МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных технологий и робототехники

Кафедра программного обеспечения информационных систем и технологий

**КУРСОВАЯ РАБОТА**

по дисциплине: “Языки программирования”

на тему: ««*Fynny FROG» («Игра формата три в ряд»)»*

Выполнил**:** ст. гр. 10702221 Парфенова У.Д.

Приняла**:** ст**.** преподавательМисякова В.А.

Минск 2022

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных технологий и робототехники

Кафедра программного обеспечения информационных систем и технологий

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовой работе**

по дисциплине **«Языки программирования»**

на тему **«***«Fynny FROG» («Игра формата три в ряд»)»*

Выполнил**:** ст. гр. 10702221 Парфенова У.Д.

Приняла**:** ст**.** преподавательМисякова В.А.

Минск 2022

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc120630762)

[1. Сценарий игры и постановка задачи 7](#_Toc120630763)

[2. Обоснование используемых языков и технологий 8](#_Toc120630764)

[3. Моделирование и проектирование программного обеспечения 9](#_Toc120630765)

[4. Реализация программного обеспечения 12](#_Toc120630766)

[ЗАКЛЮЧЕНИЕ 14](#_Toc120630767)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ И ИСТОЧНИКОВ 15](#_Toc120630768)

[ПРИЛОЖЕНИЕ А 16](#_Toc120630769)

# **ВВЕДЕНИЕ**

Python — это язык программирования, который широко используется в интернет-приложениях, разработке программного обеспечения, науке о данных и машинном обучении (ML). Разработчики используют Python, потому что он эффективен, прост в изучении и работает на разных платформах. Программы на языке Python можно скачать бесплатно, они совместимы со всеми типами систем и повышают скорость разработки. Это отличный выбор языка как для новичков, так и для опытных программистов.

Python был разработан в 80-ых годах и его автор Guido van Rossum, передал его в некоммерческую организацию Python Software Foundation, которая является администратором данного языка.

Сам язык тоже распространяется с открытым исходным кодом и его можно без ограничений использовать для любых целей. Язык является интерпретируемым объектно-ориентированным, но также и процедурным, функциональным.

Только лишь благодаря его гибкости он является одним из самых широко известных высокоуровневых языков программирования в наше время.

Качественные фреймворки, огромное количество учебных материалов, множество разнообразных и функциональных библиотек, понятный интерфейс, простота кода: все это делает Python действительно конкурентоспособным языком, который вряд ли сдаст свои позиции в дальнейшем.

Pygame – одна из самых известных и интересных библиотек языка программирования Python, предназначенная для написания компьютерных игр и мультимедиа-приложений.

Что касается игр, то для них Python - это совершенно подходящий язык.

Разработка игры на Python очень удобна благодаря читаемости кода, специализированной библиотеки Pygame, быстрому освоению материала.

Для создания любых приложений с графическим интерфейсом необходимо использовать событийно-ориентированное программирование **–** это способ построения компьютерной программы, при котором в [коде](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_(%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80)) (в головной функции программы) явным образом выделяется главный цикл приложения, тело которого состоит из двух частей: выборки события и обработки события.

По производительности интерпретируемый Python похож на все остальные подобные языки, но возможность компиляции в байт-код позволяет добиться большей производительности.

Существует несколько реализаций языка — CPython (основная), Jython, PyS60, IronPython, Stackless, Python for .NET, PyPy, python-safethread, Unladen Swallow, tinypy.

Выбор языка программирования выпал на Python по нескольким закономерным причинам. Во-первых, написание программы на Python максимально удобно. Хорошая читаемость кода и отличная база в виде библиотеки Pygame, позволила с минимальными трудностями и максимальным удовольствием написать программу за относительно короткий период времени. Во-вторых, сыграл роль производительный фактор. При должном умении разработчика, на выходе можно получить быстродейственный, конкурентноспособный продукт. В-третьих кроссплатформенность.

Python подходит и для веб-разработки, но с прикладной точки зрения обычно это не лучший выбор. Но в сферах анализа данных и машинного обучения Python несомненно сейчас вне конкуренции.

Основой написанной компьютерной игры выступает очень полезная и обширная библиотека для написания игра на Python – Pygame.

Библиотека Pygame является бесплатной. Выпущенная под лицензией LGPL, она позволяет создавать игры с открытым исходным кодом, freeware, shareware и коммерческие игры.

Одна из библиотек предоставляющих доступ к [API](https://ru.wikipedia.org/wiki/API) SDL. В тоже время дает возможность написания более высокоуровневого кода.

Pygame-приложения могут работать под [Android](https://ru.wikipedia.org/wiki/Android" \o "Android) на телефонах и планшетах с использованием подмножества Pygame для Android. На этой платформе поддерживаются звук, вибрация, клавиатура, акселерометр.

Для основных функций используется оптимизированный код на языках C и Assembler. Си-код часто в 10-20 раз быстрее, чем код python, а ассемблерный код может быть в 100 и более раз быстрее, чем код python.

В курсовой работе на этой библиотеке основан весь написанный код. Используется множество функций, встроенных в эту библиотеку (от обработки событий и нажатия клавиш до отрисовки игрового поля). Удобство написания, незамысловатый, простой в понимании синтаксис и неизмеримая по сравнению с другими библиотеками мощность обусловили выбор этой библиотеки.

Также была использована библиотека random. Это полезная библиотека, позволяющая генерировать случайные значения, которые придают игре щепотку вариативности и элементы неожиданности.

# **Сценарий игры и постановка задачи**

Как и все игры, моя игра была написана в несколько этапов:

1. Подключение и установка всех необходимых библиотек
2. Создание диалогового окна
3. Создание таблицы и заполнение её объектами
4. Написание основной логики игры (перемещение, удаление, добавление элементов)
5. Вывод очков, набранных за игру
6. Добавление музыки и звуковых эффектов для улучшения игры

После определения и структурирования поставленной задачи, было необходимо реализовать все пункты.

После запуска игры, вы увидите две кнопки "Start" и "Close" соответственно. При нажатии кнопки "Snart" перед вами появится клетчатое поле 8 на 8, в которой будет находится 7 разных элементов. Ваша задача собрать 3 и более элементов в ряд или в столбец, за это вы получаете очки. Чем больше элементов в ряду, тем больше очков вы получите. Очки убывают с течением времени. Это необходимо для того, чтобы вы пытались быстрее найти соответствие, а также делает игру интереснее. Игра завершается тогда, когда на поле не остаётся ходов или когда проходит 240 секунд. При нажатии на крестик при не окончившейся игре, результат засчитан не будет. В игре также есть топ 3 лучших поставленных рекорда. Файл с ними хранится в папке data с названием "three\_best\_score.txt"

# **Обоснование используемых языков и технологий**

Приложение разрабатывалось в среде PyCharm− это [интегрированная среда разработки](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8) и обучения на языке [Python](https://ru.wikipedia.org/wiki/Python). С помощью PyCharm можно выполнять обычные для интегрированной среды задачи: просматривать, редактировать, запускать, отлаживать программы на Python. [Редактор кода](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B4%D0%B0%D0%BA%D1%82%D0%BE%D1%80_%D0%BA%D0%BE%D0%B4%D0%B0) использует [подсветку синтаксиса](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B4%D1%81%D0%B2%D0%B5%D1%82%D0%BA%D0%B0_%D1%81%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%81%D0%B0).

Для реализации данного курсового проекта использовались такие программные модули, как:

* Pygame - набор модулей (библиотек) языка программирования Python предназначенный для написания компьютерных игр и мультимедиа-приложений.
* Random - предоставляет функции для генерации случайных чисел, букв, случайного выбора элементов последовательности.

Так же для удобства написания и дальнейшего улучшения, программа была разделена на несколько файлов. Такой подход удобен и даже необходим для написания любой программы или игры.

Для создания объектов использовались такие элементы как: классы, циклы, операторы условия, массивы и кортежи.

# **Моделирование и проектирование программного обеспечения**

Для реализации данного курсового проекта использовались такие программные модули, как pygame, random, time.

В большей степени в моей курсовой работе использовалась библиотека Pygame, благодаря которой и был реализован этот проект.

Библиотека random используется заполнения таблицы случайны образом, из-за чего игра становится интереснее и лучше.

Модуль Constants.py является модулем, в котором содержатся переменные, которые не изменяются в течении игры.

В модуле main.py. является основным и содержит в себе запуск, логику и работу всей игры.

Для более подробного изучения структуры программы составлена UML-диаграмма классов (см. рисунок 1).

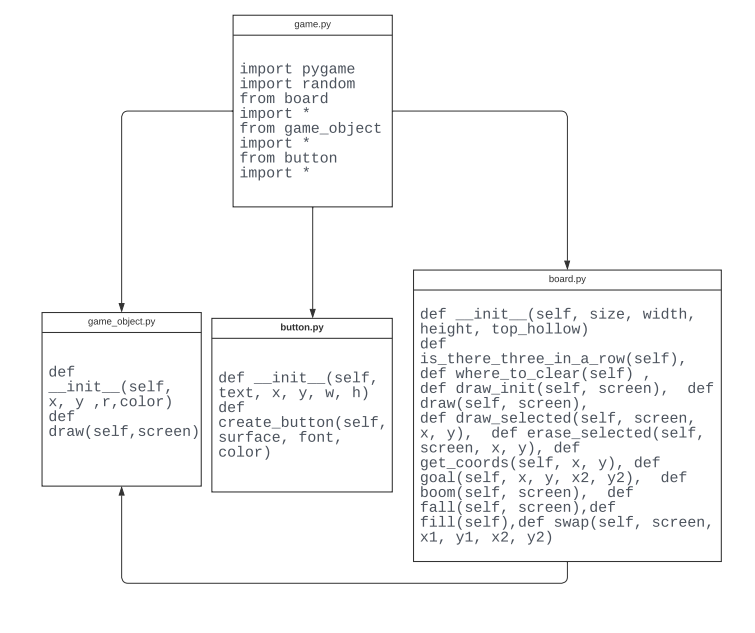


Рисунок 1. UML-диаграмма классов

# **Реализация программного обеспечения**

При запуске появляется окно, предлагающее начать игру (см. рисунок 2).

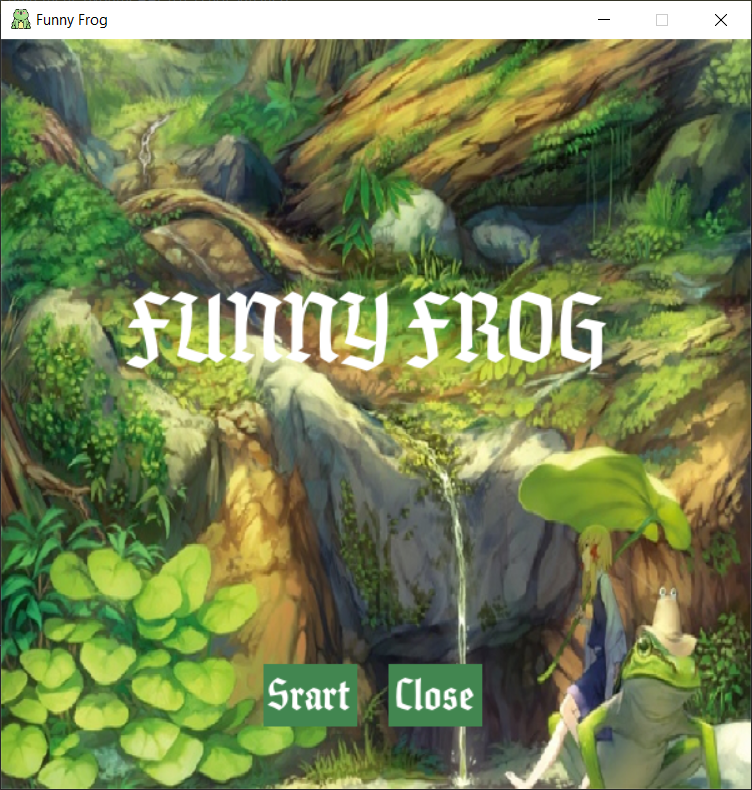


Рисунок 2. Начальное окно

Появляется окно с игрой, на дисплее игрок видит свой счет и таблицу с элементами(см. рисунок 3-4).

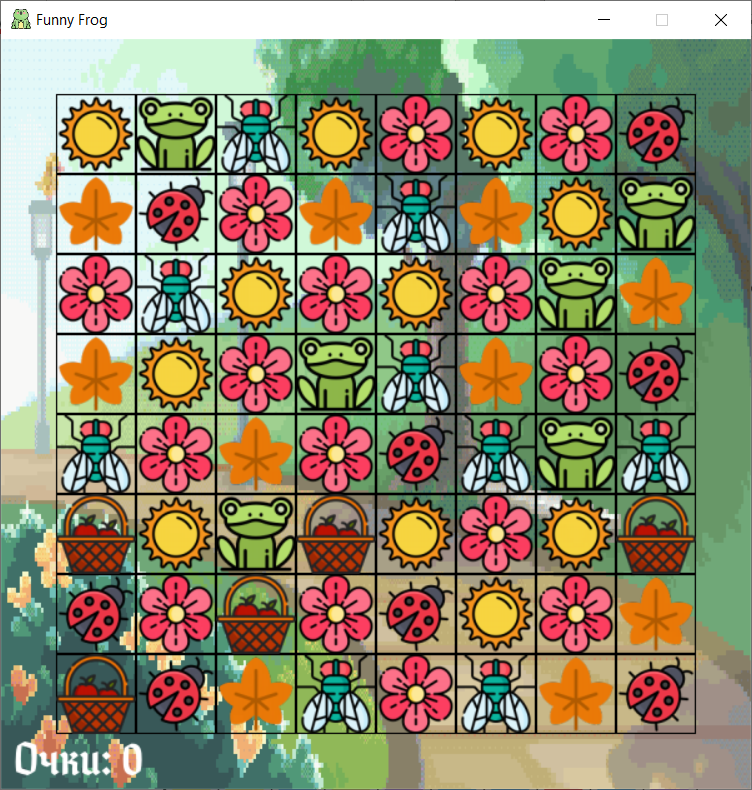


Рисунок 3. Игровое поле

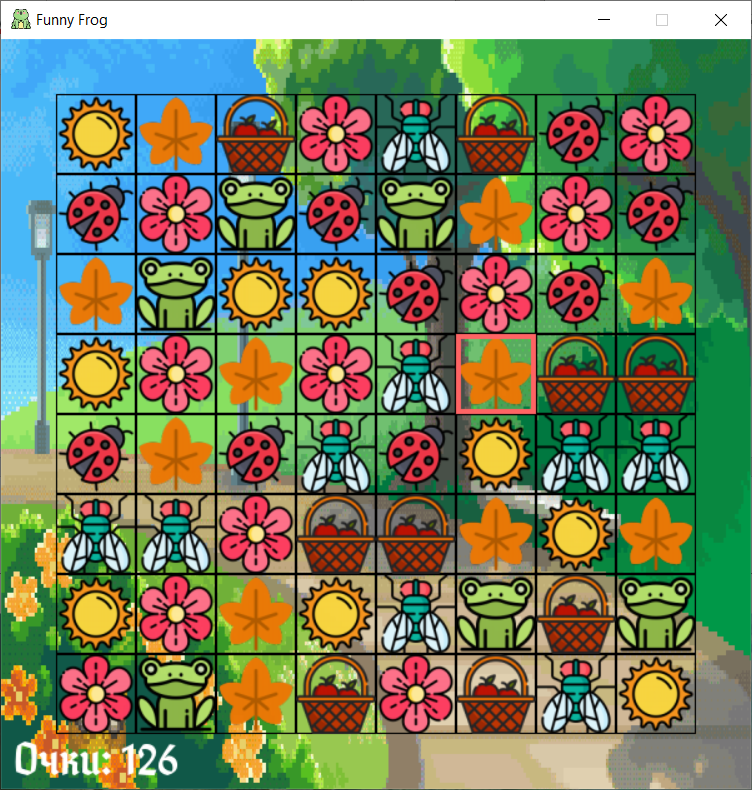


Рисунок 4. Игровое поле

При помощи мышки(тачпада) игрок перемещает элементы, складывая их в одну линию, длинной в 3 или более шариков, по горизонтали или вертикали и получает за это очки. Чем больше элементов в линии получилось, тем больше очков получает игрок (см. рисунок 5).



Рисунок 5. Игровое поле во время игры

После того как количество ходов закончилось или прошло 240 секунд, на экране появляется окно с надписью показывающая количество, набранных очков за игру. Также игрок может попробовать набрать больше очков или просто поиграть еще раз закрыв окно (см. рисунок 6).

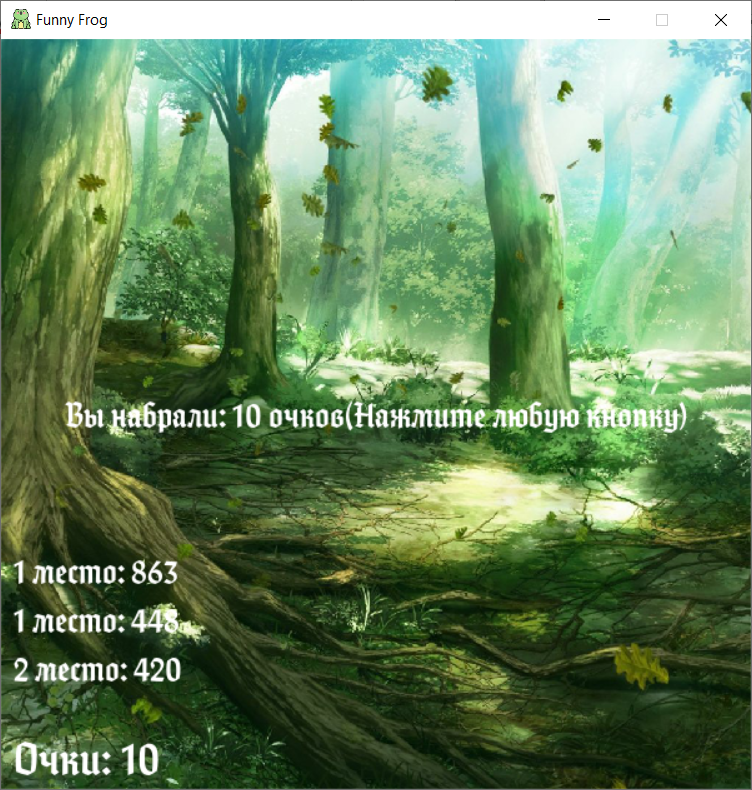


Рисунок 6. Окно игры после прохождения игры

# **ЗАКЛЮЧЕНИЕ**

В курсовом проекте написана и отлажена программа, представляющая собой игру, написанную на языке Python с использованием модуля Pygame.

В процессе работы и отладки проекта были изучены тонкости работы с библиотекой Pygame, новые методы работы, связанные с разработкой окон, шрифтов, звуков. Также изучены прорисовка объектов, создание игрового окна, подробно рассмотрены способы работы с клавиатурой и мышкой(тачпадом). Изучены методы разработки легко масштабируемого кода. Что более важно, мы попробовали себя в разработке игр и смогли протестировать свое логическое мышление. Ведь составление и написание программы очень сложное и энергозатратное занятие. Оно требует полной концентрации и умения структурировать и организовывать себя.

Игра успешно протестирована.

Данная игра была написана под влиянием такой же по смыслу игры «Homescapes».

# **СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ И ИСТОЧНИКОВ**

1. Доусон М. Программируем на Python. – СПб.: Питер, 2014. – 416 с.
2. Прохоренок Н.А. Python 3 и PyQt. Разработка приложений. – СПб.: БХВ-Петербург, 2012. – 704 с.
3. [Что такое Python? — Облачное руководство для начинающих по Python — AWS (amazon.com)](https://aws.amazon.com/ru/what-is/python/)
4. [Pygame и разработка игр. Урок 1 (younglinux.info)](https://younglinux.info/pygame/pygame)
5. [(2638) Программирование игр Pygame #34: Таблица рекордов - YouTube](https://www.youtube.com/watch?v=945FLocAkWE&list=PLjRuaCofWO0O8qv2or33DGHpQ6kN_CATM&index=35)
6. Лутц М. Программирование на Python, том I, 4-е издание. – Пер. с англ. – СПб.: Сим¬вол-Плюс, 2011. – 992 с.
7. Лутц М. Программирование на Python, том II, 4-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 992 с.
8. Лутц М. Изучаем Python, 4-е издание. – Пер. с англ. – СПб.: Символ - Плюс, 2011. – 1280 с.
9. Хахаев И.А. Практикум по алгоритмизации и программированию на Python. – М.: Альт Линукс, 2010. — 126 с. (Библиотека ALT Linux).

# **ПРИЛОЖЕНИЕ А**

**Листинг исходного кода**

**Main.py**

import os

import random

import time

import sys

import copy

import pygame

import pygame.locals as locals

from Constants import \*

timer = 0

background\_number = 0

GAMEBACKGROUND = f'game\_fon{background\_number}.gif'

def run\_game():

global timer, background\_number, GAMEBACKGROUND

"""Играем, пока не закроется окно или не останется доступных ходов"""

game\_board = get\_blank\_board()

score = 0

fill\_board\_and\_animate(game\_board, [], score)

# инициализуем переменные для новой игры

first\_selected\_pokemon = None

last\_mouse\_down\_x = None

last\_mouse\_down\_y = None

game\_is\_over = False

is\_write\_in\_file = False

string\_rendered = ''

last\_score\_deduction = time.time()

click\_continue\_text\_surf = None

count=0

while True: # главный цикл игры

# Меняем фон раз в 60 секунд

if pygame.time.get\_ticks() // 1000 - timer > 60:

timer = pygame.time.get\_ticks() // 1000

background\_number = (background\_number + 1) % 4

GAMEBACKGROUND = f'game\_fon{background\_number}.gif'

count+=1

clicked\_space = None

for event in pygame.event.get():

if event.type == locals.QUIT or (event.type == locals.KEYUP and event.key == locals.K\_ESCAPE):

main()

elif event.type == locals.KEYUP and event.key == locals.K\_BACKSPACE:

return

elif event.type == locals.MOUSEBUTTONUP:

if game\_is\_over:

main()

if event.pos == (last\_mouse\_down\_x, last\_mouse\_down\_y):

clicked\_space = check\_for\_pokemon\_click(event.pos)

else:

first\_selected\_pokemon = check\_for\_pokemon\_click((last\_mouse\_down\_x, last\_mouse\_down\_y))

clicked\_space = check\_for\_pokemon\_click(event.pos)

if not first\_selected\_pokemon or not clicked\_space:

first\_selected\_pokemon = None

clicked\_space = None

elif event.type == locals.MOUSEBUTTONDOWN:

last\_mouse\_down\_x, last\_mouse\_down\_y = event.pos

if clicked\_space and not first\_selected\_pokemon:

first\_selected\_pokemon = clicked\_space

elif clicked\_space and first\_selected\_pokemon:

first\_swapping\_pokemon, second\_swapping\_pokemon = get\_swapping\_pokemons(game\_board, first\_selected\_pokemon,

clicked\_space)

if first\_swapping\_pokemon is None and second\_swapping\_pokemon is None:

first\_selected\_pokemon = None

continue

board\_copy = getboard\_copy\_minus\_pokemons(game\_board, (first\_swapping\_pokemon, second\_swapping\_pokemon))

animate\_moving\_pokemons(board\_copy, [first\_swapping\_pokemon, second\_swapping\_pokemon], [], score)

game\_board[first\_swapping\_pokemon['x']][first\_swapping\_pokemon['y']] = second\_swapping\_pokemon['imageNum']

game\_board[second\_swapping\_pokemon['x']][second\_swapping\_pokemon['y']] = first\_swapping\_pokemon['imageNum']

matched\_pokemons = find\_matching\_pokemons(game\_board)

if not matched\_pokemons:

GAME\_SOUNDS['bad swap'].play()

animate\_moving\_pokemons(board\_copy, [first\_swapping\_pokemon, second\_swapping\_pokemon], [], score)

game\_board[first\_swapping\_pokemon['x']][first\_swapping\_pokemon['y']] = \

first\_swapping\_pokemon['imageNum']

game\_board[second\_swapping\_pokemon['x']][second\_swapping\_pokemon['y']] = \

second\_swapping\_pokemon['imageNum']

else:

score\_add = 0

while matched\_pokemons:

points = []

for pokemon\_set in matched\_pokemons:

score\_add += (10 + (len(pokemon\_set) - 3) \* 10)

for pokemon in pokemon\_set:

game\_board[pokemon[0]][pokemon[1]] = EMPTY\_SPACE

points.append({'points': score\_add,

'x': pokemon[0] \* POKEMON\_IMAGE\_SIZE + X\_MARGIN,

'y': pokemon[1] \* POKEMON\_IMAGE\_SIZE + Y\_MARGIN})

random.choice(GAME\_SOUNDS['match']).play()

score += score\_add

fill\_board\_and\_animate(game\_board, points, score)

matched\_pokemons = find\_matching\_pokemons(game\_board)

first\_selected\_pokemon = None

if (not can\_make\_move(game\_board)) :

game\_is\_over = True

background = pygame.transform.scale(load\_image(GAMEBACKGROUND), (600, 600))

SCREEN.blit(background, (0, 0))

draw\_board(game\_board)

if first\_selected\_pokemon is not None:

highlight\_space(first\_selected\_pokemon['x'], first\_selected\_pokemon['y'])

if game\_is\_over:

with open('data/three\_best\_score.txt', 'r', encoding='utf8') as file:

scores = list(map(str.strip, file.readlines()))

if click\_continue\_text\_surf is None:

font = pygame.font.Font('data/Inkulinati-Regular.ttf', 24)

GAME\_SOUNDS['complete'].play()

string\_points = f'Вы набрали: {score} очков(Нажмите любую кнопку)'

click\_continue\_text\_surf = font.render(string\_points, True, GAMEOVER\_COLOR)

click\_continue\_text\_rect = click\_continue\_text\_surf.get\_rect()

click\_continue\_text\_rect.center = int(WINDOW\_WIDTH / 2), int(WINDOW\_HEIGHT / 2)

background = pygame.transform.scale(load\_image(END\_BACKGROUND), (600, 600))

text\_coord = 400

SCREEN.blit(background, (0, 0))

for line in scores:

string\_rendered = font.render(line, True, pygame.Color('white'))

intro\_rect = string\_rendered.get\_rect()

text\_coord += 10

intro\_rect.top = text\_coord

intro\_rect.x = 10

text\_coord += intro\_rect.height

SCREEN.blit(string\_rendered, intro\_rect)

SCREEN.blit(click\_continue\_text\_surf, click\_continue\_text\_rect)

if not is\_write\_in\_file:

is\_write\_in\_file = True

set\_best\_score\_in\_file(score)

if event.type == locals.MOUSEBUTTONDOWN:

main()

elif score > 0 and time.time() - last\_score\_deduction > DEDUCT\_SPEED:

# очки постоянно уменьшаются

score -= 1

last\_score\_deduction = time.time()

draw\_score(score)

pygame.display.update()

FPS\_CLOCK.tick(FPS)

def set\_best\_score\_in\_file(score):

with open('data/three\_best\_score.txt', 'r', encoding='utf8') as read\_file:

lines = list(map(str.strip, read\_file.readlines()))

if lines:

for index, line in enumerate(lines):

if int(line.split()[-1]) < score:

lines.insert(index, f'{index + 1} место: {score}')

with open('data/three\_best\_score.txt', 'w', encoding='utf8') as write\_in\_file:

write\_in\_file.write('\n'.join(lines[:3]) + '\n')

break

elif len(lines) != 3:

with open('data/three\_best\_score.txt', 'a', encoding='utf8') as write\_in\_file:

write\_in\_file.write(f'{len(lines) + 1} место: {score}\n')

else:

with open('data/three\_best\_score.txt', 'w', encoding='utf8') as write\_in\_file:

write\_in\_file.write(f'1 место: {score}\n')

def get\_swapping\_pokemons(board, first\_xy, second\_xy):

first\_pokemon = {'imageNum': board[first\_xy['x']][first\_xy['y']],

'x': first\_xy['x'],

'y': first\_xy['y']}

second\_pokemon = {'imageNum': board[second\_xy['x']][second\_xy['y']],

'x': second\_xy['x'],

'y': second\_xy['y']}

if first\_pokemon['x'] == second\_pokemon['x'] + 1 and first\_pokemon['y'] == second\_pokemon['y']:

first\_pokemon['direction'] = LEFT

second\_pokemon['direction'] = RIGHT

elif first\_pokemon['x'] == second\_pokemon['x'] - 1 and first\_pokemon['y'] == second\_pokemon['y']:

first\_pokemon['direction'] = RIGHT

second\_pokemon['direction'] = LEFT

elif first\_pokemon['y'] == second\_pokemon['y'] + 1 and first\_pokemon['x'] == second\_pokemon['x']:

first\_pokemon['direction'] = UP

second\_pokemon['direction'] = DOWN

elif first\_pokemon['y'] == second\_pokemon['y'] - 1 and first\_pokemon['x'] == second\_pokemon['x']:

first\_pokemon['direction'] = DOWN

second\_pokemon['direction'] = UP

else:

# Не смогли свапнуть

return None, None

return first\_pokemon, second\_pokemon

def get\_blank\_board():

board = []

for x in range(BOARD\_WIDTH):

board.append([EMPTY\_SPACE] \* BOARD\_HEIGHT)

return board

def can\_make\_move(board):

one\_off\_patterns = (((0, 1), (1, 0), (2, 0)),

((0, 1), (1, 1), (2, 0)),

((0, 0), (1, 1), (2, 0)),

((0, 1), (1, 0), (2, 1)),

((0, 0), (1, 0), (2, 1)),

((0, 0), (1, 1), (2, 1)),

((0, 0), (0, 2), (0, 3)),

((0, 0), (0, 1), (0, 3)))

for x in range(BOARD\_WIDTH):

for y in range(BOARD\_HEIGHT):

for pat in one\_off\_patterns:

# проверяем, может ли мы куда-то походить

if (get\_pokemon\_at(board, x + pat[0][0], y + pat[0][1]) ==

get\_pokemon\_at(board, x + pat[1][0], y + pat[1][1]) ==

get\_pokemon\_at(board, x + pat[2][0], y + pat[2][1]) is not None) or \

(get\_pokemon\_at(board, x + pat[0][1], y + pat[0][0]) ==

get\_pokemon\_at(board, x + pat[1][1], y + pat[1][0]) ==

get\_pokemon\_at(board, x + pat[2][1], y + pat[2][0]) is not None):

return True

return False

def draw\_moving\_pokemon(pokemon, progress):

"""Осуществляет анимацию движения кристаллов"""

move\_x = 0

move\_y = 0

progress \*= 0.01

if pokemon['direction'] == UP:

move\_y = -int(progress \* POKEMON\_IMAGE\_SIZE)

elif pokemon['direction'] == DOWN:

move\_y = int(progress \* POKEMON\_IMAGE\_SIZE)

elif pokemon['direction'] == RIGHT:

move\_x = int(progress \* POKEMON\_IMAGE\_SIZE)

elif pokemon['direction'] == LEFT:

move\_x = -int(progress \* POKEMON\_IMAGE\_SIZE)

base\_x = pokemon['x']

base\_y = pokemon['y']

if base\_y == ROW\_ABOVE\_BOARD:

base\_y = -1

pixel\_x = X\_MARGIN + (base\_x \* POKEMON\_IMAGE\_SIZE)

pixel\_y = Y\_MARGIN + (base\_y \* POKEMON\_IMAGE\_SIZE)

r = pygame.Rect((pixel\_x + move\_x, pixel\_y + move\_y, POKEMON\_IMAGE\_SIZE, POKEMON\_IMAGE\_SIZE))

SCREEN.blit(POKEMON\_IMAGES[pokemon['imageNum']], r)

def pull\_down\_all\_pokemons(board):

"""Заполняет доску, чтобы заполнить пробелы"""

for x in range(BOARD\_WIDTH):

pokemons\_in\_column = []

for y in range(BOARD\_HEIGHT):

if board[x][y] != EMPTY\_SPACE:

pokemons\_in\_column.append(board[x][y])

board[x] = ([EMPTY\_SPACE] \* (BOARD\_HEIGHT - len(pokemons\_in\_column))) + pokemons\_in\_column

def get\_pokemon\_at(board, x, y):

if x < 0 or y < 0 or x >= BOARD\_WIDTH or y >= BOARD\_HEIGHT:

return None

else:

return board[x][y]

def get\_drop\_slots(board):

board\_copy = copy.deepcopy(board)

pull\_down\_all\_pokemons(board\_copy)

drop\_slots = []

for i in range(BOARD\_WIDTH):

drop\_slots.append([])

for x in range(BOARD\_WIDTH):

for y in range(BOARD\_HEIGHT - 1, -1, -1):

if board\_copy[x][y] == EMPTY\_SPACE:

possible\_pokemons = list(range(len(POKEMON\_IMAGES)))

for offsetX, offsetY in ((0, -1), (1, 0), (0, 1), (-1, 0)):

neighbor\_pokemon = get\_pokemon\_at(board\_copy, x + offsetX, y + offsetY)

if neighbor\_pokemon is not None and neighbor\_pokemon in possible\_pokemons:

possible\_pokemons.remove(neighbor\_pokemon)

new\_pokemon = random.choice(possible\_pokemons)

board\_copy[x][y] = new\_pokemon

drop\_slots[x].append(new\_pokemon)

return drop\_slots

def find\_matching\_pokemons(board):

pokemons\_to\_remove = []

board\_copy = copy.deepcopy(board)

for x in range(BOARD\_WIDTH):

for y in range(BOARD\_HEIGHT):

if get\_pokemon\_at(board\_copy, x, y) == get\_pokemon\_at(board\_copy, x + 1, y) == \

get\_pokemon\_at(board\_copy, x + 2, y) and get\_pokemon\_at(board\_copy, x, y) != EMPTY\_SPACE:

target\_pokemon = board\_copy[x][y]

offset = 0

remove\_set = []

while get\_pokemon\_at(board\_copy, x + offset, y) == target\_pokemon:

remove\_set.append((x + offset, y))

board\_copy[x + offset][y] = EMPTY\_SPACE

offset += 1

pokemons\_to\_remove.append(remove\_set)

if get\_pokemon\_at(board\_copy, x, y) == get\_pokemon\_at(board\_copy, x, y + 1) == \

get\_pokemon\_at(board\_copy, x, y + 2) and get\_pokemon\_at(board\_copy, x, y) != EMPTY\_SPACE:

target\_pokemon = board\_copy[x][y]

offset = 0

remove\_set = []

while get\_pokemon\_at(board\_copy, x, y + offset) == target\_pokemon:

remove\_set.append((x, y + offset))

board\_copy[x][y + offset] = EMPTY\_SPACE

offset += 1

pokemons\_to\_remove.append(remove\_set)

return pokemons\_to\_remove

def highlight\_space(x, y):

pygame.draw.rect(SCREEN, HIGHLIGHT\_COLOR, BOARD\_RECTS[x][y], 4)

def get\_dropping\_pokemons(board):

board\_copy = copy.deepcopy(board)

dropping\_pokemons = []

for x in range(BOARD\_WIDTH):

for y in range(BOARD\_HEIGHT - 2, -1, -1):

if board\_copy[x][y + 1] == EMPTY\_SPACE and board\_copy[x][y] != EMPTY\_SPACE:

dropping\_pokemons.append({'imageNum': board\_copy[x][y], 'x': x, 'y': y, 'direction': DOWN})

board\_copy[x][y] = EMPTY\_SPACE

return dropping\_pokemons

def animate\_moving\_pokemons(board, pokemons, points\_text, score):

progress = 0

while progress < 100:

background = pygame.transform.scale(load\_image(GAMEBACKGROUND), (600, 600))

SCREEN.blit(background, (0, 0))

draw\_board(board)

for pokemon in pokemons:

draw\_moving\_pokemon(pokemon, progress)

draw\_score(score)

for point\_text in points\_text:

points\_surf = BASIC\_FONT.render(str(point\_text['points']), True, SCORE\_COLOR)

points\_rect = points\_surf.get\_rect()

points\_rect.center = (point\_text['x'], point\_text['y'])

SCREEN.blit(points\_surf, points\_rect)

pygame.display.update()

FPS\_CLOCK.tick(FPS)

progress += MOVE\_RATE

def move\_pokemons(board, moving\_pokemons):

for pokemon in moving\_pokemons:

if pokemon['y'] != ROW\_ABOVE\_BOARD:

board[pokemon['x']][pokemon['y']] = EMPTY\_SPACE

move\_x = 0

move\_y = 0

if pokemon['direction'] == LEFT:

move\_x = -1

elif pokemon['direction'] == RIGHT:

move\_x = 1

elif pokemon['direction'] == DOWN:

move\_y = 1

elif pokemon['direction'] == UP:

move\_y = -1

board[pokemon['x'] + move\_x][pokemon['y'] + move\_y] = pokemon['imageNum']

else:

board[pokemon['x']][0] = pokemon['imageNum']

def fill\_board\_and\_animate(board, points, score):

drop\_slots = get\_drop\_slots(board)

while drop\_slots != [[]] \* BOARD\_WIDTH:

moving\_pokemons = get\_dropping\_pokemons(board)

for x in range(len(drop\_slots)):

if len(drop\_slots[x]) != 0:

moving\_pokemons.append({'imageNum': drop\_slots[x][0], 'x': x, 'y': ROW\_ABOVE\_BOARD, 'direction': DOWN})

board\_copy = getboard\_copy\_minus\_pokemons(board, moving\_pokemons)

animate\_moving\_pokemons(board\_copy, moving\_pokemons, points, score)

move\_pokemons(board, moving\_pokemons)

for x in range(len(drop\_slots)):

if len(drop\_slots[x]) == 0:

continue

board[x][0] = drop\_slots[x][0]

del drop\_slots[x][0]

def check\_for\_pokemon\_click(pos):

"""Проверяем, где кликнул пользователь (в поле или нет)"""

if pos and len(pos) == 2 and pos[0] and pos[1]:

for x in range(BOARD\_WIDTH):

for y in range(BOARD\_HEIGHT):

if BOARD\_RECTS[x][y].collidepoint(pos[0], pos[1]):

return {'x': x, 'y': y}

return None

def draw\_board(board):

for x in range(BOARD\_WIDTH):

for y in range(BOARD\_HEIGHT):

pygame.draw.rect(SCREEN, GRID\_COLOR, BOARD\_RECTS[x][y], 1)

pokemon\_to\_draw = board[x][y]

if pokemon\_to\_draw != EMPTY\_SPACE:

SCREEN.blit(POKEMON\_IMAGES[pokemon\_to\_draw], BOARD\_RECTS[x][y])

def getboard\_copy\_minus\_pokemons(board, pokemons):

board\_copy = copy.deepcopy(board)

for pokemon in pokemons:

if pokemon['y'] != ROW\_ABOVE\_BOARD:

board\_copy[pokemon['x']][pokemon['y']] = EMPTY\_SPACE

return board\_copy

def draw\_score(score):

score\_img = BASIC\_FONT.render(f'Очки: {score}', True, SCORE\_COLOR)

score\_rect = score\_img.get\_rect()

score\_rect.bottomleft = (10, WINDOW\_HEIGHT - 6)

SCREEN.blit(score\_img, score\_rect)

def terminate():

pygame.quit()

sys.exit()

def start\_screen():

FPS\_CLOCK = pygame.time.Clock()

SCREEN = pygame.display.set\_mode((WINDOW\_WIDTH, WINDOW\_HEIGHT))

intro\_text = ["FUNNY FROG"]

background = pygame.transform.scale(load\_image(START\_BACKGROUND), (600, 600))

SCREEN.blit(background, (0, 0))

font = pygame.font.Font('data/Inkulinati-Regular.ttf', 30)

font1 = pygame.font.Font('data/Inkulinati-Regular.ttf', 65)

text\_coord = 50

button2 = pygame.Rect(310, 500, 75, 50)

button = pygame.Rect(210, 500, 75, 50)

pygame.draw.rect(SCREEN, [66, 134, 80], button)

pygame.draw.rect(SCREEN, [66, 134, 80], button2)

start\_string = font.render('Srart', True, pygame.Color('white'))

rect\_start\_string = start\_string.get\_rect()

rect\_start\_string.x += 213

rect\_start\_string.y += 505

SCREEN.blit(start\_string, rect\_start\_string)

exit\_string = font.render('Close', True, pygame.Color('white'))

rect\_exit\_string = start\_string.get\_rect()

rect\_exit\_string.x += 315

rect\_exit\_string.y += 505

SCREEN.blit(exit\_string, rect\_exit\_string)

for line in intro\_text:

string\_rendered = font1.render(line, True, pygame.Color('white'))

intro\_rect = string\_rendered.get\_rect()

text\_coord += 140

intro\_rect.top = text\_coord

intro\_rect.x = 100

text\_coord += intro\_rect.height

SCREEN.blit(string\_rendered, intro\_rect)

while True:

for event in pygame.event.get():

if event.type == pygame.MOUSEBUTTONDOWN:

if button.collidepoint(event.pos):

return # начинаем игру

if event.type == pygame.MOUSEBUTTONDOWN:

if button2.collidepoint(event.pos):

terminate()

if event.type == pygame.QUIT:

terminate()

pygame.display.flip()

FPS\_CLOCK.tick(FPS)

def load\_image(name):

filename = 'data/Backgrounds/' + name

if not os.path.isfile(filename):

print(f"Файл с изображением '{filename}' не найден")

sys.exit()

image = pygame.image.load(filename)

return image

def main():

global FPS\_CLOCK, SCREEN, POKEMON\_IMAGES, GAME\_SOUNDS, BASIC\_FONT, BOARD\_RECTS

pygame.init()

FPS\_CLOCK = pygame.time.Clock()

SCREEN = pygame.display.set\_mode((WINDOW\_WIDTH, WINDOW\_HEIGHT))

pygame.display.set\_caption('Funny Frog')

pygame.display.set\_icon(pygame.image.load("data/Sprites/frog.bmp"))

BASIC\_FONT = pygame.font.Font('data/Inkulinati-Regular.ttf', 32)

# Подгружаем изображения

POKEMON\_IMAGES = []

for i in range(1, NUM\_POKEMON\_IMAGES + 1):

pokemon\_image = pygame.image.load(f'data/Sprites/pokemon{i}.png')

if pokemon\_image.get\_size() != (POKEMON\_IMAGE\_SIZE, POKEMON\_IMAGE\_SIZE):

pokemon\_image = pygame.transform.smoothscale(pokemon\_image, (POKEMON\_IMAGE\_SIZE, POKEMON\_IMAGE\_SIZE))

POKEMON\_IMAGES.append(pokemon\_image)

# Подгружаем звуки

GAME\_SOUNDS = {}

GAME\_SOUNDS['bad swap'] = pygame.mixer.Sound('data/Music/badswap.mp3')

GAME\_SOUNDS['match'] = []

GAME\_SOUNDS['background'] = []

for i in range(NUM\_MATCH\_SOUNDS):

GAME\_SOUNDS['match'].append(pygame.mixer.Sound(f'data/Music/match{i}.mp3'))

for i in range(NUM\_BACKGROUND\_SONGS):

GAME\_SOUNDS['background'].append(pygame.mixer.Sound(f'data/Music/background{i}.mp3'))

GAME\_SOUNDS['complete'] = pygame.mixer.Sound('data/Music/complete.mp3')

BOARD\_RECTS = []

for x in range(BOARD\_WIDTH):

BOARD\_RECTS.append([])

for y in range(BOARD\_HEIGHT):

r = pygame.Rect((X\_MARGIN + (x \* POKEMON\_IMAGE\_SIZE),

Y\_MARGIN + (y \* POKEMON\_IMAGE\_SIZE),

POKEMON\_IMAGE\_SIZE,

POKEMON\_IMAGE\_SIZE))

BOARD\_RECTS[x].append(r)

start\_screen()

while True:

run\_game()

if \_\_name\_\_ == '\_\_main\_\_':

main()

**Constants.py**

FPS = 60

WINDOW\_WIDTH = 600

WINDOW\_HEIGHT = 600

BOARD\_WIDTH = 8

BOARD\_HEIGHT = 8

POKEMON\_IMAGE\_SIZE = 64

NUM\_POKEMON\_IMAGES = 7 # количество типов кристаллов

NUM\_MATCH\_SOUNDS = 4 # количество типов звуков

NUM\_BACKGROUND\_SONGS = 4

MOVE\_RATE = 25 # скорость анимации падения кристаллов

DEDUCT\_SPEED = 0.8

PURPLE = (255, 0, 255)

BLUE = (0, 0, 255)

RED = (255, 100, 100)

BLACK = (0, 0, 0)

WHITE = (255, 255, 255)

HIGHLIGHT\_COLOR = RED # цвет выделение выбранной ячейки

GRID\_COLOR = BLACK # цвет сетки

GAMEOVER\_COLOR = WHITE # цвет надписи "Game Over"

GAMEOVER\_BGCOLOR = BLACK # фон окончания игры

SCORE\_COLOR = (255, 255, 255) # цвет очков набранных игроком

START\_BACKGROUND = 'start\_background.jpg'

END\_BACKGROUND = 'end\_background.jpg'

# Отступы

X\_MARGIN = int((WINDOW\_WIDTH - POKEMON\_IMAGE\_SIZE \* BOARD\_WIDTH) / 2)

Y\_MARGIN = int((WINDOW\_HEIGHT - POKEMON\_IMAGE\_SIZE \* BOARD\_HEIGHT) / 2)

# Константы направлений

UP = 'up'

DOWN = 'down'

LEFT = 'left'

RIGHT = 'right'

EMPTY\_SPACE = -1

ROW\_ABOVE\_BOARD = 'row above board'