Прямоугольники и змейки

Ким Анастасия

Я создала игру под названием “Прямоугольники и змейки”.

Правила игры

По прямоугольному полю бегает курсор. Им можно управлять с помощью клавиатуры. Кнопки W, A, S, D отвечают за перемещение. Внутри прямоугольника летают змейки(их количество зависит от уровня). При нажатии на SPACE проводится перпендикуляр к стороне, на которой стоит курсор. Если перпендикуляр пересекает змейку, то пользователь проигрывает. Если змейки не задеты, то количество срезов уменьшается на 1 и прямоугольник уменьшается. Перпендикуляр срезает меньшую часть прямоугольника. Если количество срезов равно 0 или хотя бы одна из сторон меньше 200 пикселей, то пользователь выигрывает(все результаты можно посмотреть в базе данных)

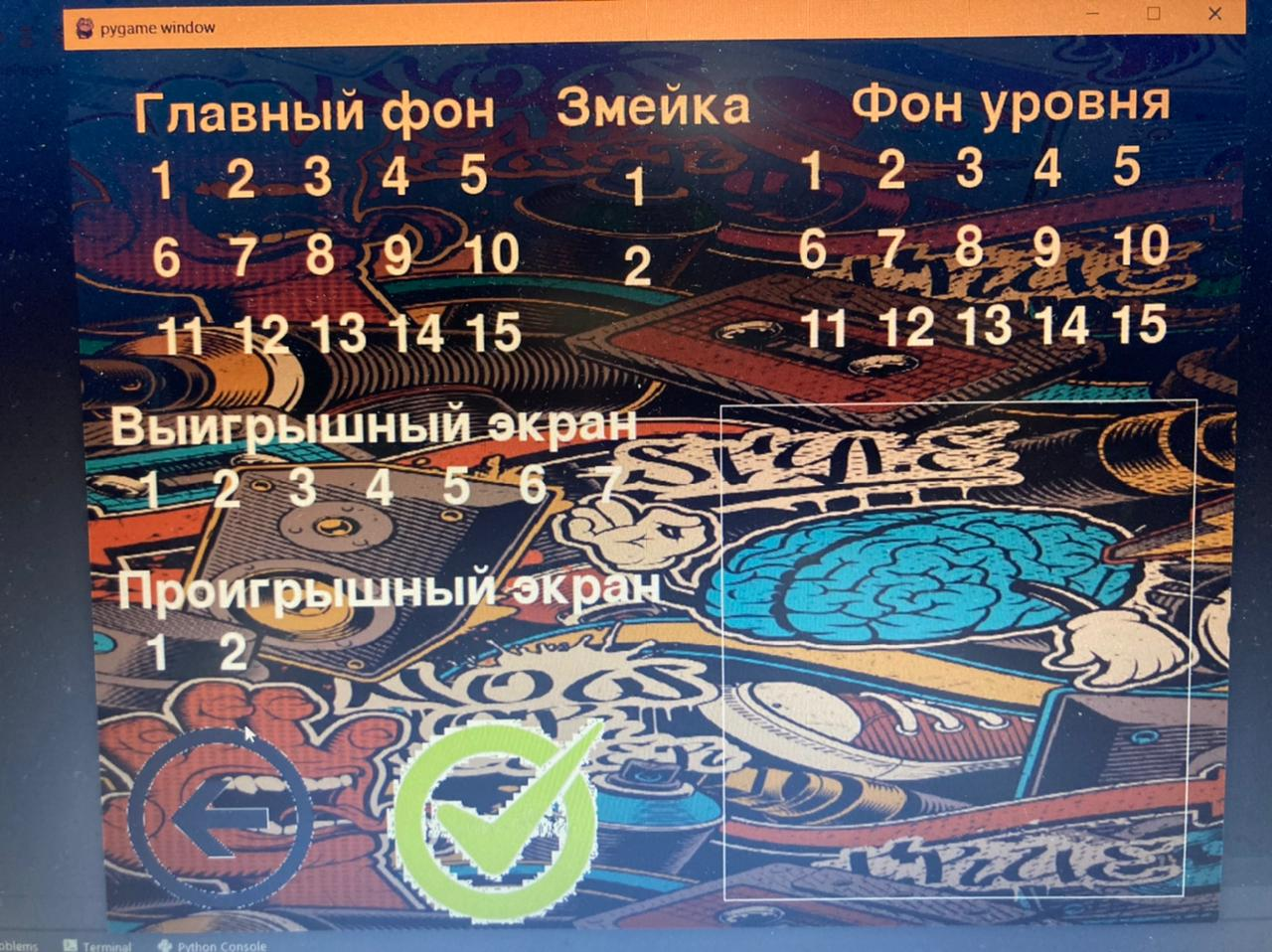
Факты о реализации

1. Я создала заставку игры, с которой начинается игра. Из нее можно выйти на Правила игры, Уровни игры, Изменить дизайн и Выйти из игры. У каждого окна(Правила, Начать игру, Изменить дизайн) есть своя функция и спрайты. Также там есть кнопка «Назад» в виде стрелки или кнопки Выхода

def splash\_screen():  
 intro\_text = ['Прямоугольники и змейки', '', 'Правила игры',  
 'Начать игру',  
 'Изменение дизайна',  
 'Выйти из игры']  
 fon = pygame.transform.scale(load\_image(main\_fon), size)  
 screen.blit(fon, (0, 0))  
 font = pygame.font.Font(None, 70)  
 text\_coord = 50  
 for line in intro\_text:  
 string\_rendered = font.render(line, True, pygame.Color('white'))  
 intro\_rect = string\_rendered.get\_rect()  
 text\_coord += 50  
 intro\_rect.top = text\_coord  
 intro\_rect.x = 10  
 text\_coord += intro\_rect.height  
 screen.blit(string\_rendered, intro\_rect)

1. Одна из идей состоит в том что для каждого набора(особенно в окне «Изменить дизайн» я создаю словарик, с ключами, равными номеру уровня/дизайна, а значение ключа – это в look\_levels() координаты текста (x, y, ширина и высота), но в «Изменить дизайн» значением ключа является список, в котором первый элемент – имя изображения в формате png, а второе – кортеж с координатами. И потом я создала функции check\_click() и check\_click\_for\_fon(), которая попадает ли мышка(координаты мышки) на определенный текст и возвращает номер уровня/номер дизайна. Затем работает функция getting\_from\_dict(x, y, dc). Если check\_click\_for\_fon(), что-то нашла,то функция возвращает номер дизайна и от чего этот дизайн. Для этого в конце списков сделала ключ и его значение с названием списка. Таким образом, можно удобно менять количество уровней(главное, чтобы все вместилось)

levels\_dict\_coord = {'1': '', '2': '', '3': '', '4': '', '5': '',  
 '6': '', '7': '', '8': '', '9': '', '10': '',  
 '11': '', '12': '', '13': '', '14': '', '15': '',  
 '16': '', '17': '', '18': '', '19': '', '20': ''  
 }  
main\_fon\_dict\_coord = {'1': ['fon1.png'], '2': ['fon2.png'], '3': ['fon3.png'], '4': ['fon4.png'], '5': ['fon5.png'],  
 '6': ['fon6.png'], '7': ['fon7.png'], '8': ['fon8.png'], '9': ['fon9.png'], '10': ['fon10.png'],  
 '11': ['fon11.png'], '12': ['victory\_fon.png'], '13': ['fon13.png'], '14': ['fon14.png'],  
 '15': ['fon15.png'],  
 'main': 'main'  
 }  
level\_fon\_dict\_coord = {'1': ['fon1.png'], '2': ['fon2.png'], '3': ['fon3.png'], '4': ['fon4.png'], '5': ['fon5.png'],  
 '6': ['fon6.png'], '7': ['fon7.png'], '8': ['fon8.png'], '9': ['fon9.png'], '10': ['fon10.png'],  
 '11': ['fon11.png'], '12': ['fon12.png'], '13': ['fon13.png'], '14': ['fon14.png'],  
 '15': ['fon15.png'],  
 'level': 'level'  
 }  
victory\_fon\_dict\_coord = {'1': ['victory\_fon.png'], '2': ['victory\_fon2.png'], '3': ['victory\_fon3.png'],  
 '4': ['victory\_fon4.png'], '5': ['victory\_fon5.png'], '6': ['victory\_fon6.png'],  
 '7': ['victory\_fon7.png'],  
 'victory': 'victory'}  
losing\_fon\_dict\_coord = {'1': ['losing\_fon1.png'], '2': ['losing\_fon2.png'], 'losing': 'losing'}  
  
snake\_dict\_coord = {'1': ['snake1.png'], '2': ['snake\_animation2.png'], 'snake': 'snake'}  
  
dc\_of\_all\_dict = {'main': main\_fon\_dict\_coord, 'level': level\_fon\_dict\_coord, 'victory': victory\_fon\_dict\_coord,  
 'losing': losing\_fon\_dict\_coord, 'snake': snake\_dict\_coord}



1. Также я сделала такую функцию игры, что можно менять дизайн фона главного экрана, уровня, проигрышного экрана окончания, выигрышного экрана окончания. При нажатии на номер дизайна помимо функции check\_сlick\_for\_fon() срабатывает функция putting\_image(), которая ставит изображение в прямоугольник. Для подтверждения выбора нужно нажать на зеленую галочку и экран поменяется, так как сработает функция changing\_design()

def check\_click\_for\_fon(mouse\_x, mouse\_y, tuple\_of\_coord, dc):  
 text\_x, text\_y, text\_w, text\_h = tuple\_of\_coord  
 start\_x, end\_x, start\_y, end\_y = text\_x, text\_x + text\_w, text\_y, text\_y + text\_h  
 if start\_x <= mouse\_x <= end\_x and start\_y <= mouse\_y <= end\_y:  
 for i in list(dc.keys()):  
 if dc[i][1] == tuple\_of\_coord:  
 number = int(i)  
 return number  
 else:  
 return ''

def getting\_num\_from\_dict(x, y, dc):  
 for i in list(dc.values())[:len(list(dc.values())) - 1]:  
 something = check\_click\_for\_fon(x, y, i[1], dc)  
 if something != '':  
 return (something, list(dc.values())[-1])  
 return ('', '')

1. Также я узнала, что можно останавливать некоторые спрайты, в то время как остальные будут работать и при этом анализировать столкновения. Я сделала несколько pygame.sprite.Group(), тогда для некоторых я делаю update(), а для некоторых я его не делаю. Чтобы понимать когда остановить змейки, ввожу переменную pause. Изначально она равна False, но после нажатия на SPACE, она становится True. Затем идет анализ, пользователь выиграл или проиграл. Если collide дает, что змейки не задеты, то victory = True, анализируем количество срезов и размеры прямоугольника. Перпендикуляр делит прямоугольник на две части. Компьютер «убирает»(срезает) меньшую, уменьшая прямоугольник. Если количество срезов > 0, то уменьшаю на 1. Если равно 0, то пользователь выиграл. Также если любая сторона прямоугольника < 200 пикселей, то пользователь тоже выиграл (вне зависимости от количества срезов)

if not victory:  
 if pause:

if count\_of\_done\_cuts < count\_cuts:  
 if cursor.rect.x == x1 - 15 or cursor.rect.x == x2 - 15:  
 if cursor.rect.y - 150 <= (y2 - y1) // 2:  
 y1 = cursor.rect.y  
 if abs(x2 - x1) < 200 or abs(y2 - y1) < 200:  
 count\_of\_done\_cuts = count\_cuts  
 break  
 launch\_level(number\_of\_level, x1, y1, x2, y2)  
 else:  
 y2 = cursor.rect.y  
 if abs(x2 - x1) < 200 or abs(y2 - y1) < 200:  
 count\_of\_done\_cuts = count\_cuts  
 break  
  
 launch\_level(number\_of\_level, x1, y1, x2, y2)  
 elif cursor.rect.y == y1 - 15 or cursor.rect.y == y2 - 15:  
 if cursor.rect.x - 150 <= (x2 - x1) // 2:  
 x1 = cursor.rect.x  
 if abs(x2 - x1) < 200 or abs(y2 - y1) < 200:  
 count\_of\_done\_cuts = count\_cuts  
 break  
  
 launch\_level(number\_of\_level, x1, y1, x2, y2)  
 else:  
 x2 = cursor.rect.x  
 if x2 - x1 < 200 or y2 - y1 < 200:  
 count\_of\_done\_cuts = count\_cuts  
 break  
 launch\_level(number\_of\_level, x1, y1, x2, y2)  
else:  
 victory = True  
 pause = True  
  
 connection = sqlite3.connect('game\_result\_data\_base.db')  
 cursor = connection.cursor()  
 date = str(datetime.datetime.now())  
 cursor.execute(  
 f"""INSERT INTO results(time, result, level) VALUES('{date}', '{str(time\_watch / 1000)}', \  
 '{str(number\_of\_level)}')""")  
 connection.commit()  
 time\_watch = 0  
 count\_of\_done\_cuts = 0  
 victory\_screen(number\_of\_level)

1. Я подключила базу данных. Туда записывается время с помощью datetime и cекундомер из pygame с помощью pygame и уровень, который был пройден.

connection = sqlite3.connect('game\_result\_data\_base.db')  
cursor = connection.cursor()  
date = str(datetime.datetime.now())  
cursor.execute(  
 f"""INSERT INTO results(time, result, level) VALUES('{date}', '{str(time\_watch / 1000)}', \  
 '{str(number\_of\_level)}')""")  
connection.commit()

1. Для секундомера я использовала pygame.Clock(). Для этого создаю глобальную переменную time\_watch = 0. Потом увеличиваю его на clock.tick\_busy\_loop(). Также вывожу время на экран для пользователя.

При этом делаю clock.tick(). Чтобы компьютер отсчитывал каждую миллисекунду, то есть time\_watch – это количество времени с момента начала игры в миллисекундах, но это не очень удобно, поэтому я вывожу на экран time\_watch / 1000, переводя значение в секунды.

time\_watch += clock.tick\_busy\_loop()  
text\_time = font.render('Время: ' + str(time\_watch / 1000), True, (255, 255, 255))  
screen.blit(text\_time, (280, 90))

1. Также для того чтобы пользователь, понял почему он проиграл(какая именно змейка задета), компьютер рисует более толстую линию красного цвета.

def update(self):  
 if pygame.sprite.spritecollide(self, all\_snakes, False):  
 if self.type\_of\_line == 'vertical':  
 pygame.draw.line(screen, (255, 0, 0), (self.ax + 1, 135), (self.ax + 1, 570), 7)  
 self.losing = True  
 elif self.type\_of\_line == 'horizontal':  
 pygame.draw.line(screen, (255, 0, 0), (135, self.ay + 1), (765, self.ay + 1), 7)  
 self.losing = True

1. В заставке проигрыша и выигрыша есть кнопки заново и выхода. Для запуска заново я снова использую launch\_level() со старыми(начальными) координатами, для того чтобы срезать и запускать заново прямоугольник(при победном случае, где перпендикуляр не задевает змейку), вне всех функций использую кортеж coord\_of\_rectangle, что делает прямоугольник гибче, так как координаты прямоугольника – аргументы launch\_level

launch\_level(number\_of\_level, x1, y1, x2, y2)

1. Чтобы удобнее менять сложность уровня я создала функцию change\_diff, которая возвращает количество срезов и змеек для каждого уровня отдельно. Это делает удобным добавление уровней

def change\_diff(number\_of\_level):  
 if number\_of\_level == 1:  
 return 10, 5  
 elif number\_of\_level == 2:  
 return 10, 6  
 elif number\_of\_level == 3:  
 return 12, 6  
 elif number\_of\_level == 4:  
 return 14, 7  
 elif number\_of\_level == 5:  
 return 16, 7  
 elif number\_of\_level == 6:  
 return 16, 8  
 elif number\_of\_level == 7:  
 return 17, 8  
 elif number\_of\_level == 8:  
 return 18, 8  
 elif number\_of\_level == 9:  
 return 19, 7  
 elif number\_of\_level == 10:  
 return 19, 8  
 elif number\_of\_level == 11:  
 return 19, 9  
 elif number\_of\_level == 12:  
 return 20, 7  
 elif number\_of\_level == 13:  
 return 20, 8  
 elif number\_of\_level == 14:  
 return 21, 7  
 elif number\_of\_level == 15:  
 return 21, 8  
 elif number\_of\_level == 16:  
 return 21, 9  
 elif number\_of\_level == 17:  
 return 22, 9  
 elif number\_of\_level == 18:  
 return 23, 8  
 elif number\_of\_level == 19:  
 return 24, 9  
 elif number\_of\_level == 20:  
 return 25, 10

1. Чтобы все фоны, поставленные на экран в текущий момент сохранялись, записываю их в текстовый файл fon.txt, а при запуске игры открываю и беру оттуда все названия фонов.

file\_of\_fon = open('fon.txt', 'w', encoding='utf-8')  
file\_of\_fon.write(main\_fon + '\n')  
file\_of\_fon.write(level\_fon + '\n')  
file\_of\_fon.write(victory\_fon + '\n')  
file\_of\_fon.write(losing\_fon + '\n')  
file\_of\_fon.write(snake\_fon + '\n')  
file\_of\_fon.close()  
changing\_design()

1. И наконец, в конце кода использую pygame.quit(), чтобы подружиться с pygame(Пункт не всерьез)

Выводы:

Работа с pygame мне очень даже понравилась. Прежде я пробовал сделать игры, но ничего не получалось. Я бы хотела сделать бонусный уровень в виде другой игры, старая идея. Скорее всего в свободное время сделаю.

Необходимые библиотеки:

pygame~=2.0.1  
Pillow~=8.1.0