V Международная конференция учащихся

НАУЧНО-ТВОРЧЕСКИЙ ФОРУМ

Вид работы: практико-ориентированный проект по информатике

«Создание игры с помощью языка программирования Python»

Автор работы: Левченков Алексей Дмитриевич,
ученик 11 «Б» класса
МАОУ СОШ №7

Руководитель: Полукарикова Алла Сергеевна,
учитель информатики
МАОУ СОШ №7

2024 г.

Оглавление

Введение

Компьютеры проникли во все сферы деятельности человека, начиная с начального образования и заканчивая изучением новейших технологий, изучения новых видов материи, неизвестных пока человечеству.

Благодаря разнообразию программного и аппаратного обеспечения сегодня возможно использование всех потенциальных возможностей компьютерных технологий. Это позволяет хранить огромное количество информации, занимая при этом минимальное место.

Среди сфер информационных технологий можно выделить основные: веб-разработка, разработка игр, мобильная разработка, разработка десктопных приложений, наука о данных и программирование встраиваемых систем. В данной исследовательской работе я ознакомлю вас с разработкой игр при помощи языка программирования Python через простую библиотеку Pygame.

Актуальность: Разработка игр является одной из самых быстрорастущих индустрий в мире. Постоянно появляются новые технологии и тренды, которые делают создание игр еще более актуальным. Разработка игр может быть полезна в научной деятельности в нескольких аспектах. Во-первых, игры могут использоваться для моделирования научных процессов и явлений, что позволяет ученым лучше понимать их. Во-вторых, игры могут помочь в обучении и распространении научных знаний, особенно среди молодежи. В-третьих, игры могут стимулировать интерес к науке и технологиям, что может привести к новым открытиям и инновациям.

Цель: создание простой стратегической игры при помощи языка программирования Python через простую библиотеку Pygame.

Задачи:

- познакомить слушателей со сферой разработки игр;
- рассказать историю о первых шагах в создании игр;
- объяснить на каких языках программирования можно создавать игры;
- внедрить понятие игровых движков, через которые можно упростить разработку игр;
- этапы разработки игр.

Основная часть

1. Из теории игр

1.1. Основные понятия

Разработка игр – процесс создания игр для различных устройств пользователя.

Разработкой компьютерных игр может заниматься как один человек, так и фирма (коллектив разработчиков). Коммерческие игры создаются командами разработчиков, нанятыми одной фирмой. Фирмы могут специализироваться на производстве игр для персональных компьютеров, игровых приставок или планшетных компьютеров. Разработка может финансироваться другой, более крупной фирмой — издателем. Фирма-издатель по окончании разработки занимается распространением игры и берёт на себя связанные с этим затраты.

Основные жанры игр: экшены, платформеры, стратегии, аркады, варгеймы, квесты, адвенчуры (приключения), ил (интерактивная литература), файтинги, РПГ (ролевые **игры**), симуляторы, гонки, настольные, спортивные, головоломки, виртуальные тиры и другие.

Также стоит учитывать возрастные ограничения для игр. Существуют игры, которые создаются для детей 3 и более лет. Это обычные и развивающие игры, которые могут обучить ребёнка чему-то хорошему. Например: тема о противостоянии добра и зла, освоение того или иного школьного материала, развитие фантазии и прочее. Достигая возраста 12 и более лет меняются интересы ребёнка. Чаще всего ребёнок такого возраста интересуется военным делом (сокращённо шутер), где главной целью является одоление своего оппонента и доказательство своего превосходства над ним. Разумеется, всё превосходство заключается в правильных построениях стратегических тактик, а также командная работа.

Самые популярные жанры игр:

1. **экшен-игры** (включая шутеры от первого лица, файтинги и платформеры);
2. **игры приключения** (включая экшены с приключениями);
3. **РПГ** (включая ММОРПГ, то есть ролевые игры);
4. **стратегии** (пошаговые и в реальном времени);
5. **симуляторы** (в том числе симуляторы спорта, жизни и градостроительства)

1.2. История создания игр

История компьютерных игр начинается в **1940-х** и **1950-х годах**, когда в академической среде разрабатывались простые игры и симуляции. Компьютерные игры длительное время не были популярны, и только в **1970-х** и **1980-х годах**, когда появились доступные для широкой публики аркадные **автоматы**, **игровые консоли** и **домашние компьютеры**, компьютерные игры становятся частью **поп-культуры**.

Появлению коммерческих компьютерных игр предшествовала уже сложившаяся индустрия развлекательных **аркадных автоматов** наподобие **пинбола** — механических игр, для запуска которых требовалось бросить монетку в щель приемника. Такие автоматы выпускались с XIX века, используя всё более сложные механизмы, а с **1930-х годов** и электричество; параллельно развивались и **музыкальные автоматы**-«джукбоксы». В **1947 году** было запатентовано «**Развлекательное устройство на основе электронно-лучевой трубки**» Томаса Голдсмита и Эстла Манна – оно считается первым специально предназначенным для игры устройством, выводившим изображение на экран, то есть «**видеоигрой**»

В начале **1950-х годов** создавались специализированные компьютеры наподобие **Nimrod** опять же для игры в **ним** и **Bertie the Brain** и **OXO** для игры в **крестики-нолики**. **Tennis for Two**, разработанная физиком Уильямом Хигинботамом, имитировала игру в **теннис** с графическим интерфейсом, используя аналоговый компьютер и **осциллограф** как средство вывода в реальном времени.

К **1960-м годам** развитие вычислительной техники — от вакуумных ламп к транзисторам, а от транзисторов к интегральным схемам — сделало компьютеры намного более мощными и доступными, чем прежде. В 1961 году группа студентов из клуба **Tech Model Railroad Club** (TMRC, клуб **железнодорожных моделистов**) при **Массачусетском технологическом институте** использовала новейший на тот момент компьютер **DEC PDP-1** для создания одной из первых компьютерных игр – космического симулятора **Spacewar!.**

Рисунок 1: игра Space War

В течение **1960-1970**-х годов в университетской и научной среде США продолжали создаваться различные игры — как учебные программы, как упражнения в программировании и просто для развлечения студентов. Их количество и сложность росли по мере того, как компьютеры (**мейнфреймы**) становились всё более доступными, с развитием **языков программирования** и появлением первых компьютерных сетей (**ARPAnet**), позволявших пользователям взаимодействовать друг с другом и делиться программирования.

1990-е годы были ознаменованы заметными инновациями в компьютерных играх. Основными достижениями называют переход от растровой графики к полностью полигональному 3D, снижение популярности аркадных игр и появление нескольких новых жанров - шутер от первого лица, стратегия в реальном времени и MMO.

В мае **1991 года** техасская студия **Id Software** выпустила на ПК игру под названием **Hovertank 3D**, которая стала прорывом в 3D графике и фактически создала жанр шутера от первого лица. В ней использовался движок Криса Грина, созданный в **1990 году**. Он позволял накладывать на каждый трёхмерный объект разнообразные текстуры, превращая одинаковые модели в визуально разные.

1.3. Языки разработки игр

Язык программирования — **формальный язык**, предназначенный для записи **компьютерных программ**. Язык программирования определяет набор **лексических**, **синтаксических** и **семантических** правил, определяющих внешний вид программы и действия, которые выполнит исполнитель (обычно — **ЭВМ**) под её управлением.

В мире насчитывается **более 700** языков программирования, и список постоянно пополняется. А большинство IT-команд в крупных компаниях и стартапах использует

несколько языков и фреймворков (набор инструментов и библиотек, которые помогают разработчикам создавать приложения быстрее и эффективнее.) одновременно. Разработчики тоже все чаще становятся билингвами и мультилингвами и осваивают от 2 до 5 языков, в том числе тестируют новые технологии.

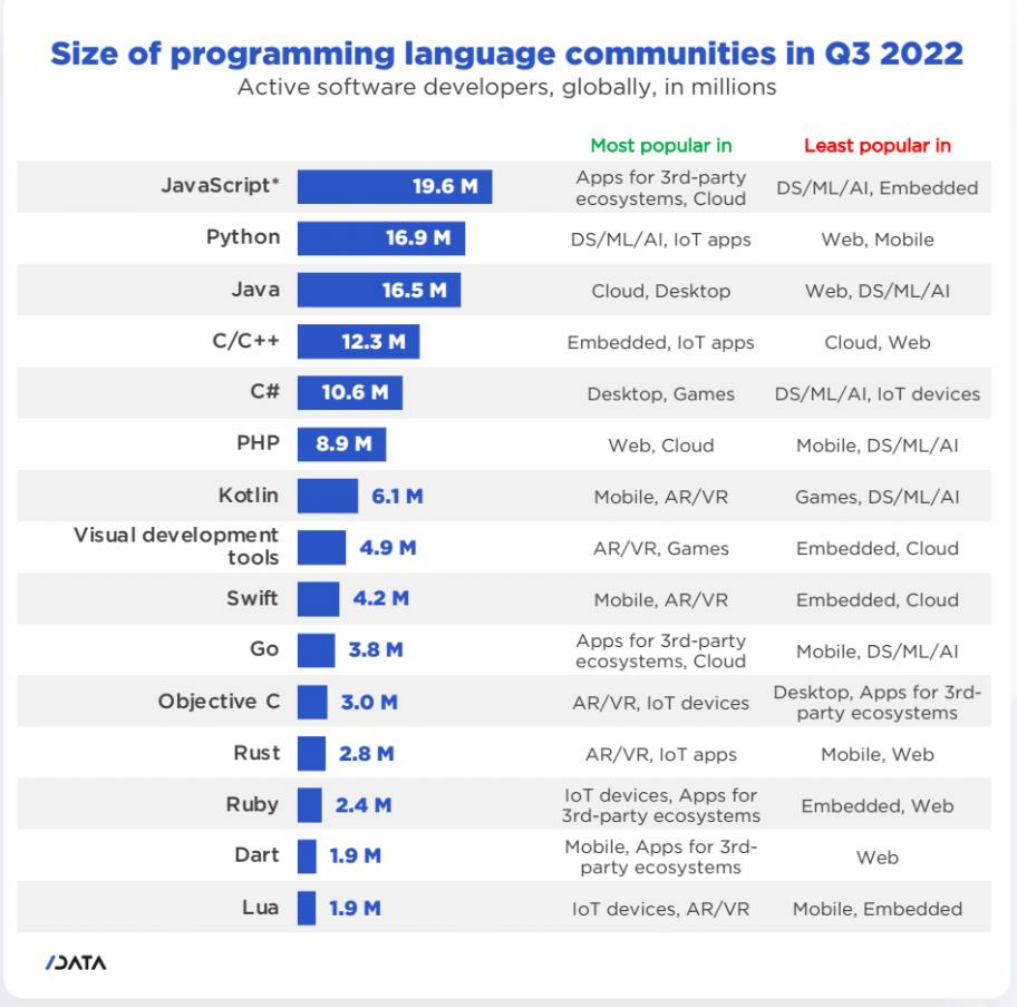


Рисунок 2: востребованные языки программирования

Язык программирования предназначен для написания компьютерных программ, которые представляют собой набор правил, позволяющих компьютеру выполнить тот или иной вычислительный процесс, организовать управление различными объектами, и т. п. Язык программирования отличается от естественных языков тем, что предназначен для управления ЭВМ, в то время как естественные языки используются, прежде всего, для общения людей между собой. Большинство языков программирования использует специальные конструкции для определения и манипулирования структурами данных и управления процессом вычислений.

Не существует общепринятой систематичной таксономии языков программирования. Есть множество черт, согласно которым можно производить классификацию языков, причём одни из них однозначно проводят разделы между языками на основе технических свойств, другие основываются на доминирующих признаках, имеют исключения и более условны, а третьи полностью субъективны и нередко сопровождаются заблуждениями, но на практике весьма распространены.

Наиболее популярными языками программирования игровых продуктов являются такие: **C++, Java, JavaScript, Python, Swift и C#**. Новичку лучше всего начать с последнего. Различают три подхода к разработке игр: полностью самостоятельно или с применением игровых движков и с использованием заготовок. Первый – самый сложный и требует серьезных навыков в программировании и математических знаний.

1.4. Игровые движки

Игровой движок — базовое программное обеспечение компьютерной игры. Разделение игры и игрового движка часто расплывчато, и не всегда студии проводят чёткую границу между ними. Но в общем случае термин «игровой движок» применяется для того программного обеспечения, которое пригодно для повторного использования и расширения, и тем самым может быть рассмотрено как основание для разработки множества различных игр без существенных изменений.

Игровые движки служат для упрощённого создания игр той или иной платформы (Android, IOS, Windows и прочее). Сам же термин «игровой движок» появился в середине 1990-х годов — в это время окончательно установилось доминирование IBM-совместимых компьютеров, а быстрые процессоры и «хитрое» программирование дали 30 и более кадров в секунду в трёхмерных играх. Игры Doom и Quake от id Software оказались настолько популярными, что другие разработчики вместо того, чтобы работать с чистого листа, лицензировали основные части программного обеспечения и создавали свою собственную графику, персонажей, оружие и уровни — «игровой контент» или «игровые ресурсы». Движок Quake был использован в более чем десяти проектах и дал серьёзный толчок развитию middleware-индустрии.

Среди самых популярных игровых движков можно выделить:

1) Gamemaker. Кроссплатформенный коммерческий движок для создания 2D-игр, созданная разработчиками из YoYoGames.

Платформы: windows и macOS.

Знания языков программирования: Game Maker Language.

Стоимость: 39$ в год.

Достоинства и недостатки:

- кроссплатформенность, поддерживаемые платформы: Windows, macOS, Linux, Opera GX, Android, iOS, Windows Phone, Tizen, Xbox, PlayStation;
- поддержка библиотек и расширений, в том числе на разных языках;
- гибкая ценовая категория, Free версия GameMaker абсолютно бесплатна;
- интеграция с несколькими системами управления версиями;
- интеграция со Steam, Google Play, App Store;
- собственный язык программирования Game Maker Language (GML), который часто критикуется профессионалами;
- несмотря на возможность работы с 3D, в GameMaker она крайне неудобна.

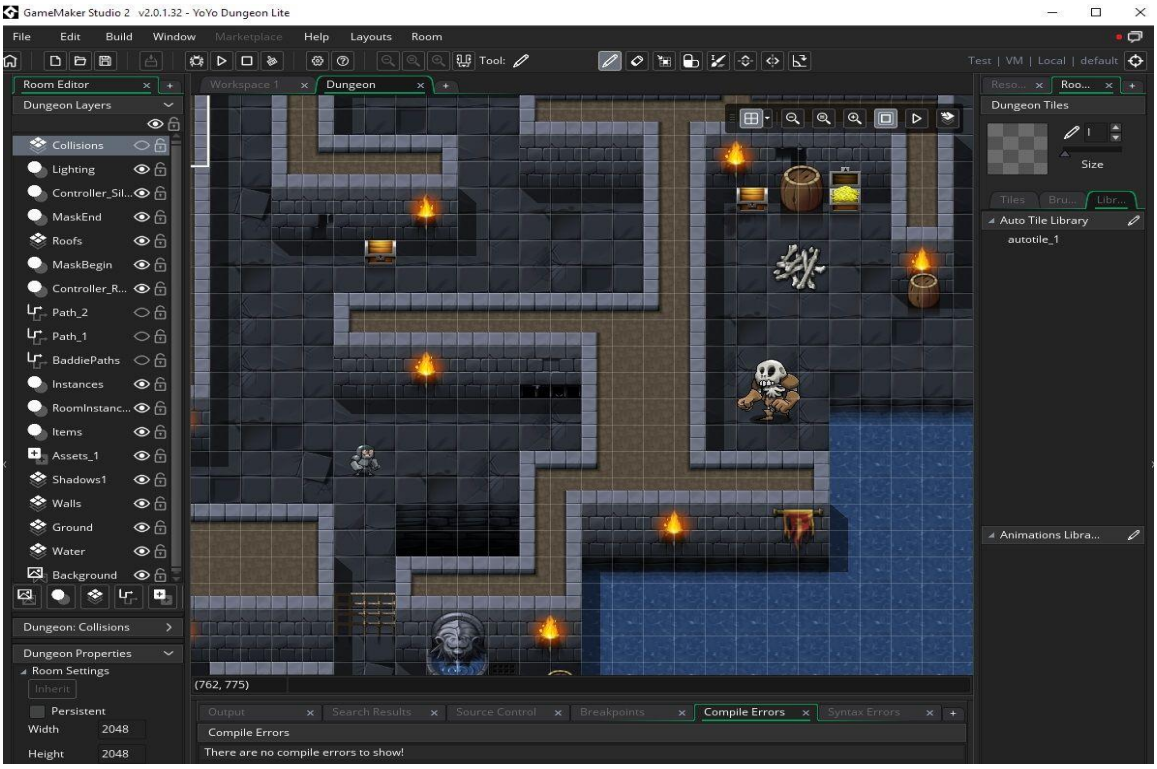


Рисунок 3: интерфейс движка Gamemaker.

2) Godot Engine: открытый кроссплатформенный 2Dи 3D-игровой движок под лицензией MIT, который разрабатывается сообществом Godot Engine Community.

Платформы: Windows, macOS, Linux.

Знания языков программирования: GDScript, C#, VisualScript.

Стоимость: полностью бесплатно.

Достоинства и недостатки:

- Открытый исходный код;
- Мультиплатформенность;
- Интуитивно понятный интерфейс;
- Обширная документация и активное сообщество;
- Ограниченный функционал;
- Меньше готовых инструментов и ассетов.

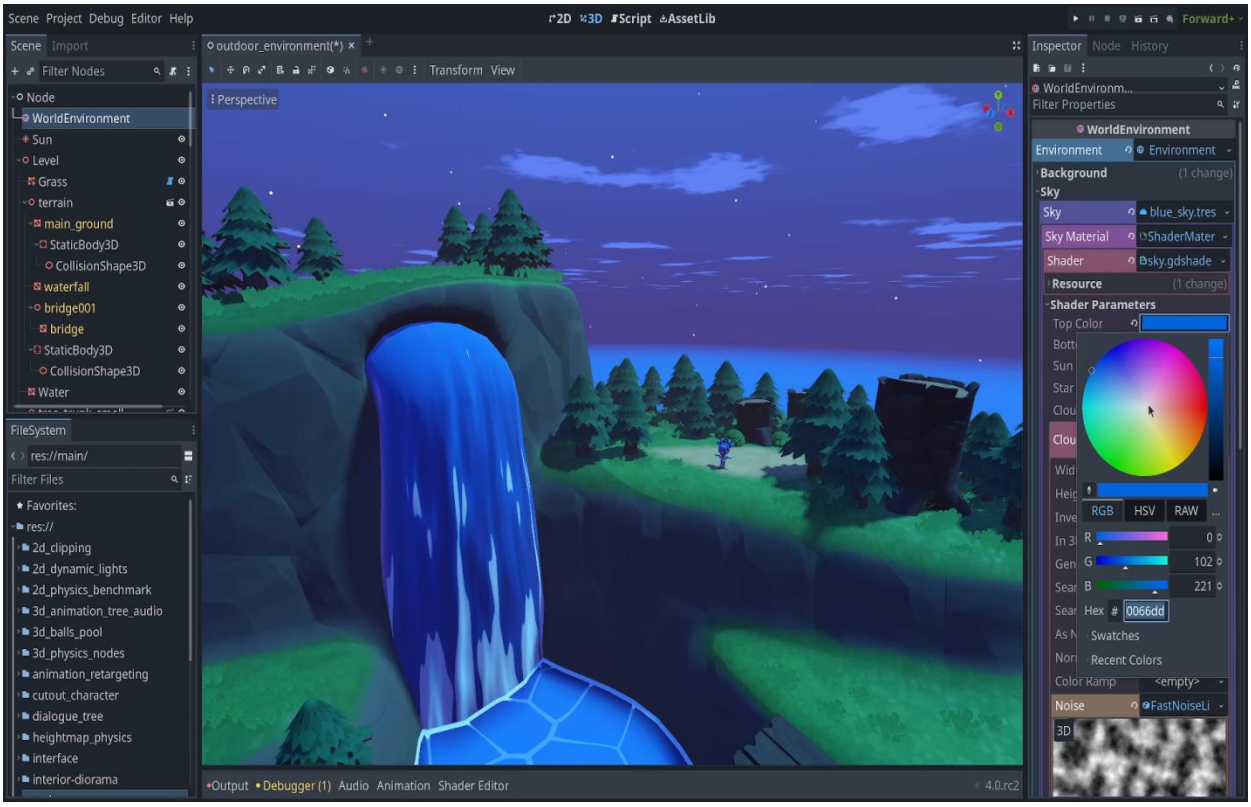


Рисунок 4: интерфейс движка Godot Engine.

3) Ren'Py: бесплатный, свободный и открытый движок для создания как некоммерческих, так и коммерческих визуальных романов (графических квестов с диалоговой системой) в 2D-графике.

Платформы: Windows, macOS, Linux

Знания языков программирования: Ren'Py и Python.

Стоимость: полностью бесплатно.

Достоинства и недостатки:

- Открытый исходный код;
- Простота использования;
- Широкий спектр возможностей;
- Кросс-платформенность;
- Ограниченная производительность (3D объекты недоступны);
- Отсутсвие официальной документации;
- Зависимость от сообщества.

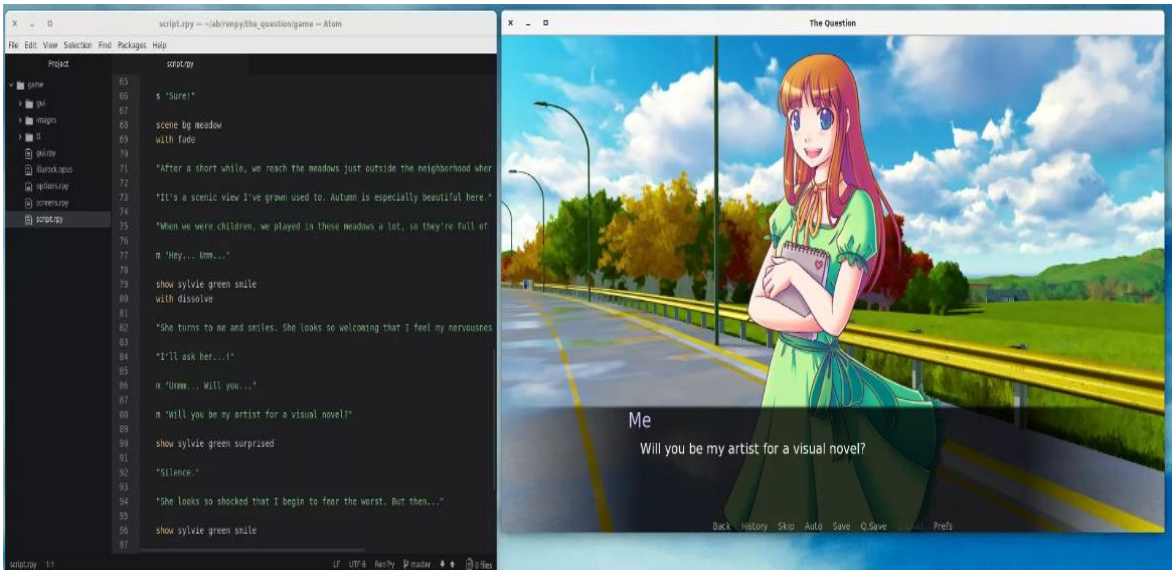


Рисунок 5: интерфейс движка Ren’Py

2. Этапы создания игры в крупных компаниях

2.1. Концептирование

На этом первом шаге команда придумывает концепцию игры, и проводит начальную проработку игрового дизайна. Главная цель данного этапа – это геймдизайнерская документация, включающая в себя **Vision** (развернутый документ, описывающий игру, как конечный бизнес-продукт) и **Concept Document** (начальную проработку всех аспектов игры).

2.2. Прототипирование

Важный этап проектирования любой игры – это создание прототипа. То, что хорошо выглядит «на бумаге», совершенно не обязательно будет интересно в реальности. Прототип реализуется для оценки основного игрового процесса, проверки различных гипотез, проведения тестов игровых механик, для проверки ключевых технических моментов.

2.3. Вертикальный срез

Цель Вертикального среза – получить минимально возможную полноценную версию игры, включающую в себя полностью реализованный основной игровой процесс. При этом высокое качество проработки обязательно нужно воплотить только для тех игровых элементов, которые существенно влияют на восприятие продукта.

2.4. Производство контента

На этом этапе производится достаточное количество контента для первого запуска на внешнюю аудиторию. Реализуются все фичи, запланированные к закрытому бета-тестированию. Это наиболее продолжительный этап, который может занимать, для крупных клиентских проектов год и более.

2.5. Закрытое бета-тестирование

На этапе CBT продукт впервые демонстрируется достаточно широкой публике, хотя и лояльной продукту или компании. Среди наиболее важных задач на этом этапе выступают: поиск и исправление гейм-дизайнерских ошибок, проблем игровой логики и устранение критических багов.

2.6. Открытое бета-тестирование

На этом этапе продолжается тестирование игры, но уже на широкой аудитории. Идет оптимизация под большие нагрузки. Игра должна быть готова для приема большого трафика. В игре реализован биллинг (технологизированный процесс выставления счетов и взимания платы за товары или услуги) и принимаются платежи.

2.7. Релиз игры

Команда разработки на этом этапе занимается исправлением технических багов, выявляемых в процессе эксплуатации и оптимизацией продукта. Геймдизайнеры занимаются тонкой настройкой геймплея под реальную ситуацию в игровом мире (особенно актуально для MMO проектов). Также реализует различные внутриигровые фичи, поддерживающие новые монетизационные схемы. И конечно идет разработка и интеграция в продукт нового контента, поддерживающего интерес игроков.

3. Этапы разработки собственной игры

3.1. Выбор редактора кода для Python

Среди самых популярных редакторов кода можно выделить:

- **Pycharm;**
- **Visual Studio Code;**
- **Sublime Text.**

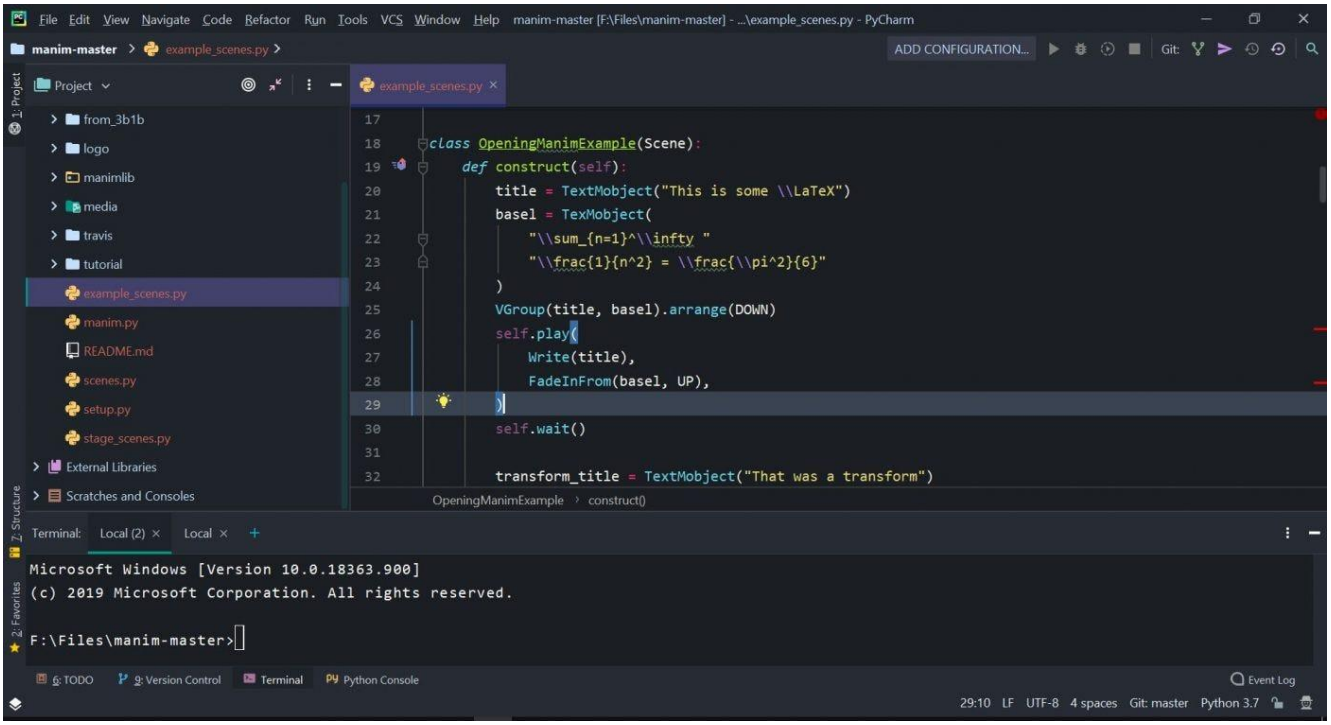
3.1.1. Pycharm

Плюсы IDE (Интегрированная среда разработки):

- Имеет важные встроенные функции.
- Разработана профессионалами специально для Python.
- Поддерживает виртуальные среды Anaconda.

Минусы:

- Основная проблема PyCharm: если у вас недорогой ПК или ноутбук и в нем нет 8 Гб оперативной памяти, то IDE немного притормаживает и работает довольно медленно.



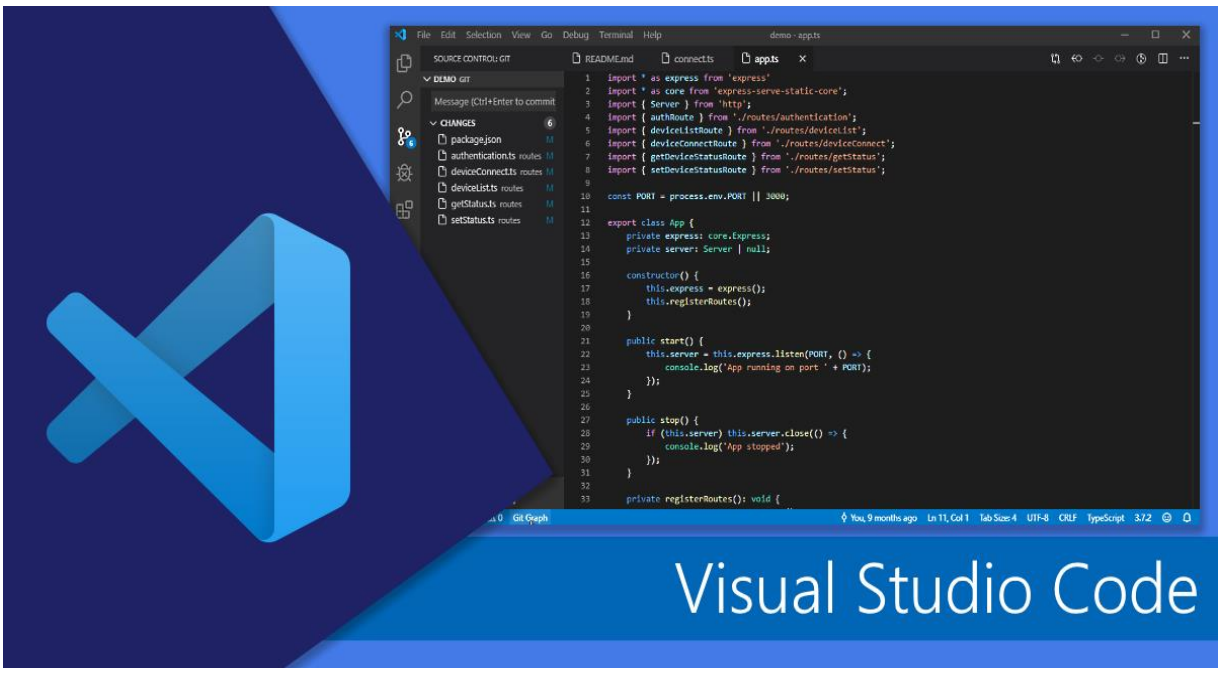
3.1.2. Visual Studio Code

Плюсы IDE:

- Это фантастическая платформа с непрерывными обновлениями.
- Потребляет немного памяти по сравнению с другими громоздкими инструментами разработки.
- Имеет встроенный терминал и прост в использовании.

Минусы:

- Иногда терминал работает не так, как хотелось бы.



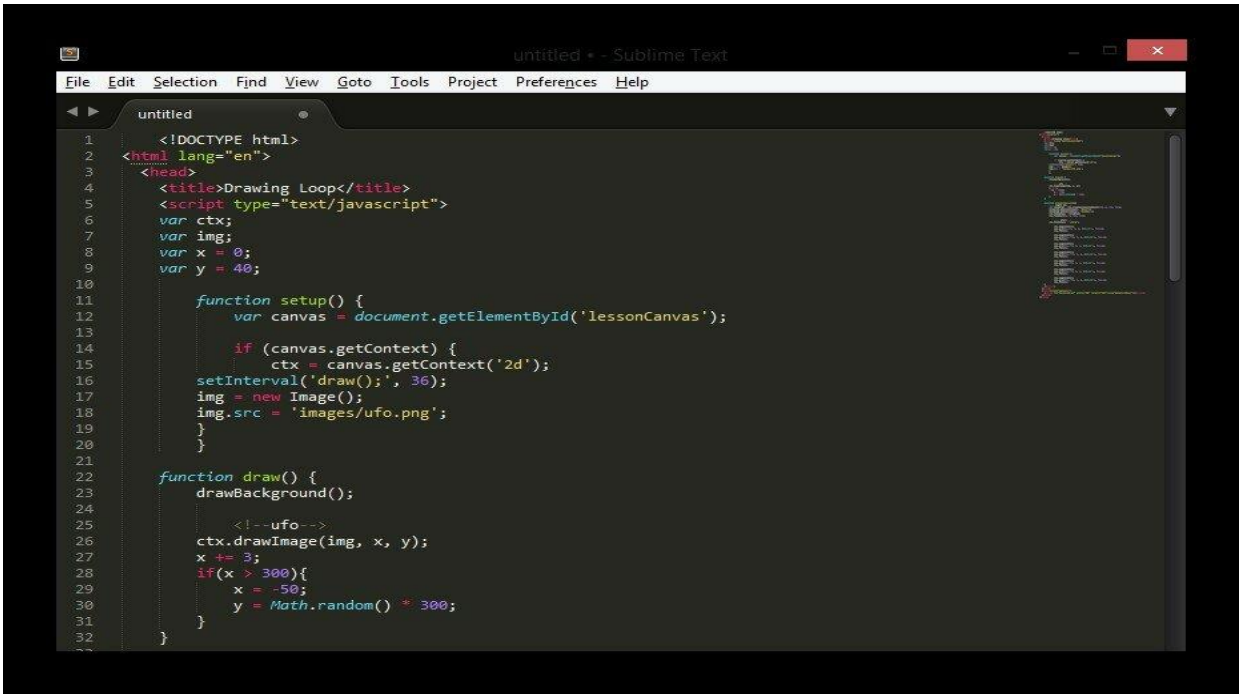
3.1.3. Sublime Text

Плюсы IDE:

- Простой и по большей части бесплатный.
- Тонко настраивается.
- Компактный и эффективный.

Минусы:

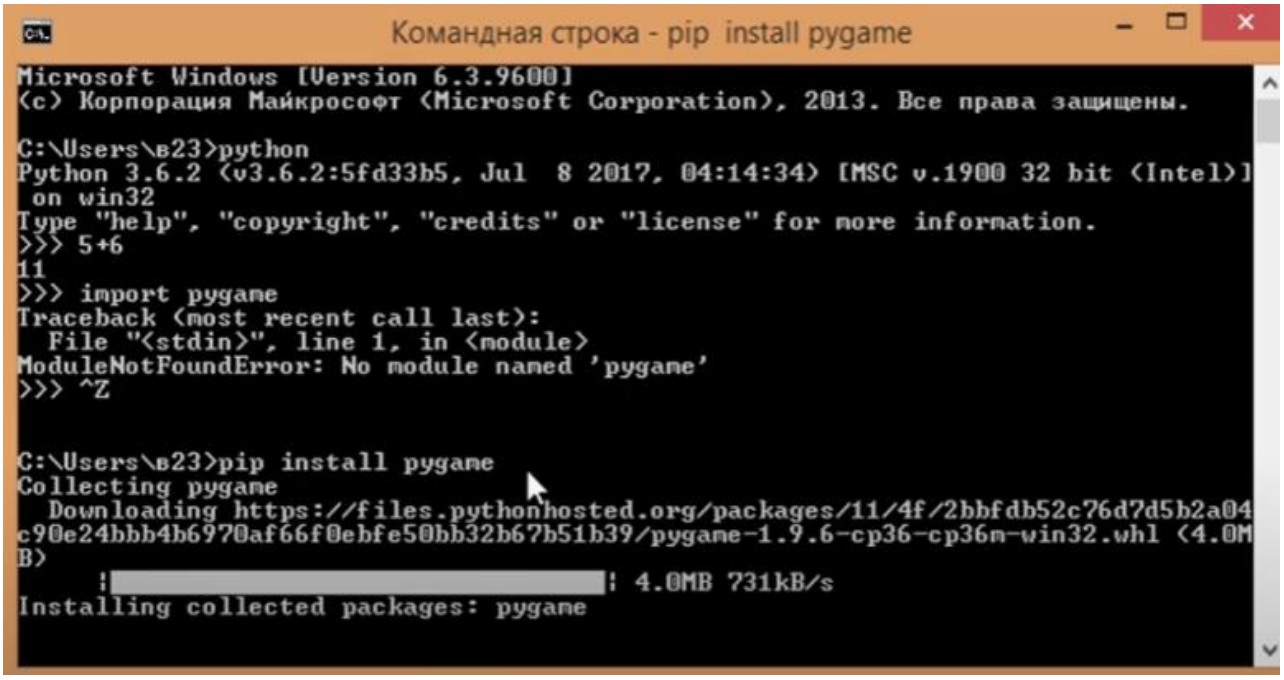
- Для удобства требует дополнительных пакетов.



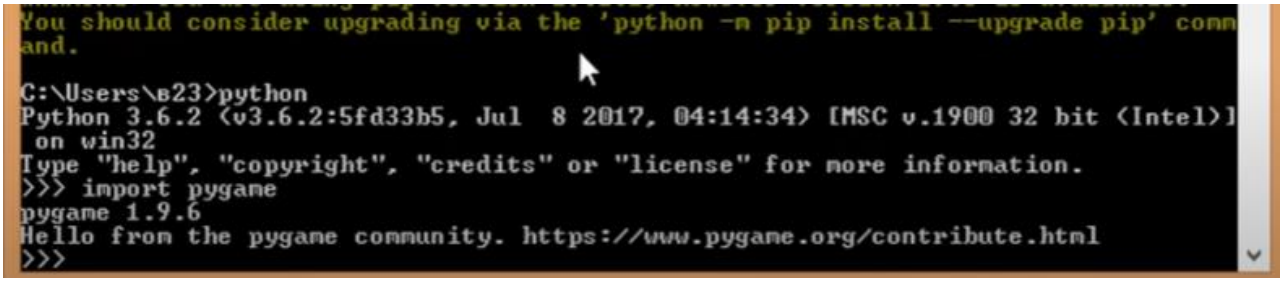
Но для разработки игры я буду использовать PyCharm.

3.2. Установка модуля PyGame

Для установки данного модуля в командной строке Windows введём: pip install pygame.

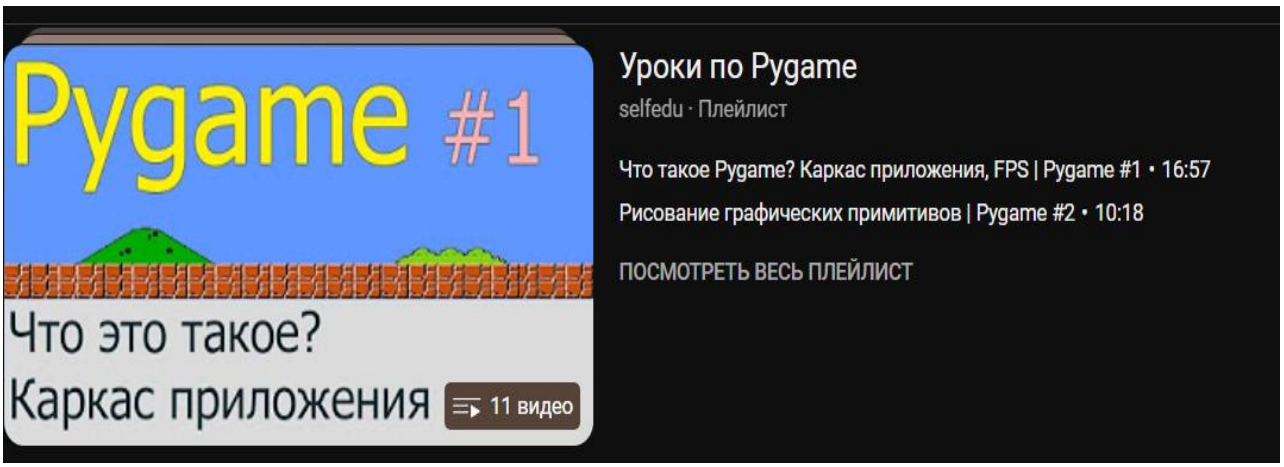


Если модуль установлен, то вас поприветствует сообщества из Pygame, иначе будет получена ошибка, как было показано сверху.



3.3. Изучеине данного модуля

Для изучения данного модуля нам поможет автор канала Youtube selfedu (https://youtube.com/@selfedu_rus?si=k_4AdCyspDawdsZD)



Заключение

В конце проекта будет предоставлен доступ к коду. Исходя из поставленной мною цели, могу сказать, что игра не доведена до полнейшего релиза, так как не хватало некоторых знаний в сфере математики и физики. Также были и недоработки в первоначальной версии игры, среди которых можно выделить отсуствие анимации у враждебных существ; проблема наложения звуковых эффектов и музыкального сопровождения (без вмешательства друг к другу); неумение рисовать. Но процесс разработки меня увлёк и замотивировал после экзаменов изучить поглубже данную профессию.

Почему я решил выбрать именно эту тему? Меня увлекают компьютерные игры и хотелось попробовать роль разработчика игр, но не учёл некоторые моменты, а именно дизайн игры и недостаток знаний по физике. Это и не дало до конца завершить игру.

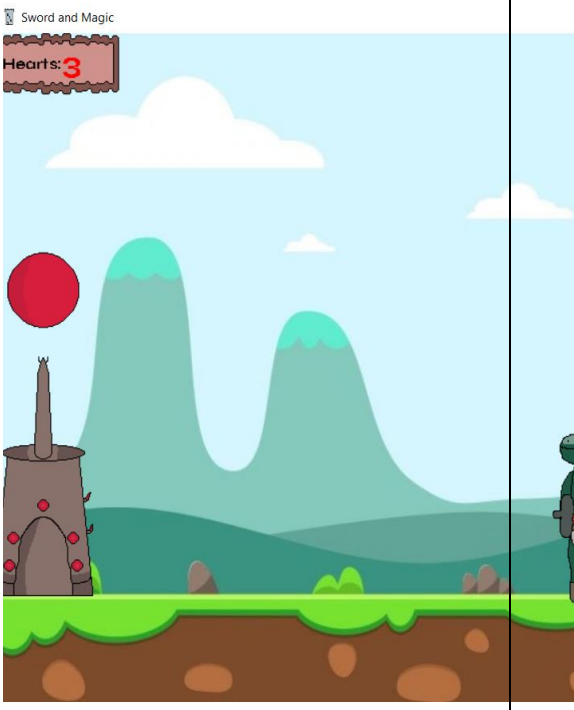
Список литературы

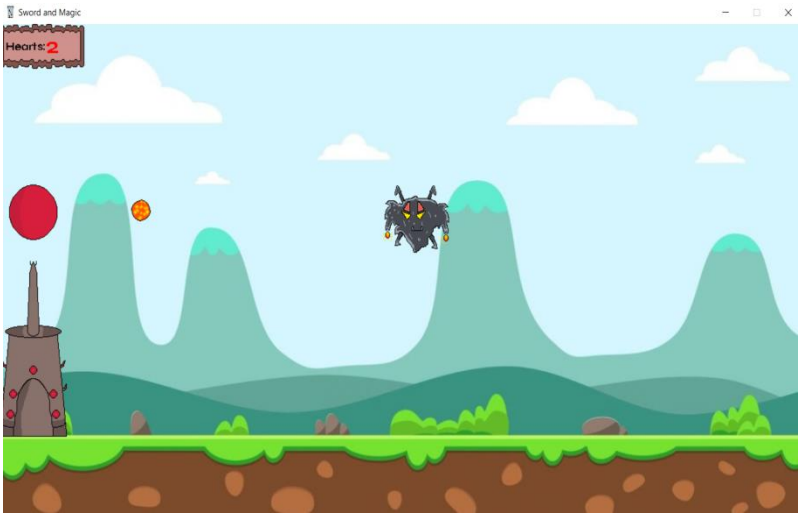
1. Роль компьютерных технологий в настоящее время https://vuzlit.com/1019090/rol_kompyuternyh_tehnologiy_nastoyaschee_vremya?ysclid=lhsw3qxfc0202971048
2. Чем создание игр помогает учёным https://www.playground.ru/misc/news/go_v_nauku_ya_sozdal_kak_igry_pomogayut_uchyonym-246098?ysclid=lhuoc1jkel283119371
3. История создания компьютерных игр 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
4. Языки программирования https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F
5. Движок Gamemaker https://ru.wikipedia.org/wiki/GameMaker
6. Движок Godot Engine https://ru.wikipedia.org/wiki/Godot
7. Движок Ren’Py https://ru.wikipedia.org/wiki/Ren%27Py
8. Выбор среды разработки https://habr.com/ru/companies/skillfactory/articles/521838/
9. Установка модуля Pygame https://www.youtube.com/watch?v=evjpa3v22-U&t=97s

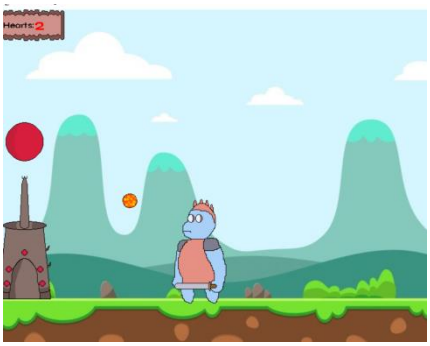
Приложения

Pycharm: https://www.jetbrains.com/pycharm/

Уроки по Pygame: https://youtu.be/RihDgtqRj58?si=rP4WOXb71N5v0Xlo







Код игры:

```
import math
import random
import pygame
pygame.mixer.pre_init(44100, -16, 1, 512)
# параметры
```

```
pygame.init()
heart_user = 3
L, H = 1200, 675
FPS = 60
shrift1 = pygame.font.Font('file_game/Syne-Bold.ttf', 15)
shrift2 = pygame.font.Font('file_game/Syne-Bold.ttf', 30)
screen_set = pygame.display.set_mode((L, H))
background = pygame.image.load('file_game/background.jpg')
icon_game = pygame.image.load('file_game/castle_icon.png')
signboard = pygame.image.load('file_game/signboard.png').convert_alpha()
clock = pygame.time.Clock()
pygame.display.set_icon(icon_game)
pygame.display.set_caption('Sword and Magic')
boom_sound = pygame.mixer.Sound('file_game/vzriv.mp3')



# Классы для взаимодействия
class FireBall:
    def __init__(self, x, y):
        self.pos = (x, y)
        x1, y1 = pygame.mouse.get_pos()
        self.diff_mouse = (x1 - x, y1 - y)
        euclideanstatement = math.hypot(*self.diff_mouse)


        if euclideanstatement == 0.0: self.diff_mouse = (0, -1)


        else: self.diff_mouse = (self.diff_mouse[0] / euclideanstatement, self.diff_mouse[1] /
euclideanstatement)


        angle = math.degrees(math.atan2(-self.diff_mouse[1], self.diff_mouse[0]))
        self.fire_ball = pygame.image.load('file_game/fire_ball.png').convert_alpha()
```

```
        self.fire_ball = pygame.transform.rotate(self.fire_ball, angle)
        self.fire_ball_rect = self.fire_ball.get_rect()

    def update_fire_ball(self):
        self.pos = (self.pos[0] + self.diff_mouse[0] * obj.speed_fire,
                    self.pos[1] + self.diff_mouse[1] * obj.speed_fire)

    def draw_fire_ball(self):
        self.fire_ball_rect = self.fire_ball.get_rect(center=self.pos)
        screen_set.blit(self.fire_ball, self.fire_ball_rect)


class Monsters:
    def __init__(self, x, y, speed, image, type_monster):
        self.x = x
        self.y = y
        self.speed = speed
        self.load_monster = image
        self.type_monster = type_monster
        self.get_rect_monster = self.load_monster.get_rect(center=(self.x, self.y + 100))


class Parameters:
    def __init__(self, H):
        # башня
        self.tower_ = pygame.image.load('file_game/tower_game.png').convert_alpha()
        self.get_tower_ = self.tower_.get_rect(centerx=50, centery=H // 2 + 65)

        # огненные снаряды
        self.fire_ball = pygame.image.load('file_game/fire_ball.png').convert_alpha()
```

```
            self.speed_fire = 10
            self.get_fire_ball = self.fire_ball.get_rect(centerx=50, centery=H // 2 - 85)
            self.pos = (self.get_fire_ball.centerx, self.get_fire_ball.centery)

            # руна
            self.rune_ = pygame.image.load('file_game/rune.png').convert_alpha()



        pygame.time.set_timer(pygame.USEREVENT, 2000)
        obj = Parameters(H)
        # Огненный шар
        fire_balls = []
        start_pos_ball = (obj.pos[0], obj.pos[1])

        # Монстры
        data_monsters = ({'screen': 'monster1.png', 'type': 'ground'},
                         {'screen': 'monster2.png', 'type': 'ground'},
                         {'screen': 'monster3.png', 'type': 'air'})
        monsters_load = [pygame.image.load('file_game/' + data_file['screen']).convert_alpha() for
data_file in data_monsters]
        monsters = []



        # настройка монстров
        def monster_option():
            x = random.randint(1000, L)
            y_ground_monsters = 360
            y_air_monsters = random.randint(100, 200)
            speed = random.randint(5, 7)
            index_monster = random.randint(0, len(data_monsters) - 1)
```

```
            if data_monsters[index_monster]['type'] == 'air': monsters.append(Monsters(x, y_air_monsters, speed, monsters_load[index_monster], data_monsters[index_monster]['type']))

            else: monsters.append(Monsters(x, y_ground_monsters, speed, monsters_load[index_monster], data_monsters[index_monster]['type']))



        while True:

            # события

            for event in pygame.event.get():

                if event.type == pygame.QUIT: exit()


                elif event.type == pygame.MOUSEBUTTONDOWN and event.button == 1: fire_balls.append(FireBall(*start_pos_ball))


                elif event.type == pygame.USEREVENT: monster_option()


            # обновление шара

            for ball in fire_balls:

                ball.update_fire_ball()


                # if not start.create_window.get_rect().collidepoint(ball.pos):

                if 1200 < ball.pos[0] or 0 > ball.pos[0] or ball.pos[1] > 550 or ball.pos[1] < 0: fire_balls.remove(ball)


            # обновление монстров

            for up_monster in monsters:

                if up_monster.get_rect_monster.x >= 100:

                    up_monster.get_rect_monster.x -= up_monster.speed


                if up_monster.get_rect_monster.x <= 105:

                    monsters.remove(up_monster)

                    heart_user -= 1
```

```
        if heart_user == 0: exit()


        # контроллер столкновений с объектами
        for ball in fire_balls:
            for monster in monsters:
                if ball.fire_ball_rect.collidepoint(monster.get_rect_monster.center) or ball.fire_ball_rect.collidepoint(monster.get_rect_monster.centerx, monster.get_rect_monster.top + 50) or ball.fire_ball_rect.collidepoint(monster.get_rect_monster.centerx, monster.get_rect_monster.bottom - 20):
                    monsters.remove(monster)
                    boom_sound.play()
                    fire_balls.remove(ball)


        # добавление объектов игры
        screen_set.blit(background, (0, 0))
        screen_set.blit(obj.tower_, obj.get_tower_)
        screen_set.blit(obj.rune_, (obj.get_tower_.x, obj.get_tower_.y - 50))
        screen_set.blit(signboard, (0, 0))


        # добавление шрифтов
        render1 = shrift1.render('Hearts: ', 0, 'BLACK')
        render2 = shrift2.render(str(heart_user), 0, 'RED')
        signboard.blit(render1, render1.get_rect(center=(40, 30)))
        screen_set.blit(render2, render1.get_rect(center=(100, 20)))


        # добавление объектов с помощью цикла
        for dr_monster in monsters: screen_set.blit(dr_monster.load_monster, dr_monster.get_rect_monster)
        for ball in fire_balls: ball.draw_fire_ball()


        pygame.display.update()
        clock.tick(FPS)
```