UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

COMPUTACIÓN GRÁFICA E INTERACCIÓN HUMANO-COMPUTADORA

LABORATORIO
GRUPO 2

MANUAL TÉCNICO - POKEPINBALL

PROFESOR: ING. JOSE ROQUE ROMAN GUADARRAMA

ALUMNO: AURELIO ROJAS ENG
NÚMERO DE CUENTA: 419240480

1. INTRODUCCIÓN

El proyecto consiste en crear un entorno virtual interactivo de un juego de pinball. En este entorno se va a tener un pinball con temática del universo de Pokémon, con el que se pueda interactuar mediante el uso del teclado y mouse, así como navegar a través de él.

Este ambiente interactivo contará con sonidos ambientales que puedan hacer que el usuario tenga una mejor experiencia, así como tener animaciones para que el usuario pueda interactuar con el ambiente. Las animaciones que incluya se harán mediante el uso de técnicas como Animación por Keyframes, Animación por banderas, Animación sencilla. También se contará con luces que hagan de la experiencia más inmersiva.

Este ambiente no tendrá en cuenta una jugabilidad directa con la máquina de Pinball, por lo que no se implementarán colisiones, física ni cosas más complejas.

El propósito de este Manual es proporcionar información acerca del desarrollo del proyecto y de puntos clave de los que se trabajaron.

2. VARIABLES GLOBALES

Se tienen variables globales con las que se va manipulando la lógica del código, específicamente para animaciones, selección de audio, uso de cámaras, uso de texturas de Skybox y de Luces.

```
Window mainWindow;
std::vector<Mesh*> meshList;
std::vector<Shader> shaderList;
std::vector<Skybox> skyboxList;
std::vector<Camera> cameralist;
std::vector<Material> materialList;
std::vector<PointLight> pointlightList;
std::vector<SpotLight> spotlightList;

int skyboxI = 0, cameraI = 0, materialI = 0, pointlightI = 0, spotlight = 0;

std::string keyFrametxt = "keyFrames.txt";
std::ofstream archivo(keyFrametxt, std::ios::app);
```

Importante destacar que ventanas solo se tiene una principal, pero para el resto de variables que se declaran se hace un listado, para poder almacenar distintas variables de cada una y poder cambiar en tiempo de ejecución las variables de cada una con la manipulación de las variables índices. También se tiene el apoyo una variable de archivo para almacenar los KeyFrames en un txt. Para la parte del tiempo, se tienen restricciones de tiempo en las que se puede manipular y restringir el tiempo de funciones y también manipular el último frame, para así tener animaciones que respeten los 60 fps.

```
GLfloat deltaTime = 0.0f;
GLfloat lastTime = 0.0f;
GLfloat oneST =0.0f, fiveST = 0.0f, tenST = 0.0f;
float lastFrame = 0.0f;
static double limitFPS = 1.0 / 60.0;
```

```
//############# VARIABLES PARA USO DE KEYFRAMES ####################
typedef struct _frame
{
    //Variables para GUARDAR Key Frames
    float movAvion_x;       //Variable para PosicionX
    float movAvion_y;       //Variable para PosicionY
    float movAvion_z;       //Variable para PosicionZ
    float movAvion_xInc;        //Variable para IncrementoX
    float movAvion_yInc;        //Variable para IncrementoY
    float movAvion_zInc;        //Variable para IncrementoZ
    float giroAvion;
    float giroAvionInc;
}FRAME;

bool animacion = false;
float posXavion = 2.0, posYavion = 5.0, posZavion = -3.0;
float movAvion_x = 0.0f, movAvion_y = 0.0f, movAvion_z = 0.0f;
float giroAvion = 0;
#define MAX_FRAMES 100 //100
int i_max_steps = 90; //90
int i_curr_steps = 6; //Depende del frame actual

FRAME KeyFrame[MAX_FRAMES];
int FrameIndex = 6;         //introducir datos
bool play = false;
int playIndex = 0; //Empieza desde 0
//#######################


//VARIABLES PARA USO DE TEXTURE-OFFSET
//Dobles valores para cada textura que se quiera se pueda mover con
float toffsetScoreboardU = 0.0f, toffsetScoreboardV = 0.0f;
int toffsetcountu = 0, toffsetcountv = 0;
```

Se tiene la estructura, para hacer los frames y toda la declaración para interpolar en tiempo de ejecución. También las variables para los offset de texturas, en donde se puede manipular el movimiento de una textura en tiempo real.

También como variable de ejecución se tiene el Engine de audio y todas las declaraciones de Modelos, Luces, Camara, Skybox, Texturas, etc. En donde se tienen valores fijos que dentro del loop principal ya no se estarían actualizando.

3. ESTRUCTURA DEL PROYECTO

Los componentes que se ocuparon para el proyecto son códigos para hacer Cámaras, Shaders, Skybox, Luces, Meshes, Models, Material y Window. Cada uno con sus funciones y sus headers, declarando todo lo que pueden o no hacer. Asimismo, se tienen las carpetas para modelos, texturas y audios, en donde en cada una de ellas se almacenó lo que se utiliza en el proyecto.

Con todo esto, se tiene un código main en donde se manda a llamar todo y se implementa para la ejecución del pinball.

4. CÓDIGO FUENTE

Las partes importantes a destacar son las siguientes:

Dentro de Window.cpp

```
//CREAR VENTANA
mainWindow = glfwCreateWindow(width, height, "Proyecto Final: Pinball", glfwGetPrimaryMonitor(), NULL);
glfwSetInputMode(mainWindow, GLFW_CURSOR, GLFW_CURSOR_DISABLED);
```

```
void Window::createCallbacks()
{
    glfwSetKeyCallback(mainWindow, ManejaTeclado);
    glfwSetCursorPosCallback(mainWindow, ManejaMouse);
    glfwSetMouseButtonCallback(mainWindow, ManejaMouseButtons);
}
```

```
void Window::ManejaMouse(GLFWwindow* window, double xPos, double yPos)
{

    Window* theWindow = static_cast<Window*>(glfwGetWindowUserPointer(window));

    if (theWindow->mouseFirstMoved)
    {
        theWindow->lastX = xPos;
        theWindow->lastY = yPos;
        theWindow->mouseFirstMoved = false;
    }

    // Calculate the offset from the center of the screen
    double xOffset = xPos - theWindow->width / 2.0;
    double yOffset = yPos - theWindow->height / 2.0;

    // Set the mouse position to the center of the screen
    glfwSetCursorPos(window, theWindow->width / 2.0, theWindow->height / 2.0);

    // Update the lastX and lastY values
    theWindow->lastX = theWindow->width / 2.0;
    theWindow->lastY = theWindow->height / 2.0;

    // Update the xChange and yChange values with the offset
    theWindow->xChange = xOffset;
    theWindow->yChange = -yOffset;


}
```

```
void Window::ManejaMouseButtons(GLFWwindow* window, int button, int action, int mods)
{
    Window* theWindow = static_cast<Window*>(glfwGetWindowUserPointer(window));

    if (button == GLFW_MOUSE_BUTTON_LEFT)
    {
        if (action == GLFW_PRESS)
        {
            theWindow->spring1_sfx = true;
            theWindow->resorte += 1.0f;
        }
        else if (action == GLFW_RELEASE)
        {
            theWindow->spring2_sfx = true;
            theWindow->resorte -= 1.0f;
        }
    }
}
```

Lo que es importante comentar aquí es la implementación del glfwGetPrimaryMonitor() dentro de la creación de la ventana, esto para poder tener pantalla completa, también deshabilitando la vista del cursor. Lo otro importante es la inclusión del CallBack de botones de mouse. Esto para la lectura del click que se desee. También el set del cursor position a la mitad de la ventana, para poder asi tener movimiento de mouse libre e infinito sin restricciones. En el Window.h solo se declaran los getters y setters de las distintas variables que se ocupan y para el resto del código se utilizó la base proporcionada por el Ing. Jose Roque Roman Guadarrama en las distintas prácticas, solo se fueron implementando todas en un solo conjunto completo.

Dentro del main.cpp

```
//Different timers for 1, 5 and 10 seconds
if (now - oneST >= 1.0f) {...
    oneST = now;
}

if (now - fiveST >= 5.0f) {...
    fiveST = now;
}
if (now - tenST >= 10.0f) {...
    tenST = now;
}
```

```
if (mainWindow.getFlipper_sfx()) {
    engine->play2D("audio/flipper.wav", false);
    mainWindow.setFlipper_sfx();
}
if (mainWindow.getSpring1_sfx()) {
    engine->play2D("audio/spring1.wav", false);
    mainWindow.setSpring1_sfx();
}
if (mainWindow.getSpring2_sfx()) {
    engine->play2D("audio/spring2.wav", false);
    mainWindow.setSpring2_sfx();
}
if (mainWindow.getBumper_sfx()) {
    engine->play2D("audio/bumper.wav", false);
    mainWindow.setBumper_sfx();
}
if (mainWindow.getCoin_sfx()) {
    engine->play2D("audio/coin.wav", false);
    mainWindow.setCoin_sfx();
}
if (mainWindow.getPidove_sfx()) {
    engine->play2D("audio/pidove.wav", false);
    mainWindow.setPidove_sfx();
}
if (mainWindow.getBirdFlap_sfx()) {
    engine->play2D("audio/birdflap.wav", false);
    mainWindow.setBirdFlap_sfx();
}
```

Lo importante para el main que diferencia el resto del código base proporcionado es la implementación de funciones que entran cada X cantidad de tiempo, como lo pueden ser 1 seg, 5 seg y 10 seg. También las condicionales de los sonidos especiales que se van a activar una vez cada que se activen banderas de sonido y reproduce el .wav correspondiente.

El resto del main se modificó con respecto al código base proporcionado, solo afectando la legibilidad del código para uso propio.

5. CONCLUSIONES

Como se puede ver en este manual, a diferencia de lo que se implementó en los laboratorios, no varía mucho, se usaron algunas lógicas distintas, pero de ahí en fuera se mantiene la misma base, utilizando el material ya existente para crear cosas que ya están. Claro está que hacer ciertos cambios también requiere cierto criterio para decidir que hacer distinto a las cosas que ya se tenían implementado. Unos de estos desafíos es tener que leer la documentación de las librerías que se ocuparon, porque si bien parecen cambios sencillos, para poder checar como se hacen correctamente hay que entender bien como se hacían las cosas. Sobre esto también se puede ver que algunas de los códigos base que se ocuparon se implementaron sobre las prácticas se trató de optimizar para tener mayor legibilidad y uso. A futuro todavía falta completar varias cosas de las cuales se seguirá expandiendo sobre este manual y explicando más a fondo.

6. ANEXOS

Biblioteca de Audio – Irrklang: https://www.ambiera.com/irrklang/
Biblioteca de Ventanas – GLFW: https://www.glfw.org/documentation
Biblioteca de Carga de Modelos: https://github.com/nothings/stb/blob/master/stb_image.h
GLEW: https://glew.sourceforge.net/
Biblioteca de Matemáticas - GLM: https://glm.g-truc.net/0.9.9/api/index.html