



# **UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO**

## **FACULTAD DE INGENIERÍA**

### **INGENIERÍA EN COMPUTACIÓN**

---

#### **Reporte de Proyecto Final: Alameda Lego**

Lab. de Computación gráfica e Interacción Humano Computadora

Profesor: Jose Roque Roman Guadarrama

Alumno: Rodríguez Agiss Zuriel Uzai

No. de Cuenta: 417101877

Email: zurieluzai2015@gmail.com

Gripo de Teoria: 04

Semestre 2020-2

Fecha de entrega: 15 de Mayo de 2020

# Reporte de Proyecto Final: Alameda Lego

## Actividades Realizadas

Para cubrir los rubros de el avatar y el kiosko se hizo uso del modelo jerárquico. A continuación se presentan los bloques de código usados.

Kiosko:

```
//Se presenta solo fragmento del código para ejemplificar la
realización ya que el código original abarca 1000 líneas (o 30 hojas)

        //-----Kiosko
                //Escaleras
        model = glm::mat4(1.0);
        model = glm::translate(model, glm::vec3(10.0f, 0.1f,
0.0f));
        model = glm::scale(model, glm::vec3(0.2f, 0.2f, 2.0f));
        glUniformMatrix4fv(uniformModel, 1, GL_FALSE,
glm::value_ptr(model));
        glEnable(GL_BLEND);
        Tpasillo.UseTexture();
        Material_opaco.UseMaterial(uniformSpecularIntensity,
uniformShininess);
        meshList[4]->RenderMesh();//1er escalón
        model = glm::translate(model, glm::vec3(1.0f, 1.0f,
0.0f));
        glUniformMatrix4fv(uniformModel, 1, GL_FALSE,
glm::value_ptr(model));
        meshList[4]->RenderMesh();//2do escalón
        model = glm::translate(model, glm::vec3(1.0f, 1.0f,
0.0f));
        glUniformMatrix4fv(uniformModel, 1, GL_FALSE,
glm::value_ptr(model));
        meshList[4]->RenderMesh();//3er escalón
        [...]
        glm::value_ptr(model));
        meshList[4]->RenderMesh();//7mo escalón
        glDisable(GL_BLEND);

        //base
        model = glm::mat4(1.0);
        model = glm::translate(model, glm::vec3(11.75f, 0.8f,
```

```

-1.75f));
    model = glm::scale(model, glm::vec3(1.0f, 1.6f, 1.0f));
    glUniformMatrix4fv(uniformModel, 1, GL_FALSE,
glm::value_ptr(model));
    glEnable(GL_BLEND);
    Tblanco.UseTexture();
    Material_brillante.UseMaterial(uniformSpecularIntensity,
uniformShininess);
    meshList[4]->RenderMesh();//cubo1
    model = glm::translate(model, glm::vec3(0.0f, 0.0f,
1.0f));
    glUniformMatrix4fv(uniformModel, 1, GL_FALSE,
glm::value_ptr(model));
    Tazul.UseTexture();
    meshList[4]->RenderMesh();//cubo2
    model = glm::translate(model, glm::vec3(0.0f, 0.0f,
1.0f));
    glUniformMatrix4fv(uniformModel, 1, GL_FALSE,
glm::value_ptr(model));
    Tblanco.UseTexture();
    meshList[4]->RenderMesh();//cubo3

```

Avatar:

```

//----- Personaje
    model = glm::mat4(1.0);
//Esta parte sirve para que en el modo tercera persona el personaje
esté siempre frente a nosotros
    if (camera.getCameraPosition().y > 2.0f) { model =
glm::translate(model, glm::vec3(0.0, 1.3, 0.0)); }
    else { model = glm::translate(model,
glm::vec3(camera.getCameraPosition())); }
    //model = glm::rotate(model,
camera.getCameraDirection().x+180, glm::vec3(0.0f, 1.0f, 0.0f));
    model = glm::scale(model, glm::vec3(0.5f, 0.5f,
0.5f));
    model = glm::translate(model, glm::vec3(5.0f, 0.0f,
0.0f));
    model = glm::scale(model, glm::vec3(0.8f, 1.0f,
1.0f));
    glUniformMatrix4fv(uniformModel, 1, GL_FALSE,
glm::value_ptr(model));

```

```

        glEnable(GL_BLEND);
        Tnegro.UseTexture();

Material_brillante.UseMaterial(uniformSpecularIntensity,
uniformShininess);
        meshList[4]->RenderMesh();//Cuerpo
        //capa
        model = glm::translate(model, glm::vec3(-0.6f,
-0.2f, 0.0f));
        model = glm::scale(model, glm::vec3(0.1f, 1.4f,
1.5f));
        glUniformMatrix4fv(uniformModel, 1, GL_FALSE,
glm::value_ptr(model));
        glEnable(GL_BLEND);
        Tnegro.UseTexture();
        Material_opaco.UseMaterial(uniformSpecularIntensity,
uniformShininess);
        meshList[4]->RenderMesh();//capa
        //cinturon
        model = glm::scale(model, glm::vec3(10.0f, 0.71f,
0.66f));
        model = glm::translate(model, glm::vec3(0.6f, 0.2f,
0.0f));

        model = glm::translate(model, glm::vec3(0.0f,
-0.625f, 0.0f));
        model = glm::scale(model, glm::vec3(1.0f, 0.25f,
1.0f));
        glUniformMatrix4fv(uniformModel, 1, GL_FALSE,
glm::value_ptr(model));
        Tamarillo.UseTexture();

Material_brillante.UseMaterial(uniformSpecularIntensity,
uniformShininess);
        meshList[4]->RenderMesh();//Cintura

        model = glm::translate(model, glm::vec3(0.0f, -2.5f,
0.25f));
        model = glm::scale(model, glm::vec3(1.0f, 4.0f,
0.4f));
        glUniformMatrix4fv(uniformModel, 1, GL_FALSE,
glm::value_ptr(model));
        Tnegro.UseTexture();
        meshList[4]->RenderMesh();//pierna derecha

```

```

        model = glm::translate(model, glm::vec3(0.0f,
-0.625f, 0.0f));
        model = glm::scale(model, glm::vec3(1.2f, 0.25f,
1.0f));
        glUniformMatrix4fv(uniformModel, 1, GL_FALSE,
glm::value_ptr(model));
        Tpasillo.UseTexture();
        meshList[4]->RenderMesh();//Pie derecho

```

Fuera de eso los demás elementos incorporados como bancas, árboles y botes de basura se insertaron siguiendo un proceso similar.

```

//banca1
        model = glm::mat4(1.0);
        model = glm::translate(model, glm::vec3(-14.0f, 0.7f,
6.0f));
        model = glm::scale(model, glm::vec3(2.0f, 0.25f, 0.75f));
        model = glm::rotate(model, 90 * toRadians, glm::vec3(1.0f,
0.0f, 0.0f));
        glUniformMatrix4fv(uniformModel, 1, GL_FALSE,
glm::value_ptr(model));
        glEnable(GL_BLEND);
        Tcafe.UseTexture();
        Material_opaco.UseMaterial(uniformSpecularIntensity,
uniformShininess);
        meshList[4]->RenderMesh();
        glDisable(GL_BLEND);

        model = glm::mat4(1.0);
        model = glm::translate(model, glm::vec3(-14.75f, 0.3f,
6.0f));
        model = glm::scale(model, glm::vec3(0.3f, 0.6f, 0.5f));
        glUniformMatrix4fv(uniformModel, 1, GL_FALSE,
glm::value_ptr(model));
        glEnable(GL_BLEND);
        Tcafe.UseTexture();
        Material_opaco.UseMaterial(uniformSpecularIntensity,
uniformShininess);
        meshList[4]->RenderMesh();//pata1 de banca
        model = glm::translate(model, glm::vec3(5.0f, 0.0f,
0.0f));

```

```

        glUniformMatrix4fv(uniformModel, 1, GL_FALSE,
glm::value_ptr(model));
        meshList[4]->RenderMesh();//pata2 de banca
        glDisable(GL_BLEND);
[...]
        //Arbol1
        model = glm::mat4(1.0);
        model = glm::translate(model, glm::vec3(5.0f, 2.5f,
-6.0f));
        model = glm::scale(model, glm::vec3(3.0f, 5.0f, 3.0f));
        model = glm::rotate(model, rot * toRadians,
glm::vec3(0.0f, 1.0f, 0.0f));
        glUniformMatrix4fv(uniformModel, 1, GL_FALSE,
glm::value_ptr(model));
        //blending: transparencia o traslucidez
        glEnable(GL_BLEND);
        glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
        Tarbol1.UseTexture(); //
        Material_opaco.UseMaterial(uniformSpecularIntensity,
uniformShininess);
        meshList[3]->RenderMesh();
        glDisable(GL_BLEND);

```

Para lograr que los árboles roten se hace lo siguiente

```

//Se definen dos variables

float rot; // para animacion de arboles
Bool play; //Para determinar si se anima o no

//En esta función que sirve para detectar teclas oprimidas de teclado
se declara que al oprimirse tecla espaciadora la variable play se va
a prender o apagar dependiendo del estado en el que estaba
void inputkey(bool* keys)
{
    if (keys[GLFW_KEY_SPACE]) {
        play = !play;
    }
    [...]
}

Con esta función hacemos que rot aumente su valor en cada iteración
en caso de estar prendida la bandera
void animate(void)

```

```
{
    //Movimiento del objeto
    if (play)
        rot = rot + 0.2;
}
```

Y ya al momento de definir los árboles le indicamos que debe rotar y le mandamos la variable rot para indicarle cuánto

```
//Arbol1
    model = glm::mat4(1.0);
    model = glm::translate(model, glm::vec3(5.0f, 2.5f,
-6.0f));
    model = glm::scale(model, glm::vec3(3.0f, 5.0f, 3.0f));
    model = glm::rotate(model, rot * toRadians,
glm::vec3(0.0f, 1.0f, 0.0f));
```

Para alternar entre cámara tercera persona y cámara aérea

En la función que determina qué hacer de acuerdo a la tecla, se indica el cambio de posición en la cámara

```
if (keys[GLFW_KEY_J])
{
    camera = Camera(glm::vec3(positionP), glm::vec3(0.0f,
2.0f, 0.0f), 0.0f, 0.0f, 5.0f, 0.0f);
}
if (keys[GLFW_KEY_K])
{
    camera = Camera(glm::vec3(0.0, 9.0, 0.0), glm::vec3(0.0f,
2.0f, 0.0f), 90.0f, 0.0f, 10.0f, 1.0f);
}
```

Para crear el ciclo de día y noche

```
//Se define e inicializa la variable segundos que irá almacenando los
segundos desde el inicio de la ejecución del programa
int segundos = 0;
```

```

//Posteriormente se sigue una lógica similar a la siguiente

segundos = 0
startTime = glfwGetTime();

while (!mainWindow.shouldClose())
{
//Toda la ejecución por frame del programa
[...]
    endTime = glfwGetTime();
    double resultado = endTime - startTime;
    //printf("end - start: %.2f-%.2f: %.2f\n",
endTime,startTime,resultado);
    if ((endTime - startTime) >= 1) {
        printf("Segundos: %d\n",segundos);
        segundos++;
        startTime = endTime;
    }

//Y así nos aseguramos de que en cualquier momento del programa
sabemos cuántos segundos han transcurrido. Eso nos sirve para la
siguiente función

//Queremos un ciclo en el que cada segundo sea una hora
double getSunIntensity(int hours) {

    //sacamos la hora del día asumiendo que cada 24 segundos es un
día
    double actualHour = hours % 24;
    //printf("Actual hour: %f\n", actualHour);
    //Lo limitamos al intervalo -0.5:0.5 para aplicar la
distribución normal que es la que más se acerca a la forma en que la
intensidad del sol aumenta y decrece a lo largo del día. El valor
máximo entonces debe ser 1
    double x = actualHour / 24 - 0.5;
    //printf("x: %.2f\n", x);
    int m = 0;
    double s = 0.4;
    static const float inv_sqrt_2pi = 0.3989422804014327;
    float a = (x - m) / s;

    return inv_sqrt_2pi / s * std::exp(-2 * a * a);
}

```



//Y de esta forma hacemos que la luz direccional (sol) se reenderice en cada iteración pero contemplando el valor de la intensidad del sol en ese momento

```
mainLight = DirectionalLight(1.0f, 1.0f, 0.6f,  
    //1.0f, 1.0f,  
    getSunIntensity(segundos),  
    getSunIntensity(segundos),  
    0.0f, 100.0f, -1.0f);
```

//Para hacer que las luminarias se prendan y apaguen en la noche definimos esta función que va a tomar el valor de segundos en ese momento y va a determinar la hora del día para saber si es de día o de noche y retornar un 1 y un 0

```
int getPublicLightsStatus() {  
    double horaActual = segundos % 24;  
  
    if (horaActual < 8 or horaActual>18)  
        return 1;  
    else  
        return 0;  
}
```

//Estos valores los usamos al redefinir las luces de faro y las luces del kiosko

```
pointLights[1] = PointLight(1.0f * getPublicLightsStatus(), 1.0f *  
    getPublicLightsStatus(), 1.0f * getPublicLightsStatus(),  
    0.0f * getPublicLightsStatus(), 1.0f *  
    getPublicLightsStatus(),  
    2.0f, 1.5f, -3.0f,  
    0.3f, 0.2f, 0.1f);
```

```
spotLights[1] = SpotLight(0.0f * getPublicLightsStatus()* lightsOn,  
    1.0f * getPublicLightsStatus()* lightsOn, 1.0f *  
    getPublicLightsStatus()* lightsOn,  
    1.0f * getPublicLightsStatus(), 1.0f *  
    getPublicLightsStatus(),  
    12.0f, 5.0f, -1.5f,  
    -1.0f * getLocVariation(frames), -1.0f, 0.0f,  
    1.0f, 0.0f, 0.0f,  
    20.0f);
```

Ahí vemos que hay una variable llamada `lightsOn` en las luces del kiosko (las luces spotlight). Esta variable es para que el usuario determine con teclado.

Su valor cambia al oprimir la tecla 0. Y se multiplica por los colores ya que en caso de ser 0 devolverá (0,0,0) en el valor rgb, o lo que es lo mismo, luz apagada

```
if (keys[GLFW_KEY_0])
{
    if (lightsOn == 1)
        lightsOn = 0;
    else
        lightsOn = 1;
}
```

Para hacer que las luces se muevan como en espectáculo

```
//se le añade un pequeño offset a la dirección al momento de definir
las luces spotlight en cada iteración.
spotLights[0] = SpotLight(1.0f * getPublicLightsStatus()* lightsOn,
1.0f * getPublicLightsStatus()* lightsOn, 0.0f *
getPublicLightsStatus()* lightsOn,
    1.0f * getPublicLightsStatus(), 1.0f *
getPublicLightsStatus(),
    12.0f, 5.0f, 1.5f,
    -1.0f * getLocVariation(frames), -1.0f, 0.0f,
    1.0f, 0.0f, 0.0f,
    20.0f);
```

Esta función es la siguiente:

```
double getLocVariation(int frames) {
    //120 va a ser 2Pi <=>360°
    double t = (frames *360) / 120;

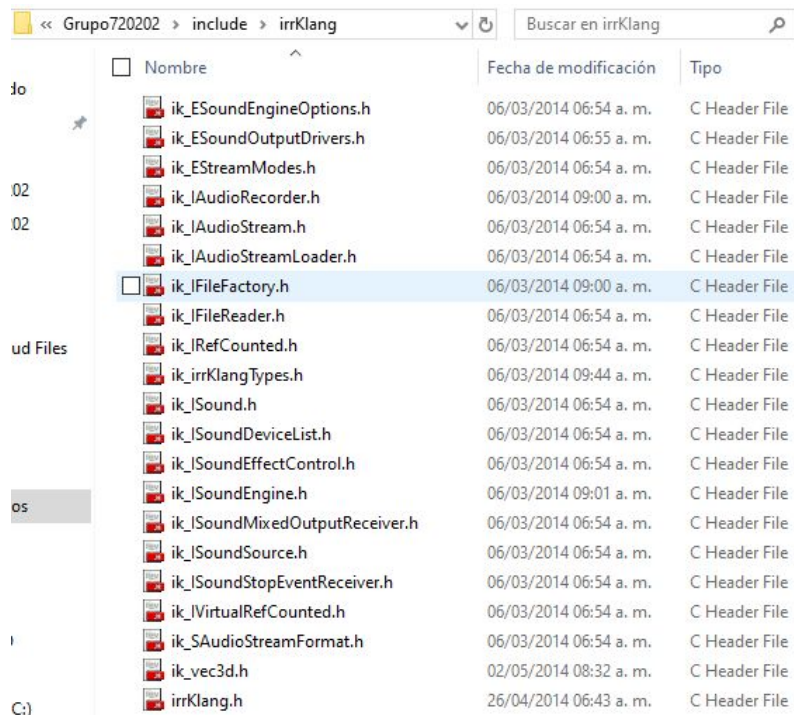
    return sin(t*toRadians);
}
```

// Y lo que hace es tomar el número de frame (recordar que en aproximadamente se renderizan 120 frames por segundo) y la manda a una función dependiente del tiempo (Seno, en este caso) que devuelva

un valor que oscile entre 0 y 1. Para definir la variable frames es muy similar a segundos. Solo que en cada segundo hay que volver a inicializar frames a 0

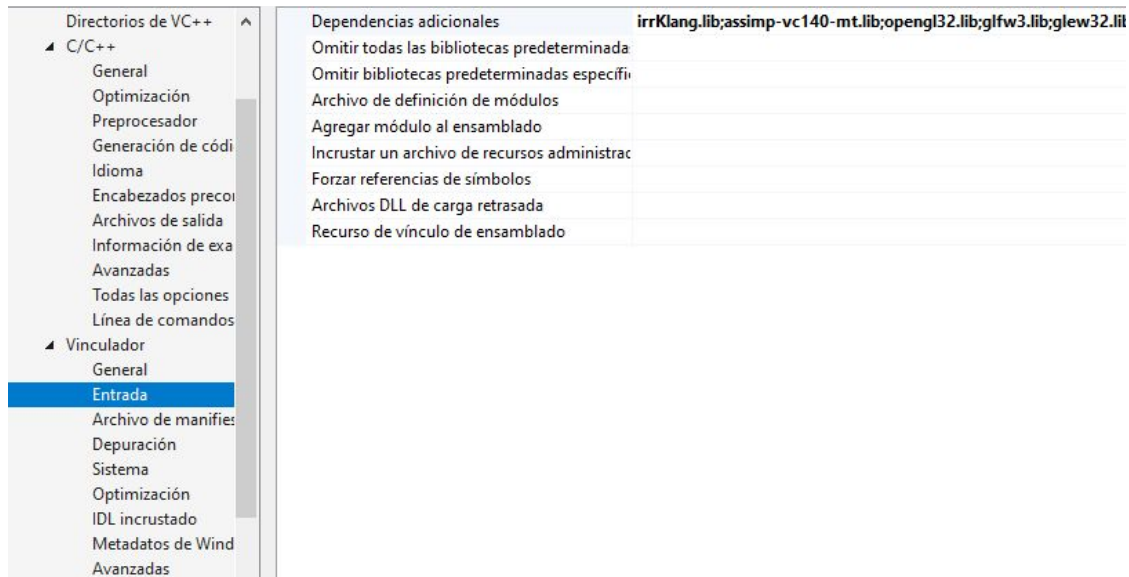
Para insertar la música se uso una librería externa llamada irrKlang. De la página oficial se descarga un comprimido que contiene todo el código fuente necesario.

Se inserta todo el contenido en el comprimido en include en una carpeta llamada irrKlang en nuestra propia carpeta include



	Nombre	Fecha de modificación	Tipo
	ik_ESoundEngineOptions.h	06/03/2014 06:54 a. m.	C Header File
	ik_ESoundOutputDrivers.h	06/03/2014 06:55 a. m.	C Header File
	ik_EStreamModes.h	06/03/2014 06:54 a. m.	C Header File
	ik_IAudioRecorder.h	06/03/2014 09:00 a. m.	C Header File
	ik_IAudioStream.h	06/03/2014 06:54 a. m.	C Header File
	ik_IAudioStreamLoader.h	06/03/2014 06:54 a. m.	C Header File
	ik_IFileFactory.h	06/03/2014 09:00 a. m.	C Header File
	ik_IFileReader.h	06/03/2014 06:54 a. m.	C Header File
	ik_IRefCounted.h	06/03/2014 06:54 a. m.	C Header File
	ik_irrKlangTypes.h	06/03/2014 09:44 a. m.	C Header File
	ik_ISound.h	06/03/2014 06:54 a. m.	C Header File
	ik_ISoundDeviceList.h	06/03/2014 06:54 a. m.	C Header File
	ik_ISoundEffectControl.h	06/03/2014 06:54 a. m.	C Header File
	ik_ISoundEngine.h	06/03/2014 09:01 a. m.	C Header File
	ik_ISoundMixedOutputReceiver.h	06/03/2014 06:54 a. m.	C Header File
	ik_ISoundSource.h	06/03/2014 06:54 a. m.	C Header File
	ik_ISoundStopEventReceiver.h	06/03/2014 06:54 a. m.	C Header File
	ik_IVirtualRefCounted.h	06/03/2014 06:54 a. m.	C Header File
	ik_SAudioStreamFormat.h	06/03/2014 06:54 a. m.	C Header File
	ik_vec3d.h	02/05/2014 08:32 a. m.	C Header File
	irrKlang.h	26/04/2014 06:43 a. m.	C Header File

Y los archivos de la carpeta bin con extensión .dll se añaden a la carpeta raíz del proyecto teniendo que añadirse esto a la configuración



Lo que prosigue es añadir la configuración dada en los ejemplos para activar o apagar el engine de irrKlang. Para ello se usa una clase aparte aunque bien se pudo haber hecho suelto.

```
include "Sonido.h"
#include <stdio.h>
#include<irrKlang/irrKlang.h>
using namespace irrklang;

Sonido::Sonido()
{
    // start the sound engine with default parameters
    estado = false;
}

void Sonido::Reproduce()
{
    if (!estado) {
        engine = createIrrKlangDevice();
        if (!engine)
            printf("\nError al cargar motor de audio.\n");

        // play some sound stream, looped
        engine->play2D("legoSong.mp3", true);
        estado = true;
    }
}
```

```
void Sonido::Stop()
{
    if (estado)
    {
        engine->drop();
        estado = false;
    }
}

Sonido::~Sonido()
{
}
```

### **Problemas presentados**

Más allá de la talacha requerida no se presentaron problemas

### **Conclusiones**

El proyecto me permitió aplicar absolutamente todo lo aprendido y aunque debo admitir que fue muy abrumador debido a la gran cantidad de temas vistos en el curso, el tener un proyecto tan definido ayudó a separar mis grandes dudas en dudas más pequeñas.