UNIVERSIDAD DE GUADALAJARA

CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍAS

Sem. Algoritmia

Reporte de práctica

Nombre del alumno:	Ricardo David Lopez Arellano
Profesor:	Erasmo Gabriel Martınez Soltero
Título de la práctica:	“Tarea 11. Dijkstra”
Fecha:	27 abril 2023

Introducción

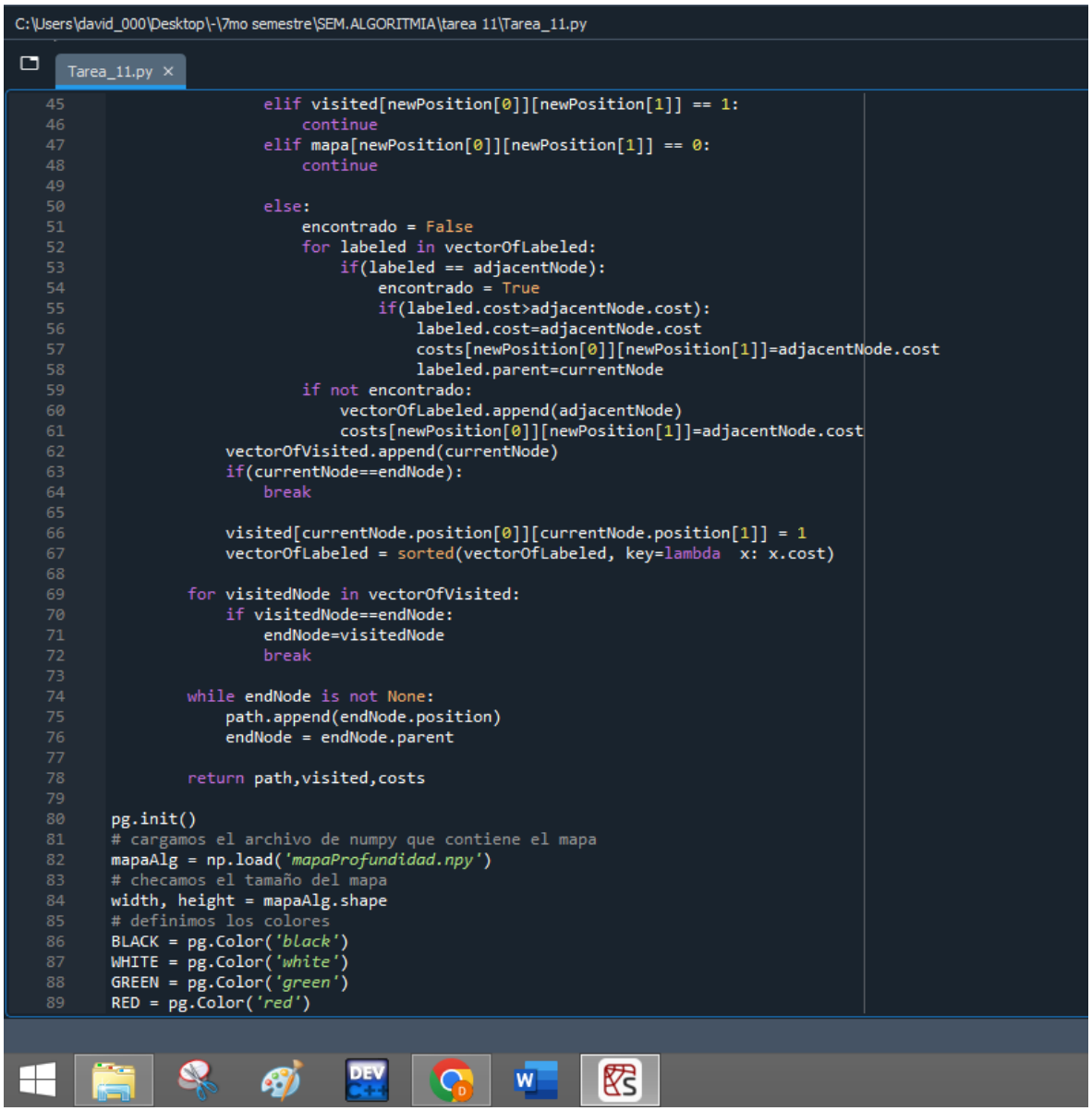
En esta practica se realizo el algoritmo de dijkstra.

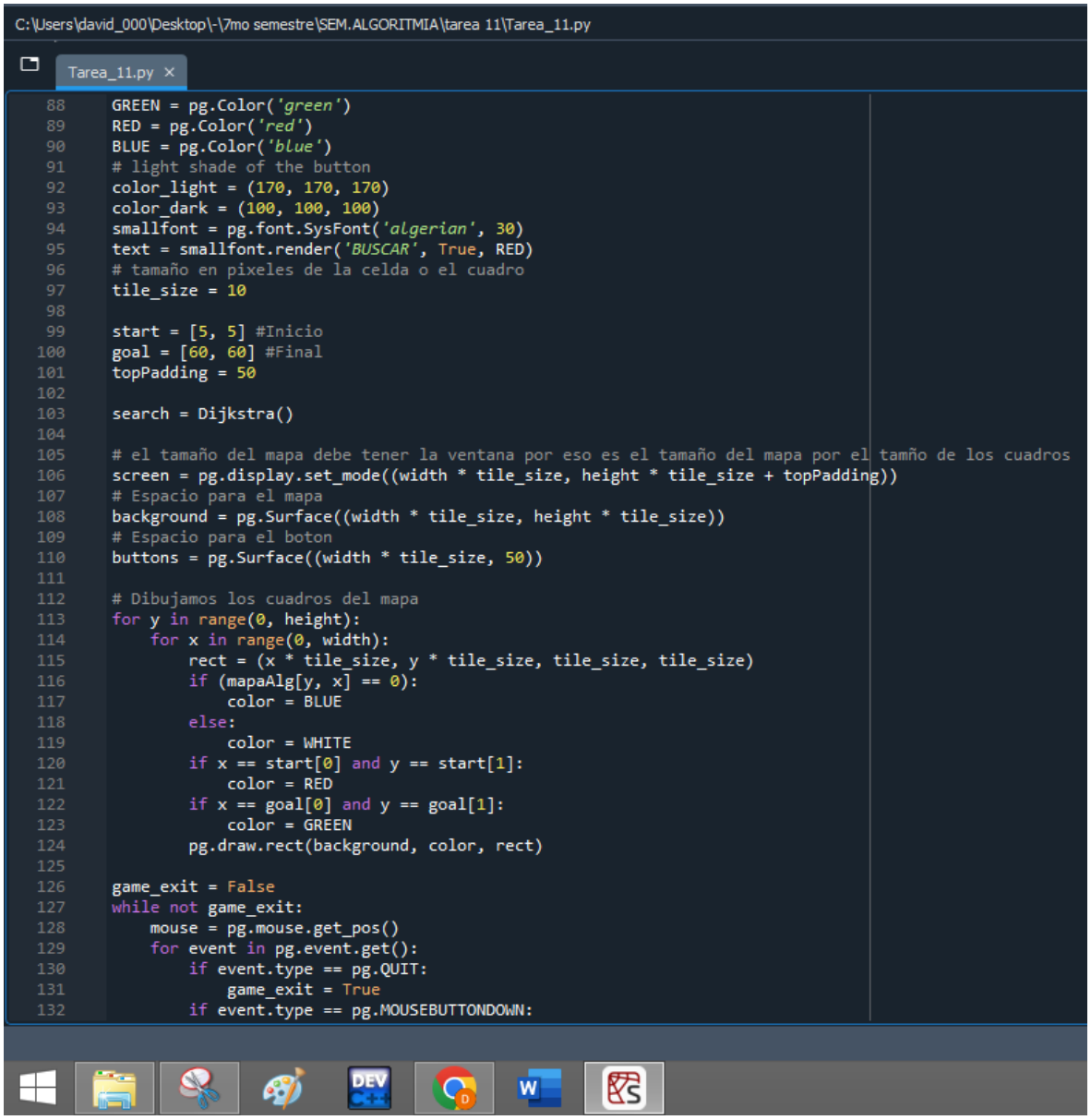
Metodología

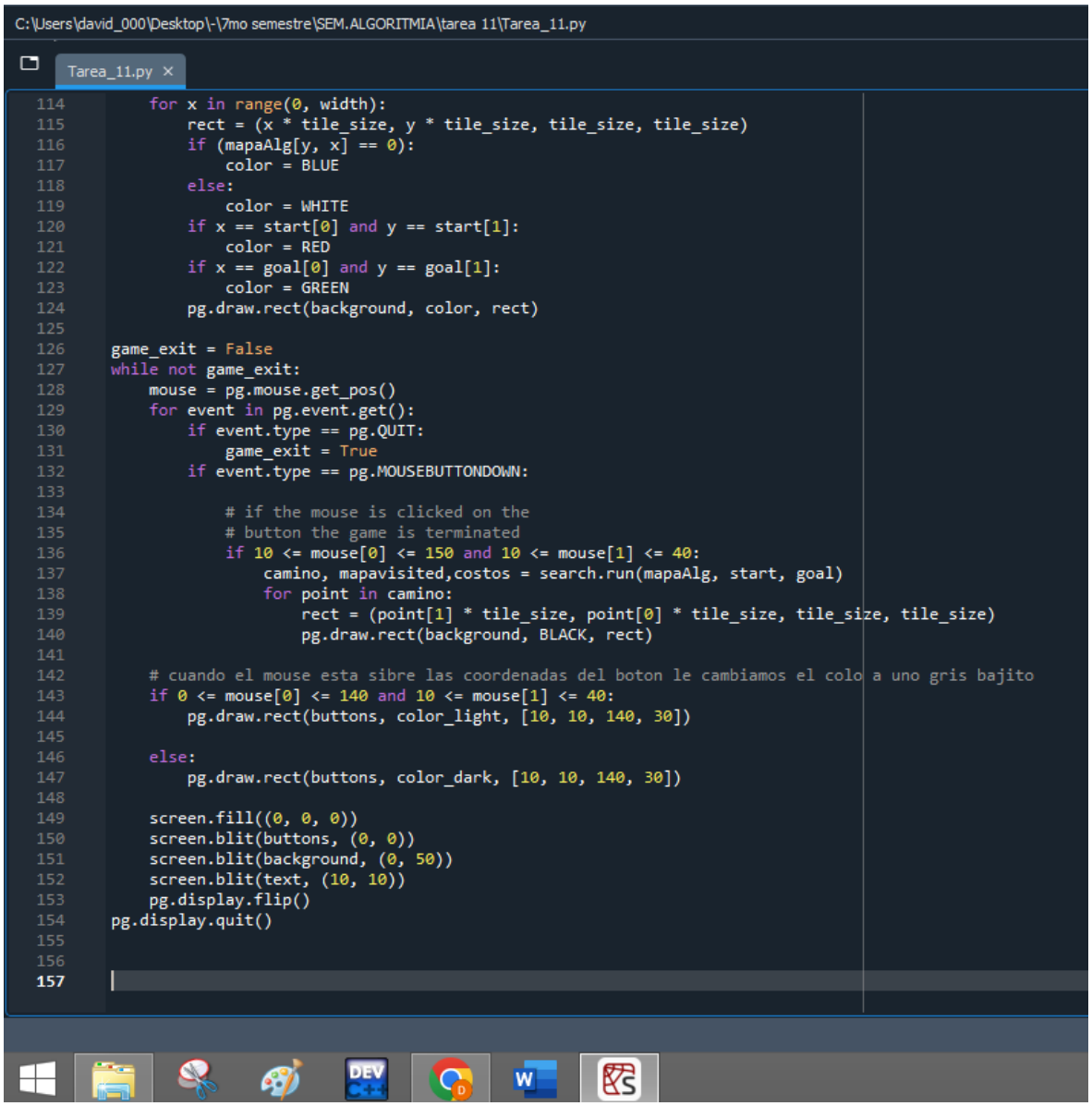
C:\Users\david_000\Desktop\-\7mo semestre\SEM.ALGORITMIA\tarea 11\Tarea_11.py

Tarea_11.py

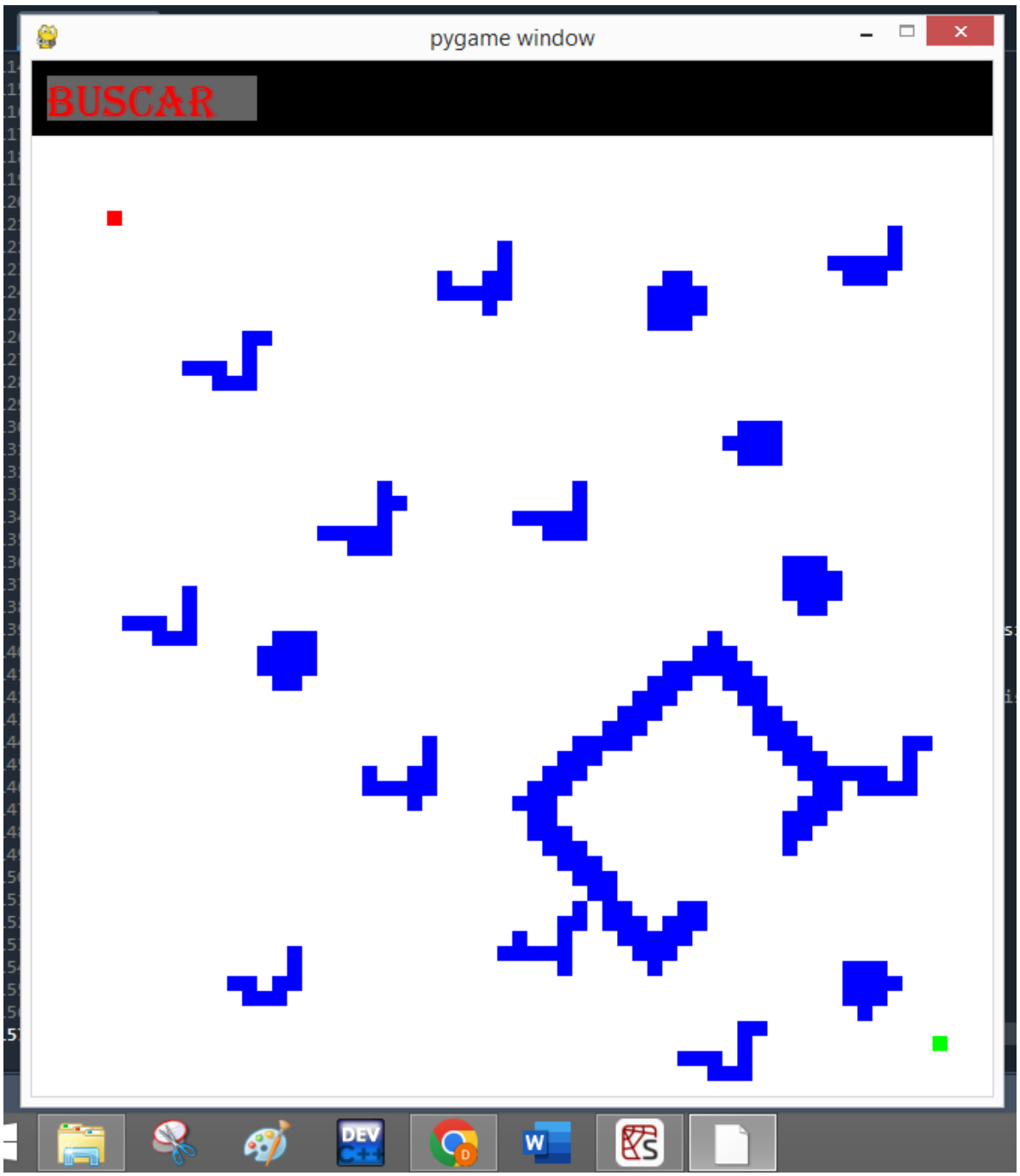
```python
import pygame as pg
import numpy as np

class MapaNode:
    def __init__(self, position,cost, parent=None):
        self.parent = parent
        self.position = position
        self.cost = cost

    def __eq__(self, other):
        return self.position[0] == other.position[0] and self.position[1] == other.position[1]

class Dijkstra(object):
    def run(self, mapa, start, end):
        mapa = mapa.astype(np.float64)
        unique, counts = np.unique(mapa, return_counts=True)
        nodosEnUno = counts[1]
        path = []
        vectorOfVisited = []
        vectorOfLabeled = []
        mapaRows, mapaCols = np.shape(mapa)
        visited = np.zeros(mapa.shape)
        costs = np.zeros(mapa.shape)
        vectorOfLabeled.append(MapaNode(start[::-1],0))
        endNode = MapaNode(end[::-1],0)
        while (len(vectorOfVisited) != nodosEnUno):
            currentNode = vectorOfLabeled.pop(0)

            movements = [[-1, -1, 1.4],
                         [0, -1, 1],
                         [1, -1, 1.4],
                         [-1, 0, 1],
                         [1, 0, 1],
                         [-1, 1, 1.4],
                         [0, 1, 1],
                         [1, 1, 1.4]
                         ]

            for movement in movements:
                newPosition = [currentNode.position[0] + movement[0], currentNode.position[1] + movement[1]]
                adjacentNode = MapaNode(newPosition,currentNode.cost+movement[2],currentNode)

                if newPosition[0] < 0 or newPosition[1] < 0 or newPosition[1] >= mapaCols or newPosition[0] >= mapaRows:
                    continue
                elif visited[newPosition[0]][newPosition[1]] == 1:
```

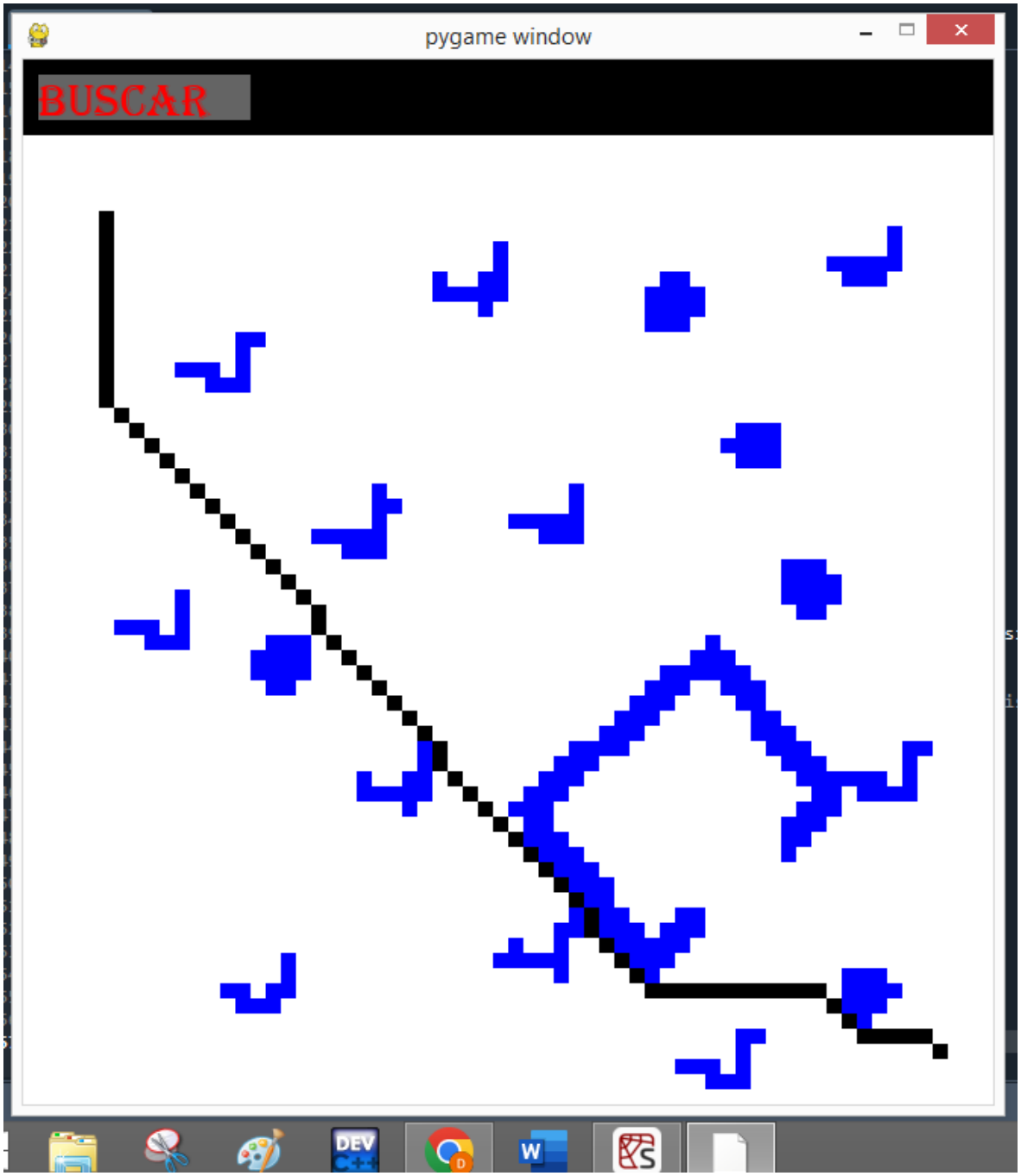

C:\Users\david_000\Desktop\-\7mo semestre\SEM.ALGORITMIA\tarea 11\Tarea_11.py
Tarea_11.py
45 elif visited[newPosition[0]][newPosition[1]] == 1:
46 continue
47 elif mapa[newPosition[0]][newPosition[1]] == 0:
48 continue
49
50 else:
51 encontrado = False
52 for labeled in vectorOfLabeled:
53 if(labeled == adjacentNode):
54 encontrado = True
55 if(labeled.cost>adjacentNode.cost):
56 labeled.cost=adjacentNode.cost
57 costs[newPosition[0]][newPosition[1]]=adjacentNode.cost
58 labeled.parent=currentNode
59 if not encontrado:
60 vectorOfLabeled.append(adjacentNode)
61 costs[newPosition[0]][newPosition[1]]=adjacentNode.cost
62 vectorOfVisited.append(currentNode)
63 if(currentNode==endNode):
64 break
65
66 visited[currentNode.position[0]][currentNode.position[1]] = 1
67 vectorOfLabeled = sorted(vectorOfLabeled, key=lambda x: x.cost)
68
69 for visitedNode in vectorOfVisited:
70 if visitedNode==endNode:
71 endNode=visitedNode
72 break
73
74 while endNode is not None:
75 path.append(endNode.position)
76 endNode = endNode.parent
77
78 return path,visited,costs
79
80 pg.init()
81 # cargamos el archivo de numpy que contiene el mapa
82 mapaAlg = np.load('mapaProfundidad.npy')
83 # checamos el tamaño del mapa
84 width, height = mapaAlg.shape
85 # definimos los colores
86 BLACK = pg.Color('black')
87 WHITE = pg.Color('white')
88 GREEN = pg.Color('green')
89 RED = pg.Color('red')


C:\Users\david_000\Desktop\-\7mo semestre\SEM.ALGORITMIA\tarea 11\Tarea_11.py
Tarea_11.py
GREEN = pg.Color('green')
RED = pg.Color('red')
BLUE = pg.Color('blue')
# light shade of the button
color_light = (170, 170, 170)
color_dark = (100, 100, 100)
smallfont = pg.font.SysFont('algerian', 30)
text = smallfont.render('BUSCAR', True, RED)
# tamaño en pixeles de la celda o el cuadro
tile_size = 10
start = [5, 5] #Inicio
goal = [60, 60] #Final
topPadding = 50
search = Dijkstra()
# el tamaño del mapa debe tener la ventana por eso es el tamaño del mapa por el tamño de los cuadros
screen = pg.display.set_mode((width * tile_size, height * tile_size + topPadding))
# Espacio para el mapa
background = pg.Surface((width * tile_size, height * tile_size))
# Espacio para el boton
buttons = pg.Surface((width * tile_size, 50))
# Dibujamos los cuadros del mapa
for y in range(0, height):
for x in range(0, width):
rect = (x * tile_size, y * tile_size, tile_size, tile_size)
if (mapaAlg[y, x] == 0):
color = BLUE
else:
color = WHITE
if x == start[0] and y == start[1]:
color = RED
if x == goal[0] and y == goal[1]:
color = GREEN
pg.draw.rect(background, color, rect)
game_exit = False
while not game_exit:
mouse = pg.mouse.get_pos()
for event in pg.event.get():
if event.type == pg.QUIT:
game_exit = True
if event.type == pg.MOUSEBUTTONDOWN:


C:\Users\david_000\Desktop\-\7mo semestre\SEM.ALGORITMIA\tarea 11\Tarea_11.py
Tarea_11.py
for x in range(0, width):
rect = (x * tile_size, y * tile_size, tile_size, tile_size)
if (mapaAlg[y, x] == 0):
color = BLUE
else:
color = WHITE
if x == start[0] and y == start[1]:
color = RED
if x == goal[0] and y == goal[1]:
color = GREEN
pg.draw.rect(background, color, rect)
game_exit = False
while not game_exit:
mouse = pg.mouse.get_pos()
for event in pg.event.get():
if event.type == pg.QUIT:
game_exit = True
if event.type == pg.MOUSEBUTTONDOWN:
# if the mouse is clicked on the
# button the game is terminated
if 10 <= mouse[0] <= 150 and 10 <= mouse[1] <= 40:
camino, mapavisited,costos = search.run(mapaAlg, start, goal)
for point in camino:
rect = (point[1] * tile_size, point[0] * tile_size, tile_size, tile_size)
pg.draw.rect(background, BLACK, rect)
# cuando el mouse esta sibre las coordenadas del boton le cambiamos el colo a uno gris bajito
if 0 <= mouse[0] <= 140 and 10 <= mouse[1] <= 40:
pg.draw.rect(buttons, color_light, [10, 10, 140, 30])
else:
pg.draw.rect(buttons, color_dark, [10, 10, 140, 30])
screen.fill((0, 0, 0))
screen.blit(buttons, (0, 0))
screen.blit(background, (0, 50))
screen.blit(text, (10, 10))
pg.display.flip()
pg.display.quit()

Resultados





Conclusiones

En esta actividad se logro la practica como se deseaba, ya que con la explicación del video del profe en su canal fui haciendolo a la par.

Link del video:

https://youtu.be/npxpeUTqwSo