МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

кафедра програмних засобів

Звіт

з лабораторної роботи № 3

з дисципліни «Мультипарадигмальне програмування»

на тему:

«Застосування принципів мультипарадигмального програмування для розроблення програмного проекту»

Виконав

Ст. гр. КНТ-137 Козлов В.В.

Прийняв

Качан О.І

1. Мета роботи

Навчитися розробляти ігрові додатки на основі використання існуючих рушіїв та приймати рішення з вибору необхідних засобів мультипарадигмального програмування для реалізації їхньої функціональності.

1. Завдання на лабораторну роботу
   1. Варіант 9

Ознайомитись з теоретичними відомостями, необхідними для виконання роботи.

Визначити функціональні вимоги до програмного забезпечення у відповідності до індивідуального завдання, узгодженого з викладачем.

* + 1. Завдання а
  1. Текст завдання

Гірськолижник, яким керує користувач, бере участь у змаганнях зі слалому. Траса позначена жердинами (ворота), у простір між якими має потрапити гірськолижник. За влучне проходження воріт користувач отримує бали. Якщо користувач збиває або торкається жердин, то бали відраховуються. Гірськолижник може збільшувати швидкість або зменшувати. У випадку руху з великою швидкістю на повороті він може вилетіти з траси. Результат визначає час проходження траси та кількість балів. Якщо кількість балів нижче деякого рівня,то час не зараховується.

* 1. Лістинг програми

# Program for laboratory work 3, task a  
  
import pygame  
import random  
from os import path  
from datetime import datetime  
  
img\_dir = path.join(path.dirname(\_\_file\_\_), 'img')  
  
WIDTH = 720  
HEIGHT = 480  
FPS = 30  
MAX\_DISTANCE = 60000  
  
pygame.init()  
win = pygame.display.set\_mode((WIDTH, HEIGHT))  
pygame.display.set\_caption("SkierGame")  
pygame.display.set\_icon(pygame.image.load(path.join(img\_dir, 'SkierGame.png')))  
clock = pygame.time.Clock()  
  
text\_font = pygame.font.SysFont("Comic Sans MS", 30)  
score\_font = pygame.font.SysFont("Times New Roman", 15)  
  
tree\_img = [pygame.image.load(path.join(img\_dir, "tree\_01.png")), pygame.image.load(path.join(img\_dir, "tree\_02.png"))]  
gate\_img = pygame.image.load(path.join(img\_dir, "gate.png"))  
  
  
class Player():  
 def \_\_init\_\_(self):  
 skier\_stay\_img\_name = ["skier\_stay\_01.png", "skier\_stay\_02.png", "skier\_stay\_03.png",  
 "skier\_stay\_04.png", "skier\_stay\_05.png", "skier\_stay\_06.png"]  
 self.skier\_stay\_img = []  
 for img in skier\_stay\_img\_name:  
 self.skier\_stay\_img.append(pygame.image.load(path.join(img\_dir, img)))  
  
 skier\_down\_img\_name = ["skier\_down\_01.png", "skier\_down\_02.png", "skier\_down\_03.png",  
 "skier\_down\_04.png", "skier\_down\_05.png", "skier\_down\_06.png",  
 "skier\_down\_07.png", "skier\_down\_08.png", "skier\_down\_09.png",  
 "skier\_down\_10.png"]  
 self.skier\_down\_img = []  
 for img in skier\_down\_img\_name:  
 self.skier\_down\_img.append(pygame.image.load(path.join(img\_dir, img)))  
  
 skier\_left\_img\_name = ["skier\_left\_01.png", "skier\_left\_02.png", "skier\_left\_03.png",  
 "skier\_left\_04.png", "skier\_left\_05.png", "skier\_left\_06.png"]  
 self.skier\_left\_img = []  
 for img in skier\_left\_img\_name:  
 self.skier\_left\_img.append(pygame.image.load(path.join(img\_dir, img)))  
  
 skier\_right\_img\_name = ["skier\_right\_01.png", "skier\_right\_02.png", "skier\_right\_03.png",  
 "skier\_right\_04.png", "skier\_right\_05.png", "skier\_right\_06.png"]  
 self.skier\_right\_img = []  
 for img in skier\_right\_img\_name:  
 self.skier\_right\_img.append(pygame.image.load(path.join(img\_dir, img)))  
  
 self.skier\_death\_imgs = [pygame.image.load(path.join(img\_dir, "skier\_death\_01.png")),  
 pygame.image.load(path.join(img\_dir, "skier\_death\_02.png"))]  
 self.skier\_death\_img = self.skier\_death\_imgs[random.randint(0, 1)]  
  
 self.skier\_win\_img = pygame.image.load(path.join(img\_dir, "skier\_win.png"))  
  
 self.x = (WIDTH - 32) // 2  
 self.y = HEIGHT // 4  
 self.speed = 0  
 self.speed\_mul = 0  
 self.corner\_speed = 0  
 self.corner\_speed\_mul = 0  
 self.anim\_count = 0  
 self.left = False  
 self.right = False  
 self.hitbox = pygame.Rect(self.x + 8, self.y + 4, 15, 23)  
 self.distance = 0  
 self.score = 0  
 self.stay = False  
 self.win = False  
 self.death = False  
 self.ride\_time = 0  
  
 def get\_score(self):  
 return self.score  
  
 def get\_ride\_time(self):  
 return self.ride\_time  
  
 def play\_intro(self):  
 self.stay = True  
  
 def play\_death(self):  
 self.death = True  
 self.speed = 0  
 self.ride\_time = datetime.now() - self.ride\_time  
  
 def is\_dead(self):  
 return self.death  
  
 def play\_win(self):  
 self.win = True  
 self.speed = 0  
 self.distance = MAX\_DISTANCE  
 self.ride\_time = datetime.now() - self.ride\_time  
  
 def is\_win(self):  
 return self.win  
  
 def start\_ride(self):  
 self.stay = False  
 self.anim\_count = 0  
 self.speed = 5  
 self.ride\_time = datetime.now()  
  
 def turn\_left(self):  
 if not self.left:  
 self.left = True  
 self.right = False  
 self.anim\_count = 0  
 if self.corner\_speed >= 0:  
 self.corner\_speed\_mul = 0  
 else:  
 self.corner\_speed\_mul = (30 \* -self.corner\_speed - 3) // 19  
  
 if self.corner\_speed > 0:  
 self.corner\_speed -= 2  
 else:  
 self.corner\_speed = -(3 + (self.corner\_speed\_mul//3) + (self.corner\_speed\_mul//5) + (self.corner\_speed\_mul//10))  
 self.corner\_speed\_mul += 1  
 self.speed\_mul = max(0, self.speed\_mul - (self.speed\_mul // 30 + self.speed\_mul // 75))  
  
 def turn\_right(self):  
 if not self.right:  
 self.right = True  
 self.left = False  
 self.anim\_count = 0  
 if self.corner\_speed <= 0:  
 self.corner\_speed\_mul = 0  
 else:  
 self.corner\_speed\_mul = (30 \* self.corner\_speed - 3) // 19  
  
 if self.corner\_speed < 0:  
 self.corner\_speed += 2  
 else:  
 self.corner\_speed = (3 + (self.corner\_speed\_mul//3) + (self.corner\_speed\_mul//5) + (self.corner\_speed\_mul//10))  
 self.corner\_speed\_mul += 1  
 self.speed\_mul = max(0, self.speed\_mul - (1 + self.speed\_mul // 30 + self.speed\_mul // 75))  
  
 def go\_down(self):  
 if self.right or self.left:  
 self.right = False  
 self.left = False  
 self.anim\_count = 0  
  
 if self.corner\_speed < 0:  
 self.corner\_speed += 1  
 elif self.corner\_speed > 0:  
 self.corner\_speed -= 1  
  
 self.speed\_mul = min(200, self.speed\_mul + 1)  
  
 def get\_pos\_x(self):  
 return self.x  
  
 def get\_player\_speed(self):  
 return self.speed  
  
 def get\_player\_distance(self):  
 return self.distance  
  
 def update(self):  
 if self.x + self.corner\_speed > WIDTH - 32:  
 self.x = WIDTH - 32  
 self.corner\_speed = 0  
 elif self.x + self.corner\_speed < 0:  
 self.x = 0  
 self.corner\_speed = 0  
 else:  
 self.x += self.corner\_speed  
 self.hitbox.x = self.x + 8  
 self.speed = 5 + self.speed\_mul // 20  
 self.distance += self.speed  
  
 def update\_score(self, points):  
 self.score += points  
  
 def draw(self):  
 if self.stay:  
 win.blit(self.skier\_stay\_img[self.anim\_count // 5], (self.x, self.y))  
 elif self.death:  
 win.blit(self.skier\_death\_img, (self.x, self.y))  
 elif self.win:  
 win.blit(self.skier\_win\_img, (self.x, self.y))  
 elif self.left:  
 win.blit(self.skier\_left\_img[self.anim\_count // 5], (self.x, self.y))  
 elif self.right:  
 win.blit(self.skier\_right\_img[self.anim\_count // 5], (self.x, self.y))  
 else:  
 win.blit(self.skier\_down\_img[self.anim\_count // 3], (self.x, self.y))  
  
 self.anim\_count += 1  
 if self.anim\_count >= FPS:  
 self.anim\_count = 0  
 # pygame.draw.rect(win, (255, 0, 0), self.hitbox, 1)  
  
  
class Tree():  
 def \_\_init\_\_(self):  
 self.x = random.randint(32, WIDTH - 64)  
 self.y = HEIGHT + 32  
 self.image = tree\_img[random.randint(0, 1)]  
 self.hitbox = pygame.Rect(self.x + 10, self.y + 17, 13, 15)  
 self.hit = False  
  
 def update(self, player\_speed):  
 self.y -= player\_speed  
 self.hitbox.y = self.y + 17  
  
 def draw(self):  
 win.blit(self.image, (self.x, self.y))  
 # pygame.draw.rect(win, (255, 0, 0), self.hitbox, 1)  
  
  
class Gate():  
 def \_\_init\_\_(self, player\_distance):  
 self.width = max(300 - player\_distance // 250, 100)  
 self.x = random.randint(32, WIDTH - 32 - 2 \* 22 - self.width)  
 self.y = HEIGHT + 32  
 self.hitbox = pygame.Rect(self.x + 10, self.y + 16, 12 + self.width + 12, 16)  
 self.hitbox\_left\_gate = pygame.Rect(self.x + 10, self.y + 16, 12, 16)  
 self.hitbox\_right\_gate = pygame.Rect(self.x + 22 + self.width, self.y + 16, 12, 16)  
 self.check = False  
  
 def is\_check(self):  
 return self.check  
  
 def check\_gate(self):  
 self.check = True  
  
 def draw(self):  
 win.blit(gate\_img, (self.x, self.y))  
 pygame.draw.line(win, (0, 0, 255), (self.x + 22, self.y + 31), (self.x + 22 + self.width - 1, self.y + 31), 1)  
 win.blit(gate\_img, (self.x + 12 + self.width, self.y))  
 # pygame.draw.rect(win, (255, 0, 0), self.hitbox, 1)  
  
 def update(self, player\_speed):  
 self.y -= player\_speed  
 self.hitbox.y = self.y + 16  
 self.hitbox\_left\_gate.y = self.y + 16  
 self.hitbox\_right\_gate.y = self.y + 16  
  
  
class Lift(pygame.sprite.Sprite):  
 def \_\_init\_\_(self, picture):  
 self.lift\_images = [pygame.image.load(path.join(path.join(img\_dir, "lift\_02.png"))),  
 pygame.image.load(path.join(path.join(img\_dir, "lift\_01.png")))]  
 self.image = self.lift\_images[picture]  
 self.rect = self.image.get\_rect()  
 if picture == 0:  
 self.speed = -3  
 self.rect.x = 200 + 32  
 self.rect.y = HEIGHT + 1500  
 else:  
 self.speed = 3  
 self.rect.x = 200  
 self.rect.y = -500  
 super().\_\_init\_\_()  
  
 def update(self, \*args):  
 if self.speed < 0:  
 self.rect.y += (-args[0] + self.speed)  
 if self.rect.y <= -500:  
 self.rect.y = HEIGHT + 1500  
 else:  
 if args[0] == 0:  
 self.rect.y += self.speed  
 if self.rect.y >= HEIGHT + 1500:  
 self.rect.y = -500  
 else:  
 self.rect.y += (-args[0] + self.speed)  
 if self.rect.y <= -500:  
 self.rect.y = HEIGHT + 1500  
  
  
def play\_intro():  
 intro = True  
 player.play\_intro()  
 while intro:  
 clock.tick(FPS)  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 pygame.quit()  
 exit()  
  
 keys = pygame.key.get\_pressed()  
 if keys[pygame.K\_SPACE]:  
 player.start\_ride()  
 break  
  
 win.fill((255, 255, 255))  
 pygame.draw.line(win, (0, 0, 255), (32, 0), (32, HEIGHT), 2)  
 pygame.draw.line(win, (0, 0, 255), (WIDTH - 32, 0), (WIDTH - 32, HEIGHT), 2)  
 all\_sprites.update(player.get\_player\_speed())  
 all\_sprites.draw(win)  
 text = text\_font.render("Press space to start!", True, (0, 0, 255))  
 win.blit(text, (WIDTH // 2 - text.get\_rect().width // 2, HEIGHT // 2 - 15))  
 player.draw()  
 pygame.display.update()  
  
  
def draw\_score():  
 pygame.draw.rect(win, (255, 255, 255), (WIDTH - 100, 0, 100, 90))  
 pygame.draw.rect(win, (0, 0, 0), (WIDTH - 100, 0, 100, 90), 1)  
 text = score\_font.render("Score: {}".format(player.get\_score()), True, (0, 0, 0))  
 win.blit(text, (WIDTH - 100 + 5, 5))  
 if player.is\_dead() or player.is\_win():  
 time = datetime(2019, 5, 25, 00, 00, 00) + player.get\_ride\_time()  
 text = score\_font.render("Time: {}".format((time.strftime("%M:%S.%f"))[0:9]), True, (0, 0, 0))  
 else:  
 time = datetime(2019, 5, 25, 00, 00, 00) + (datetime.now() - player.get\_ride\_time())  
 text = score\_font.render("Time: {}".format((time.strftime("%M:%S.%f"))[0:9]), True, (0, 0, 0))  
 win.blit(text, (WIDTH - 100 + 5, 25))  
 text = score\_font.render("Finish: {}m".format((MAX\_DISTANCE - player.get\_player\_distance())//30), True, (00, 0, 0))  
 win.blit(text, (WIDTH - 100 + 5, 45))  
 text = score\_font.render("Speed: {}m/s".format(player.get\_player\_speed()), True, (0, 0, 0))  
 win.blit(text, (WIDTH - 100 + 5, 65))  
  
  
def play\_game\_over():  
 game\_over = True  
 while game\_over:  
 clock.tick(FPS)  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 pygame.quit()  
 exit()  
  
 keys = pygame.key.get\_pressed()  
 if keys[pygame.K\_RETURN] or keys[pygame.K\_KP\_ENTER]:  
 break  
  
 win.fill((255, 255, 255))  
 pygame.draw.line(win, (0, 0, 255), (32, 0), (32, HEIGHT), 2)  
 pygame.draw.line(win, (0, 0, 255), (WIDTH - 32, 0), (WIDTH - 32, HEIGHT), 2)  
 for gate in gates\_list:  
 gate.draw()  
 for tree in tree\_list:  
 tree.draw()  
 all\_sprites.draw(win)  
  
 pygame.draw.rect(win, (255, 255, 255), (160, 170, 400, 140))  
 pygame.draw.rect(win, (0, 0, 255), (160, 170, 400, 140), 2)  
 if player.is\_win():  
 text = text\_font.render("Your score: {}".format(player.get\_score()), True, (0, 0, 255))  
 win.blit(text, (WIDTH // 2 - text.get\_rect().width // 2, HEIGHT // 2 - 3 \* text.get\_rect().height // 2))  
 time = datetime(2019, 5, 25, 00, 00, 00) + player.get\_ride\_time()  
 text = text\_font.render("Your time: {}".format((time.strftime("%M:%S.%f"))[0:9]), True, (0, 0, 255))  
 win.blit(text, (WIDTH // 2 - text.get\_rect().width // 2, HEIGHT // 2 - text.get\_rect().height // 2))  
 text = text\_font.render("Press enter to continue!", True, (0, 0, 255))  
 win.blit(text, (WIDTH // 2 - text.get\_rect().width // 2, HEIGHT // 2 + text.get\_rect().height // 2))  
 elif player.is\_dead():  
 text = text\_font.render("Oops...", True, (0, 0, 255))  
 win.blit(text, (WIDTH // 2 - text.get\_rect().width // 2, HEIGHT // 2 - text.get\_rect().height))  
 text = text\_font.render("Press enter to continue!", True, (0, 0, 255))  
 win.blit(text, (WIDTH // 2 - text.get\_rect().width // 2, HEIGHT // 2))  
 player.draw()  
 draw\_score()  
 pygame.display.update()  
  
  
play = True  
while play:  
 player = Player()  
 tree\_list = list()  
 distance\_to\_generate\_tree = random.randint(5, 30)  
 gates\_list = list()  
 distance\_to\_generate\_gate = 500  
 all\_sprites = pygame.sprite.Group()  
 all\_sprites.add(Lift(0))  
 all\_sprites.add(Lift(1))  
  
 play\_intro()  
 run = True  
 while run:  
 clock.tick(FPS)  
  
 if distance\_to\_generate\_tree <= 0:  
 tree\_list.append(Tree())  
 distance\_to\_generate\_tree = random.randint(25, 125)  
 else:  
 distance\_to\_generate\_tree -= player.get\_player\_speed()  
  
 if distance\_to\_generate\_gate <= 0:  
 gates\_list.append(Gate(player.get\_player\_distance()))  
 distance\_to\_generate\_gate = 500  
 else:  
 distance\_to\_generate\_gate -= player.get\_player\_speed()  
  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 pygame.quit()  
 exit()  
  
 keys = pygame.key.get\_pressed()  
 if keys[pygame.K\_LEFT]:  
 player.turn\_left()  
 elif keys[pygame.K\_RIGHT]:  
 player.turn\_right()  
 else:  
 player.go\_down()  
  
 win.fill((255, 255, 255))  
 pygame.draw.line(win, (0, 0, 255), (32, 0), (32, HEIGHT), 2)  
 pygame.draw.line(win, (0, 0, 255), (WIDTH - 32, 0), (WIDTH - 32, HEIGHT), 2)  
  
 for gate in gates\_list:  
 gate.update(player.get\_player\_speed())  
 if gate.y <= -64:  
 gates\_list.pop(gates\_list.index(gate))  
 else:  
 if gate.y + 32 < (HEIGHT // 4) and not gate.is\_check():  
 gate.check\_gate()  
 player.update\_score(-15)  
 else:  
 if not gate.is\_check() and gate.hitbox.colliderect(player.hitbox):  
 gate.check\_gate()  
 if gate.hitbox\_left\_gate.colliderect(player.hitbox) or gate.hitbox\_right\_gate.colliderect(player.hitbox):  
 player.update\_score(5)  
 else:  
 player.update\_score(15)  
 gate.draw()  
  
 player.update()  
 if player.get\_player\_distance() >= MAX\_DISTANCE:  
 player.play\_win()  
 run = False  
 player.draw()  
  
 for tree in tree\_list:  
 tree.update(player.get\_player\_speed())  
 if tree.y <= -64:  
 tree\_list.pop(tree\_list.index(tree))  
 else:  
 if tree.hitbox.colliderect(player.hitbox):  
 player.play\_death()  
 run = False  
 tree.draw()  
  
 if player.get\_pos\_x() <= 10 or player.get\_pos\_x() >= WIDTH - 32 - 10:  
 player.play\_death()  
 run = False  
  
 all\_sprites.update(player.get\_player\_speed())  
 all\_sprites.draw(win)  
 draw\_score()  
 pygame.display.update()  
  
 play\_game\_over()  
  
pygame.quit()

* 1. Скріншот роботи програми



Рисунок 3.1 – Скріншот роботи програми

Висновки

Протягом виконання цієї лабораторної роботи ознайомився з основними теоретичними відомостями щодо ігрового рушія Pygame.

Навчився розробляти програми мовою програмування Python на основі використання ігрового рушія.