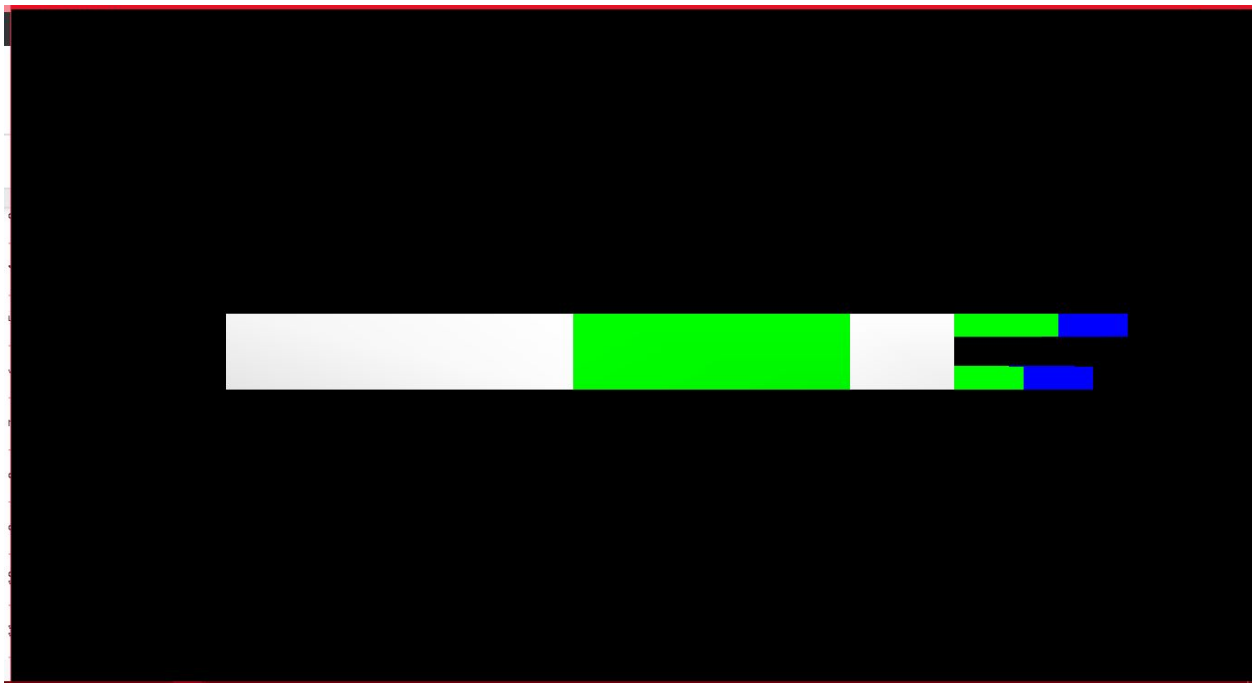
Padilla Herrera Carlos Ignacio

Reporte Practica 5.

1. Se evaluará con el archivo de código indicado durante la sesión**.**



2. Indique los problemas que se enfrentó al construir el modelo indicado.

Me fue difícil evaluar el código y visualizar la posición de los dedos. En un principio se me complicó ver la rotación de la mano con respecto al codo. Me fue difícil de igual manera visualizar como quedaría el código de antemano. La gran mayoría de los problemas que tuve con los giros y rotaciones fueron por que entendí mal en un principio la función del push y pop matrix.

Conclusiones

Con el desarrollo de la práctica pude entender de mejor manera los conceptos de teoría y la importancia de poner atención al detalle. La práctica me pareció ilustrativa para el tema y adquirí algo de paciencia para la escritura de código. Además de poder visualizar ligeramente mejor las rotaciones,además del funcionamiento de push y pop y su utilidad a la hora de aplicar transformaciones.

```
//Semestre 2020 - 1
//************************************************************//
//************************************************************//
//************** Alumno (s): *********************************//
//*************    Padilla Herrera Carlos Ignacio                    ******//
//*************    10 de septiembre del 2019 a las 9:00 am hasta 11:00 am    ******//
//************************************************************//
#include "Main.h"

float transZ = -5.0f;
float transX = 0.0f;
float angleX = 0.0f;
float angleY = 0.0f;
int screenW = 0.0;
int screenH = 0.0;

float red[3] = { 1.0, 0.0, 0.0 };
float green[3] = { 0.0,1.0,0.0 };
float blue[3] = { 0.0,0.0,1.0 };
float white[3] = { 1.0,1.0,1.0 };

float rot_hombro = 0.0f;
float rot_codo = 0.0f;
float rot_muneca = 0.0f;
float rot_dedo1 = 0.0f;
float rot_dedo2 = 0.0f;
float rot_dedo3 = 0.0f;
float rot_dedo4 = 0.0f;



GLfloat Position[] = { 0.0f, 3.0f, 0.0f, 1.0f };                    // Light Position
GLfloat Position2[] = { 0.0f, 0.0f, -5.0f, 1.0f };                  // Light Position

void InitGL(void)     // Inicializamos parametros
{

        glShadeModel(GL_SMOOTH);                                                        //
Habilitamos Smooth Shading
        glClearColor(0.0f, 0.0f, 0.0f, 0.0f);                               // Negro de fondo
        glClearDepth(1.0f);                                                             //
Configuramos Depth Buffer
```

```
	glEnable(GL_DEPTH_TEST);                                                    //
Habilitamos Depth Testing

	//Configuracion luz
	glLightfv(GL_LIGHT0, GL_POSITION, Position);
	glLightfv(GL_LIGHT0, GL_SPOT_DIRECTION, Position2);
	glEnable(GL_LIGHTING);
	glEnable(GL_LIGHT0);


	glDepthFunc(GL_LEQUAL);                                                     // Tipo
de Depth Testing a realizar
	glEnable(GL_COLOR_MATERIAL);
	glHint(GL_PERSPECTIVE_CORRECTION_HINT, GL_NICEST);
}

void prisma(float color[3])
{
	GLfloat vertice[8][3] = {
		{ 0.5 ,-0.5, 0.5 },    //Coordenadas Vértice 0 V0
		{ -0.5 ,-0.5, 0.5 },    //Coordenadas Vértice 1 V1
		{ -0.5 ,-0.5, -0.5 },    //Coordenadas Vértice 2 V2
		{ 0.5 ,-0.5, -0.5 },    //Coordenadas Vértice 3 V3
		{ 0.5 ,0.5, 0.5 },    //Coordenadas Vértice 4 V4
		{ 0.5 ,0.5, -0.5 },    //Coordenadas Vértice 5 V5
		{ -0.5 ,0.5, -0.5 },    //Coordenadas Vértice 6 V6
		{ -0.5 ,0.5, 0.5 },    //Coordenadas Vértice 7 V7
	};

	glColor3fv(color);
	glBegin(GL_POLYGON);       //Front
	glNormal3f(0.0f, 0.0f, 1.0f);
	glVertex3fv(vertice[0]);
	glVertex3fv(vertice[4]);
	glVertex3fv(vertice[7]);
	glVertex3fv(vertice[1]);
	glEnd();

	glBegin(GL_POLYGON);       //Right
	glNormal3f(1.0f, 0.0f, 0.0f);
	glVertex3fv(vertice[0]);
	glVertex3fv(vertice[3]);
	glVertex3fv(vertice[5]);
```

```
        glVertex3fv(vertice[4]);
        glEnd();

        glBegin(GL_POLYGON);       //Back
        glNormal3f(0.0f, 0.0f, -1.0f);
        glVertex3fv(vertice[6]);
        glVertex3fv(vertice[5]);
        glVertex3fv(vertice[3]);
        glVertex3fv(vertice[2]);
        glEnd();

        glBegin(GL_POLYGON);  //Left
        glNormal3f(-1.0f, 0.0f, 0.0f);
        glVertex3fv(vertice[1]);
        glVertex3fv(vertice[7]);
        glVertex3fv(vertice[6]);
        glVertex3fv(vertice[2]);
        glEnd();

        glBegin(GL_POLYGON);  //Bottom
        glNormal3f(0.0f, -1.0f, 0.0f);
        glVertex3fv(vertice[0]);
        glVertex3fv(vertice[1]);
        glVertex3fv(vertice[2]);
        glVertex3fv(vertice[3]);
        glEnd();

        glBegin(GL_POLYGON);  //Top
        glNormal3f(0.0f, 1.0f, 0.0f);
        glVertex3fv(vertice[4]);
        glVertex3fv(vertice[5]);
        glVertex3fv(vertice[6]);
        glVertex3fv(vertice[7]);
        glEnd();
}

void display(void)   // Creamos la funcion donde se dibuja
{
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);  // Limpiamos pantalla
y Buffer
        //glMatrixMode(GL_MODELVIEW);
        glLoadIdentity();
```

```
glTranslatef(transX, 0.0f, transZ);
glRotatef(angleY, 0.0, 1.0, 0.0);
glRotatef(angleX, 1.0, 0.0, 0.0);


//Poner Código Aquí.


glTranslatef(1.25f, 0.0f, 0.0f);
glRotatef(rot_hombro, 0.0f, 0.0f, 1.0f); //practicar sobre qué eje rotan las cosas
glTranslatef(1.25f, 0.0f, 0.0f);
glPushMatrix();
        glScalef(2.5f, 1.0f, 1.0f);
        prisma(white); //objeto A
glPopMatrix();


glTranslatef(1.25f, 0.0f, 0.0f);
glRotatef(rot_codo, 0.0f, 1.0f, 0.0f);
glTranslatef(1.0f, 0.0f, 0.0f);
glPushMatrix();
        glScalef(2.0f, 1.0f, 1.0f);
        prisma(green); //objeto B
glPopMatrix();


glTranslatef(1.0f, 0.0f, 0.0f);
glRotatef(rot_muneca, 0.0f, 0.0f, 1.0f);
glTranslatef(0.375f, 0.0f, 0.0f);
glPushMatrix();
        glScalef(0.75f, 1.0f, 1.0f);
        prisma(white); // Objeto C
glPopMatrix();


glPushMatrix();

        glTranslatef(0.375f, -0.35f, 0.375f);
        glRotatef(rot_dedo1, 0.0f, 1.0f, 0.0f);
        glTranslatef(0.25f, 0.0f, 0.0f);
        glPushMatrix();
                glScalef(0.5f, 0.3f, 0.25f);
                prisma(green); // Objeto F
```

```
        glPopMatrix();


        glTranslatef(0.25f, 0.0f, 0.0f);
        glRotatef(rot_dedo2, 0.0f, 1.0f, 0.0f);
        glTranslatef(0.25f, 0.0f, 0.0f);
        glPushMatrix();
                glScalef(0.5f, 0.3f, 0.25f);
                prisma(blue); //Objeto G
        glPopMatrix();


glPopMatrix();



glPushMatrix();


        glTranslatef(0.375f, 0.35f, 0.375f);
        glRotatef(rot_dedo1, 0.0f, 1.0f, 0.0f);
        glTranslatef(0.375f, 0.0f, 0.0f);
        glPushMatrix();
                glScalef(0.75f, 0.3f, 0.25f);
                prisma(green); // Objeto D
        glPopMatrix();


        glTranslatef(0.375f, 0.0f, 0.0f);
        glRotatef(rot_dedo2, 0.0f, 1.0f, 0.0f);
        glTranslatef(0.25f, 0.0f, 0.0f);
        glPushMatrix();
                glScalef(0.5f, 0.3f, 0.25f);
                prisma(blue);
        prisma(blue); //Objeto E


glPopMatrix();


glPushMatrix();
```

```
        glTranslatef(0.375f, -0.35f, 0.0f);
        glRotatef(rot_dedo1, 0.0f, 1.0f, 0.0f);
        glTranslatef(0.5f, 0.0f, 0.0f);
        glPushMatrix();
                glScalef(1.0f, 0.3f, 0.25f);
                prisma(green); // Objeto H
        glPopMatrix();


        glTranslatef(0.5f, 0.0f, 0.0f);
        glRotatef(rot_dedo2, 0.0f, 1.0f, 0.0f);
        glTranslatef(0.375f, 0.0f, 0.0f);
        glPushMatrix();
                glScalef(0.75f, 0.3f, 0.25f);
                prisma(blue);
        prisma(blue); //Objeto I


glPopMatrix();



glPushMatrix();


        glTranslatef(0.375f, 0.0f, 0.0f);
        glRotatef(rot_dedo1, 0.0f, 1.0f, 0.0f);
        glTranslatef(0.5f, 0.0f, 0.0f);
        glPushMatrix();
                glScalef(1.0f, 0.3f, 0.25f);
                prisma(green); // Objeto J
        glPopMatrix();


        glTranslatef(0.5f, 0.0f, 0.0f);
        glRotatef(rot_dedo2, 0.0f, 1.0f, 0.0f);
        glTranslatef(0.5f, 0.0f, 0.0f);
        glPushMatrix();
                glScalef(1.0f, 0.3f, 0.25f);
                prisma(blue);
        prisma(blue); //Objeto K
```

```
	glPopMatrix();



	glutSwapBuffers();
	// Swap The Buffers
}

void reshape(int width, int height)   // Creamos funcion Reshape
{
	if (height == 0)                                                                //
Prevenir division entre cero
	{
		height = 1;
	}

	glViewport(0, 0, width, height);

	glMatrixMode(GL_PROJECTION);                                                    //
Seleccionamos Projection Matrix
	glLoadIdentity();

	// Tipo de Vista
	glFrustum(-0.1, 0.1, -0.1, 0.1, 0.1, 50.0);

	glMatrixMode(GL_MODELVIEW);                                                     //
Seleccionamos Modelview Matrix
	//glLoadIdentity();
}

void keyboard(unsigned char key, int x, int y)  // Create Keyboard Function 20201
{
	switch (key) {
	case 'w':
	case 'W':
		transZ += 0.2f;
		break;
	case 's':
	case 'S':
		transZ -= 0.2f;
		break;
	case 'a':
	case 'A':
```

```
            transX += 0.2f;
            break;
case 'd':
case 'D':
            transX -= 0.2f;
            break;


case 'r':
            if (90.0f >= rot_hombro)
                        rot_hombro += 1.0f;
            break;
case 'R':
            if (-120.0f <= rot_hombro)
                        rot_hombro -= 1.0f;
            break;
case 't':
            if (rot_codo <= 2.0f)
                        rot_codo += 1.0f;
            break;
case 'T':
            if (rot_codo >= -110.0f)
                        rot_codo -= 1.0f;
case 'y':
            if (rot_muneca <= 90.0f)
                        rot_muneca += 1.0f;
            break;
case 'Y':
            if (rot_muneca >= -110.0f)
                        rot_muneca -= 1.0f;
            break;
case 'u':
            if (rot_dedo1 <= 90.0f)
                        rot_dedo1 += 1.0f;
            break;
case 'U':
            if (rot_dedo1 >= -110.0f)
                        rot_dedo1 -= 1.0f;
            break;
case 'i':
            if (rot_dedo2 <= 90.0f)
                        rot_dedo2 += 1.0f;
            break;
```

```
	case 'l':
		if (rot_dedo2 >= -110.0f)
			rot_dedo2 -= 1.0f;
		break;
	case 27:	// Cuando Esc es presionado...
		exit(0);   // Salimos del programa
		break;
	default:	// Cualquier otra
		break;
	}
	glutPostRedisplay();
}

void arrow_keys(int a_keys, int x, int y)  // Funcion para manejo de teclas especiales (arrow keys)
{
	switch (a_keys) {
	case GLUT_KEY_UP:		// Presionamos tecla ARRIBA...
		angleX += 2.0f;
		break;
	case GLUT_KEY_DOWN:		// Presionamos tecla ABAJO...
		angleX -= 2.0f;
		break;
	case GLUT_KEY_LEFT:
		angleY += 2.0f;
		break;
	case GLUT_KEY_RIGHT:
		angleY -= 2.0f;
		break;
	default:
		break;
	}
	glutPostRedisplay();
}


int main(int argc, char** argv)   // Main Function
{
	glutInit(&argc, argv); // Inicializamos OpenGL
	glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH); // Display Mode (Clores RGB y alpha | Buffer Doble )
	screenW = glutGet(GLUT_SCREEN_WIDTH);
	screenH = glutGet(GLUT_SCREEN_HEIGHT);
```

```
	glutInitWindowSize(screenW, screenH);	// Tamaño de la Ventana
	glutInitWindowPosition(0, 0);  //Posicion de la Ventana
	glutCreateWindow("Practica 5 20192"); // Nombre de la Ventana
	printf("Resolution H: %i \n", screenW);
	printf("Resolution V: %i \n", screenH);
	InitGL();                                               // Parametros iniciales de la
aplicacion
	glutDisplayFunc(display);  //Indicamos a Glut función de dibujo
	glutReshapeFunc(reshape);  //Indicamos a Glut función en caso de cambio de tamano
	glutKeyboardFunc(keyboard);	//Indicamos a Glut función de manejo de teclado
	glutSpecialFunc(arrow_keys);	//Otras
	glutMainLoop();	//

	return 0;
}
```