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** |  | Коробка Ілля |  | ІП-91 |
|  |  | (№ групи, прізвище, ім’я, по батькові ) |  |  |
|  |  |  |  |  |
| **Прийняв** |  | ас. Очеретяний О. К. |  |  |
|  |  | (посада, прізвище, ім’я, по батькові ) |  |  |

Київ 2021

# **Завдання лабораторної роботи**

Темою цієї роботи є створення основи для проекту, над яким подалі буде вестися робота.  
  
**Задача роботи:**Створити базову класичну ретро-гру, на вибір зі списку нижче:

* Pac-Man

Гра має покривати базовий функціонал згідно її геймплею для ОДНОГО гравця.

# **Опис використаних технологій**

Я використовував мову програмування Python, а також бібліотеки turtle freegames random

# **Опис програмного коду**

|  |
| --- |
| from random import choice |
|  | from turtle import \* |
|  |  |
|  | from freegames import floor, vector |
|  |  |
|  | state = {'score': 0} |
|  | path = Turtle(visible=False) |
|  | writer = Turtle(visible=False) |
|  | aim = vector(5, 0) |
|  | pacman = vector(-40, -80) |
|  | ghosts = [ |
|  | [vector(-180, 160), vector(5, 0)], |
|  | [vector(-180, -160), vector(0, 5)], |
|  | [vector(100, 160), vector(0, -5)], |
|  | [vector(100, -160), vector(-5, 0)], |
|  | ] |
|  | # fmt: off |
|  | tiles = [ |
|  | 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, |
|  | 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, |
|  | 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, |
|  | 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, |
|  | 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, |
|  | 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, |
|  | 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, |
|  | 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, |
|  | 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, |
|  | 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, |
|  | 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, |
|  | 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, |
|  | 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, |
|  | 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, |
|  | 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, |
|  | 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, |
|  | 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, |
|  | 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, |
|  | 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, |
|  | 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, |
|  | ] |
|  | # fmt: on |
|  |  |
|  |  |
|  | def square(x, y): |
|  | "Draw square using path at (x, y)." |
|  | path.up() |
|  | path.goto(x, y) |
|  | path.down() |
|  | path.begin\_fill() |
|  |  |
|  | for count in range(4): |
|  | path.forward(20) |
|  | path.left(90) |
|  |  |
|  | path.end\_fill() |
|  |  |
|  |  |
|  | def offset(point): |
|  | "Return offset of point in tiles." |
|  | x = (floor(point.x, 20) + 200) / 20 |
|  | y = (180 - floor(point.y, 20)) / 20 |
|  | index = int(x + y \* 20) |
|  | return index |
|  |  |
|  |  |
|  | def valid(point): |
|  | "Return True if point is valid in tiles." |
|  | index = offset(point) |
|  |  |
|  | if tiles[index] == 0: |
|  | return False |
|  |  |
|  | index = offset(point + 19) |
|  |  |
|  | if tiles[index] == 0: |
|  | return False |
|  |  |
|  | return point.x % 20 == 0 or point.y % 20 == 0 |
|  |  |
|  |  |
|  | def world(): |
|  | "Draw world using path." |
|  | bgcolor('black') |
|  | path.color('blue') |
|  |  |
|  | for index in range(len(tiles)): |
|  | tile = tiles[index] |
|  |  |
|  | if tile > 0: |
|  | x = (index % 20) \* 20 - 200 |
|  | y = 180 - (index // 20) \* 20 |
|  | square(x, y) |
|  |  |
|  | if tile == 1: |
|  | path.up() |
|  | path.goto(x + 10, y + 10) |
|  | path.dot(2, 'white') |
|  |  |
|  |  |
|  | def move(): |
|  | "Move pacman and all ghosts." |
|  | writer.undo() |
|  | writer.write(state['score']) |
|  |  |
|  | clear() |
|  |  |
|  | if valid(pacman + aim): |
|  | pacman.move(aim) |
|  |  |
|  | index = offset(pacman) |
|  |  |
|  | if tiles[index] == 1: |
|  | tiles[index] = 2 |
|  | state['score'] += 1 |
|  | x = (index % 20) \* 20 - 200 |
|  | y = 180 - (index // 20) \* 20 |
|  | square(x, y) |
|  |  |
|  | up() |
|  | goto(pacman.x + 10, pacman.y + 10) |
|  | dot(20, 'yellow') |
|  |  |
|  | for point, course in ghosts: |
|  | if valid(point + course): |
|  | point.move(course) |
|  | else: |
|  | options = [ |
|  | vector(5, 0), |
|  | vector(-5, 0), |
|  | vector(0, 5), |
|  | vector(0, -5), |
|  | ] |
|  | plan = choice(options) |
|  | course.x = plan.x |
|  | course.y = plan.y |
|  |  |
|  | up() |
|  | goto(point.x + 10, point.y + 10) |
|  | dot(20, 'red') |
|  |  |
|  | update() |
|  |  |
|  | for point, course in ghosts: |
|  | if abs(pacman - point) < 20: |
|  | return |
|  |  |
|  | ontimer(move, 100) |
|  |  |
|  |  |
|  | def change(x, y): |
|  | "Change pacman aim if valid." |
|  | if valid(pacman + vector(x, y)): |
|  | aim.x = x |
|  | aim.y = y |
|  |  |
|  |  |
|  | setup(420, 420, 370, 0) |
|  | hideturtle() |
|  | tracer(False) |
|  | writer.goto(160, 160) |
|  | writer.color('white') |
|  | writer.write(state['score']) |
|  | listen() |
|  | onkey(lambda: change(5, 0), 'Right') |
|  | onkey(lambda: change(-5, 0), 'Left') |
|  | onkey(lambda: change(0, 5), 'Up') |
|  | onkey(lambda: change(0, -5), 'Down') |
|  | world() |
|  | move() |
|  | done() |

# **Скріншоти роботи програмного застосунку**

# 

# **Написання логіки тестів**

Були проведені тести із застасунком. Які він пройшов успишно