МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Ижевский государственный технический университет имени М.Т. Калашникова»

(ФГБОУ ВО «ИжГТУ имени М.Т. Калашникова»)

Факультет «Информатика и вычислительная техника»

Кафедра «Программное обеспечение»

Направление подготовки 09.03.04 «Программная инженерия»

ОТЧЕТ ПО ПРАКТИКЕ

|  |
| --- |
|  |
| *(наименование типа практики)* |
| *(полное наименование профильной организации)*  Выполнил обучающийся /\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Добрыдень Данил Дмитриевич, 1 курс, Б23-191-3 |
| *(подпись) (ФИО, курс, номер группы)*  Дата сдачи отчета: « » 2024 г. |
| Дата аттестации « » 20 г. |
| Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Руководитель практики от  ИЖГТУ имени М.Т. Калашникова / \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ |
| *(подпись) (И.О. Фамилия, должность, ученая степень)*  Заведующий кафедрой /\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ |
| *(подпись) (И.О. Фамилия, должность, ученая степень)* |

ВВЕДЕНИЕ

Цель проекта - разработка консольного приложения, позволяющего

работать с графическими объектами и добавлять анимационные эффекты на произвольную тему. В процессе выполнения проекта будут изучены теоретические основы создания анимации, а также реализована анимация на выбранную тему.

Цель практики - закрепление пройденного материала, а также самостоятельное изучение дополнительной информации.

Задачи учебной практики:

1. Реализация анимационных эффектов.
2. Реализация обработки событий клавиатуры (управление клавишами).
3. Использование рекурсии (фрактала).
4. Использование случайных чисел.
5. ОСНОВНАЯ ЧАСТЬ

1.1 Словесное описание проекта

В данной работе рассматривается создание простой версии популярной игры “Flappy Bird”.

Основная цель проекта — разработка игры с использованием библиотеки pygame.

Постановка задачи

Проект представляет собой аркадную игру, где игрок управляет птицей посредством нажатия на левую кнопку мыши, либо на пробел, чтобы пролететь между зелеными трубами и не врезаться ни в них, ни в небо, ни в землю. Цель игры — продержаться как можно дольше, набирая очки за каждую преодоленную трубу.

Основные задачи:

1. Разработка игровой механики управления птицей.
2. Создание динамических препятствий.
3. Обеспечение интерфейса пользователя.
4. Оптимизация производительности игры.

Ожидаемый результат: работающая игра, с начислением очков и интересным геймплеем.

1.2 Математическая постановка

В основе игры лежит несколько математических задач:

**Движение птицы под воздействием гравитации**:

Позиция птицы обновляется по формуле: , где ​ — начальная позиция, — скорость, g — гравитация.

**Перемещение фона и препятствий**:

Скорость движения объектов задана константами и изменяется с течением времени:.

Анимация взмаха крыльями птицы реализована с помощью формулы , где frame принимает значение от 1 до 3, что обозначает положение крыльев, k – константа, равная 0.2, отвечающая за скорость смены анимации и t – константа, равная 3, отвечающая за количество различных положений крыльев.

1.3 Описание программной реализации:

Основные действия программы выполняются в цикле while, пока булевая переменная play равна True.

Создание фона приложения

Игра реализована так, что двигаются по оси Y трубы и задний фон, следовательно, для оптимизации, фон приложения копируется и удаляется, когда выходит за границы видимости игрока.

*Создание труб*

Трубы создаются в цикле for, скорость их появление и частота – константы, после того как трубы оказываются за границей видимости игрового поля, они удаляются из массива.

*Состояния игры:*

1. *Старт игры*

Пока игрок не нажмет на одну из кнопок(левая кнопка мыши или клавиша пробел) фон статичен, птица находится в левой части экрана и так же статична, трубы не создаются.

1. *Процесс игры*

Как только, игрок нажимает на левую кнопку мыши или на клавишу пробел, игра начинается, фон повторяется и трубы начинают создаваться, первые две трубы создают пространство для прохода ровно посередине, последующие создаются рандомным способом.

После прохождения между каждыми трубами, игроку начисляется 10 очков и при достижении каждой новой сотни очков, игра ускоряется, тем самым усложняя прохождение.

1. *Столкновение*

При столкновении с трубами, небом или нижней частью экрана, птица возвращается на начальные координаты, игрок теряет одну из своих трёх жизней и продолжает пока жизни не закончатся. Тогда игра завершается, игрок теряет все накопленные очки.

*Анимация полёта птицы*

Анимация взмаха крыльями птицы реализована с помощью формулы , где frame принимает значение от 1 до 3, что обозначает положение крыльев, k – константа, равная 0.2, отвечающая за скорость смены анимации и t – константа, равная 3, отвечающая за количество различных положений крыльев.

***Цикл программы:***

1) Обработка событий, таких как нажатие клавиши пробел и левой кнопки мыши.

2) Обновление и отрисовка объектов на экране.

3) Проверка столкновений и завершение игры при нуле жизней.

4) Обновление и отображение счета игрока, ускорение игры при достижении каждой 100 очков.

*Визуальная составляющая****:***  
Использование текстур для отображения труб, птицы и заднего фона.  
Позиционирование каждого элемента на экране в соответствии с его координатами на игровом поле.

***Управление:***

Для маневрирования между трубами используются одиночные или многократные нажатия левой кнопки мыши или клавиши пробел для прыжка

1.4 Листинг текста программы

Файл main.py:

import pygame  
from random import randint  
  
pygame.init()  
  
WIDTH, HEIGHT = 800, 600  
FPS = 60  
  
window = pygame.display.set\_mode((WIDTH, HEIGHT))  
clock = pygame.time.Clock()  
  
pygame.display.set\_caption('Flappy Bird')  
pygame.display.set\_icon(pygame.image.load('images/1.png'))  
  
font1 = pygame.font.Font(None, 35)  
font2 = pygame.font.Font(None, 80)  
  
imgBG = pygame.image.load('images/background.png')  
imgbird = pygame.image.load('images/0.png')  
imgPT = pygame.image.load('images/top.png')  
imgPB = pygame.image.load('images/bottom.png')  
  
py, sy, ay = HEIGHT // 2, 0, 0  
player = pygame.Rect(WIDTH // 3, py, 40, 30)  
frame = 0  
  
state = 'start'  
timer = 10  
  
pipes = []  
bges = []  
pipescores = []  
  
pipespeed = 3  
pipegatesize = 200  
pipeGatePos = HEIGHT // 2  
  
bges.append(pygame.Rect(0, 0, 390, 708))  
  
lives = 3  
scores = 0  
  
play = True  
while play:  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT: play = False  
 press = pygame.mouse.get\_pressed()  
 keys = pygame.key.get\_pressed()  
 click = press[0] or keys[pygame.K\_SPACE] \\ определение нажатия клавиш  
  
 if timer: timer -= 1  
  
 frame = (frame + 0.2) % 3 \\ Анимация движения крыльев  
  
 for i in range(len(bges)-1, -1, -1):  
 bg = bges[i]  
 bg.x -= pipespeed // 2  
  
 if bg.right < 0: bges.remove(bg)  
  
 if bges[-1].right <= WIDTH: bges.append(pygame.Rect(bges[-1].right, 0, 390, 708))  
 \\ движение заднего фона  
  
 for i in range(len(pipes)-1, -1, -1):  
 pipe = pipes[i]  
 pipe.x -= pipespeed  
  
 if pipe.right < 0:  
 pipes.remove(pipe)  
 if pipe in pipescores: pipescores.remove(pipe)  
 \\ удаление труб

if state == 'start':  
 if click and not timer and not len(pipes): state = 'play'  
  
 py += (HEIGHT // 2 - py) \* 0.1  
 player.y = py  
 \\ статичное состояние до нажатия клавиш  
 elif state == 'play':  
 if click: ay = -2  
 else: ay = 0  
  
 py += sy  
 sy = (sy + ay + 1) \* 0.98  
 player.y = py  
 \\ начало игры и физика движения птицы

if not len(pipes) or pipes[-1].x < WIDTH - 200:  
 pipes.append(pygame.Rect(WIDTH, 0, 98, pipeGatePos - pipegatesize // 2))  
 pipes.append(pygame.Rect(WIDTH, pipeGatePos + pipegatesize // 2, 98, HEIGHT - pipeGatePos + pipegatesize // 2))  
  
 pipeGatePos += randint(-100,100)  
 if pipeGatePos < pipegatesize: pipeGatePos = pipegatesize  
 elif pipeGatePos > HEIGHT - pipegatesize: pipeGatePos = HEIGHT - pipegatesize  
 \\ рандомное появление труб и условие, чтобы трубы создавались в пределах игровой зоны

if player.top < 0 or player.bottom > HEIGHT: state = 'fall'  
 \\ проверка на падение или удар об небо птицы

for pipe in pipes:  
 if player.colliderect(pipe): state = 'fall'  
  
 if pipe.right < player.left and pipe not in pipescores:  
 pipescores.append(pipe)  
 scores += 5  
 pipespeed = 3 + scores // 100

\\ начисление очков за пройденные трубы и ускорение игры

elif state == 'fall':  
 sy, ay = 0, 0  
 pipeGatePos = HEIGHT // 2  
  
 lives -= 1  
 if lives:  
 state = 'start'  
 timer = 60  
 else:  
 state = 'game over'  
 timer = 150  
 \\ прекращение игры в случае, когда закончились жизни  
  
 else:  
 py += sy  
 sy = (sy + ay + 1) \* 0.98  
 player.y = py  
 \\ падение птицы за границы экрана и возвращение в начальные координаты  
 if not timer: play = False  
  
 for bg in bges: window.blit(imgBG, bg)  
  
 for pipe in pipes:  
 if not pipe.y:  
 rect = imgPT.get\_rect(bottomleft = pipe.bottomleft)  
 window.blit(imgPT, rect)  
 else:  
 rect = imgPB.get\_rect(topleft = pipe.topleft)  
 window.blit(imgPB, rect)  
 \\ отрисовка труб  
  
  
 image = imgbird.subsurface(40 \* int(frame), 0, 40, 30)  
 image = pygame.transform.rotate(image, -sy \* 2.5)  
 window.blit(image, player)  
 \\ отрисовка птицы и анимация крыльев

text = font1.render('Очки: ' + str(scores), 1, pygame.Color('black'))  
 window.blit(text, (10,10))  
 \\ счётчик очков

text = font1.render('Жизни: ' + str(lives), 1, 'black')  
 window.blit(text, (10, HEIGHT - 30))  
 \\ счётчик жизней

pygame.display.update()  
 clock.tick(FPS)  
  
pygame.quit()

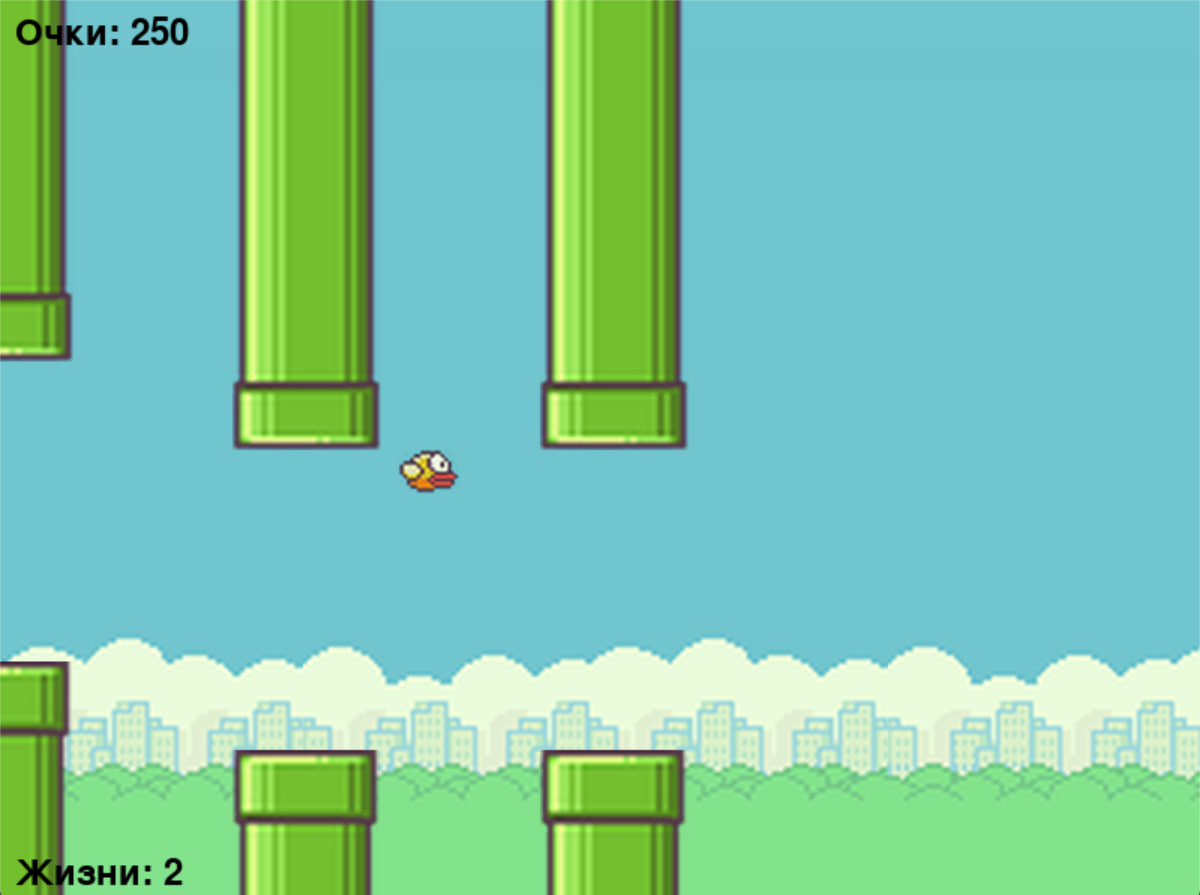
Тестовые примеры работы

Рисунок 1 - Пример работы программы 1

ЗАКЛЮЧЕНИЕ

В рамках данного проекта была реализована аркадная игра Flappy Bird на языке Python с использованием библиотеки pygame. Были решены задачи по созданию и управлению объектами игры, обработке коллизий и добавлению анимации. В игру также можно добавить таблицу лидеров по набранным очкам и доработать геймплей, сделать его более разнообразным.

Добавление таблицы лидеров в игру позволит игрокам соревноваться за высокие места и стимулировать их к повышению своего счета. Таблица лидеров может отображать лучшие результаты игроков, их имена или никнеймы, количество очков, достигнутых в игре. Это может стать дополнительным источником мотивации для игроков и создать дополнительный интерес к игре.

В доработке геймплея можно добавить новые уровни сложности, различные бонусы или препятствия, чтобы игра стала более динамичной и увлекательной. Также можно рассмотреть возможность добавления выбора персонажа игроком, что может добавить в игру элементы персонализации и разнообразия.

Доработка геймплея также может включать работу над улучшением управления, добавление новых механик игры, исправление ошибок и улучшение общей оптимизации игры.

В целом, развитие проекта позволит не только улучшить игровой опыт игроков, но и увеличить активность и участие пользователей в игре. Добавление новых возможностей и улучшений поможет продлить жизнь игры и привлечь новых игроков.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Библиотека Pygame: [Электронный ресурс]. https://habr.com/ru/articles/588605/ (Дата обращения: 26.05.2024)
2. Андреева, О. В. Основы алгоритмизации и программирования на языке Python : учебник / О. В. Андреева, О. И. Ремизова. — Москва : Издательский Дом МИСиС, 2022. — 149 c.
3. Инди курс по Pygame - учимся создавать свои игры: [Электронный ресурс]. https://stepik.org/course/118434/promo (Дата обращения: 05.05.2024)