Міністерство освіти і науки України

Національний технічний університет України «КПІ ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Програмування інтелектуальних інформаційних систем

**ЗВІТ**

до лабораторних робіт

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Виконав**  **студент** |  | Бакун Нікіта Валерійович |  |  |
|  |  | (№ групи, прізвище, ім’я, по батькові ) |  |  |
|  |  |  |  |  |
| **Прийняв** |  | ас. Очеретяний О. К. |  |  |
|  |  | (посада, прізвище, ім’я, по батькові ) |  |  |

Київ 2021

# **Завдання лабораторної роботи**

Завданням даної лаболаторної роботи було створення гри Space Inviders з такими етапами:

* dfs
* bfs
* ucs
* Створення оцінки часу
* Астероїди

# **Опис використаних технологій**

В даній роботі використано мову Python та алгоритми, з якими ми ознайомлювалися на попередніх лекціях  
**Пошук в глибину (англ. Depth-first search, DFS) -** один з методів обходу графа. Стратегія пошуку в глибину, як і випливає з назви, полягає в тому, щоб йти «вглиб» графа, наскільки це можливо. Алгоритм пошуку описується рекурсивно: перебираємо всі вихідні з даної вершини ребра. Якщо ребро веде в вершину, яка не була розглянута раніше, то запускаємо алгоритм від цієї нерозглянутих вершини, а після повертаємося і продовжуємо перебирати ребра. Повернення відбувається в тому випадку, якщо в даній вершині не залишилося ребер, які ведуть в нерозглянутих вершину. Якщо після завершення алгоритму не всі вершини були розглянуті, то необхідно запустити алгоритм від однієї з нерозглянутих вершин.

**Пошук в ширину (англ. Breadth-first search, BFS) -** один з методів обходу графа. Нехай заданий граф G = (V, E)} G = (V, E) і виділена вихідна вершина s . Алгоритм пошуку в ширину систематично обходить всі ребра G для «відкриття» всіх вершин, досяжних з s, обчислюючи при цьому відстань (мінімальна кількість ребер) від s до кожної досяжною з s вершини. Алгоритм працює як для орієнтованих, так і для неорієнтованих графів.

Пошук в ширину має таку назву тому, що в процесі обходу ми йдемо вшир, тобто перед тим як приступити до пошуку вершин на відстані k + 1, виконується обхід вершин на відстані k.

**Рівномірний пошук вартост**і-це варіант алгоритму Діджикстри. Тут, замість того, щоб вставляти всі вершини до черги пріоритетів, ми вставляємо лише джерело, потім по одній вставку, коли це необхідно. На кожному кроці ми перевіряємо, чи елемент вже знаходиться у черзі пріоритетів (за допомогою відвіданого масиву). Якщо так, ми виконуємо клавішу зменшення, інакше вставляємо її.

Цей варіант Дейкстри корисний для нескінченних графіків і тих графіків, які занадто великі, щоб представити їх у пам'яті. Уніфікований пошук витрат в основному використовується в штучному інтелекті.

# **Опис програмного коду**

В головному файлі проекту main.py визиваються по натиску клавіші алгоритми та відбувається зміна алгоритмів

import GlobalVariables as gv  
  
  
def main():  
 gv.CAN\_EDIT = True  
 gv.PG\_LIB.display.set\_caption(**"Space Invaders"**) *# Назва застосунку* run = True  
 clock = gv.PG\_LIB.time.Clock()  
 while run:  
 clock.tick(gv.FPS) *#Виставлено кількість оновлень у секунду* gv.FrameCreator\_LIB.updateFrame()  
 if gv.LIVES <= 0 or gv.GOOD\_SHIP.health <= 0: *# умови програшу* gv.LOST = True  
 gv.LOST\_COUNT += 1  
  
 if gv.LOST: *# час показання екрану програшу* if gv.LOST\_COUNT > gv.FPS \* 3:  
 run = False  
  
 else:  
 continue  
 if len(gv.ENEMIES) == 0: *# збільшення рівня складності* gv.LEVEL += 1  
 gv.WAVE\_LENGTH += 5  
 for i in range(gv.WAVE\_LENGTH): *# створення масиву з ворогів* enemy = gv.ShipCreator.Enemy(gv.RANDOM\_LIB.randrange(50, gv.WIDTH - 100),  
 gv.RANDOM\_LIB.randrange(-1500, -100),  
 gv.RANDOM\_LIB.choice([**"red"**, **"blue"**, **"purple"**])) *# створення місця та колір ворога* gv.ENEMIES.append(enemy)  
 for event in gv.PG\_LIB.event.get():  
 if event.type == gv.PG\_LIB.QUIT:  
 run = False  
  
 keys = gv.PG\_LIB.key.get\_pressed() *# рух корабля по натиску на клавішу та стрільба* if keys[gv.PG\_LIB.K\_a] and gv.GOOD\_SHIP.x - gv.PLAYER\_VEL > 0: *# left* gv.GOOD\_SHIP.x -= gv.PLAYER\_VEL  
 if keys[gv.PG\_LIB.K\_d] and gv.GOOD\_SHIP.x + gv.PLAYER\_VEL + gv.GOOD\_SHIP\_SIZEX < gv.WIDTH: *# right* gv.GOOD\_SHIP.x += gv.PLAYER\_VEL  
 if keys[gv.PG\_LIB.K\_w] and gv.GOOD\_SHIP.y - gv.PLAYER\_VEL > 0: *# up* gv.GOOD\_SHIP.y -= gv.PLAYER\_VEL  
 if keys[gv.PG\_LIB.K\_s] and gv.GOOD\_SHIP.y + gv.PLAYER\_VEL + gv.GOOD\_SHIP\_SIZEY < gv.HEIGHT: *# down* gv.GOOD\_SHIP.y += gv.PLAYER\_VEL  
 if keys[gv.PG\_LIB.K\_SPACE]:  
 gv.GOOD\_SHIP.shoot()  
 for enemy in gv.ENEMIES[:]:  
 enemy.move(gv.ENEMY\_VEL)  
 enemy.move\_lasers(gv.LASER\_VEL, gv.GOOD\_SHIP)  
  
 if gv.RANDOM\_LIB.randrange(0, 20) == 1: *# рандомізація пострілу ворога* enemy.shoot()  
  
 if gv.LaserCreator.collide(enemy, gv.GOOD\_SHIP): *# дотик до ворога* gv.GOOD\_SHIP.health -= 10  
 gv.ENEMIES.remove(enemy)  
 gv.SCORE -= 20  
  
 if enemy.y + gv.BAD\_SHIP\_SIZEY + 10 > gv.HEIGHT: *# проходження ворогу до низу екрану* gv.LIVES -= 1  
 gv.ENEMIES.remove(enemy)  
 gv.SCORE -= 100  
  
 gv.GOOD\_SHIP.move\_lasers(-gv.LASER\_VEL, gv.ENEMIES)  
  
 gv.FrameCreator\_LIB.updateFrame()  
  
  
if \_\_name\_\_ == **'\_\_main\_\_'**:  
 title\_font = gv.PG\_LIB.font.SysFont(**"comicsans"**, 70)  
 run = True  
 main\_menu = True  
 while run:  
 gv.WINDOW.blit(gv.BACKGROUND\_PNG, (0, 0)) *#створення графіки для гри* title\_label = title\_font.render(**"Press any button to begin..."**, 1, (255, 255, 255))  
  
 gv.WINDOW.blit(title\_label, (gv.WIDTH / 2 - title\_label.get\_width() / 2,350))  
 if main\_menu:  
 main\_menu = False  
  
 gv.PG\_LIB.display.update()  
 for event in gv.PG\_LIB.event.get():  
 if event.type == gv.PG\_LIB.QUIT:  
 run = False  
 if event.type == gv.PG\_LIB.KEYDOWN or event.type == gv.PG\_LIB.MOUSEBUTTONDOWN:  
 main()  
 gv.PG\_LIB.quit()

У файлі LaserCreator.py реалізовано клас лазер та показано методи та поля даного класу. Змім відносно попередної лр не відбулося

import GlobalVariables  
  
  
class Laser:  
 def \_\_init\_\_(self, x, y, img):  
 self.x = x  
 self.y = y  
 self.img = img  
 self.mask = GlobalVariables.PG\_LIB.mask.from\_surface(self.img)  
  
 def draw(self, window): *# створення лазеру* window.blit(self.img, (self.x, self.y))  
  
 def move(self, vel): *# рух лазеру* self.y += vel  
  
 def off\_screen(self, height): *# вихід лазеру за рамки* return not (self.y <= height and self.y >= 0)  
  
 def collision(self, obj): *# дотикання до лазеру* return collide(self, obj)  
  
  
def collide(obj1, obj2): *# чи дотикнулись об'єкти* offset\_x = obj2.x - obj1.x  
 offset\_y = obj2.y - obj1.y  
 return obj1.mask.overlap(obj2.mask, (offset\_x, offset\_y)) is not None

У файлі GlobalVariables.py розміщені усі глобальні змінні. Вони винесені в окремий файл для зручного використання та доступу. В подальшому таке розміщення повинно спростити роботу з проектом. У даній лабораторній роботі додано деякі змінні

import pygame  
import os  
import time  
import random  
import FrameCreator  
import ShipCreator as sc  
import LaserCreator  
*# created global variables of libraries*OS\_LIB = os  
PG\_LIB = pygame  
TIME\_LIB = time  
RANDOM\_LIB = random  
FrameCreator\_LIB = FrameCreator  
ShipCreator = sc  
LaserCreator = LaserCreator  
*# Created global variables of Integer values*LOST = False  
FPS = 60  
LEVEL = 1  
LIVES = 5  
SCORE = 0  
WIDTH, HEIGHT = 750, 750  
PLAYER\_VEL = 5  
LASER\_VEL = 4  
ENEMIES = []  
WAVE\_LENGTH = 5  
ENEMY\_VEL = 1  
LOST\_COUNT = 0  
GOOD\_SHIP\_SIZEX, GOOD\_SHIP\_SIZEY = 50, 50  
BAD\_SHIP\_SIZEX, BAD\_SHIP\_SIZEY = 40, 40  
LASER\_SIZEX = 10  
LASER\_SIZEY = 20  
*# Created global variables of Font*PG\_LIB.font.init()  
MAIN\_FONT = PG\_LIB.font.SysFont(**"comicsans"**, 50)  
LOST\_FONT = PG\_LIB.font.SysFont(**"comicsans"**, 60)  
  
WINDOW = PG\_LIB.display.set\_mode((WIDTH, HEIGHT))  
  
*# Created global variables of boats images*GOOD\_SHIP\_PNG = PG\_LIB.transform.scale(PG\_LIB.image.load(os.path.join(**"assets"**, **"good\_ship.png"**)), (GOOD\_SHIP\_SIZEX, GOOD\_SHIP\_SIZEY))  
  
BAD\_SHIP\_RED\_PNG = PG\_LIB.transform.scale(PG\_LIB.image.load(os.path.join(**"assets"**, **"bad\_ship\_red.png"**)), (BAD\_SHIP\_SIZEX, BAD\_SHIP\_SIZEX))  
BAD\_SHIP\_BLUE\_PNG = PG\_LIB.transform.scale(PG\_LIB.image.load(os.path.join(**"assets"**, **"bad\_ship\_blue.png"**)), (BAD\_SHIP\_SIZEX, BAD\_SHIP\_SIZEX))  
BAD\_SHIP\_PURPLE\_PNG = PG\_LIB.transform.scale(PG\_LIB.image.load(os.path.join(**"assets"**, **"bad\_ship\_purple.png"**)), (BAD\_SHIP\_SIZEX, BAD\_SHIP\_SIZEX))  
  
BAD\_BULLET\_PNG = PG\_LIB.transform.scale(PG\_LIB.image.load(os.path.join(**"assets"**, **"bad\_bullet.png"**)), (LASER\_SIZEX, LASER\_SIZEY))  
GOOD\_BULLET\_PNG = PG\_LIB.transform.scale(PG\_LIB.image.load(os.path.join(**"assets"**, **"good\_bullet.png"**)), (LASER\_SIZEX, LASER\_SIZEY))  
  
*# Created global variables of background images*BACKGROUND\_PNG = PG\_LIB.transform.scale(PG\_LIB.image.load(os.path.join(**"assets"**, **"background.png"**)), (WIDTH, HEIGHT))  
  
*# Created ships*GOOD\_SHIP = sc.Player(300, 650)  
  
COLOR\_MAP = {  
 **"red"**: BAD\_SHIP\_RED\_PNG,  
 **"blue"**: BAD\_SHIP\_BLUE\_PNG,  
 **"purple"**: BAD\_SHIP\_PURPLE\_PNG  
}

У файлі ShipCrearor.py реалізовано клас астероїдів.

import GlobalVariables as gv  
  
  
class Ship:  
 COOLDOWN = 30 *# час перезарядки човна* def \_\_init\_\_(self, x, y, health=100):  
 self.x = x  
 self.y = y  
 self.health = health  
 self.ship\_img = None  
 self.laser\_img = None  
 self.lasers = []  
 self.cool\_down\_counter = 0  
  
 def draw(self, window): *# створення човна* gv.WINDOW.blit(self.ship\_img, (self.x, self.y))  
 for laser in self.lasers:  
 laser.draw(gv.WINDOW)  
  
 def move\_lasers(self, vel, obj): *# рух пострілів* self.coldown()  
 for laser in self.lasers:  
 laser.move(vel)  
 if laser.off\_screen(gv.HEIGHT):  
 self.lasers.remove(laser)  
 elif laser.collision(obj):  
 obj.health -= 10  
 self.lasers.remove(laser)  
  
 def shoot(self): *# постріл* if self.cool\_down\_counter == 0:  
 laser = gv.LaserCreator.Laser(self.x + 20, self.y, self.laser\_img)  
 self.lasers.append(laser)  
 self.cool\_down\_counter = 1  
  
 def coldown(self): *# механіка перезарядки* if self.cool\_down\_counter >= self.COOLDOWN:  
 self.cool\_down\_counter = 0  
 elif self.cool\_down\_counter >= 0:  
 self.cool\_down\_counter += 1  
  
  
class Player(Ship):  
 def \_\_init\_\_(self, x, y, health=100):  
 super().\_\_init\_\_(x, y, health)  
 self.ship\_img = gv.GOOD\_SHIP\_PNG  
 self.laser\_img = gv.GOOD\_BULLET\_PNG  
 self.mask = gv.PG\_LIB.mask.from\_surface(self.ship\_img)  
 self.max\_health = health  
  
 def move\_lasers(self, vel, objs): *#рух лазерів* self.coldown()  
 for laser in self.lasers:  
 laser.move(vel)  
 if laser.off\_screen(gv.HEIGHT):  
 self.lasers.remove(laser)  
 else:  
 for obj in objs:  
 if laser.collision(obj):  
 objs.remove(obj)  
 gv.SCORE += 50  
 if laser in self.lasers:  
 self.lasers.remove(laser)  
  
 def healthbar(self, window): *# створення здоров'я* gv.PG\_LIB.draw.rect(window, (255, 0, 0), (self.x, self.y + gv.GOOD\_SHIP\_SIZEY + 10, gv.GOOD\_SHIP\_SIZEX, 10))  
 gv.PG\_LIB.draw.rect(window, (0, 255, 0), (self.x, self.y + gv.GOOD\_SHIP\_SIZEY + 10, gv.GOOD\_SHIP\_SIZEX \* (  
 1 - ((self.max\_health - self.health) / self.max\_health)), 10))  
  
 def draw(self, window):  
 super().draw(window)  
 self.healthbar(window)  
  
  
class Enemy(Ship):  
 def \_\_init\_\_(self, x, y, color, health=100):  
 super().\_\_init\_\_(x, y, health)  
 self.ship\_img = gv.COLOR\_MAP[color]  
 self.laser\_img = gv.BAD\_BULLET\_PNG  
 self.mask = gv.PG\_LIB.mask.from\_surface(self.ship\_img)  
  
 def move(self, vel):  
 self.y += vel  
  
 def shoot(self):  
 if self.cool\_down\_counter == 0:  
 laser = gv.LaserCreator.Laser(self.x + 10, self.y, self.laser\_img)  
 self.lasers.append(laser)  
 self.cool\_down\_counter = 1

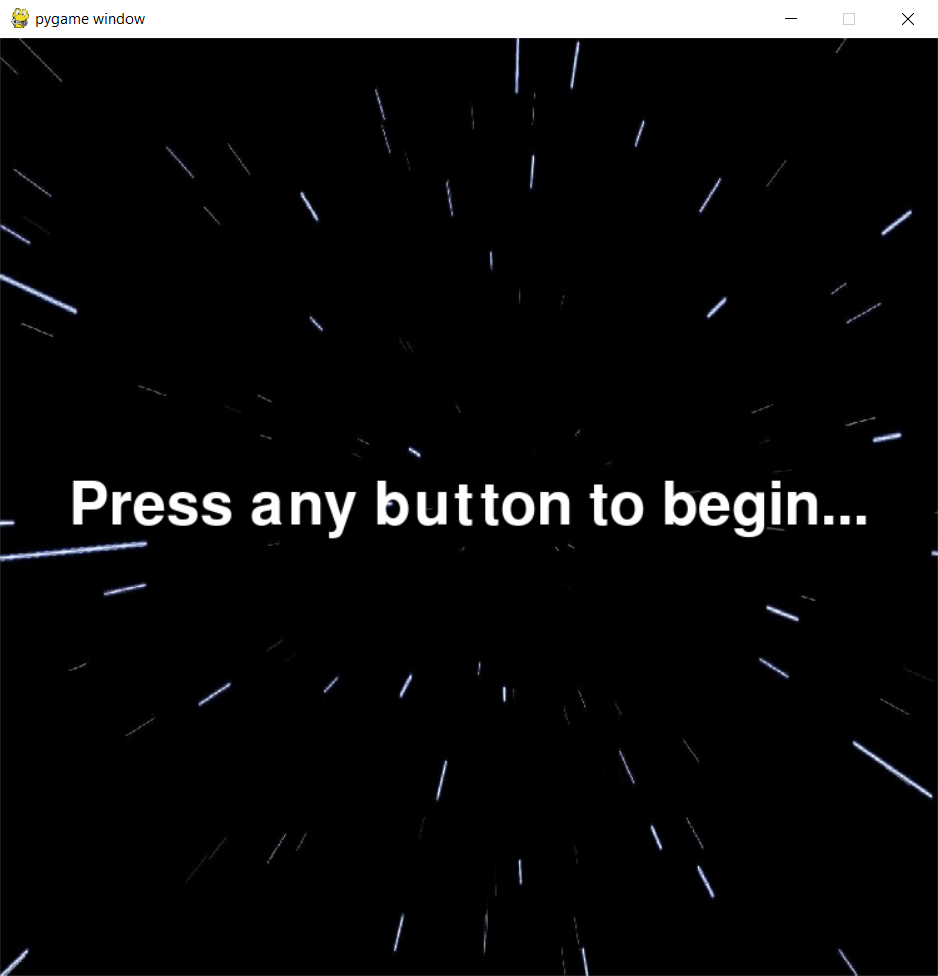
в FrameCreator.py змін не відбулося

import GlobalVariables as gv  
  
  
def updateFrame(): *# оновлення фрейму* gv.WINDOW.blit(gv.BACKGROUND\_PNG, (0, 0))  
  
 lives\_label = gv.MAIN\_FONT.render(**f"Lives:** {gv.LIVES}**"**, 1, (255, 255, 255))  
 level\_label = gv.MAIN\_FONT.render(**f"Level:** {gv.LEVEL}**"**, 1, (255, 255, 255))  
 score\_label = gv.MAIN\_FONT.render(**f"Score:** {gv.SCORE}**"**, 1, (255, 255, 255))  
 gv.WINDOW.blit(lives\_label, (10, 10))  
 gv.WINDOW.blit(level\_label, (gv.WIDTH - level\_label.get\_width() - 10, 10))  
 gv.WINDOW.blit(score\_label,(10,40))  
 for enemy in gv.ENEMIES:  
 enemy.draw(gv.WINDOW)  
  
 gv.GOOD\_SHIP.draw(gv.WINDOW)  
  
 if gv.LOST: *# програш* lost\_label = gv.LOST\_FONT.render(**f"YOU LOST!"**, 1, (255, 255, 255))  
 score\_label = gv.LOST\_FONT.render(**f"YOUR SCORE IS:** {gv.SCORE}**"**, 1, (255, 255, 255))  
 gv.WINDOW.blit(lost\_label, (gv.WIDTH / 2 - lost\_label.get\_width() / 2, 250))  
 gv.WINDOW.blit(score\_label, (gv.WIDTH / 2 - score\_label.get\_width() / 2, 350))  
  
 gv.PG\_LIB.display.update()

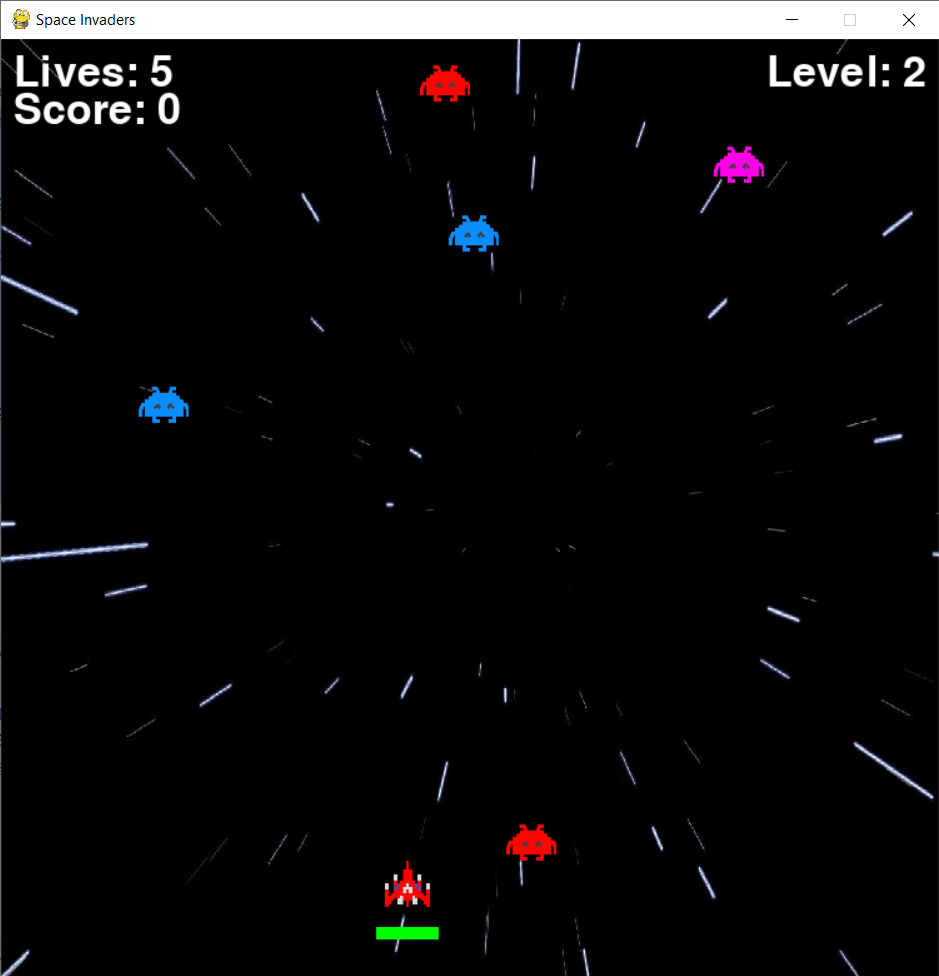
У Algoryhtmms.py реалізовано усі алгоритми

# **Скріншоти роботи програмного застосунку**

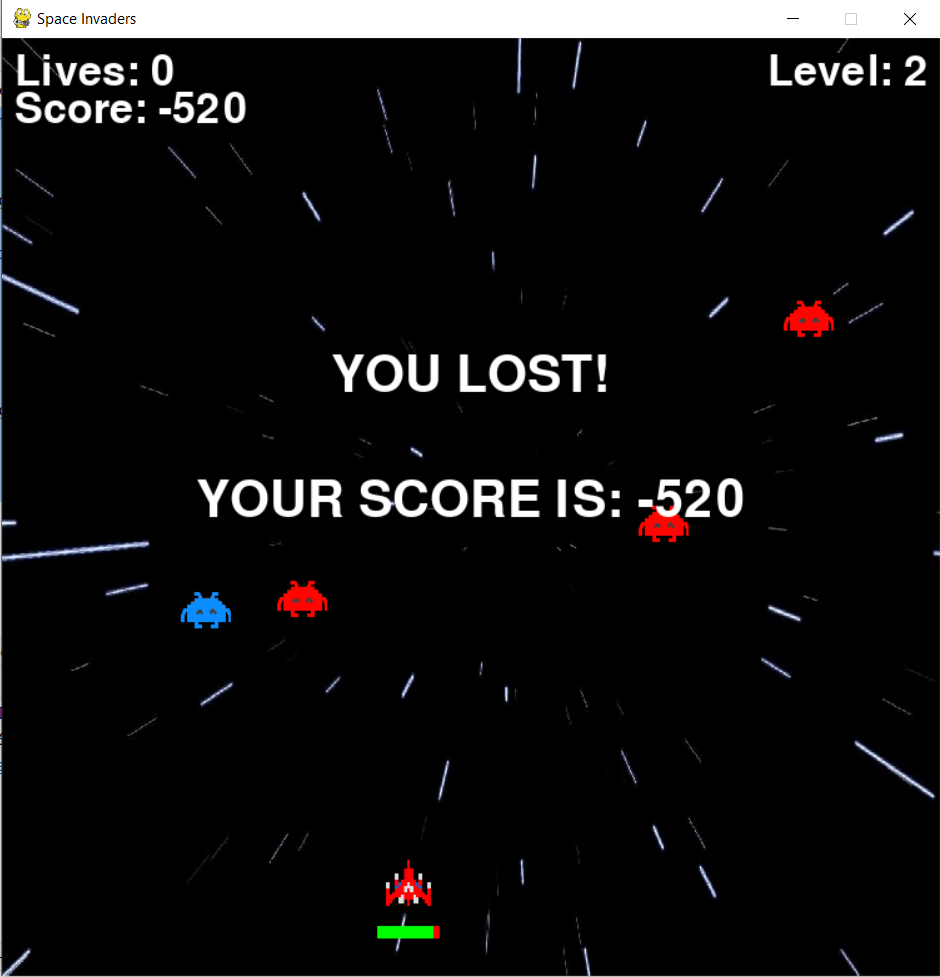
import GlobalVariables as gv  
import numpy  
  
*# empty = 0  
# current = 1  
# enemy = 2  
# asteroids = 3*matrix = numpy.full((int(750 / 50), int(750 / 50)), 0)  
visitMatrix = numpy.full((int(750 / 50), int(750 / 50)), 0)  
path = [[7, 14]]  
numofEnemy = 1 *# TODO: get num from matrix*arrOfPath = []  
listOfVisited = [[7, 14]]  
arrOfList = []  
arrBeforePath = []  
ucsListOfVisited = [[14, 6]]  
lenMatrix = numpy.full((int(750 / 50), int(750 / 50)), 0)  
ucsList = []  
arrUcsList = []  
enemyCoords = []  
  
  
def createStartMatrix():  
 for i in gv.ASTEROIDS:  
 for j in range(int((i.x - 20) / 50), int((i.x + 20) / 50)):  
 for k in range(int((i.y - 20) / 50), int((i.y + 20) / 50)):  
 if 0 < j < len(matrix) and 0 < k < len(matrix):  
 matrix[k][j] = 3  
 gv.VisitMatrix[k][j] = 1  
 for i in gv.ENEMIES:  
 for j in range(int((i.x - 20) / 50), int((i.x + 20) / 50)):  
 for k in range(int((i.y - 20) / 50), int((i.y + 20) / 50)):  
 if 0 < j < len(matrix) and 0 < k < len(matrix):  
 matrix[k][j] = 2  
 gv.VisitMatrix[k][j] = 1  
  
 *# global numofEnemy, path  
 # for i in matrix:  
 # for j in i:  
 # if j == 2:  
 # numofEnemy += 1  
 # if j == 1:  
 # path = [[int(gv.GOOD\_SHIP.x / 50), int(gv.GOOD\_SHIP.y / 50)]]* matrix[0][0] = 0  
 matrix[int(gv.currPoint[1])][int(gv.currPoint[0])] = 1  
 gv.VisitMatrix[int(gv.currPoint[1])][int(gv.currPoint[0])] = 1  
  
  
  
  
def createVisitMatrix(matrix, visitMatrix):  
 for i in range(len(matrix)):  
 for j in range(len(matrix[i])):  
 if matrix[i][j] != 0:  
 visitMatrix[i][j] = 1  
 else:  
 visitMatrix[i][j] = 0  
  
  
def dfs(matrix, visitMatrix, curX=path[-1][0], curY=path[-1][1]):  
 *# print(path)  
 # for i in visitMatrix:  
 # print(\*i)* visitMatrix[curX][curY] = 1  
 while len(path) > 0:  
 curX = path[-1][0]  
 curY = path[-1][1]  
 if curX + 1 < len(matrix) and matrix[curX + 1][curY] == 2:  
 path.append([curX + 1, curY])  
 break  
 elif curY + 1 < len(matrix) and matrix[curX][curY + 1] == 2:  
 path.append([curX, curY + 1])  
 break  
 elif curX - 1 >= 0 and matrix[curX - 1][curY] == 2:  
 path.append([curX - 1, curY])  
 break  
 elif curY - 1 >= 0 and matrix[curX][curY - 1] == 2:  
 path.append([curX, curY - 1])  
 break  
  
 step = False  
 *# print(path)  
 # for i in visitMatrix:  
 # print(\*i)* if curX + 1 < len(matrix) and visitMatrix[curX + 1][curY] == 0 and not step:  
 visitMatrix[curX + 1][curY] = 1  
 path.append([curX + 1, curY])  
 step = True  
 elif curY + 1 < len(matrix) and visitMatrix[curX][curY + 1] == 0 and not step:  
 visitMatrix[curX][curY + 1] = 1  
 path.append([curX, curY + 1])  
 step = True  
 elif curY - 1 >= 0 and visitMatrix[curX][curY - 1] == 0 and not step:  
 visitMatrix[curX][curY - 1] = 1  
 path.append([curX, curY - 1])  
 step = True  
 elif curX - 1 >= 0 and visitMatrix[curX - 1][curY] == 0 and not step:  
 visitMatrix[curX - 1][curY] = 1  
 path.append([curX - 1, curY])  
 step = True  
 elif not step:  
 path.remove(path[-1])  
  
  
def bfs(matrix, visitMatrix, curX=listOfVisited[-1][0], curY=listOfVisited[-1][1]):  
 listOfVisited.append([curX, curY])  
 while len(listOfVisited) > 0:  
 *#print(listOfVisited)  
 # for i in visitMatrix:  
 # print(\*i)* curX = listOfVisited[0][0]  
 curY = listOfVisited[0][1]  
 if curX + 1 < len(matrix) and matrix[curX + 1][curY] == 2:  
 listOfVisited.append([curX + 1, curY])  
 break  
 elif curY + 1 < len(matrix) and matrix[curX][curY + 1] == 2:  
 listOfVisited.append([curX, curY + 1])  
 break  
 elif curX - 1 >= 0 and matrix[curX - 1][curY] == 2:  
 listOfVisited.append([curX - 1, curY])  
 break  
 elif curY - 1 >= 0 and matrix[curX][curY - 1] == 2:  
 listOfVisited.append([curX, curY - 1])  
 break  
 step = False  
 *# print(path)  
 # for i in visitMatrix:  
 # print(\*i)* if curX + 1 < len(matrix) and visitMatrix[curX + 1][curY] == 0:  
 visitMatrix[curX + 1][curY] = 1  
 listOfVisited.append([curX + 1, curY])  
  
 if curY + 1 < len(matrix) and visitMatrix[curX][curY + 1] == 0:  
 visitMatrix[curX][curY + 1] = 1  
 listOfVisited.append([curX, curY + 1])  
  
 if curY - 1 >= 0 and visitMatrix[curX][curY - 1] == 0:  
 visitMatrix[curX][curY - 1] = 1  
 listOfVisited.append([curX, curY - 1])  
  
 if curX - 1 >= 0 and visitMatrix[curX - 1][curY] == 0:  
 visitMatrix[curX - 1][curY] = 1  
 listOfVisited.append([curX - 1, curY])  
 arrBeforePath.append(path[-1])  
 listOfVisited.remove(listOfVisited[0])  
  
  
  
def ucs(matrix, visitMatrix, curX=ucsListOfVisited[-1][0], curY=ucsListOfVisited[-1][1]):  
 ucsListOfVisited.append([curX, curY])  
 while len(ucsListOfVisited) > 0:  
 *# print(ucsListOfVisited)  
 # for i in visitMatrix:  
 # print(\*i)* curX = ucsListOfVisited[0][0]  
 curY = ucsListOfVisited[0][1]  
 if curX + 1 < len(matrix) and matrix[curX + 1][curY] == 2:  
 ucsListOfVisited.append([curX + 1, curY])  
 break  
 elif curY + 1 < len(matrix) and matrix[curX][curY + 1] == 2:  
 ucsListOfVisited.append([curX, curY + 1])  
 break  
 elif curX - 1 >= 0 and matrix[curX - 1][curY] == 2:  
 ucsListOfVisited.append([curX - 1, curY])  
 break  
 elif curY - 1 >= 0 and matrix[curX][curY - 1] == 2:  
 ucsListOfVisited.append([curX, curY - 1])  
 break  
 step = False  
 *# print(path)  
 # for i in visitMatrix:  
 # print(\*i)* if curX + 1 < len(matrix) and visitMatrix[curX + 1][curY] == 0:  
 if lenMatrix[curX + 1][curY] == 0:  
 lenMatrix[curX + 1][curY] = lenMatrix[curX][curY] + 1  
 elif lenMatrix[curX + 1][curY] > lenMatrix[curX][curY] + 1:  
 lenMatrix[curX + 1][curY] = lenMatrix[curX][curY] + 1  
 visitMatrix[curX + 1][curY] = 1  
 ucsListOfVisited.append([curX + 1, curY])  
  
 if curY + 1 < len(matrix) and visitMatrix[curX][curY + 1] == 0:  
 if lenMatrix[curX][curY + 1] == 0:  
 lenMatrix[curX][curY + 1] = lenMatrix[curX][curY] + 1  
 elif lenMatrix[curX][curY + 1] > lenMatrix[curX][curY] + 1:  
 lenMatrix[curX][curY + 1] = lenMatrix[curX][curY] + 1  
 visitMatrix[curX][curY + 1] = 1  
 ucsListOfVisited.append([curX, curY + 1])  
  
 if curY - 1 >= 0 and visitMatrix[curX][curY - 1] == 0:  
 if lenMatrix[curX][curY - 1] == 0:  
 lenMatrix[curX][curY - 1] = lenMatrix[curX][curY] + 1  
 elif lenMatrix[curX][curY - 1] > lenMatrix[curX][curY] + 1:  
 lenMatrix[curX][curY - 1] = lenMatrix[curX][curY] + 1  
 visitMatrix[curX][curY - 1] = 1  
 ucsListOfVisited.append([curX, curY - 1])  
  
 if curX - 1 >= 0 and visitMatrix[curX - 1][curY] == 0:  
 if lenMatrix[curX - 1][curY] == 0:  
 lenMatrix[curX - 1][curY] = lenMatrix[curX][curY] + 1  
 elif lenMatrix[curX - 1][curY] > lenMatrix[curX][curY] + 1:  
 lenMatrix[curX - 1][curY] = lenMatrix[curX][curY] + 1  
 visitMatrix[curX - 1][curY] = 1  
 ucsListOfVisited.append([curX - 1, curY])  
 arrBeforePath.append(path[-1])  
 ucsListOfVisited.remove(ucsListOfVisited[0])  
  
  
def findEnemyCoords(matrix):  
 for i in range(len(matrix)):  
 for j in range(len(matrix[i])):  
 if matrix[i][j] == 2:  
 enemyCoords.append([i, j])



1 – Початкове меню гри



2 – момент з гри



3 – фрейм програшу