Índice

1. pyDakarDeath es un juego de carrera de carros que recreamos a base de una combinación de ideas originales (niveles) como la idea base del profeso. La finalidad de este proyecto es mejorar el uso de tkinter, pygame, csv, pandas y hacer un cliente servidor para poder jugar en dos computadoras al mismo tiempo utilizando pyhton únicamente ya que tkinter, pygame, csv y pandas son bibliotecas que contiene python por lo tanto reforzamos ese lenguaje de programación. En este documento aparece la bitácora, descripción del problema, los ejemplos y análisis de resultados y la estadística de cuánto tiempo invertimos en este proyecto.

2. Descripción del problema
   pyDakarDeath es un juego que está en una plataforma cliente servidor. El juego se trata de una carrera entre dos jugadores en un desierto con obstáculos, trampas mortales y carros tontos. El jugador acumula puntos según vaya avanzando y destruyendo a los carros tontos esto último se puede lograr por medio de disparos o empujando al rival contra un objeto mortal la velocidad del carro dependerá de cómo maneje las teclas para ir aumentado o reduciendo la velocidad del carro , si el jugador choca contra un objeto mortal, los objetos aparecerán con mayor frecuencia conforme el nivel vaya aumentando, automáticamente perderá y si se sale de los límites establecidos se irán reduciendo el puntaje también se le agregara un modo de pausar el juego y volverlo a iniciar desde el punto donde se dejó anteriormente. El juego ira almacenado los datos del jugador como su nombre de usuario y su puntaje. El juego mostrara las mejores puntuaciones de cada usuario en una ventana como también en otra ventana aparece el elegir o crear nuevo usuario y también estará el botón que nos llevara directo al juego y por ultimo un botón que nos permita salir y cerrar el juego completamente El jugador con más puntos gana la partida.

3. Bitácora
   Lunes 13 mayo

   Empezamos investigando el uso de pygame y procedimos a instalarlo en la computadora usando el símbolo de sistema con el pip install pygame, procedimos a consultar la documentación de pygame para irnos familiarizando con algunos comando y utilidades que facilita esta biblioteca.

Martes 14 de mayo

Procedimos a empezar con la interface del juego usando tkinter como lo son el menú de seleccionar jugador, mejores puntajes, salir y empezar el juego, también se intentó trabajar con el archivo json para ir almacenado los datos del juego, también se le

Miércoles 15 de mayo

Se siguió tratando de usar el json para almacenar los datos, escogimos las imágenes que se iban a usar para los carros en el juego, también se fue decidiendo las dificultades de cada nivel en el juego

Jueves 16 de mayo

Se fue a aclarar algunas dudas sobre el json al profe durante la mañana y en base de eso se empezó a modificar el uso del json para implementar las sugerencias que el profe facilito. Y empezamos con el juego usando pygame

Viernes 17 de mayo

El profesor facilito un código para hacer el cliente servidor se analizó el código para tener una idea de cómo hacer que el servidor corriera en el juego que estamos creando.

Sábado 18 al lunes 20 de mayo

Se sigue trabajando únicamente en el juego con pygame

Martes 21 de mayo

Se le volvió a preguntar al profesor sobre el json que estaba dando problemas para guardar los datos y un error que estaba dándonos en el servidor por sugerencia del profesor dejamos de utilizar el json que nos estaba dando problemas y pasamos a almacenar todo mejor en un archivo csv por lo cual se tuvo que crear este tipo de archivo desde 0 por lo cual se cambió el código que teníamos en json probamos este código el cual si almacenaba los datos y decidimos manipularlo por medio de pandas.

Miércoles 22 a domingo 26 de mayo

Se sigue trabajando en el juego

4. Análisis de resultados

Se importa lo que necesitamos con
import tkinter as tk

```
import pygame
import random
import socket
import struct
import pickle
import logging
import threading
import time
from os import path
from random import randint
```

Definimos lo de destruir las ventanas con

```
def destruirMenu():
   Menu.destroy()
   canvaElegirJ()
def destruirElegirJ():
   ElegirJ.destroy()
   canvaMenu()
def IrAlJuego():
   ElegirJ.destroy()
   Main.iconify()
   Juego()
```

Definimos una función que nos permite agregar los nombres de los usuarios

```
def add():
   values=[name.get(), btn.cget('text')]
   data=pickle.dumps(values)
   sock.sendto(data, (multicast_addr, port))
```

Definimos la canva menú junto con sus botones y la definimos como global el menu

```
def canvaMenu():
   global Menu
   global Nombre
   Menu=tk.Canvas(Main,height=700,width=600,bg="DarkGoldenrod2")
   Menu.pack(fill=tk.BOTH, expand=True)
   NuevoJuegoBtn=tk.Button(Menu,text="Nueva
Partida",bg="DarkGoldenrod2",fg="White",bd=0,font=("Times", 24),command=destruirMenu)
   NuevoJuegoBtn.pack(side=tk.TOP)
   PuntajeBtn=tk.Button(Menu,text="Mejores
Puntuaciones",bg="DarkGoldenrod2",fg="White",font=("Times", 24),bd=0)
   PuntajeBtn.pack(side=tk.TOP)
   SalirBtn=tk.Button(Menu,text="Salir",bg="DarkGoldenrod2",fg="White",font=("Times",
24),command=Main.destroy,bd=0)
   SalirBtn.pack(side=tk.TOP)
   Main.mainloop()
```

Definimos la canva del elegir jugador con sus botones y el global de elegirj

```
def canvaElegirJ():
```

```
    global ElegirJ
    global entrada
    ElegirJ=tk.Canvas(Main, height = 700, width = 600,bg="DarkGoldenrod2")
    ElegirJ.pack(fill=tk.BOTH, expand=True)
    lblNuevoJugador=tk.Label(ElegirJ,text="Si eres un nuevo Jugador, ingresa tu nombre aquí",bg="DarkGoldenrod2",fg="white",bd=0,font=("Times", 18))
    lblNuevoJugador.pack(side=tk.TOP)
    entrada=tk.Entry(ElegirJ)
    entrada.pack(side=tk.TOP)
    lblCargarJugador=tk.Label(ElegirJ,text="Si ya habias jugado antes, busca tu nombre aquí",bg="DarkGoldenrod2",fg="white",bd=0,font=("Times", 18))
    lblCargarJugador.pack(side=tk.TOP)

btnSiguiente=tk.Button(ElegirJ,text="Siguiente",bg="DarkGoldenrod2",fg="white",bd=0,command=IrAlJuego,font=("Times", 18))
    btnSiguiente.pack(side=tk.TOP)
    btnVolver=tk.Button(ElegirJ,text="Volver al Menu Principal",bg="DarkGoldenrod2",fg="white",bd=0,command=destruirElegirJ,font=("Times", 18))
    btnVolver.pack(side=tk.TOP)
```

Definimos el juego, la constante del color de fondo y tipo de letra que se va a usar junto con la constante del nombre del juego y el tamaño de la pantalla con

```
def Juego():
    pygame.init()
    letra=pygame.font.Font(None,20)
    colorFondo=(255,180,40)
    win= pygame.display.set_mode((500,500))
    pygame.display.set_caption("pyDakarDeath")
```

Lo que usamos para la constante y variable del vehículo y el jugador

```
    Carro1=pygame.image.load("car1.png")
    Jugador=Carro1.get_rect()
    Jugador.centerx=500/2
    Jugador.centery=450
    velocidad=5
```

la variable de uno de los obstáculos junto con las variables que se van a modificar dentro del while como el de los puntajes junto con un true que se termina cuando el programa se termina y ocasiona que se termine el while

```
    Cactus1=pygame.image.load("Cactus1.png")
    Obstaculo1=Cactus1.get_rect()
    velocidadObst=10
    aparece=False
    punt=0
    run=True
```

el loop que permite que la pantalla se mantenga actualizada junto con el while que se termina con el true

```
    while run:
        puntuacion=letra.render("X"+str(punt), 0,(0,0,0))
        pygame.time.delay(100)
```

```
        for event in pygame.event.get():
            if event.type==pygame.QUIT:
                run=False
        keys= pygame.key.get_pressed()
        if keys[pygame.K_LEFT]and Jugador.left>0+velocidad:
            Jugador.centerx-=velocidad
        if keys[pygame.K_RIGHT]and Jugador.right<500-velocidad:
            Jugador.centerx+=velocidad
        if keys[pygame.K_UP]and Jugador.top>0+velocidad:
            Jugador.centery-=velocidad
        if keys[pygame.K_DOWN]and Jugador.bottom<500-velocidad:
            Jugador.centery+=velocidad
        if(Obstaculo1.centerx>=550):
            aparece=False
        punt+=1
        win.fill(colorFondo)
        if(aparece==False):
            Obstaculo1.centery=-10
            Obstaculo1.centerx=randint(10,450)
            win.blit(Cactus1,Obstaculo1)
            aparece=True
        else:
            Obstaculo1.centery+=velocidadObst
            win.blit(Cactus1,Obstaculo1)
        win.blit(puntuacion,(0,0))
        win.blit(Carro1,Jugador)
        pygame.display.flip()
    pygame.quit()
Main=tk.Tk()
Main.title("pyDakarDeath")
Main.config(bg="black")
Main.geometry("600x700+0+0")
Menu=tk.Canvas(Main,height=700,width=600,bg="DarkGoldenrod2")
Menu.pack(fill=tk.BOTH, expand=True)
NuevoJuegoBtn=tk.Button(Menu,text="Nueva
Partida",bg="DarkGoldenrod2",fg="White",bd=0,font=("Times", 24),command=destruirMenu)
NuevoJuegoBtn.pack(side=tk.TOP)
PuntajeBtn=tk.Button(Menu,text="Mejores
Puntuaciones",bg="DarkGoldenrod2",fg="White",font=("Times", 24),bd=0)
PuntajeBtn.pack(side=tk.TOP)
SalirBtn=tk.Button(Menu,text="Salir",bg="DarkGoldenrod2",fg="White",font=("Times",
24),command=Main.destroy,bd=0)
SalirBtn.pack(side=tk.TOP)

Y se maneja el servidor
multicast_addr = '224.0.0.3'
bind_addr = '0.0.0.0'
port = 3000
```

```
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
membership = socket.inet_aton(multicast_addr) + socket.inet_aton(bind_addr)
Main.mainloop()
```

Para el servidor se uso

```
import random
import socket
import pickle
import threading
import time
import pandas as pd
from tkinter import *
from os import path
from random import randint
window = Tk()
window.title("Server")
window.geometry('350x200')
multicast_addr = '224.0.0.3'
bind_addr = '0.0.0.0'
port = 3000
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
membership = socket.inet_aton(multicast_addr) + socket.inet_aton(bind_addr)
sock.setsockopt(socket.IPPROTO_IP, socket.IP_ADD_MEMBERSHIP, membership)
sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
sock.bind((bind_addr, port))
name = "Server"
def thread_function():
   while True:
      message, address = sock.recvfrom(255)
      values = pickle.loads(message)
      if(values[0] != name and values[1] == "puntajes"):
         df = pd.read_csv('datos.csv', delimiter = ',')
         for index, row in df.iterrows():
            print(row[1])
            values=["puntajes", row[0], row[1]]
            data=pickle.dumps(values)
            sock.sendto(data, (multicast_addr, port))
      elif(values[0] != name and values[1] == "Add"):
         df = pd.read_csv('datos.csv', delimiter = ',')
         df = df.append({'usuario': values[0], "score":0}, ignore_index=True)
         df.to_csv('datos.csv', index=False)
      elif (values[0] != name and values[1] == "Update"):
         df = pd.read_csv('datosata.json', delimiter = ',')
         df.loc[df['usuario'] == values[0], 'puntajes'] = int(values[2])
         df.to_csv('datos.csv', index=False)

x = threading.Thread(target=thread_function)
x.start()
```

window.mainloop()

5. Estadística de tiempo

Análisis de requerimientos	Horas 5
Diseño de la aplicación y diagrama de clases	Horas 4
Investigación de funciones	Horas 6
Programación	Horas 58
Documentación interna	Horas 2 y 30
Pruebas	Horas 6
Elaboración documento	Horas 4 y 30
TOTAL	Horas 86