Міністерство освіти і науки України

Національний університет «Запорізька Політехніка»

Кафедра програмних засобів

ЗВІТ

з лабораторної роботи №3

з дисципліни «Системний аналіз» на тему:

«Системний аналіз під час повторного використання коду»

Виконав

Студент групи КНТ-122 О. А. Онищенко

Прийняли

Викладач Л. Ю. Дейнега

2024

СИСТЕМНИЙ АНАЛІЗ ПІД ЧАС ПОВТОРНОГО ВИКОРИСТАННЯ КОДУ

Мета роботи

Навчитися застосовувати принципи системного аналізу під час повторного використання коду, навчитися розробляти ігрові застосунки на основі використання існуючих рушіїв з врахуванням результатів системного аналізу проблеми.

Завдання

- На стенді стоїть деяка кількість об’єктів. Деякі з них рухаються. Користувач кидає в них набір предметів. Кожен з них має свої фізичні характеристики (прискорення, точність потрапляння). Кожен предмет можна кинути тільки один раз. Кожен об’єкт на стенді у разі потрапляння в нього приносить певну кількість очок. Падаючи, предмети можуть збивати інші предмети, приносячи додаткову кількість очок. Необхідно набрати максимальну кількість очок. Результати користувача зберігаються.

Код програми

```
import pygame
from pygame.locals import *

import json
import math
import random
from os import path
from datetime import datetime

from rich.console import Console
from rich.traceback import install
```

```
install()
console = Console()


class Stand:
    def __init__(self, x, y, moving):
        self.x = x
        self.y = y
        self.moving = moving
        self.points = random.randint(1, 10)
        self.lastMoved = pygame.time.get_ticks()
        self._radius = random.randint(10, 20)
        self._color = generateColor()

    def draw(self, screen):
        pygame.draw.circle(screen, self._color, (self.x, self.y),
self._radius)

    def move(self):
        coordinates = generateCoordinates()
        self.x = coordinates["x"]
        self.y = coordinates["y"]
        self.lastMoved = pygame.time.get_ticks()

    def checkCollateral(self, target):
        dx = self.x - target.x
        dy = self.y - target.y
        distance = (dx**2 + dy**2) ** 0.5
        return distance <= self._radius + target._radius


class Thrown:
    def __init__(self, x, y, speed):
        self.x = x
        self.y = y
        self.speed = speed
        self.direction = None
        self.accuracy = random.randint(7, 11)
        self._radius = 7

    def draw(self, screen):
        pygame.draw.circle(screen, "darkgoldenrod1", (self.x, self.y),
self._radius)

    def throw(self, target):
        if not self.direction:
            dx, dy = target[0] - self.x, target[1] - self.y
            distance = (dx**2 + dy**2) ** 0.5
```

```
            self.direction = (dx / distance, dy / distance)
            distortion = (11 - self.accuracy) / 10
            randomAngle = (random.random() * 2 - 1) * distortion
            self.direction = (
                self.direction[0] * math.cos(randomAngle)
                - self.direction[1] * math.sin(randomAngle),
                self.direction[0] * math.sin(randomAngle)
                + self.direction[1] * math.cos(randomAngle),
            )

    def update(self):
        if self.direction:
            self.x += self.direction[0] * self.speed
            self.y += self.direction[1] * self.speed


def generateColor() -> str:
    colorsPool = [
        "yellow",
        "wheat1",
        "violetred1",
        "violet",
        "turquoise1",
        "tomato",
        "thistle1",
        "steelblue1",
        "springgreen",
        "rosybrown1",
        "seagreen1",
        "powderblue",
        "paleturquoise1",
        "palegreen1",
        "orchid1",
        "olivedrab1",
        "moccasin",
    ]
    return random.choice(colorsPool)


def generateCoordinates(
    screenWidth=800, screenHeight=400, offset=100, maximumRadius=20
):
    return {
        "x": random.randint(offset, screenWidth - maximumRadius),
        "y": random.randint(offset, screenHeight - maximumRadius),
    }


def generateObjects(
```

```
    standNumber=random.randint(1, 10), thrownNumber=random.randint(1, 5)
):
    stand = [
        Stand(
            x=generateCoordinates()["x"],
            y=generateCoordinates()["y"],
            moving=random.randint(0, 1),
        )
        for i in range(standNumber)
    ]
    thrown = [
        Thrown(x=10, y=10, speed=random.randint(1, 10)) for i in
range(thrownNumber)
    ]
    return {"stand": stand, "thrown": thrown}


def checkCollision(ball, stand):
    dx = ball.x - stand.x
    dy = ball.y - stand.y
    distance = (dx**2 + dy**2) ** 0.5
    return distance <= ball._radius + stand._radius


def startGame():
    objects = generateObjects(
        standNumber=random.randint(1, 10), thrownNumber=random.randint(1,
5)
    )
    standObjects, throwableObjects = objects["stand"], objects["thrown"]
    thrown = []
    score = 0
    return standObjects, throwableObjects, thrown, score


def readExistingLogs(logFile):
    try:
        with open(logFile, "r", encoding="utf-8") as f:
            return json.load(f)
    except:
        return []


def formLogEntry(standObjects, throwableObjects, thrown, score):
    return {
        "stand": len(standObjects),
        "thrown": len(throwableObjects),
        "thrown": len(thrown),
        "score": score,
```

```
        "time": datetime.now().strftime("%d-%m-%Y %H:%M:%S"),
    }


def writeLogs(logFile, logs):
    with open(logFile, "w", encoding="utf-8") as f:
        json.dump(logs, f, ensure_ascii=False, indent=2)


pygame.init()
screen = pygame.display.set_mode((800, 400))
pygame.display.set_caption("Thrower Game")
clock = pygame.time.Clock()

STAND_MOVING_INTERVAL = 3000

standObjects, throwableObjects, thrown, score = startGame()

currentDir = path.dirname(path.abspath(__file__))
logFile = path.join(currentDir, "..", "data", "log.json")
logs = readExistingLogs(logFile)

while True:
    for event in pygame.event.get():
        ESCAPE_KEY_PRESSED = event.type == KEYDOWN and event.key ==
K_ESCAPE
        if event.type == QUIT or ESCAPE_KEY_PRESSED:
            writeLogs(logFile, logs)
            exit()
        elif event.type == MOUSEBUTTONDOWN and event.button == 1:
            if throwableObjects:
                targetBall = throwableObjects.pop()
                targetBall.throw(pygame.mouse.get_pos())
                thrown.append(targetBall)
        ENTER_OR_SPACE = event.type == KEYDOWN and (
            event.key == K_RETURN or event.key == K_SPACE
        )
        if ENTER_OR_SPACE:
            logEntry = formLogEntry(standObjects, throwableObjects,
thrown, score)
            logs.append(logEntry)
            standObjects, throwableObjects, thrown, score = startGame()
    screen.fill("white")

    for stand in standObjects:
        if stand.moving:
            if pygame.time.get_ticks() - stand.lastMoved >
STAND_MOVING_INTERVAL:
                stand.move()
```

```
            stand.draw(screen)
        for ball in throwableObjects:
            ball.draw(screen)
        for ball in thrown:
            ball.update()
            ball.draw(screen)
            for stand in standObjects:
                if checkCollision(ball, stand):
                    score += stand.points
                    thrown.remove(ball)
                    collaterals = [s for s in standObjects if
stand.checkCollateral(s)]
                    for collateral in collaterals:
                        score += collateral.points // 2
                        standObjects.remove(collateral)
                    break
        scoreText = pygame.font.SysFont("Verdana", 18).render(
            f"Score: {score} | Throwables left: {len(throwableObjects)}",
True, "black"
        )
        screen.blit(scoreText, (525, 10))
        pygame.display.update()
        clock.tick(60)
```

Результати виконання

Після виконання коду отримуємо наступні результати:

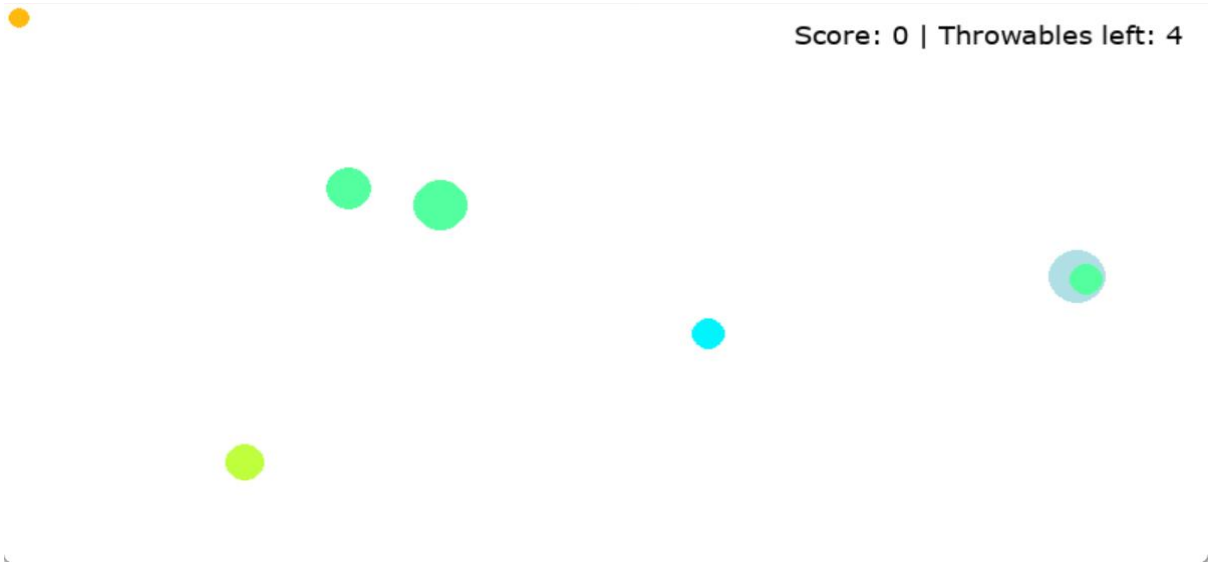

Рисунок 1.1 – Результати виконання програми

Висновки

Таким чином, ми навчилися застосовувати принципи системного аналізу під час повторного використання коду, а також навчилися розробляти ігрові застосунки на основі використання існуючих рушіїв з врахуванням результатів системного аналізу проблеми.

Контрольні питання

Що таке ігровий рушій?

Ігровий рушій - це програмний фреймворк, призначений насамперед для розробки відеоігор. Він включає відповідні бібліотеки та допоміжні програми, такі як редактор рівнів. Він впорядковує, прискорює і спрощує процес розробки гри, керуючи технічними аспектами, такими як графіка, фізика, звук і введення даних користувачем.

Для чого використовуються ігрові рушії?

Ігрові рушії використовуються для створення ігор для відеоігрових консолей та інших типів комп'ютерів. Вони забезпечують платформу для запуску гри, завантажують світ, поміщають у нього гравця та керують його перебуванням. Наявність рушія, який вже має способи обробки таких речей, як рендеринг, фізика, освітлення дозволяє швидко створювати моделі персонажів і змушувати їх поводитися певним чином.

Які частини комп’ютерної гри зазвичай дозволяє контролювати ігровий рушій?

Ігровий рушій зазвичай керує кількома компонентами комп'ютерної гри. До них відносяться рушій рендерингу 2D або 3D графіки, фізичний рушій або рушій виявлення зіткнень (і реакція на зіткнення), звук, сценарії, анімація, мережа, потокова передача даних, управління пам'яттю, багатопотоковість, підтримка локалізації, графіка сцени і відеопідтримка для кінематографа.