Benny A. Ruiz Jiménez.

A01328177.

Tarea 1, Segundo Parcial.

Piramide con Rotación.

Dibujo de la Pirámide:

Base de la Pirámide.

```
glBegin(GL_POLYGON);
        glColor3f(1,1,1);
        glVertex3d(0,0,0);
        glVertex3d(1,0,0);
        glVertex3d(1,0,1);
        glVertex3d(0,0,1);
        glEnd();
```

Usamos GL_Triangle para hacer un Triángulo relleno que

Parte trasera de la Pirámide.

```
//Parte trasera
        glColor3f(0.2, 0.658, 0.101);
        glVertex3d(0,1,0);
        glVertex3d(1,-1,-1);
        glVertex3d(-1,-1,-1);
```

Lado Izquierdo de la Pirámide.

```
//ParteIzquierda
        glColor3f(0.658, 0.101, 0.145);
        glVertex3d(0,1,0);
        glVertex3d(-1,-1,-1);
        glVertex3d(-1,-1,1);
```

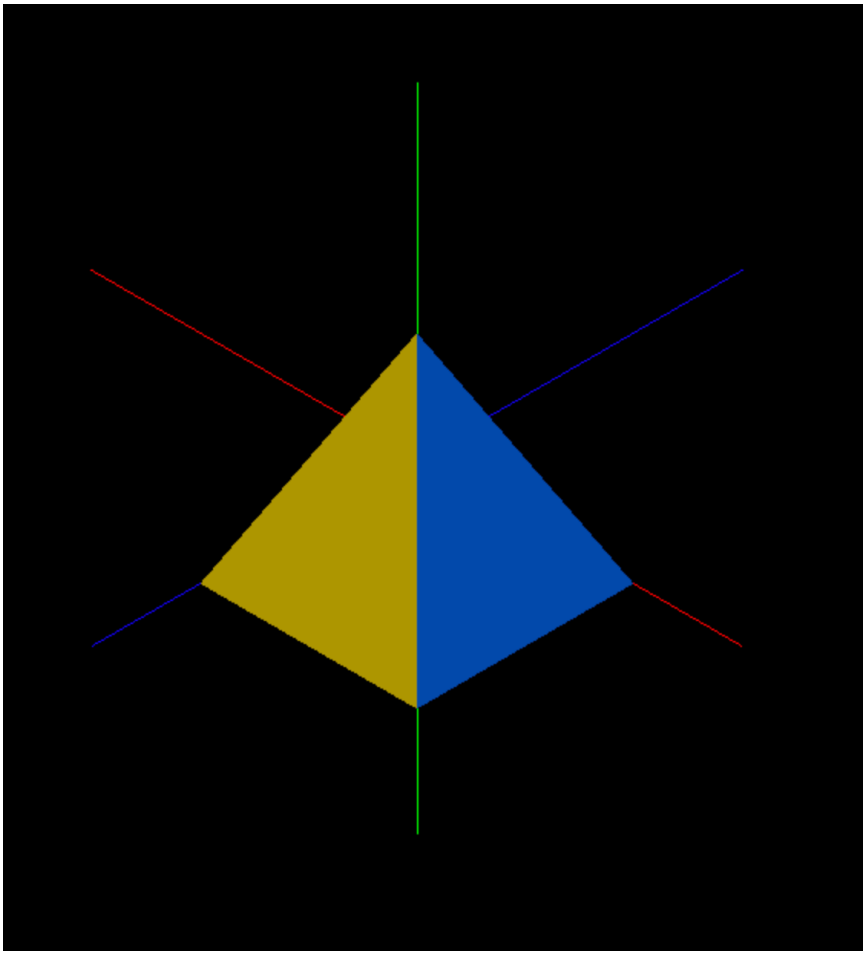
Lado frontal de la pirámide.

```
//Parte delantera.
        glColor3f(.682,.6,.141);
        glVertex3d(0,1,0);
        glVertex3d(-1,-1,1);
        glVertex3d(1,-1,1);
```

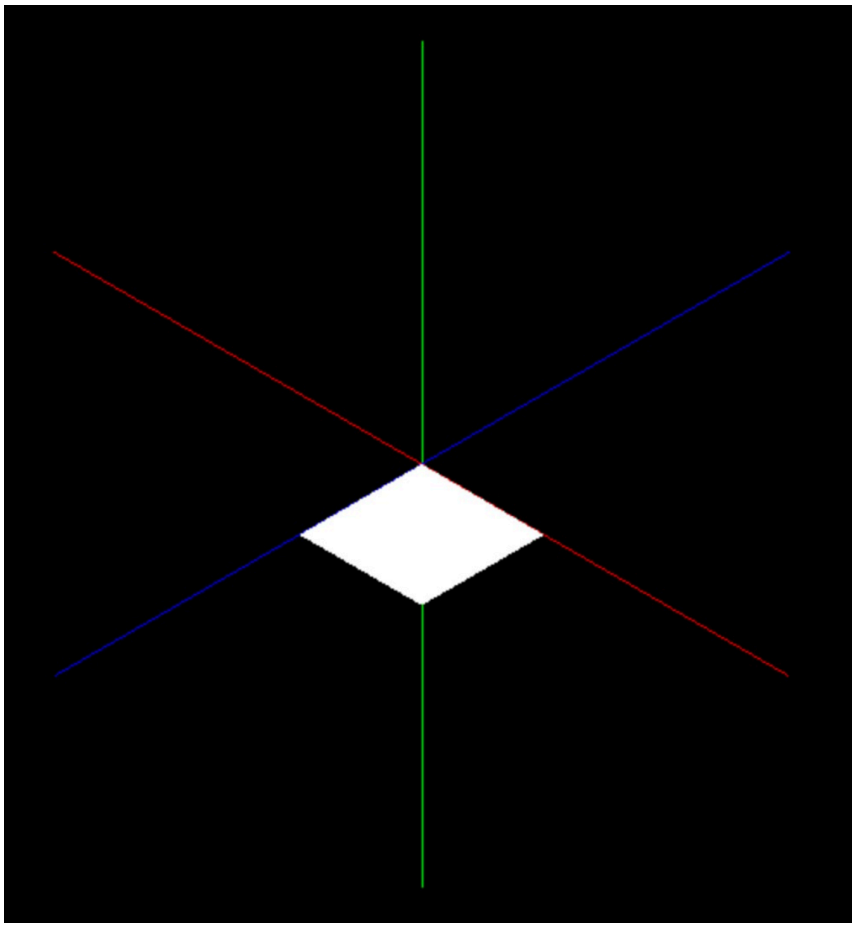
Lado derecho de la pirámide.

```
//Parte Derecha
        glColor3f(0.101, 0.301, 0.658);
        glVertex3d(0,1,0);
        glVertex3d(1,-1,1);
        glVertex3d(1,-1,-1);
        glEnd();
```

Después de esto, podemos observar la pirámide ya dibujada.



Aunque como observación, el cuadro ‘base’ a pesar de tener las mismas medidas que el resto de la pirámide, este se ve mucho más chico que las paredes de esta, por lo que al rotar se puede ver vacía esta parte.



Debido a que por enfermedad me ausenté de 2 clases, ya no pude concluir la tarea hasta donde se tenía proyectado, por lo que subo de esta forma la tarea para mostrar lo que se ha avanzado hasta ahora.

Segunda Parte.

Después de ponerme al corriente con el tema de las matrices, cambié la estructura de mi código.

Primero, declaramos nuestra pirámide en un arreglo de [4][5], cómo usted estableció en clases agregamos una fila de 1s para cumplir la condición de matriz cuadrada. Esta vez dibujé en centro la pirámide.

```
//Declaramoz la Matriz Piramide, que contiene los 5 vertices de la piramide.
float piramide[4][5] =
{
    //x1     x2      x3       x4       x5
    {-1,     1,      1,      -1,      0},
    //y1     y2      y3       y4       y5
    { 0,     0,      0,       0,      2},
    //z1     z2      z3       z4       z5
    {-1,    -1,      1,       1,      0},

    { 1,     1,      1,       1,      1}, //cumpla la condicion de Matriz Cuadrada.
};
```

Posteriormente, declaré nuestras 3 matrices de rotación para su eje correspondient

```
float rotaX[4][4] =   //Matriz Identidad de Rotacion en X.
{
    {1, 0,  0,  0},
    {0, 1,  0,  0},
    {0, 0,  1,  0},
    {0, 0,  0,  1},
};

float rotaY[4][4] =   //Matriz Identidad de Rotacion en Y.
{
    {1, 0,  0,  0},
    {0, 1,  0,  0},
    {0, 0,  1,  0},
    {0, 0,  0,  1},
};
float rotaZ[4][4] =   //Matriz Identidad de Rotacion en Z.
{
    {1, 0,  0,  0},
    {0, 1,  0,  0},
    {0, 0,  1,  0},
    {0, 0,  0,  1},
};
```

Operación de multiplicación.

Implementación del For para limpiar las matrices de Almacenamiento.

```
for (int i = 0; i<4; i++) //Limpia la matriz en donde se guardan los resultadoss
{
    for(int j = 0; j<5; j++)
    {
        mA[i][j]=0;
        mB[i][j]=0;
        mC[i][j]=0;
    }
}
```

Las matrices de almacenamiento las declaré previamente de esta forma.

```
float mA[4][5] =   //Matriz DONDE se almacenan los resultados de la multiplicacion de matrices.
{
    {0, 0,  0,  0,  0},
    {0, 0,  0,  0,  0},
    {0, 0,  0,  0,  0},
    {0, 0,  0,  0,  0},
};
float mB[4][5] =   //Matriz DONDE se almacenan los resultados de la multiplicacion de matrices.
{
    {0, 0,  0,  0,  0},
    {0, 0,  0,  0,  0},
    {0, 0,  0,  0,  0},
    {0, 0,  0,  0,  0},
};
float mC[4][5] =   //Matriz DONDE se almacenan los resultados de la multiplicacion de matrices.
{
    {0, 0,  0,  0,  0},
    {0, 0,  0,  0,  0},
    {0, 0,  0,  0,  0},
    {0, 0,  0,  0,  0},
};
```

Matriz de Multiplicación EJE X. Almacenamos el resultado en mC.

```
for(int x = 0; x < 4; x++){
    //Recorre las filas
    for(int y = 0; y<5; y++){
        //recorre las columnas de la matriz ROTACION para ir haciendo la sumatoria
        for(int z = 0; z<4; z++){
            mC[x][y] += rotaX[x][z] * piramide[z][y];
        }
    }
}
```

Matriz de Multiplicación EJE Y. Almacenamos el resultado en la matriz auxiliar mB.

```
for(int x = 0; x < 4; x++){
    //Recorre las filas
    for(int y = 0; y<5; y++){

        //recorre las columnas de la matriz ROTACION para ir haciendo la sumatoria
        for(int z = 0; z<4; z++){
            mB[x][y] += rotaY[x][z] * mC[z][y];
        }
    }
}
```

Matriz de Multiplicación EJE Z. Almacenamos el resultado en la matriz auxiliar mA.

```
for(int x = 0; x < 4; x++){
    //Recorre las filas
    for(int y = 0; y<5; y++){

        //recorre las columnas de la matriz ROTACION para ir haciendo la sumatoria
        for(int z = 0; z<4; z++){
            mA[x][y] += rotaZ[x][z] * mB[z][y];
        }
    }
}
```

Por último guardamos el resultado obtenido de esta operación en nuestra matriz ‘pirámide’

Si queremos observar solamente la rotación en el eje X, colocamos el código de esta forma.

```
//Guarda los resultados en piramide;
for(int i = 0; i<4; i++)
{
    for(int j = 0; j<5;j++)
    {
        piramide[i][j] = mC[i][j];
    }
}
```

Si queremos observar solamente la rotación en los ejes X, Y, colocamos el código de esta forma.

```
for(int i = 0; i<4; i++)
{
    for(int j = 0; j<5;j++)
    {
        piramide[i][j] = mB[i][j];
    }
}
```

Si queremos observar la rotación completa en el eje X, Y, Z, colocamos el código de esta forma.

```
//Guarda los resultados en piramide;
for(int i = 0; i<4; i++)
{
    for(int j = 0; j<5;j++)
    {
        piramide[i][j] = mA[i][j];
    }
}
```

Cambio en el dibujado de la pirámide.

```
glBegin(GL_POLYGON);
glColor3f(.6,.6,.6);//color del poligono
glVertex3d(piramide[0][0],piramide[1][0],piramide[2][0]); //punto1.
glVertex3d(piramide[0][1],piramide[1][1],piramide[2][1]); //punto2.
glVertex3d(piramide[0][2],piramide[1][2],piramide[2][2]); //punto3
glVertex3d(piramide[0][3],piramide[1][3],piramide[2][3]); //punto4
glEnd();


//Usamos GLTriangles y le damos los vertices a trazar (esta vez en 3 dimensiones)
glBegin(GL_TRIANGLES);

//Parte trasera
glColor3f(0.2, 0.658, 0.101);
glVertex3d(piramide[0][4],piramide[1][4],piramide[2][4]);
glVertex3d(piramide[0][0],piramide[1][0],piramide[2][0]);
glVertex3d(piramide[0][1],piramide[1][1],piramide[2][1]);

//ParteIzquierda
glColor3f(0.658, 0.101, 0.145);
glVertex3d(piramide[0][4],piramide[1][4],piramide[2][4]);
glVertex3d(piramide[0][1],piramide[1][1],piramide[2][1]);
glVertex3d(piramide[0][2],piramide[1][2],piramide[2][2]);

//Parte delantera.
glColor3f(.682,.6,.141);
glVertex3d(piramide[0][4],piramide[1][4],piramide[2][4]);
glVertex3d(piramide[0][2],piramide[1][2],piramide[2][2]);
glVertex3d(piramide[0][3],piramide[1][3],piramide[2][3]);

//Parte Derecha
glColor3f(0.101, 0.301, 0.658);
glVertex3d(piramide[0][4],piramide[1][4],piramide[2][4]);
glVertex3d(piramide[0][3],piramide[1][3],piramide[2][3]);
glVertex3d(piramide[0][4],piramide[1][4],piramide[2][4]);
glEnd();
```

Se puede observar que la estructura no cambia, solamente se llaman los puntos desde el arreglo.

Puntos Finales:

Declaramos nuestro ángulo, transformamos esta declaración.

Damos valor a nuestras variables senos y coseno.

Le damos nuevos valores a nuestras matrices rotaX, rotaY, rotaZ.

Efectuamos la multiplicación y obtendremos nuestra rotación en X,Y y Z.

usleep() es para que se pueda apreciar la rotación más suave, pero para eso necesitamos importar la librería <unistd.h>

Matrices de Rotación 3D sobre los ejes

$$R_z = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_y = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

```
angulo = 1;
angulo = (angulo/180)*M_PI;

seno = sin(angulo);
coseno = cos(angulo);
rotaX[1][1] = coseno;
rotaX[1][2] = -1*seno;
rotaX[2][1] = seno;
rotaX[2][2] = coseno;

rotaY[0][0] = coseno;
rotaY[0][2] = seno;
rotaY[2][0] = -1*seno;
rotaY[2][2] = coseno;

rotaZ[0][0] = coseno;
rotaZ[0][1] = -1*seno;
rotaZ[1][0] = seno;
rotaZ[1][1] = coseno;

multiplicacion();
usleep(5000);
```

Conclusiones Finales.

Hay un sinfín de operaciones matriciales en animaciones que hoy en día nos pudieran parecer triviales, pero antes de efectuarlas en necesario conocer pertinentemente las operaciones con matrices y su estructura; para poder realizar estas de la forma correcta, fue lo que me retrasó al inicio, al faltar a clases no comprendía correctamente la estructura de estas pero una vez conociendo la estructura y funcionamiento pude lograr la animación pedida.