UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
SEGUNDO SEMESTRE 202

B

MANUAL TÉCNICO
FASE 3

Nombre: Angel Francisco Sique Santos
Carné: 202012039

INTRODUCCIÓN

La empresa Usac Games desea implementar un videojuego que permitan desarrollar la agilidad mental de los usuarios, por lo cual ha planeado desarrollar una aplicación con el juego Batalla naval y le solicita a usted como estudiante de estructura de datos poder implementar algoritmos, funciones y estructuras que permitan que el juego tenga un rendimiento óptimo y fluido.

REQUISITOS MÍNIMOS

- 500mb de disco duro
- 2gb de RAM
- Linux o MacOs
- Equipo Intel Pentium o superior
- Python 3.10

LIBRERÍAS

- Graphviz
- Pillow 9.2.0
- PySimpleGui 4.60.3
- Glove
- SHA256
- JsonCPP
- json
- webbrowser
- request

OPCIONES DEL PROGRAMA

```python
usuario_global = {
    'nick': '',
    'password': '',
    'monedas': '1000',
    'edad': '',
    'id': '',
    'juegos': '0',
    'puntos': '0',
    'fallos': '0'
}

invitado = {
    'nick': 'Invitado',
    'password': 'Invitado',
    'monedas': '1000',
    'edad': '0',
    'id': '-1',
    'juegos': '0',
    'puntos': '0',
    'fallos': '0'
}
```

Creamos dos json con los datos del usuario con sesión iniciada y otro con los datos del jugador que ser el invitado.

```python
def tablero_1vs1(tamanio :int):
    usuario_global['puntos'] = usuario_global['monedas']
    invitado['puntos'] = invitado['monedas']
    usuario_global['fallos'] = '0'
    invitado['fallos'] = '0'
    sg.theme('DarkTeal4')   # Add a touch of color
    if(tamanio >= 10):
        layout = []
        botones = []
        #Numero de barcos
        constante = int(((tamanio - 1)/10)+1)
        Portaaviones = 1*constante
        Submarino = 2*constante
        Destructor = 3*constante
        Buque = 4*constante
        vidas = 3
        layout.append(
            [
            sg.OptionMenu(('Portaaviones', 'Submarino', 'Destructor', 'Buque'),'Portaaviones',key='barco'),
            sg.Text('Vidas',text_color='#FFARF1',font='Futura 15'),sg.Text(vidas,text_color='#FFARF1',font='Futura 15'),
```

```python
    event, values = window.read()
    if event == sg.WIN_CLOSED:…
    elif event == 'Retroceder un movimiento':…
    else:
        for k in range(1 , len(layout)):
            for l in range(len(layout) - 1):
                if(layout[k][l].ButtonText == event and layout[k][l].ButtonColor[1] == '#6c7b95'):
                    print(values['barco'])
                    if(colorear_botones(values['barco'],k,l,matriz,layout)):
                        while True:
                            event2, values2 = window.read()
                            if event2 == sg.WIN_CLOSED:
                                break
                            else:
                                for k2 in range(1 , len(layout)):
                                    for l2 in range(len(layout) - 1):
                                        if(layout[k2][l2].ButtonText == event2 and layout[k2][l2].ButtonColor[1] ==
                                            if(values['barco'] == 'Portaaviones' and portaavionesNoPintado(k,l,k2,l2
                                            elif(values['barco'] == 'Submarino' and submarinoNoPintado(k,l,k2,l2,mat
                                            elif(values['barco'] == 'Destructor' and destructorNoPintado(k,l,k2,l2,m
                                            elif(values['barco'] == 'Buque'):…
                                            limpiar_botones(layout)
                            break
                        break
```

Luego creamos el tablero donde se colocaran los barcos del usuario con sesión iniciada. Después de colocar los barcos el jugador 2 hará el primer ataque, cuando termine de de hacer el ataque el jugador 2 colocara sus barcos y el jugador 1 hará el segundo ataque.

```
            break

        if(revisar_tablero(matriz,layout)):

            sg.popup('Ganaste')
            break
        window.hide()
        sg.popup_ok('Turno de colocar barcos del jugador 2')
        crear_tablero(tamanio,matriz,window,layout)
```

```
    sg.popup_ok('Turno del jugador 2')
    window.hide()
    crear_tablero_con_matriz(matriz2, window2,layout2,matriz,window,layout)
    break
```

Al final de esto se llama una función que pide como parámetros la matriz y el tablero del jugador 1 y los del jugador 2. Esta función entra en recursividad llamándose a sí misma mientras el juego se lleva a acabo, dependiendo del turno del jugador se muestra el tablero

```
for k in range(1 , len(layout)):
    for l in range(len(layout) - 1):
        if(layout[k][l].ButtonText == event ):
            if window.Title == "Jugador 2":
                disparos.conexion(k-1,l)
            if(pintar_disparo(k, l, matriz, layout)):
                #agregar_movimiento(k-1,l,nombreJuego,usuario_global['id'])
                if(window.Title == "Jugador 2"):
                    usuario_global['monedas'] = int(usuario_global['monedas']) + 20
                    layout[0][1].update(usuario_global['monedas'])
                    actualizar_monedas(+20)
                else:
                    invitado['monedas'] = int(invitado['monedas']) + 20
                    layout[0][1].update(invitado['monedas'])
            else:
                #agregar_movimiento(k-1,l,nombreJuego,usuario_global['id'])
                if(window.Title == "Jugador 2"):
                    usuario_global['fallos'] = int(usuario_global['fallos']) + 1
```

```
else:
    sg.popup_ok('Turno del ' + window.Title)
    window.hide()
    crear_tablero_con_matriz(matriz2, window2, layout2,matriz,window,layout)
break
```

del contrincante para que se puedan ejecutar los ataques.

Al finalizar el juego sin importar el quien gane o pierda se muestran los resultados de la partida, los puntos, los ataques, la lista de adyacencia y el grafo de los ataques, todos estos datos son unicamente del jugador 1, que es el jugador con sesión iniciada.

```
sg.popup('El ganador es ' + window2.Title)
#Crear graphviz con datos de la partida
try:
    f = open('resultado.dot', 'w')
    f.write('digraph G {\n')
    f.write('rankdir=LR;\n')
    f.write('node [shape=box];\n')
    f.write('a [label="{} ha ganado: {} monedas \\n Y tuvo {} fallos"];\n'.format(
        usuario_global['nick'], str(int(usuario_global['monedas']) - int(usuario_global['puntos'])),
        str(usuario_global['fallos'])))
    f.write('b [label="Jugador 2 ha ganado: {} monedas \\n Y tuvo {} fallos"];\n'.format(
        str(int(invitado['monedas']) - int(invitado['puntos'])), str(invitado['fallos'])))

    f.write('a;\n')
    f.write('b;\n')
    f.write('}')
    f.close()
    os.system('dot -Tpng resultado.dot -o resultado.png')
except:
    pass
```

```
layout = [
            [
                [sg.Image(data = bio.getvalue(), key='imagen'),
                sg.Image(data = bio2.getvalue(), key='imagen2')],
                [sg.Image(data = bio3.getvalue(), key='imagen3'),
                sg.Image(data = bio4.getvalue(), key='imagen4')]

            ]
        ]

window3 = sg.Window('Tablero', layout, element_justification='c')

while True:
    event, values = window3.read()
    if event == sg.WIN_CLOSED:
        break
```

Si antes de que alguien gane el juego se decide abandonar se descontaran los puntos del jugador que abandonó el juego.

```
try:
    matriz.graficarNeato('1vs1')
    if(window.Title == "Jugador 2"):
        usuario_global['monedas'] -= 20
    sg.popup_ok('El juego ha terminado')
    sg.popup_ok(usuario_global['nick'] + ' ' + 'Ha ganado: ' +
        str(int(usuario_global['monedas']) - int(usuario_global['puntos'])) + ' monedas')
    sg.popup_ok('El jugador 2 ha ganado: ' + str(int(invitado['monedas']) - int(invitado['puntos'])) + ' monedas')
    break
except:
    pass
```

El administrador y el usuario pueden ver el reporte de la ultima partida que fue realizada.

```
    window.un_hide()
if event == 'Reporte de ultima partida':
    try:
        image = Image.open('./matriz_Tablero.png')
        image = image.resize((500,500),Image.Resampling.LANCZOS)

        bio = io.BytesIO()
        image.save(bio, format='PNG')

        image2 = Image.open('./grafo.png')
        image2 = image2.resize((500,500),Image.Resampling.LANCZOS)

        bio2 = io.BytesIO()
        image2.save(bio2, format='PNG')

        image3 = Image.open('./lista.png')
        image3 = image3.resize((500,500),Image.Resampling.LANCZOS)

        bio3 = io.BytesIO()
        image3.save(bio3, format='PNG')
```

```
image4 = Image.open('./resultado.png')
image4 = image4.resize((500,500),Image.Resampling.LANCZOS)

bio4 = io.BytesIO()
image4.save(bio4, format='PNG')

layout = [
            [
                [sg.Image(data = bio.getvalue(), key='imagen'),
                sg.Image(data = bio2.getvalue(), key='imagen2')],
                [sg.Image(data = bio3.getvalue(), key='imagen3'),
                sg.Image(data = bio4.getvalue(), key='imagen4')]

            ]
        ]

window3 = sg.Window('Tablero', layout, element_justification='c')

while True:
    event, values = window3.read()
    if event == sg.WIN_CLOSED:
```

En esta parte se recuperan las imágenes creadas de la ultima partida y se muestran en una ventana.

Para mostrar los resultados de los disparos realizados se guardan en una lista de adyacencia donde primero se verifica que sea el jugador 1 el que esté atacando y después de eso se agre
ga la posición del disparo para poder ser graficada utilizando graphviz y poder mostrarse.

```
if window.Title == "Jugador 2":
    disparos.conexion(k-1,l)
```

Para imprimir se recorre la lista y se va escribiendo un archivo .dot para ser convertido en un .png que será mostrado en los reportes.

```
def imprimirGraphviz(self):
    f = open('grafo.dot','w')
    f.write('digraph G {\n')
    f.write('bgcolor="#5DA7DB"\n')
    f.write('node [ style=filled,shape = oval, fillcolor="lightblue:lightblue1"]\n')
    for i in self.v:
        aux = i.inicio
        while aux != None:
            f.write(str(i.dato) + '->' + str(self.v[aux.index].dato) + ';\n')
            aux = aux.siguiente
    f.write('}')
    f.close()
    os.system('dot -Tpng grafo.dot -o grafo.png')
```

```
def imprimirLista(self):
    f = open('lista.dot','w')
    f.write('digraph G {\n')
    f.write('graph [rankdir = LR ]\n')
    f.write('nodesep = 0\n')
    f.write('bgcolor="#5DA7DB"\n')
    f.write('node [ style=filled,shape = box, fillcolor="lavenderblush:lavenderblush1"]\n')
    rank = '{rank = same; '
    label = ''
    apuntador = ''
    for i in self.v:
        rank += "\"" + "i" + str(i.dato) + "i\" "
        label += "\"" + "i" + str(i.dato) + "i\" " + "[label = \"" + str(i.dato) + "\"]\n"
        apuntador += "\"" + "i" + str(i.dato) + "i\" " + "->"
    f.write(rank + '};\n')
    f.write(label + '\n')
    apuntador = apuntador[:-2]
    f.write(apuntador + ' [arrowhead = none]\n')
    for i in self.v:
        aux = i.inicio
        apuntador = ''
        count = 0
```

```
    while aux != None:
        f.write('i' + str(i.dato) + 'i' + str(self.v[aux.index].dato) + ' [label = \"' + str(self.v[aux.index].dato)
        apuntador += 'i' + str(i.dato) + 'i' + str(self.v[aux.index].dato) + '->'
        aux = aux.siguiente
        count += 1

    if count > 0:
        apuntador = apuntador[:-2]
        f.write( 'i' + str(i.dato) + 'i->' + apuntador + '\n')


f.write('}')
f.close()
os.system('dot -Tpng lista.dot -o lista.png')
```