Paniagua Trejo Erick
Bermúdez Sotelo Gustavo
Manual Proyecto Final “Feria”
8/05/19

Objetivo

El presente manual tiene como objetivo explicar el programa generado para el proyecto final del laboratorio de computación gráfica, esto es explicando el código y mostrando el resultado.

Bibliotecas

En primer lugar se deben de declarar las bibliotecas y cabeceras para poder referenciar las variables declaradas en esos archivos y poder utilizarlos sin error alguno, se deben de llamar con el comando “#include” y a continuación el nombre de la cabecera a llamar entre comillas.

Variables

En primer lugar se deben de declarar las variables a utilizar para poder hacer el programa y que tenga un funcionamiento autónomo, a continuación se muestran las variables necesarias para almacenar los datos de los key frames, las posiciones, el movimiento del monito y el incremento para hacer la interpolación que forma parte de la animación.

```
// Animación de carro montaña rusa ( i, I )
// Noche (n) Día (m)
#include "texture.h"
#include "figuras.h"
#include "Camera.h"
#include "cmodel/CModel.h"
#include "Main.h"
//Solo para Visual Studio 2015
#if (_MSC_VER == 1900)
#   pragma comment( lib, "legacy_stdio_definitions.lib" )
#endif

//NEW//////////////////NEW//////////////////NEW//////////////////NEW////////////////
static GLuint ciudad_display_list;  //Display List for the Monito

//NEW// Keyframes
float posX =0, posY = 2.5, posZ =-3.5, rotRodIzq = 0;
float giroMonito = 0;
float movBrazoDer = 0.0;

#define MAX_FRAMES 9
int i_max_steps = 90;
int i_curr_steps = 0;
typedef struct _frame
{
    //Variables para GUARDAR Key Frames
    float posX;     //Variable para PosicionX
    float posY;     //Variable para PosicionY
    float posZ;     //Variable para PosicionZ
    float incX;     //Variable para IncrementoX
    float incY;     //Variable para IncrementoY
    float incZ;     //Variable para IncrementoZ
    float rotRodIzq;
    float rotInc;
    float giroMonito;
    float giroMonitoInc;
    float movBrazoDer;
    float movBrazoDerInc;

}FRAME;
```

Se declaran las matrices que dan lugar a los diferentes valores de la iluminación local (Difuso, Especular y Ambiental).

```
FRAME KeyFrame[MAX_FRAMES];
int FrameIndex=5;           //introducir datos
bool play=false;
int playIndex=0;

//NEW//////////////////NEW//////////////////NEW//////////////////NEW////////////////

int w = 500, h = 500;
int frame=0,time,timebase=0;
char s[30];

CCamera objCamera;  //Create objet Camera

GLfloat g_lookupdown = 0.0f;    // Look Position In The Z-Axis (NEW)

int font=(int)GLUT_BITMAP_HELVETICA_18;

//GLfloat Diffuse[]= { 1.0f, 1.0f, 1.0f, 1.0f };              // Diffuse Light Values
GLfloat Diffuse[]= { 0.5f, 0.5f, 0.5f, 1.0f };             // Diffuse Light Values
GLfloat Specular[] = { 1.0, 1.0, 1.0, 1.0 };               // Specular Light Values
GLfloat Position[]= { 0.0f, 7.0f, -5.0f, 0.0f };           // Light Position
GLfloat Position2[]= { 0.0f, 0.0f, -5.0f, 1.0f };          // Light Position

GLfloat m_dif1[] = { 0.0f, 0.2f, 1.0f, 1.0f };             // Diffuse Light Values
GLfloat m_spec1[] = { 0.0, 0.0, 0.0, 1.0 };               // Specular Light Values
GLfloat m_amb1[] = { 0.0, 0.0, 0.0, 1.0 };                // Ambiental Light Values
GLfloat m_s1[] = {18};

GLfloat m_dif2[] = { 0.8f, 0.2f, 0.0f, 1.0f };             // Diffuse Light Values
GLfloat m_spec2[] = { 0.0, 0.0, 0.0, 1.0 };               // Specular Light Values
GLfloat m_amb2[] = { 0.0, 0.0, 0.0, 1.0 };                // Ambiental Light Values
GLfloat m_s2[] = {22};

CTexture text1;
CTexture text2;
CTexture text4; //Pavimento
CTexture text5; //Pasto01
CTexture text6; //Casa01
```


Por otra parte tenemos la declaración de variables con las palabras “CTexture” y Cfiguras”, la primera nos ayudará a declarar variables para el texturizado de los prismas a implementar, esto ayuda a dar realismo al escenario, con la segunda es la declaración de variables para figuras.

```
//////////////////////////////////// Montaña Rusa//////////////////////
CTexture text7; // Tubo amarillo
CTexture text8; //Apoyo metálico
CTexture text9; //Bandera carreras
//////////////////////////////////////////////////////////////////////////

//////////////////////Feria/////////////////
CTexture text10;
CTexture text11;
CTexture text12;
//////////////////////////////////////////


CTexture tree;

CFiguras fig1;
CFiguras fig2;
CFiguras fig3;
CFiguras fig4;  //Pasto01
CFiguras fig5;  //Casa01
CFiguras fig6;
CFiguras fig7;  //Para crear Monito

//////////////////////////////////// Montaña Rusa//////////////////////
CFiguras fig8;  // Tubo Montaña Rusa
CFiguras fig9;  // Apoyo metálico
CFiguras fig10; //Bandera carreras
//////////////////////////////////////////////////////////////////////////

////////////////////////////// Feria ////////////////////////////////
CFiguras fig11; //Tubo azul
CFiguras fig12; //Puesto
CFiguras fig13; //Puesto 2
//////////////////////////////////////////////////////////////////////

//////////////////////////////////////////////////////////////////////
```

Aquí se declaran las variables que nos ayudarán para el movimiento del carro, la rotación del mismo, posición de la cámara y valores booleanos que nos ayudarán a determinar el principio y fin de un circuito determinado.

```
//Animación del coche
float angRot = 0.0;
float movKitX;
float movKitZ;
float movKitY;
float rotKit = 0.0;
float rotKitZ = 0.0;
float rotKitX = 0.0;
float rotTires = 0.0;
bool g_fanimacion = false;
bool g_avanza = false;

//Cámara
float camaraX = 0.0;
float camaraY = 0.0;
float camaraZ = 0.0;



// Recorrido del carro en la montaña rusa
bool circuito = false;
bool recorrido1 = true;
bool recorrido2 = false;
bool recorrido3 = false;
bool recorrido4 = false;
bool recorrido5 = false;
bool recorrido6 = false;
bool recorrido7 = false;
bool recorrido8 = false;
bool recorrido9 = false;
bool recorrido10 = false;
bool recorrido11 = false;
```

Para poder calcular el material de los distintos objetos que hay en la feria se declaran siempre tres matrices, los valores de la luz difusa, la especular y la posición.

```
GLfloat SunDiffuse[] = { 1.0f, 1.0f, 1.0f, 1.0f };          // Diffuse Light Values
GLfloat SunSpecular[] = { 1.0, 1.0, 1.0, 1.0 };             // Specular Light Values
GLfloat SunPosition[] = { 0.0f, 0.0f, 0.0f, 1.0f };         // Light Position

GLfloat montanaDiffuse[] = { 1.0f, 1.0f, 1.0f, 1.0f };      // Diffuse Light Values
GLfloat montanaSpecular[] = { 1.0, 1.0, 1.0, 1.0 };         // Specular Light Values
GLfloat montanaPosition[] = { 0.0f, 0.0f, 0.0f, 1.0f };     // Light Position

GLfloat pavimentoDiffuse[] = { 1.0f, 1.0f, 1.0f, 1.0f };    // Diffuse Light Values
GLfloat pavimentoSpecular[] = { 1.0, 1.0, 1.0, 1.0 };       // Specular Light Values
GLfloat pavimentoPosition[] = { 0.0f, 0.0f, 0.0f, 1.0f };   // Light Position

GLfloat focoDiffuse[] = { 1.0f, 1.0f, 1.0f, 1.0f };         // Diffuse Light Values
GLfloat focoSpecular[] = { 1.0, 1.0, 1.0, 1.0 };            // Specular Light Values
GLfloat focoPosition[] = { 0.0f, 0.0f, 0.0f, 1.0f };        // Light Position

GLfloat puestoDiffuse[] = { 1.0f, 1.0f, 1.0f, 1.0f };       // Diffuse Light Values
GLfloat puestoSpecular[] = { 1.0, 1.0, 1.0, 1.0 };          // Specular Light Values
GLfloat puestoPosition[] = { 0.0f, 0.0f, 0.0f, 1.0f };      // Light Position
```

Objetos

Sobre la función llamada ciudad se comienza a construir la esfera que actuará como el sol y como la luna, se activa el material con glMaterialfv y sus parámetros, con glutSolidSphere se crea la esfera.
Para el pavimento es el mismo cuerpo, se añade material, texturizado y se crea la figura.

```
void ciudad ()
{

    ////////////////////////////////////// Sol-Luna //////////////////////////////////
    glPushMatrix(); //Sol
    glTranslatef(50, 70, -50);
    glMaterialfv(GL_FRONT, GL_DIFFUSE, SunDiffuse); // Material Sol-Luna
    glMaterialfv(GL_FRONT, GL_SPECULAR, SunSpecular); // Material Sol-Luna
    glDisable(GL_LIGHTING);
    glColor3f(1.0, 1.0, 1.0);   //Sol-Luna Blanco
    glutSolidSphere(7.0, 12, 12);  //Draw Sun (radio,H,V);
    glEnable(GL_LIGHTING);
    glPopMatrix(); //Fin Sol - Luna
    ////////////////////////////////////////////////////////////////////////////////

    ////////////////////////////// Pavimento ////////////////////////////////////////

    glPushMatrix(); // Pavimento
    glTranslatef(-47, 0.0, -19);
    glRotatef(90, 0, 1, 0);
    glScalef(200, 0.1, 250);
    glMaterialfv(GL_FRONT, GL_DIFFUSE, pavimentoDiffuse); // Material Pavimento
    glMaterialfv(GL_FRONT, GL_SPECULAR, pavimentoSpecular); // Material Pavimento
    glDisable(GL_LIGHTING);
    glColor3f(0.200, 0.200, 0.200); //Pavimento Oscuro
    fig3.prisma2(text4.GLindex, 0);
    glEnable(GL_LIGHTING);
    glPopMatrix();

    //////////////////////////  Fin Pavimento ////////////////////////////////////////
```

Montaña

Aquí se crea la montaña rusa, se declaran los materiales, color verde, su textura y la creación del prisma.

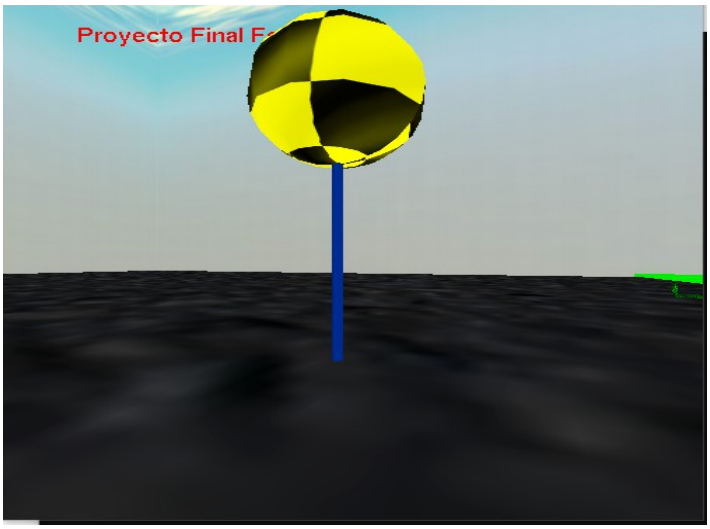
```
////////////////////////////// Montaña Rusa ////////////////////////////////////

    glPushMatrix(); ////////////////// Montaña Rusa tubo 1
    glTranslatef(20.0, 1.5, -20.0);
    glRotatef(90, 0, 1, 0);
    glScalef(7.0, 0.2, 0.1);
    glMaterialfv(GL_FRONT, GL_DIFFUSE, montanaDiffuse); // Material Montaña
    glMaterialfv(GL_FRONT, GL_SPECULAR, montanaSpecular); // Material Montaña
    glDisable(GL_LIGHTING);
    glColor3f(0, 1, 0); // Montaña luz verde
    fig8.prisma(1, 4, 1, text7.GLindex);
    glEnable(GL_LIGHTING);
    glPopMatrix();

    glPushMatrix(); ////////////////// Montaña Rusa línea perpendicular 1
    glTranslatef(19, 1.5, -33.0);
    glRotatef(90, 1, 0, 0);
    glScalef(0.5, 0.2, 0.1);
    glMaterialfv(GL_FRONT, GL_DIFFUSE, montanaDiffuse); // Material Montaña
    glMaterialfv(GL_FRONT, GL_SPECULAR, montanaSpecular); // Material Montaña
    glDisable(GL_LIGHTING);
    glColor3f(0, 1, 0); // Montaña luz verde
    fig8.prisma(1, 4, 1, text7.GLindex);
    glEnable(GL_LIGHTING);
    glPopMatrix();

    glPushMatrix(); ////////////////// Montaña Rusa línea perpendicular 2
    glTranslatef(19, 1.5, -30.0);
    glRotatef(90, 1, 0, 0);
    glScalef(0.5, 0.2, 0.1);
    glMaterialfv(GL_FRONT, GL_DIFFUSE, montanaDiffuse); // Material Montaña
    glMaterialfv(GL_FRONT, GL_SPECULAR, montanaSpecular); // Material Montaña
    glDisable(GL_LIGHTING);
    glColor3f(0, 1, 0); // Montaña luz verde
    fig8.prisma(1, 4, 1, text7.GLindex);
    glEnable(GL_LIGHTING);
    glPopMatrix();
```

Aquí se presenta un poco del resultado de construir prisma a prisma la montaña

Poste de luz

A continuación se presenta la creación de los postes de luz y su resultado

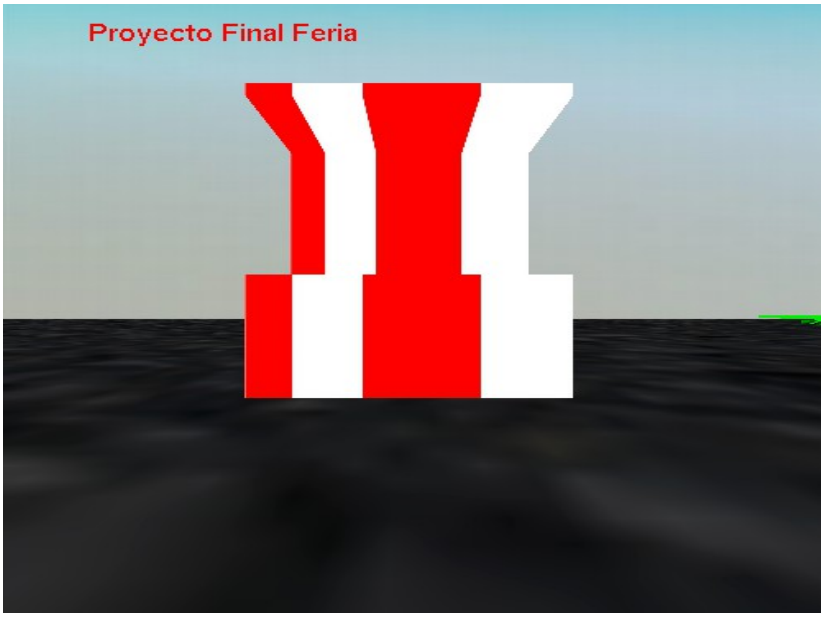
```
 ///////////////// Postes de luz ////////////////////////////////////////

glPushMatrix(); ////////////// Poste
glTranslatef(10, 3, 0.6);
glRotatef(90, 0, 0, 1);
glScalef(1.0, 0.1, 0.1);
glMaterialfv(GL_FRONT, GL_DIFFUSE, montanaDiffuse); // Material Montaña
glMaterialfv(GL_FRONT, GL_SPECULAR, montanaSpecular); // Material Montaña
glDisable(GL_LIGHTING);
glColor3f(0.8, 0.8, 0.8);
fig11.prisma(1, 4, 1, text10.GLindex);
glEnable(GL_LIGHTING);
glPopMatrix();

glPushMatrix(); //Foco
glTranslatef(10, 5, 0.6);
glMaterialfv(GL_FRONT, GL_DIFFUSE, focoDiffuse); // Material Foco
glMaterialfv(GL_FRONT, GL_SPECULAR, focoSpecular); // Material Foco
glDisable(GL_LIGHTING);
glColor3f(1, 1, 0); //Amarillo
fig11.esfera(1, 12, 12, text9.GLindex);
glEnable(GL_LIGHTING);
glPopMatrix(); //Fin Foco
```



Puesto

El código de los puestos es similar a los demás objetos, aquí el código y su resultado.

```
////////////////////////////////////// Puestos Izquierdos //////////////////////////////////////

glPushMatrix(); //Puesto 1
glTranslatef(-10, 2, -15);
glScalef(1.0, 1.0, 1.0);
glMaterialfv(GL_FRONT, GL_DIFFUSE, puestoDiffuse); // Material Foco
glMaterialfv(GL_FRONT, GL_SPECULAR, puestoSpecular); // Material Foco
glDisable(GL_LIGHTING);
glColor3f(1, 1, 1); //Blanco
fig12.prisma(2, 4, 2, text11.GLindex);
glEnable(GL_LIGHTING);
glPopMatrix(); //Fin Puesto

glPushMatrix(); //Puesto 2
glTranslatef(-10, 6, -15);
glScalef(1.0, 0.1, 1.0);
glMaterialfv(GL_FRONT, GL_DIFFUSE, puestoDiffuse); // Material Foco
glMaterialfv(GL_FRONT, GL_SPECULAR, puestoSpecular); // Material Foco
glDisable(GL_LIGHTING);
glColor3f(1, 1, 1); //Blanco
fig12.prisma(2, 4, 2, text11.GLindex);
glEnable(GL_LIGHTING);
glPopMatrix(); //Fin Pesto

glPushMatrix(); //Puesto 3
glTranslatef(-10, 4.5, -16.0);
glScalef(1.0, 1.5, 0.1);
glMaterialfv(GL_FRONT, GL_DIFFUSE, puestoDiffuse); // Material Foco
glMaterialfv(GL_FRONT, GL_SPECULAR, puestoSpecular); // Material Foco
glDisable(GL_LIGHTING);
glColor3f(1, 1, 1); //Blanco
fig12.prisma(2, 4, 2, text11.GLindex);
glEnable(GL_LIGHTING);
glPopMatrix(); //Fin Foco
```



Árbol

Para generar un árbol con transparencia es necesario crearlo por planos, para darle un toque tridimensional más adecuado y estético, se declaran sus coordenadas con los vértices del plano y se le añade profundidad.

```
void arbol_blend()
{
    glPushMatrix();
        glDisable(GL_LIGHTING);
        glEnable(GL_BLEND);     // Turn Blending On
        //glDisable(GL_DEPTH_TEST);   // Turn Depth Testing Off
        glBlendFunc(GL_SRC_ALPHA,GL_ONE_MINUS_SRC_ALPHA);
        glBindTexture(GL_TEXTURE_2D, tree.GLindex);
        glBegin(GL_QUADS); //plano
            glColor3f(1.0, 1.0, 1.0);
            glNormal3f( 0.0f, 0.0f, 1.0f);
            glTexCoord2f(0.0f, 0.0f); glVertex3f(-10.0, 0.0, 0.0);
            glTexCoord2f(1.0f, 0.0f); glVertex3f(10.0, 0.0, 0.0);
            glTexCoord2f(1.0f, 1.0f); glVertex3f(10.0, 20.0, 0.0);
            glTexCoord2f(0.0f, 1.0f); glVertex3f(-10.0, 20.0, 0.0);
        glEnd();
    glPopMatrix();

    glPushMatrix();////////////////////////////////////////////////////////////////////
        glRotatef(45, 0, 1, 0);
        glBegin(GL_QUADS); //plano
        glColor3f(0.9, 0.9, 0.0);
            //glColor3f(1.0, 1.0, 1.0);
            glNormal3f( 0.0f, 0.0f, 1.0f);
            glTexCoord2f(0.0f, 0.0f); glVertex3f(-10.0, 0.0, 0.0);
            glTexCoord2f(1.0f, 0.0f); glVertex3f(10.0, 0.0, 0.0);
            glTexCoord2f(1.0f, 1.0f); glVertex3f(10.0, 20.0, 0.0);
            glTexCoord2f(0.0f, 1.0f); glVertex3f(-10.0, 20.0, 0.0);
        glEnd();
    glPopMatrix();
```

Texturas

Esta parte del código muestra la declaración de las texturas que son llamadas desde la ubicación donde se encuentran los bmp's o los tga's, cual sea el caso, se carga la imagen, la textura y se implementa.

```
//////////////////////////////////////////////Montaña Rusa///////
text7.LoadBMP("montana_rusa/tubo_amarillo.bmp");        /////////
text7.BuildGLTexture();                                 /////////
text7.ReleaseImage();                                   ////////
                                                        ///////
text8.LoadBMP("montana_rusa/apoyo.bmp");               ////////
text8.BuildGLTexture();                              ////////////
text8.ReleaseImage();                                ////////////
                                                        ///////
text9.LoadBMP("montana_rusa/bandera_carrera.bmp");      //////
text9.BuildGLTexture();                              //////////
text9.ReleaseImage();                                //////////
////////////////////////////////////////////////////////////////


//////////////////////////////// Feria/////////
text10.LoadBMP("montana_rusa/tubo_azul.bmp");////
text10.BuildGLTexture();                    /////
text10.ReleaseImage();                    ///////
                                          ///////
text11.LoadBMP("montana_rusa/puesto.bmp"); ////
text11.BuildGLTexture();                 ////////
text11.ReleaseImage();                  //////////
                                        //////////
text4.LoadBMP("montana_rusa/pavimento_1.bmp");//
text4.BuildGLTexture();                 ////////
text4.ReleaseImage();                   //////////
                                        ////////
text12.LoadBMP("montana_rusa/puesto2.bmp"); ////
text12.BuildGLTexture();                ////////
text12.ReleaseImage();                   ////////
////////////////////////////////////////////////
```

KeyFrames

En estas variables se guardan los key frames para poder hacer una animación automática sin intervención del teclado.

```
' '

KeyFrame[0].posX = 0;
KeyFrame[0].posY = 2.5;
KeyFrame[0].posZ = -3.5;
KeyFrame[0].rotRodIzq = 0;
KeyFrame[0].giroMonito = 0;
KeyFrame[0].movBrazoDer = 0;

KeyFrame[1].posX = 20;
KeyFrame[1].posY = 2.5;
KeyFrame[1].posZ = -3.5;
KeyFrame[1].rotRodIzq = 60;
KeyFrame[1].giroMonito = -90.0;
KeyFrame[1].movBrazoDer = 40;

KeyFrame[2].posX = 20;
KeyFrame[2].posY = 2.5;
KeyFrame[2].posZ = 4.0;
KeyFrame[2].rotRodIzq = -20.0;
KeyFrame[2].giroMonito = 0;
KeyFrame[2].movBrazoDer = 0;

KeyFrame[3].posX = 20;
KeyFrame[3].posY = 2.5;
KeyFrame[3].posZ = 4.0;
KeyFrame[3].rotRodIzq = 40.0;
KeyFrame[3].giroMonito = 45;
KeyFrame[3].movBrazoDer = 90;

KeyFrame[4].posX = 20;
KeyFrame[4].posY = 2.5;
KeyFrame[4].posZ = 4.0;
KeyFrame[4].rotRodIzq = -40.0;
KeyFrame[4].giroMonito = -45;
KeyFrame[4].movBrazoDer = -90;
//NEW//////////////////NEW//////////////////NEW//////////////////
```

Animación

Una de las partes más importantes del código es la de la animación.

Se tiene que para poder usar la animación se declara con sentencias de ciclo if, cuando se entra a un if se activa el movimiento incrementando las variables para poder mover el objeto, en este caso el carrito.

```
void animacion()
{
    fig3.text_izq-= 0.01;
    fig3.text_der-= 0.01;
    if(fig3.text_izq<-1)
        fig3.text_izq=0;
    if(fig3.text_der<0)
        fig3.text_der=1;


    //Movimiento del coche
    if(g_fanimacion)
    {
        if(g_avanza)
        {
            movKitZ +=1.0;
            rotTires -= 20;
            if(movKitZ>130)
                g_avanza = false;
        }
        else
        {
            movKitZ -=1.0;
            rotTires += 20;
            if(movKitZ<0)
                g_avanza = true;
        }
    }
```

Circuito

Para activar el recorrido del circuito se hace por medio de un valor booleano, entra a la condición y sale cuando es mayor que la variable de alguno de los tres ejes del sistema, entonces se desactiva el recorrido actual y se activa el siguiente, y así se va hasta volver al estado en el que se comenzó.

```
if(circuito)
{
    if(recorrido1)
    {

        movKitZ = movKitZ + 1;
        rotKit = (45, 1, 0, 1);

        if(movKitZ>13)
        {
            recorrido1 = false;
            recorrido2 = true;
        }
    }
    if(recorrido2)
    {
        rotKit = 270;
        movKitY = movKitY + 2;
        movKitZ = movKitZ + 2;
        movKitX = movKitX - 0.9;


        if(movKitY > 88)
        {
            recorrido2 = false;
            recorrido3 = true;

        }
    }
```

Para realizar el movimiento del monito. Se utiliza la interpolación con las variables de los incrementos y las posiciones para poder animarlo.

En esta parte se usan las teclas para poder usar las diferentes animaciones, mover la cámara y activar y desactivar noche.

```
//Movimiento del monito
if (play)
{
    if (i_curr_steps >= i_max_steps) //end of animation between frames?
    {
        playIndex++;
        if (playIndex>FrameIndex - 2)    //end of total animation?
        {
            printf("termina anim\n");
            playIndex = 0;
            play = false;
        }
        else //Next frame interpolations
        {
            i_curr_steps = 0; //Reset counter
                              //Interpolation
            interpolation();
        }
    }
    else
    {
        //Draw animation
        posX += KeyFrame[playIndex].incX;
        posY += KeyFrame[playIndex].incY;
        posZ += KeyFrame[playIndex].incZ;

        rotRodIzq += KeyFrame[playIndex].rotInc;
        giroMonito += KeyFrame[playIndex].giroMonitoInc;
        movBrazoDer += KeyFrame[playIndex].movBrazoDerInc;

        i_curr_steps++;
    }
}
```

Teclas

La i e I se utilizan para activar la animación del recorrido del carrito, con K se almacena el Key Frame para hacer una animación.

```
case 'O':       //
case 'o':
        g_fanimacion^= true; //Activamos/desactivamos la animacíon
        circuito = false;
    break;

case 'i':       //
case 'I':
        circuito^= true; //Activamos/desactivamos la animacíon
        g_fanimacion = false;
    break;

case 'k':       //
case 'K':
    if (FrameIndex<MAX_FRAMES)
    {
        saveFrame();
    }

    break;

case 'l':
case 'L':
    if (play == false && (FrameIndex>1))
    {

        resetElements();
        //First Interpolation
        interpolation();

        play = true;
        playIndex = 0;
        i_curr_steps = 0;
```

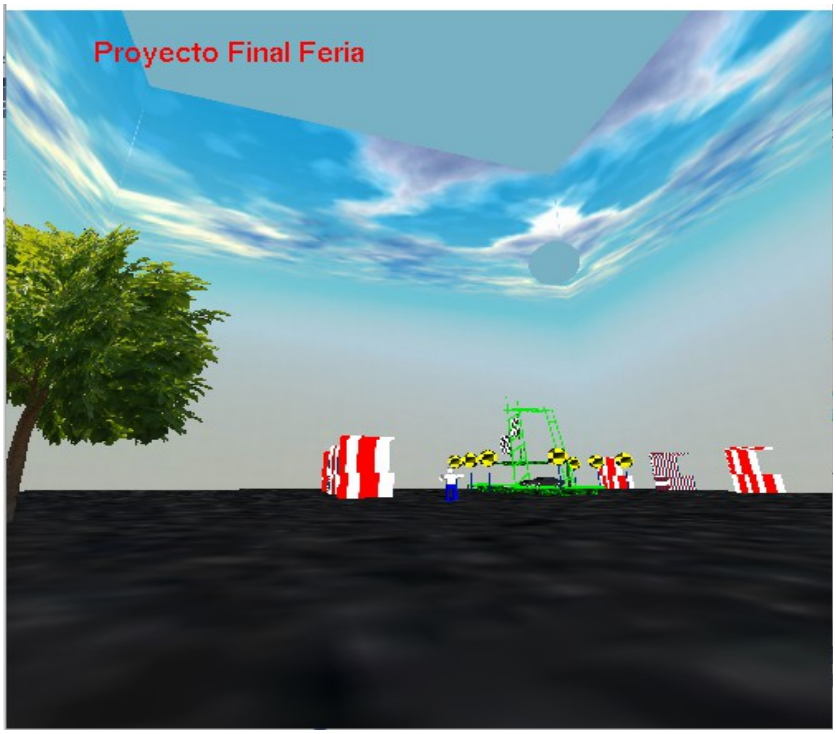
Noche-Día

Para la tecla n se asigno el habilitar la noche apagando el escenario y encendiendo el brillo de las texturas del carro y el monito.

Aquí el resultado:

```
case 'n':

    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    glEnable(GL_LIGHT1);
    glEnable ( GL_COLOR_MATERIAL );
    glEnable(GL_CULL_FACE);
    glCullFace(GL_BACK);
    break;

case 'm':

    glDisable(GL_LIGHTING);
    glDisable(GL_LIGHT0);
    glDisable(GL_LIGHT1);
    glDisable(GL_COLOR_MATERIAL);
    glDisable(GL_CULL_FACE);
    glCullFace(GL_BACK);
    break;

case 27:        // Cuando Esc es presionado...
    exit ( 0 );   // Salimos del programa
    break;
default:        // Cualquier otra
    break;
```

Para desactivarlo se eligió la tecla m y poder regresar al día, quedando como resultado:



Y con la tecla escape se termina el programa.