Министерство науки и высшего образования РФ
федеральное государственное автономное образовательное учреждение
высшего образования
«Омский государственный технический университет»

Лицей Академии Яндекса

Разработка и реализация игры “Ленточки”

Выполнили:
Мамонова Василиса Максимовна
Пирогова Екатерина Алексеевна

Омск, 2023

РЕФЕРАТ

Отчет 11 с., 9 рис., 3 источн., 1 прил.

ПРОЕКТ, РАЗРАБОТКА ИГР, PYTHON, PYGAME, КЛЕТЧАТОЕ ПОЛЕ, РЕЖИМЫ, ТАКТИКА

Предмет исследования – игра на клеточном поле.

Цель исследования – разработка игры “Ленточки” для знакомства с работой одной из библиотек Python – Pygame.

В процессе выполнения проекта мы изучили более подробно создание компьютерных игр с помощью библиотеки Pygame. В результате выполнения проекта мы разработали игру на клетчатом поле под названием “Ленточки”.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Компьютерные игры - один из широко известных видов досуга в современном мире. В них играют люди всех возрастов: и школьники, и дети, и взрослые - в особенности они популярны среди подростков. Ежегодно разрабатываются сотни игр, среди которых есть стратегии, аркады, РПГ и многие другие.

Заинтересовавшись процессом реализации, захотелось создать свою игру на клетчатом поле под названием “Ленточки”.

Цель исследования – разработка игры “Ленточки” для знакомства с одной из библиотек Python – Pygame.

Для достижения поставленной цели требуется решить следующие задачи:

1) предложить идею для игры;
2) спроектировать приложение для игры;
3) реализовать спроектированное приложение.

1 Описание игры

“Ленточки” - простая тактическая игра, суть которой в позиционной борьбе за пространство. На поле 16 х 16 игроки один за другим чертят небольшие линии. В игре есть два режима: “ленточки 2 на 1”, “ленточки разного размера”. В первом игроки перекрывают 2 любых клетки подряд: т.е. например, первый игрок проводит вертикальную линию, занимающую две стороны клеточки. Во втором режиме меняются лишь длины ленточек (от двух клеток до пяти).

Второй игрок делает то же самое, но его линия не может пересекать или соприкасаться с уже существующими заграждениями. По мере заполнения поля, остается все меньше свободного пространства. Игрок, которому уже некуда поставить свою черту, проигрывает.

2 Проектирование игры

Для работы программы планируется создать три класса: AnimatedSprite, Board и Board1. Первый предназначен для анимации спрайтов в конце игры.

Классы Board и Board1 рассчитаны для режимов игры “ленточки 2 на 1” и “ленточки разных размеров” соответственно. В обоих будут находиться инициализация, функции set_view, render, get_click, get_cell, on_click, begin и win_check.

Сперва создается поле, где указываются его размеры, переменная для определения хода игроков и формируются цвета будущих ленточек, а во втором классе еще появляется переменная, которая выбирает длины ленточек.

Set_view отвечает за настройку внешнего вида.

Функция render рисует поле и сами ленточки.

Get_click проверяет нажатие игрока:

- Если пользователь нажал на пустое пространство вне поля, то ничего не происходит;

- Если первый раз нажать на пустую клетку, то она выберется, закраситься в серый и на экран выведется соответствующее сообщение. Вторую клетку надо нажать в соответствии с правилами, иначе под полем появится текст об ошибке и выбранные клетки сбросятся.
- Если нажать на клетку, которая уже занята, то выведется сообщение “Эта клетка уже занята”

Функция get_cell возвращает координаты клетки в виде кортежа по переданным координатам мыши.

On_click изменяет поле, опираясь на полученные координаты клетки.

Begin возвращает клетку в первоначальный вид, если сделан неверный ход.

Функция win_check отвечает за проверку наличия возможных ходов. Если таковых нет, то игра заканчивается и пишется, какой игрок победил.

Вне классов находятся функции load_image, terminate, main, minor, start_menu, rules и choose_mode.

Первая проверяет наличие изображений, в противном случае выводит ошибку.

Функция terminate выходит из игры и закрывает окно.

В main находятся параметры поля, сообщения, которые пишутся во время игры под полем и говорят о результате, для режима “ленточки 2 на 1”. То же самое делает minor, но для режима “ленточки разных размеров”.

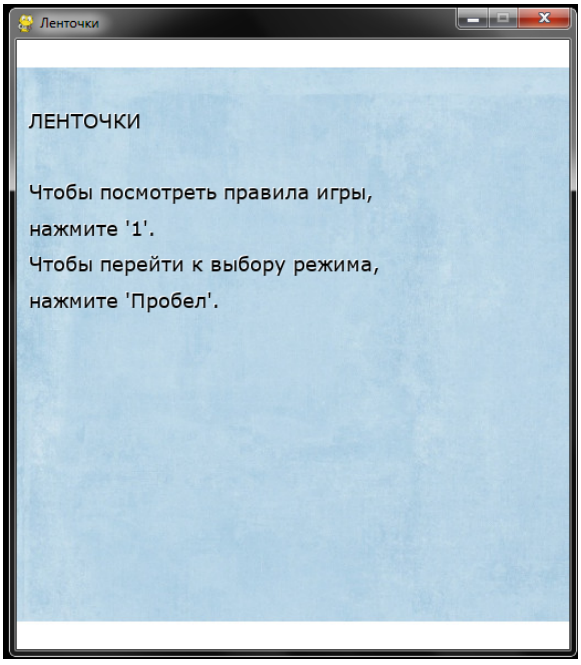
Start_menu - первоначальное окна игры.

Rules - окно с описанием правил.

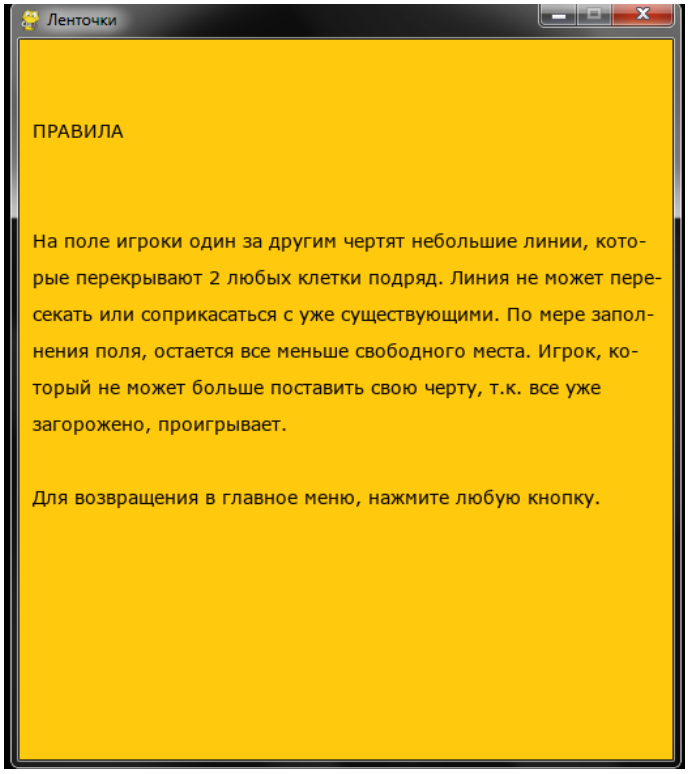
Choose_mode - окно выбора режима перед началом игры.

3 Реализация игры

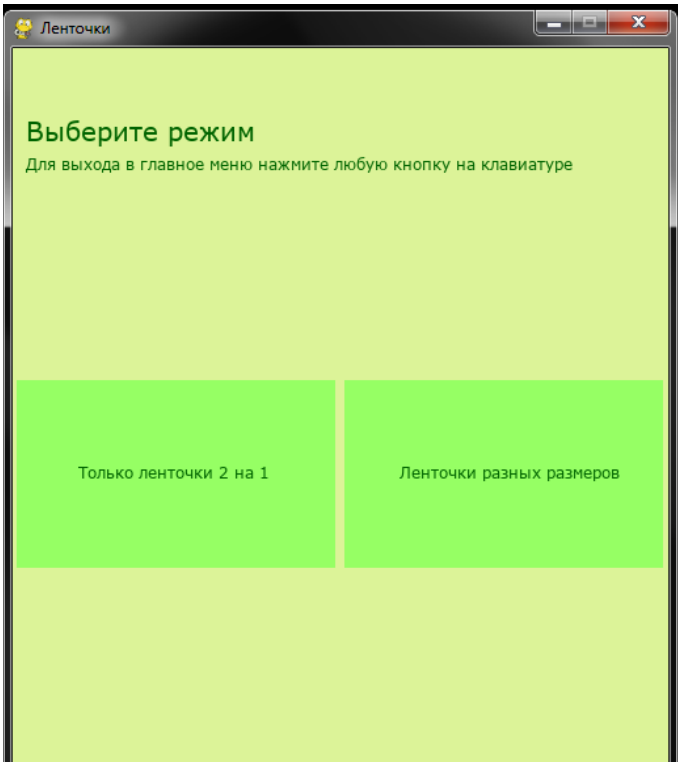
Первоначальное окно (функция start_menu)



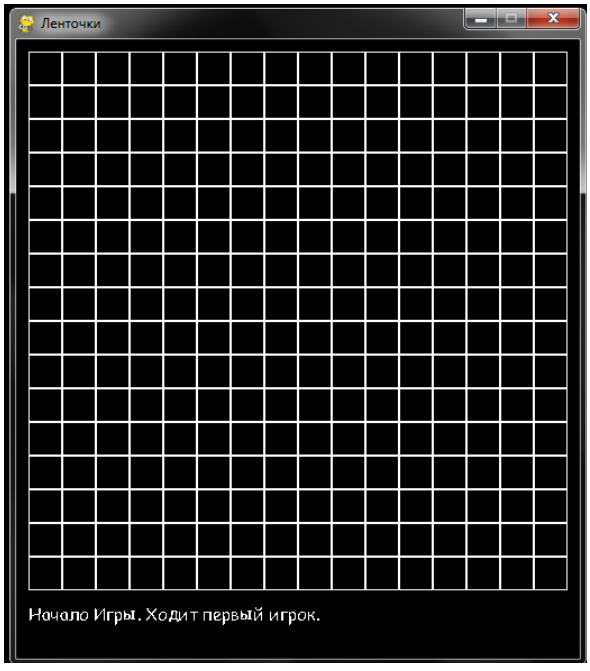
Окно описания правил игры (функция rules)

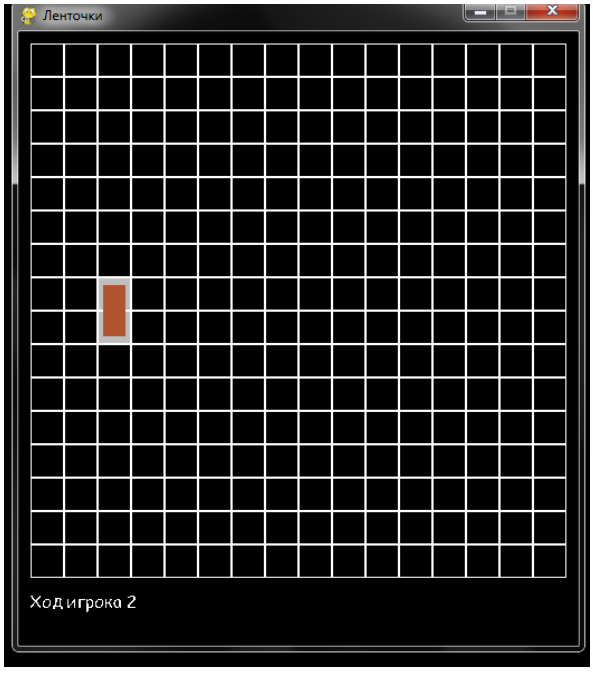


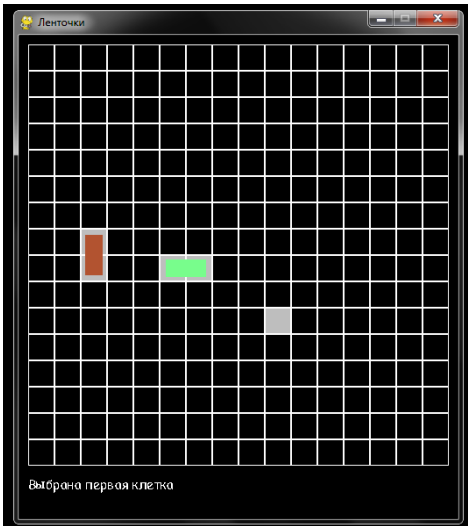
Окно выбора режима (функция choose_mode)

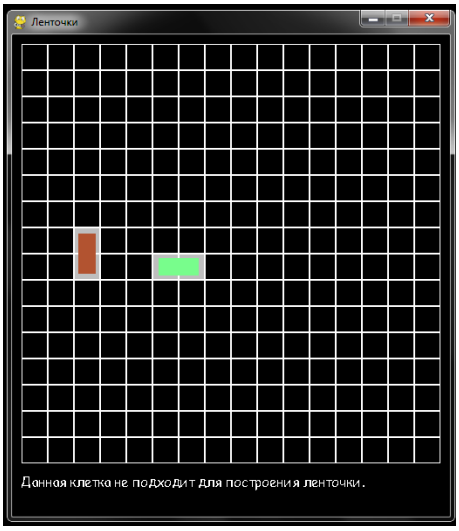


Игра в режиме “только ленточки 2 на 1 (класс Board)

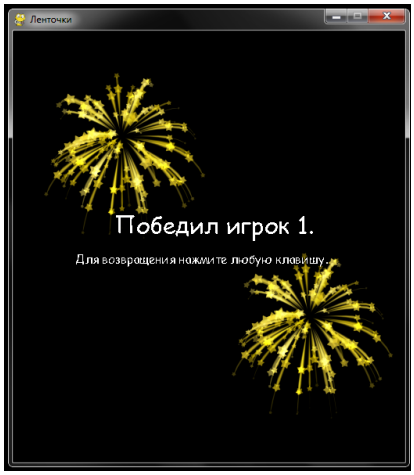




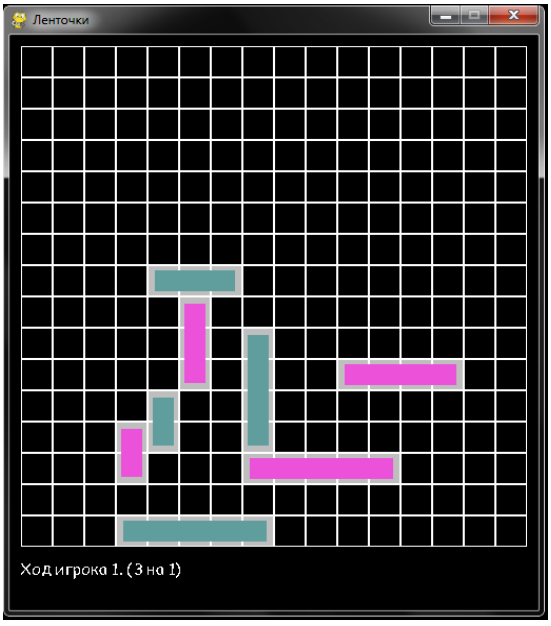




Конечное изображение



Режим “ленточки разных размеров”



ЗАКЛЮЧЕНИЕ

В ходе работы над проектом мы решили все задачи, которые ставили перед собой.

В начале мы взяли за основу тактическую игру “Ленточки”, над которой работали в дальнейшем, прописали её правила и условия выигрыша.

На втором этапе мы спроектировали приложение: создали классы и прописали функции, отвечающие за сам процесс игры.

На последнем этапе нашей работы мы реализовали приложение, представляющее собой игру на клеточном поле.

В дальнейшем планируется создать режим с игрой против искусственного интеллекта, разные уровни сложности и возможность выбора размера поля.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 - https://masterok.livejournal.com/805315.html - идея для игры

2 - https://lyceum.yandex.ru/courses/766/groups/6082/lessons/3551/materials/9817 - информация по созданию игр на клеточном поле с помощью библиотеки pygame

3 - https://russianblogs.com/article/56371249557/ - документация по pygame.mixer

ПРИЛОЖЕНИЕ А

(обязательное)

Листинг

main.py:

```
import pygame
import os
import sys
import random


class AnimatedSprite(pygame.sprite.Sprite):
    def __init__(self, sheet, columns, rows, x, y):
        super().__init__(all_sprites)
        self.frames = []
        self.cut_sheet(sheet, columns, rows)
        self.cur_frame = 0
        self.image = self.frames[self.cur_frame]
        self.rect = self.rect.move(x, y)

    def cut_sheet(self, sheet, columns, rows):
            self.rect = pygame.Rect(0, 0, sheet.get_width() //
columns,
                                    sheet.get_height() // rows)
        for j in range(rows):
            for i in range(columns):
                frame_location = (self.rect.w * i, self.rect.h *
j)
                self.frames.append(sheet.subsurface(pygame.Rect(
                    frame_location, self.rect.size)))

    def update(self):
        self.cur_frame = (self.cur_frame + 1) % len(self.frames)
        self.image = self.frames[self.cur_frame]


class Board:
    def __init__(self, width, height):
        self.width = width
        self.height = height
        self.board = [[0] * width for _ in range(height)]
        self.move = 1  # эта переменная определяет чей сейчас ход

        self.left = 10
        self.top = 10
        self.cell_size = 30
```

```
            self.flag = True
            self.ribbons = {}  # здесь будут хранится соединения
            self.ribbon_color1 = (190, 190, 190)
                 self.ribbon_color2 = (random.choice(range(1, 256)),
random.choice(range(1, 256)), random.choice(range(1, 256)))
                while self.ribbon_color1[0] in range(180, 211) and
self.ribbon_color1[1] in range(180, 211) \
                      and self.ribbon_color1[2] in range(180, 211):  #
проверка, что цвет не близок к серому
                self.ribbon_color1 = (
                                              random.choice(range(1, 256)),
random.choice(range(1, 256)), random.choice(range(1, 256)))
                while self.ribbon_color2[0] in range(180, 211) and
self.ribbon_color2[1] in range(180, 211) \
                    and self.ribbon_color2[2] in range(180, 211):
                self.ribbon_color2 = (
                                              random.choice(range(1, 256)),
random.choice(range(1, 256)), random.choice(range(1, 256)))
            flag = True
            while flag:  # проверка, что цвета не близки
                 if abs(self.ribbon_color2[0] - self.ribbon_color1[0])
>= 50 \
                                            and abs(self.ribbon_color2[1] -
self.ribbon_color1[1]) >= 50 \
                                            and abs(self.ribbon_color2[1] -
self.ribbon_color1[1]) >= 50:
                    flag = False
                else:
                    self.ribbon_color2 = (
                                             random.choice(range(1, 256)),
random.choice(range(1, 256)), random.choice(range(1, 256)))

        def set_view(self, left, top, cell_size):
            self.left = left
            self.top = top
            self.cell_size = cell_size

        def render(self, screen):
            colors = [pygame.Color('black'), pygame.Color('grey')]
            ribbon_colors = [self.ribbon_color1, self.ribbon_color2]
            for y in range(self.height):
                for x in range(self.width):
                    pygame.draw.rect(screen, colors[self.board[y][x]],
(x * self.cell_size + self.left,

y * self.cell_size + self.top,

self.cell_size, self.cell_size))
                    pygame.draw.rect(screen, pygame.Color('white'), (x
* self.cell_size + self.left,
```

```
                                                                    y
* self.cell_size + self.top,

self.cell_size,

self.cell_size), 1)
                        if (x, y) in self.ribbons.keys():  # прорисовка
ленточек
                            if self.ribbons[(x, y)][0] == "left":
                                                    pygame.draw.rect(screen,
ribbon_colors[self.ribbons[(x, y)][1] - 1],
                                                   ((x - 1) * self.cell_size
+ self.left + 7,
                                                        y * self.cell_size +
self.top + 5,
                                                   2 * self.cell_size - 14,
                                                   self.cell_size - 10))
                            if self.ribbons[(x, y)][0] == "up":
                                                    pygame.draw.rect(screen,
ribbon_colors[self.ribbons[(x, y)][1] - 1],
                                                       (x * self.cell_size +
self.left + 5,
                                                    (y - 1) * self.cell_size
+ self.top + 7,
                                                   self.cell_size - 10,
                                                        2 * self.cell_size -
14))

        def get_click(self, mouse_pos):
            cell = self.get_cell(mouse_pos)
              if cell:  # проверка, нажали ли на клетку или на пустое
пространство
                if self.board[cell[1]][cell[0]] == 1:
                    text = "Эта клетка уже занята."
                    pygame.mixer.Sound("data/wrong move.wav").play()
                elif self.flag:
                    self.on_click(cell)
                    self.first_cell = cell
                    self.flag = not self.flag
                    text = "Выбрана первая клетка"
                    pygame.mixer.Sound("data/click.wav").play()
                else:
                        if (abs(self.first_cell[0] - cell[0]) == 1 and
self.first_cell[1] == cell[1]) \
                                or (abs(self.first_cell[1] - cell[1]) == 1
and self.first_cell[0] == cell[0]):
                            self.on_click(cell)
                                if self.first_cell[0] - cell[0] == 1:  #
проверяем как находятся относительно друг друга клетки
                                    self.ribbons[self.first_cell] = ["left",
self.move]
                                self.ribbons[cell] = ["right", self.move]
```

```
                        elif self.first_cell[0] - cell[0] == -1:
                            self.ribbons[self.first_cell] = ["right",
self.move]
                            self.ribbons[cell] = ["left", self.move]
                        elif self.first_cell[1] - cell[1] == 1:
                            self.ribbons[self.first_cell] = ["up",
self.move]
                            self.ribbons[cell] = ["down", self.move]
                        else:
                            self.ribbons[self.first_cell] = ["down",
self.move]
                            self.ribbons[cell] = ["up", self.move]
                        self.move = 2 if self.move == 1 else 1
                        text = f"Ход игрока {self.move}"
                        pygame.mixer.Sound("data/click.wav").play()
                    else:
                        self.begin(self.first_cell)
                        text = 'Данная клетка не подходит для
построения ленточки.'
                        pygame.mixer.Sound("data/wrong
move.wav").play()
                    self.flag = not self.flag
                return text

        def get_cell(self, mouse_pos):
            for y in range(self.height):
                for x in range(self.width):
                    if mouse_pos[0] in range(x * self.cell_size +
self.left, (x + 1) * self.cell_size + self.left) and \
                            mouse_pos[1] in range(y * self.cell_size +
self.top, (y + 1) * self.cell_size + self.top):
                        return x, y
            return None

        def on_click(self, cell):
            self.board[cell[1]][cell[0]] = 1

        def begin(self, cell):
            self.board[cell[1]][cell[0]] = 0

        def win_check(self):  # проверка на наличие возможных ходов
            end = True
            for y in range(self.height):
                for x in range(self.width):
                    if (x, y) not in self.ribbons.keys():
                        if x != 0:
                            if (x - 1, y) not in self.ribbons.keys():
                                end = False
                        if x != self.width - 1:
                            if (x + 1, y) not in self.ribbons.keys():
                                end = False
                        if y != 0:
```

```
                            if (x, y - 1) not in self.ribbons.keys():
                                end = False
                        if y != self.height - 1:
                            if (x, y + 1) not in self.ribbons.keys():
                                end = False
                    if not end:
                        break
            return end


    class Board1:
        def __init__(self, width, height):
            self.width = width
            self.height = height
            self.board = [[0] * width for _ in range(height)]
            self.move = 0  # эта переменная определяет чей сейчас ход
            self.moves = [random.randint(2, 5) for _ in range(128)]

            self.left = 10
            self.top = 10
            self.cell_size = 30

            self.flag = True
            self.ribbons = {}  # здесь будут хранится соединения
            self.ribbon_color1 = (190, 190, 190)
                self.ribbon_color2 = (random.choice(range(1, 256)),
random.choice(range(1, 256)), random.choice(range(1, 256)))
                while self.ribbon_color1[0] in range(180, 211) and
self.ribbon_color1[1] in range(180, 211) \
                        and self.ribbon_color1[2] in range(180, 211):  #
проверка, что цвет не близок к серому
                    self.ribbon_color1 = (
                                            random.choice(range(1, 256)),
random.choice(range(1, 256)), random.choice(range(1, 256)))
                while self.ribbon_color2[0] in range(180, 211) and
self.ribbon_color2[1] in range(180, 211) \
                    and self.ribbon_color2[2] in range(180, 211):
                    self.ribbon_color2 = (
                                            random.choice(range(1, 256)),
random.choice(range(1, 256)), random.choice(range(1, 256)))
            flag = True
            while flag:  # проверка, что цвета не близки
                if abs(self.ribbon_color2[0] - self.ribbon_color1[0])
>= 50 \
                                        and abs(self.ribbon_color2[1] -
self.ribbon_color1[1]) >= 50 \
                                        and abs(self.ribbon_color2[1] -
self.ribbon_color1[1]) >= 50:
                    flag = False
                else:
                    self.ribbon_color2 = (
```

```
                                                random.choice(range(1,  256)),
random.choice(range(1, 256)), random.choice(range(1, 256)))

        def set_view(self, left, top, cell_size):
            self.left = left
            self.top = top
            self.cell_size = cell_size

        def render(self, screen):
            colors = [pygame.Color('black'), pygame.Color('grey')]
            ribbon_colors = [self.ribbon_color1, self.ribbon_color2]
            for y in range(self.height):
                for x in range(self.width):
                    pygame.draw.rect(screen, colors[self.board[y][x]],
(x * self.cell_size + self.left,

y * self.cell_size + self.top,

self.cell_size, self.cell_size))
                    pygame.draw.rect(screen, pygame.Color('white'), (x
* self.cell_size + self.left,
                                                                     y
* self.cell_size + self.top,

self.cell_size,

self.cell_size), 1)
                    if (x, y) in self.ribbons.keys():  # прорисовка
ленточек
                        if self.ribbons[(x, y)][0] == "left":
                            pygame.draw.rect(screen,
ribbon_colors[self.ribbons[(x, y)][1] - 1],
                                             ((x - self.ribbons[(x,
y)][2] + 1) * self.cell_size + self.left + 7,
                                              y * self.cell_size +
self.top + 5,
                                             self.ribbons[(x, y)][2]
* self.cell_size - 14,
                                             self.cell_size - 10))
                        if self.ribbons[(x, y)][0] == "up":
                            pygame.draw.rect(screen,
ribbon_colors[self.ribbons[(x, y)][1] - 1],
                                             (x * self.cell_size +
self.left + 5,
                                             (y - self.ribbons[(x,
y)][2] + 1) * self.cell_size + self.top + 7,
                                             self.cell_size - 10,
                                              self.ribbons[(x, y)][2]
* self.cell_size - 14))

        def get_click(self, mouse_pos):
            cell = self.get_cell(mouse_pos)
```

```
            if cell:  # проверка, нажали ли на клетку или на пустое
пространство
                if self.board[cell[1]][cell[0]] == 1:
                    text = "Эта клетка уже занята."
                    pygame.mixer.Sound("data/wrong move.wav").play()
                elif self.flag:
                    self.on_click(cell)
                    self.first_cell = cell
                    self.flag = not self.flag
                    text = "Выбрана первая клетка"
                    pygame.mixer.Sound("data/click.wav").play()
                else:
                                if (abs(self.first_cell[0] - cell[0]) ==
self.moves[self.move] - 1 and self.first_cell[1]
                                    == cell[1]) or (abs(self.first_cell[1] -
cell[1]) == self.moves[self.move] - 1 and
                                                    self.first_cell[0] ==
cell[0]):
                                    if self.first_cell[0] - cell[0] ==
self.moves[self.move] - 1:
                                if self.moves[self.move] > 2:
                                    if any([self.board[cell[1]][i] for i
in range(cell[0] + 1, self.first_cell[0])]):
                                        self.begin(self.first_cell)
                                        text = f'Данная клетка не подходит
для построения ленточки. ' \
                                               f'({self.moves[self.move]}
на 1)'
                                        pygame.mixer.Sound("data/wrong
move.wav").play()
                                        self.flag = not self.flag
                                        return text
                                self.on_click(cell)
                                    self.ribbons[self.first_cell] = ["left",
self.move % 2 + 1, self.moves[self.move]]
                                self.ribbons[cell] = ["right", self.move %
2 + 1, self.moves[self.move]]
                                    for i in range(cell[0] + 1,
self.first_cell[0]):
                                self.board[cell[1]][i] = 1
                                elif self.first_cell[0] - cell[0] ==
-self.moves[self.move] + 1:
                                if self.moves[self.move] > 2:
                                    if any([self.board[cell[1]][i] for i
in range(self.first_cell[0] + 1, cell[0])]):
                                        self.begin(self.first_cell)
                                        text = f'Данная клетка не подходит
для построения ленточки. ' \
                                               f'({self.moves[self.move]}
на 1)'
                                        pygame.mixer.Sound("data/wrong
move.wav").play()
```

```
                                    self.flag = not self.flag
                                    return text
                                self.on_click(cell)
                                self.ribbons[self.first_cell] = ["right",
self.move % 2 + 1, self.moves[self.move]]
                                self.ribbons[cell] = ["left", self.move %
2 + 1, self.moves[self.move]]
                                for i in range(self.first_cell[0] + 1,
cell[0]):
                                    self.board[cell[1]][i] = 1
                                elif self.first_cell[1] - cell[1] ==
self.moves[self.move] - 1:
                                if any([self.board[i][cell[0]] for i in
range(cell[1] + 1, self.first_cell[1])]):
                                    self.begin(self.first_cell)
                                    text = f'Данная клетка не подходит для
построения ленточки. ' \
                                           f'({self.moves[self.move]} на
1)'
                                    pygame.mixer.Sound("data/wrong
move.wav").play()
                                    self.flag = not self.flag
                                    return text
                                self.on_click(cell)
                                    self.ribbons[self.first_cell] = ["up",
self.move % 2 + 1, self.moves[self.move]]
                                self.ribbons[cell] = ["down", self.move %
2 + 1, self.moves[self.move]]
                                    for i in range(cell[1] + 1,
self.first_cell[1]):
                                    self.board[i][cell[0]] = 1
                            else:
                                if any([self.board[i][cell[0]] for i in
range(self.first_cell[1] + 1, cell[1])]):
                                    self.begin(self.first_cell)
                                    text = f'Данная клетка не подходит для
построения ленточки. ' \
                                           f'({self.moves[self.move]} на
1)'
                                    pygame.mixer.Sound("data/wrong
move.wav").play()
                                    self.flag = not self.flag
                                    return text
                                self.on_click(cell)
                                  self.ribbons[self.first_cell] = ["down",
self.move % 2 + 1, self.moves[self.move]]
                                self.ribbons[cell] = ["up", self.move % 2
+ 1, self.moves[self.move]]
                                for i in range(self.first_cell[1] + 1,
cell[1]):
                                    self.board[i][cell[0]] = 1
                            self.move += 1
```

```
                                text = f"Ход игрока {self.move % 2 + 1}.
({self.moves[self.move]} на 1)"
                            pygame.mixer.Sound("data/click.wav").play()
                        else:
                            self.begin(self.first_cell)
                                  text = f'Данная клетка не подходит для
построения ленточки. ({self.moves[self.move]} на 1)'
                                            pygame.mixer.Sound("data/wrong
move.wav").play()
                        self.flag = not self.flag
                    return text

        def get_cell(self, mouse_pos):
            for y in range(self.height):
                for x in range(self.width):
                    if mouse_pos[0] in range(x * self.cell_size +
self.left, (x + 1) * self.cell_size + self.left) and \
                            mouse_pos[1] in range(y * self.cell_size +
self.top, (y + 1) * self.cell_size + self.top):
                        return x, y
            return None

        def on_click(self, cell):
            self.board[cell[1]][cell[0]] = 1

        def begin(self, cell):
            self.board[cell[1]][cell[0]] = 0

        def win_check(self):  # проверка на наличие возможных ходов
            end = True
            for y in range(self.height):
                for x in range(self.width):
                    if self.board[y][x] == 0:
                        if x >= self.moves[self.move]:
                                if not any([self.board[y][i] for i in
range(x - self.moves[self.move] + 1, x)]):
                                    end = False
                                    break
                        if x <= self.width - self.moves[self.move]:
                                if not any([self.board[y][i] for i in
range(x, x + self.moves[self.move])]):
                                    end = False
                                    break
                        if y >= self.moves[self.move]:
                                    if not any([self.board[i][x]] for i in
range(y - self.moves[self.move] + 1, y)):
                                    end = False
                                    break
                        if y <= self.height - self.moves[self.move]:
                                if not any([self.board[i][x] for i in
range(y, y + self.moves[self.move])]):
                                    end = False
```

```
                                    break
                        if not end:
                            break
                return end


        def load_image(name, color_key=None):
            fullname = os.path.join('data', name)
            try:
                image = pygame.image.load(fullname)
            except pygame.error as message:
                print(f'В папке отсутствует файл: {name}')
                raise SystemExit(message)
            if color_key == -1:
                color_key = image.get_at((0, 0))
                image.set_colorkey(color_key)
            else:
                image = image.convert_alpha()
            return image


        def terminate():
            pygame.quit()
            sys.exit()


        pygame.mixer.pre_init(44100, -16, 1, 512)
        pygame.init()
        pygame.font.init()

        size = 500, 550
        screen = pygame.display.set_mode(size)
        pygame.display.set_caption("Ленточки")
        all_sprites = pygame.sprite.Group()


        def main():
            my_font = pygame.font.SysFont('Comic Sans MS', 15)
              text_surface = my_font.render('Начало Игры. Ходит первый
игрок.', False, (255, 255, 255))

            board = Board(16, 16)
            board.set_view(10, 10, 30)
            running = True
            victory = False

            while running:
                for event in pygame.event.get():
                    if event.type == pygame.QUIT:
                        running = False
                    elif event.type == pygame.MOUSEBUTTONDOWN:
                        if not victory:
```

```
                        text = board.get_click(event.pos)
                        if text:
                            text_surface = my_font.render(text, False,
(255, 255, 255))
                            if board.win_check() and "Выбрана" not in
text:
                            pygame.mixer.Sound("data/win.wav").play()
                            victory = True
                            win(1 if board.move % 2 + 1 == 2 else 2)
                elif event.type == pygame.KEYDOWN:
                    start_menu()
            screen.fill((0, 0, 0))
            board.render(screen)
            screen.blit(text_surface, (10, 500))
            pygame.display.flip()
        pygame.quit()


    def minor():
        my_font = pygame.font.SysFont('Comic Sans MS', 15)

        board = Board1(16, 16)
        board.set_view(10, 10, 30)
        running = True
        victory = False

          text_surface = my_font.render(f'Начало Игры. Ходит первый
игрок.({board.moves[board.move]} на 1)', False,
                                      (255, 255, 255))

        while running:
            for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    running = False
                elif event.type == pygame.MOUSEBUTTONDOWN:
                    if not victory:
                        text = board.get_click(event.pos)
                        if text:
                            text_surface = my_font.render(text, False,
(255, 255, 255))
                        if "Выбрана" not in text if text else "":
                            if board.win_check():

pygame.mixer.Sound("data/win.wav").play()
                                victory = True
                                win(1 if board.move % 2 + 1 == 2 else
2)
                elif event.type == pygame.KEYDOWN:
                    start_menu()
            screen.fill((0, 0, 0))
            board.render(screen)
            screen.blit(text_surface, (10, 500))
```

```
            try:
                pygame.display.flip()
            except pygame.error:
                break
        pygame.quit()


    def start_menu():
        intro_text = ["ЛЕНТОЧКИ", "",
                      "Чтобы посмотреть правила игры,",
                      "нажмите '1'.",
                      "Чтобы перейти к выбору режима,",
                      "нажмите 'Пробел'."]

        fon = pygame.transform.scale(load_image('fon.jpg'), size)
        screen.blit(fon, (0, 0))
        font = pygame.font.SysFont('Verdana', 18)
        text_coord = 50
        for line in intro_text:
                            string_rendered   =   font.render(line,   1,
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)
        text_coord = 51
        for line in intro_text:
                            string_rendered   =   font.render(line,   1,
pygame.Color('black'))
            intro_rect = string_rendered.get_rect()
            text_coord += 10
            intro_rect.top = text_coord
            intro_rect.x = 11
            text_coord += intro_rect.height
            screen.blit(string_rendered, intro_rect)

        while True:
            for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    terminate()
                elif event.type == pygame.KEYDOWN:
                    if event.key == pygame.K_SPACE:
                        pygame.mixer.Sound("data/click.wav").play()
                        choose_mode()
                    elif event.key == pygame.K_1:
                        pygame.mixer.Sound("data/click.wav").play()
                        rules()
            try:
                pygame.display.flip()
            except pygame.error:
```

```
                break


    def rules():
        intro_text = ["ПРАВИЛА", "", "",
                      "На поле игроки один за другим чертят небольшие
линии, кото-",
                     "рые перекрывают 2 любых клетки подряд. Линия не
может пере-",
                      "секать или соприкасаться с уже существующими.
По мере запол-",
                       "нения поля, остается все меньше свободного
места. Игрок, ко-",
                      "торый не может больше поставить свою черту,
т.к. все уже",
                     "загорожено, проигрывает.", "",
                      "Для возвращения в главное меню, нажмите любую
кнопку."]

        fon = pygame.transform.scale(load_image('fon2.jpg'), size)
        screen.blit(fon, (0, 0))
        font = pygame.font.SysFont('Verdana', 14)
        text_coord = 50
        for line in intro_text:
                        string_rendered   =   font.render(line,   1,
pygame.Color('Black'))
            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)

        while True:
            for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    terminate()
                elif event.type == pygame.KEYDOWN or \
                        event.type == pygame.MOUSEBUTTONDOWN:
                    pygame.mixer.Sound("data/click.wav").play()
                    start_menu()
            try:
                pygame.display.flip()
            except pygame.error:
                break


    def choose_mode():
        fon = pygame.transform.scale(load_image('fon3.jpg'), size)
        screen.blit(fon, (0, 0))
        font = pygame.font.SysFont('Verdana', 20)
        font1 = pygame.font.SysFont('Verdana', 12)
```

```
          string_rendered = font.render("Выберите режим", 1, (0, 100,
0))
        intro_rect = string_rendered.get_rect()
        intro_rect.top, intro_rect.x = 50, 10
        screen.blit(string_rendered, intro_rect)
          string_rendered = font1.render("Для выхода в главное меню
нажмите любую кнопку на клавиатуре", 1, (0, 100, 0))
        intro_rect = string_rendered.get_rect()
        intro_rect.top, intro_rect.x = 80, 10
        screen.blit(string_rendered, intro_rect)

         pygame.draw.rect(screen, (150, 255, 100), (3, 253, 243, 143),
0)
          pygame.draw.rect(screen, (150, 255, 100), (253, 253, 243,
143), 0)

          string_rendered = font1.render("Только ленточки 2 на 1", 1,
(0, 100, 0))
        intro_rect = string_rendered.get_rect()
        intro_rect.top, intro_rect.x = 315, 50
        screen.blit(string_rendered, intro_rect)
         string_rendered = font1.render("Ленточки разных размеров", 1,
(0, 100, 0))
        intro_rect = string_rendered.get_rect()
        intro_rect.top, intro_rect.x = 315, 295
        screen.blit(string_rendered, intro_rect)

        while True:
            for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    terminate()
                elif event.type == pygame.KEYDOWN:
                    pygame.mixer.Sound("data/click.wav").play()
                    start_menu()
                elif event.type == pygame.MOUSEBUTTONDOWN:
                     if event.pos[0] in range(3, 246) and event.pos[1]
in range(253, 396):
                        pygame.mixer.Sound("data/click.wav").play()
                        main()
                            elif event.pos[0] in range(253, 496) and
event.pos[1] in range(253, 396):
                        pygame.mixer.Sound("data/click.wav").play()
                        minor()
            try:
                pygame.display.flip()
            except pygame.error:
                break


    def win(player):
        running = True
        fps = 20
```

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

         firework1 = AnimatedSprite(load_image("Firework.png"), 6, 5,
10, 40)
         firework2 = AnimatedSprite(load_image("Firework.png"), 6, 5,
250, 270)
        while running:
            for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    running = False
                elif event.type == pygame.KEYDOWN:
                    pygame.mixer.Sound("data/click.wav").play()
                    start_menu()
            screen.fill((0, 0, 0))
            all_sprites.draw(screen)
            my_font = pygame.font.SysFont('Comic Sans MS', 15)
            my_font1 = pygame.font.SysFont('Comic Sans MS', 30)
             text_surface = my_font1.render(f'Победил игрок {player}.',
False, (255, 255, 255))
            screen.blit(text_surface, (130, 225))
               text_surface = my_font.render('Для возвращения нажмите
любую клавишу.', False, (255, 255, 255))
            screen.blit(text_surface, (80, 280))
            firework1.update()
            firework2.update()
            clock.tick(fps)
            pygame.display.flip()
        pygame.quit()


    start_menu()
```