****

SEGUNDO PROYECTO DE TALLER DE PROGRAMACIÓN.

**ELABORADO POR:**

Yaritza Elena López Bustos.

María Fernanda Álvarez Martínez.

**PROFESOR:**

Antonio González Torres.

**I SEMESTRE**

**CARTAGO, 2019**

Contenido

[Introducción 3](#_Toc7385976)

[Descripción del problema. 4](#_Toc7385977)

[Descripción de la solución con UML. 6](#_Toc7385978)

[Análisis de resultados. 7](#_Toc7385979)

[Bitácora de actividades. 8](#_Toc7385980)

[Conclusión. 9](#_Toc7385981)

# Introducción

El presente proyecto consiste en la creación de un código para el juego Dakar Death, cuyo cual es un juego de carros con el único objetivo de sobrevivir el máximo tiempo posible, enfrentándose a otros jugadores y a diversos objetos distribuidos en el mapa, durante el juego se deberá destruir a los jugadores rivales y, además, sobrevivir a los obstáculos que se presenten. El nivel de dificultad aumentará pasados los 2 minutos de sobrevivencia.

El manejo de multijugador se realiza mediante sockets e hilos, con servidores y clientes para el manejo de ambos jugadores en diferentes pantallas.

Además, para el manejo de los jugadores se utilizan joysticks con ayuda e los arduinos.

Para la creación de dicho proyecto se utilizaron las bibliotecas de Pygame y Tkinter, implementado clases, programación orientada a objetos, entre otros.

# Descripción del problema.

Desarrollar dos controles para el juego pyDakarDeath. Cada control requiere usar un Arduino, 1 joystick, 1 botón, un LED y un programa que envía mediante una comunicación serial entre el microcontrolador y una computadora los códigos del joystick y el botón cuando es presionado. El juego tiene que ser multi-jugador y puede ser usado por dos jugadores de forma simultánea, en donde cada uno puede jugar haciendo uso de un joystick y un botón. Los joysticks permiten que los jugadores se muevan en diferentes direcciones y ataquen con disparos a los vehículos enemigos y a los obstáculos. El funcionamiento de los de los joysticks, botones y LEDs tiene que ser de la siguiente manera:

1. Se utiliza un joystick para mover el vehículo del jugador hacia adelante, atrás, izquierda y derecha.
2. El botón se utiliza para disparar. Cuando el botón es presionado lanza un disparo y se enciende el LED. El objetivo de los disparos son los vehículos enemigos, los obstáculos u otro competidor. Cuando el jugador acierta con un disparo sobre un objetivo, este se destruye.
3. El LED se enciende cada vez que se hace un disparo y se apaga de forma inmediata.

Para establecer la comunicación serial entre el Arduino y la computadora utilice la biblioteca pySerial. La instalación se puede realizar desde la consola, tanto en Windows como en Linux, con el comando python -m pip install pyserial. Una vez que se ha realizado la instalación, se puede hacer la lectura de los datos del dispositivo de control usando el programa en Python con el comando “import serial” para importar pySerial. Tener que cuenta que los datos enviados y recibidos por el Arduino son datos tipo string codificados en ASCII.

PyDakarDeath 2.0 es la segunda versión de un juego en el que pueden competir dos jugadores usando cada uno un joystick y un botón y de acuerdo con su puntaje pueden pasar de un nivel a otro, por lo que el juego contará con al menos 3 niveles. En esta versión del juego el sistema tendrá un menú con las siguientes opciones:

1. Registrar nombre de usuario: El sistema permite que el jugador ingrese el nombre de su usuario al cual se asociará el puntaje más alto que logre en las competiciones. El puntaje de cada usuario se almacenará en un archivo CSV.
2. Ver puntaje: Cuando esta opción es seleccionada, el sistema muestra el puntaje de los 5 mejores jugadores.
3. Iniciar partida: Al seleccionar esta opción el jugador debe seleccionar el nombre de usuario con el cual jugará la partida.
4. Guardar partida: El jugador puede guardar el estado de una partida al seleccionar esta opción de menú. El sistema debe almacenar las posiciones actuales de todos los carros del juego y los puntajes de los jugadores.
5. Reiniciar partida: Esta opción es utilizada para continuar una partida que había sido iniciada con anterioridad, con toda la información de las posiciones de los carros y los puntajes de los jugadores.
6. Salir: El usuario se sale del sistema y termina la partida, si estuviera jugando en ese momento.

Los jugadores pasan de nivel cuando el jugador ha superado 2 minutos de juego sin sufrir una colisión o ser destruido. Los niveles superiores del juego cuentan con mayor número de obstáculos y enemigos. El juego cuenta con sonidos de aceleración, disminución de la velocidad, frenado y explosiones al chocar. Las reglas básicas del juego son las siguientes:

1. El tiempo de cada partida del juego es de 2 minutos.
2. El juego permite que dos jugadores compitan a la vez.
3. El sistema controla un gran número de “dummy vehicles” que corren a una velocidad constante.
4. El punto de partida de la competencia, y donde se completa se encuentra indicado en la pista por una zona marcada como un tablero.
5. La velocidad de los vehículos de los jugadores es variable y depende de las maniobras que hagan con los controles.
6. El puntaje de cada jugador se calcula con base en el número de enemigos y obstáculos que elimina durante la competencia.
7. Si un jugador sufre una colisión con un objeto puede quedar destruido.

# Descripción de la solución con UML.

Main

Player

Player

Enemigo

Bala

Network

Enemigo

* def movimientoDerecha
* def movimientoIzquierda
* def movimientoDerecha
* def movimientoizquierda
* def \_\_movimiento
* def destrucción
* def disparar1
* def disparar2
* def dibujar
* def hit
* def hit1
* def dibujar
* def comportamientoImages
* def \_\_movimientos
* def \_\_descenso
* def \_\_movimientolateral
* def \_\_ataque
* def \_\_disparo
* def hit
* def connect
* def send

Network

Colisiones

* def dibujar
* def trayectoria
* def dibujar
* def cargarEnemigo
* def detenerJuego

Bala

* Def colisiones
* Def level\_2
* Def detenerJuego
* Def Guardar
* Def close
* Def send\_data
* Def parse\_data

# Análisis de resultados.

1. En este proyecto se logró la movilidad de los jugadores con sus respectivos disparos cada uno, la creación de los diversos mapas, además se logró crear los diversos ¨enemigos¨.
2. Por otro lado, dicho proyecto también es deficiente en el registro de las puntuaciones en un archivo Json y asociarlo con el respectivo jugador.
3. Se logró el registro de los ¨nicknames ¨ en un archivo Json.
4. Además, se logró crear el desierto (fono) con los respetivos obstáculos
5. El proyecto logra conectarse a un servidor para cumplir con el objetivo multijugador
6. El juego posee 3 niveles, cada uno aumenta su dificulta

# Bitácora de actividades.

|  |  |
| --- | --- |
| **Actividades.** | **Horas.** |
| Análisis de requerimientos. | 3 |
| Diseño de aplicación y diagrama de clases. | 30 |
| Investigación de funciones. | 15 |
| Programación. | 29 |
| Documentación Interna. | 4 |
| Pruebas. | 13 |
| Elaboración del documento | 3 |
| **TOTAL** | 97 |

# Conclusión.