Авторы проекта: Савка Матвей Иванович и Егор Клисак Викторович

Идея проекта

Идея проекта заключалась в том, чтобы создать кооперативную игру в жанре файтинг, которая будет вызывать азарт и интерес у игроков.

Как была придумана идея

Всё началось с того, как наш преподаватель попросил придумать идею для проекта видеоигры. Сначала для меня стоял вопрос выбора жанра. Я выбирал между платформером и файтингом. В конце концов я выбрал файтинг.

Название игры

Название игры, конечно же, было придумано с ориентировкой на популярную серию игр Mortal Kombat. А вообще Mortal Fight переводится как смертельная битва.

Структура данных, описание работы приложения

Применённые сторонние библиотеки



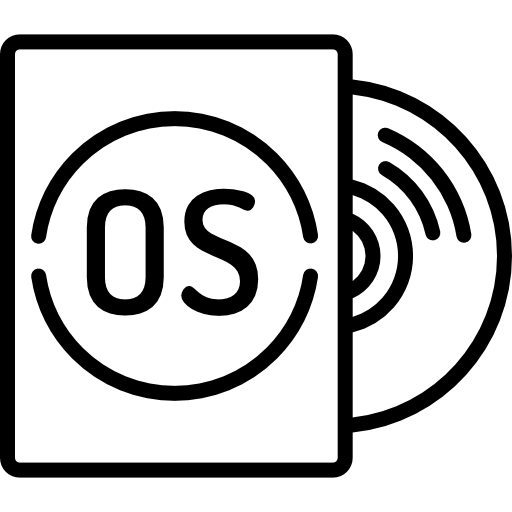
Игровой движок



Создание видеозаставки перед игрой



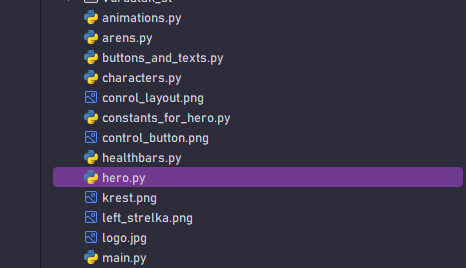
Нужна для применения асинхронности в игре



Нужна для работы с файлами

Описание работы участков кода

Всего файлов типа py в проекте 8



Начнём с основного файла main

import pygame # импортируем pygame как игровой движок  
import asyncio # импортируем asyncio для создания асинхронности  
import os # импортируем os для работы с файлами  
from moviepy.editor import VideoFileClip # импортируем нужную функцию из moviepy для создания видеозаставки  
from healthbars import Healthbars # Импортируем класс Healthbars  
from constants\_for\_hero import \* # Импортируем все константы персонажей  
from hero import Hero # импортируем класс Hero  
from buttons\_and\_texts import \*

# Инициализация Pygame  
pygame.init()  
  
# Собираем информацию о разрешении экрана пользователя  
user\_screen\_info = pygame.display.Info()  
user\_screen\_width = user\_screen\_info.current\_w  
user\_screen\_height = user\_screen\_info.current\_h  
  
screen = pygame.display.set\_mode((user\_screen\_width**,** user\_screen\_height)) # Задаём разрешение основного окна  
pygame.display.set\_caption("Mortal Fight") # Задаём название программе  
icon = pygame.image.load("logo.jpg") # Загружаем логотип  
pygame.display.set\_icon(icon) # Выставляем логотип  
  
video\_clip = VideoFileClip("заставка.mp4") # Загрузка видезаставки  
video\_length = video\_clip.duration # продолжительность видео в секундах  
  
pygame.mixer.init() # инициализируем функцию добавления музыки  
  
# константы для отслеживания текущего окна  
MENU\_WINDOW = 0  
FIGHT\_WINDOW = 1  
  
# Линия земли, на которой стоят персонажи  
ground = int(0.94 \* user\_screen\_height)  
  
# Определяющие положение персонажа переменные  
x = 0.08 \* user\_screen\_width  
throw\_back\_power = 40  
jump\_power = 20  
  
# Определяющие положение второго персонажа переменные  
x2 = 0.8 \* user\_screen\_width  
throw\_back\_power2 = 40  
jump\_power2 = 20  
  
# Значения хэлф баров  
health\_1 = 100  
health\_2 = 100  
  
ch\_win\_flag = False # Флаг, означсающий что один из игроков победил  
pause\_flag = False # Флаг, означающий что игра находится на паузе  
  
health = Healthbars(user\_screen\_width**,** user\_screen\_height) # Объявляем класс хэлфбаров  
  
  
# Функция для воспроизведения видеозаставки  
async def play\_video(clip): # Создаём функцию для проигрыша видео, она выполняется асинхронно  
 start\_time = pygame.time.get\_ticks()  
 clip.audio.write\_audiofile("temp\_audio.wav")  
 pygame.mixer.music.load('temp\_audio.wav')  
 pygame.mixer.music.play()  
  
 while pygame.mixer.music.get\_busy():  
 elapsed = (pygame.time.get\_ticks() - start\_time) / 1000.0  
 if elapsed > video\_length:  
 break  
 frame = clip.get\_frame(elapsed)  
 surf = pygame.surfarray.make\_surface(frame.swapaxes(0**,** 1))  
 screen.blit(surf**,** (0**,** 0))  
 pygame.display.flip()  
 pygame.time.wait(int(1000 / clip.fps))  
 await asyncio.sleep(0) # Позволяет другим задачам выполниться  
  
 pygame.mixer.music.stop()  
 pygame.mixer.music.unload()  
 if os.path.exists('temp\_audio.wav'):  
 os.remove('temp\_audio.wav')  
  
  
# Создание асинхронной функции для вызова воспроизведения заставки  
async def main():  
 video\_clip = VideoFileClip("заставка.mp4") # Загрузка видеоклипа  
 await play\_video(video\_clip) # Вызов функции воспроизведения заставки  
  
# Создание и запуск цикла событий asyncio  
loop = asyncio.get\_event\_loop()  
loop.run\_until\_complete(main())  
  
pygame.mixer.music.load("menu\_music.mp3") # Загружаем музыку  
pygame.mixer.music.set\_volume(0.2) # Выставляем громкость  
pygame.mixer.music.play(-1) # Запускаем бесконечный цикл проигрывания  
  
sound = pygame.mixer.Sound("turn.mp3") # звук нажатия на кнопку  
sound\_2 = pygame.mixer.Sound("slap.mp3") # звук попадания по персонажам  
  
  
def key\_check(): # Проверка нажатий  
 *'''  
 Функция для проверки нажатий на клавиши и их передачи обработчикам класса Hero  
 :return:  
 '''* keys = pygame.key.get\_pressed() # Обработчик нажатия клавиш  
 # списки с флагами, которые передадим в классы персонажей  
 res1 = []  
 res2 = []  
 # в зависимости от нажатых клавиш, добавляем в списки нужные флаги  
 if not ch\_win\_flag and not pause\_flag:  
 # для первого персонажа  
 if keys[pygame.K\_a]:  
 res1.append(LEFT)  
 res1.append(RUN)  
 if keys[pygame.K\_d]:  
 res1.append(RIGHT)  
 res1.append(RUN)  
 if keys[pygame.K\_w]:  
 res1.append(JUMP)  
 if keys[pygame.K\_s]:  
 res1.append(SQUAT)  
 if keys[pygame.K\_f]:  
 res1.append(FIGHT)  
  
 # для второго персонажа  
 if keys[pygame.K\_j]:  
 res2.append(LEFT)  
 res2.append(RUN)  
 if keys[pygame.K\_l]:  
 res2.append(RIGHT)  
 res2.append(RUN)  
 if keys[pygame.K\_i]:  
 res2.append(JUMP)  
 if keys[pygame.K\_k]:  
 res2.append(SQUAT)  
 if keys[pygame.K\_h]:  
 res2.append(FIGHT)  
  
 # отправляем флаги в обработчик событий персонажа  
 hero1.process\_events(res1)  
 hero2.process\_events(res2)  
  
 # вызываем метод для обработки передвижений, ударов и прыжков персонажа  
 hero1.move()  
 hero2.move()  
  
  
animation\_delay = 100 # время между обновлениями анимаций героев  
UPDATE\_FRAMES = pygame.USEREVENT + 1 # создаём событие для обновления кадров и присваиваем ему номер  
pygame.time.set\_timer(UPDATE\_FRAMES**,** animation\_delay)  
  
health.health\_on\_all\_arenas(arens) # рисует хелф бары на всех аренах для отображения в меню  
  
flag = MENU\_WINDOW # флаг, показывающий активное окно  
  
running = True # флаг работы  
sound\_flag = True # флаг нынешнего состояния звука  
  
control\_button\_pressed = False # открыта ли помощь по управлению  
  
clock = pygame.time.Clock() # создаём объект часов  
  
while running:  
 clock.tick(60) # обновление экрана 60 раз в секунду  
  
 if flag == MENU\_WINDOW:  
 # отрисовка всех компонентов для меню  
 arena = pygame.transform.scale(arens[arenas\_count]**,** (0.25 \* user\_screen\_width**,** 0.3 \* user\_screen\_height))  
 character\_choice = pygame.transform.scale(characters[character1\_count]**,** (0.09 \* user\_screen\_width**,** 0.3 \* user\_screen\_height))  
 character2\_choice = pygame.transform.scale(characters[character2\_count]**,** (0.09 \* user\_screen\_width**,** 0.3 \* user\_screen\_height))  
 screen.fill((192**,** 6**,** 13))  
 screen.blit(text\_surface**,** ((user\_screen\_width - text\_surface.get\_width()) / 2**,** 0.04 \* user\_screen\_height))  
 pygame.draw.rect(screen**,** (170**,** 0**,** 0)**,** play\_button)  
 screen.blit(play\_text**,** play\_text\_rect)  
 pygame.draw.rect(screen**,** (170**,** 0**,** 0)**,** exit\_button)  
 screen.blit(exit\_text**,** exit\_text\_rect)  
 pygame.draw.rect(screen**,** (170**,** 0**,** 0)**,** aren\_window)  
 screen.blit(arena**,** arena\_rect)  
 screen.blit(arena\_text**,** (0.2 \* user\_screen\_width**,** 0.46 \* user\_screen\_height))  
 screen.blit(left\_strelka**,** left\_strelka\_rect)  
 screen.blit(right\_strelka**,** right\_strelka\_rect)  
 screen.blit(character\_choice**,** character\_choice\_rect)  
 screen.blit(left\_strelka\_ch1**,** left\_strelka\_ch1\_rect)  
 screen.blit(right\_strelka\_ch1**,** right\_strelka\_ch1\_rect)  
 screen.blit(character2\_choice**,** character2\_choice\_rect)  
 screen.blit(left\_strelka\_ch2**,** left\_strelka\_ch2\_rect)  
 screen.blit(right\_strelka\_ch2**,** right\_strelka\_ch2\_rect)  
 screen.blit(ch1\_choice\_text**,** (0.6 \* user\_screen\_width**,** 0.54 \* user\_screen\_height))  
 screen.blit(ch2\_choice\_text**,** (0.8 \* user\_screen\_width**,** 0.54 \* user\_screen\_height))  
 screen.blit(control\_image**,** control\_rect)  
 # теперь компоненты, зависящие от флагов  
 if sound\_flag:  
 screen.blit(sound\_on**,** sound\_on\_rect)  
 else:  
 screen.blit(sound\_off**,** sound\_off\_rect)  
  
 # если нажали кнопку с помощью с управлением, отрисовываем окно с показом управления  
 if control\_button\_pressed:  
 screen.blit(control\_layout**,** control\_layout\_rect)  
 screen.blit(back\_image\_from\_control**,** back\_image\_rect\_from\_control)  
  
 elif flag == FIGHT\_WINDOW:  
 # отрисовка элементов в окне боя  
 screen.blit(arens[arenas\_count]**,** (0**,** 0)) # отрисовываем фон  
 health.draw(screen**,** health\_dict[hero1]**,** health\_dict[hero2])  
 pygame.draw.rect(screen**,** (170**,** 0**,** 0)**,** back\_button)  
 screen.blit(back\_image**,** back\_image\_rect)  
 screen.blit(ch1\_text**,** (0.02 \* user\_screen\_width**,** 0.05 \* user\_screen\_height))  
 screen.blit(ch2\_text**,** (0.8 \* user\_screen\_width**,** 0.05 \* user\_screen\_height))  
 if not pause\_flag:  
 screen.blit(pause\_image**,** pause\_image\_rect)  
 else:  
 screen.blit(play\_image**,** play\_image\_rect)  
 key\_check() # вызываем проверку нажатий  
  
 heroes.draw(screen) # отрисовываем персонажей  
  
 # проверка поражения персонажей  
 if health\_dict[hero1] <= 0:  
 screen.blit(ch2\_win\_text**,** (0.25 \* user\_screen\_width**,** 0.46 \* user\_screen\_height))  
 ch\_win\_flag = True  
 pygame.draw.rect(screen**,** (170**,** 0**,** 0)**,** restart\_button)  
 screen.blit(restart\_image**,** restart\_image\_rect)  
 # запускаем анимацию смерти  
 hero1.process\_events([DEAD])  
  
 elif health\_dict[hero2] <= 0:  
 screen.blit(ch1\_win\_text**,** (0.25 \* user\_screen\_width**,** 0.46 \* user\_screen\_height))  
 ch\_win\_flag = True  
 pygame.draw.rect(screen**,** (170**,** 0**,** 0)**,** restart\_button)  
 screen.blit(restart\_image**,** restart\_image\_rect)  
  
 # запускаем анимацию смерти  
 hero2.process\_events([DEAD])  
  
 pygame.display.update() # обновляем окно  
  
 # Скрипт выхода из игры  
 for event in pygame.event.get():  
 if event.type == pygame.MOUSEBUTTONDOWN:  
 # далее проверки на нажатия кнопок  
 if play\_button.collidepoint(event.pos):  
 sound.play()  
 flag = FIGHT\_WINDOW  
 pygame.mixer.music.load("music.mp3") # Загружаем музыку  
 if sound\_flag:  
 pygame.mixer.music.set\_volume(0.2)  
 else:  
 pygame.mixer.music.set\_volume(0)  
 pygame.mixer.music.play(-1) # Запускаем бесконечный цикл проигрывания  
  
 # создаем персонажей  
 heroes = pygame.sprite.Group()  
  
 hero1 = Hero(x**,** ground**,** throw\_back\_power**,** jump\_power**,** 1000**,** heroes**,** direction=RIGHT**,** character=ALL\_CHARACTERS[character1\_count])  
 hero2 = Hero(x2**,** ground**,** throw\_back\_power2**,** jump\_power2**,** 1000**,** heroes**,** direction=LEFT**,** character=ALL\_CHARACTERS[character2\_count])  
  
 health\_dict = {hero1: health\_1**,** hero2: health\_2}  
  
 hero1.set\_enemy(hero2**,** health\_dict)  
 hero2.set\_enemy(hero1**,** health\_dict)  
  
 # при нажатии на стрелки изменяем счетчики выбора арены и персонажа соответственно  
 elif left\_strelka\_rect.collidepoint(event.pos):  
 sound.play()  
 arenas\_count -= 1  
 if arenas\_count < 0:  
 arenas\_count = len(arens) - 1  
 elif right\_strelka\_rect.collidepoint(event.pos):  
 sound.play()  
 arenas\_count += 1  
 if arenas\_count >= len(arens):  
 arenas\_count = 0  
 elif left\_strelka\_ch1\_rect.collidepoint(event.pos):  
 sound.play()  
 character1\_count -= 1  
 if character1\_count < 0:  
 character1\_count = len(characters) - 1  
 elif right\_strelka\_ch1\_rect.collidepoint(event.pos):  
 sound.play()  
 character1\_count += 1  
 if character1\_count >= len(characters):  
 character1\_count = 0  
 elif left\_strelka\_ch2\_rect.collidepoint(event.pos):  
 sound.play()  
 character2\_count -= 1  
 if character2\_count < 0:  
 character2\_count = len(characters) - 1  
 elif right\_strelka\_ch2\_rect.collidepoint(event.pos):  
 sound.play()  
 character2\_count += 1  
 if character2\_count >= len(characters):  
 character2\_count = 0  
 elif sound\_on\_rect.collidepoint(event.pos) or sound\_off\_rect.collidepoint(event.pos):  
 sound.play()  
 sound\_flag = not sound\_flag # Переключаем звук на противоположное состояние  
 if sound\_flag:  
 pygame.mixer.music.set\_volume(0.2)  
 else:  
 pygame.mixer.music.set\_volume(0)  
 elif back\_image\_rect\_from\_control.collidepoint(event.pos):  
 control\_button\_pressed = False  
 sound.play()  
 elif restart\_button.collidepoint(event.pos):  
 sound.play()  
 # сбрасываем параметры  
 ch\_win\_flag = False  
 x = 0.08 \* user\_screen\_width  
  
 x2 = 0.8 \* user\_screen\_width  
  
 # создаем персонажей  
 heroes = pygame.sprite.Group()  
  
 hero1 = Hero(x**,** ground**,** throw\_back\_power**,** jump\_power**,** 1000**,** heroes**,** direction=RIGHT**,** character=ALL\_CHARACTERS[character1\_count])  
 hero2 = Hero(x2**,** ground**,** throw\_back\_power2**,** jump\_power2**,** 1000**,** heroes**,** direction=LEFT**,** character=ALL\_CHARACTERS[character2\_count])  
  
 health\_dict = {hero1: health\_1**,** hero2: health\_2} # пересоздаём словарь со здоровьем  
  
 hero1.set\_enemy(hero2**,** health\_dict)  
 hero2.set\_enemy(hero1**,** health\_dict)  
 elif control\_rect.collidepoint(event.pos):  
 control\_button\_pressed = True  
 sound.play()  
 elif pause\_image\_rect.collidepoint(event.pos) or play\_image\_rect.collidepoint(event.pos):  
 sound.play()  
 pause\_flag = not pause\_flag  
 elif back\_button.collidepoint(event.pos):  
 pause\_flag = False  
 sound.play()  
 pygame.mixer.music.load("menu\_music.mp3") # Загружаем музыку  
 if sound\_flag:  
 pygame.mixer.music.set\_volume(0.2)  
 else:  
 pygame.mixer.music.set\_volume(0)  
 pygame.mixer.music.play(-1) # Запускаем бесконечный цикл проигрывания  
 flag = MENU\_WINDOW  
 ch\_win\_flag = False  
  
 x = 0.08 \* user\_screen\_width  
  
 x2 = 0.8 \* user\_screen\_width  
  
 # создаем персонажей  
 heroes = pygame.sprite.Group()  
  
 hero1 = Hero(x**,** ground**,** throw\_back\_power**,** jump\_power**,** 1000**,** heroes**,** direction=RIGHT**,** character=ALL\_CHARACTERS[character1\_count])  
 hero2 = Hero(x2**,** ground**,** throw\_back\_power2**,** jump\_power2**,** 1000**,** heroes**,** direction=LEFT**,** character=ALL\_CHARACTERS[character2\_count])  
  
 health\_dict = {hero1: health\_1**,** hero2: health\_2} # пересоздаём словарь со здоровьем  
  
 hero1.set\_enemy(hero2**,** health\_dict)  
 hero2.set\_enemy(hero1**,** health\_dict)  
  
 elif exit\_button.collidepoint(event.pos):  
 running = False  
  
 if event.type == UPDATE\_FRAMES and flag == FIGHT\_WINDOW: # обновление кадров анимации персонажей  
 heroes.update()  
  
 if event.type == pygame.QUIT: # выход из игры  
 running = False  
 pygame.quit()

Теперь перейдём к файлу buttons\_and\_texts

import pygame # Импортируем pygame  
# Импортируем нужное из других файлов  
from arens import arens**,** arenas\_count  
from characters import characters**,** character1\_count**,** character2\_count  
  
'''  
Файл, содержащий создание кнопок, текстов и изображений на кнопках  
'''  
  
# Собираем информацию о разрешении экрана пользователя  
user\_screen\_info = pygame.display.Info()  
user\_screen\_width = user\_screen\_info.current\_w  
user\_screen\_height = user\_screen\_info.current\_h  
  
# Текст - заголовок игры на входном экране  
game\_entery\_name = pygame.font.Font("Fonts/unispace bd.ttf"**,** int(user\_screen\_height \* 0.16))  
text\_surface = game\_entery\_name.render("MORTAL FIGHT"**,** True**,** (255**,** 107**,** 107))  
  
# Кнопка начала игры  
play\_button = pygame.Rect(0.78 \* user\_screen\_width / 2**,** 0.21 \* user\_screen\_height**,** 0.22 \* user\_screen\_width**,** 0.12 \* user\_screen\_height)  
play\_text\_font = pygame.font.Font("Fonts/unispace bd.ttf"**,** int(user\_screen\_height \* 0.12))  
play\_text = play\_text\_font.render("PLAY"**,** True**,** (255**,** 107**,** 107))  
play\_text\_rect = play\_text.get\_rect(center=play\_button.center)  
  
# Кнопка выхода из игры  
exit\_button = pygame.Rect(0.78 \* user\_screen\_width / 2**,** 0.35 \* user\_screen\_height**,** 0.22 \* user\_screen\_width**,** 0.12 \* user\_screen\_height)  
exit\_text\_font = pygame.font.Font("Fonts/unispace bd.ttf"**,** int(user\_screen\_height \* 0.12))  
exit\_text = exit\_text\_font.render("EXIT"**,** True**,** (255**,** 107**,** 107))  
exit\_text\_rect = exit\_text.get\_rect(center=exit\_button.center)  
  
# Кнопка возврата в меню игры  
back\_button = pygame.Rect(0.96 \* user\_screen\_width / 2**,** 0.02 \* user\_screen\_height**,** 0.04 \* user\_screen\_width**,** 0.05 \* user\_screen\_height)  
back\_image = pygame.transform.scale(pygame.image.load("krest.png").convert\_alpha()**,** (0.03 \* user\_screen\_width**,** 0.04 \* user\_screen\_height))  
back\_image\_rect = back\_image.get\_rect(center=back\_button.center)  
  
# Заголовок окна выбора арены  
arena\_text\_font = pygame.font.Font("Fonts/unispace bd.ttf"**,** int(user\_screen\_height \* 0.12))  
arena\_text = arena\_text\_font.render("ARENA"**,** True**,** (255**,** 107**,** 107))  
  
# Окно выбора арены  
aren\_window = pygame.Rect(0.75 \* user\_screen\_width / 4.5**,** 0.6 \* user\_screen\_height**,** 0.25 \* user\_screen\_width**,** 0.3 \* user\_screen\_height)  
arena = pygame.transform.scale(arens[arenas\_count]**,** aren\_window.size)  
arena\_rect = arena.get\_rect(center=aren\_window.center)  
  
# Стрелки выбора арены  
left\_strelka = pygame.transform.scale(pygame.image.load("left\_strelka.png")**,** (0.05 \* user\_screen\_width**,** 0.07 \* user\_screen\_height))  
left\_strelka\_rect = left\_strelka.get\_rect(topleft=(0.1 \* user\_screen\_width**,** 0.69 \* user\_screen\_height))  
right\_strelka = pygame.transform.scale(pygame.image.load("right\_strelka.png")**,** (0.05 \* user\_screen\_width**,** 0.07 \* user\_screen\_height))  
right\_strelka\_rect = right\_strelka.get\_rect(topleft=(0.43 \* user\_screen\_width**,** 0.69 \* user\_screen\_height))  
  
# Иконка включенного звука  
sound\_on = pygame.transform.scale(pygame.image.load("soundon.png")**,** (0.07 \* user\_screen\_width**,** 0.07 \* user\_screen\_height))  
sound\_on\_rect = sound\_on.get\_rect(topleft=(0.86 \* user\_screen\_width**,** 0.09 \* user\_screen\_height))  
  
# Иконка выключенного звука  
sound\_off = pygame.transform.scale(pygame.image.load("soundoff.png")**,** (0.07 \* user\_screen\_width**,** 0.07 \* user\_screen\_height))  
sound\_off\_rect = sound\_off.get\_rect(topleft=(0.86 \* user\_screen\_width**,** 0.09 \* user\_screen\_height))  
  
# Окно выбора первого персонажа  
character\_choice\_window = pygame.Rect(0.75 \* user\_screen\_width / 1.3**,** 0.6 \* user\_screen\_height**,** 0.1 \* user\_screen\_width**,** 0.3 \* user\_screen\_height)  
character\_choice = pygame.transform.scale(characters[character1\_count]**,** character\_choice\_window.size)  
character\_choice\_rect = character\_choice.get\_rect(center=character\_choice\_window.center)  
  
# Стрелки для выбора первого персонажа  
left\_strelka\_ch1 = pygame.transform.scale(pygame.image.load("left\_strelka.png")**,** (0.02 \* user\_screen\_width**,** 0.07 \* user\_screen\_height))  
left\_strelka\_ch1\_rect = left\_strelka\_ch1.get\_rect(topleft=(0.55 \* user\_screen\_width**,** 0.69 \* user\_screen\_height))  
right\_strelka\_ch1 = pygame.transform.scale(pygame.image.load("right\_strelka.png")**,** (0.02 \* user\_screen\_width**,** 0.07 \* user\_screen\_height))  
right\_strelka\_ch1\_rect = right\_strelka\_ch1.get\_rect(topleft=(0.68 \* user\_screen\_width**,** 0.69 \* user\_screen\_height))  
  
# Окно выбора второго персонажа  
character2\_choice\_window = pygame.Rect(0.77 \* user\_screen\_width**,** 0.6 \* user\_screen\_height**,** 0.1 \* user\_screen\_width**,** 0.3 \* user\_screen\_height)  
character2\_choice = pygame.transform.scale(characters[character2\_count]**,** character2\_choice\_window.size)  
character2\_choice\_rect = character2\_choice.get\_rect(center=character2\_choice\_window.center)  
  
# Стрелки для выбора вторго персонажа  
left\_strelka\_ch2 = pygame.transform.scale(pygame.image.load("left\_strelka.png")**,** (0.02 \* user\_screen\_width**,** 0.07 \* user\_screen\_height))  
left\_strelka\_ch2\_rect = left\_strelka\_ch2.get\_rect(topleft=(0.75 \* user\_screen\_width**,** 0.69 \* user\_screen\_height))  
right\_strelka\_ch2 = pygame.transform.scale(pygame.image.load("right\_strelka.png")**,** (0.02 \* user\_screen\_width**,** 0.07 \* user\_screen\_height))  
right\_strelka\_ch2\_rect = right\_strelka\_ch2.get\_rect(topleft=(0.87 \* user\_screen\_width**,** 0.69 \* user\_screen\_height))  
  
# Сообщения о выйгрыше игроков  
ch1\_win\_text\_font = pygame.font.Font("Fonts/unispace bd.ttf"**,** int(user\_screen\_height \* 0.12))  
ch1\_win\_text = ch1\_win\_text\_font.render("PLAYER 1 WIN"**,** True**,** (255**,** 107**,** 107))  
ch2\_win\_text\_font = pygame.font.Font("Fonts/unispace bd.ttf"**,** int(user\_screen\_height \* 0.12))  
ch2\_win\_text = ch1\_win\_text\_font.render("PLAYER 2 WIN"**,** True**,** (255**,** 107**,** 107))  
  
# Кнопка начала новой игры  
restart\_button = pygame.Rect(0.96 \* user\_screen\_width / 2**,** 0.62 \* user\_screen\_height**,** 0.07 \* user\_screen\_width**,** 0.14 \* user\_screen\_height)  
restart\_image = pygame.transform.scale(pygame.image.load("restart.png").convert\_alpha()**,** (0.07 \* user\_screen\_width**,** 0.14 \* user\_screen\_height))  
restart\_image\_rect = restart\_image.get\_rect(center=restart\_button.center)  
  
# Надписи обозначающие игроков под хэлфбарами  
ch1\_font = pygame.font.Font("Fonts/unispace bd.ttf"**,** int(user\_screen\_height \* 0.04))  
ch1\_text = ch1\_font.render("PLAYER 1"**,** True**,** (255**,** 107**,** 107))  
ch2\_font = pygame.font.Font("Fonts/unispace bd.ttf"**,** int(user\_screen\_height \* 0.04))  
ch2\_text = ch1\_font.render("PLAYER 2"**,** True**,** (255**,** 107**,** 107))  
  
# Первый персонаж выбор  
ch1\_choice\_text\_font = pygame.font.Font("Fonts/unispace bd.ttf"**,** int(user\_screen\_height \* 0.04))  
ch1\_choice\_text = ch1\_choice\_text\_font.render("P1"**,** True**,** (255**,** 107**,** 107))  
  
# Второй персонаж выбор  
ch2\_choice\_text\_font = pygame.font.Font("Fonts/unispace bd.ttf"**,** int(user\_screen\_height \* 0.04))  
ch2\_choice\_text = ch2\_choice\_text\_font.render("P2"**,** True**,** (255**,** 107**,** 107))  
  
# Кнопка открытия раскладки клавиатуры  
control\_image = pygame.transform.scale(pygame.image.load("control\_button.png")**,** (0.07 \* user\_screen\_width**,** 0.07 \* user\_screen\_height))  
control\_rect = control\_image.get\_rect(topleft=(0.06 \* user\_screen\_width**,** 0.09 \* user\_screen\_height))  
  
# Окно раскладки клавиатуры  
control\_layout = pygame.transform.scale(pygame.image.load("conrol\_layout.png")**,** (user\_screen\_width**,** user\_screen\_height))  
control\_layout\_rect = control\_layout.get\_rect(topleft=(0.01 \* user\_screen\_width**,** 0.01 \* user\_screen\_height))  
  
# кнопка возврата из окна с показом управления  
back\_button\_from\_control = pygame.Rect(0.96 \* user\_screen\_width / 2**,** 0.02 \* user\_screen\_height**,** 0.25 \* user\_screen\_width**,** 0.06 \* user\_screen\_height)  
back\_image\_from\_control = pygame.transform.scale(pygame.image.load("krest.png").convert\_alpha()**,** (0.05 \* user\_screen\_width**,** 0.06 \* user\_screen\_height))  
back\_image\_rect\_from\_control = back\_image\_from\_control.get\_rect(center=back\_button\_from\_control.center)  
  
# Кнопка паузы  
pause\_image = pygame.transform.scale(pygame.image.load("pause.png").convert\_alpha()**,** (0.03 \* user\_screen\_width**,** 0.04 \* user\_screen\_height))  
pause\_image\_rect = pause\_image.get\_rect(topleft=(0.53 \* user\_screen\_width**,** 0.03 \* user\_screen\_height))  
  
# Кнопка продолжения игры  
play\_image = pygame.transform.scale(pygame.image.load("play.png").convert\_alpha()**,** (0.03 \* user\_screen\_width**,** 0.04 \* user\_screen\_height))  
play\_image\_rect = play\_image.get\_rect(topleft=(0.53 \* user\_screen\_width**,** 0.03 \* user\_screen\_height))

Теперь перейдём к файлу animations

import pygame # Импортируем pygame  
  
  
'''  
Файл содержит все анимации персонажей  
'''  
  
# Анимации деда Максима в момент неподвижности вправо  
ded\_maxim\_stay = [pygame.image.load("Character\_st/1st.png")**,** pygame.image.load("Character\_st/2st.png")**,** pygame.image.load("Character\_st/3st.png")**,** pygame.image.load("Character\_st/4st.png")**,** pygame.image.load("Character\_st/5st.png")**,** pygame.image.load("Character\_st/6st.png")**,** pygame.image.load("Character\_st/7st.png")]  
  
# Анимации деда Максима во время атаки  
ded\_maxim\_fight = [pygame.image.load("Character\_fight/Attack\_1.png")**,** pygame.image.load("Character\_fight/Attack\_2.png")**,** pygame.image.load("Character\_fight/Attack\_3.png")**,** pygame.image.load("Character\_fight/Attack\_4.png")**,** pygame.image.load("Character\_fight/Attack\_5.png")]  
  
# Анимации деда Максима во время бега  
ded\_maxim\_run = [pygame.image.load("charact\_run/run1.png")**,** pygame.image.load("charact\_run/run2.png")**,** pygame.image.load("charact\_run/run3.png")**,** pygame.image.load("charact\_run/run4.png")**,** pygame.image.load("charact\_run/run5.png")**,** pygame.image.load("charact\_run/run6.png")**,** pygame.image.load("charact\_run/run7.png")**,** pygame.image.load("charact\_run/run8.png")]  
  
# Анимации деда Максима во время прыжка  
ded\_maxim\_jump = [pygame.image.load("Character\_jump/Jump\_3.png")**,** pygame.image.load("Character\_jump/Jump\_4.png")**,** pygame.image.load("Character\_jump/Jump\_5.png")**,** pygame.image.load("Character\_jump/Jump\_6.png")**,** pygame.image.load("Character\_jump/Jump\_7.png")**,** pygame.image.load("Character\_jump/Jump\_8.png")]  
  
# Анимации деда Максима во время приседа  
ded\_maxim\_squat = [pygame.image.load('Character\_squat/Squat\_1.png')**,** pygame.image.load('Character\_squat/Squat\_2.png')**,** pygame.image.load('Character\_squat/Squat\_3.png')**,** pygame.image.load('Character\_squat/Squat\_3.png')**,** pygame.image.load('Character\_squat/Squat\_2.png')**,** pygame.image.load('Character\_squat/Squat\_1.png')]  
  
# Анимации деда Максима во время смерти  
ded\_maxim\_dead = [pygame.image.load("Character\_dead/Dead\_1.png")**,** pygame.image.load("Character\_dead/Dead\_2.png")**,** pygame.image.load("Character\_dead/Dead\_3.png")**,** pygame.image.load("Character\_dead/Dead\_4.png")**,** pygame.image.load("Character\_dead/Dead\_5.png")]  
  
# Анимации Вурдалака во время неподвижности вправо  
vurdalak\_stay = [pygame.image.load("Vurdalak\_st/St\_1.png")**,** pygame.image.load("Vurdalak\_st/St\_2.png")**,** pygame.image.load("Vurdalak\_st/St\_3.png")**,** pygame.image.load("Vurdalak\_st/St\_4.png")**,** pygame.image.load("Vurdalak\_st/St\_5.png")**,** pygame.image.load("Vurdalak\_st/St\_6.png")**,** pygame.image.load("Vurdalak\_st/St\_7.png")**,** pygame.image.load("Vurdalak\_st/St\_8.png")**,** pygame.image.load("Vurdalak\_st/St\_9.png")**,** pygame.image.load("Vurdalak\_st/St\_10.png")**,** pygame.image.load("Vurdalak\_st/St\_11.png")**,** pygame.image.load("Vurdalak\_st/St\_12.png")]  
  
# Анимации Вурдалака во время бега  
vurdalak\_run = [pygame.image.load("Vurdalak\_run/Run\_1.png")**,** pygame.image.load("Vurdalak\_run/Run\_2.png")**,** pygame.image.load("Vurdalak\_run/Run\_3.png")**,** pygame.image.load("Vurdalak\_run/Run\_4.png")]  
  
# Анимации Вурдалака во время удара  
vurdalak\_fight = [pygame.image.load("Vurdalak\_fight/Ft\_1.png")**,** pygame.image.load("Vurdalak\_fight/Ft\_2.png")**,** pygame.image.load("Vurdalak\_fight/Ft\_3.png")**,** pygame.image.load("Vurdalak\_fight/Ft\_4.png")**,** pygame.image.load("Vurdalak\_fight/Ft\_5.png")]  
  
# Анимации Вурдалака во время прыжка  
vurdalak\_jump = [pygame.image.load("Vurdalak\_jump/Jump1.png")**,** pygame.image.load("Vurdalak\_jump/Jump2.png")**,** pygame.image.load("Vurdalak\_jump/Jump3.png")**,** pygame.image.load("Vurdalak\_jump/Jump4.png")]  
  
# Анимации Вурдалака во время приседания  
vurdalak\_squat = [pygame.image.load("Vurdalak\_squat/Squat\_1.png")**,** pygame.image.load("Vurdalak\_squat/Squat\_2.png")**,** pygame.image.load("Vurdalak\_squat/Squat\_3.png")]  
  
# Анимации Вурдалака во время смерти  
vurdalak\_dead = [pygame.image.load("Vurdalak\_dead/Dead\_1.png")**,** pygame.image.load("Vurdalak\_dead/Dead\_2.png")**,** pygame.image.load("Vurdalak\_dead/Dead\_3.png")**,** pygame.image.load("Vurdalak\_dead/Dead\_4.png")**,** pygame.image.load("Vurdalak\_dead/Dead\_5.png")**,** pygame.image.load("Vurdalak\_dead/Dead\_6.png")]

Теперь перейдём к файлу arens

import pygame # Импортируем pygame  
  
# Собираем информацию о разрешении экрана пользователя  
user\_screen\_info = pygame.display.Info()  
user\_screen\_width = user\_screen\_info.current\_w  
user\_screen\_height = user\_screen\_info.current\_h  
  
arenas\_count = 2 # Счётчик выбранной арены  
  
# Загружаем изображения арен  
arens = [pygame.transform.scale(pygame.image.load("arenas/location.jpg")**,** (user\_screen\_width**,** user\_screen\_height))**,** pygame.transform.scale(pygame.image.load("arenas/location2.jpg")**,** (user\_screen\_width**,** user\_screen\_height))**,** pygame.transform.scale(pygame.image.load("arenas/location1.jpg")**,** (user\_screen\_width**,** user\_screen\_height))**,** pygame.transform.scale(pygame.image.load("arenas/location3.jpg")**,** (user\_screen\_width**,** user\_screen\_height))**,** pygame.transform.scale(pygame.image.load("arenas/location4.jpg")**,** (user\_screen\_width**,** user\_screen\_height))]

Теперь перейдём к файлу characters

import pygame # Импортируем pygame  
  
# Собираем информацию о разрешении экрана пользователя  
user\_screen\_info = pygame.display.Info()  
user\_screen\_width = user\_screen\_info.current\_w  
user\_screen\_height = user\_screen\_info.current\_h  
  
character1\_count = 0 # Счётчик выбранного первого персонажа  
character2\_count = 1 # Счётчик выбранного второго персонажа  
  
# Загружаем изображения персонажей, которые будут отображаться при их выборе  
characters = [pygame.transform.scale(pygame.image.load("Characters\_images/Ded\_Maxim.png")**,** (user\_screen\_width**,** user\_screen\_height))**,** pygame.transform.scale(pygame.image.load("Characters\_images/Vurdalak.png")**,** (user\_screen\_width**,** user\_screen\_height))]

Теперь перейдём к файлу constants\_for\_hero

*'''  
В файле собраны константы для управления персонажами  
'''*RUN = 0 # Обозначает, что персонаж бежит  
JUMP = 1 # Обозначает, что персонаж прыгает  
FIGHT = 2 # Обозначает, что персонаж совершает удар  
SQUAT = 3 # Обозначает, что персонаж приседает  
DEAD = 8 # Обозначает, что персонаж погиб  
  
LEFT = 4 # Обозначает, что персонаж смотрит влево  
RIGHT = 5 # Обозначает, что персонаж смотрит вправо  
  
DED\_MAXIM = 6 # Обозначает, что выбран персонаж дед Максим  
VURDALAK = 7 # Обозначает, что выбран персонаж Вурдалак  
  
ALL\_CHARACTERS = [DED\_MAXIM**,** VURDALAK] # Список персонажей

Теперь перейдём к файлу healthbars

import pygame # Импортируем pygame  
  
  
# Создаём класс полосок здоровья  
class Healthbars():  
 def \_\_init\_\_(self**,** screen\_width**,** screen\_height): # Инициализируем класс  
 self.screen\_width = screen\_width  
 self.screen\_height = screen\_height  
  
 def health\_on\_all\_arenas(self**,** arenas):  
 for arena in arenas:  
 # Рисуем первый хэлфбар  
 pygame.draw.rect(arena**,** (255**,** 255**,** 255)**,** # цвет  
 (0.02 \* self.screen\_width - 2**,** 0.02 \* self.screen\_height - 2**,** # положение  
 100 \* 3 + 4**,** 44)) # размер  
 pygame.draw.rect(arena**,** (0**,** 255**,** 0)**,** (0.02 \* self.screen\_width**,** 0.02 \* self.screen\_height**,** 100 \* 3**,** 40))  
  
 # аналогично для второго игрока  
 pygame.draw.rect(arena**,** (255**,** 255**,** 255)**,** (0.8 \* self.screen\_width - 2**,** 0.02 \* self.screen\_height - 2**,** 100 \* 3 + 4**,** 44))  
 pygame.draw.rect(arena**,** (0**,** 255**,** 0)**,** (0.8 \* self.screen\_width**,** 0.02 \* self.screen\_height**,** 100 \* 3**,** 40))  
  
 def draw(self**,** screen**,** current\_health\_1**,** current\_health\_2): # Отрисовка хэлф баров во время игры  
 pygame.draw.rect(screen**,** (255**,** 255**,** 255)**,** (0.02 \* self.screen\_width - 2**,** 0.02 \* self.screen\_height - 2**,** 100 \* 3 + 4**,** 44))  
 pygame.draw.rect(screen**,** (0**,** 255**,** 0)**,** (0.02 \* self.screen\_width**,** 0.02 \* self.screen\_height**,** current\_health\_1 \* 3**,** 40))  
  
 pygame.draw.rect(screen**,** (255**,** 255**,** 255)**,** (0.8 \* self.screen\_width - 2**,** 0.02 \* self.screen\_height - 2**,** 100 \* 3 + 4**,** 44))  
 pygame.draw.rect(screen**,** (0**,** 255**,** 0)**,** (0.8 \* self.screen\_width**,** 0.02 \* self.screen\_height**,** current\_health\_2 \* 3**,** 40))

Теперь перейдём к файлу hero

import pygame  
from animations import \*  
from constants\_for\_hero import \*  
  
# Инициализация Pygame  
pygame.init()  
user\_screen\_info = pygame.display.Info()  
user\_screen\_width = user\_screen\_info.current\_w  
user\_screen\_height = user\_screen\_info.current\_h  
  
screen = pygame.display.set\_mode((user\_screen\_width**,** user\_screen\_height)) # Задаём разрешение основного окна  
  
  
class Hero(pygame.sprite.Sprite):  
 *'''  
 Класс персонаж  
 '''* def \_\_init\_\_(self**,** x**,** ground**,** throw\_back\_power**,** jump\_power**,** fight\_cool\_down**,** \*groups**,** direction=RIGHT**,** character=DED\_MAXIM):  
 *'''  
 :param x: координата х левого верхнего угла  
 :param ground: у координата земли  
 :param throw\_back\_power: сила откидывания противника  
 :param jump\_power: начальная скорость прыжка в пикселях  
 :param fight\_cool\_down: время в миллисекундах, которое игрок не сможет атаковать после использования атаки  
 :param enemy\_group: группа спрайтов противников  
 :param groups: группы спрайтов  
 :param direction: направление персонажа. Константа программы constants\_for\_hero  
 :param character: имя выбранного персонажа. Константа программы constants\_for\_hero  
 '''* super().\_\_init\_\_(\*groups)  
 # загружаем разные анимации в зависимости от выбранного персонажа  
 # для каждого состояния героя загружаем обычные,  
 # отраженные налево анимации и для каждой картинки вычисляем маску  
 if character == DED\_MAXIM:  
 speed = 0.01 \* user\_screen\_height  
 power = 30  
 self.anim\_stay = [pygame.transform.scale(el**,** (0.09 \* user\_screen\_width**,** 0.28 \* user\_screen\_height))  
 for el in ded\_maxim\_stay]  
 self.anim\_stay\_l = [pygame.transform.flip(el**,** True**,** False) for el in self.anim\_stay]  
 self.masks\_stay = [pygame.mask.from\_surface(im) for im in self.anim\_stay]  
 self.masks\_stay\_l = [pygame.mask.from\_surface(im) for im in self.anim\_stay\_l]  
  
 self.anim\_fight = [pygame.transform.scale(el**,** (0.2 \* user\_screen\_width**,** 0.28 \* user\_screen\_height))  
 for el in ded\_maxim\_fight]  
 self.anim\_fight\_l = [pygame.transform.flip(el**,** True**,** False) for el in self.anim\_fight]  
 self.masks\_fight = [pygame.mask.from\_surface(im) for im in self.anim\_fight]  
 self.masks\_fight\_l = [pygame.mask.from\_surface(im) for im in self.anim\_fight\_l]  
  
 self.anim\_run = [pygame.transform.scale(el**,** (0.15 \* user\_screen\_width**,** 0.28 \* user\_screen\_height))  
 for el in ded\_maxim\_run]  
 self.anim\_run\_l = [pygame.transform.flip(el**,** True**,** False) for el in self.anim\_run]  
 self.masks\_run = [pygame.mask.from\_surface(im) for im in self.anim\_run]  
 self.masks\_run\_l = [pygame.mask.from\_surface(im) for im in self.anim\_run\_l]  
  
 self.anim\_jump = [pygame.transform.scale(el**,** (0.22 \* user\_screen\_width**,** 0.28 \* user\_screen\_height))  
 for el in ded\_maxim\_jump]  
 self.anim\_jump\_l = [pygame.transform.flip(el**,** True**,** False) for el in self.anim\_jump]  
 self.masks\_jump = [pygame.mask.from\_surface(im) for im in self.anim\_jump]  
 self.masks\_jump\_l = [pygame.mask.from\_surface(im) for im in self.anim\_jump\_l]  
  
 self.anim\_squat = [pygame.transform.scale(el**,** (0.2 \* user\_screen\_width**,** 0.28 \* user\_screen\_height))  
 for el in ded\_maxim\_squat]  
 self.anim\_squat\_l = [pygame.transform.flip(el**,** True**,** False) for el in self.anim\_squat]  
 self.masks\_squat = [pygame.mask.from\_surface(im) for im in self.anim\_squat]  
 self.masks\_squat\_l = [pygame.mask.from\_surface(im) for im in self.anim\_squat\_l]  
  
 self.anim\_dead = [pygame.transform.scale(el**,** (0.16 \* user\_screen\_width**,** 0.28 \* user\_screen\_height))  
 for el in ded\_maxim\_dead]  
 self.anim\_dead\_l = [pygame.transform.flip(el**,** True**,** False) for el in self.anim\_dead]  
 # для смерти маски уже не нужны  
 elif character == VURDALAK:  
 speed = 0.02 \* user\_screen\_height  
 power = 15  
 self.anim\_stay = [pygame.transform.scale(el**,** (0.12 \* user\_screen\_width**,** 0.32 \* user\_screen\_height))  
 for el in vurdalak\_stay]  
 self.anim\_stay\_l = [pygame.transform.flip(el**,** True**,** False) for el in self.anim\_stay]  
 self.masks\_stay = [pygame.mask.from\_surface(im) for im in self.anim\_stay]  
 self.masks\_stay\_l = [pygame.mask.from\_surface(im) for im in self.anim\_stay\_l]  
  
 self.anim\_fight = [pygame.transform.scale(el**,** (0.12 \* user\_screen\_width**,** 0.35 \* user\_screen\_height))  
 for el in vurdalak\_fight]  
 self.anim\_fight\_l = [pygame.transform.flip(el**,** True**,** False) for el in self.anim\_fight]  
 self.masks\_fight = [pygame.mask.from\_surface(im) for im in self.anim\_fight]  
 self.masks\_fight\_l = [pygame.mask.from\_surface(im) for im in self.anim\_fight\_l]  
  
 self.anim\_run = [pygame.transform.scale(el**,** (0.14 \* user\_screen\_width**,** 0.32 \* user\_screen\_height))  
 for el in vurdalak\_run]  
 self.anim\_run\_l = [pygame.transform.flip(el**,** True**,** False) for el in self.anim\_run]  
 self.masks\_run = [pygame.mask.from\_surface(im) for im in self.anim\_run]  
 self.masks\_run\_l = [pygame.mask.from\_surface(im) for im in self.anim\_run\_l]  
  
 self.anim\_jump = [pygame.transform.scale(el**,** (0.12 \* user\_screen\_width**,** 0.35 \* user\_screen\_height))  
 for el in vurdalak\_jump]  
 self.anim\_jump\_l = [pygame.transform.flip(el**,** True**,** False) for el in self.anim\_jump]  
 self.masks\_jump = [pygame.mask.from\_surface(im) for im in self.anim\_jump]  
 self.masks\_jump\_l = [pygame.mask.from\_surface(im) for im in self.anim\_jump\_l]  
  
 self.anim\_squat = [pygame.transform.scale(el**,** (0.12 \* user\_screen\_width**,** 0.35 \* user\_screen\_height))  
 for el in vurdalak\_squat]  
 self.anim\_squat\_l = [pygame.transform.flip(el**,** True**,** False) for el in self.anim\_squat]  
 self.masks\_squat = [pygame.mask.from\_surface(im) for im in self.anim\_squat]  
 self.masks\_squat\_l = [pygame.mask.from\_surface(im) for im in self.anim\_squat\_l]  
  
 self.anim\_dead = [pygame.transform.scale(el**,** (0.15 \* user\_screen\_width**,** 0.28 \* user\_screen\_height))  
 for el in vurdalak\_dead]  
 self.anim\_dead\_l = [pygame.transform.flip(el**,** True**,** False) for el in self.anim\_dead]  
 # для смерти маски уже не нужны  
  
 self.is\_enemy\_hit = False # флаг для того, чтобы нельзя было одним ударом нанести урон несколько раз  
 self.hiten = 0 # флаг был ли ударен персонаж для кратковременной смены цвета.  
 # Численно равен количеству перекрашенных кадров анимации  
  
 # счетчики текущих кадров анимации  
 self.cur\_frame\_stay = 0  
 self.cur\_frame\_fight = 0  
 self.cur\_frame\_run = 0  
 self.cur\_frame\_jump = 0  
 self.cur\_frame\_squat = 0  
 self.cur\_frame\_dead = 0  
  
 # флаги, показывающие какое действие выполняет герой  
 self.is\_run = False  
 self.is\_fight = False  
 self.is\_jump = False  
 self.is\_squat = False  
 self.is\_dead = False  
  
 # флаги направления  
 if direction == LEFT:  
 self.left = True  
 self.right = False  
 else:  
 self.left = False  
 self.right = True  
  
 # присваиваем текущую картинку, маску и размещаем бойца  
 self.image = self.anim\_stay[self.cur\_frame\_stay] if self.right else self.anim\_stay\_l[self.cur\_frame\_stay]  
 self.mask = self.masks\_stay[self.cur\_frame\_stay] if self.right else self.masks\_stay\_l[self.cur\_frame\_stay]  
 self.rect = self.image.get\_rect()  
 self.rect.x**,** self.rect.y = x**,** ground - self.rect.height  
  
 self.ground = ground # линия земли  
  
 self.fight\_enabled = True # флаг возможности атаки  
  
 # параметры персонажа  
 self.speed = speed  
 self.power = power  
 self.jump\_power = jump\_power  
 self.throw\_back\_power = throw\_back\_power  
  
 # переменные для обработки прыжка  
 self.jump\_speed = 0 # сюда присваивается скорость, с которой падает или поднимается персонаж в данный момент  
 self.jump\_enable = True # флаг возможности прыжка  
  
 self.fight\_cool\_down = fight\_cool\_down # время, которое нельзя атаковать после использования удара  
 self.last\_fight\_time = 0 # время последней атаки  
  
 def set\_enemy(self**,** enemy**,** health\_dict):  
 *'''  
 Назначить врага персонажа  
 :param enemy: враг, объект pygame.Sprite  
 :param health\_dict: словарб, который содержит хдоровье персонажей. Ключом здоровья противника является enemy  
 :return:  
 '''* self.enemy = enemy  
 self.health\_dict = health\_dict  
  
 def hit(self**,** n\_frames):  
 *'''  
 Назначить количество кадров анимации, которое персонаж будет перекрашен  
 :param n\_frames: количество кадров  
 :return:  
 '''* self.hiten = n\_frames  
  
 def move\_back(self**,** n\_pix):  
 *'''  
 Метод откидывает персонажа  
 :param n\_pix: количество пикселей по х  
 :return:  
 '''* self.rect = self.rect.move(n\_pix**,** 0)  
  
 def frame\_swap(self):  
 *'''  
 Метод, который сменяет счетчики анимаций  
 :return:  
 '''* self.cur\_frame\_stay = (self.cur\_frame\_stay + 1) % len(self.anim\_stay)  
 self.cur\_frame\_run = (self.cur\_frame\_run + 1) % len(self.anim\_run)  
 if self.is\_fight:  
 self.cur\_frame\_fight += 1  
 if self.cur\_frame\_fight == len(self.anim\_fight):  
 self.cur\_frame\_fight = 0  
 self.is\_fight = False  
 if self.is\_jump:  
 self.cur\_frame\_jump += 1  
 if self.cur\_frame\_jump == len(self.anim\_jump):  
 self.cur\_frame\_jump = 0  
 if self.is\_squat:  
 self.cur\_frame\_squat += 1  
 if self.cur\_frame\_squat == len(self.anim\_squat):  
 self.cur\_frame\_squat = 0  
 self.is\_squat = False  
  
 if self.is\_dead:  
 if self.cur\_frame\_dead < len(self.anim\_dead) - 1:  
 self.cur\_frame\_dead += 1  
  
 def image\_swap(self):  
 *'''  
 Метод присваивает спрайту картинку и маску в зависимости от счетччика анимаций и флагов состояния персонажа  
 :return:  
 '''* # размеры предыдущей картинки для того, чтобы отцентрировать по ширине и оставить низ на той же высоте у новой  
 last\_image\_width = self.image.get\_width()  
 last\_image\_height = self.image.get\_height()  
 if self.is\_dead: # смерть - непрерываемый процесс, он перекрывает все остальные события  
 self.image = self.anim\_dead[self.cur\_frame\_dead] if self.right \  
 else self.anim\_dead\_l[self.cur\_frame\_dead]  
  
 elif self.is\_fight:  
 self.image = self.anim\_fight[self.cur\_frame\_fight] if self.right\  
 else self.anim\_fight\_l[self.cur\_frame\_fight]  
 self.mask = self.masks\_fight[self.cur\_frame\_fight] if self.right\  
 else self.masks\_fight\_l[self.cur\_frame\_fight]  
 elif self.is\_jump:  
 self.image = self.anim\_jump[self.cur\_frame\_jump] if self.right\  
 else self.anim\_jump\_l[self.cur\_frame\_jump]  
 self.mask = self.masks\_jump[self.cur\_frame\_jump] if self.right \  
 else self.masks\_jump\_l[self.cur\_frame\_jump]  
 elif self.is\_squat:  
 self.image = self.anim\_squat[self.cur\_frame\_squat] if self.right \  
 else self.anim\_squat\_l[self.cur\_frame\_squat]  
 self.mask = self.masks\_squat[self.cur\_frame\_squat] if self.right \  
 else self.masks\_squat\_l[self.cur\_frame\_squat]  
 elif self.is\_run:  
 self.image = self.anim\_run[self.cur\_frame\_run] if self.right else self.anim\_run\_l[self.cur\_frame\_run]  
 self.mask = self.masks\_run[self.cur\_frame\_run] if self.right \  
 else self.masks\_run\_l[self.cur\_frame\_run]  
 else:  
 self.image = self.anim\_stay[self.cur\_frame\_stay] if self.right else self.anim\_stay\_l[self.cur\_frame\_stay]  
 self.mask = self.masks\_stay[self.cur\_frame\_stay] if self.right \  
 else self.masks\_stay\_l[self.cur\_frame\_stay]  
  
 # если персонажа ударили, делаем картинку полупрозрачной  
 if self.hiten:  
 self.image = self.image.copy()  
 self.image.set\_alpha(200)  
 self.hiten -= 1  
  
 new\_image\_width = self.image.get\_width()  
 new\_image\_height = self.image.get\_height()  
  
 # выравниваем новый кадр анимации  
 self.rect = self.rect.move((last\_image\_width - new\_image\_width) / 2**,** 0)  
  
 def process\_events(self**,** flags):  
 *'''  
 Метод для обработки событий  
 :param flags: список из констант, содержащихся в программе constants\_for\_hero  
 :return:  
 '''* current\_time = pygame.time.get\_ticks()  
  
 # проверка на флаги состояний и примваивание нужных значений флагам  
 if DEAD in flags:  
 self.is\_dead = True  
  
 if RUN in flags:  
 self.is\_run = True  
 else:  
 self.is\_run = False  
  
 # далее непрерываемые процессы  
 if JUMP in flags and not self.is\_squat:  
 self.is\_jump = True  
  
 if SQUAT in flags:  
 self.is\_squat = True  
  
 # проверка возможности атаки  
 if current\_time - self.last\_fight\_time >= self.fight\_cool\_down:  
 self.fight\_enabled = True  
 if FIGHT in flags and self.fight\_enabled:  
 self.fight\_enabled = False  
 self.last\_fight\_time = current\_time  
 self.is\_fight = True  
  
 # направления  
 if LEFT in flags:  
 self.left = True  
 self.right = False  
  
 if RIGHT in flags:  
 self.left = False  
 self.right = True  
  
 def move(self):  
 *'''  
 Метод для передвижения персонажа в зависимости от флагов состояния а так же для обработки ударов  
 :return:  
 '''* global screen  
 # двигаем персонажа в зависимости от флагов  
 if self.is\_run:  
 if self.left:  
 if self.rect.x - self.speed > 0:  
 self.rect = self.rect.move(-self.speed**,** 0)  
 else:  
 self.rect.x = 0  
 else:  
 if self.rect.x + self.image.get\_width() + self.speed < user\_screen\_width:  
 self.rect = self.rect.move(self.speed**,** 0)  
 else:  
 self.rect.x = user\_screen\_width - self.image.get\_width()  
 if self.is\_jump:  
 if self.jump\_enable:  
 self.jump\_speed = -self.jump\_power  
 self.jump\_enable = False  
  
 if self.rect.y + self.rect.height + self.jump\_speed < self.ground:  
 self.rect = self.rect.move(0**,** self.jump\_speed)  
 self.jump\_speed += 1  
 else:  
 self.rect.y = self.ground - self.rect.height  
  
 if self.rect.y + self.rect.height == self.ground:  
 self.jump\_speed = 0  
 self.jump\_enable = True  
 self.is\_jump = False  
 self.cur\_frame\_jump = 0  
  
 # обрабатываем атаку  
 if self.is\_fight:  
 if not self.is\_enemy\_hit and pygame.sprite.collide\_mask(self**,** self.enemy)\  
 and (self.rect.x <= self.enemy.rect.x if self.right else self.rect.x >= self.enemy.rect.x)\  
 and (int(self.rect.y + self.rect.height / 2) in  
 range(int(self.enemy.rect.y)**,** int(self.enemy.rect.y + self.enemy.rect.height))):  
 self.is\_enemy\_hit = True  
 self.enemy.hit(5)  
 self.enemy.move\_back(self.throw\_back\_power if self.right else -self.throw\_back\_power)  
 pygame.mixer.init()  
 sound\_2 = pygame.mixer.Sound("slap.mp3")  
 sound\_2.play()  
 self.health\_dict[self.enemy] -= self.power  
 else:  
 self.is\_enemy\_hit = False  
  
 def update(self):  
 *'''  
 Обновление  
 Метод запускает смену счетчика анимаций, а так же присваивание нужной картинки  
 :return:  
 '''* self.image\_swap()  
 self.frame\_swap()