Universidad Nacional Autónoma de México.

Facultad de Ingeniería.

Computación gráfica.

Proyecto final.

Manual Técnico.

Félix González Zepeda.
José de Jesús Hernández Castro.

Semestre 2020-2

ÍNDICE

INTRODUCCIÓN

El proyecto propuesto es un entorno en 3D, tipo alameda y con temática estilo LEGO. Se contemplan elementos de escenarios del mundo de *Cuphead*, nuestro protagonista y personaje de videojuegos, así como la inclusión de Zero-Two, personaje del anime *Darling in the Franxx*.

Este trabajo pretende implementar los conocimientos adquiridos en el curso teórico de computación gráfica, desde la creación de primitivas con la API **GLUT,** modelos de visualización, modelado jerárquico de figuras, modelos de iluminación locales, texturizado, hasta animaciones sencillas y complejas, logrando así la construcción de un entorno 3D interactivo. El desarrollo del proyecto parte de la construcción y uso de figuras de bloque tipo lego, finalizando en la animación de ciertos elementos, por ejemplo, la simulación del transcurso del día y la noche junto con el cambio de escenario que esta conlleva.

El software que se utilizó contempla **Visual Studio 2019**, **OpenGL** con su conjunto de herramientas de utilidad **GLUT 3.7**,**GIMP 2.10.12** este fue utilizado como software de edición de texturas e imágenes para dar efectos y el formato adecuado para el proyecto, 3ds Max 2020/2021, este último con el fin de utilizar modelos 3D, la API de audio **IrrKlang 1.6.0.**

LIBRERÍAS.

```
"Main.h" // Resumen de carga de librerías básicas para la implementación de OpenGL
        <windows.h> //Funciones exclusivas del sistema windows
        <math.h> //Funciones matemáticas
        <stdio.h> //Entrada/Salida estándar
        <stdlib.h> //Funciones estándar
        <glut.h> //Conjunto de herramientas de utilidad para OpenGL
"texture.h" //Carga de texturas
"figuras.h" //Primitivas
"Camera.h" //Manejo de cámara general
"CModel.h" //Carga de modelos 3D
<irrKlang.h> //Manejo y carga de recursos de audio
```

CÁMARA EN TERCERA PERSONA.

De Camera.h

//Con este método inicializamos los valores generales de la cámara, los vectores posición, vista y la normal arriba.

```
void CCamera::Position_Camera(float pos_x,  float pos_y,  float pos_z, float view_x, float view_y, float view_z, float up_x,   float up_y,
float up_z)
{
        mPos      = tVector3(pos_x,  pos_y,  pos_z ); // set position
        mView     = tVector3(view_x, view_y, view_z); // set view
        mUp               = tVector3(up_x,   up_y,   up_z  ); // set the up vector
}
```

//Método para el movimiento de rotación de la cámara respecto al vector de vista.

```
void CCamera::Rotate_Camera(float speed)
{
        tVector3 vVector = mPos-mView;          // Get the view vector

        mPos.z = (float)(mView.z + sin(speed)*vVector.x + cos(speed)*vVector.z);
        mPos.x = (float)(mView.x + cos(speed)*vVector.x - sin(speed)*vVector.z);
}
```

De main.cpp

//Vinculamos el movimiento del modelo de cualquiera de los personajes, al vector vista de la cámara, garantizando un movimiento conjunto y una posición fija de la cámara.

```
glPushMatrix();
        glTranslatef(objCamera.mView.x, objCamera.mView.y, objCamera.mView.z);
        glRotatef(_02ang, 0.0, 1.0, 0.0);
        Dibuja_personaje();
glPopMatrix();
```

//Dentro de la función arrow_keys(), tanto para el caso del cursor de movimiento izquierdo, como derecho, se implemento el siguiente código:

```
objCamera.Rotate_Camera(-CAMERASPEED);
printf("\nCoordenadas de la camara: %f, %f", objCamera.mPos.x, objCamera.mPos.z);
_02ang += atan(CAMERASPEED) * 180 / IK_PI32;
```

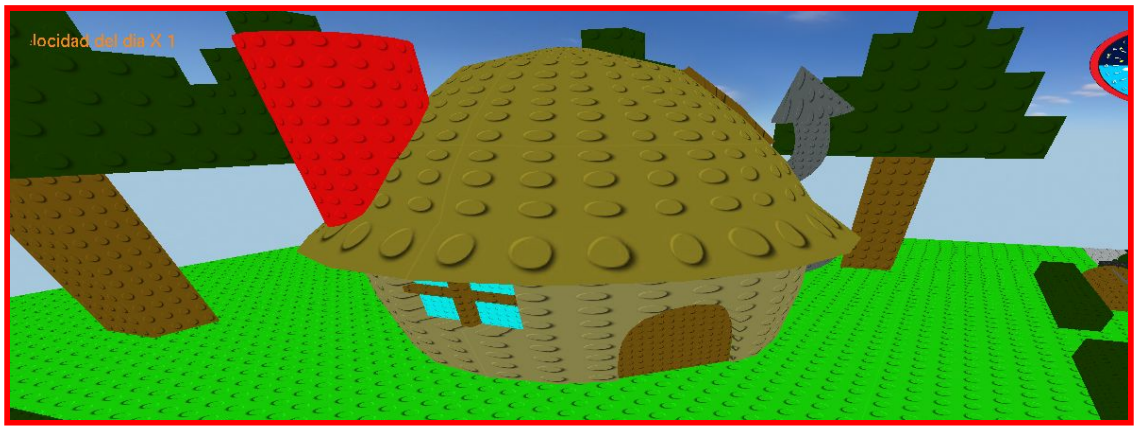
//Con esto garantizamos que la rotación de la cámara sea exactamente la misma que la del personaje, obteniendo el angulo cuya tangente sean los radianes que estamos pasando como parámetro al método Rotate_Camera()

ESCENARIO Y OBJETOS 3D

//Esta función carga los modelos 3D y esto se hace con ayuda de la librería “CModel.h"

```
CModel House; //Se declara el nombre del modelo

void Models_Load() {
//Cargar el modelo
        House._3dsLoad("Models/Casa/Casa.3ds");  //House es el nombre del modelo de la clase CModel
                                                 // _3dsLoad es el método de la clase que carga el modelo
        House.LoadTextureImages();               //Se encarga de cargar las texturas
        House.GLIniTextures();                   //Se encarga de iniciar las texturas en  OpenGL
        House.ReleaseTextureImages();            //Libera las texturas de memoria
}
```



TEXTURIZADO Y MATERIALES

- //Esta función carga las texturas

```
void Texture_Load() {

        t_Moon.LoadTGA("Textures/Objects/t_moon.tga");//Carga la textura para la luna
        t_Moon.BuildGLTexture();                      //Construye la textura en OpenGL
        t_Moon.ReleaseImage();                        //Libera la textura de memoria
```

```
//Carga de texturas para la parte del torso  de Zero_Two
t_02_Back.LoadTGA("Textures/02/t_back.tga"); //Carga la textura delantera del torso
t_02_Back.BuildGLTexture();
t_02_Back.ReleaseImage();

t_02_Front.LoadTGA("Textures/02/t_front.tga");  //Carga la textura trasera del torso
t_02_Front.BuildGLTexture();
t_02_Front.ReleaseImage();

t_02_Sides.LoadTGA("Textures/02/t_sides.tga");  //Carga la textura de los costados del torso
t_02_Sides.BuildGLTexture();
t_02_Sides.ReleaseImage();
```

```
//Texturas de la interfaz en pantalla
clock.LoadTGA("Textures/GUI/day_night_clock.tga");//Carga la textura para el reloj que marca el día y la noche
clock.BuildGLTexture();
clock.ReleaseImage();

manecilla.LoadTGA("Textures/GUI/manecilla.tga");//Carga la textura para la manecilla del Reloj
manecilla.BuildGLTexture();
manecilla.ReleaseImage();
}
```

KIOSCO Y AVATAR JERÁRQUICOS

```
//Esta función dibuja al personaje Zero-Two

void Dibuja_personaje() {

	glPushMatrix();//Dibuja Zero
		glDisable(GL_LIGHTING);
		Personaje1.torso(t_02_Front.GLindex,t_02_Back.GLindex,t_02_Sides.GLindex);
		glEnable(GL_LIGHTING);
		glTranslatef(0.0,-0.62,-0.0);
		glScalef(1.0,0.25,0.5);
		Personaje1.prismaM(_02DiffuseR, _02Specular, _02Shininess);
	glPopMatrix();
	//Piernas
	glPushMatrix();
		glTranslatef(0.08, -1.02, 0.0);
		glRotatef(_02rotLegDer, 1.0, 0.0, 0.0);
		glRotatef(-90, 0.0, 1.0, 0.0);
		glRotatef(-90, 1.0, 0.0, 0.0);
		glScalef(0.8, 1.2, 0.8);
		glScalef(0.35, 0.35, 0.35);
		Personaje1.pierna(_02DiffuseB, _02Specular, _02Shininess, _02DiffuseW, _02Specular, _02Shininess);
	glPopMatrix();
	glPushMatrix();
	/*	glRotatef(rotLeg,1.0,0.0,0.0);*/
		glTranslatef(-0.5, -1.02, 0.0);
		glRotatef(_02rotLegIzq, 1.0, 0.0, 0.0);
		glRotatef(-90, 0.0, 1.0, 0.0);
		glRotatef(-90, 1.0, 0.0, 0.0);
		glScalef(0.8, 1.2, 0.8);
		glScalef(0.35, 0.35, 0.35);
		Personaje1.pierna(_02DiffuseB, _02Specular, _02Shininess, _02DiffuseW, _02Specular, _02Shininess);
	glPopMatrix();
	//Cabeza
	glPushMatrix();
		glTranslatef(0.0, -1.7, 0.0);
		glDisable(GL_LIGHTING);
		glScalef(0.2, 0.2, 0.2);
		glRotatef(180.0, 0.0, 1.0, 0.0);
		Zero.GLrender(NULL, _SHADED, 1.0);
		glEnable(GL_LIGHTING);
	glPopMatrix();
	//Brazos
	glPushMatrix();
```

```
			glTranslatef(0.0, -2.0, -0.1);
			glScalef(0.2, 0.2, 0.2);
			glRotatef(180.0, 0.0, 1.0, 0.0);
			ArmZero.GLrender(NULL, _SHADED, 1.0);
		glPopMatrix();
}

//Esta función dibuja al personaje Cuphead

void Dibuja_personaje2() {
		glPushMatrix();//Dibuja Cuphead
			glDisable(GL_LIGHTING);
			Personaje2.torso(t_01_Front.GLindex, t_01_Back.GLindex, t_01_Sides.GLindex);
			glEnable(GL_LIGHTING);
			glTranslatef(0.0, -0.62, -0.0);
			glScalef(1.0, 0.25, 0.5);
			Personaje2.prismaM(Diffuserojo, Specularcup, Shininesscup);
		glPopMatrix();
		//Piernas
		glPushMatrix();
			glTranslatef(0.08, -1.02, 0.0);
			glRotatef(cuprotpieder, 1.0, 0.0, 0.0);
			glRotatef(-90, 0.0, 1.0, 0.0);
			glRotatef(-90, 1.0, 0.0, 0.0);
			glScalef(0.8, 1.2, 0.8);
			glScalef(0.35, 0.35, 0.35);
			Personaje2.pierna(Diffuseblanco, Specularcup, Shininesscup, Diffusecafe, Specularcup, Shininesscup);
		glPopMatrix();
		glPushMatrix();
		/*	glRotatef(rotLeg,1.0,0.0,0.0);*/
			glTranslatef(-0.5, -1.02, 0.0);
			glRotatef(cuprotpieizq, 1.0, 0.0, 0.0);
			glRotatef(-90, 0.0, 1.0, 0.0);
			glRotatef(-90, 1.0, 0.0, 0.0);
			glScalef(0.8, 1.2, 0.8);
			glScalef(0.35, 0.35, 0.35);
			Personaje2.pierna(Diffuseblanco, Specularcup, Shininesscup, Diffusecafe, Specularcup, Shininesscup);
		glPopMatrix();
		//Cabeza
		glPushMatrix();
			glTranslatef(0.0, -1.7, 0.0);
			glDisable(GL_LIGHTING);
			glScalef(0.2, 0.2, 0.2);
			glRotatef(180.0, 0.0, 1.0, 0.0);
			Cuphead.GLrender(NULL, _SHADED, 1.0);
			glEnable(GL_LIGHTING);
		glPopMatrix();
		//Brazos
		glPushMatrix();
			glTranslatef(0.0, -2.0, -0.1);
			glDisable(GL_LIGHTING);
			glScalef(0.2, 0.2, 0.2);
			glRotatef(180.0, 0.0, 1.0, 0.0);
			ArmCup.GLrender(NULL, _SHADED, 1.0);
			glEnable(GL_LIGHTING);
		glPopMatrix();
}
```

```
//Aquí comienza la construcción del kiosko

void kiosco()
{
	glPushMatrix();//Plataforma principal
		glPushMatrix();
			glPushMatrix();
				Dibuja_plataforma(2,2, _02DiffuseG, _02Specular, _02Shininess);
				for (int i = 0; i < 4;i++) {
					glTranslatef(0.0, y_lego_brick, 0.0);
					Dibuja_plataforma(2,2, _02DiffuseG, _02Specular, _02Shininess);
				}
			glPopMatrix();
			glPushMatrix();
				glTranslatef(z_lego_brick * 3.5,0.0,0.0);
				glPushMatrix();
					glTranslatef(0.0,0.0,x_lego_brick*0.75);
					glRotatef(90,0.0,1.0,0.0);
					Dibuja_torre_2X4(4, _02DiffuseG, _02Specular, _02Shininess);
				glPopMatrix();
				glPushMatrix();
					glTranslatef(0.0, 0.0, -x_lego_brick * 0.25);
					glRotatef(90, 0.0, 1.0, 0.0);
					Dibuja_torre_2X4(4, _02DiffuseG, _02Specular, _02Shininess);
				glPopMatrix();
					glTranslatef(z_lego_brick, 0.0, x_lego_brick*0.25);
					glRotatef(90, 0.0, 1.0, 0.0);
					Dibuja_torre_2X4(4, _02DiffuseG, _02Specular, _02Shininess);
				glPopMatrix();

				glPushMatrix();
					glTranslatef(-z_lego_brick * 3.5, 0.0, 0.0);
					glPushMatrix();
						glTranslatef(0.0, 0.0, x_lego_brick * 0.75);
						glRotatef(90, 0.0, 1.0, 0.0);
						Dibuja_torre_2X4(4, _02DiffuseG, _02Specular, _02Shininess);
					glPopMatrix();
					glPushMatrix();
						glTranslatef(0.0, 0.0, -x_lego_brick * 0.25);
						glRotatef(90, 0.0, 1.0, 0.0);
						Dibuja_torre_2X4(4, _02DiffuseG, _02Specular, _02Shininess);
					glPopMatrix();
						glTranslatef(-z_lego_brick, 0.0, x_lego_brick * 0.25);
						glRotatef(90, 0.0, 1.0, 0.0);
						Dibuja_torre_2X4(4, _02DiffuseG, _02Specular, _02Shininess);
					glPopMatrix();

					glPushMatrix();
```

```
                glTranslatef(0.0,0.0, -z_lego_brick * 3);
                glPushMatrix();
                        glTranslatef(x_lego_brick/2,0.0,0.0);
                        Dibuja_torre_2X4(4, _02DiffuseG, _02Specular, _02Shininess);
                glPopMatrix();
                glPushMatrix();
                        glTranslatef(-x_lego_brick / 2, 0.0, 0.0);
                        Dibuja_torre_2X4(4, _02DiffuseG, _02Specular, _02Shininess);
                glPopMatrix();
                glTranslatef(0.0,0.0,-z_lego_brick);
                Dibuja_torre_2X4(4, _02DiffuseG, _02Specular, _02Shininess);
        glPopMatrix();
glPopMatrix();

//Escaleras
glPushMatrix();
        glTranslatef(0.0,0.0,z_lego_brick*3);
        Dibuja_torre_2X4(4, _02DiffuseG, _02Specular, _02Shininess);
        glPushMatrix();
                glTranslatef(-x_lego_brick, 0.0, 0.0);
                Dibuja_torre_2X4(4, _02DiffuseG, _02Specular, _02Shininess);
        glPopMatrix();
        glPushMatrix();
                glTranslatef(x_lego_brick, 0.0, 0.0);
                Dibuja_torre_2X4(4, _02DiffuseG, _02Specular, _02Shininess);
        glPopMatrix();
        glTranslatef(0.0, 0.0, z_lego_brick);
        glPushMatrix();
                glTranslatef(-x_lego_brick/2, 0.0, 0.0);
                Dibuja_torre_2X4(4, _02DiffuseG, _02Specular, _02Shininess);
        glPopMatrix();
        glPushMatrix();
                glTranslatef(x_lego_brick/2, 0.0, 0.0);
                Dibuja_torre_2X4(4, _02DiffuseG, _02Specular, _02Shininess);
        glPopMatrix();

        glTranslatef(0.0, 0.0, z_lego_brick);
        glPushMatrix();
                glTranslatef(-x_lego_brick/2, 0.0, 0.0);
                Dibuja_torre_2X4(3, _02DiffuseG, _02Specular, _02Shininess);
        glPopMatrix();
        glPushMatrix();
                glTranslatef(x_lego_brick/2, 0.0, 0.0);
                Dibuja_torre_2X4(3, _02DiffuseG, _02Specular, _02Shininess);
        glPopMatrix();

        glTranslatef(0.0, 0.0, z_lego_brick);
        glPushMatrix();
                glTranslatef(-x_lego_brick/2, 0.0, 0.0);
                Dibuja_torre_2X4(2, _02DiffuseG, _02Specular, _02Shininess);
        glPopMatrix();
        glPushMatrix();
                glTranslatef(x_lego_brick/2, 0.0, 0.0);
                Dibuja_torre_2X4(2, _02DiffuseG, _02Specular, _02Shininess);
        glPopMatrix();

        glTranslatef(0.0, 0.0, z_lego_brick);
        glPushMatrix();
                glTranslatef(-x_lego_brick/2, 0.0, 0.0);
                Dibuja_torre_2X4(1, _02DiffuseG, _02Specular, _02Shininess);
        glPopMatrix();
        glPushMatrix();
```

```
                glTranslatef(x_lego_brick/2, 0.0, 0.0);
                Dibuja_torre_2X4(1, _02DiffuseG, _02Specular, _02Shininess);
        glPopMatrix();

        glTranslatef(0.0, 0.0, z_lego_brick);
        glPushMatrix();
                glTranslatef(-x_lego_brick/2, 0.0, 0.0);
                Dibuja_torre_2X4(0, _02DiffuseG, _02Specular, _02Shininess);
        glPopMatrix();
        glPushMatrix();
                glTranslatef(x_lego_brick/2, 0.0, 0.0);
                Dibuja_torre_2X4(0, _02DiffuseG, _02Specular, _02Shininess);
        glPopMatrix();
glPopMatrix();

glPushMatrix();//Barrotes
        glTranslatef(x_lego_brick*1.75,y_lego_brick*5,z_lego_brick*2);
        Dibuja_torre_2X2(10, _02DiffuseGr, _02Specular, _02Shininess);
glPopMatrix();

glPushMatrix();
        glTranslatef(-x_lego_brick * 1.75, y_lego_brick * 5, -z_lego_brick );
        Dibuja_torre_2X2(10, _02DiffuseGr, _02Specular, _02Shininess);
glPopMatrix();

glPushMatrix();
        glTranslatef(x_lego_brick * 1.75, y_lego_brick * 5, -z_lego_brick );
        Dibuja_torre_2X2(10, _02DiffuseGr, _02Specular, _02Shininess);
glPopMatrix();

glPushMatrix();
        glTranslatef(-x_lego_brick * 1.75, y_lego_brick * 5, z_lego_brick * 2);
        Dibuja_torre_2X2(10, _02DiffuseGr, _02Specular, _02Shininess);
glPopMatrix();

glPushMatrix();
        glTranslatef(-x_lego_brick*0.75 , y_lego_brick * 5, -z_lego_brick * 3);
        Dibuja_torre_2X2(10, _02DiffuseGr, _02Specular, _02Shininess);
glPopMatrix();

glPushMatrix();
        glTranslatef(x_lego_brick * 0.75, y_lego_brick * 5, -z_lego_brick * 3);
        Dibuja_torre_2X2(10, _02DiffuseGr, _02Specular, _02Shininess);
glPopMatrix();

glPushMatrix();
        glTranslatef(-x_lego_brick * 0.75, y_lego_brick * 5, z_lego_brick * 4);
        Dibuja_torre_2X2(10, _02DiffuseGr, _02Specular, _02Shininess);
glPopMatrix();

glPushMatrix();
        glTranslatef(x_lego_brick * 0.75, y_lego_brick * 5, z_lego_brick * 4);
        Dibuja_torre_2X2(10, _02DiffuseGr, _02Specular, _02Shininess);
glPopMatrix();

glPushMatrix();//Techo
        glTranslatef(0.0, y_lego_brick * 16, 0.0);
        glPushMatrix();//primer capa
                Dibuja_plataforma(2, 2, _02DiffuseGr, _02Specular, _02Shininess);
        glPopMatrix();
        glPushMatrix();
                glTranslatef(0.0,0.0,z_lego_brick*5);
```

```
            Dibuja_torre_2X4(0, _02DiffuseGr, _02Specular, _02Shininess);
glPopMatrix();
glPushMatrix();
    glTranslatef(z_lego_brick * 3.5, 0.0, 0.0);
    glPushMatrix();
            glTranslatef(0.0, 0.0, x_lego_brick * 0.75);
            glRotatef(90, 0.0, 1.0, 0.0);
            Dibuja_torre_2X4(0, _02DiffuseGr, _02Specular, _02Shininess);
    glPopMatrix();
   glPushMatrix();
            glTranslatef(0.0, 0.0, -x_lego_brick * 0.25);
            glRotatef(90, 0.0, 1.0, 0.0);
            Dibuja_torre_2X4(0, _02DiffuseGr, _02Specular, _02Shininess);
    glPopMatrix();
            glTranslatef(z_lego_brick, 0.0, x_lego_brick * 0.25);
            glRotatef(90, 0.0, 1.0, 0.0);
            Dibuja_torre_2X4(0, _02DiffuseGr, _02Specular, _02Shininess);
glPopMatrix();

glPushMatrix();
   glTranslatef(-z_lego_brick * 3.5, 0.0, 0.0);
   glPushMatrix();
            glTranslatef(0.0, 0.0, x_lego_brick * 0.75);
            glRotatef(90, 0.0, 1.0, 0.0);
            Dibuja_torre_2X4(0, _02DiffuseGr, _02Specular, _02Shininess);
   glPopMatrix();
   glPushMatrix();
            glTranslatef(0.0, 0.0, -x_lego_brick * 0.25);
            glRotatef(90, 0.0, 1.0, 0.0);
            Dibuja_torre_2X4(0, _02DiffuseGr, _02Specular, _02Shininess);
   glPopMatrix();
            glTranslatef(-z_lego_brick, 0.0, x_lego_brick * 0.25);
            glRotatef(90, 0.0, 1.0, 0.0);
            Dibuja_torre_2X4(0, _02DiffuseGr, _02Specular, _02Shininess);
glPopMatrix();

glPushMatrix();
            glTranslatef(0.0, 0.0, -z_lego_brick * 3);
            glPushMatrix();
                    glTranslatef(x_lego_brick / 2, 0.0, 0.0);
                    Dibuja_torre_2X4(0, _02DiffuseGr, _02Specular, _02Shininess);
            glPopMatrix();
            glPushMatrix();
                    glTranslatef(-x_lego_brick / 2, 0.0, 0.0);
                    Dibuja_torre_2X4(0, _02DiffuseGr, _02Specular, _02Shininess);
            glPopMatrix();
            glTranslatef(0.0, 0.0, -z_lego_brick);
            Dibuja_torre_2X4(0, _02DiffuseGr, _02Specular, _02Shininess);
glPopMatrix();
glPushMatrix();
            glTranslatef(0.0, 0.0, z_lego_brick * 3);
            Dibuja_torre_2X4(0, _02DiffuseGr, _02Specular, _02Shininess);
            glPushMatrix();
                    glTranslatef(-x_lego_brick, 0.0, 0.0);
                    Dibuja_torre_2X4(0, _02DiffuseGr, _02Specular, _02Shininess);
            glPopMatrix();
            glPushMatrix();
                    glTranslatef(x_lego_brick, 0.0, 0.0);
                    Dibuja_torre_2X4(0, _02DiffuseGr, _02Specular, _02Shininess);
            glPopMatrix();
            glTranslatef(0.0, 0.0, z_lego_brick);
            glPushMatrix();
```

```
                glTranslatef(-x_lego_brick / 2, 0.0, 0.0);
                Dibuja_torre_2X4(0, _02DiffuseGr, _02Specular, _02Shininess);
            glPopMatrix();
            glPushMatrix();
                glTranslatef(x_lego_brick / 2, 0.0, 0.0);
                Dibuja_torre_2X4(0, _02DiffuseGr, _02Specular, _02Shininess);
            glPopMatrix();
glPopMatrix();

glPushMatrix();//Segunda capa

            glTranslatef(x_lego_brick*0.25,y_lego_brick,0.0);
            Dibuja_plataforma(1, 1, _02DiffuseGr, _02Specular, _02Shininess);
            glPushMatrix();
                glTranslatef(-x_lego_brick*0.25, 0.0, z_lego_brick * 4.5);
                Dibuja_torre_2X2(0, _02DiffuseGr, _02Specular, _02Shininess);
            glPopMatrix();
            glPushMatrix();
                glTranslatef(0.0,0.0,-z_lego_brick*2);
                Dibuja_torre_2X4(0, _02DiffuseGr, _02Specular, _02Shininess);
                glTranslatef(-x_lego_brick*0.75,0.0,0.0);
                Dibuja_torre_2X2(0, _02DiffuseGr, _02Specular, _02Shininess);
                glTranslatef(0.0,0.0,z_lego_brick*0.75);
                Dibuja_torre_1X2(0, _02DiffuseGr, _02Specular, _02Shininess);
                glTranslatef(-x_lego_brick*0.5, 0.0, -z_lego_brick*0.25);
                Dibuja_torre_2X2(0, _02DiffuseGr, _02Specular, _02Shininess);
            glPopMatrix();

            glPushMatrix();
              glTranslatef(0.0,0.0,-z_lego_brick*2.75);
              glPushMatrix();
                glTranslatef(x_lego_brick * 0.25, 0.0, 0.0);
                Dibuja_torre_1X2(0, _02DiffuseGr, _02Specular, _02Shininess);
              glPopMatrix();
              glPushMatrix();
                glTranslatef(-x_lego_brick*0.25,0.0,0.0);
                Dibuja_torre_1X2(0, _02DiffuseGr, _02Specular, _02Shininess);
              glPopMatrix();
              glPushMatrix();
                glTranslatef(-x_lego_brick * 0.75, 0.0, 0.0);
                Dibuja_torre_1X2(0, _02DiffuseGr, _02Specular, _02Shininess);
              glPopMatrix();
              glPushMatrix();
                glTranslatef(-x_lego_brick*0.25, 0.0, -z_lego_brick*0.75);
                Dibuja_torre_2X2(0, _02DiffuseGr, _02Specular, _02Shininess);
               glPopMatrix();
            glPopMatrix();

            glPushMatrix();
              glTranslatef(z_lego_brick*1.5,0.0,0.0);
              glPushMatrix();
                glRotatef(90,0.0,1.0,0.0);
                Dibuja_torre_2X4(0, _02DiffuseGr, _02Specular, _02Shininess);
            glPopMatrix();

            glPushMatrix();
                glTranslatef(z_lego_brick, 0.0, 0.0);
                glRotatef(90, 0.0, 1.0, 0.0);
                Dibuja_torre_2X4(0, _02DiffuseGr, _02Specular, _02Shininess);
            glPopMatrix();

            glPushMatrix();
```

```
            glTranslatef(x_lego_brick*0.5,0.0,z_lego_brick*1.5);
            Dibuja_torre_2X2(0, _02DiffuseGr, _02Specular, _02Shininess);
glPopMatrix();

glPushMatrix();
            glTranslatef(x_lego_brick, 0.0, z_lego_brick*0.5);
            Dibuja_torre_2X2(0, _02DiffuseGr, _02Specular, _02Shininess);
glPopMatrix();

glPushMatrix();
            glTranslatef(0.0, 0.0, z_lego_brick * 1.5);
            Dibuja_torre_2X2(0, _02DiffuseGr, _02Specular, _02Shininess);
glPushMatrix();
            glTranslatef(0.0,0.0,z_lego_brick);
            Dibuja_torre_2X2(0, _02DiffuseGr, _02Specular, _02Shininess);
glPopMatrix();
glPushMatrix();
            glTranslatef(0.0, 0.0, -z_lego_brick*3);
            Dibuja_torre_2X2(0, _02DiffuseGr, _02Specular, _02Shininess);
glPopMatrix();
glPopMatrix();

glPopMatrix();

glPushMatrix();
            glTranslatef(0.0, 0.0, z_lego_brick * 2);
            Dibuja_torre_2X4(0, _02DiffuseGr, _02Specular, _02Shininess);
            glTranslatef(-x_lego_brick * 0.75, 0.0, 0.0);
            Dibuja_torre_2X2(0, _02DiffuseGr, _02Specular, _02Shininess);
            glTranslatef(0.0, 0.0, -z_lego_brick * 0.75);
            Dibuja_torre_1X2(0, _02DiffuseGr, _02Specular, _02Shininess);
            glTranslatef(-x_lego_brick * 0.5, 0.0, z_lego_brick * 0.25);
            Dibuja_torre_2X2(0, _02DiffuseGr, _02Specular, _02Shininess);
            glTranslatef(0.0, 0.0, z_lego_brick);
            Dibuja_torre_2X2(0, _02DiffuseGr, _02Specular, _02Shininess);
            glTranslatef(-x_lego_brick * 0.5, 0.0, -z_lego_brick);
            Dibuja_torre_2X2(0, _02DiffuseGr, _02Specular, _02Shininess);
            glTranslatef(-x_lego_brick * 0.5, 0.0, -z_lego_brick);
            Dibuja_torre_2X2(0, _02DiffuseGr, _02Specular, _02Shininess);
glPopMatrix();

glPushMatrix();
   glTranslatef(-z_lego_brick * 1.5, 0.0, 0.0);
glPushMatrix();
            glRotatef(90, 0.0, 1.0, 0.0);
            Dibuja_torre_2X4(0, _02DiffuseGr, _02Specular, _02Shininess);
glPopMatrix();
glPushMatrix();
            glTranslatef(-z_lego_brick, 0.0, 0.0);
            glRotatef(90, 0.0, 1.0, 0.0);
            Dibuja_torre_2X4(0, _02DiffuseGr, _02Specular, _02Shininess);
glPopMatrix();
  glPushMatrix();
            glTranslatef(-z_lego_brick*2, 0.0, 0.0);
            glRotatef(90, 0.0, 1.0, 0.0);
            Dibuja_torre_2X4(0, _02DiffuseGr, _02Specular, _02Shininess);
  glPopMatrix();
glPopMatrix();

glPushMatrix();
            glTranslatef(0.0, 0.0, z_lego_brick * 2.75);
            Dibuja_torre_1X4(0, _02DiffuseGr, _02Specular, _02Shininess);
```

```
            glTranslatef(0.0, 0.0, z_lego_brick*0.75);
            Dibuja_torre_2X4(0, _02DiffuseGr, _02Specular, _02Shininess);
   glPushMatrix();
            glTranslatef(-x_lego_brick*0.75, 0.0, 0.0);
            Dibuja_torre_2X2(0, _02DiffuseGr, _02Specular, _02Shininess);
            glTranslatef(0.0,0.0,-z_lego_brick*0.75);
            Dibuja_torre_1X2(0, _02DiffuseGr, _02Specular, _02Shininess);
   glPopMatrix();
glPopMatrix();

glPushMatrix();//Tercera capa
   glTranslatef(-x_lego_brick*0.25, y_lego_brick, z_lego_brick*0.5);
   Dibuja_plataforma(1, 1, _02DiffuseGr, _02Specular, _02Shininess);

glPushMatrix();
            glTranslatef(0.0,0.0,-z_lego_brick*2);
            Dibuja_torre_2X4(0, _02DiffuseGr, _02Specular, _02Shininess);
            glTranslatef(0.0,0.0,-z_lego_brick*0.75);
            Dibuja_torre_1X4(0, _02DiffuseGr, _02Specular, _02Shininess);
glPopMatrix();

glPushMatrix();
            glTranslatef(0.0, 0.0, z_lego_brick * 2);
            Dibuja_torre_2X4(0, _02DiffuseGr, _02Specular, _02Shininess);
            glTranslatef(0.0, 0.0, z_lego_brick * 0.75);
            Dibuja_torre_1X4(0, _02DiffuseGr, _02Specular, _02Shininess);
glPopMatrix();

glPushMatrix();
   glTranslatef(z_lego_brick*1.5,0.0,0.0);
   glPushMatrix();
      glRotatef(90,0.0,1.0,0.0);
      Dibuja_torre_2X4(0, _02DiffuseGr, _02Specular, _02Shininess);
   glPopMatrix();
            glTranslatef(z_lego_brick,0.0,0.0);
   glPushMatrix();
            glRotatef(90, 0.0, 1.0, 0.0);
            Dibuja_torre_2X4(0, _02DiffuseGr, _02Specular, _02Shininess);
   glPopMatrix();
 glPopMatrix();
     glPushMatrix();
            glTranslatef(-z_lego_brick * 1.5, 0.0, 0.0);
            glPushMatrix();
            glRotatef(90, 0.0, 1.0, 0.0);
            Dibuja_torre_2X4(0, _02DiffuseGr, _02Specular, _02Shininess);
            glPopMatrix();
            glTranslatef(-z_lego_brick, 0.0, 0.0);
        glPushMatrix();
            glRotatef(90, 0.0, 1.0, 0.0);
            Dibuja_torre_2X4(0, _02DiffuseGr, _02Specular, _02Shininess);
        glPopMatrix();
        glPopMatrix();

glPushMatrix();
            glTranslatef(x_lego_brick*0.75,0.0,z_lego_brick*1.5);
            Dibuja_torre_2X2(0,_02DiffuseGr,_02Specular,_02Shininess);
glPopMatrix();

glPushMatrix();
            glTranslatef(-x_lego_brick * 0.75, 0.0, z_lego_brick * 1.5);
            Dibuja_torre_2X2(0, _02DiffuseGr, _02Specular, _02Shininess);
glPopMatrix();
```

```
glPushMatrix();
        glTranslatef(-x_lego_brick * 0.75, 0.0, -z_lego_brick * 1.5);
        Dibuja_torre_2X2(0, _02DiffuseGr, _02Specular, _02Shininess);
glPopMatrix();

glPushMatrix();
        glTranslatef(x_lego_brick * 0.75, 0.0, -z_lego_brick * 1.5);
        Dibuja_torre_2X2(0, _02DiffuseGr, _02Specular, _02Shininess);
glPopMatrix();

glPushMatrix();//cuarte capa
        glTranslatef(0.0,y_lego_brick,0.0);
        Dibuja_plataforma(1,1,_02DiffuseGr,_02Specular,_02Shininess);
glPushMatrix();
        glTranslatef(z_lego_brick * 1.25,0.0,0.0);
glPushMatrix();
        glTranslatef(0.0,0.0,x_lego_brick*0.25);
        glRotatef(90,0.0,1.0,0.0);
        Dibuja_torre_1X4(0, _02DiffuseGr, _02Specular, _02Shininess);
glPopMatrix();
glPushMatrix();
        glTranslatef(0.0, 0.0, -x_lego_brick * 0.5);
        glRotatef(90, 0.0, 1.0, 0.0);
        Dibuja_torre_1X2(0, _02DiffuseGr, _02Specular, _02Shininess);
glPopMatrix();
glPushMatrix();
        glTranslatef(z_lego_brick*0.75, 0.0, 0.0);
        Dibuja_torre_2X2(0, _02DiffuseGr, _02Specular, _02Shininess);
glPopMatrix();
glPopMatrix();

glPushMatrix();
        glTranslatef(-z_lego_brick * 1.25, 0.0, 0.0);
        glPushMatrix();
        glTranslatef(0.0, 0.0, x_lego_brick * 0.25);
        glRotatef(90, 0.0, 1.0, 0.0);
        Dibuja_torre_1X4(0, _02DiffuseGr, _02Specular, _02Shininess);
glPopMatrix();
glPushMatrix();
        glTranslatef(0.0, 0.0, -x_lego_brick * 0.5);
        glRotatef(90, 0.0, 1.0, 0.0);
        Dibuja_torre_1X2(0, _02DiffuseGr, _02Specular, _02Shininess);
glPopMatrix();
glPushMatrix();
        glTranslatef(-z_lego_brick * 0.75, 0.0, 0.0);
        Dibuja_torre_2X2(0, _02DiffuseGr, _02Specular, _02Shininess);
glPopMatrix();
glPopMatrix();

glPushMatrix();
        glTranslatef(0.0,0.0,z_lego_brick*2);
        Dibuja_torre_2X2(0, _02DiffuseGr, _02Specular, _02Shininess);
glPopMatrix();

glPushMatrix();
        glTranslatef(0.0, 0.0, -z_lego_brick * 2);
        Dibuja_torre_2X2(0, _02DiffuseGr, _02Specular, _02Shininess);
glPopMatrix();

glPushMatrix();//quinta capa
        glTranslatef(0.0, y_lego_brick,0.0);
```

```
                                                glPushMatrix();
                                                        glTranslatef(0.0,0.0,z_lego_brick*0.5);
                                                        Dibuja_torre_2X4(0, _02DiffuseGr, _02Specular, _02Shininess);
                                                glPopMatrix();
                                                glPushMatrix();
                                                        glTranslatef(0.0, 0.0, -z_lego_brick * 0.5);
                                                        Dibuja_torre_2X4(0, _02DiffuseGr, _02Specular, _02Shininess);
                                                glPopMatrix();

                                                glPushMatrix();//Punta
                                                        glTranslatef(0.0,y_lego_brick,0.0);
                                                        Dibuja_torre_2X2(0, _02DiffuseGr, _02Specular, _02Shininess);
                                                glPopMatrix();
                                        glPopMatrix();
                                glPopMatrix();
                        glPopMatrix();
                glPopMatrix();
        glPopMatrix();

}
```

GEOMETRÍAS LEGO

//Para la construcción de las primitivas del bloque estilo LEGO, se siguió una estructura básica de prisma cuadrangular como estructura principal y cilindros pequeños sobre el prisma para simular las partes que embonan entre sí.

//Como ejemplo, la siguiente función construye un bloque estilo LEGO de 2X4 (2 cilindros de largo y 4 de ancho).

```
void CFiguras::lego_2X4(GLfloat diffuse1[], GLfloat specular1[], GLfloat shininess1)  //Funcion creacion prisma
{
        glDisable(GL_TEXTURE_2D);

        GLfloat vertice[8][3] = {
                                        {1 ,-0.25, 0.5},    //Coordenadas Vértice 0 V0
                                        {-1 ,-0.25, 0.5 },    //Coordenadas Vértice 1 V1
                                        {-1 ,-0.25, -0.5},    //Coordenadas Vértice 2 V2
                                        {1 ,-0.25, -0.5},    //Coordenadas Vértice 3 V3
                                        {1 ,0.25, 0.5},    //Coordenadas Vértice 4 V4
                                        {1 ,0.25, -0.5},    //Coordenadas Vértice 5 V5
                                        {-1 ,0.25, -0.5},    //Coordenadas Vértice 6 V6
                                        {-1 ,0.25, 0.5},    //Coordenadas Vértice 7 V7
        };

        glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, diffuse1);
```

```
glMaterialfv(GL_FRONT, GL_SPECULAR, specular1);
glMaterialf(GL_FRONT, GL_SHININESS, shininess1);

//Comienza la construcción del prisma en primer lugar.

glBegin(GL_POLYGON);        //Front
        glColor3f(1.0, 1.0, 1.0);
        glNormal3f(0.0f, 0.0f, 1.0f);
        glTexCoord2f(1.0f, 0.0f); glVertex3fv(vertice[1]);
        glTexCoord2f(1.0f, 1.0f); glVertex3fv(vertice[0]);
        glTexCoord2f(0.0f, 1.0f); glVertex3fv(vertice[4]);
        glTexCoord2f(0.0f, 0.0f); glVertex3fv(vertice[7]);
glEnd();

        .
        .
        .
```

Demás caras

```
        .
        .
        .

//Posterior a ello, construimos los 8 cilindros que embonan con los otros bloques y los ubicamos sobre el prisma


float radio = 0.1;
float altura = 0.2;
int resolucion = 30;

float v1[] = { 0.0, 0.0, 0.0 };
float v2[] = { 0.0, 0.0, 0.0 };
float v3[] = { 0.0, 0.0, 0.0 };
float v4[] = { 0.0, 0.0, 0.0 };
float v5[] = { 0.0, 0.0, 0.0 };

float angulo = 2 * 3.14 / resolucion;

//float ctext_s = 1/resolucion-1;
float ctext_s = 1.0 / resolucion;
float ctext_t = 0.0;

glPushMatrix();
        glTranslatef(0.75,0.25,0.25);
        for (int i = 0; i < resolucion; i++)
        {
                v2[0] = radio * cos(angulo * i);
                v2[1] = 0;
                v2[2] = radio * sin(angulo * i);

                v3[0] = radio * cos(angulo * (i + 1));
                v3[1] = 0;
                v3[2] = radio * sin(angulo * (i + 1));

                v4[0] = radio * cos(angulo * i);
                v4[1] = altura;
                v4[2] = radio * sin(angulo * i);

                v5[0] = radio * cos(angulo * (i + 1));
                v5[1] = altura;
                v5[2] = radio * sin(angulo * (i + 1));

                glBegin(GL_POLYGON);
                        glNormal3f(0.0f, -1.0f, 0.0f);
                        glVertex3f(0.0, 0.0, 0.0);
                        glVertex3fv(v2);
                        glVertex3fv(v3);
                glEnd();

                glBegin(GL_POLYGON);
                        glNormal3f(0.0f, 1.0f, 0.0f);
                        glVertex3f(0.0, altura, 0.0);
```

```
                glVertex3fv(v4);
                glVertex3fv(v5);
        glEnd();

        glBegin(GL_POLYGON);
                glNormal3f(v2[0], 0.0f, v2[2]);
                glTexCoord2f(ctext_s * i, 0.0f);            glVertex3fv(v2);
                glTexCoord2f(ctext_s * (i + 1), 0.0f);      glVertex3fv(v3);
                glTexCoord2f(ctext_s * (i + 1), 1.0f);      glVertex3fv(v5);
                glTexCoord2f(ctext_s * i, 1.0f);            glVertex3fv(v4);
        glEnd();
    }
glPopMatrix();

    .
    .
    .
```

Demás caras

```
    .
    .
    .
```

ILUMINACIÓN

```
//Esta función crea y administra las luces estas solo se verán reflejadas en los materiales

void Luces() {

    //Luz principal del entorno
    glLightfv(GL_LIGHT0,GL_AMBIENT,LightAmbient);//Se implementa una luz ambiental para todo el escenario
    glLightfv(GL_LIGHT0, GL_POSITION, LightPosition);//Se le implementó una posición para que alumbrara con cierta
                                                     //inclinacion

//En este caso se utilizaron dos focos para darle un tono amarillo al kiosco por la noche y que diera el efecto de que era iluminado por los faroles
    //Luz del foco1
    glLightfv(GL_LIGHT1, GL_DIFFUSE, focoDiffuse);//Se utiliza iluminación puntal para dar efecto que proviene de un objeto
    glLightfv(GL_LIGHT1, GL_SPECULAR, NightLightSpecular);//Se utiliza para que el reflejo de la luz luzca más en el objeto
    glLightfv(GL_LIGHT1, GL_SPOT_DIRECTION, focoDirection);//Se utiliza para enfocar la luz en un cierto en cierta dirección
    glLightf(GL_LIGHT1, GL_SPOT_CUTOFF, 180);//Se usa para dibuja un cono de iluminación y se le da un ángulo  que
                                             //queremos que este tenga
    glLightf(GL_LIGHT1, GL_CONSTANT_ATTENUATION, 2.5);//Se le asigna cierta atenuación a la iluminación para que esta
                                                      //pueda verse más natural en los objetos y no exceda en la
```

//En esta parte se utilizaron cuatro foco con distintos colores de iluminación y también se implemento una esfera la cual simularia ser el foco.

```
	//Luz del foco3
	glLightfv(GL_LIGHT3, GL_DIFFUSE, focoDiffuseverde);//Aqui se le dio una tonalidad verde a la iluminación
	glLightfv(GL_LIGHT3, GL_SPECULAR, NightLightSpecular);////Se utiliza para que el reflejo de la luz luzca más en el objeto
	glLightfv(GL_LIGHT3, GL_SPOT_DIRECTION, focoDirection2);//Se utiliza para enfocar la luz en un cierto en cierta dirección
	glLightf(GL_LIGHT3, GL_SPOT_CUTOFF, 10);//Se usa para dibuja un cono de iluminación y se le da un ángulo  que
                                                //queremos que este tenga
	glLightf(GL_LIGHT3, GL_SPOT_EXPONENT, 128);//Se le aplico para que pudiese atenuar o intensificar la iluminación

		.
		.
		.
	Resto de focos
		.
		.
		.
```

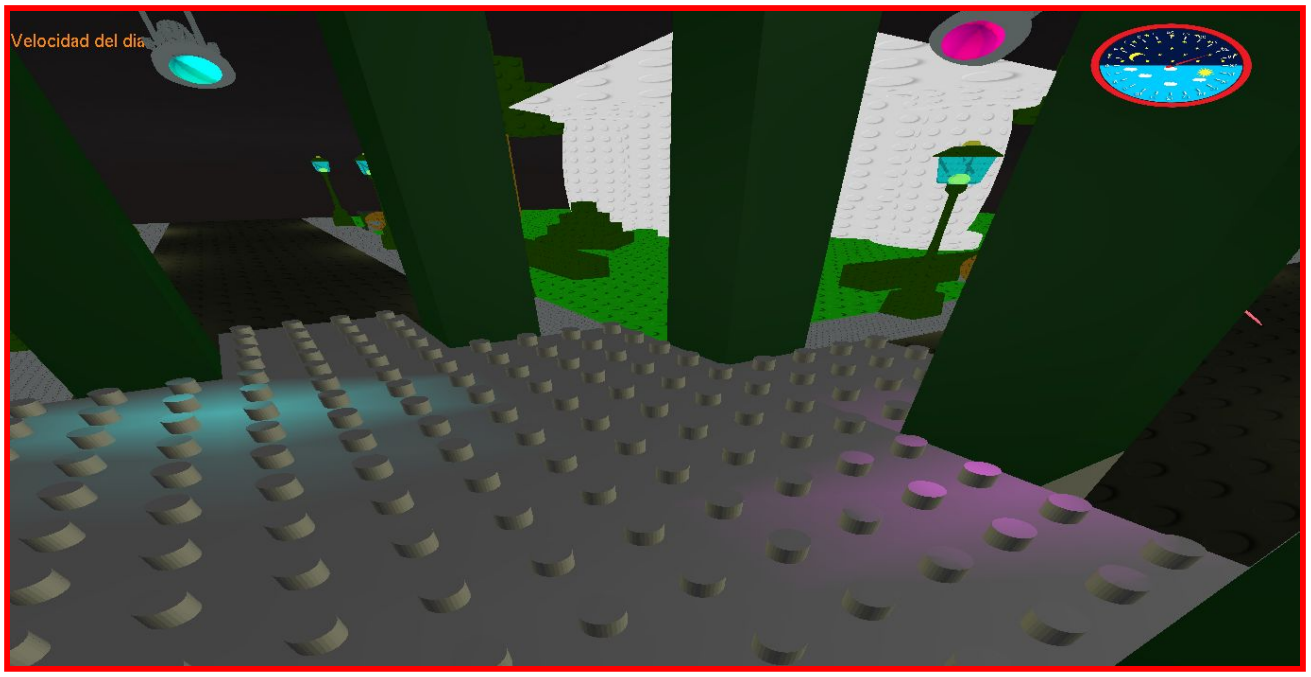
//En esta sección se les aplicó un condicional para que las luces solo se activarán durante el ciclo de la noche

```
	if (night == true) {
		glLightfv(GL_LIGHT0, GL_DIFFUSE, NightLightDiffuse);
		glLightfv(GL_LIGHT0, GL_SPECULAR, NightLightSpecular);
	}
	else {
		glLightfv(GL_LIGHT0, GL_DIFFUSE, DayLightDiffuse);
		glLightfv(GL_LIGHT0, GL_SPECULAR, DayLightSpecular);
	}


	if (night == true) {
		glEnable(GL_LIGHT1);
		glEnable(GL_LIGHT2);
		glEnable(GL_LIGHT3);
		glEnable(GL_LIGHT4);
		glEnable(GL_LIGHT5);
		glEnable(GL_LIGHT6);

	}
	else {
		glDisable(GL_LIGHT1);
		glDisable(GL_LIGHT2);
		glDisable(GL_LIGHT3);
		glDisable(GL_LIGHT4);
		glDisable(GL_LIGHT5);
		glDisable(GL_LIGHT6);
	}

}
```



ANIMACIONES

A continuación se detallan las transformaciones utilizadas en las animaciones dentro del entorno virtual.

- **Animación del encendido/apagado de lámpara (Kiosko)**: En primer lugar se dibuja la geometría de cada lámpara siendo el último elemento el foco (esfera), una vez hecho esto, se crea una luz del tipo reflector (spotlight) con su componente difusa del color correspondiente, ángulo de apertura de 10° (20° en total), exponente de luz en 128 (máximo enfoque), dirección {0.0,-1.0, 0.0} (apuntando al origen del sistema de referencia) y posición igual a la del foco (esfera). Para lograr el efecto de fuente de luz, los pasos son:

    1. Desactivar modelo de iluminación.
    2. Dibujar la geometría del foco.
    3. Ubicar la fuente de luz en las coordenadas de la geometría.
    4. Activar modelo de iluminación.

- **Animación de rotación de lámpara (Kiosko):** Es una rotación sencilla, respecto al centro de la lámpara, controlada por las teclas VBNM para los cuatro objetos y restringida en 45° (90° totales) con un condicional.

- **Animación movimiento de piernas avanzar/retroceder en personajes:** La rotación se realiza respecto al centro del cilindro que compone la unión con el torso, está restringida en 40° (80° totales), para asegurar un movimiento armónico de ambas piernas, se utilizaron 2 recorridos llamados step1, step2 para Zero-Two y step3, step4 para Cuphead, los cuales se activan/desactivan simultáneamente. La animación está condicionada a las banderas _3rdPerson, firstCharacter y movePer, permitiendo que solo exista animación si el personaje se mueve.

- **Animación recorrido aéreo de cámara:** Para realizar esta animación nos ayudamos del método Rotate_Camera(), de la clase CCamera, de la cabecera Camera.h y utilizando el objeto airCamera se realiza la llamada de manera constante con un valor fijo para la velocidad de rotación. Se utiliza una bandera para determinar si estamos en modo cámara aérea.

- **Animación de la luna**: Se aplicó una función del tipo $y = Az^2 + Bz + C$ (parábola) con $z\varepsilon[90,-90]$ para el movimiento de traslación, en donde:

    a. A es un valor negativo para lograr la orientación “hacia abajo” de la parábola y menor a 1. Mientras más pequeño, mayor es la apertura de la trayectoria.
    b. B es igual a 0 para no desplazar la trayectoria con referencia al sistema de coordenadas del mundo.
    c. C el valor del máximo en y (altura máxima).

    Durante el recorrido, se utilizó una transformación de rotación de 0° a 360° sobre el eje vertical de la geometría de la luna.

- **Animación del reloj dia/noche**: Para lograr el texturizado en 2D sobre la pantalla, se implementó una función dedicada a cambiar el modelo de visualización de perspectiva a ortogonal, luego se procede a dibujar la texturas del reloj y la manecilla, esta última con una rotación cíclica de 0° a 360° sobre el eje Z. Finalmente se regresa al modelo en perspectiva.

  Una vez que amanece, se reproduce el sonido de un gallo, para evitar la reproducción continua, la llamada al play2D() se realiza dentro del caso contrario else, cada vez que el ciclo de los 360° se cumple.

- **Efecto de cambio de escenario dia/noche**: Debido a que las texturas no interactúan con el modelo de iluminación, para lograr el efecto de oscurecimiento durante la noche, estos fueron los pasos realizados:

  1. A cada una de las texturas originales, se aplicó un filtro de iluminación baja en un editor de imágenes.
  2. Se cargaron todas las texturas, tanto para el día como para la noche.
  3. En un arreglo de 40 elementos, se almacenaron los índices de las texturas de día.
  4. Se implementó una función dedicada al intercambio de índices entre las texturas de día y las de noche.
  5. Se implementó una función dedicada a la recuperación de los índices originales, desde el arreglo, de las texturas de día.

  El proceso de re-texturizado depende de una variable del tipo bool (bandera).

  *Los modelos en formato .3ds, que cuentan con texturizado, no tienen ninguna modificación.

Animación por Keyframes

```
//Con esta función iniciamos/reiniciamos el índice del arreglo de frames y los valores dentro de éste

void Init_Key_Frame() {
	FrameIndex = 0;
	for (int i = 0; i < MAX_FRAMES; i++)
	{

		KeyFrame[i].rotlamp1 = 0;
		KeyFrame[i].rotlamp2 = 0;
		KeyFrame[i].rotlamp3 = 0;
		KeyFrame[i].rotlamp4 = 0;
		KeyFrame[i].rotinc1 = 0;
		KeyFrame[i].rotinc2 = 0;
		KeyFrame[i].rotinc3 = 0;
		KeyFrame[i].rotinc4 = 0;
	}
}

//Esta función guarda el frame

void guardaFrame ()
{
	printf("frameindex %d\n", FrameIndex);

	KeyFrame[FrameIndex].rotlamp1 = rotlamp1;
	KeyFrame[FrameIndex].rotlamp2 = rotlamp2;
	KeyFrame[FrameIndex].rotlamp3 = rotlamp3;
	KeyFrame[FrameIndex].rotlamp4 = rotlamp4;

	FrameIndex++;
}

//Esta función sirve para resetear las rotaciones de las lamparas

void resetElements()
{


	rotlamp1 = KeyFrame[0].rotlamp1;
	rotlamp2 = KeyFrame[0].rotlamp2;
	rotlamp3 = KeyFrame[0].rotlamp3;
	rotlamp4 = KeyFrame[0].rotlamp4;
```

```
}


//Función de interpolación


void interpolacion()
{
	KeyFrame[playIndex].rotinc1 = (KeyFrame[playIndex + 1].rotlamp1 - KeyFrame[playIndex].rotlamp1) / i_max_steps;
	KeyFrame[playIndex].rotinc2 = (KeyFrame[playIndex + 1].rotlamp2 - KeyFrame[playIndex].rotlamp2) / i_max_steps;
	KeyFrame[playIndex].rotinc3 = (KeyFrame[playIndex + 1].rotlamp3 - KeyFrame[playIndex].rotlamp3) / i_max_steps;
	KeyFrame[playIndex].rotinc4 = (KeyFrame[playIndex + 1].rotlamp4 - KeyFrame[playIndex].rotlamp4) / i_max_steps;
}

///Función de submenú con opciones para almacenar, reproducir, y resetear los frames.

void menuKeyFrame(int id)
{
	switch (id)
	{
	case 0:	//Save KeyFrame
		if (FrameIndex < MAX_FRAMES)
		{
			guardaFrame();
		}
		break;

	case 1:	//Play animation
		if (play == false && FrameIndex > 1)
		{
			resetElements();
			//First Interpolation
			interpolacion();

			play = true;
			playIndex = 0;
			i_curr_steps = 0;
		}
		else
		{
			play = false;
		}
		break;
	case 2: //Reset buffer
		Init_Key_Frame();
	}
}

//Función de menu para GLUT
void menu(int id)
{

}
```

AUDIO

```
//Funcion creada para inicializar el audio se implementa la ayuda de la biblioteca “irrKlang.h”
void startAudio() {

	engine = createIrrKlangDevice();//Es el motor de carga del audio

	if (!engine) {
		printf("\nError al inciar el sistema de audio");//Mensaje utilizado en caso de que el sonido correspondiente
								//no se encuentre en la ruta específica
	}

	snd1 = engine->play2D("Audio/nostalgic.wav", true, false, true);//Se implementa para asignar el sonido a ejecutar
									//y en donde se encuentra localizado con su ruta
									//carpeta y nombre correspondiente en formato .wav
	//Inicializamos recursos de sonido
	jumpZ = engine->addSoundSourceFromFile("Audio/zeroJump.wav");//Este se utilizó para asignar un sonido a cada
									//personaje este es para ZeroTwo
	jumpZ->setDefaultVolume(0.4); //Se usa para darle un volumen general al audio

	rooster = engine->addSoundSourceFromFile("Audio/rooster.wav");//Se utilizó para el sonido de un gallo que se
									//reproduce cada vez que amanece
	rooster->setDefaultVolume(0.4);//Se usa para darle un volumen general al audio

	cupson = engine->addSoundSourceFromFile("Audio/Cuphead.wav");//Este se utilizo para asignar un sonido a cada
									//personaje este es para Cuphead
	cupson->setDefaultVolume(0.4);//Se usa para darle un volumen general al audio

}
```

Listado de funciones en el código fuente principal y su utilidad.

- **void Init_Key_Frame():** Inicializa/Reinicia las variables del buffer de KeyFrames.
- **void startAudio():** Inicializa el sistema y los recursos de audio.
- **void Texture_Load():** Carga todas las texturas utilizadas en el programa.
- **void Models_Load():** Carga todos los modelos con formato .3ds.
- **void Luces():** Crea y controla todas las luces del programa.
- **void Oscurecer():** Intercambia los índices de las texturas de día y noche.
- **void Restaurar_Indice():** Recupera los índices originales de las texturas de día.
- **void guardaFrame ():** Permite guardar el estado de las variables de rotación, así como el indice del KeyFrame hasta el momento en que es llamada.
- **void resetElements():** Reinicia los elementos guardados en el buffer de KeyFrames para reproducir la animación desde el inicio.
- **void interpolacion():** Obtención de los valores del incremento de rotación por interpolación de los últimos valores de rotación dentro del rango de máximos “movimientos” permitidos.
- **void Dibuja_suelo ():** Construye las geometrías que componen el suelo.
- **void Dibuja_plataforma(int largo, int ancho, GLfloat diffuse[], GLfloat specular[], GLfloat shininess):** Dibuja una plataforma de bloques de LEGO de tamaño variable.
- **void Dibuja_torre_2X4(int altura,GLfloat diffuse [], GLfloat specular [], GLfloat shininess):** Dibuja una torre de bloques de lego del tipo 2X4 y altura variable.
- **void Dibuja_torre_1X4(int altura,GLfloat diffuse [], GLfloat specular [], GLfloat shininess):** Dibuja una torre de bloques de lego del tipo 1X4 y altura variable.
- **void Dibuja_torre_2X2(int altura,GLfloat diffuse [], GLfloat specular [], GLfloat shininess):** Dibuja una torre de bloques de lego del tipo 2X2 y altura variable.
- **void Dibuja_torre_1X2(int altura,GLfloat diffuse [], GLfloat specular [], GLfloat shininess):** Dibuja una torre de bloques de lego del tipo 1X2 y altura variable.
- **void Dibuja_personaje():** Dibuja a Zero-Two.
- **void Dibuja_personaje2():** Dibuja a Cuphead.
- **void OnScreen_GUI():** Cambia el modo de visualización de perspectiva a ortogonal, dibuja las texturas 2D sobre la escena 3D, regresa el modo de visualización a perspectiva.
- **void InitGL(GLvoid):** Parámetros iniciales de la aplicación.
- **void pintaTexto(float x, float y, float z, void *font,char *string):** Muestra texto en pantalla.
- **void menuKeyFrame(int id):** Implementa las opciones para el sub-menú relacionado a la animación por KeyFrames.
- **void menu(int id):** Función de menú que contiene las opciones del sub-menú anterior.
- **void display():** Función de dibujo.
- **void animación():** Controla los parámetros, banderas y condiciones de las animaciones.
- **void reshape (int width , int height):** Función de redimensionado y modelo de visualización de la ventana.
- **void keyboard (unsigned char key, int x, int y):** Función para el manejo de teclado.
- **void arrow_keys (int a_keys, int x, int y):** Función para el manejo de teclas especiales.
- **int main (int argc, char** argv):** Función principal.

Listado de funciones de la cabecera figuras.h y su utilidad. (PRIMITIVAS)

- **void esfera(GLfloat radio, int meridianos, int paralelos, GLuint text)**: Esfera texturizada.
- **void cilindro(float radio, float altura, int resolucion, GLuint text)**: Cilindro texturizado.
- **void cono(float altura, float radio, int resolucion, GLuint text)**: Cono texturizado.
- **void prisma (float altura, float largo, float profundidad, GLuint text)**: Prisma con texturizado homogéneo.
- **void prisma2(GLuint text, GLuint text2, int rep_u, int rep_v)**: Prisma con texturizado de repetición variable en ambas coordenadas UV.
- **void prismaM(GLfloat diffuse1[], GLfloat specular1[], GLfloat shininess1):** Prisma que admite parametros de un material.
- **void lego_2X4(GLfloat diffuse1[], GLfloat specular1[], GLfloat shininess1):** Bloque de LEGO de 2X4 que admite parametros de un material
- **void lego_2X2(GLfloat diffuse1[], GLfloat specular1[], GLfloat shininess1):** Bloque de LEGO de 1X4 que admite parametros de un material
- **void lego_1X4(GLfloat diffuse1[], GLfloat specular1[], GLfloat shininess1):** Bloque de LEGO de 2X2 que admite parametros de un material

- **void lego_1X2(GLfloat diffuse1[], GLfloat specular1[], GLfloat shininess1):** Bloque de LEGO de 1X2 que admite parametros de un material
- **void torso(GLuint text, GLuint text2, GLuint text3):** Trapezoide con coordenadas de texturizado espaciales STUV que corrige la deformación de una textura plana. Diseñada especialmente para construir un torso estilo LEGO.
- **void pierna(GLfloat diffuse1[], GLfloat specular1[], GLfloat shininess1, GLfloat diffuse2[], GLfloat specular2[], GLfloat shininess2):** Primitiva que construye una pierna estilo LEGO con admisión de parametros de dos materiales.
- **void plano(GLuint text)**: Plano texturizado.
- **void skybox(float altura, float largo, float profundidad, GLuint text)**: Cubemap del tipo skybox con coordenadas U dividido en 4 y V dividido en 3
- **void texturaGUI(GLuint text)**: Función para la creación de texturas en 2D, pensadas para la GUI sobre la escena 3D.

Anexo: Código fuente de cabeceras.

Camera.h

```
#include "Main.h"
#define CAMERASPEED 0.07f // The Camera Speed
//NEW//////////////////NEW//////////////////NEW//////////////////NEW////////////////

//////////////////////////////////////
//The tVector3 Struct
//////////////////////////////////////
typedef struct tVector3 // expanded 3D vector struct
{
	tVector3() {} // constructor
	tVector3 (float new_x, float new_y, float new_z) // initialize constructor
	{x = new_x; y = new_y; z = new_z;}
	// overload + operator so that we easier can add vectors
	tVector3 operator+(tVector3 vVector) {return tVector3(vVector.x+x, vVector.y+y, vVector.z+z);}
	// overload - operator that we easier can subtract vectors
	tVector3 operator-(tVector3 vVector) {return tVector3(x-vVector.x, y-vVector.y, z-vVector.z);}
	// overload * operator that we easier can multiply by scalars
	tVector3 operator*(float number)  {return tVector3(x*number, y*number, z*number);}
	// overload / operator that we easier can divide by a scalar
	tVector3 operator/(float number)  {return tVector3(x/number, y/number, z/number);}

	float x, y, z; // 3D vector coordinates
}tVector3;

//////////////////////////////////////
//The CCamera Class
//////////////////////////////////////
class CCamera
{
	public:
		tVector3 mPos;
		tVector3 mView;
		tVector3 mUp;

		void Strafe_Camera(float speed);
		void Move_Camera(float speed);
		void Rotate_Camera(float speed);
		void Rotate_View(float speed);
		void UpDown_Camera(float speed);
		void Position_Camera(float pos_x, float pos_y, float pos_z, float view_x, float view_y, float view_z, float up_x,
		 float up_y,float up_z);
};
```

Se implementa en el caso de la cámara aérea.

figuras.h

```
#include "Main.h"

class CFiguras
{
	public:

	float text_der;
	float text_izq;

	void esfera(GLfloat radio, int meridianos, int paralelos, GLuint text); //Funcíon creacion esfera
	void cilindro(float radio, float altura, int resolucion, GLuint text);		//Funcíon creacion cilindro
	void cono(float altura, float radio, int resolucion, GLuint text);		//Funcíon creacion cono
	void prisma (float altura, float largo, float profundidad, GLuint text);//Funcíon creacion prisma
	void prisma2(GLuint text, GLuint text2, int rep_u, int rep_v);
	void prismaM(GLfloat diffuse1[], GLfloat specular1[], GLfloat shininess1);
	void lego_2X4(GLfloat diffuse1[], GLfloat specular1[], GLfloat shininess1);
	void lego_2X2(GLfloat diffuse1[], GLfloat specular1[], GLfloat shininess1);
	void lego_1X4(GLfloat diffuse1[], GLfloat specular1[], GLfloat shininess1);
	void lego_1X2(GLfloat diffuse1[], GLfloat specular1[], GLfloat shininess1);
	void torso(GLuint text, GLuint text2, GLuint text3);
	void pierna(GLfloat diffuse1[], GLfloat specular1[], GLfloat shininess1, GLfloat diffuse2[], GLfloat specular2[], GLfloat
shininess2);
	void plano(GLuint text);
	void skybox(float altura, float largo, float profundidad, GLuint text);
	void texturaGUI(GLuint text);

};
```

texture.h

```
#include "Main.h"

class CTexture
{
public:
	char* name;
	unsigned char*  imageData;
	int bpp;			// Image Color Depth In Bits Per Pixel
	int width;			// Image Width
	int height;			// Image Height
	unsigned int  GLindex;

	bool LoadTGA(char* filename);  // Loads A TGA File Into Memory
	bool LoadBMP(char* filename);
	void BuildGLTexture();
	void ReleaseImage();
};
```

Contacto de los programadores.

jose_hernandez_castro@outlook.com
+52 55 5053 3845.

fgz.gonz@gmail.com
+52 55 30 14 1446.