Муниципальное автономное общеобразовательное учреждение

средняя общеобразовательная школа № 94 города Тюмени

Индивидуальный проект

**Игра «Тетрис» на Python**

Автор:

Ганичев Кирилл Викторович,

10 класс

Научный руководитель:

Семёнова Татьяна Петровна

МАОУ СОШ № 94 города Тюмени

г. Тюмень

2024

**Оглавление**

1. Введение
2. 1. Технологии для решения задач
3. 2. Файловая структура проекта и реализация
4. 3. Написание кода для реализации поставленных задач
5. 4. Функции и результат
6. Заключение

**Введение**

Проблема: На компьютере не установлена по умолчанию игра «Тетрис», создадим свою. А также это будет отличный опыт и практика в программировании.

Цели проекта:

1. Реализовать корректный геймплей игры Тетрис
2. Создать простой и приятный для пользования интерфейс
3. Реализовать хранение рекорда
4. Создать структуру файлов начала, концовки игры и самого процесса

**ГЛАВА I. Технологии для решения поставленных задач**

Для реализации поставленных задач, воспользуемся таким языком программирования, как Python.

Будем использовать:

1. Стандартную библиотеку Python
2. Библиотеку для создания игр «Pygame»
3. Библиотеку для отладки «icecream»
4. Редактор кода PyCharm и систему контроля версий git

**ГЛАВА** **II . Файловая структура проекта и план реализации**

Создадим файл для стартовой анимации, анимации при окончании игры, самого геймплея и общий файл для их соединения:

* start\_screen.py
* result\_screen.py
* game.py
* main.py

Также создадим файл figures.py для функционала фигур. Добавим шрифт SAIBA-45 для анимации.

**ГЛАВА III. Написание кода для реализации задач**

В файле main.py создадим начальный код в функции main, который при запуске файла запускает функционал файла start\_screen.

Фрагмент кода из функции main:

--------------------------------------------------------

pygame.init()

size = pygame.display.get\_desktop\_sizes()[0]

surface = pygame.display.set\_mode(size)

clock = pygame.time.Clock()

running = start\_screen(surface, clock, FPS)

while running:

pass

pygame.quit()

--------------------------------------------------------

И также напишем код в самом файле start\_screen.py с классами Text для текста, StartAnim для начала анимации и EndAnim для её завершения.

Напишем код для figures.py, вот его фрагмент:

--------------------------------------------------------

figures\_pos = [[(-1, 0), (-2, 0), (0, 0), (1, 0)], # координаты 7-ми фигур

[(0, -1), (-1, -1), (-1, 0), (0, 0)],

[(-1, 0), (-1, 1), (0, 0), (0, -1)],

[(0, 0), (-1, 0), (0, 1), (-1, -1)],

[(0, 0), (0, -1), (0, 1), (-1, -1)],

[(0, 0), (0, -1), (0, 1), (1, -1)],

[(0, 0), (0, -1), (0, 1), (-1, 0)]]

figures = [[pygame.Rect(x + W // 2, y + 1, 1, 1) for x, y in figure\_pos]

for figure\_pos in figures\_pos] # расположение по центру сверху

figure\_rect = pygame.Rect(0, 0, TILE - 2, TILE - 2)

--------------------------------------------------------

Приступим к самому функционалу игры, фрагмент кода из файла game.py:

--------------------------------------------------------

grid = [pygame.Rect(x \* TILE, y \* TILE, TILE, TILE)

for x in range(10) for y in range(20)]

...

figure = deepcopy(choice(figures))

field = [[0 for i in range(W)] for j in range(H)]

score = 0

lines = 0

scores = {0: 0, 1: 100, 2: 300, 3: 700, 4: 1500} # опред. кол-во линий за раз и соотв. Награда

def check\_borders():

…

center = figure[0]

figure\_old = deepcopy(figure)

if rotate:

for i in range(4):

x = figure[i].y - center.y

y = figure[i].x - center.x

figure[i].x = center.x - x

figure[i].y = center.y + y

if not check\_borders():

figure = deepcopy(figure\_old)

…

[pygame.draw.rect(game\_sc, (40, 40, 40), i\_rect, 1) for i\_rect in grid]

--------------------------------------------------------

Напишем код для анимации экрана окончания игры в файле result\_screen.py;

Основная часть происходит в функции result\_screen (приведен фрагмент кода):

def result\_screen(surface: pygame.Surface, clock: pygame.time.Clock, fps: int, score: int, prev\_score: int) -> bool:

you\_loose\_text = Text((w // 2, h // 3), [f'Your score: {score}', f'Best score: {prev\_score}'], '#9F9F9F', w // 20)

restart\_text = Text((w // 2, h // 3 \* 2), ['Press r', 'To restart'], '#9F9F9F', w // 20, 5)

--------------------------------------------------------

И напоследок добавим в main.py функционал сохранения счета:

def create\_scorefile(file\_name: str):

open(fr'{os.path.curdir}\{file\_name}', 'wt', encoding='utf-8').write('0')

return open\_scorefile(file\_name)

def open\_scorefile(file\_name: str):

return open(fr'{os.path.curdir}\{file\_name}', 'rt', encoding='utf-8')\

if os.path.exists(fr'{os.path.curdir}\{file\_name}') else create\_scorefile(file\_name)

def write\_scorefile(file\_name: str, result: int) -> None:

if not os.path.exists(fr'{os.path.curdir}\{file\_name}'):

create\_scorefile(file\_name)

open(fr'{os.path.curdir}\{file\_name}', 'wt', encoding='utf-8').write(str(result))

И свяжем же наши файлы между собой, чтобы они запускались последовательно (фрагмент функции main в файле main.py):

running = start\_screen(surface, clock, FPS)

while running:

result = game(clock, FPS)

if result == -1:

break

running = result\_screen(surface, clock, FPS, result, prev\_result)

if result > prev\_result:

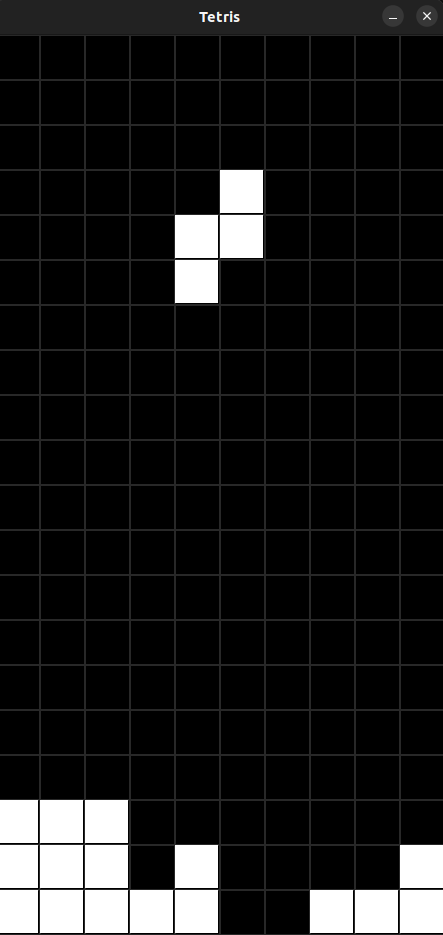
prev\_result = ic(result)

write\_scorefile('scores', result)

**ГЛАВА** **IV . Функции и результат**

Посмотрим что получилось:



**Заключение**

Объём программного кода составил: main.py (51 строка) + game.py (137 строк) + figures.py (17 строк) + start\_screen.py (148 строк) + result\_screen.py (48 строк) + scores (1 строка) = 402 строки.

Итог: все поставленные задачи проекта были выполнены.

Возможности для доработки: добавление дополнительных функциональностей на экран самой игры, например начисление счета на самом экране. Раскраска фигур для красоты и понимания, какая фигура где легла.