DOCUMENTACION PROYECTO FINAL

Graficación Por Computadora

9 DE DICIEMBRE DE 2025

URENA ANDRADE CARLOS ALBERTO
S22002224

Contenido

Introduccion

En el presente proyecto se verá una animación de un videojuego la cual esta diseñada en OpenGL en el lenguaje de programación C, en dicho proyecto se aplico lo visto durante la EE de Graficacion Por Computadora

Requisitos

- Tener OpenGL descargado en tu dispositivo.
- Tener memoria suficiente para poder descargar el proyecto.
- Ejecutarlo en consola

Librerías

```
#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glut.h>
#include<math.h>
#include<string.h>
#include<stdlib.h>
#define STB_IMAGE_IMPLEMENTATION
#include "stb_image.h"
```

Gl/gl.h, GL/glu.h, GL/glut.h fueron utilizadas para poder usar las funciones de OpenGL

String.g usada para copiar cadenas de textos y poder almacenar dichos valores

Stb_image.h usada para agregar texturas al proyecto

Estructuras Utilizadas

- Listas simples
- Colas
- Arboles binarios
- Pilas

Prototipos de funciones

```
//Prototipos de funciones
void enqueue(struct Cola *cola, int tam, int x, int y, int escena,int textura);
void insertarcamino(struct Camino *cola, int x, int y);
void ListinsertarParte(struct PartePersonaje **lista, int x, int y);
void swapPartes(struct NodoArbol *nodo1, struct NodoArbol *nodo2);
void insertarNodoArbol(struct NodoArbol **raiz, float color[3], char nombre[]);
void dibujarpersonaje(struct NodoArbol *raiz);
void iniciarpersonaje(struct NodoArbol **raiz);
void initobstaculos(struct Cola *cola);
void dibujarobstaculos(struct Cola *cola);
void CrearCamino(struct Camino *camino);
static void reshape01(int w, int h);
void texto(char *texto,int x, int y, float color[],void* font);
void display();
void menu();
void pausa();
static void init01(void);
GLuint CargarTextura(const char *ruta);
void Animacion(int num) ;
void teclado(unsigned char tecla,int x, int y);
```

InsertarCamino, ListInsertarParte, InsertarNodoArbol, enqueue son funciones para insertar nodos en las diferentes estructuras creadas

swapPartes es una función fundamental para la animación del personaje ya que intercambia las listas de los puntos del personaje para que de la ilusión de caminar

dibujarpersonaje, dibujarobstaculos son funciones que sirven para dibujar al personaje o a los diferentes obstáculos de las diferentes escenas

iniciarpersonaje, initobstaculos, crearcamino Son funciones que guardan los datos necesarios desde un inicio en las diferentes estructuras de datos para poder crear la animación.

Texto es una función que sirve para escribir en las diferentes ventanas emergentes

Display, menu y pausa don las diferentes ventanas que se mostraran a lo largo del programa las cuales el usuario podrá ver

CargarTextura es la función que me ayuda a procesar y cargar las texturas y poder guardarlas en variables para poder usarlas después

Animación es la función que se repite y me ayuda a hacer la animación completa del programa

Teclado es la función que me dice que sucede si seleccionas ciertas teclas y me ayuda a cambiar de ventanas

Variables Globales

```
//Variables Globales
struct NodoArbol *Raiz = NULL;
struct NodoArbol *Aux = NULL;
struct NodoArbol *Aux2 = NULL;
struct camino *caminoaux;
struct Cola Cola;
struct Camino Camino;
int texturapiso;
int texturafondo;
int caja;
int gujero;
int tesoro;
int pasoAnim = 0;
int escena = 0;
int ventana = 0;
```

Las variables globales que utilice en el programa principalmente fueron estructuras, contadores o las variables para guardar las texturas

Raiz, Aux, Aux2 son arboles binarios, raíz es el árbol principal, este guarda la forma estática del personaje, Aux y Aux 2 contienen variantes de ciertas partes del cuerpo del personaje que se van intercambiando por medio de la función swappartes para dar la ilusión de movimiento

Caminioaux es un puntero para ayudarme a recorrer el camino que tiene que tomar mi personaje sin tener que modificar la lista en la que se encuentra el camino

Cola, Camino ambas son colas solo que de diferentes structs, cola es una cola que guarda los diferentes obstáculos de los diferentes escenarios y camino guarda las coordenadas del camino que tiene que recorrer el personaje

pasoAnim, escena, ventana son contadores para saber en cual de las variantes estamos, en que paso de la animación, en que escena o en que ventana estamos actualmente

texturapiso, texturafondo, caja, gujero, tesoro son las variables en las que guardo las referencias a las texturas para poder utilizarlas después

Implementación

```
int main(int argc, char** argv)
{
    glutInit(&argc, argv); // inicializa las bibliotecas de glut

    // configuración del modo de visualización de la pantalla
    glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);
    glutInitWindowSize(900, 700);
    glutInitWindowPosition(300, 50);
    glutCreateWindow("La Aventura de GLEEP");

    init01();
    iniciarpersonaje(&Raiz);
    initobstaculos(&Cola);
    CrearCamino(&Camino);
    caminoaux = Camino.inicio;

    glutDisplayFunc(menu);
    glutReshapeFunc(reshape01);
    glutKeyboardFunc(teclado);
    glutMainLoop(); // loop principal
    return 0;
}
```

En el main después de crear la ventana inicializo una serie de variables utilizando la función init01()

```
static void init01(void)
{
    glClearColor (1.0, 1.0, 1.0, 0.0);  // Limpiar a color RGB A
    glShadeModel (GL_FLAT); // El valor default es GL_SMOOTH
    glEnable(GL_BLEND);
    glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);

    Cola.inicio = NULL;
    Cola.final = NULL;
    Camino.inicio = NULL;
    Camino.final = NULL;
    texturapiso = CargarTextura("vecteezy_stone-tiles-texture-in-cartoon-style_3678912.jpg");
    texturafondo = CargarTextura("png-clipart-cartoon-drawing-sky-cloud-clouds-cartoon-blue-atmosphere-thumbnail.png");
    caja = CargarTextura("g5q1_cfw8_210729-removebg-preview.png");
    gujero = CargarTextura("agujero.png");
    tesoro = CargarTextura("cofre.png");
}
```

1 funcion Init01()

Posteriormente cargo los datos con las funciones **iniciarpersonaje**, **initobstaculos**, **crearcamino**

```
void initobstaculos(struct Cola *cola)
{
    //Escena 1
    enqueue(cola, 60, 350,130,0,caja);
    enqueue(cola, 60, 410,130,0,caja);
    enqueue(cola, 60, 410,190,0,caja);
    enqueue(cola, 60, 470,130,0,caja);
    enqueue(cola, 60, 470,190,0,caja);
    enqueue(cola, 60, 470,250,0,caja);
    enqueue(cola, 60, 530,130,0,caja);
    enqueue(cola, 60, 530,190,0,caja);
    enqueue(cola, 60, 530,250,0,caja);
    enqueue(cola, 60, 530,310,0,caja);
    enqueue(cola, 350,735,-75,0,gujero);
    enqueue(cola, 60, 710,370,0,caja);
    enqueue(cola, 60, 770,370,0,caja);
    enqueue(cola, 60, 830,370,0,caja);

    //Escena 2
    enqueue(cola, 900,450,-350,1,gujero);
    enqueue(cola, 60, 120,340,1,caja);
    enqueue(cola, 60, 180,340,1,caja);
    enqueue(cola, 60, 240,340,1,caja);
    enqueue(cola, 60, 400,280,1,caja);
    enqueue(cola, 60, 560,360,1,caja);
    enqueue(cola, 60, 790,200,1,caja);
    enqueue(cola, 60, 850,200,1,caja);

    enqueue(cola, 900,450,-350,2,gujero);
    enqueue(cola, 60, 80,300,2,caja);
    enqueue(cola, 60, 140,300,2,caja);
    enqueue(cola, 60, 200,300,2,caja);
    enqueue(cola, 60, 260,300,2,caja);
    enqueue(cola, 60, 320,300,2,caja);
    enqueue(cola, 60, 380,300,2,caja);
    enqueue(cola, 60, 440,300,2,caja);
    enqueue(cola, 60, 500,300,2,caja);
    enqueue(cola, 60, 560,300,2,caja);
    enqueue(cola, 60, 620,300,2,caja);
    enqueue(cola, 60, 680,300,2,caja);
    enqueue(cola, 60, 740,300,2,caja);
    enqueue(cola, 60, 800,300,2,caja);
    enqueue(cola, 60, 800,360,2,tesoro);


}
```

3 InitObstaculos()

```
void iniciarpersonaje(struct NodoArbol **raiz)
{
    float aux[3]={0.8,1.0,0.29};
    insertarNodoArbol(raiz,aux,"Brazo");
    ListinsertarParte(&((*raiz)->parte),11,14);
    ListinsertarParte(&((*raiz)->parte),11,10);
    ListinsertarParte(&((*raiz)->parte),13,10);
    ListinsertarParte(&((*raiz)->parte),13,13);

    insertarNodoArbol(&((*raiz)->izq),aux,"Cuerpo");
    ListinsertarParte(&((*raiz)->izq->parte),8,21);
    ListinsertarParte(&((*raiz)->izq->parte),8,5);
    ListinsertarParte(&((*raiz)->izq->parte),9,5);
    ListinsertarParte(&((*raiz)->izq->parte),9,2);
    ListinsertarParte(&((*raiz)->izq->parte),10,1);
    ListinsertarParte(&((*raiz)->izq->parte),11,1);
    ListinsertarParte(&((*raiz)->izq->parte),12,2);
    ListinsertarParte(&((*raiz)->izq->parte),12,4);
    ListinsertarParte(&((*raiz)->izq->parte),13,5);
    ListinsertarParte(&((*raiz)->izq->parte),15,5);
    ListinsertarParte(&((*raiz)->izq->parte),16,6);
    ListinsertarParte(&((*raiz)->izq->parte),17,6);
    ListinsertarParte(&((*raiz)->izq->parte),18,7);
    ListinsertarParte(&((*raiz)->izq->parte),19,8);
    ListinsertarParte(&((*raiz)->izq->parte),19,9);
    ListinsertarParte(&((*raiz)->izq->parte),20,9);
    ListinsertarParte(&((*raiz)->izq->parte),20,13);
    ListinsertarParte(&((*raiz)->izq->parte),24,13);
    ListinsertarParte(&((*raiz)->izq->parte),25,14);
    ListinsertarParte(&((*raiz)->izq->parte),25,15);
    ListinsertarParte(&((*raiz)->izq->parte),24,16);
    ListinsertarParte(&((*raiz)->izq->parte),21,16);
    ListinsertarParte(&((*raiz)->izq->parte),20,17);
    ListinsertarParte(&((*raiz)->izq->parte),20,18);
    ListinsertarParte(&((*raiz)->izq->parte),23,18);
    ListinsertarParte(&((*raiz)->izq->parte),23,19);
    ListinsertarParte(&((*raiz)->izq->parte),25,19);
    ListinsertarParte(&((*raiz)->izq->parte),25,21);
    ListinsertarParte(&((*raiz)->izq->parte),24,22);
    ListinsertarParte(&((*raiz)->izq->parte),18,22);
    ListinsertarParte(&((*raiz)->izq->parte),18,23);

    insertarNodoArbol(&((*raiz)->izq->izq),aux,"Pieizq");
    ListinsertarParte(&((*raiz)->izq->izq->parte),16,6);
    ListinsertarParte(&((*raiz)->izq->izq->parte),16,2);
    ListinsertarParte(&((*raiz)->izq->izq->parte),17,1);
    ListinsertarParte(&((*raiz)->izq->izq->parte),18,1);
    ListinsertarParte(&((*raiz)->izq->izq->parte),19,2);
    ListinsertarParte(&((*raiz)->izq->izq->parte),19,8);
```

***2** IniciarPersonaje()*

```
void CrearCamino(struct Camino *camino)
{
    int y=0;
    for(int x = 0; x <= 900; x+=3)
    {   if(x == 210)
            y = 10;

        if((x > 210 && y < 60)||(x > 270 && y < 120) || (x > 330 && y < 180) || (x > 390 && y < 240))
            y += 6;

        if(x < 645)
        {
            if(x > 560 && y > 300)
            {
                y -= 3;
            }else
                if(x > 500 && y < 360)
                    y += 6;
        }

        if(x > 800 && y < 400)
            y += 5;

        insertarcamino(camino, x, y);
    }

    for(int x = 0; x<=900;x += 3)
    {   if(x > 0 && x < 200 && y > 270)
            y -=6;

        if(x > 200 && x < 220)
            y+= 6;

        if(x > 220 &&  x < 300)
            y -=4;

        if(x >370 && x < 520 && y < 280)
            y+=3;

        if(x > 520 && x < 540)
            y+= 6;

        if(x > 540 &&  x < 730)
            y -=3;

        if(x > 800 &&  x < 900)
            y += 3;

        insertarcamino(camino, x, y);
    }
```

4 Funcion CrearCamino()

Una vez creado e inicializado todo empieza la animación con la función animación

```
void Animacion(int num)
{
    if(ventana != 2)
    {
    switch(pasoAnim) {
        case 0:

            break;

        case 1:
            swapPartes(Raiz->izq->izq, Aux->izq);
            swapPartes(Raiz, Aux->der);
            break;

        case 2:
            swapPartes(Raiz->izq, Aux2->izq);
            swapPartes(Raiz, Aux2->der);
            break;
    }

    pasoAnim = (pasoAnim + 1) % 3;

    switch(pasoAnim) {
        case 0:
            break;

        case 1:
            swapPartes(Raiz->izq->izq, Aux->izq);
            swapPartes(Raiz, Aux->der);
            break;

        case 2:
            swapPartes(Raiz->izq, Aux2->izq);
            swapPartes(Raiz, Aux2->der);
            break;
    }
    caminoaux = caminoaux->sig;

    if(caminoaux == NULL)
        caminoaux = Camino.inicio->sig;

    if(caminoaux->coord.x == 900)
        escena =    (escena + 1) % 3;

    glutPostRedisplay();
    glutTimerFunc(200, Animacion, 0);
    }else
        glutTimerFunc(200, Animacion, 0);
}
```

La función verifica en que ventana estamos para saber si continuar o pausar la animación, si la animación continua verifica en que paso de la animación del personaje estamos y restaura la las listas al estado base, aumenta a la siguiente animación cambia las listas para redibujar el siguiente frame y aumenta a camino auxiliar al siguiente el cual sirve para hacer un gltranslate y mover al personaje y al final vuelve a llamar a la animación.

```
void display()
{
    glClear(GL_COLOR_BUFFER_BIT); // Limpia el búfer de color

    glColor3f(1.0, 1.0, 1.0);
    glBindTexture(GL_TEXTURE_2D, texturafondo);
    glEnable(GL_TEXTURE_2D);
    glBegin(GL_QUADS);
        glTexCoord2f(0,0); glVertex2f(0,700);
        glTexCoord2f(1,0); glVertex2f(900,700);
        glTexCoord2f(1,1); glVertex2f(900,0);
        glTexCoord2f(0,1); glVertex2f(0,0);
    glEnd();
    glBindTexture(GL_TEXTURE_2D, 0);
    glDisable(GL_TEXTURE_2D);

    glColor3f(1.0, 1.0, 1.0);
    glBindTexture(GL_TEXTURE_2D, texturapiso);
    glEnable(GL_TEXTURE_2D);
    glBegin(GL_QUADS);
        glTexCoord2f(0,0); glVertex2f(0,0);
        glTexCoord2f(1,0); glVertex2f(900,0);
        glTexCoord2f(1,1); glVertex2f(900,100);
        glTexCoord2f(0,1); glVertex2f(0,100);
    glEnd();
    glBindTexture(GL_TEXTURE_2D, 0);
    glDisable(GL_TEXTURE_2D);

    dibujarobstaculos(&Cola);

    glPushMatrix();
    glTranslatef(caminoaux->coord.x, caminoaux->coord.y, 0);
    dibujarpersonaje(Raiz);
    glPopMatrix();

    // intercambio de buffers (porque usamos GLUT_DOUBLE)
    glutSwapBuffers();
}
```

Esta es la función para dibujar, lo que hace es primeramente dibujar el fondo con el suelo que ambas son texturas, posteriormente dibuja los obstáculos correspondientes a la escena y de ahí hace glpushmatrix para guardar la matriz antes de realizar la transformación con gltranslate para poder dibujar al personaje en la posición que se debería de mover y al finalizar esto regresa a la normalidad a la matriz con glpopmatrix.

Manual de Usuario

```
C:\Users\carli\OneDrive\Escritorio\GC_UrenaCarlos_ProyectoFinal>gcc GC_CAUA_ProyectoFinal.c -o pelicula.exe -
I"C:\MinGW\include" -L"C:\MinGW\lib" -lfreeglut -lopengl32 -lglu32

C:\Users\carli\OneDrive\Escritorio\GC_UrenaCarlos_ProyectoFinal>pelicula.exe
Textura cargada: vecteezy_stone-tiles-texture-in-cartoon-style_3678912.jpg - 1920x1761, canales: 3
ID de textura generada: 1
Textura cargada: png-clipart-cartoon-drawing-sky-cloud-clouds-cartoon-blue-atmosphere-thumbnail.png - 348x218
, canales: 3
ID de textura generada: 2
Textura cargada: g5q1_cfw8_210729-removebg-preview.png - 428x428, canales: 4
ID de textura generada: 3
Textura cargada: agujero.png - 337x237, canales: 4
ID de textura generada: 4
Textura cargada: cofre.png - 512x512, canales: 4
ID de textura generada: 5
```

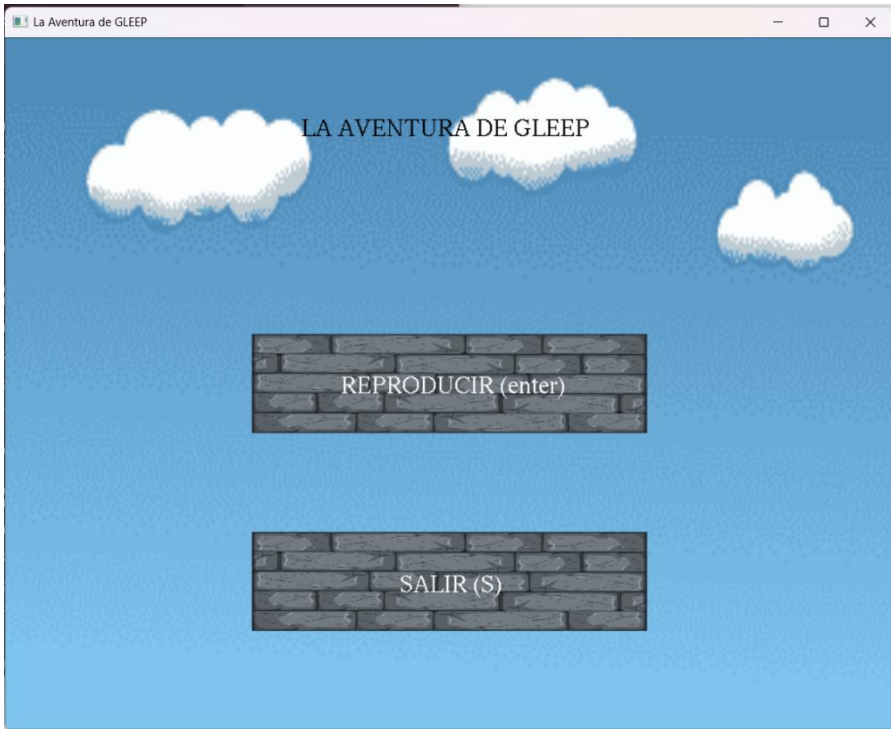
Abres la terminal y pones la ruta donde se encuentra tu proyecto y ejecutas la siguiente línea de código:

gcc GC_CAUA_ProyectoFinal.c -o pelicula.exe -I"C:\MinGW\include" -L"C:\MinGW\lib" -lfreeglut -lopengl32 -lglu32

si no arrojo errores ejecútalo con el siguiente comando:

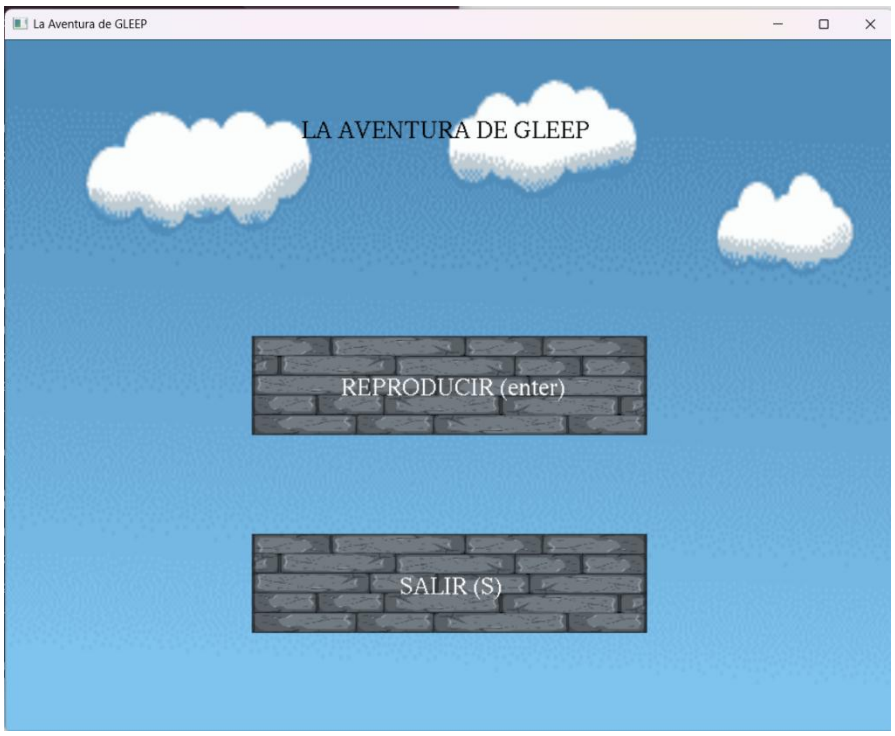
pelicula.exe

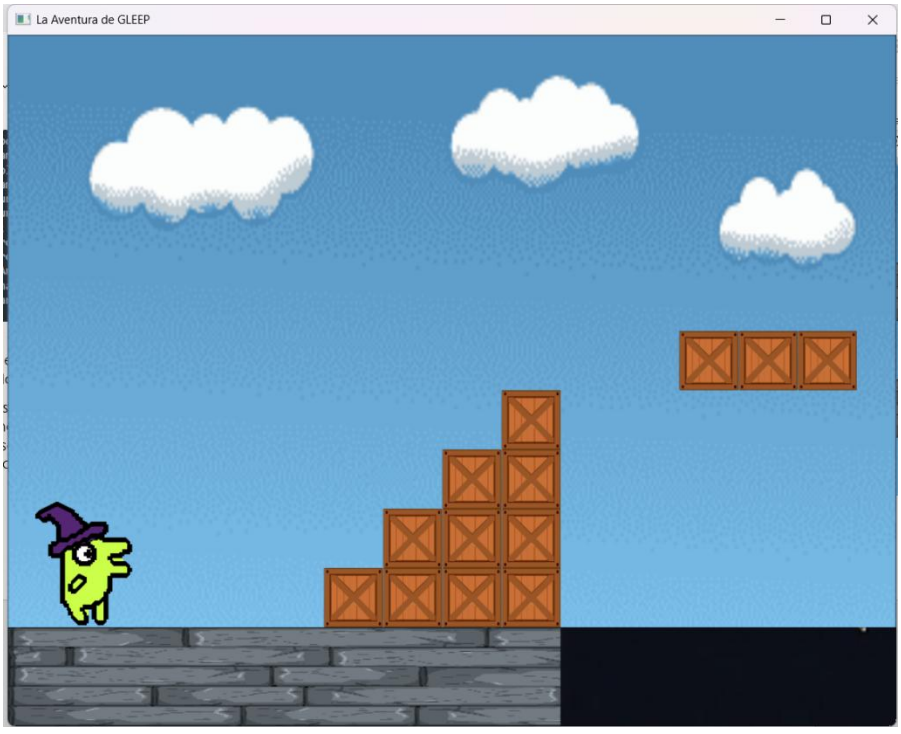
y debería de aparecerte la siguiente imagen

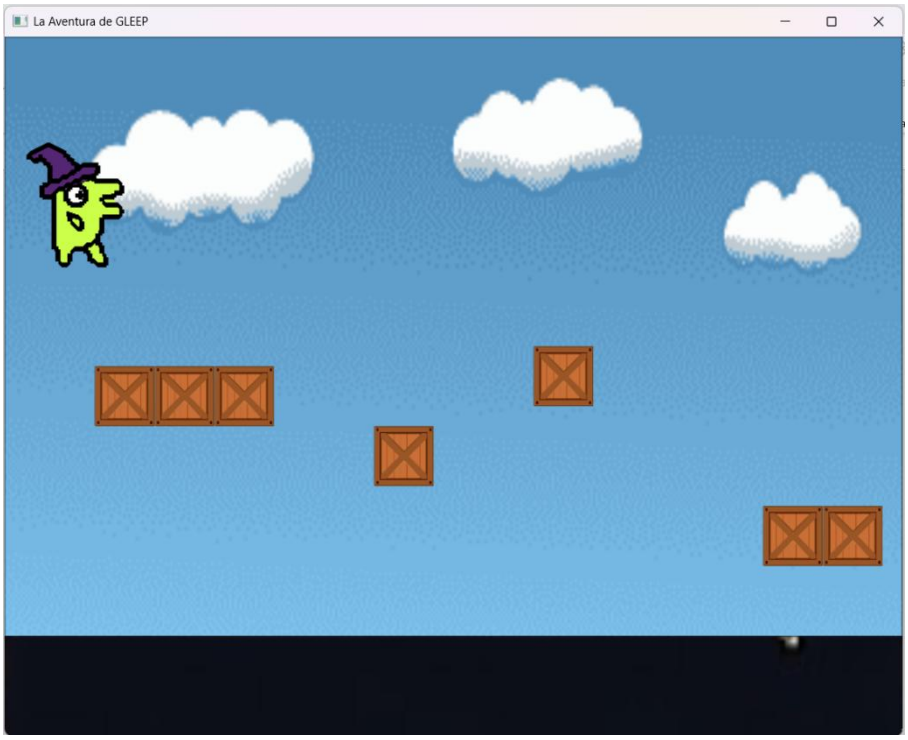


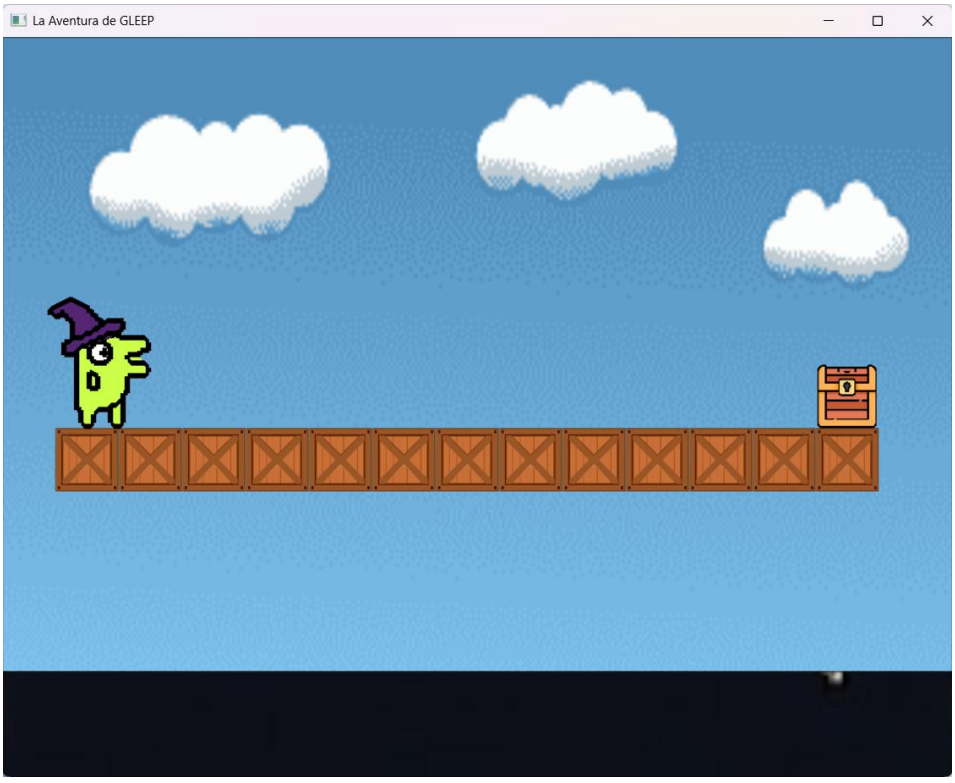
Ya esta el programa en ejecución, únicamente presione enter para ver la animación y P para ir al menú de pausa y ver las diferentes opciones.

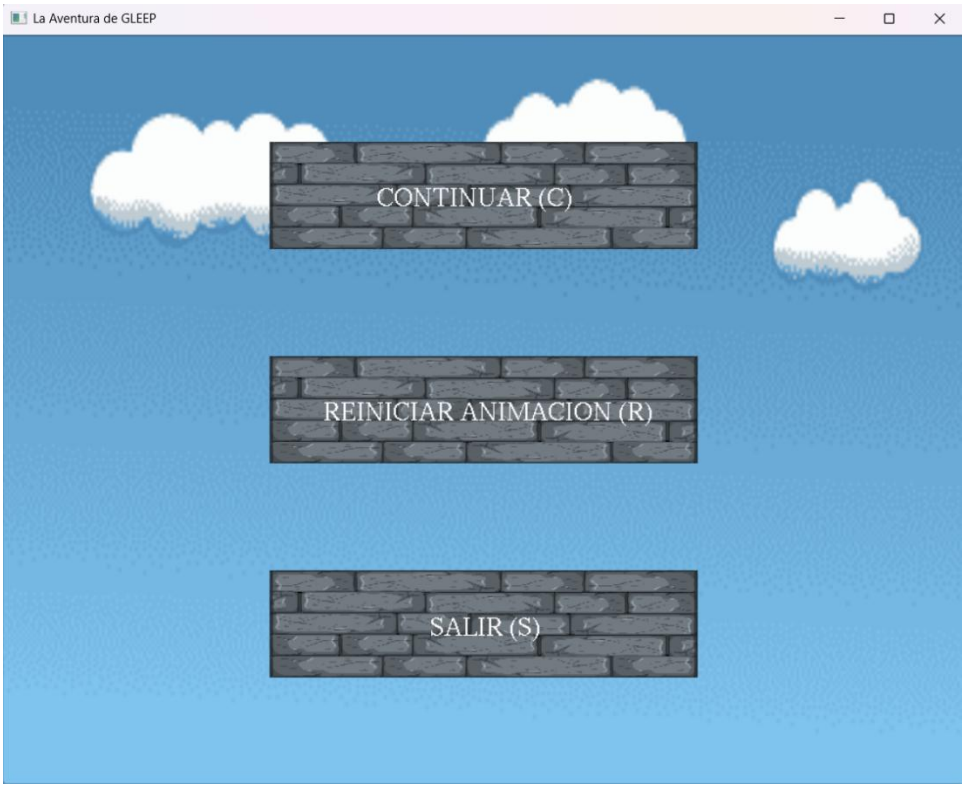
Resultados


La Aventura de GLEEP
LA AVENTURA DE GLEEP
REPRODUCIR (enter)
SALIR (S)


La Aventura de GLEEP


La Aventura de GLEEP


La Aventura de GLEEP


La Aventura de GLEEP
CONTINUAR (C)
REINICIAR ANIMACION (R)
SALIR (S)

Conclusión

Después de haber terminado el proyecto me di cuenta que OpenGL es una buena herramienta para graficar diferentes cosas tanto en 3D como en 2D, igualmente aprendí de la importancia de las estructuras de datos para poder crear animaciones o en general guardar datos de manera correcta, terminado el proyecto lo único que podría decir que se me complico fue un tanto la organización mía de tiempos.