|  |
| --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  федеральное государственное автономное образовательное учреждение высшего образования  «САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ» |

Кафедра информационных систем и технологий (№42)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ОТЧЁТ ПО ПРАКТИКЕ  ЗАЩИЩЁН С ОЦЕНКОЙ  Руководитель |  | | | |
| старший преподаватель |  |  |  | Гуков С.Ю. |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

ОТЧЁТ ПО ПРАКТИКЕ

|  |  |  |
| --- | --- | --- |
| вид практики | учебная | |
| тип практики |  | |
| на тему индивидуального задания | |  |
| Реализация компьютерной игры «Файтинг» на языке «Python» | | | |
|  | | | |

|  |  |
| --- | --- |
| выполнен | Дворецким Никитой Михайловичем |
| фамилия, имя, отчество обучающегося в творительном падеже | |

|  |  |  |  |
| --- | --- | --- | --- |
| по направлению подготовки | 09.03.02 |  | Информационные системы и технологии |
|  | код |  | наименование направления |
|  | | | |
| наименование направления | | | |
| направленности | 09.03.02 |  | Информационные системы и технологии |
|  | код |  | наименование направленности |
| в медиаиндустрии | | | |
| наименование направленности | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Обучающийся группы № | 4328 |  |  |  | Дворецкий Н. М. |
|  | номер |  | подпись, дата |  | инициалы, фамилия |

Санкт–Петербург 2024

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

09.03.02 Информационные системы и технологии

на прохождение *учебной* практики обучающегося направления подготовки/ специальности

Дворецкий Никита Михайлович

1. Фамилия, имя, отчество обучающегося:

4328

1. Группа:
2. Тема индивидуального задания:

Реализация компьютерной игры «Файтинг» на языке «Python»

1. Исходные данные:

PyCharm; Язык программирования Python; Компьютерная игра «Файтинг» с использованием библиотеки «Pygame»

1. Содержание отчетной документации:
   1. индивидуальное задание;
   2. отчёт, включающий в себя:
   * титульный лист;
   * материалы о выполнении индивидуального задания (содержание определяется кафедрой);
   * выводы по результатам практики;
   * список использованных источников.
   1. отзыв руководителя от профильной организации (при прохождении практики в профильной организации).
2. Срок представления отчета на кафедру: «01» июня 2024 г.

старший преподаватель

Гуков С.Ю.

Руководитель практики

должность, уч. степень, звание подпись, дата инициалы, фамилия

СОГЛАСОВАНО

Руководитель практики от профильной организации

должность подпись, дата инициалы, фамилия

Задание принял к исполнению:

Дворецкий Н. М.

Обучающийся

дата подпись инициалы, фамилия

Санкт–Петербург 2024

Оглавление

[Введение 5](#_Toc167823226)

[1. О проекте 7](#_Toc167823227)

[1.1 Тема 7](#_Toc167823228)

[1.2 Описание 8](#_Toc167823229)

[2. Пользовательская документация 11](#_Toc167823230)

[2.1 Описание взаимодействия пользователя с игрой 11](#_Toc167823231)

[3. Техническая документация 13](#_Toc167823232)

[3.1 Структура программы 13](#_Toc167823233)

[3.1.1 Основное тело программы 14](#_Toc167823234)

[3.1.2 Функции дополнительных состояний 18](#_Toc167823235)

[3.1.3 Вспомогательные функции 19](#_Toc167823236)

[3.1.4 Класс Button 20](#_Toc167823237)

[3.1.5 Класс HPbar 21](#_Toc167823238)

[3.1.6 Классы Player 22](#_Toc167823239)

[3.2 Описание переменных и функций 23](#_Toc167823240)

[4. Тестирование программы 29](#_Toc167823241)

[6. Загрузка проекта на GitHub 32](#_Toc167823242)

[ЗАКЛЮЧЕНИЕ 33](#_Toc167823243)

[Список источников 34](#_Toc167823244)

[ПРИЛОЖЕНИЕ 1 35](#_Toc167823245)

[ПРИЛОЖЕНИЕ 2 39](#_Toc167823246)

[ПРИЛОЖЕНИЕ 3 43](#_Toc167823247)

[ПРИЛОЖЕНИЕ 4 43](#_Toc167823248)

[ПРИЛОЖЕНИЕ 5 44](#_Toc167823249)

[ПРИЛОЖЕНИЕ 6 47](#_Toc167823250)

# Введение

Компьютерные игры получили свое начало в 1950-ых. Первыми играми оказались «Крестики-нолики» и «Теннис», созданные в 1952 и 1958 годах соответственно. Эти игры увидели свет благодаря сложившейся на тот момент культуре игровых и аркадных автоматов. Однако в те времена эти игры не сыскали популярности, ведь тогда компьютеры были единичными устройствами, располагающимися в дорогих лабораториях, институтах и научных центрах, и поэтому 1950-е назвали «десятилетием фальстартов». Со временем количество игр увеличилось, как и количество компьютеров, что повлияло на появление новых направлений в развитии электронно-вычислительной техники. В частности, это породило игровые приставки, пользующиеся огромной популярностью до сих пор. Сегодня же каждый день выходит огромное количество игр, в связи с чем игровая индустрия является очень конкурентной средой. И зачастую именно игры влияют на мощность компьютеров, обгоняя по требованиям представленные компоненты.

В данной работе будет представлен процесс разработки игры «Файтинг» и итоговый результат работы с языком программирования «Python» и библиотекой разработки игр «Pygame».

Язык программирования «Python» позволяет разработать игры различных масштабов и жанров, особенно с библиотекой «Pygame», которая в свою очередь служит инструментом не только для создания игр, но и любых других приложений с графическим интерфейсом.

Разработка происходила в «PyCharm» – среде программирования языка «Python», позволяющей удобно работать со сторонними библиотеками и внешними файлами.

В ходе отчета большое внимание будет уделено разработке и структуре проекта, работе с файлами и графическим интерфейсом: различные фоны, персонажи и HUD.

Сегодня игровая индустрия позволяет найти игру по душе абсолютно любому человеку, а также создать проект мечты, в котором будет раскрыто то, что желанно именно Вам, ведь разработка — это интересный и увлекательный процесс, в котором человек излагает свои мысли и чувства.

# О проекте

## Тема

Игра “Файтинг” – игра, имитирующая рукопашный бой двух персонажей. Этот жанр зародился в 1979, на аркадных игровых автоматах, однако расцвел после 1991. Чтобы реализовать эту игру требуется, как знание и понимание жанра, так и умение работать с анимацией, а также высокое понимание основ программирования.

Цель проекта – разработать игру в жанре файтинг на языке программирования «Python» с использованием библиотеки «Pygame» в среде программирования «PyCharm». Для достижения цели были сформулированы следующие задачи:

* Подготовить модели персонажей и фоны игры;
* Обрабатывать пользовательский ввод с клавиатуры;
* Реализовать передвижение персонажей по полю игры;
* Создать пользовательский интерфейс – полосы здоровья и время;
* Фиксировать удары персонажей.

Язык программирования «Python» был выбран из-за простоты работы с ним, а также широких возможностей и функционала. «Pygame» - библиотека для работы с графическим интерфейсом, имеющая множество различных сценариев использования, будь то реализация приложения, или создание игры. Эта библиотека имеет много функций, которые окажутся полезными при разработке игры.

В начале работы над проектом были проанализированы подобные игры, в частности “Street Fighter”, откуда мне удалось вынести основы жанра и примеры реализации функционала игры.

## Описание

Итоговый проект представляет из себя игру, где противостоят два игрока, победителем остается тот, кому удается нокаутировать оппонента.

Игра содержит 4 основных состояния:

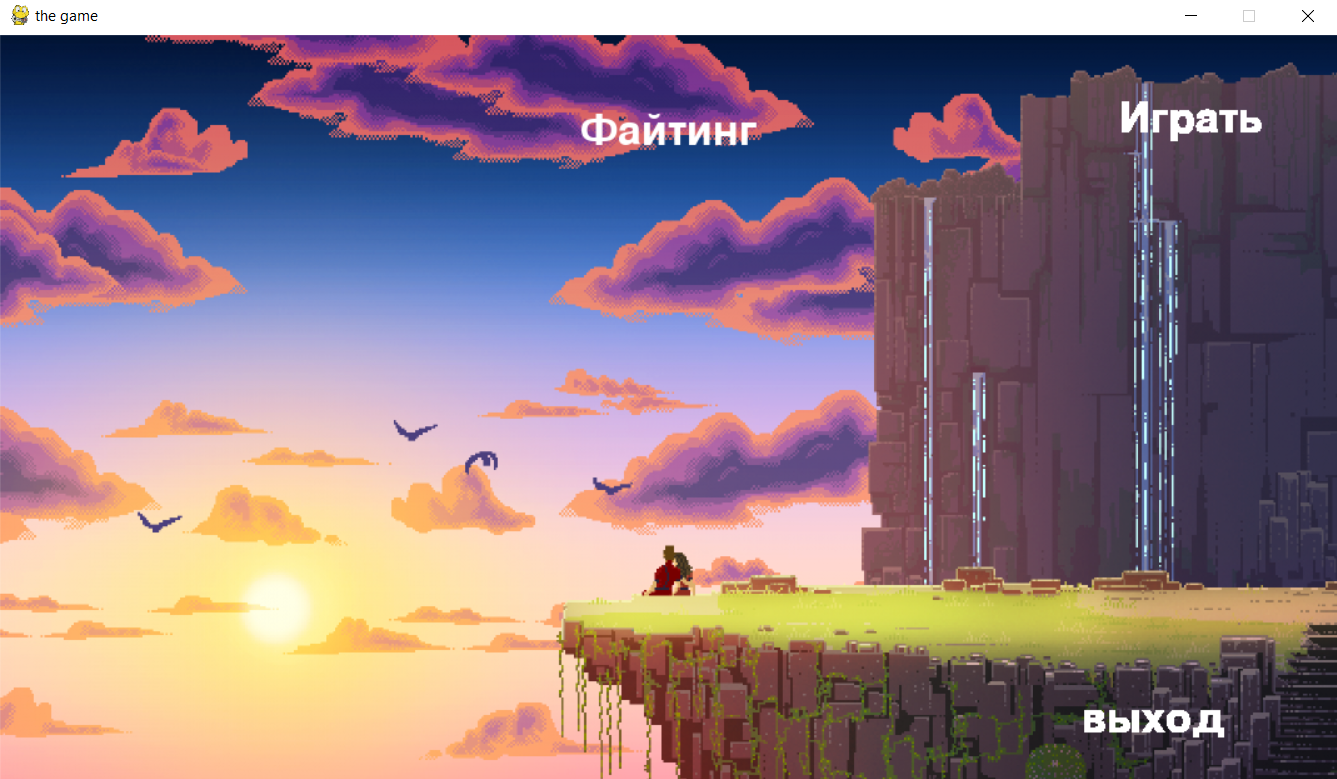
1. Меню – изначальное состояние, в котором игроку представляется возможность запуска игры и выхода из нее.

Рисунок 1 – Меню

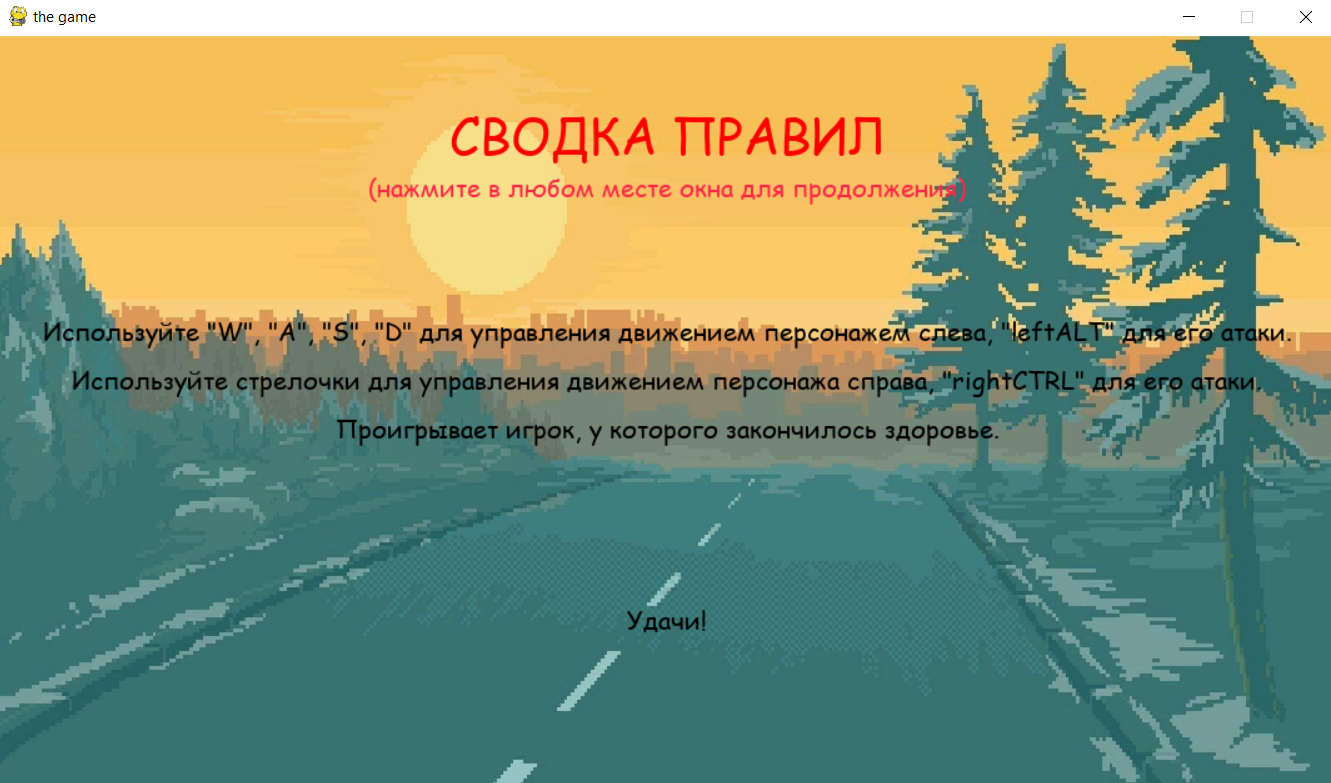
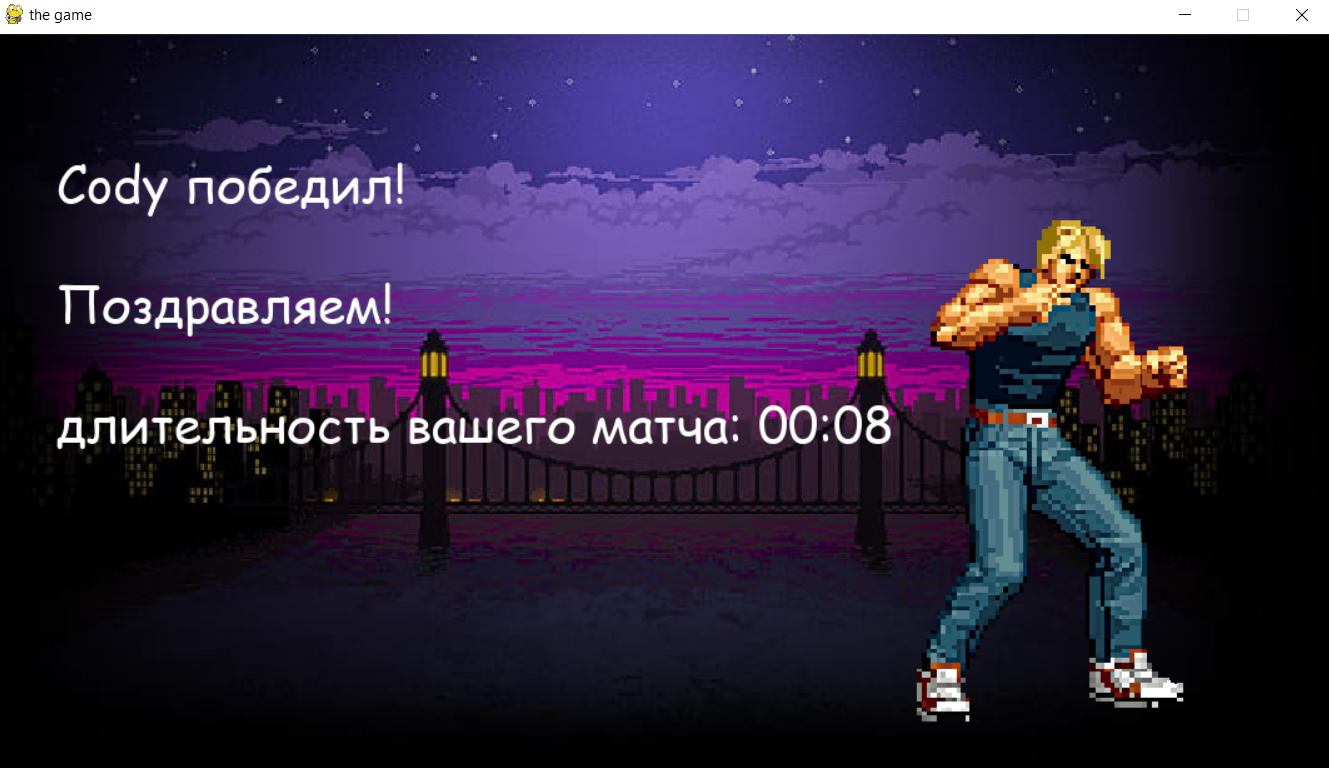
1. Экран правил – окно, в котором описаны правила и управление.

Рисунок 2 – Экран правил

1. Игра – главное состояние, в котором игрокам представляется возможным противостоять друг другу, пытаясь нокаутировать соперника.

Рисунок 3 – Процесс игры

1. Экран завершения – окно, демонстрирующее персонажа-победителя, а также показывающий время, за которое закончился раунд.

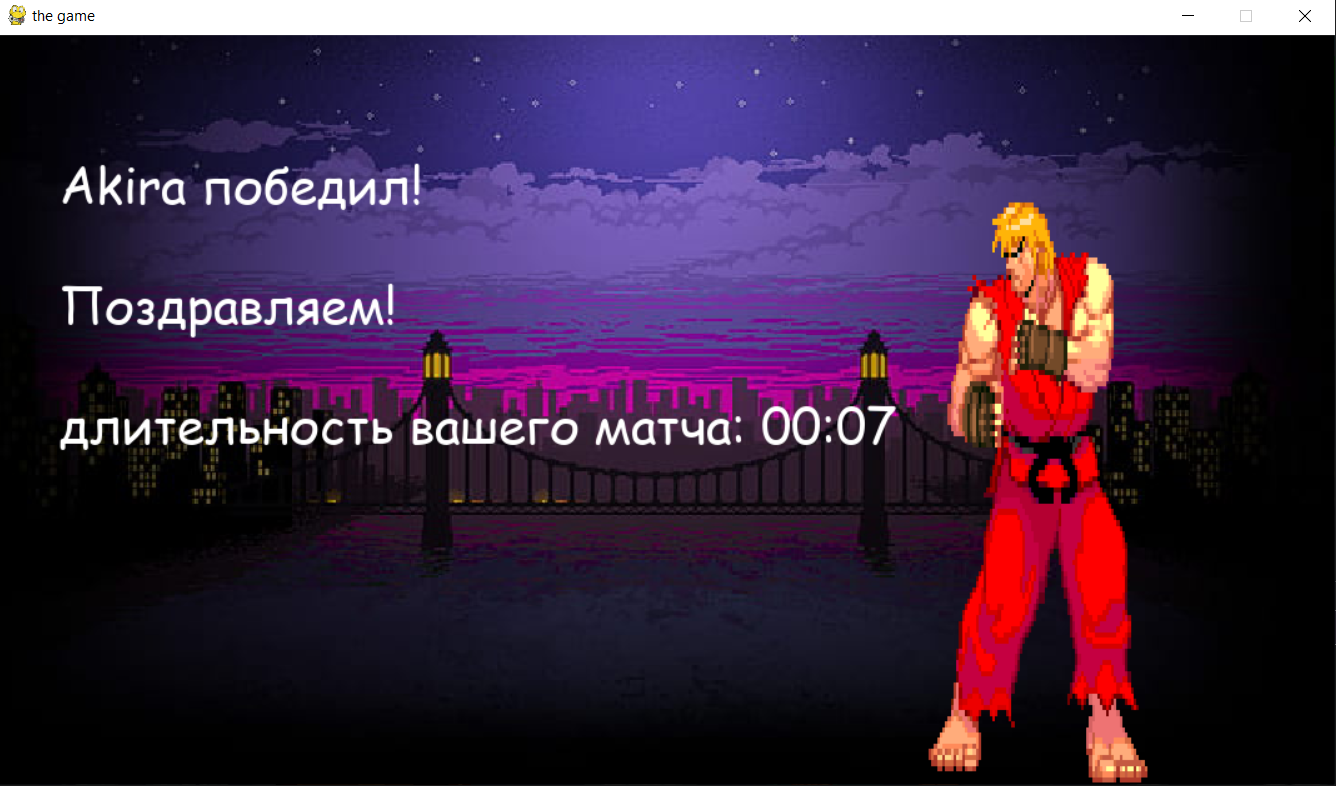
Рисунок 4 – Экран завершения 1

Рисунок 5 – Экран завершения 2

# Пользовательская документация

## Описание взаимодействия пользователя с игрой

Пользователь взаимодействует с программой путем нажатия клавиш клавиатуры и компьютерной мыши. Указанные действия так или иначе влияют на игровой процесс, совершая действия, соответствующее нажатой клавише.

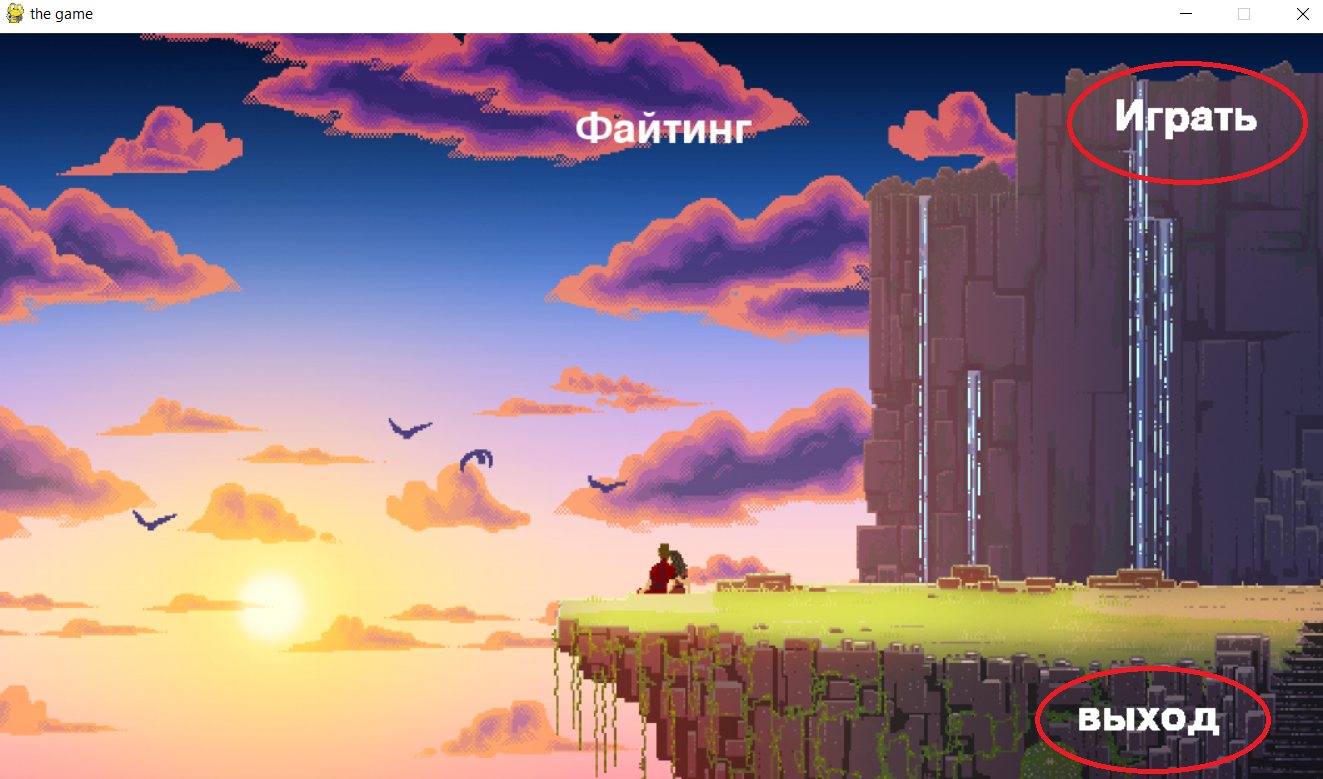
В первом состоянии игры (п.1.2) – меню – управление осуществляется компьютерной мышью, нажатие левой кнопки устройства ввода по кнопкам, реализованным в окне игры, совершает действие, указанное на кнопке: кнопка «Играть» запускает игру, а кнопка «Выход» завершает рабочую сессию.

Рисунок 6 – Функциональные кнопки в меню

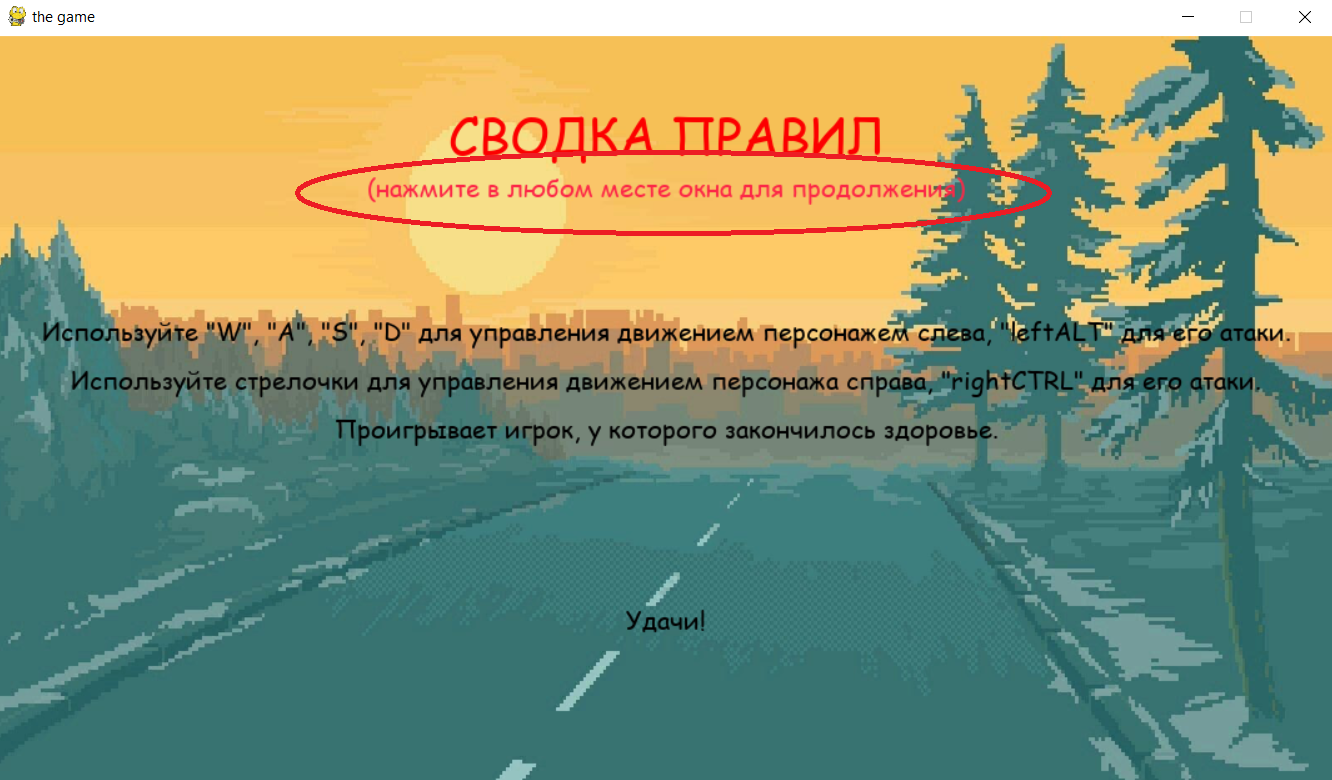
В состоянии 2 (п.1.2) – экран правил – реализован пропуск принципа взаимодействия с игрой путем нажатия любой клавиши клавиатуры или нажатия кнопкой мыши в любой точке действующего окна, также это состояние завершается само по прошествии 10 секунд.

Рисунок 7 – Подсказка о пропуска экрана правил

В состоянии игры (п.1.2) управление происходит клавишами “W” (прыжок), “A” (движение влево), “S” (присед), “D” (движение вправо) и “LeftALT” (удар) для игрока слева и “↑” (прыжок), “←” (движение влево),“↓” (присед), “→” (движение вправо) и “RightCTRL” (удар) для игрока слева.

В состоянии экрана завершения (п.1.2) управление не предусмотрено.

# Техническая документация

## Структура программы

Рассматриваемая программа состоит из 6 файлов с функциями и классами, а также из 20 различных изображений в числе которых различные задние фоны и спрайты.

Данный проект написан с использованием ООП (объектно-ориентированное программирование) – метода программирования при котором используется объединение однотипных объектов в группы, совокупности, классы.

Структура рассматриваемой программы:

1. main.py – файл, содержащий основное тело программы, в котором происходит создание дополнительных переменных, объектов классов и вызов вспомогательных функций;
2. screens.py – файл, в котором реализованы дополнительные экраны, вызывающие состояния «Меню», «Экран правил», «Экран завершения» (п.1.2), а так же функции меняющие состояние игры – начать игру или выйти из игры;
3. button.py – файл, внутри которого располагается класс Buttons, создающий объекты типа кнопки, используемые в состоянии 1 (п.1.2);
4. player.py – набор классов, реализующих состояния персонажей: Playerstay, Playerrun, Playerjump, Playersit, Playerattack, Playerdeath;
5. hud.py – файл с классом HPbar, принужденный создавать полосы здоровья персонажей;
6. tools.py – вспомогательные функции для загрузки и масштабирования изображений.

### Основное тело программы

Часть программы, реализующая объединение всех файлов (п.3.1) и ответственная за работу игры находится в файле main.py (п.3.1).

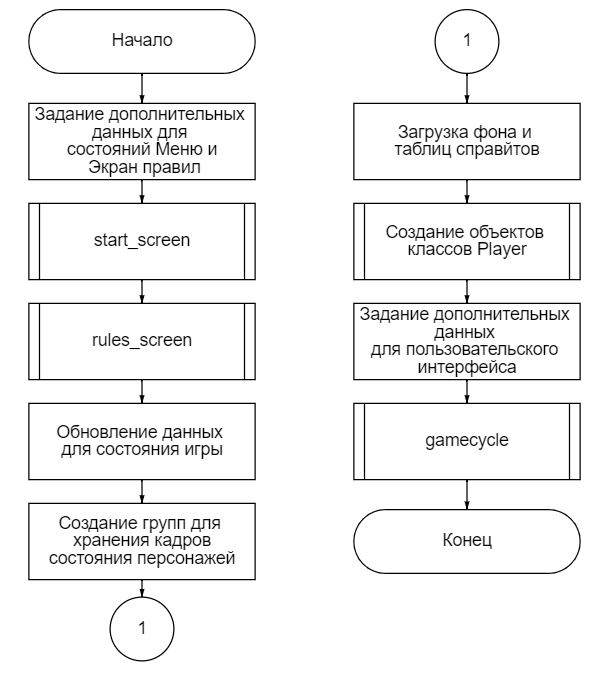
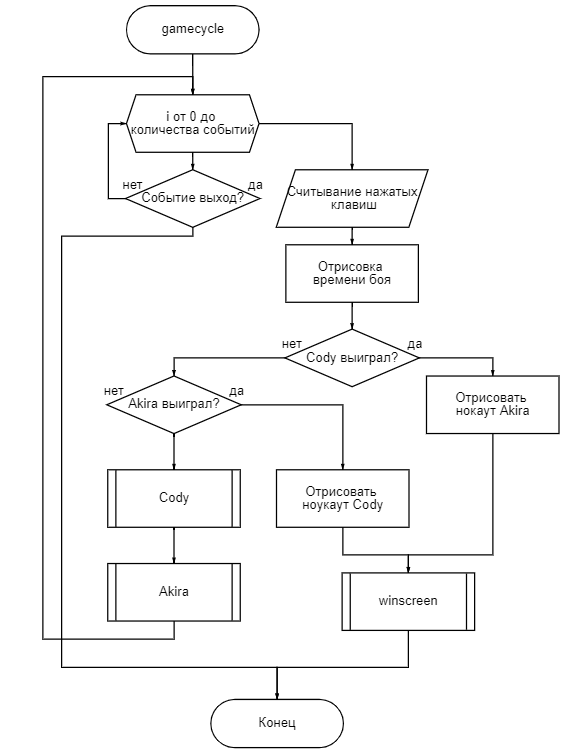
Данный файл представляет из себя создание дополнительных переменных для хранения групп кадров каждого состояния персонажа и реализацию различных взаимодействий пользователя с программой.

Рисунок 8 – Блок-схема main.py

Рисунок 9 – Блок-схема подпрограммы gamecycle.

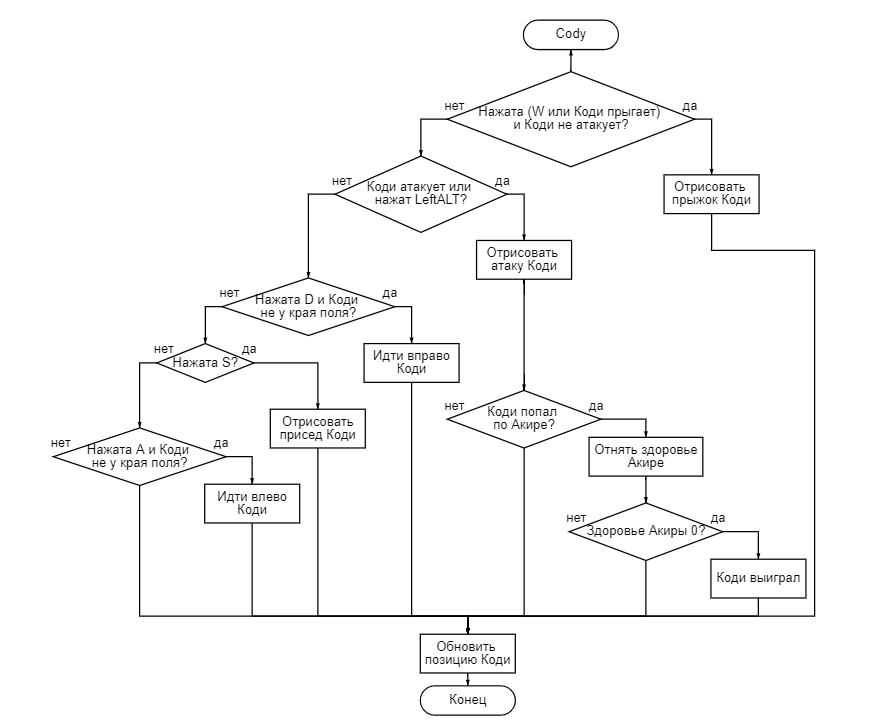
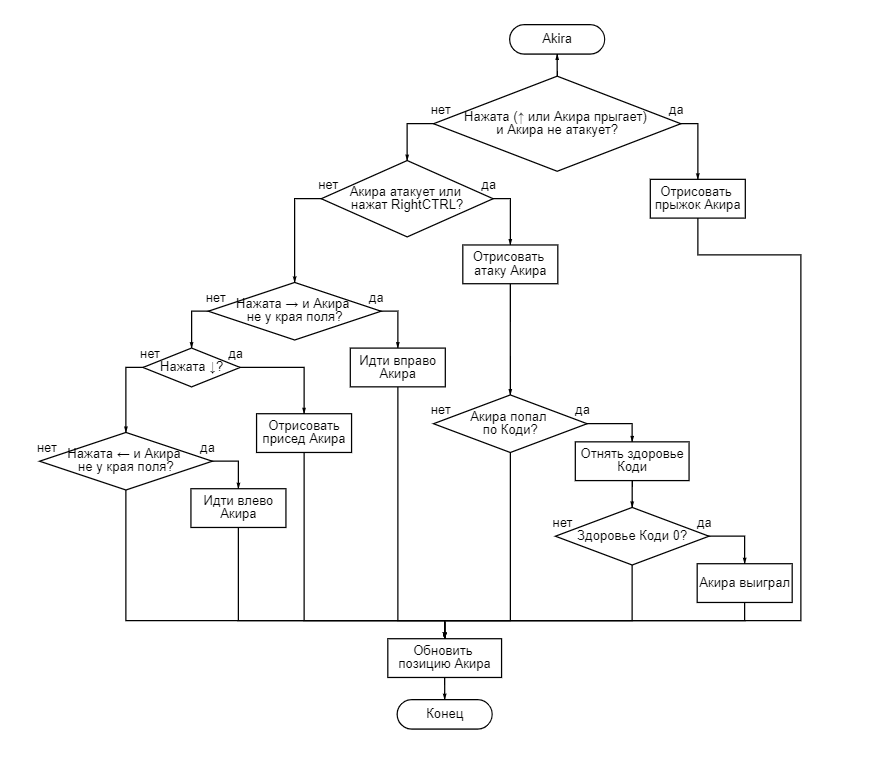
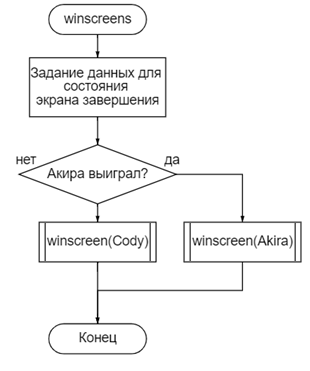
Рисунок 10 – Блок-схема подпрограммы Cody.

Рисунок 11 – Блок-схема подпрограммы Akira.

Рисунок 12 – Блок-схема подпрограммы winscreens.

### Функции дополнительных состояний

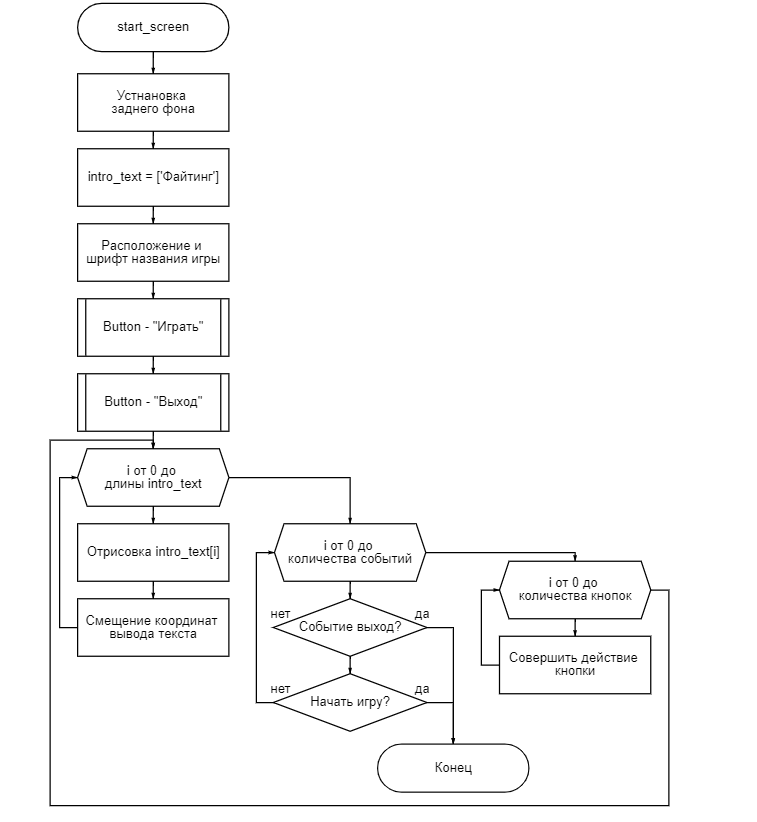
Функции дополнительных состояний: состояние Меню (start\_screen), состояние Экран правил (rules\_screen), состояние Экран завершения (winscreen) (п.1.2) – расположены в файле screens.py (п.3.1).

Рисунок 13 – Блок-схема функции start\_screen.

Функция rules\_screen работает практически по такому же принципу, как и start\_screen, за исключением смены шрифтов, цветов и отсутствия кнопок.

Функция winscreen работает аналогично двум предыдущим, отличаясь лишь загрузкой добавочного изображения победителя.

### Вспомогательные функции

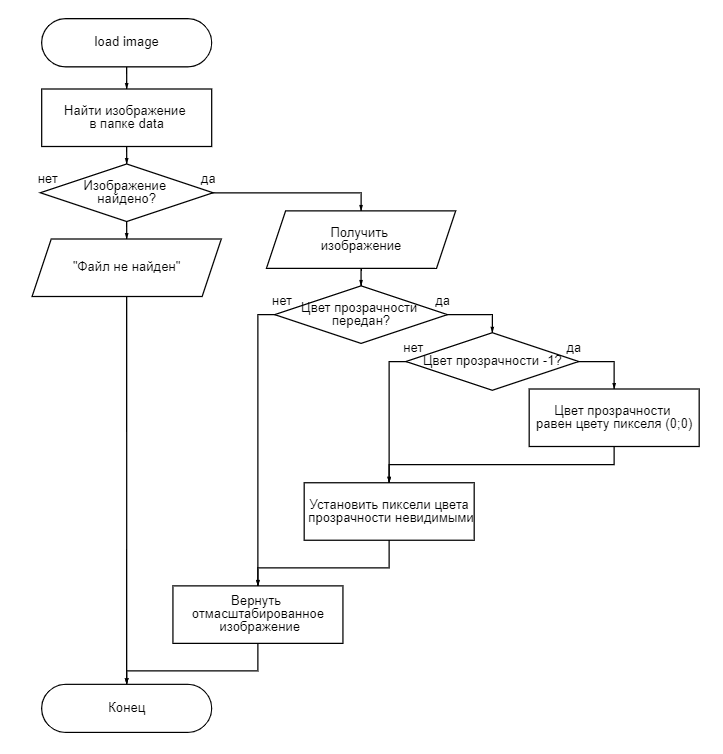
Файл tools.py (п.3.1) содержит вспомогательную функции загрузки и масштабирования изображений.

Рисунок 14 – Блок-схема функции load\_image.

### Класс Button

Класс Button создает объекты-кнопки, использующиеся в состоянии 1 (п.1.2) и находится в файле button.py (п.3.1).

Данный класс присваивает кнопке общие значения, например: цвет, и значения актуальные для каждой кнопки в отдельности, например: размер, вызываемая функция.

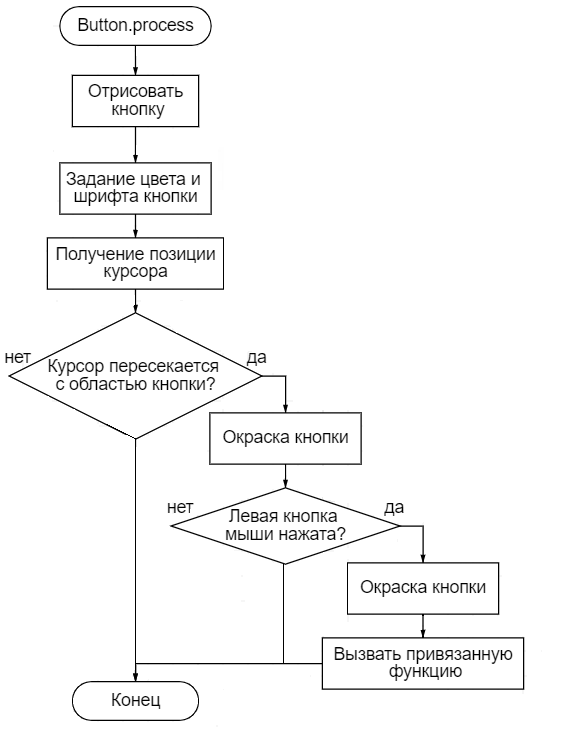
 С объектами типа Button возможно взаимодействие лишь одной функцией, которая обновляет цвета при нажатии и наведении курсором и выполняет процедуры, которые должна выполнить рассматриваемая кнопка.

Рисунок 15 – Блок-схема функции Button.process.

### Класс HPbar

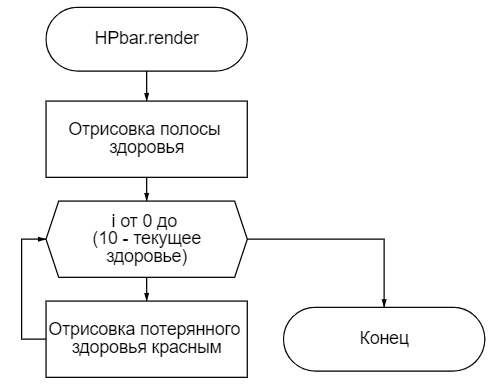
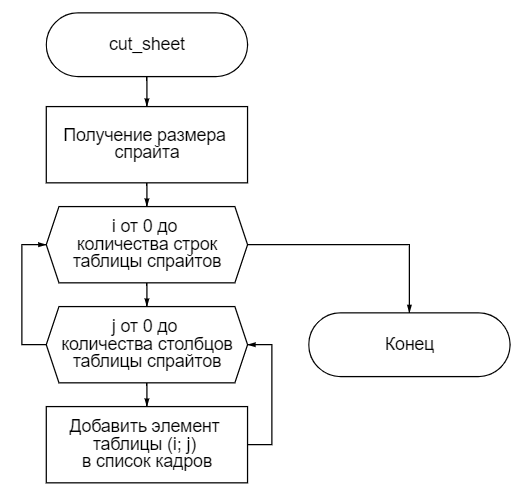
HPbar класс, располагающийся в файле hud.py (п.3.1) и создающий полосы здоровья персонажей. При создании объекта присваиваются общие значения (здоровье игроков) и отдельные для каждого (расположение на экране).

Рисунок 16 – Блок-схема функции HPbar.render.

### Классы Player

Классы файла player.py (п.3.1) идентичны по функциональному содержанию, но предназначены для разных состояний персонажей: Playerstay, Playerrun, Playerjump, Playersit, Playerattack, Playerdeath.  
Рисунок 17 – Блок-схема функции cut\_sheet классов Player.

## Описание переменных и функций

В таблице 1 перечислены используемые в ходе создания игры переменные и их типы, а также функции и их назначение.

Таблица 1. Используемые переменные и функции

|  |  |  |
| --- | --- | --- |
| Название переменной | Тип переменной | Назначение переменной |
| FPSM | int | Количество кадров в секунду в меню |
| clock | pygame.time.clock | Отслеживание времени во время игры |
| width | int | Ширина окна в пикселях |
| height | int | Высота окна в пикселях |
| screen | pygame.Surface | Рабочая область игры |
| button\_objects | list | Массив кнопок |
| timerules | time.time | Время запуска состояния Экрана правил |
| size | tuple | Высота и ширина окна |
| GRAVITY | int | Гравитация в игре |
| screen\_rect | tuple | Крайние координаты окна |
| all\_sprites\_stay1 | pygame.sprite.Group | Группа спрайтов стоящего первого персонажа |
| all\_sprites\_run1 | pygame.sprite.Group | Группа спрайтов бега первого персонажа |
| all\_sprites\_jump1 | pygame.sprite.Group | Группа спрайтов прыжка первого персонажа |
| all\_sprites\_sit1 | pygame.sprite.Group | Группа спрайтов приседа первого персонажа |
| all\_sptrites\_attack1 | pygame.sprite.Group | Группа спрайтов атаки первого персонажа |

Продолжение таблицы 1.

|  |  |  |
| --- | --- | --- |
| Название переменной | Тип переменной | Назначение переменной |
| all\_sprites\_stay1 | pygame.sprite.Group | Группа спрайтов стоящего второго персонажа |
| all\_sprites\_run2 | pygame.sprite.Group | Группа спрайтов бега второго персонажа |
| all\_sprites\_jump2 | pygame.sprite.Group | Группа спрайтов прыжка второго персонажа |
| all\_sprites\_sit2 | pygame.sprite.Group | Группа спрайтов приседа второго персонажа |
| all\_sprites\_attack2 | pygame.sprite.Group | Группа спрайтов атаки второго персонажа |
| all\_sprites\_cody\_death | pygame.sprite.Group | Группа спрайтов нокаута первого персонажа |
| all\_sprites\_akira\_death | pygame.sprite.Group | Группа спрайтов нокаута второго персонажа |
| image | pygame.Surface | Задний фон |
| pl\_stay1 | pygame.Surface | Таблица кадров стоящего первого персонажа |
| pl\_sit1 | pygame.Surface | Таблица кадров приседа первого персонажа |
| pl\_run1 | pygame.Surface | Таблица кадров бега первого персонажа |
| pl\_at1 | pygame.Surface | Таблица кадров атаки первого персонажа |
| pl\_jump1 | pygame.Surface | Таблица кадров прыжка первого персонажа |
| pl\_de1 | pygame.Surface | Таблица кадров нокаута первого персонажа |

Продолжение таблицы 1.

|  |  |  |
| --- | --- | --- |
| Название переменной | Тип переменной | Назначение переменной |
| pl\_stay2 | pygame.Surface | Таблица кадров стоящего второго персонажа |
| pl\_sit2 | pygame.Surface | Таблица кадров приседа второго персонажа |
| pl\_run2 | pygame.Surface | Таблица кадров бега второго персонажа |
| pl\_at2 | pygame.Surface | Таблица кадров атаки второго персонажа |
| pl\_jump2 | pygame.Surface | Таблица кадров прыжка второго персонажа |
| pl\_de2 | pygame.Surface | Таблица кадров нокаута второго персонажа |
| player\_st1 | Playerstay | Объект стоящего первого персонажа |
| player\_st2 | Playerstay | Объект стоящего второго персонажа |
| player\_sit1 | Playersit | Объект приседа первого персонажа |
| player\_sit2 | Playersit | Объект приседа второго персонажа |
| player\_ju1 | Playerjump | Объект прыжка первого персонажа |
| player\_ju2 | Playerjump | Объект прыжка второго персонажа |
| player\_run1 | Playerrun | Объект бега первого персонажа |
| player\_run2 | Playerrun | Объект бега второго персонажа |

Продолжение таблицы 1.

|  |  |  |
| --- | --- | --- |
| Название переменной | Тип переменной | Назначение переменной |
| player\_at1 | Playerattack | Объект атаки первого персонажа |
| player\_at2 | Playerattack | Объект атаки второго персонажа |
| player\_de1 | Playerdeath | Объект нокаута первого персонажа |
| player\_de2 | Playerdeath | Объект нокаута второго персонажа |
| pl1\_pos | list | Начальная позиция первого персонажа |
| pl2\_pos | list | Начальная позиция второго персонажа |
| time | int | Прошедшее время от начала игры |
| counter | int | Счетчик кадров нокаута |
| font | pygame.font.Font | Шрифт для времени игры |
| font1 | pygame.font.Font | Шрифт для отображения имен |
| name | pygame.Surface | Имя второго персонажа |
| name1 | pygame.Surface | Имя первого персонажа |
| text\_x | int | Координата времени боя по абсциссе |
| text\_y | int | Координата времени боя по ординате |
| fps | int | Кадры в секунду в игре |
| safe\_zone | int | Расстояние на котором удары не задевают персонажей |
| Cody\_is\_win | bool | Проверка на победу |

Продолжение таблицы 1.

|  |  |  |
| --- | --- | --- |
| Название переменной | Тип переменной | Назначение переменной |
| bool | Akira\_is\_win | Проверка на победу |
| time1 | int | Минуты с начала игры |
| time2 | int | Секунды с начала игры |
| keys | Pygame.key. ScancodeWrapper | Обнаружение нажатых клавиш |
| Playerstay.cut\_sheet Playersit.cut\_sheet Playerrun.cut\_sheet Playerjump.cut\_sheet Playerattack.cut\_sheet Playerdeath.cut\_sheet | - | Нарезает таблицу кадров |
| Playerstay.update\_pos Playersit.update\_pos Playerrun.update\_pos Playerjump.update\_pos Playerattack.update\_pos Playerdeath.update\_pos | - | Обновляет позицию персонажа |
| Playerstay.update  Playersit.update  Playerrun.update Playerjump.update  Playerattack.update  Playerdeath.update | - | Сменяет текущий кадр анимации следующим |
| HPbar.render | - | Выводит и обновляет здоровье |
| Button.process | - | Считывает нажатие кнопки и совершает назначенное действие |
| start\_screen | - | Запускает состояние Меню (п.1.2) |

Продолжение таблицы 1.

|  |  |  |
| --- | --- | --- |
| Название переменной | Тип переменной | Назначение переменной |
| winscreen | - | Запускает состояние Экран завершения (п.1.2) |
| rules\_screen | - | Запускает состояние Экран правил (п.1.2) |
| load\_image | - | Загружает изображние и масштабирует его |

# Тестирование программы

Для корректной работы программы и ее взаимодействия с пользователем требовалось проведение тестов, начиная от проверки простых переменных-счетчиков, заканчивая правильностью считывания нажатий игроком.

1. Тестирование игрового процесса:

Процесс игры является главной составляющей проекта и его отладке было выделено отдельное внимание. Были проведены проверки на корректность движений персонажей, на правильность регистрации удара, на работоспособность уклонения-приседа.

1. Тестирование состояний игры:

Состояния обязаны безошибочно сменять друг друга, а также верно выполнять функции, заложенные в них. В частности, функционирование кнопок состояния Меню, перехода по нажатию экрана правил (п.1.2).

1. Устранение ошибок:

В ходе работы над программой были выявлены различные ошибки – некорректность работы кнопок, перемещение персонажа в стартовую позицию при приседе, неправильное отображение пользовательского интерфейса, неверная обработка нанесения удара, а также невозможность ходьбы персонажа слева при нахождении персонажа справа у границы. Все найденные ошибки были исправлены и игра достигла состояния корректной работы.

После полного тестирования и устранения ошибок игра достигла ожидаемой работоспособности, что говорит о важности отладки во время работы с программой.

# Видео-демонстрация механик игры

Видео-демонстрация механик игры представляет собой видеозапись работоспособности и функционала игры, в число которых входят действия персонажей, система регистрации ударов, состояния меню, экрана правил и экрана завершения.

Ссылка на видеоролик: <https://youtu.be/Q_dY-MFLvn8>

В представленном видеоролике продемонстрирован полный игровой процесс, ожидающий игроков, а также пример игры, в котором раскрыты различные аспекты файтинга.

# Загрузка проекта на GitHub

Для сохранения проекта в облачной доступности была выбрана популярная платформа GitHub, позволяющая удобно оперировать с проектом, добавляя новые файлы и данные или изменяя уже существующие.

Данный проект был сохранен в репозитории и содержит все требуемые внешние данные (спрайты и фоны), файлы с программой и документацию README.md, содержащую краткое описание проекта и его работу.

Ссылка на репозиторий: <https://github.com/w0nkest/uch_practice>

# ЗАКЛЮЧЕНИЕ

В ходе работы мне удалось освоить библиотеку Pygame языка Python и закрепить владения языком программирования, изучив работу с объектно-ориентированным программированием, что позволило создать отлично работающую игру жанра файтинг.

Целью проекта являлось создание игры подобно уже существующим для большего понимания процесса разработки игр и получения новых навыков и опыта для применения их в дальнейшем.

Библиотека Pygame позволила углубиться в работу с графическим интерфейсом и реализовать игру, научиться фиксировать различные действия на экране согласно нажатию клавиш.

Разработка игровых приложений является интересным процессом, требующим обширное понимание в программировании, работе с графикой и построении логических моделей. Для себя мне удалось открыть создание игр как новую и интересную сферу в которой я могу попытать свои силы и, возможно, реализовать себя как программиста.

# Список источников

1. Статья об истории игр // stepgames URL: <https://stepgames.ru/blog/istoriya-kompyuternyh-igr> (Дата обращения: 09.05.2024)
2. Статья об истории игр // Википедия URL: <https://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%82%D0%BE%D1%80%D0%B8%D1%8F_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D1%8B%D1%85_%D0%B8%D0%B3%D1%80> (Дата обращения: 09.05.2024)
3. Файтинг – жанр // Википедия URL: <https://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D1%82%D0%B8%D0%BD%D0%B3> (Дата обращения: 10.05.2024)

# ПРИЛОЖЕНИЕ 1

main.py

import pygame  
from screens import start\_screen**,** rules\_screen**,** winscreen  
import time  
import sys  
from player import Playerstay**,** Playersit**,** Playerjump**,** Playerattack**,** Playerdeath**,** Playerrun  
from hud import HPbar  
from tools import load\_image  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 pygame.init()  
 FPSM = **60** clock = pygame.time.Clock()  
 width**,** height = **1070, 600** screen = pygame.display.set\_mode((width**,** height))  
 pygame.display.set\_caption('the game')  
 button\_objects = []  
 start\_screen(screen**,** clock**,** button\_objects**,** width**,** height**,** FPSM)  
 timerules = time.time()  
 rules\_screen(screen**,** clock**,** width**,** height**,** timerules**,** FPSM)  
  
 size = width**,** height = **1360, 640** # 1700, 800  
 GRAVITY = **10** screen\_rect = (**0, 0,** width**,** height)  
 screen = pygame.display.set\_mode(size)  
  
 all\_sprites\_stay1 = pygame.sprite.Group()  
 all\_sprites\_run1 = pygame.sprite.Group()  
 all\_sprites\_jump1 = pygame.sprite.Group()  
 all\_sprites\_sit1 = pygame.sprite.Group()  
 all\_sprites\_attack1 = pygame.sprite.Group()  
  
 all\_sprites\_stay2 = pygame.sprite.Group()  
 all\_sprites\_run2 = pygame.sprite.Group()  
 all\_sprites\_jump2 = pygame.sprite.Group()  
 all\_sprites\_sit2 = pygame.sprite.Group()  
 all\_sprites\_attack2 = pygame.sprite.Group()  
  
 all\_sprites\_cody\_death = pygame.sprite.Group()  
 all\_sprites\_akira\_death = pygame.sprite.Group()  
  
 image = pygame.transform.scale(load\_image('fon.jpg')**,** (**1360, 640**)) # (1700, 800)  
  
 pl1\_HPbar = HPbar(**12, 12,** False) # 12, 12  
 pl2\_HPbar = HPbar(**800, 12**) # 1000, 15  
  
 pl1\_pos = [**80, 240**] # [100, 300]  
  
 pl\_stay1 = load\_image('player1\_stay.png'**,** -**1,** size=**0.8**)  
 pl\_run1 = load\_image('player1\_run.png'**,** -**1,** size=**0.8**)  
 pl\_at1 = load\_image('player1\_attack.png'**,** -**1,** size=**0.8**)  
 pl\_sit1 = load\_image('player1\_sit.png'**,** -**1,** size=**0.8**)  
 pl\_jump1 = load\_image('player1\_jump.png'**,** -**1,** size=**0.8**)  
 pl\_de1 = load\_image('cody\_death.png'**,** -**1,** size=**0.8**)  
  
 player\_de1 = Playerdeath(pl\_de1**, 6, 1,** pl1\_pos[**0**]**,** pl1\_pos[**1**]**,** all\_sprites\_cody\_death**,** False)  
 player\_ju1 = Playerjump(pl\_jump1**, 6, 1,** pl1\_pos[**0**]**,** pl1\_pos[**1**]**,** all\_sprites\_jump1**,** False)  
 player\_sit1 = Playersit(pl\_sit1**, 5, 1,** pl1\_pos[**0**]**,** pl1\_pos[**1**]**,** all\_sprites\_sit1**,** False)  
 player\_at1 = Playerattack(pl\_at1**, 5, 1,** pl1\_pos[**0**]**,** pl1\_pos[**1**]**,** all\_sprites\_attack1**,** False)  
 player\_st1 = Playerstay(pl\_stay1**, 2, 1,** pl1\_pos[**0**]**,** pl1\_pos[**1**]**,** all\_sprites\_stay1**,** False)  
 player\_ru1 = Playerrun(pl\_run1**, 8, 1,** pl1\_pos[**0**]**,** pl1\_pos[**1**]**,** all\_sprites\_run1**,** False)  
  
 pl2\_pos = [**1040, 206**] # [1300, 257]  
  
 pl\_stay2 = load\_image('player\_stay.png'**,** -**1,** size=**0.8**)  
 pl\_run2 = load\_image('player\_run.png'**,** -**1,** size=**0.8**)  
 pl\_jump2 = load\_image('player\_jump.png'**,** -**1,** size=**0.8**)  
 pl\_sit2 = load\_image('player\_sit.png'**,** -**1,** size=**0.8**)  
 pl\_at2 = load\_image('player\_attack.png'**,** -**1,** size=**0.8**)  
 pl\_de2 = load\_image('booom.png'**,** -**1,** size=**0.8**)  
  
 player\_at2 = Playerattack(pl\_at2**, 11, 1,** pl2\_pos[**0**]**,** pl2\_pos[**1**]**,** all\_sprites\_attack2)  
 player\_sit2 = Playersit(pl\_sit2**, 6, 1,** pl2\_pos[**0**]**,** pl2\_pos[**1**]**,** all\_sprites\_sit2)  
 player\_ju2 = Playerjump(pl\_jump2**, 11, 1,** pl2\_pos[**0**]**,** pl2\_pos[**1**]**,** all\_sprites\_jump2)  
 player\_st2 = Playerstay(pl\_stay2**, 10, 1,** pl2\_pos[**0**]**,** pl2\_pos[**1**]**,** all\_sprites\_stay2)  
 player\_ru2 = Playerrun(pl\_run2**, 11, 1,** pl2\_pos[**0**]**,** pl2\_pos[**1**]**,** all\_sprites\_run2)  
 player\_de2 = Playerdeath(pl\_de2**, 3, 2,** pl2\_pos[**0**]**,** pl2\_pos[**1**]**,** all\_sprites\_akira\_death)  
  
 time = **0** counter = **10** font = pygame.font.Font(None**, 120**) # 150  
 font1 = pygame.font.Font(None**, 80**) # 100  
 name = font1.render('Akira'**,** True**,** 'White')  
 name1 = font1.render('Cody'**,** True**,** 'White')  
 text\_x = **572** # 715  
 text\_y = **16** # 20  
 safe\_zone = **184** # 230  
 fps = **10** Cody\_is\_win = False  
 Akira\_is\_win = False  
  
 while True:  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 sys.exit()  
 keys = pygame.key.get\_pressed()  
 screen.blit(image**,** (**0, 0**))  
 time1 = time // **1000** // **60** if time1 < **10**:  
 time1 = '0' + str(time1)  
 time2 = time // **1000** % **60** if time2 < **10**:  
 time2 = '0' + str(time2)  
 text = font.render(f'{time1}:{time2}'**,** True**,** (**255, 200, 0**))  
 screen.blit(text**,** (text\_x**,** text\_y))  
 screen.blit(name1**,** (**12, 92**)) # (15, 115)  
 screen.blit(name**,** (**1200, 92**)) # (1500, 115)  
 pl1\_HPbar.render(screen)  
 pl2\_HPbar.render(screen)  
 keys = pygame.key.get\_pressed()  
 if Cody\_is\_win:  
 all\_sprites\_stay1.draw(screen)  
 player\_st1.update()  
 all\_sprites\_akira\_death.draw(screen)  
 player\_de2.update()  
 counter -= **1** if counter == **0**:  
 break  
 elif Akira\_is\_win:  
 all\_sprites\_stay2.draw(screen)  
 player\_st2.update()  
 all\_sprites\_cody\_death.draw(screen)  
 player\_de1.update()  
 counter -= **1** if counter == **0**:  
 break  
 else:  
 if (keys[pygame.K\_w] or player\_ju1.is\_jumping) and not player\_at1.is\_attacking:  
 player\_sit1.is\_sitting = False  
 player\_ju1.is\_jumping = True  
 player\_ju1.update()  
 all\_sprites\_jump1.draw(screen)  
 elif keys[pygame.K\_LALT] or player\_at1.is\_attacking:  
 player\_sit1.is\_sitting = False  
 player\_sit1.cur\_frame = **0** player\_at1.is\_attacking = True  
 player\_at1.update()  
 all\_sprites\_attack1.draw(screen)  
 if abs(pl1\_pos[**0**] - pl2\_pos[**0**]) < safe\_zone + **20** and player\_at1.cur\_frame == **4** \  
 and not player\_sit2.is\_sitting: # 25  
 pl2\_HPbar.hp -= **1** if pl2\_HPbar.hp == **0**:  
 Cody\_is\_win = True  
 elif keys[pygame.K\_d] and pl1\_pos[**0**] < **1160** and abs(pl1\_pos[**0**] - pl2\_pos[**0**]) > safe\_zone: # 1450  
 player\_ru1.way = **1** player\_sit1.cur\_frame = **0** all\_sprites\_run1.draw(screen)  
 player\_ru1.update(pl1\_pos**,** pl2\_pos)  
 elif keys[pygame.K\_s]:  
 player\_sit1.is\_sitting = True  
 player\_sit1.update()  
 all\_sprites\_sit1.draw(screen)  
 elif keys[pygame.K\_a] and **0** < pl1\_pos[**0**]:  
 player\_sit1.is\_sitting = False  
 player\_ru1.way = -**1** player\_sit1.cur\_frame = **0** all\_sprites\_run1.draw(screen)  
 player\_ru1.update(pl1\_pos**,** pl2\_pos)  
 else:  
 player\_sit1.is\_sitting = False  
 player\_sit1.cur\_frame = **0** all\_sprites\_stay1.draw(screen)  
 player\_st1.update()  
 player\_de1.update\_pos(pl1\_pos**,** pl2\_pos)  
 player\_ju1.update\_pos(pl1\_pos**,** pl2\_pos)  
 player\_ru1.update\_pos(pl1\_pos**,** pl2\_pos)  
 player\_st1.update\_pos(pl1\_pos**,** pl2\_pos)  
 player\_at1.update\_pos(pl1\_pos**,** pl2\_pos)  
 player\_sit1.update\_pos(pl1\_pos**,** pl2\_pos)  
  
 if (keys[pygame.K\_UP] or player\_ju2.is\_jumping) and not player\_at2.is\_attacking:  
 player\_sit2.is\_sitting = False  
 player\_ju2.is\_jumping = True  
 player\_ju2.update()  
 all\_sprites\_jump2.draw(screen)  
 elif keys[pygame.K\_RCTRL] or player\_at2.is\_attacking:  
 player\_sit2.is\_sitting = False  
 player\_at2.is\_attacking = True  
 all\_sprites\_attack2.draw(screen)  
 player\_at2.update()  
 if abs(pl1\_pos[**0**] - pl2\_pos[**0**]) < safe\_zone + **20** and player\_at2.cur\_frame == **5** \  
 and not player\_sit1.is\_sitting: # 25  
 pl1\_HPbar.hp -= **2** if pl1\_HPbar.hp == **0**:  
 Akira\_is\_win = True  
 elif keys[pygame.K\_RIGHT] and pl2\_pos[**0**] < **1160**: # 1450  
 player\_sit2.is\_sitting = False  
 player\_ru2.way = -**1** player\_sit2.cur\_frame = **0** all\_sprites\_run2.draw(screen)  
 player\_ru2.update(pl1\_pos**,** pl2\_pos)  
 elif keys[pygame.K\_DOWN]:  
 player\_sit2.is\_sitting = True  
 player\_sit2.update()  
 all\_sprites\_sit2.draw(screen)  
 elif keys[pygame.K\_LEFT] and **0** < pl2\_pos[**0**] and abs(pl1\_pos[**0**] - pl2\_pos[**0**]) > safe\_zone:  
 player\_sit2.is\_sitting = False  
 player\_ru2.way = **1** player\_sit2.cur\_frame = **0** all\_sprites\_run2.draw(screen)  
 player\_ru2.update(pl1\_pos**,** pl2\_pos)  
 else:  
 player\_sit2.is\_sitting = False  
 player\_sit2.cur\_frame = **0** all\_sprites\_stay2.draw(screen)  
 player\_st2.update()  
 player\_at2.update\_pos(pl1\_pos**,** pl2\_pos)  
 player\_ru2.update\_pos(pl1\_pos**,** pl2\_pos)  
 player\_ju2.update\_pos(pl1\_pos**,** pl2\_pos)  
 player\_sit2.update\_pos(pl1\_pos**,** pl2\_pos)  
 player\_st2.update\_pos(pl1\_pos**,** pl2\_pos)  
 player\_de2.update\_pos(pl1\_pos**,** pl2\_pos)  
  
 time += clock.get\_time()  
  
 clock.tick(fps)  
 pygame.display.flip()  
  
 width**,** height = **1070, 600** screen = pygame.display.set\_mode((width**,** height))  
 if Akira\_is\_win:  
 winscreen(screen**,** clock**,** width**,** height**,** 'Akira'**,** time1**,** time2**,** FPSM)  
 else:  
 winscreen(screen**,** clock**,** width**,** height**,** 'Cody'**,** time1**,** time2**,** FPSM)  
  
 pygame.quit()

# ПРИЛОЖЕНИЕ 2

player.py

import pygame  
  
  
class Playerstay(pygame.sprite.Sprite):  
 def \_\_init\_\_(self**,** sheet**,** columns**,** rows**,** x**,** y**,** sprite\_group**,** right=True):  
 super().\_\_init\_\_(sprite\_group)  
 self.r = right  
 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]  
  
 def update\_pos(self**,** pl1\_pos**,** pl2\_pos):  
 if self.r:  
 self.rect = self.rect.move(pl2\_pos[**0**] - self.rect[**0**]**,** pl2\_pos[**1**] - self.rect[**1**])  
 else:  
 self.rect = self.rect.move(pl1\_pos[**0**] - self.rect[**0**]**,** pl1\_pos[**1**] - self.rect[**1**])  
  
  
class Playerrun(pygame.sprite.Sprite):  
 def \_\_init\_\_(self**,** sheet**,** columns**,** rows**,** x**,** y**,** sprite\_group**,** right=True):  
 super().\_\_init\_\_(sprite\_group)  
 self.r = right  
 self.frames = []  
 self.cut\_sheet(sheet**,** columns**,** rows)  
 self.cur\_frame = **0** self.way = **1** 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**,** pl1\_pos**,** pl2\_pos):  
 self.cur\_frame = (self.cur\_frame + **1**) % len(self.frames)  
 self.image = self.frames[self.cur\_frame \* self.way]  
 if self.way == **1**:  
 self.rect = self.rect.move(-**24, 0**) # (-30, 0)  
 if self.r:  
 pl2\_pos[**0**] -= **24** # 30  
 else:  
 pl1\_pos[**0**] += **24** # 30  
 else:  
 self.rect = self.rect.move(**24, 0**) # (30, 0)  
 if self.r:  
 pl2\_pos[**0**] += **24** # 30  
 else:  
 pl1\_pos[**0**] -= **24** # 30  
  
 def update\_pos(self**,** pl1\_pos**,** pl2\_pos):  
 if self.r:  
 self.rect = self.rect.move(pl2\_pos[**0**] - self.rect[**0**]**,** pl2\_pos[**1**] - self.rect[**1**])  
 else:  
 self.rect = self.rect.move(pl1\_pos[**0**] - self.rect[**0**]**,** pl1\_pos[**1**] - self.rect[**1**])  
  
  
class Playerjump(pygame.sprite.Sprite):  
 def \_\_init\_\_(self**,** sheet**,** columns**,** rows**,** x**,** y**,** sprite\_group**,** right=True):  
 super().\_\_init\_\_(sprite\_group)  
 self.r = right  
 self.frames = []  
 self.cut\_sheet(sheet**,** columns**,** rows)  
 if right:  
 self.r = right  
 self.mask = pygame.mask.from\_surface(self.frames[**6**])  
 else:  
 self.mask = pygame.mask.from\_surface(self.frames[**3**])  
 self.cur\_frame = **0** self.image = self.frames[self.cur\_frame]  
 self.rect = self.rect.move(x**,** y)  
 self.is\_jumping = False  
  
 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]  
 x = **3** y = **5** if self.r:  
 x = **6** y = **10** if self.cur\_frame < x:  
 self.rect = self.rect.move(**0,** -**44**) # (0, -55)  
 else:  
 self.rect = self.rect.move(**0, 36**) # (0, 45)  
 if self.cur\_frame == y:  
 self.is\_jumping = False  
  
 def update\_pos(self**,** pl1\_pos**,** pl2\_pos):  
 if not self.is\_jumping:  
 if self.r:  
 self.rect = self.rect.move(pl2\_pos[**0**] - self.rect[**0**]**,** pl2\_pos[**1**] - self.rect[**1**])  
 else:  
 self.rect = self.rect.move(pl1\_pos[**0**] - self.rect[**0**]**,** pl1\_pos[**1**] - self.rect[**1**])  
  
  
class Playersit(pygame.sprite.Sprite):  
 def \_\_init\_\_(self**,** sheet**,** columns**,** rows**,** x**,** y**,** sprite\_group**,** right=True):  
 super().\_\_init\_\_(sprite\_group)  
 self.r = right  
 self.frames = []  
 self.cut\_sheet(sheet**,** columns**,** rows)  
 self.cur\_frame = -**1** self.is\_sitting = False  
 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):  
 x = **4** if self.r:  
 x = **5** if self.cur\_frame != x:  
 self.cur\_frame = (self.cur\_frame + **1**) % len(self.frames)  
 self.image = self.frames[self.cur\_frame]  
 self.is\_sitting = True  
  
 def update\_pos(self**,** pl1\_pos**,** pl2\_pos):  
 if self.r:  
 self.rect = self.rect.move(pl2\_pos[**0**] - self.rect[**0**]**,** pl2\_pos[**1**] - self.rect[**1**])  
 else:  
 self.rect = self.rect.move(pl1\_pos[**0**] - self.rect[**0**]**,** pl1\_pos[**1**] - self.rect[**1**])  
  
  
class Playerattack(pygame.sprite.Sprite):  
 def \_\_init\_\_(self**,** sheet**,** columns**,** rows**,** x**,** y**,** sprite\_group**,** right=True):  
 super().\_\_init\_\_(sprite\_group)  
 self.r = right  
 self.frames = []  
 self.cut\_sheet(sheet**,** columns**,** rows)  
 self.cur\_frame = **0** self.is\_attacking = False  
 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]  
 x = **4** if self.r:  
 x = **10** if self.cur\_frame == x:  
 self.is\_attacking = False  
  
 def update\_pos(self**,** pl1\_pos**,** pl2\_pos):  
 if self.r:  
 self.rect = self.rect.move(pl2\_pos[**0**] - self.rect[**0**]**,** pl2\_pos[**1**] - self.rect[**1**])  
 else:  
 self.rect = self.rect.move(pl1\_pos[**0**] - self.rect[**0**]**,** pl1\_pos[**1**] - self.rect[**1**])  
  
  
class Playerdeath(pygame.sprite.Sprite):  
 def \_\_init\_\_(self**,** sheet**,** columns**,** rows**,** x**,** y**,** sprite\_group**,** right=True):  
 super().\_\_init\_\_(sprite\_group)  
 self.frames = []  
 self.cut\_sheet(sheet**,** columns**,** rows)  
 self.cur\_frame = **0** self.is\_right = right  
 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):  
 if self.cur\_frame != **5**:  
 self.cur\_frame = (self.cur\_frame + **1**) % len(self.frames)  
 if not self.is\_right:  
 self.rect = self.rect.move(-**64, 0**) # (-80, 0)  
 self.image = self.frames[self.cur\_frame]  
  
 def update\_pos(self**,** pl1\_pos**,** pl2\_pos):  
 x = **0** if self.is\_right:  
 x = **120** self.rect = self.rect.move(pl1\_pos[**0**] - self.rect[**0**] + x**,** pl1\_pos[**1**] - self.rect[**1**])

# ПРИЛОЖЕНИЕ 3

hud.py

import pygame  
  
  
class HPbar:  
 def \_\_init\_\_(self**,** x**,** y**,** right=True):  
 self.hp = **10** self.x = x  
 self.y = y  
 self.is\_right = right  
  
 def render(self**,** screen):  
 pygame.draw.rect(screen**,** pygame.Color('Grey')**,** (self.x**,** self.y**, 544, 80**)**, 4**) # (680, 100), 5  
 pygame.draw.rect(screen**,** pygame.Color('Green')**,** (self.x + **5,** self.y + **5, 536, 72**)) # (670, 90)  
 if self.hp < **0**:  
 self.hp = **0** for i in range(**10** - self.hp):  
 if self.is\_right:  
 pygame.draw.rect(screen**,** pygame.Color('Red')**,** (self.x + **4** + i \* **54,** self.y + **4, 54, 72**))  
 # self.x + 5 + i \* 67, self.y + 5, 67, 90  
 else:  
 pygame.draw.rect(screen**,** pygame.Color('Red')**,** ((self.x + **5** + **9** \* **54**) - i \* **54,** self.y + **4, 54, 72**))  
 # (self.x + 5 + 9 \* 67) - i \* 67, self.y + 5, 67, 90  
 hp\_font = pygame.font.Font(None**, 80**) # 100  
 hp\_text = hp\_font.render(f"100 / {self.hp \* **10**}"**,** True**,** 'White')  
 if self.is\_right:  
 hp\_text = hp\_font.render(f"{self.hp \* **10**} / 100"**,** True**,** 'White')  
 hp\_text\_x = self.x + **160** # 200  
 hp\_text\_y = self.y + **16** # 20  
 screen.blit(hp\_text**,** (hp\_text\_x**,** hp\_text\_y))

# ПРИЛОЖЕНИЕ 4

tools.py

import os**,** sys**,** pygame  
  
  
def load\_image(name**,** colorkey=None**,** size=**1**):  
 fullname = os.path.join('data'**,** name)  
 if not os.path.isfile(fullname):  
 print(f"Файл с изображением '{fullname}' не найден")  
 sys.exit()  
 image = pygame.image.load(fullname)  
 if colorkey is not None:  
 image = image.convert()  
 if colorkey == -**1**:  
 colorkey = image.get\_at((**0, 0**))  
 image.set\_colorkey(colorkey)  
 return pygame.transform.scale(image**,** (int(image.get\_width() \* size)**,** int(image

# ПРИЛОЖЕНИЕ 5

screen.py

from tools import load\_image  
import sys**,** time  
import pygame  
from button import Button  
STARTGAME = False  
  
  
def playfunc():  
 global STARTGAME  
 STARTGAME = True  
  
  
def terminate():  
 print("TERMINATION")  
 pygame.quit()  
 sys.exit()  
  
  
def start\_screen(screen**,** clock**,** button\_objects**,** width**,** height**,** FPSM=**60**):  
 global STARTGAME  
 intro\_text = ["Файтинг"]  
  
 fon = pygame.transform.scale(load\_image('bgmenu1.png')**,** (width**,** height))  
 screen.blit(fon**,** (**0, 0**))  
 font = pygame.font.Font(None**, 50**)  
 text\_coord = **50** Button(width - int(**175** \* (width / **1070**))**,** int(**50** \* (height / **600**))**, 150, 70,** button\_objects**,** 'Играть'**,** playfunc)  
 Button(width - int(**205** \* (width / **1070**))**,** height - int((**70** \* (height / **600**)))**, 150, 50,** button\_objects**,** 'выход'**,** terminate)  
 for line in intro\_text:  
 string\_rendered = font.render(line**,** True**,** pygame.Color('white'))  
 intro\_rect = string\_rendered.get\_rect()  
 text\_coord += **10** intro\_rect.top = text\_coord  
 intro\_rect.x = width // **2** - intro\_rect.width // **2** 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 STARTGAME:  
 return  
 for obj in button\_objects:  
 obj.process(screen)  
 pygame.display.flip()  
 clock.tick(FPSM)  
  
  
def winscreen(screen**,** clock**,** width**,** height**,** name**,** time1**,** time2=**0,** FPSM=**60**):  
 intro\_text = [f'{name} победил!'**,** 'Поздравляем!'**,** f'длительность вашего матча: {time1}:{time2}'**,** ]  
  
 fon = pygame.transform.scale(load\_image('endscreenbg.jpg')**,** (width**,** height))  
 screen.blit(fon**,** (**0, 0**))  
 font = pygame.font.SysFont('comicsansms'**, 40**)  
 text\_coord = **50** player\_win = load\_image(f'{name}\_win.png'**,** -**1**)  
 screen.blit(player\_win**,** (**700, 80**))  
  
 for line in intro\_text:  
 string\_rendered = font.render(line**,** True**,** pygame.Color('white'))  
 intro\_rect = string\_rendered.get\_rect()  
 text\_coord += **40** intro\_rect.top = text\_coord  
 intro\_rect.x = **50** 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()  
  
 pygame.display.flip()  
 clock.tick(FPSM)  
  
  
def rules\_screen(screen**,** clock**,** width**,** height**,** timerules**,** FPSM=**60**):  
 intro\_text = ["СВОДКА ПРАВИЛ"**,** "(нажмите в любом месте окна для продолжения)"**,** ""**,** ""**,** 'Используйте "W", "A", "S", "D" для управления движением персонажем слева, "leftALT" для его атаки.'**,** 'Используйте стрелочки для управления движением персонажа справа, "rightCTRL" для его атаки.'**,** 'Проигрывает игрок, у которого закончилось здоровье.'**,** ''**,** ''**,** ''**,** 'Удачи!'  
 ]  
 fon = pygame.transform.scale(load\_image('pixelroad\_bg2.jpg')**,** (width**,** height))  
 screen.blit(fon**,** (**0, 0**))  
  
 # head text of rules screen  
 mainfont = pygame.font.SysFont('comicsansms'**, 40**)  
 font = pygame.font.SysFont('comicsansms'**, 20**)  
 string\_rendered = mainfont.render(intro\_text[**0**]**,** True**,** (**252, 0, 0**))  
 intro\_rect = string\_rendered.get\_rect()  
 intro\_rect.top = **50** intro\_rect.x = width // **2** - string\_rendered.get\_width() // **2** text\_coord = intro\_rect.height + **50** screen.blit(string\_rendered**,** intro\_rect)  
 heightmaintext = string\_rendered.get\_height()  
  
 # tip to skip rules screen  
 string\_rendered = font.render(intro\_text[**1**]**,** True**,** (**252, 40, 71**))  
 intro\_rect = string\_rendered.get\_rect()  
 intro\_rect.top = **50** + heightmaintext  
 intro\_rect.x = width // **2** - string\_rendered.get\_width() // **2** text\_coord += intro\_rect.height  
 screen.blit(string\_rendered**,** intro\_rect)  
  
 # rules text  
 for line in intro\_text[**2**:]:  
 string\_rendered = font.render(line**,** True**,** (**0, 0, 0**))  
 intro\_rect = string\_rendered.get\_rect()  
 text\_coord += **10** intro\_rect.top = text\_coord  
 intro\_rect.x = width // **2** - string\_rendered.get\_width() // **2** 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 or \  
 int(time.time() - timerules) >= **10**:  
 return  
 pygame.display.flip()  
 clock.tick(FPSM)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 pygame.init()  
 STARTGAME = False  
 FPSM = **60** clock = pygame.time.Clock()  
 width**,** height = **1070, 600** screen = pygame.display.set\_mode((width**,** height))  
 button\_objects = []  
 start\_screen(screen**,** clock**,** button\_objects**,** width**,** height**,** FPSM)  
 timerules = time.time()  
 rules\_screen(screen**,** clock**,** width**,** height**,** timerules**,** FPSM)

# ПРИЛОЖЕНИЕ 6

button.py

import pygame  
  
  
class Button:  
 def \_\_init\_\_(self**,** x**,** y**,** width\_b**,** height\_b**,** button\_objects**,** buttonText='Button'**,** onclickFunction=None):  
 self.x = x  
 self.y = y  
 self.width = width\_b  
 self.height = height\_b  
 self.onclickFunction = onclickFunction  
 self.alreadyPressed = False  
 self.buttonText = buttonText  
  
 self.fillColors = {  
 'normal': '#ffffff'**,** 'hover': '#666666'**,** 'pressed': '#333333'**,** }  
  
 self.buttonSurface = pygame.Surface((self.width**,** self.height))  
 self.buttonRect = pygame.Rect(self.x**,** self.y**,** self.width**,** self.height)  
 self.buttonSurf = pygame.font.Font(None**, 50**).render(self.buttonText**,** True**,** (**255, 255, 255**))  
 button\_objects.append(self)  
  
 def process(self**,** screen):  
 screen.blit(self.buttonSurf**,** (self.x**,** self.y))  
 screen.blit(self.buttonSurf**,** self.buttonRect)  
  
 self.buttonSurf = pygame.font.Font(None**, 50**).render(self.buttonText**,** True**,** (**255, 255, 255**))  
 mousePos = pygame.mouse.get\_pos()  
 if self.buttonRect.collidepoint(mousePos):  
 self.buttonSurf = pygame.font.Font(None**, 50**).render(self.buttonText**,** True**,** self.fillColors['hover'])  
 if pygame.mouse.get\_pressed(num\_buttons=**3**)[**0**]:  
 self.buttonSurf = pygame.font.Font(None**, 50**).render(self.buttonText**,** True**,** self.fillColors['pressed'])  
 self.onclickFunction()  
 self.alreadyPressed = True  
 else:  
 self.alreadyPressed = False