НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

КУРСОВА РОБОТА

з дисципліни «Інженерія програмного забезпечення»
на тему: «Розробка гри “Скок” на мові Python з використанням бібліотеки для створення ігор PyGame»

Студента 2 курсу групи ІО-13
напряму підготовки: 123 - Комп’ютерна інженерія
номер залікової : 1330
Шудрика Андрія Олександровича

Керівник доцент, к.т.н., с.н.с.
Антонюк Андрій Іванович

Національна оцінка _____
Кількість балів _____

Члени комісії: _____ _____
(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

_____ _____
(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

_____ _____
(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

Київ - 2023 рік

Зміст

Вступ

Курсова робота на тему " Розробка гри “Скок” на мові Python з використанням бібліотеки для створення ігор PyGame» присвячена розробленню повноцінної гри на мові Python з використанням бібліотеки PyGame. У роботі розглядаються різні аспекти розробки програмного додатку, включаючи проєктування графічного інтерфейсу користувача, реалізацію логіки гри, використання звукових ефектів та графіки.
В роботі розглядаються різні методи та технології програмування, такі як мова програмування Python, бібліотеки для розробки графічного інтерфейсу, такі як PyGame, а також бібліотеки для реалізації графіки та звуку.
Крім того, у роботі проводиться детальний аналіз ефективності розробленої програми та описуються можливості для подальшого розвитку та вдосконалення додатку.
Отже, курсова робота є детальним описом розробки програмного додатку з графічним інтерфейсом користувача для гри "Скок". Вона може бути корисною для студентів та програмістів, які цікавляться розробкою ігрових додатків та графічного програмування.

1. Обґрунтування вибору теми

Тема курсової роботи "Розробка гри на мові Python з використанням бібліотеки Pygame" є обґрунтованою з декількох причин:

Актуальність: Галузь ігрової розробки постійно розвивається, і стає все більш популярною в наш час. Python є однією з найбільш популярних мов програмування в галузі ігрової розробки.

Доступність: Python є легко вивчуваною мовою програмування, що дозволяє початківцям ефективно вивчати та використовувати її для розробки гри.

Популярність Pygame: Pygame є однією з найпопулярніших бібліотек для розробки ігор на Python, що дозволяє створити гру з графічним інтерфейсом користувача та взаємодією з клавіатурою та мишею.

Практичність: Розробка гри на Python з використанням Pygame дозволяє здобути практичні навички в програмуванні, графічному дизайні та ігровій

розробці, що може бути корисним в подальшій професійній діяльності.

Таким чином, тема курсової роботи "Розробка гри на мові Python з використанням бібліотеки Pygame" є обґрунтованою та актуальною, та дозволить розширити свої знання в галузі програмування та ігрової розробки.

2. Мета та завдання роботи

Мета розробки гри "Скок" на мові Python з використанням бібліотеки Pygame полягає в наступному:

Розробка гри на Python з використанням Pygame дає можливість студенту продемонструвати свої навички програмування, зокрема у створенні ігор та графічних застосунків; оволодіти новими технологіями в галузі ігрової розробки, такими як робота з графікою, звуком та взаємодією з користувачем.

Завдання курсової роботи – залучити охочих до розробки гри на Python задля розширення кругозору у галузі ігрової розробки та ознайомити з основними поняттями та принципами розробки ігор.

Зокрема, метою розробки гри "Скок" може бути задоволення від створення власної гри та підвищення мотивації для подальшого вивчення програмування та ігрової розробки.

3. Огляд технологій

Python є широко поширеною високорівневою мовою програмування, яка застосовується для написання застосунків, веб-розробки та складних обчислень. Підтримує модульну структуру, має велику бібліотеку даних. Це готові функції, які можна налаштувати під конкретні потреби клієнта. Наприклад, шаблонні рішення у вигляді форм, видів сторінок та типових блоків. Але при цьому кожен із них має безліч індивідуальних налаштувань, що дозволяє зробити продукт унікальним.. Мова досить потужна, самостійна та підходить для розробки застосунків та програм різної складності.

Бібліотека Pygame є потужним інструментом для розробки ігор та інших графічних застосунків на мові програмування Python. Ось деякі з переваг, які надає ця бібліотека:

Простота використання: Pygame пропонує простий та інтуїтивний інтерфейс програмування, що дозволяє швидко розробляти графічні застосунки.

Крос-платформеність: Pygame підтримує багато операційних систем, включаючи Windows, MacOS та Linux.

Гнучкість: бібліотека Pygame дозволяє розробникам створювати різні типи ігор, від простих 2D-ігор до складних 3D-ігор.

Наявність різноманітних інструментів та функцій: Pygame має вбудовані функції для роботи з графікою, звуком, мережевими операціями та іншими функціями.

Відкритий код: Pygame є вільним та відкритим програмним забезпеченням з ліцензією LGPL, що дозволяє розробникам використовувати та модифікувати бібліотеку за своїми потребами.

4. Аналіз гри "Скок"

Гра “Скок” різновидом аркадних ігор. Гравець керує платформою, яка може рухатися вліво та вправо, та має за мету знищити всі блоки, що знаходяться на екрані, відбиваючи м'ячик між платформою та блоками. Гра має досить простий геймплей, але водночас дуже захоплива та динамічна.

Різноманітність рівнів. Гра має багато різних рівнів складності, що дозволяє гравцям покращувати свої навички та постійно вдосконалюватися.

Ігрова механіка. Використання платформи для відбивання м'ячика дозволяє гравцям взаємодіяти з грою та контролювати рух м'ячика. Це додає грі динамічності та захоплюючості.

Візуальний дизайн. Гра має яскравий та кольоровий дизайн, що дозволяє гравцям відчувати себе в ігровому світі. Водночас, дизайн є дуже простим та не заважає гравцям концентруватися на грі.

5. Проектування програмного додатку:

5.1. Аналіз геймплею та механік гри

Гра має наступні механіки та елементи геймплею:

Платформа: гравець керує платформою знизу екрану, яка використовується для відбивання м'яча. Гравець може рухати платформу вправо або вліво, щоб забезпечити відбиття м'яча в потрібному напрямку.

М'яч: м'яч відскакує від платформи та стін на екрані, доки не зіткнеться блоком або не впаде вниз екрану. Гравець може відбивати м'яч, керуючи платформою.

Блоки: блоки розташовані у верхній частині екрану та мають різні кольори та ступені міцності. Гравець повинен розбивати блоки, відбиваючи м'яч в їхньому напрямку. Кожен блок має певну кількість "життів", тобто може бути розбитий лише після декількох ударів.

Бонуси: іноді з розбитих блоків випадають бонуси, які можуть збільшити платформу, прискорити м'яч або надати інші переваги гравцю.

Життя: гравець має обмежену кількість життів, що позначається у верхній частині екрану. Якщо м'яч впаде вниз екрану, гравець втрачає життя. Гра закінчується, коли гравець витрачає всі свої життя.

Арканоїд має дуже простий та зрозумілий геймплей, що робить його доступним для гравців будь-якого рівня досвіду.

Додавання звукових ефектів
Для забезпечення гри звуковими ефектами, я завантажив та додав у папку різні звуки, а потім за допомогою існуючої бібліотеки mixer адаптував їх у гру.

5.2. Use Case гри Арканоїд:

Передумови:

Гравець запустив гру "Арканоїд".
Основні кроки:

Гравець починає гру, натискаючи кнопку "Старт".
Гравець використовує ракетку для відбивання м'яча та знищення блоків.
Гравець може використовувати монетки, які випадають зі знищених блоків, щоб отримати більшу кількість балів.
Гра закінчується, коли гравець пропускає м'яч і він падає вниз за межі ракетки декілька разів.

Альтернативні варіанти:

Гравець може паузити гру, натиснувши кнопку "Пауза".
Гравець може вийти з гри, натиснувши кнопку "Вихід".

Післяумови:

Гравець має можливість почати нову гру або вийти з гри.

6. Розробка программного забезпечення:

6.1 Створення прецедентів для гри «Скок»:

Запуск гри:

1. Користувач запускає гру Арканоїд.

Система завантажує необхідні ресурси і показує початковий екран гри.

2. Переміщення ракетки:

Користувач використовує клавіші вліво та вправо для переміщення ракетки горизонтально.
Система змінює позицію ракетки залежно від введення користувача.

3. Пуск кульки:

Система розпочинає рух кульки у визначеному напрямку після запуску гри.

4. Взаємодія кульки з цеглами:

Кулька зіткнулась з цеглою.
Система розраховує новий напрямок руху кульки після зіткнення з цеглою.
Система змінює спрайт цілої цеглини на зруйнованую після першого зіткнення.
Система видаляє знищену цеглу з екрану після другого зіткнення.

5. Зіткнення кульки з ракеткою:

Кулька зіткнулась з ракеткою.
Система розраховує новий напрямок руху кульки.

6. Зіткнення кульки з нижнім краєм екрану:

Кулька зіткнулась з нижнім краєм екрану, не зіткнувшись з ракеткою.
Система зменшує кількість доступних життів гравця.
Якщо доступні життя закінчилися, гра закінчується.

7. Завершення рівня:

Всі цегли на рівні були знищені кулькою.
Система переходить до наступного рівня, відновлює цегли та розміщує нову кульку.

8. Завершення гри:

Всі доступні рівні були пройдені або гравець втратив всі життя.
Система показує підсумкові дані, такі як рахунок гравця та досягнення.
Користувач може почати нову гру або вийти з гри.

6.2.Ескіз графічного дизайну (рис. 6.2.1, рис. 6.2.2, 6.2.3, 6.2.4):

Рис. 6.2.1

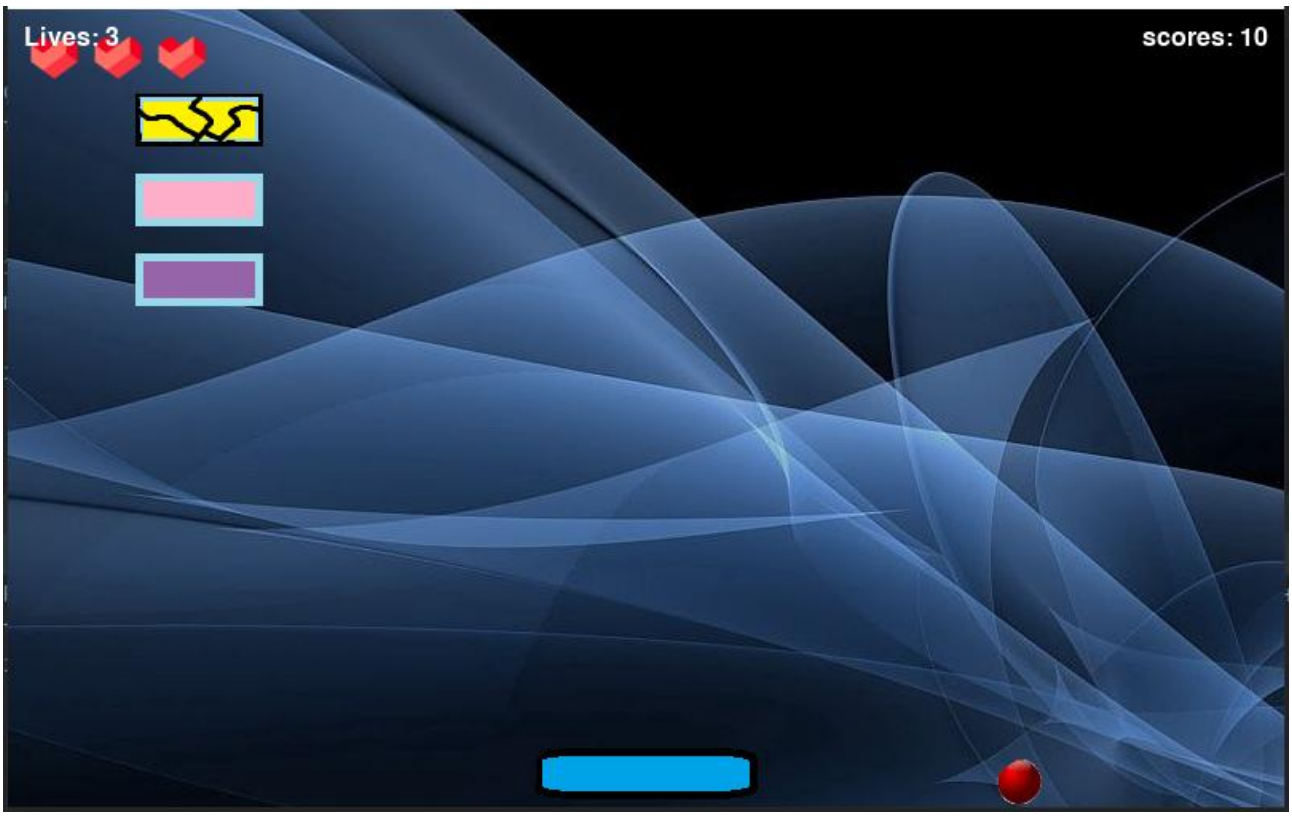

Рис. 6.2.2

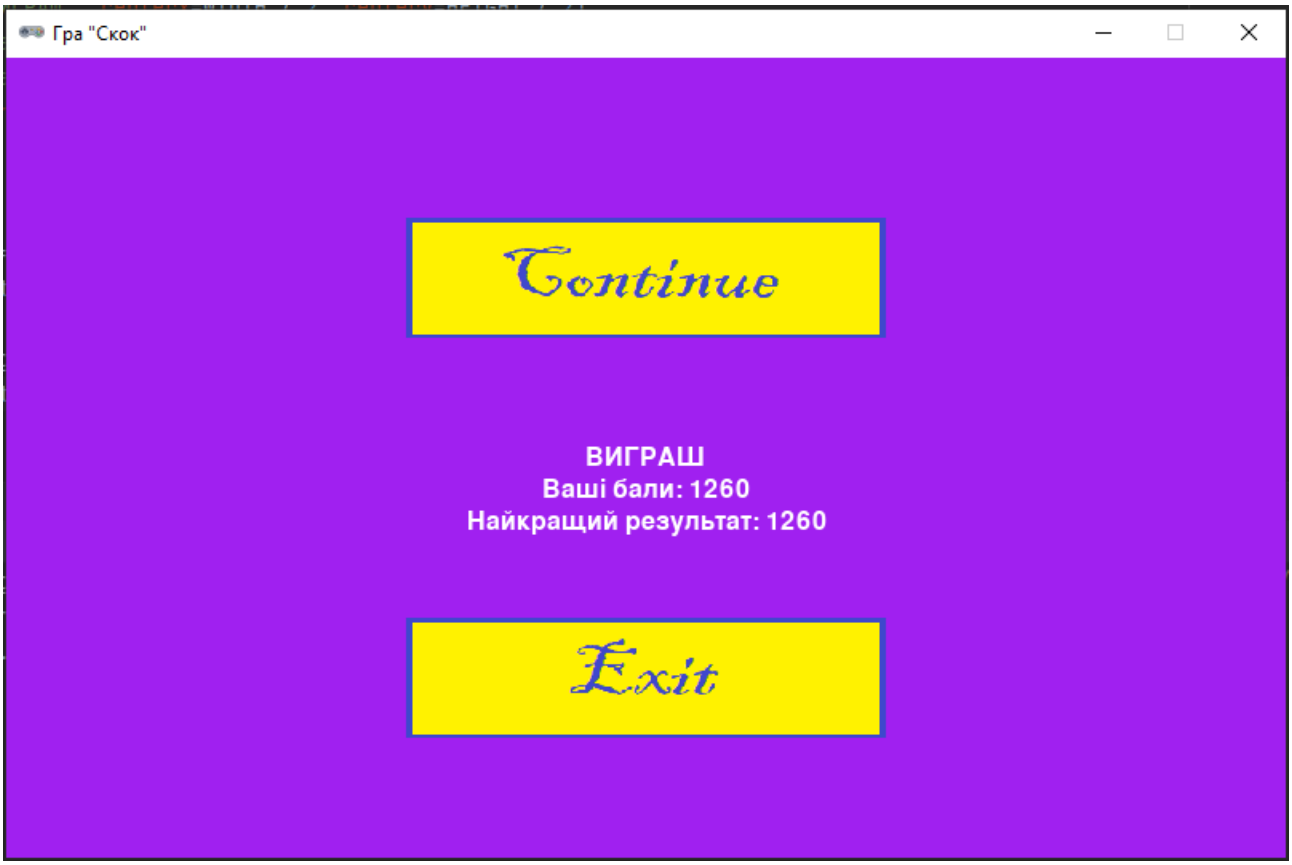

Рис. 6.2.3

Рис. 6.2.4

6.3. Програмний код

```
import pygame
import pygame.mixer
import pgzrun
from pgzero.keyboard import keyboard
from pgzero.builtins import Actor
from random import *


def start():
    pygame.init()


pygame.mixer.init()
TITLE = "Гра \"Скок\""
WIDTH = 800
HEIGHT = 500
'''Ініціалізація екрану'''
window = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Pygame")
surface = pygame.Surface((20, 200))
pygame.transform.scale(surface, ((100, 100)))
'''Ініціалізуємо ракетку'''
paddle = Actor("paddleblue.png")
'''Ініціалізуємо м'яч'''
ball = Actor("ballred2.png")
'''Характеристики ракетки '''
paddle.x = 100
paddle.y = 480
'''Характеристики м'яча '''
ball.x = 100
```

```
ball.y = HEIGHT / 2
ball_speed_x = -3
ball_speed_y = -3
bar_list = []
broke_bar_list = []
lives_list = []
lives = 3
game_over = True
scores = 0
list_of_scores = []
state = "play"
'''Місце зберігання різновидів цеглинок '''
colored_box_list = ["brick.png", "brick2.png", "brick3.png"]
colored_box_list2 = ["brick4.png"]
coin_speed_y = 2
coin_list = []
level = 0
num = 0
'''Ініціалізація звуків'''
sound_ball_rocket = pygame.mixer.Sound("sounds/rocket_ball_sound.wav")
sound_coin_rocket = pygame.mixer.Sound("sounds/money_sound.mp3")
sound_brick_crack = pygame.mixer.Sound("sounds/crack_sound.mp3")
sound_brick_full_crack = pygame.mixer.Sound("sounds/full_crack_sound.mp3")
bg_image = pygame.image.load("images/bg.jpg")
sound_ball_wall = pygame.mixer.Sound("sounds/ball_wall_sound.wav")
restart_image = pygame.image.load("images/restart_image.png")
start_image = pygame.image.load("images/start_image.png")
continue_image = pygame.image.load("images/continue_image.png")
exit_image = pygame.image.load("images/exit_image.png")


def draw():
    '''Метод відображення ігрових елементів'''
    global scores, state, list_of_scores, coin_list, ball_speed_x, ball_speed_y,
coin_speed_y, restart_image, lives, level

    screen.fill((113, 128, 87))
    w = 300
    h = 75
    new_start_image = pygame.transform.scale(start_image, (w, h))
    window.blit(new_start_image, (250, 100))  # point

    if level == 2:
        pass


    '''Алгоритм початку гри'''
    if not game_over:
        screen.fill((0, 128, 0))
        window.blit(bg_image, (0, 0))

        for bar in bar_list:
            bar.draw()

        paddle.draw()
        ball.draw()
        for coin in coin_list:
            coin.draw()

        for life in lives_list:
            life.draw()
        screen.draw.text("Lives: " + str(lives), topleft=(10, 10))
        screen.draw.text("scores: " + str(scores), topright=(790, 10))
        screen.draw.text("level: " + str(level), topright=(790, 30))
```

```
        # screen.draw.text("scores_list: " + str(list_of_scores), topright=(790,
30))


    elif state == "win":
        '''Випадок збиття всіх цеглин (перемога)'''
        ball_speed_x = 0  # заморозив м'яч на місці
        ball_speed_y = 0
        coin_speed_y = 0
        screen.fill("purple")
        screen.draw.text("ВИГРАШ", centerx=WIDTH / 2, centery=HEIGHT / 2)
        screen.draw.text("Ваші бали: {}".format(scores), centerx=WIDTH / 2,
centery=HEIGHT / 2 + 20)
        screen.draw.text("Найкращий результат: {}".format(max(list_of_scores)),
centerx=WIDTH / 2,
                         centery=HEIGHT / 2 + 40)

        w = 300
        h = 75
        new_continue_image = pygame.transform.scale(continue_image, (w, h))
        window.blit(new_continue_image, (250, 100))  # point

        new_exit_image = pygame.transform.scale(exit_image, (w, h))
        window.blit(new_exit_image, (250, 350))  # point

    elif state == "loose":
        '''Випадок нульової кількості життів (програш)'''
        ball_speed_x = 0  # заморозив м'яч на місці
        ball_speed_y = 0
        coin_speed_y = 0

        screen.fill("lightgreen")
        w = 300
        h = 75
        new_restart_image = pygame.transform.scale(restart_image, (w, h))
        window.blit(new_restart_image, (250, 100))  # point


        new_exit_image = pygame.transform.scale(exit_image, (w, h))
        window.blit(new_exit_image, (250, 350))  # point

        screen.draw.text("ПРОГРАШ", centerx=WIDTH / 2, centery=HEIGHT / 2)
        screen.draw.text("Ваші бали: {}".format(scores), centerx=WIDTH / 2,
centery=HEIGHT / 2 + 20)
        screen.draw.text("Найкращий результат: {}".format(max(list_of_scores)),
centerx=WIDTH / 2,
                         centery=HEIGHT / 2 + 40)



class Brick(Actor):
    def __init__(self, image, hits_required):
        super().__init__(image)
        self.hits_required = hits_required
        self.hits = 0


def place_bars(x, y, image, hits_reqired):
    '''Ініціалізація цеглинок'''
    global colored_box_list, num
    bar_x = x
    bar_y = y
    for i in range(num):
        bar = Brick(image, hits_reqired)
```

```
        bar.x = bar_x
        bar.y = bar_y
        bar_x += 100
        bar_list.append(bar)


def init_bars(x, y):
    global colored_box_list
    for colored_box in colored_box_list:
        place_bars(x, y, colored_box, "hits_required")
        y += 50


init_bars(120, 70)


def place_lives(x_l, y_l, image_l):
    global lives
    '''Ініціалізація життів'''
    life_x = x_l
    life_y = y_l
    for i in range(lives):
        life = Actor(image_l)
        life.x = life_x
        life.y = life_y
        life_x += 40
        lives_list.append(life)


place_lives(30, 30, "heart.png")


def update():
    global coin_speed_y, coin_list, ball_speed_y, ball_speed_x, scores,
bar_list, state, game_over, list_of_scores, colored_box_list2, broke_bar_list
    '''Оновлення стану гри'''
    if len(lives_list) == 0:
        place_lives(30, 30, "heart.png")

    for bar in bar_list:
        if ball.colliderect(bar):

            bar.hits += 1
            '''Алгоритм зміни графіки цеглинки після першого збиття'''
            if bar.hits == 1:
                sound_brick_crack.play()
                # ball_speed_y *= -1
                if bar.image == "brick.png":
                    bar.image = "brick4.png"
                elif bar.image == "brick2.png":
                    bar.image = "brick5.png"
                elif bar.image == "brick3.png":
                    bar.image = "brick6.png"

            elif bar.hits == 2:
                '''Алгоритм видалення зруйнованої цеглинки'''
                bar_list.remove(bar)
                sound_brick_full_crack.play()
                coin = Actor("m1f.png")
                coin.x = bar.x
                coin.y = bar.y
                coin_list.append(coin)

            scores += 10
```

```
            if len(bar_list) == 0:
                '''Нарахування балів за пройдений рівень'''
                scores += 1000
                list_of_scores.append(scores)
                state = "win"
                game_over = True
                ball_speed_x = 0
                ball_speed_y = 0
                coin_speed_y = 0
            ball_speed_y *= -1

            rand = randint(0, 1)
            if rand:
                ball_speed_x *= -1

    if ball.colliderect(paddle) and ball_speed_y > 0:

        ball_speed_y *= -1
        sound_ball_rocket.play()
        rand = randint(0, 1)
        if rand:
            ball_speed_x *= -1

    update_paddle()
    update_ball()
    update_coin()


def update_paddle():
    '''Взаємодія користувача з ракеткою'''
    if keyboard.left and paddle.x > 70:
        paddle.x -= 10
    if keyboard.right and paddle.x < WIDTH - 70:
        paddle.x += 10
    if keyboard.up:
        paddle.y -= 10
    if keyboard.down:
        paddle.y += 10


def update_coin():
    global coin_speed_y, coin_list, scores
    '''Алгоритм створення монетки, що випадає з цеглинки'''
    for coin in coin_list:
        coin_speed_y = 2
        coin.y += coin_speed_y
        if coin.colliderect(paddle) and not game_over:
            coin_list.remove(coin)
            scores += 100
            sound_coin_rocket.play()
        elif coin.y > HEIGHT:
            coin_list.remove(coin)


def update_ball():
    '''Оновлення м'яча'''
    global ball_speed_x, ball_speed_y, screen, game_over, lives, state, scores,
list_of_scores, coin_speed_y
    if not game_over:
        ball.x += ball_speed_x
        ball.y += ball_speed_y
        if ball.x >= WIDTH - 2 or ball.x <= 0:
            ball_speed_x *= -1
            sound_ball_wall.play()
```

```
        elif ball.y <= 0:
            ball_speed_y *= -1
            sound_ball_wall.play()
        elif ball.y >= HEIGHT:
            ball_speed_y *= -1
            sound_ball_wall.play()

            if len(lives_list) > 0:
                lives_list.pop(int(len(lives_list) - 1))
                lives -= 1

    if lives == 0:
        list_of_scores.append(scores)
        if state == "play":
            state = "loose"
        game_over = True  # Кінець гри
        ball_speed_x = 0  # заморозив м'яч на місці
        ball_speed_y = 0
        coin_speed_y = 0


def restart_game():
    '''Перезапуск гри'''
    global game_over, lives, ball_speed_x, ball_speed_y, scores, lives_box,
state, lives_list, scores, coin_list, lives_list
    coin_list = []
    game_over = False
    state = 'play'
    lives = 3
    paddle.x = WIDTH / 2
    paddle.y = 480
    init_bars(120, 70)
    ball_speed_x = -3
    ball_speed_y = -3
    ball.x = WIDTH / 2
    ball.y = HEIGHT / 2

    if ball.y >= HEIGHT:
        ball_speed_y *= -1
        if len(lives_list) > 0:
            lives_list.pop(int(len(lives_list) - 1))
            lives -= 1



def start():
    global game_over
    game_over = False

def on_mouse_down(pos):
    '''Взаємодія користувача з клавішами меню'''
    global lives, state, scores, level, num, exit_image
    x, y = pos

    if 1 < x < 100 and 1 < y < 100:
        start()
        state = 'play'
    elif 250 <= x <= 550 and 75 <= y <= 175:
        if state == 'loose':
            scores = 0
            num = 1
        level += 1
        num += 1
        restart_game()
```

```
    elif 250 <= x <= 550 and 350 <= y <= 425:
        if state == 'loose':
            quit_game()


def pause():
    global ball_speed_x, ball_speed_y
    ball_speed_x = 0
    ball_speed_y = 0

def quit_game():
    quit()




pgzrun.go()
```

6.4 Тестування програмного забезпечення

Користувач може розпочати гру натисканням кнопки під назваою «Start» у головному вікні програми.
Після цього користувач може керувати ракеткою за допомогою клавіш.
Користувач може ловити монетки ракеткою.
Користувач може завершити гру виграшем або програшем.
Користувач може почати гру заново або продовжити гру.
Всі колізії об’єктів перевірені і відбуваються коректно.

7. Опис гри та її правила

Гра "Скок» - це гра, в якій гравець керує платформою, яка знаходиться внизу екрану, і має відбивати кулю, яка летить вгору. На екрані знаходяться різноманітні блоки, які потрібно знищити, ударяючи по них кулю платформою. Як тільки куля вдаряє в блок, блок зникає, і гравець отримує певну кількість очок. Якщо куля вдаряється в стіну зліва або справа, вона відскакує в протилежному напрямку. Якщо куля вдаряється в платформу, вона також відскакує в протилежному напрямку.

У грі є декілька типів блоків з різною міцністю, які потрібно знищити. Найпростіші блоки знищуються одним ударом кулі, тоді як більш міцні блоки потребують кількох ударів. Деякі блоки мають спеціальні властивості, такі як додаткові життя, збільшення або зменшення платформи, зміна швидкості кулі та інші.

Мета гри полягає в тому, щоб знищити всі блоки на екрані, отримуючи якомога більше очок. Якщо куля вдаряється в нижню стіну, гравець втрачає одне зі своїх життів. Гравець має обмежену кількість життів, і якщо він втрачає всі свої життя, гра закінчується.

Гравець може також отримати додаткові життя або бонуси, збиваючи певні блоки або залишки після знищення блоків.

Огляд результатів роботи: у ході даної роботи, був створений різновид популярної гри, програмне забезпечення створене на Пайтоні. Я ознайомився з бібліотекою Pygame, яка ідеально підходить для написання ігор на Пайтоні початківцями і не тільки. Я вважаю, що тим, хто хоче навчитися створювати ігри, також буде корисно ознайомитися з даною бібліотекою. Вона містить готові ігрові, елементи, доволі інтуїтивна поведінка функцій, існує багато готових шаблонних функцій, необхідних для взаємодії об’єктів гри.

Перспективи подальшого розвитку гри: гру можно удосконалювати значним чином. Можна допрацювати графіку, дизайн гри, пропрацювати детальніше функціональність та алгоритмічність, передбачити можливість додавати гравців, створити більшу кількість проходження рівнів.

ВИСНОВОК

У ході роботи над проєктом, я створив гру «Скок» з графічним інтерфейсом, що є різновидом популярної гри «Арканоїд». Вивчив теоретичні знання щодо бібліотеки Pygame та її пакетів, а також закріпив їх застосування на практиці.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

https://www.pygame.org/news
https://ua.depositphotos.com/vector-images/%D1%81%D0%BF%D1%80%D0%B0%D0%B9%D1%82.html