Итоговый проект по дисциплине «Основы алгоритмизации и программирования» «Bullfinch» игра на языке программирования python

Группа ИС-24

Выполнили и разработали

Соколова Екатерина

и Линник Виктория

Преподаватель:Манакова Ольга Петровна

Задачи нашего проекта

- Написать программный код с разумной логикой, протестировать его и решить возможные ошибки
- Разработать интуитивно понятный интерфейс
- Сделать идею дизайна уникальной, оригинальной
- Сделать приложение с определенной практической ценностью

Принятие решения

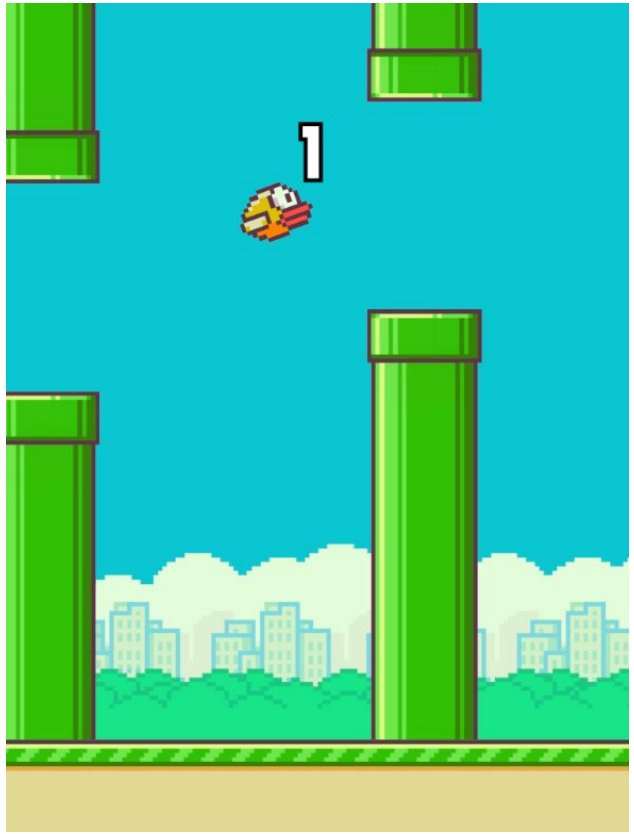
- Наш глаз пал на уже многими полюбившуюся игру Flappy Bird, которую мы решили не просто скопировать, а переделать дизайн под наш русский колорит. Главным героем стал снегирь, на заднем фоне лес, а обычные трубы теперь украшены снегом и гирляндами.
- Также мы решили, что нам нужен хоть какой-то минимальный интерфейс, чтобы обеспечить комфорт обычного пользователя, мы ввели меню, музыку в меню и в геймплей.

Дизайн программы

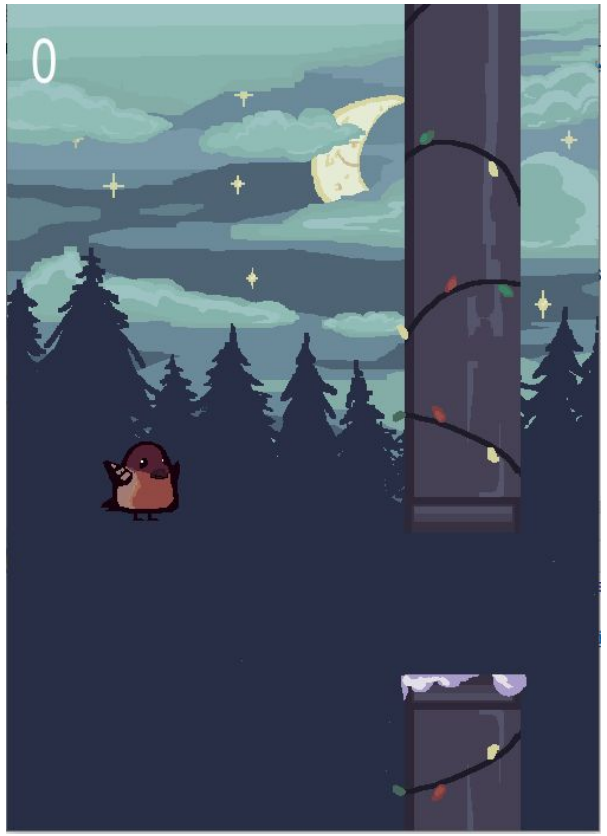
Почему дизайн в нашей программе такой, какой есть:

- Старый дизайн был слишком ярким, цветовая палитра стала более нежной, чтобы каждый человек, открывая игру, мог не напрягать свое зрение;
- Графика более проработанная — игра с оригинальным дизайном, так как обычный дизайн Flappy Bird многим надоел, а с обновленным дизайном, игра выглядит новее и свежее;
- Добавлена расслабляющая музыка — так как игра иногда доставляет нервозность при проигрыше, выбор с расслабляющей музыкой, кажется, логичным;

Сравнение двух версий



старая версия

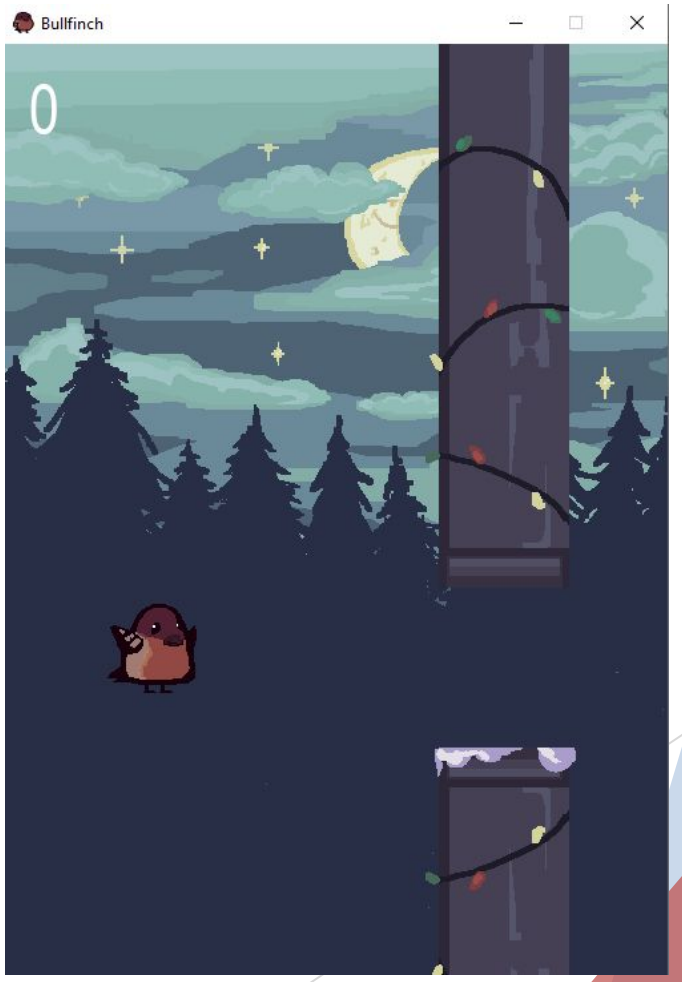


новая версия

Интерфейс (более подробное рассмотрение)

<- Экран меню

Игровой процесс ->



Демонстрация в видео:

Практическое значение программы

- Игра предназначена для коротания времени и возможно соревнования вместе с друзьями. В ней вам предстоит сыграть за птичку, в нашем случае снегиря, что летит по зимнему лесу и пытается не врезаться в трубы.
- Мы выбрали именно эту игру, потому что многие наши ровесники играли в нее в детстве, она осталась в памяти как приятное воспоминание, которое мы сделали лучше.

Библиотеки

- В данной программе мы использовали несколько библиотек:
- Pygame – основная библиотека для создания графического интерфейса
- Random – для того, чтобы трубы появлялись в на случайной высоте. А также sys (в ней содержатся некоторые переменные, которые тесно связаны с интерпретатором)
- Tkinter — кросс-платформенная событийно-ориентированная графическая библиотека на основе средств Tk, написанная Стином Лумхольтом и Гвидо ван Россумом. Входит в стандартную библиотеку Python. С ее помощью мы реализовали кнопки в меню.

Код программы

Импорты библиотек -

```
from tkinter import *
import pygame
from pygame.locals import *  # noqa
import sys
import random
```

Создание дисплея и
Некоторая подготовка, -
описанная в комментариях.

```
screen = pygame.display.set_mode((500, 700))

#загрузка image для меню
background = pygame.image.load("assets/background.png").convert()
start_img = pygame.image.load("assets/button_start.png").convert_alpha()
exit_img = pygame.image.load("assets/button_quit.png").convert_alpha()

#загрузка иконки-лого
icon = pygame.image.load("assets/icon.png").convert_alpha()

#меняем в окне название на нашу игру и иконку игры тоже меняем
pygame.display.set_icon(icon)
pygame.display.set_caption("Bullfinch")

#подключение музыки
pygame.mixer.init()
music = pygame.mixer.music.load('music/bird.mp3')
pygame.mixer.music.play(-1)
pygame.mixer.music.set_volume(0.1)
```

Код программы

Создание класса, что будет Использоваться для создания и работы кнопок (универсален, подходит для любой программы)

```
class Button():
    new *
    def __init__(self, x, y, image, scale):
        width = image.get_width()
        height = image.get_height()
        self.image = pygame.transform.scale(image, (int(width * scale), int(height * scale)))
        self.rect = self.image.get_rect()
        self.rect.center = (x, y)
        self.clicked = False
    new *
    def draw(self):
        action = False
        pos = pygame.mouse.get_pos()

        if self.rect.collidepoint(pos):
            if pygame.mouse.get_pressed()[0] == 1 and self.clicked == False:
                self.clicked = True
                action = True

        if pygame.mouse.get_pressed()[0] == 0:
            self.clicked = False

        screen.blit(self.image, (self.rect.x, self.rect.y))
        return action

start_button = Button(250, 490, start_img, 1)
exit_button = Button(250, 600, exit_img, 1)
```

Код программы

Создание основного класса и функция с переменными, которые потребуются нам позже. Также загрузка изображений (спрайтов, фона).

```
class Bullfinch:
# функция с переменными
    new *
    def __init__(self):
        self.screen = pygame.display.set_mode((500, 700))
        self.bird = pygame.Rect(65, 50, 50, 50)
        self.background = pygame.image.load("assets/background.png").convert()
        self.birdSprites = [pygame.image.load("assets/1.png").convert_alpha(),
                            pygame.image.load("assets/2.png").convert_alpha(),
                            pygame.image.load("assets/dead.png")]
        self.wallUp = pygame.image.load("assets/bottom.png").convert_alpha()
        self.wallDown = pygame.image.load("assets/top.png").convert_alpha()
        self.gap = 130
        self.wallx = 400
        self.birdY = 350
        self.jump = 0
        self.jumpSpeed = 10
        self.gravity = 5
        self.dead = False
        self.sprite = 0
        self.counter = 0
        self.offset = random.randint(-110, 110)
```

Код программы

Создание функции, позволяющей трубам появляться в случайном положении

```
def updateWalls(self):
    self.wallx -= 2
    if self.wallx < -80:
        self.wallx = 400
        self.counter += 1
        self.offset = random.randint(-110, 110)
```

Код программы

Создание функции, дающей снегирю возможность прыгать, а также обработка его "смерти" (столкновения с трубами)

```
def birdUpdate(self):
    if self.jump:
        self.jumpSpeed -= 1
        self.birdY -= self.jumpSpeed
        self.jump -= 1
    else:
        self.birdY += self.gravity
        self.gravity += 0.2
    self.bird[1] = self.birdY
    upRect = pygame.Rect(self.wallx,
                         360 + self.gap - self.offset + 10,
                         self.wallUp.get_width() - 10,
                         self.wallUp.get_height())
    downRect = pygame.Rect(self.wallx,
                           0 - self.gap - self.offset - 10,
                           self.wallDown.get_width() - 10,
                           self.wallDown.get_height())
    if upRect.colliderect(self.bird):
        self.dead = True
    if downRect.colliderect(self.bird):
        self.dead = True
    if not 0 < self.bird[1] < 720:
        self.bird[1] = 50
        self.birdY = 50
        self.dead = False
        self.counter = 0
        self.wallx = 400
        self.offset = random.randint(-110, 110)
        self.gravity = 5
```

Код программы

Создание основной функции запуска программы с обработкой событий с клавиатуры, мыши, задача fps и использование предыдущих функций.

```
def run(self):
    clock = pygame.time.Clock()
    pygame.font.init()
    font = pygame.font.SysFont("Arial", 50)
    while True:
        clock.tick(60)
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                sys.exit()
            if (event.type == pygame.KEYDOWN and event.key == pygame.K_SPACE or event.type == pygame.MOUSEBUTTONDOWN) and not self.dead:
                self.jump = 17
                self.gravity = 5
                self.jumpSpeed = 10
            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_ESCAPE:
                    game_paused = True
                    return game_paused

        self.screen.fill((255, 255, 255))
        self.screen.blit(self.background, (0, 0))
        self.screen.blit(self.wallUp,
                         (self.wallx, 360 + self.gap - self.offset))
        self.screen.blit(self.wallDown,
                         (self.wallx, 0 - self.gap - self.offset))
        self.screen.blit(font.render(str(self.counter),
                                     -1,
                                     (255, 255, 255)),
                         (20, 20))
        if self.dead:
            self.sprite = 2
        elif self.jump:
            self.sprite = 1
        self.screen.blit(self.birdSprites[self.sprite], (70, self.birdY))
        if not self.dead:
            self.sprite = 0
        self.updateWalls()
        self.birdUpdate()
        pygame.display.update()
```

Код программы

Запуск меню и самой программы.

```
run = True
while run:
    screen.blit(background, (0, 0))
    if start_button.draw():
        if __name__ == "__main__":
            Bullfinch().run()
    if exit_button.draw():
        run = False

    for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    run = False
    pygame.display.update()
pygame.quit()
```

Тестировка

- Мы прислали нашу программу нескольким знакомым, чтобы проверить совместимость с другими компьютерами и найти возможные ошибки и баги. Проблем не было обнаружено, потенциальные пользователи оценили продукт.
- Принято решение в будущем доработать программу добавив в нее:
    1. Систему рекордов, для соревнования между друзьями;
    2. Экран проигрыша игры и создание цикла запуска игрового процесса снова;
    3. Пауза в игре;

Завершение

- В ходе этого проекта была разработана игра, основанная на «Flappy Bird». Цель и задачи были выполнены, готовый год протестирован и выложен на GitHub.

Ура! Мы научились писать мини-игры на pythone и на движке pygame!