Entender los conceptos geométricos de escalamiento y reflexión.
   - Implementar y aplicar transformaciones de escalamiento y reflexión.
2. **BASE TEORICA**

2.1. Escalamiento General en 2D.

Cualquier operación de escalado debe mantener un punto fijo en su posición original mientras se modifica el resto de los puntos bajo consideración. En el caso general, el punto fijo puede ser cualquier punto arbitrario en el espacio 2D. En este caso, debemos trasladar el objeto bajo escala para que el punto fijo coincida con el origen antes de aplicar las ecuaciones de escalamiento.

Luego, la traslación se realiza nuevamente para revertir el efecto de la primera traslación. Esta secuencia de operaciones es similar a la que se implementó en Rotación General.

Por lo tanto, dado un vértice ˙**p1 = [x1, y1]** y un punto fijo ˙**p0 = [x0, y0]**, podemos expresar la secuencia de operaciones como:

$$\underbrace{\begin{bmatrix} x_2 \\ y_2 \end{bmatrix}}_{\dot{\mathbf{p}}_2} = \underbrace{\begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}}_{\dot{S}(s_x, s_y)} \underbrace{\begin{bmatrix} x_1 - x_0 \\ y_1 - y_0 \end{bmatrix}}_{\dot{\mathbf{p}}_1 - \dot{\mathbf{p}}_0} + \underbrace{\begin{bmatrix} x_0 \\ y_0 \end{bmatrix}}_{\dot{\mathbf{p}}_0}$$

donde ˙**S** es la matriz de escala; ***sx*** y ***sy*** son los factores de escala; **[−x0, −y0]** y **[x0, y0]** son los vectores de traslación; ˙**p0** es el punto fijo; ˙**p1** es el punto anterior al escalamiento; y ˙**p2** es el punto después de escalar.

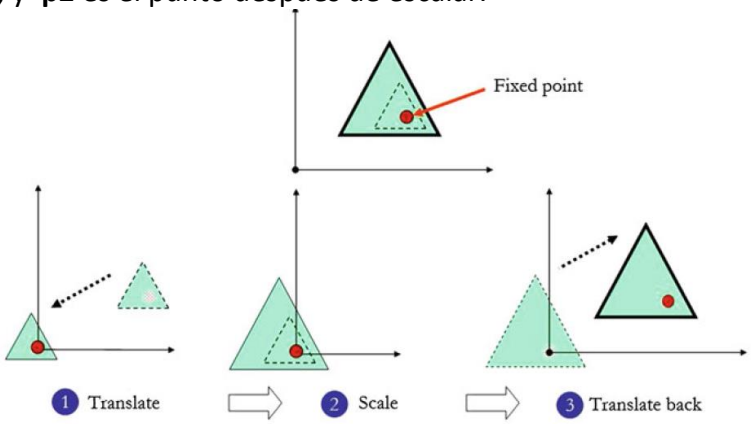

Fig. 1. Escalamiento general en 2D.

En coordenadas homogéneas, la operación de escala se aplica como:

$$\underbrace{\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}}_{\mathbf{p}_2} = \underbrace{\begin{bmatrix} 1 & 0 & x_0 \\ 0 & 1 & y_0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathrm{T}([x_0, y_0]^T)} \underbrace{\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathrm{S}(s_x, s_y)} \underbrace{\begin{bmatrix} 1 & 0 & -x_0 \\ 0 & 1 & -y_0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathrm{T}([-x_0, -y_0]^T)} \underbrace{\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}}_{\mathbf{p}_1}$$

$$= \underbrace{\begin{bmatrix} s_x & 0 & -x_0 s_x + x_0 \\ 0 & s_y & -y_0 s_y + y_0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathrm{T}([x_0, y_0]^T)\mathrm{S}(s_x, s_y)\mathrm{T}([-x_0, -y_0]^T)} \underbrace{\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}}_{\mathbf{p}_1},$$

donde **S** es la matriz de escala; **Sx** y **Sy** son los factores de escala; **T** son las matrices de traslación, **[–x0, –y0]** y **[x0, y0]** son los vectores de traslación; **p0** es el punto fijo; **p1** es el punto antes de escalar; y **p2** es el punto después de escalar.

2.2. Reflexión con respecto al eje X.

Una operación de reflexión 2D sobre el eje x es equivalente a realizar una operación de escalado no uniforme utilizando factores de 1 y -1 a lo largo del eje x y el eje y respectivamente (es decir, los valores de x permanecen sin cambios mientras se invierten los valores de y). Por lo tanto, para reflejar un punto ˙**p1 = [x1, y1]** sobre el eje x para obtener ˙**p2 = [x2, y2]**, podemos usar**:**

$$x_2 = x_1,$$
$$y_2 = -y_1.$$

En forma de matriz podemos escribir:

$$\underbrace{\begin{bmatrix} x_2 \\ y_2 \end{bmatrix}}_{\dot{\mathbf{p}}_2} = \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}}_{\dot{\mathrm{Ref}}_x} \underbrace{\begin{bmatrix} x_1 \\ y_1 \end{bmatrix}}_{\dot{\mathbf{p}}_1}$$

donde ˙ **Refx** es una matriz 2 × 2 que representa la operación de reflexión sobre el eje **x** para puntos no homogéneos en el espacio 2D.

En coordenadas homogéneas, para reflejar un punto 2D **p1 = [x1, y1, 1]** sobre el eje **x** para obtener **p2 = [x2, y2, 1]**, usamos:

$$\underbrace{\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}}_{\mathbf{p}_2} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathrm{Ref}_x} \underbrace{\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}}_{\mathbf{p}_1}$$

donde **Refx** es una matriz de 3 × 3 que representa la operación de reflexión sobre el eje **x** para puntos homogéneos en espacio 2D.

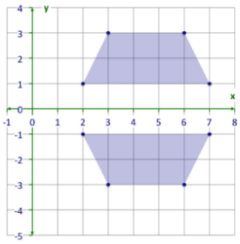

Fig. 2 Reflexión con respecto al eje X.

2.3. Reflexión con respecto al eje Y.

Una operación de reflexión 2D sobre el eje **y** es equivalente a realizar una operación de escalamiento no uniforme utilizando factores de −1 y 1 a lo largo del eje **x** y el eje **y** respectivamente (es decir, los valores de **y** permanecen sin cambios mientras se invierten los valores de **x**). Por lo tanto, para reflejar un punto ˙**p1 = [x1, y1]** sobre el eje **y** para obtener ˙**p2 = [x2, y2]**, podemos usar:

$$x_2 = -x_1$$
$$y_2 = y_1.$$

Alternativamente podemos escribir:

$$\underbrace{\begin{bmatrix} x_2 \\ y_2 \end{bmatrix}}_{\dot{\mathbf{p}}_2} = \underbrace{\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}}_{\dot{\mathrm{Ref}}_y} \underbrace{\begin{bmatrix} x_1 \\ y_1 \end{bmatrix}}_{\dot{\mathbf{p}}_1}$$

donde ˙ **Refy** es una matriz 2 × 2 que representa la operación de reflexión sobre el eje **y** para puntos no homogéneos en el espacio 2D.

En coordenadas homogéneas, para reflejar un punto 2D **p1 = [x1, y1, 1]** sobre el eje **y** para obtener **p2 = [x2, y2, 1]**, usamos:

$$\underbrace{\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}}_{\mathbf{p}_2} = \underbrace{\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathrm{Ref}_y} \underbrace{\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}}_{\mathbf{p}_1}$$

donde **Refy** es una matriz 3 × 3 que representa la operación de reflexión sobre el eje **y** para puntos homogéneos en espacio 2D

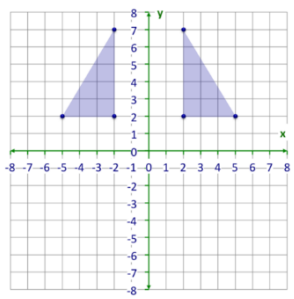

Fig. 3. Reflexión con respecto al eje Y

3. DESARROLLO DE LA PRÁCTICA.

3.1. Escalamiento de un triángulo.

```
#include<math.h>
#include<cstdlib>
#include<GL/glew.h>
#include<GL/glut.h>
#include<stdio.h>
using namespace std;
double ptoEscaladoX(double x1, double x0, double Sx)
{
	return x0 + (x1 - x0)*Sx;
}
double ptoEscaladoY(double y1, double y0, double Sy)
{
	return y0 + (y1 - y0)*Sy;
}
```

```
//--escalamiento de triangulo
void escalaTriangulo(double x1, double y1, double x2, double y2, double x3,
double y3, double Sx, double Sy)
{
	int x1e, y1e, x2e, y2e;
	//--dibujar el triángulo original
	glBegin(GL_LINE_LOOP);
	 glVertex2f(x1, y1);
	 glVertex2f(x2, y2);
	 glVertex2f(x3, y3);
	glEnd();

	//--escalar cada punto manteniendo fijo el punto [x3,y3]
	x1e = ptoEscaladoX(x1, x3, Sx);
	y1e = ptoEscaladoY(y1, y3, Sy);

	x2e = ptoEscaladoX(x2, x3, Sx);
	y2e = ptoEscaladoY(y2, y3, Sy);

	//--dibujar el triángulo escalado
	glBegin(GL_LINE_LOOP);
	 glVertex2f(x1e, y1e);
	 glVertex2f(x2e, y2e);
	 glVertex2f(x3, y3);
	glEnd();
}
//--despliega el gráfico
void display()
{
	double x1, y1, x2, y2, x3, y3, Sx, Sy;
	x1 = 120.0;
	y1 = 160.0;
	x2 = 150.0;
	y2 = 300.0;
	x3 = 60.0;
	y3 = 100.0;
	Sx = 1.5;
	Sy = 1.5;

	glClear(GL_COLOR_BUFFER_BIT);  //--establece el color de la ventana
	escalaTriangulo(x1, y1, x2, y2, x3, y3, Sx, Sy);
	glFlush(); //--fuerza la ejecución de los comandos de OpenGL
}

void myinit()
{
	glClearColor(1.0, 1.0, 1.0, 1.0); //--Establece el color de fondo.
	glColor3f(1.0, 0.0, 0.0);
	glPointSize(1.0); //--tamño de los puntos
	glMatrixMode(GL_PROJECTION); //--activar la matriz de transformación
	glLoadIdentity(); //--cargar la identidad
	gluOrtho2D(0.0, 499.0, 0.0, 499.0);//establece una ventana de dibujo 2D
}

void main(int argc, char** argv)
{
	/*  Inicializacion GLUT estándar*/
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(500, 500); /* ventana 500x500 pixeles */
	glutInitWindowPosition(0, 0);
	glutCreateWindow("Escalamiento"); /* título de la ventana*/
	glutDisplayFunc(display);
	myinit(); /* fija o establece los atributos */
	glutMainLoop(); /* entra a un ciclo de evento */
```

```
}
```

3.2. Reflexión con respecto al eje X.

```
#include<math.h>
#include<cstdlib>
#include<GL/glew.h>
#include<GL/glut.h>
#include<stdio.h>
using namespace std;
//--refleja un triangulo con respecto al eje X
void reflejaTrianguloX(double x1, double y1, double x2, double y2, double x3,
double y3)
{
	double x1r, y1r, x2r, y2r, x3r, y3r;
	//--dibuja ejes X , Y
	glBegin(GL_LINES);
	glVertex2f(0, 250);
	glVertex2f(500, 250);
	glVertex2f(50, 0);
	glVertex2f(50, 500);
	glEnd();
	//-- convierte puntos a nueva escala
	x1 = x1 + 50;
	y1 = y1 + 250;
	x2 = x2 + 50;
	y2 = y2 + 250;
	x3 = x3 + 50;
	y3 = y3 + 250;


	//--dibujar el triangulo original
	glBegin(GL_LINE_LOOP);
	glVertex2f(x1, y1);
	glVertex2f(x2, y2);
	glVertex2f(x3, y3);
	glEnd();
	//--refleja triangulo con respecto al eje X

	x1r = x1;
	y1r = -y1+500;
	x2r = x2;
	y2r = -y2+500;
	x3r = x3;
	y3r = -y3+500;

	//--dibuja triangulo reflejado
	glBegin(GL_LINE_LOOP);
	glVertex2f(x1r, y1r);
	glVertex2f(x2r, y2r);
	glVertex2f(x3r, y3r);
	glEnd();

}
//--despliega el gráfico
void display()
{
	double x1, y1, x2, y2, x3, y3;
	x1 = 120.0;
	y1 = 160.0;
	x2 = 100.0;
	y2 = 200.0;
	x3 = 10.0;
	y3 = 100.0;

	glClear(GL_COLOR_BUFFER_BIT);  //--establece el color de la ventana
```

```
        reflejaTrianguloY(x1, y1, x2, y2, x3, y3);
        glFlush(); //--fuerza la ejecución de los comandos de OpenGL
}

void myinit()
{
        glClearColor(1.0, 1.0, 1.0, 1.0);
        glColor3f(1.0, 0.0, 0.0);
        glPointSize(1.0); //--tamño de los puntos
        glMatrixMode(GL_PROJECTION); //--activar la matriz de transformación
        glLoadIdentity();
        gluOrtho2D(0.0, 499.0, 0.0, 499.0); //--establece una ventana de dibujo
2D
}

void main(int argc, char** argv)
{
        /*  Inicializacion GLUT estándar*/
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
        glutInitWindowSize(500, 500); /* ventana 500x500 pixeles */
        glutInitWindowPosition(0, 0);
        glutCreateWindow("Reflexión"); /* título de la ventana*/
        glutDisplayFunc(display);
        myinit(); /* fija o establece los atributos */
        glutMainLoop(); /* entra a un ciclo de evento */
}
```

3.3. Reflexión con respecto al eje Y.

```
 //--refleja triangulo con respecto al eje Y
void reflejaTrianguloY(double x1, double y1, double x2, double y2, double x3,
double y3)
{
        double x1r, y1r, x2r, y2r, x3r, y3r;
        //--dibuja ejes X , Y
        glBegin(GL_LINES);
        glVertex2f(250, 0);
        glVertex2f(250, 500);
        glVertex2f(0, 50);
        glVertex2f(500, 50);
        glEnd();
        //-- convierte puntos a nueva escala
        x1 = x1 + 250;
        y1 = y1 + 50;
        x2 = x2 + 250;
        y2 = y2 + 50;
        x3 = x3 + 250;
        y3 = y3 + 50;
        //--dibujar el triangulo original
        glBegin(GL_LINE_LOOP);
        glVertex2f(x1, y1);
        glVertex2f(x2, y2);
        glVertex2f(x3, y3);
        glEnd();
        //--refleja triangulo con respecto al eje X

        x1r = -x1+500;
        y1r = y1;
        x2r = -x2+500;
        y2r = y2;
        x3r = -x3+500;
        y3r = y3;

        //--dibuja triangulo reflejado
        glBegin(GL_LINE_LOOP);
        glVertex2f(x1r, y1r);
```

```
        glVertex2f(x2r, y2r);
        glVertex2f(x3r, y3r);
        glEnd();

    }
```

4. ACTIVIDADES EN CLASE

4.1. Escribir un programa que escale un círculo dos veces su dimensión original manteniendo fijo su centro.

4.2. Escribir un programa que refleje un polígono de 5 lados con respecto al eje X.

5. TAREA.

5.1. Escribir un programa para realizar reflexión general en 2D de un triángulo.

1. BIBLIOGRAFIA

- ✓ Donald Hearn, M. Pauline Baker (2006); Gráficas por Computadora con OpenGL; Pearson Prentice Hall; 3° edición; Madrid.
- ✓ Rimon Elias (2019); Digital Media A problema-solving Aproach for Computer Graphics; eBook. New York
- ✓ Sumanta Guha (2019); Computer Graphics Through OpenGL; Third Edition; Taylor & Francis Group.
  NW.
- ✓ Gordon V.Scott, Clevenger Jhon (2019). Computer Graphics Programming in OpenGl with C++. Mercury Learning and Information.