

Использование библиотеки Pygame для разработки обучающей логической игры 2048

Языков Кирилл Александрович

*Приамурский государственный университет имени Шолом-Алейхема
студент*

Аннотация

В данной работе представлен разработанный с использованием языка Python и библиотеки Pygame вариант популярной логической игры 2048. Проект направлен на создание удобного и визуально привлекательного игрового приложения с базовой игровой механикой, звуковым сопровождением и системой рекордов. Особое внимание уделено интерфейсу, пользовательскому опыту и расширяемости кода для дальнейших исследований в области интерактивных игр и образовательного программирования.

Ключевые слова: Python, Pygame, 2048, логическая игра, программирование, игровая механика, пользовательский интерфейс, звуковое сопровождение, система рекордов.

Using the Pygame library to develop an educational logic game 2048

Yazykov Kirill Alexandrovich

*Sholom-Aleichem Priamursky State University
Student*

Abstract

This paper presents a variant of the popular logic game 2048 developed using Python and the Pygame library. The project aims to create a user-friendly and visually appealing game application with basic game mechanics, soundtrack and a high score system. Special attention is paid to the interface, user experience and extensibility of the code for further research in the field of interactive games and educational programming

Keywords: Python, Pygame, 2048, logic game, programming, game mechanics, user interface, soundtrack, high score system.

1 Введение

1.1 Актуальность

Разработка интерактивных приложений и игр остаётся важной областью компьютерных наук и образовательных технологий. Игра 2048, благодаря своей простоте и увлекательности, часто используется как обучающий проект для начинающих программистов. Современные решения требуют улучшения

визуализации и пользовательского опыта, что стимулирует развитие навыков работы с графическими библиотеками и мультимедийными компонентами.

1.2 Обзор исследований

В своей работе Е.О. Новиков подробно объясняет алгоритм работы игры и проводит глубокий анализ возможностей Pygame для создания более сложных игр. Особое внимание уделено методам, связанным с графикой и взаимодействием с пользователем, что дает возможность разрабатывать динамичные игры с графическим интерфейсом [1]. В статье А.А. Шамышева исследуется применение графической библиотеки PyGame как средства для обучения программированию. Автор подчеркивает, что благодаря визуальной составляющей и доступности инструментов, PyGame позволяет эффективно вовлекать студентов в процесс написания кода, создавая понятные и интересные проекты. Отмечается, что создание интерактивных приложений с использованием PyGame способствует развитию алгоритмического мышления и мотивации к обучению [2]. Работа И.А. Чиркова описывает создание 2D игры с использованием библиотеки PyGame, ориентированной на соревновательное взаимодействие двух игроков. Игра реализована для платформы Windows и предоставляет игрокам возможность управления персонажами, а также взаимодействия с игровым окружением. Автор демонстрирует потенциал PyGame в реализации многопользовательских игровых механик и подчеркивает важность практического применения библиотеки для разработки полноценных игровых проектов [3]. М.М.Белолобский и М.А. Сорочинский в своей статье провели анализ ключевых функциональных возможностей библиотеки PyGame, включая её модули для обработки графики, звука и пользовательского ввода. Авторы рассматривают PyGame как «движок» для создания 2D игр и мультимедийных приложений, подчеркивая его доступность и гибкость, что делает его эффективным инструментом в образовательных и любительских проектах [4]. В исследовании П.В. Кулинчи рассмотрена роль библиотеки PyGame в современном геймдеве. Автор акцентирует внимание на её востребованности среди начинающих разработчиков благодаря простоте освоения и широким возможностям. Отмечено, что игры на PyGame могут не только служить учебными проектами, но и становиться коммерчески успешными, особенно при достаточном уровне креативности и технической проработки. В статье приводится обоснование, почему PyGame остаётся актуальным инструментом в сфере 2D-разработки [5].

1.3 Цель исследования

Создать полнофункциональную, красочную и удобную для пользователя версию игры 2048 с использованием Python и Pygame, обеспечив при этом звуковые эффекты и сохранение рекордов. Рассмотреть основные методы реализации игровой логики и улучшения визуального восприятия.

2 Материалы и методы

Для разработки игры 2048 использовался язык программирования Python версии 3.11. В качестве библиотеки для создания графического пользовательского интерфейса и визуализации игрового поля применена библиотека Pygame версии 2.6.1, которая обеспечивает удобные методы для отрисовки графики, обработки пользовательских событий и работы со звуком.

Основные методы реализации включают алгоритмы сдвига и объединения плиток по правилам игры, обработку событий клавиатуры для управления движением, а также динамическое обновление интерфейса. Для улучшения визуального восприятия используется цветовое кодирование значений плиток и отображение текущего счёта с рекордом.

Звуковое сопровождение реализовано с помощью модуля pygame.mixer, обеспечивающего воспроизведение эффектов при слиянии плиток и добавлении новых. Для сохранения данных о рекорде применяется работа с локальным файлом, позволяющая сохранять и загружать максимальный счёт между сессиями.

Структура кода построена модульно, что обеспечивает удобство расширения функционала, например, добавления анимации или новых игровых режимов. Основное внимание уделено сочетанию простоты реализации и визуальной привлекательности, а также стабильности работы программы.

3 Результаты и обсуждения

1. Импорт библиотек и инициализация Pygame (рис. 1).

A screenshot of a code editor with a dark background. The code is written in Python and shows the following lines:

```
import pygame
import random
import sys
import os

pygame.init()
```

Рисунок 1 – Импорт библиотек

В этом блоке импортируем необходимые библиотеки:

- pygame — для графики и звука;
- random — для случайного выбора позиций и значений плиток;
- sys и os — для работы с файлами и системой.

Затем вызываем `pygame.init()`, чтобы запустить все модули Pygame.

2. Настройка игрового окна и основных параметров (рис. 2).

```
WIDTH, HEIGHT = 500, 700
SCREEN = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("2048 Красочная версия")
```

Рисунок 2 – Настройка окна и названия

Здесь задаются размеры окна и создаётся объект экрана SCREEN, куда будет выводиться графика. Также устанавливается заголовок окна.

3. Определение цветов и шрифтов (рис. 3).

```
BACKGROUND_COLOR = (250, 248, 239)
TILE_COLORS = {
    0: (205, 193, 180),
    2: (238, 228, 218),
    4: (237, 224, 200),
    8: (242, 177, 121),
    16: (245, 149, 99),
    32: (246, 124, 95),
    64: (246, 94, 59),
    128: (237, 207, 114),
    256: (237, 204, 97),
    512: (237, 200, 80),
    1024: (237, 197, 63),
    2048: (237, 194, 46)
}

FONT = pygame.font.SysFont("arial", 40, bold=True)
SCORE_FONT = pygame.font.SysFont("arial", 30, bold=True)
TITLE_FONT = pygame.font.SysFont("arial", 60, bold=True)
```

Рисунок 3 – Код определения цветов и шрифтов

Здесь задаются цвета для фона и плиток в зависимости от их значений. Для текста задаются разные шрифты для плиток, счёта и заголовка.

4. Настройка игрового поля (рис. 4).

```
GRID_SIZE = 4
TILE_SIZE = WIDTH // GRID_SIZE
```

Рисунок 4 – Код настройки игрового поля

Определяется размер сетки 4x4 и вычисляется размер каждой плитки на основе ширины окна.

5. Звуковые эффекты и файл для рекорда (рис. 5).

```
# Звуки
merge_sound = pygame.mixer.Sound(pygame.mixer.Sound.file if hasattr(pygame.mixer.Sound, 'file') else None)
# Звуки
try:
    merge_sound = pygame.mixer.Sound('merge.wav')
except pygame.error:
    merge_sound = None

try:
    add_sound = pygame.mixer.Sound('add.wav')
except pygame.error:
    add_sound = None

# Путь для таблицы рекордов
SCORE_FILE = "highscore.txt"
```

Рисунок 5 – Код звуковых эффектов и рекорда

В этой части объявляются переменные для звуков. Также задается имя файла для сохранения рекорда.

6. Инициализация данных игрового поля и счета (рис. 6).

```
# Глобальные переменные
GRID = [[0] * GRID_SIZE for _ in range(GRID_SIZE)]
score = 0
highscore = 0
```

Рисунок 6 – Код инициализации данных игрового поля и счета

Создаётся двумерный список (матрица) для хранения значений плиток — изначально всё заполнено нулями. Обнуляются переменные для текущего счёта и рекорда.

7. Загрузка и сохранение рекорда (рис. 7).

```
def load_highscore():
    global highscore
    if os.path.exists(SCORE_FILE):
        with open(SCORE_FILE, "r") as f:
            try:
                highscore = int(f.read())
            except:
                highscore = 0

def save_highscore():
    global highscore
    with open(SCORE_FILE, "w") as f:
        f.write(str(highscore))
```

Рисунок 7 – Код загрузки и сохранения кода

Функции для чтения рекорда из файла и записи рекорда в файл, чтобы рекорд сохранялся между сессиями игры.

8. Добавление новой плитки на поле (рис. 8).

```
def add_tile():
    empty = [(r, c) for r in range(GRID_SIZE) for c in range(GRID_SIZE) if GRID[r][c] == 0]
    if empty:
        r, c = random.choice(empty)
        GRID[r][c] = 2 if random.random() < 0.9 else 4
        if add_sound:
            add_sound.play()
```

Рисунок 8 – Код добавления новой плитки на поле

Находим случайную пустую клетку и ставим туда плитку со значением 2 или 4. Если подключены звуки, проигрываем звук добавления.

9. Отрисовка игрового поля (рис. 9).

```
def draw_grid():
    SCREEN.fill(BACKGROUND_COLOR)
    for r in range(GRID_SIZE):
        for c in range(GRID_SIZE):
            value = GRID[r][c]
            color = TILE_COLORS.get(value, (60, 58, 50))
            rect = pygame.Rect(c * TILE_SIZE + 5, r * TILE_SIZE + 160, TILE_SIZE - 10, TILE_SIZE - 10)
            pygame.draw.rect(SCREEN, color, rect, border_radius=8)
            if value:
                text_surface = FONT.render(str(value), True, (119, 110, 101) if value <= 4 else (255, 255, 255))
                text_rect = text_surface.get_rect(center=rect.center)
                SCREEN.blit(text_surface, text_rect)
```

Рисунок 9 – Код отрисовки игрового поля

Очищаем экран и отрисовываем каждую плитку с соответствующим цветом и числом в центре.

10. Логика сдвига и объединения плиток (рис.10).

```
def slide(row):
    global score
    new_row = [i for i in row if i != 0]
    i = 0
    while i < len(new_row) - 1:
        if new_row[i] == new_row[i + 1]:
            new_row[i] *= 2
            score += new_row[i]
            if merge_sound:
                merge_sound.play()
            new_row.pop(i + 1)
            new_row.append(0)
            i += 1
        i += 1
    new_row = [i for i in new_row if i != 0]
    return new_row + [0] * (GRID_SIZE - len(new_row))
```

Рисунок 10. Код логики сдвига и объединения плиток

Удаляем нули из строки, объединяет равные соседние плитки (с увеличением значения и прибавкой к счёту), затем добавляем нули в конец для сохранения длины.

11. Движение плиток в разные стороны (рис. 11).

```
def move_left():
    global GRID
    GRID = [slide(row) for row in GRID]

def move_right():
    global GRID
    GRID = [slide(row[::-1])[:-1] for row in GRID]

def move_up():
    global GRID
    GRID = list(map(list, zip(*GRID)))
    GRID = [slide(row) for row in GRID]
    GRID = list(map(list, zip(*GRID)))

def move_down():
    global GRID
    GRID = list(map(list, zip(*GRID)))
    GRID = [slide(row[::-1])[:-1] for row in GRID]
    GRID = list(map(list, zip(*GRID)))
```

Рисунок 11 – Код движения плиток в разные стороны

Используем функцию `slide` для сдвига строк, а для движения вверх и вниз — транспонируем поле, чтобы повторно использовать логику сдвига.

12. Проверка, возможны ли ходы (рис. 12).

```
def can_move():
    for r in range(GRID_SIZE):
        for c in range(GRID_SIZE):
            if GRID[r][c] == 0:
                return True
            if c < GRID_SIZE - 1 and GRID[r][c] == GRID[r][c + 1]:
                return True
            if r < GRID_SIZE - 1 and GRID[r][c] == GRID[r + 1][c]:
                return True
    return False
```

Рисунок 12 – Код проверки хода

Проверяем наличие пустых клеток или соседних одинаковых плиток. Если ходов нет, игра заканчивается.

13. Отрисовка заголовка и счета (рис. 13).

```
def draw_header():
    title = TITLE_FONT.render("2048", True, (119, 110, 101))
    SCREEN.blit(title, (20, 20))
    score_text = SCORE_FONT.render(f"Счёт: {score}", True, (119, 110, 101))
    SCREEN.blit(score_text, (20, 90))
    highscore_text = SCORE_FONT.render(f"Рекорд: {highscore}", True, (119, 110, 101))
    SCREEN.blit(highscore_text, (300, 90))
```

Рисунок 13 – Код заголовка и счета

Рисуем в верхней части экрана название игры, текущий счет и рекорд с использованием заданных шрифтов и цветов.

14. Основной игровой цикл (рис. 14).

```
def main():
    global score, highscore

    load_highscore()

    clock = pygame.time.Clock()
    add_tile()
    add_tile()

    while True:
        draw_grid()
        draw_header()
        pygame.display.flip()

        if not can_move():
            if score > highscore:
                highscore = score
                save_highscore()
            print("Game Over")
            pygame.time.wait(2000)
            pygame.quit()
            sys.exit()

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                if score > highscore:
                    highscore = score
                    save_highscore()
                pygame.quit()
                sys.exit()
            elif event.type == pygame.KEYDOWN:
                before = [row[:] for row in GRID]
                if event.key == pygame.K_LEFT:
                    move_left()
                elif event.key == pygame.K_RIGHT:
                    move_right()
                elif event.key == pygame.K_UP:
                    move_up()
                elif event.key == pygame.K_DOWN:
                    move_down()
                if GRID != before:
                    add_tile()
                    if score > highscore:
                        highscore = score

        clock.tick(60)
```

Рисунок 14 – Код основного цикла игры

Запускаем игру, обрабатываем события (нажатия клавиш, закрытие окна), обновляем поле, добавляем новые плитки и проверяем условия окончания игры.

15. Точка входа (рис.15).

```
if __name__ == "__main__":  
    main()
```

Рисунок 15. Код точки входа

Запускает игру.

16. Внешний вид приложения (рис. 16).

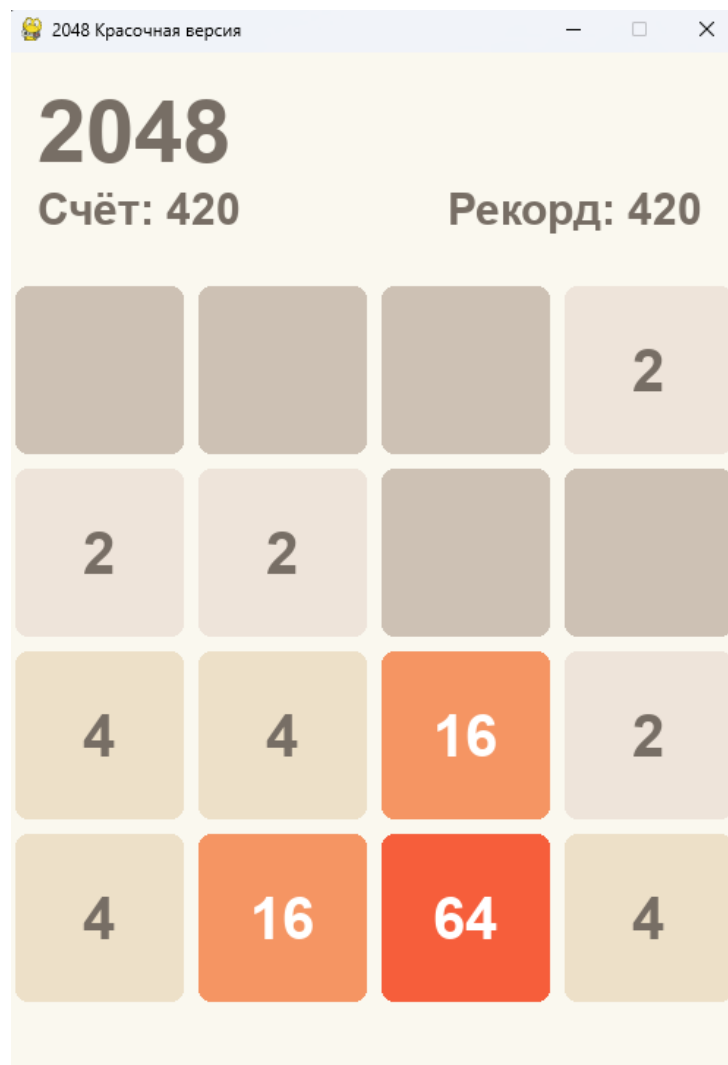


Рисунок 16. Внешний вид приложения

4 Выводы

Разработанный проект демонстрирует возможности использования Python и Pygame для создания интерактивных и визуально привлекательных

игр с сохранением пользовательских данных и звуковым сопровождением. Полученный опыт показывает, что простая логическая игра может быть успешно реализована с применением базовых графических и звуковых методов, что делает данный подход полезным для обучения и дальнейших исследований в области геймдевелопмента и интерактивного программирования.

Библиографический список

1. Новиков Е.О. Разработка программы «Игра морской бой» на языке Python // Прогрессивные технологии и экономика в машиностроении: сборник трудов XIV Всероссийской научно-практической конференции для студентов и учащейся молодежи. 2023. С. 207-210
2. Шамышев А.А. Использование графической библиотеки pygame для обучения программированию // Молодежная наука в развитии регионов. 2017. Т. 1. С. 147-149.
3. Чирков И.А. 2D игра с применением библиотеки pygame // Elibrary. 2020.
4. Белолубский М.М., Сорочинский М.А. Изучение модуля "pygame" для создания игр на языке программирования python // Современное образование: традиции и инновации. 2020. №. 2. С. 220-222.
5. Кулинча П.В. Разработка игр с использованием библиотеки pygame на языке программирования python // Дневник науки. 2023. №. 6(78).