IX Всероссийская (с международным участием) научная конференция учащихся имени Н.И. Лобачевского

Секция «Проектная деятельность школьных IT-команд»

Создание сюжетной игры на языке Python с использованием библиотеки Pygame

Хуснуллин Марсель, 9 класс
Крахмальников Илья, 9 класс

Направляющие организации:

МБОУ “Гимназия №125”, г. Казань
Лицей имени Н. И. Лобачевского при КФУ, г. Казань

Научный руководитель:
учитель математики и информатики
высшей квалификационной
категории Николаев Э.Н.

Казань, 2024

Оглавление

Введение

В наше время видеоигры являются столь же популярным развлечением детей, как и, например, футбол в прошлом. И данное увлечение далеко не плохое, ведь при помощи игр можно обучаться и познавать много нового.

Сейчас в сфере видеоигр большинство из них направлено на развлечение и заработок компаний. Такой подход далеко не плох для больших компаний. Но для обычных пользователей это проблема.

Как известно целевая аудитория видеоигр — это дети. Их привлекает приятный дизайн и интересный развлекательный геймплей. Но это увлечение порой отвлекает детей от учебы.

Мы придерживаемся этого мнения и поэтому мы решили создать собственную игру, которая поможет детям не забывать банальных основ арифметики и логики и ускорить реакцию, с сюжетом и внутренним миром при помощи Python и библиотеки pygame.

Наш проект — это уникальная компьютерная игра, созданная с целью предложить игрокам захватывающий и увлекательный игровой опыт, который приносит радость и удовлетворение. Наша главная цель заключается в создании игры, которая не только развлекает, но и вдохновляет, заставляя игроков погрузиться в захватывающий мир исследования и приключений.

Идея проекта вдохновлена инди игрой Toby Fox’a “Undertale”. Например, дизайн главного героя позаимствован у Фриск – главной героини “Undertale”. Также, некоторая музыка была взята из этой игры, т. к. треки этой игры не находятся под авторскими правами и находятся в открытом доступе.

Цель: изучить python и pygame, продумать сюжет игры, создать дизайн и продумать логику. Привить интерес к арифметике, логике. Развить внимательность и скорость принятия решений путем задач внутри игры. Создание захватывающего приключенческого игрового опыта, вдохновленного игрой Undertale.

Задачи: сформировать умение ставить учебную задачу, выделять и осознавать, что усвоено и что еще подлежит усвоению; развить коммуникативные навыки и навыки сотрудничества в процессе групповой работы.

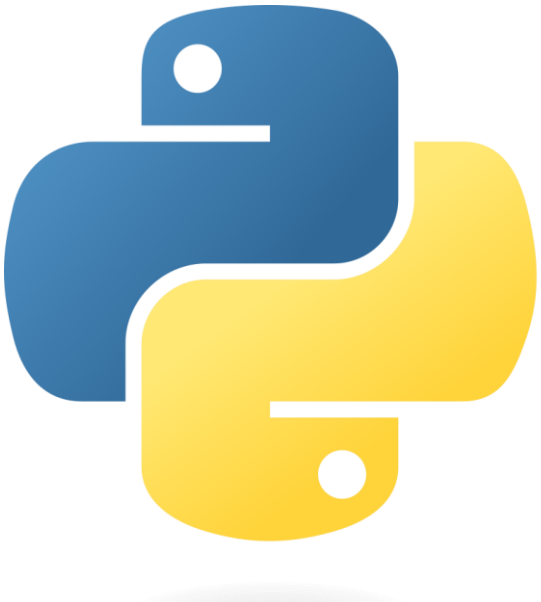
1.Основы Python и Pygame

Python и Pygame представляют собой мощный инструментарий для разработки игр, который позволяет программистам и дизайнерам создавать увлекательные проекты с нуля. Рассмотрим более подробно, как каждый из этих инструментов влияет на разработку игр.

Python — это универсальный язык программирования, который отлично подходит для разработки игр благодаря своей простоте и гибкости. Он предлагает широкий набор библиотек и инструментов, которые упрощают процесс создания игровой логики, обработки ввода и вывода, а также управления ресурсами игры.

Основные преимущества Python для разработки игр:

1. Простота и читаемость кода: Python имеет простой и понятный синтаксис, который делает код более читаемым и легким для понимания, что ускоряет процесс разработки.
2. Богатый экосистем: Python обладает огромным количеством библиотек и фреймворков, которые облегчают разработку игр, включая библиотеки для работы с графикой, звуком, искусственным интеллектом и многим другим.
3. Кроссплатформенность: Python поддерживает множество операционных систем, что позволяет создавать игры, которые могут быть запущены на различных устройствах без необходимости изменения исходного кода.



Pygame — это библиотека, которая предоставляет разработчикам широкие возможности для создания игр, включая работу с графикой, звуком, управлением вводом и многим другим. Она основана на библиотеке SDL (Simple DirectMedia Layer), что обеспечивает высокую производительность и кроссплатформенность.

Основные возможности Pygame:

1. Работа с графикой: Pygame предоставляет инструменты для создания и отображения графики в игре, включая рисование спрайтов, текстур, анимаций и многое другое.

2. Звук и музыка: Библиотека позволяет воспроизводить звуки и музыку в игре, что создает более полноценный игровой опыт для игроков.

3. Управление вводом: Pygame обеспечивает возможность обработки пользовательского ввода с клавиатуры, мыши, геймпадов и других устройств, что позволяет создавать интерактивные игры с разнообразным управлением.

Все эти возможности делают Python и Pygame идеальным выбором для разработки игр, независимо от уровня опыта разработчика.

2.Выбор названия игры и структуры сюжета

Выбор подходящего названия для игры — это важный этап, который должен отражать её суть и привлекать внимание потенциальных игроков. В случае нашей игры, которая сочетает в себе элементы приключения, обучения и фэнтези, название "Entangled-Tale" было выбрано с учетом этих аспектов. Оно отражает сложность и загадочность мира игры, где главный герой попадает в запутанную сказку, полную тайн и опасностей.

Структура сюжета игры предполагает разделение на три акта, каждый из которых представляет собой уникальный мир с собственными задачами и вызовами для главного героя:

I Акт: Подземелье

Главный герой начинает свое путешествие в загадочном подземелье, где сталкивается с магом русского языка и физики. Здесь он должен решать задачи на знание основ арифметики и физики, чтобы продвигаться дальше по сюжету.

II Акт: Волшебный лес

После покорения подземелья герой отправляется в волшебный лес, где его ждут испытания с биологией и лабиринты. Здесь он встречает мага математики, который предлагает ему решать арифметические задачи, чтобы продвигаться дальше.

III Акт: Небеса

Завершая свое приключение, герой достигает небес, где ему предстоит выполнить квест по сборке яблок и стать невидимым для прохождения мимо рыцаря. Здесь его ожидает финальное сражение с боссом - магом математики.

Каждый акт содержит ряд уникальных локаций, которые добавляют разнообразие и интерес к игре. Для перемещения между локациями используются двери, которые подчеркивают путешествие главного героя по загадочному миру Entangled-Tale.

Также в игре будет реализованы авто-сохранения в разных местах для создания возможности вернутся в нужное место после смерти.

3.Разработка графики и дизайна игры

При выборе графического стиля для нашей игры мы обратили внимание на преимущества 2D графики в контексте использования библиотеки Pygame. В отличие от создания трехмерного мира, использование 2D графики существенно упрощает процесс разработки и обладает рядом других преимуществ:

1. Удобство и быстрота разработки: Использование 2D спрайтов позволяет сосредоточиться на дизайне и деталях игровых объектов, минимизируя время, затраченное на программирование сложных трехмерных моделей и анимаций.

2. Простота в оптимизации: 2D графика обычно требует меньше вычислительных ресурсов по сравнению с 3D, что делает игру более доступной для широкого круга игроков, включая тех, у кого слабые компьютеры.

3. Возможность создания уникального стиля: Пиксель-арт, выбранный нами для визуализации игровых объектов, обладает своей особенной эстетикой, которая привлекает внимание и создает неповторимый антураж игры.

Для создания графики и дизайна игры мы будем использовать спрайты - графические объекты, которые представляют собой растровые изображения. Спрайты будут использоваться для представления главного героя, неигровых персонажей, объектов взаимодействия, а также для создания заднего фона и интерфейсных элементов.

Важным инструментом для создания спрайтов в нашем проекте будет web-сервис pixilart.com, который предоставляет возможность создания пиксель-арт изображений непосредственно в браузере. Этот сервис обеспечит удобство и доступность при создании графики для нашей игры.

Изображения, которые нам необходимо создать включают в себя:

- Изображения игрока (с видом со всех сторон и анимированные кадры)
- Карты для всех локаций
- Спрайты неигровых персонажей
- Спрайты интерактивных объектов
- Заставки
- Меню Game Over
- Субтитры

Этот процесс разработки графики и дизайна игры играет ключевую роль в создании неповторимой атмосферы и визуальной привлекательности игрового мира, что сделает нашу игру более увлекательной для игроков.

4.Реализация структуры

Для начала работы над игрой необходимо создать новый проект в среде разработки PyCharm и подключить его к системе контроля версий Git для удобного совместного ведения проекта. Создадим также новый репозиторий на платформе GitHub и загрузим наш проект в него.

После настройки проекта начнем с создания основного файла main.py. Также создадим директорию data, в которой будут храниться все ресурсы игры.

4.1. Импорт необходимых библиотек

Для работы с игровыми объектами, базами данных SQL, генерации случайных событий и обращения к системному времени в процессе игры, импортируем следующие библиотеки:

- pygame: для работы с игровыми объектами.
- sqlite3: для использования баз данных SQL в игре.
- random: для создания случайных событий в игре.
- datetime: для обращения к системному времени в процессе игры.
- sys: для полного закрытия программы.
- os: для работы с файлами.

4.2. Функция загрузки изображений

Для загрузки изображений создадим функцию “load_image”, которая будет проверять корректность имени файла и наличие файла в директории data. В случае проблем с файлом, функция выведет ошибку.

4.3. Главный цикл игры

Основной цикл while будет запускаться вместе с игрой и инициализировать Pygame.

```
if __name__ == '__main__':
    pygame.init()
```

4.4. Экраны игры

Создадим экраны игры, такие как начальный экран и меню инструкции (рис. 1 и рис. 2), а также меню выбора сохранений (рис. 3). При нажатии на любую клавишу на начальном экране игрок переходит к меню выбора сохранений. Они будут реализованы в функции “start_screen”.

Эти экраны помогут игроку ориентироваться в игре и выбирать необходимые опции.

4.5. Автоматические сохранения

Сохранения будут автоматическими и индицируются музыкальным сопровождением. Если при переходе на новую локацию музыка начнется с начала, это будет означать, что произошло сохранение.

Таким образом, мы создали основу для разработки игры, включая настройку проекта, импорт необходимых библиотек, функцию загрузки изображений, главный цикл игры и экраны игры. Эти шаги помогут нам структурировать проект и приступить к созданию основных игровых механик и функциональности.

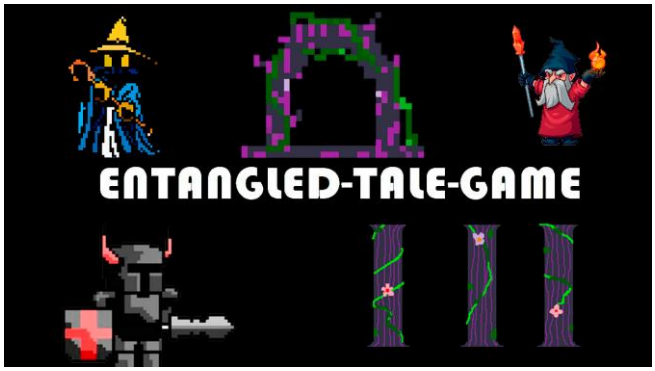


рис. 1 - начальный экран

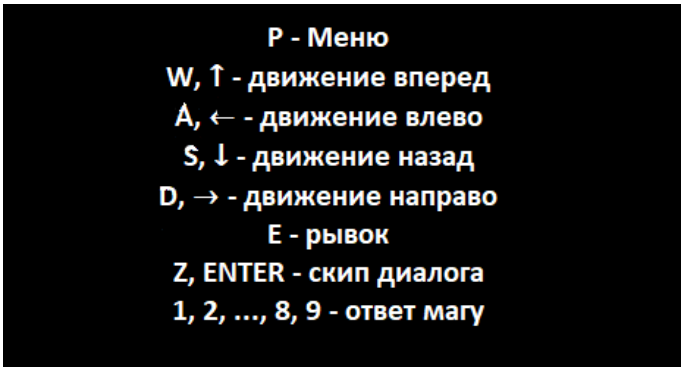


рис. 2 - инструкция управления

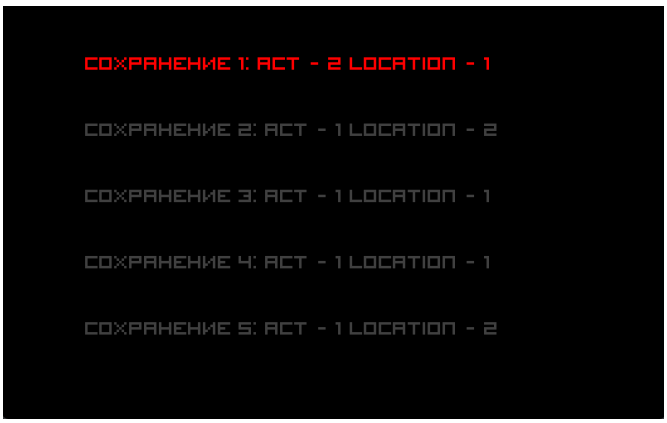


рис. 3 - меню выбора сохранений

5.Реализация игрока, перемещения и прорисовки локаций

Для воплощения игрового персонажа и перемещения по локациям мы создадим соответствующие классы и функции.

5.1. Игрок

Для работы с игровым персонажем создадим класс “Player”, включающий три основные функции:

- __init__ (для реализации создания игрока)
- stop (Анимация игрока при остановке)
- update (для обновления игрока и взаимодействия с другими игроками)

5.2. Перемещение игрока

Для обработки перемещения игрока создадим обработчик нажатия на клавиши:

```
keys = pygame.key.get_pressed() - для отслеживания нажатых клавиш
player.update(keys[pygame.K_UP] or keys[pygame.K_w],
  keys[pygame.K_DOWN] or keys[pygame.K_s],
  keys[pygame.K_LEFT] or keys[pygame.K_a],
  keys[pygame.K_RIGHT] or keys[pygame.K_d])
```

В функции “update” класса “Player” будем принимать переменные, хранящие информацию о нажатии клавиш: move_up, move_down, move_left, move_right

В дальнейшем изменяем положение игрока:

```
self.rect.x -= self.run
self.x -= self.run
```

А также анимацию игрока:

```
image=load_image(f'players_image/m.c.left_walk_{self.step}.png')
```

5.3. Прорисовка локаций

Для прорисовки локаций создадим функции, принимающие аргумент для обозначения локации в акте: “a1_location”, “a2_location”, “a3_location”. В этих функциях будем описывать создание локаций в соответствии с концепцией игры.

Таким образом, мы реализовали основные компоненты для работы с игровым персонажем, их перемещением и прорисовкой локаций в игре. Эти шаги позволят нам создать интерактивное и захватывающее игровое пространство для игрока.

6.Программирование спрайтов и актов

В нашем коде большинство классов соответствуют определённым спрайтам, начиная от класса игрока, заканчивая классами боссов 1, 2 и 3 актов. В функциях же мы контролируем различные события и создаём скрипты для уменьшения нагрузки на основной цикл. Вот некоторые классы и функции:

```
25 usages   Mar1o +1
class Door(pygame.sprite.Sprite):  # Дверь
    Explain | Test | Document | Fix | Ask
    Mar1o +1
    def __init__(self, pos_x, pos_y, act, tip):
        super().__init__(all_sprites)
        if act == 1:
            image_path = load_image(f'doors/door_act1_{tip}.png')
        elif act == 2:
            image_path = load_image('doors/door_act2.png')
        if act == 3:
            image_path = load_image('doors/door_act3.png')
        if tip == 1:
            self.image = pygame.transform.scale(image_path, size: (120, 96))
        elif tip == 2:
            self.image = pygame.transform.scale(image_path, size: (112, 67))
        elif tip == 3:
            self.image = pygame.transform.scale(image_path, size: (120, 120))
        self.rect = self.image.get_rect().move(pos_x, pos_y + 20)
        self.mask = pygame.mask.from_surface(self.image)
```

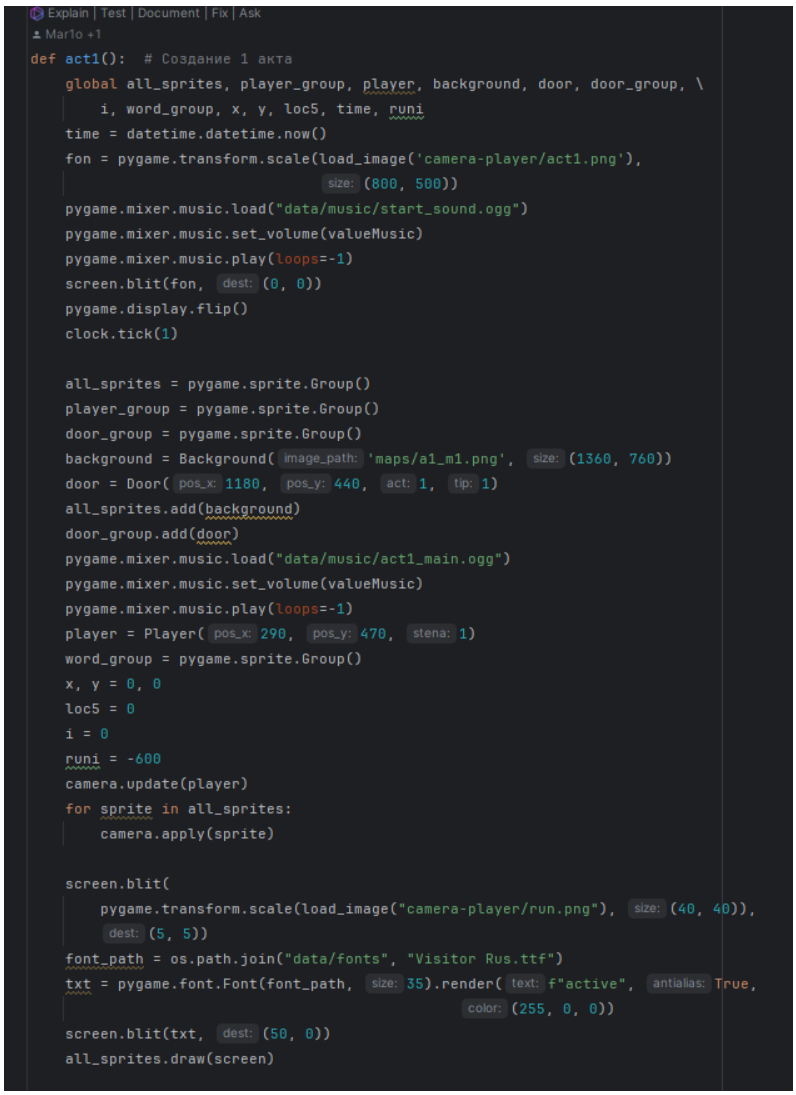
Класс двери

```
class Boss_act1(pygame.sprite.Sprite):  # Босс 1 акт
    image = load_image('npc/wizard_physics.png')
    image = pygame.transform.scale(image, size: (300, 200))

    Explain | Test | Document | Fix | Ask
    124476
    def __init__(self, pos_x, pos_y):
        super().__init__(all_sprites)
        self.image = Boss_act1.image
        self.rect = self.image.get_rect().move(pos_x, pos_y)
        self.mask = pygame.mask.from_surface(self.image)
```

Класс босса 1 акта

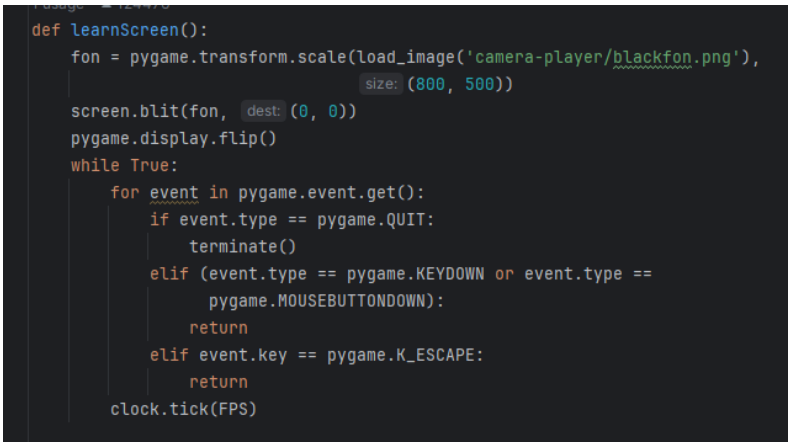
```
7 usages   Mar1o +1
class Sign(pygame.sprite.Sprite):  # Табличка
    image = load_image('objects/sign.png')
    image = pygame.transform.scale(image, size: (51, 54))

    Explain | Test | Document | Fix | Ask
    Mar1o +1
    def __init__(self, x_pos, y_pos):
        super().__init__( *groups: all_sprites, sign_group)
        self.image = Sign.image
        self.rect = self.image.get_rect().move(x_pos, y_pos)
```

Класс табличек



```
Explain | Test | Document | Fix | Ask
Mar1o +1
def act1():  # Создание 1 акта
    global all_sprites, player_group, player, background, door, door_group, \
        i, word_group, x, y, loc5, time, runi
    time = datetime.datetime.now()
    fon = pygame.transform.scale(load_image('camera-player/act1.png'),
                                 size: (800, 500))
    pygame.mixer.music.load("data/music/start_sound.ogg")
    pygame.mixer.music.set_volume(valueMusic)
    pygame.mixer.music.play(loops=-1)
    screen.blit(fon,  dest: (0, 0))
    pygame.display.flip()
    clock.tick(1)

    all_sprites = pygame.sprite.Group()
    player_group = pygame.sprite.Group()
    door_group = pygame.sprite.Group()
    background = Background( image_path: 'maps/a1_m1.png',  size: (1360, 760))
    door = Door( pos_x: 1180,  pos_y: 440,  act: 1,  tip: 1)
    all_sprites.add(background)
    door_group.add(door)
    pygame.mixer.music.load("data/music/act1_main.ogg")
    pygame.mixer.music.set_volume(valueMusic)
    pygame.mixer.music.play(loops=-1)
    player = Player( pos_x: 290,  pos_y: 470,  stena: 1)
    word_group = pygame.sprite.Group()
    x, y = 0, 0
    loc5 = 0
    i = 0
    runi = -600
    camera.update(player)
    for sprite in all_sprites:
        camera.apply(sprite)

    screen.blit(
        pygame.transform.scale(load_image("camera-player/run.png"),  size: (40, 40)),
         dest: (5, 5))
    font_path = os.path.join("data/fonts", "Visitor Rus.ttf")
    txt = pygame.font.Font(font_path,  size: 35).render( text: f"active",  antialias: True,
                                                         color: (255, 0, 0))
    screen.blit(txt,  dest: (50, 0))
    all_sprites.draw(screen)
```

Функция начала 1 акта



```
def learnScreen():
    fon = pygame.transform.scale(load_image('camera-player/blackfon.png'),
                                 size: (800, 500))
    screen.blit(fon,  dest: (0, 0))
    pygame.display.flip()
    while True:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                terminate()
            elif (event.type == pygame.KEYDOWN or event.type ==
                  pygame.MOUSEBUTTONDOWN):
                return
            elif event.key == pygame.K_ESCAPE:
                return
        clock.tick(FPS)
```

Функция экрана-инструкции управления

Остальной же код вы можете просмотреть в коде проекта на github по ссылке из списка литературы, потому что он бы занимал слишком много места (в коде 2366 строк).

7. Реализация дизайна

7.1 Дизайн карт

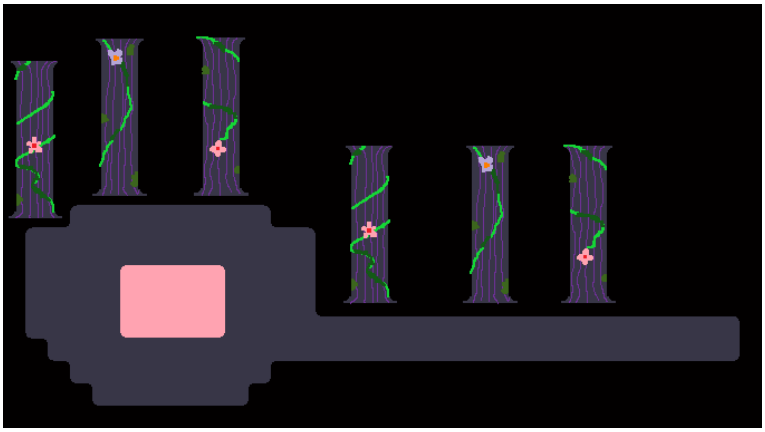


рис. 4 - Акт I, локация №1

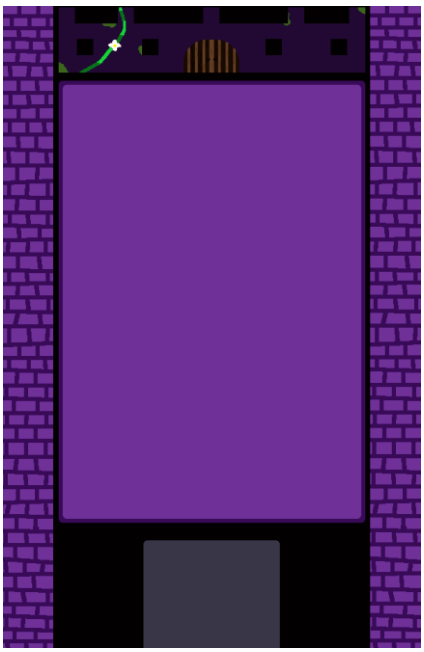


рис. 5 - Акт I, локация №2

рис. 6 - Акт I, локация №3, мини-игра “уклонись от букв”

рис. 7 - Акт I, локация №4 - арифметическая битва 1 акта

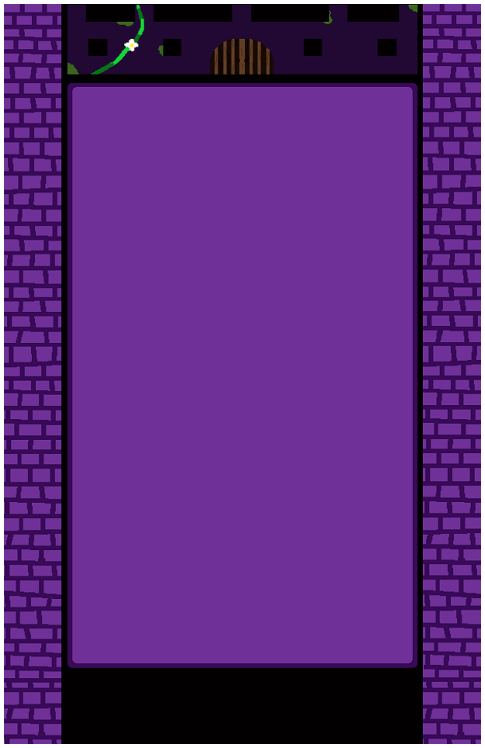


рис. 8 - Акт I, локация №5 - комната перед боссом с табличками.

рис. 9 - Акт I, локация №9 - комната босса 1 акта

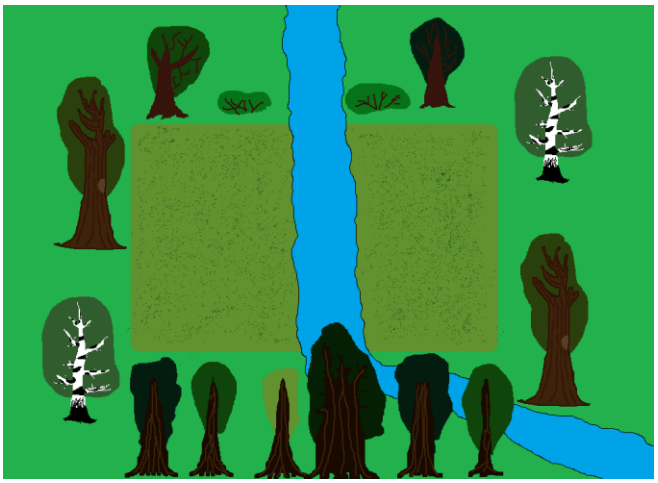


рис. 10 - Акт II, локация №1

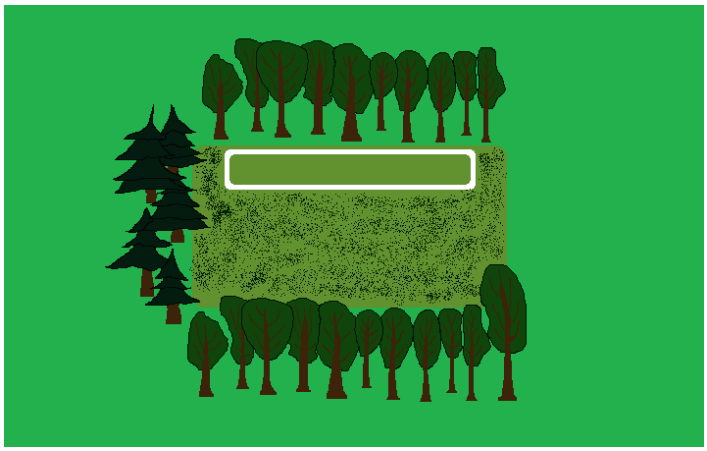


рис. 11 - Акт II, локация №2 - Комната для анатомической игры в ассоциацию

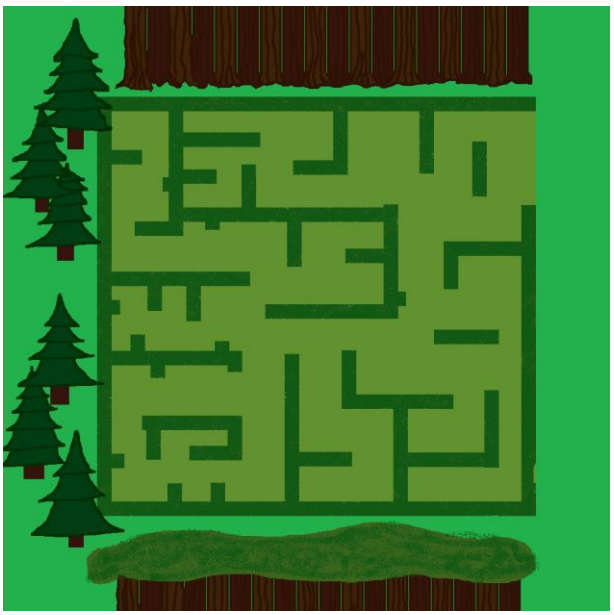


рис. 12 - Акт II, локация №3 - Лабиринт с сундуком для получения монеты

рис. 13 - Акт II, локация №4 - мост для дальнейшего прохождения

рис. 14 - Акт II, локация №5 - комната арифметическая битва 2 акта

рис. 15 - Акт II, локация №6 - комната босса 2 акта

рис. 16 - Акт III, локация №1

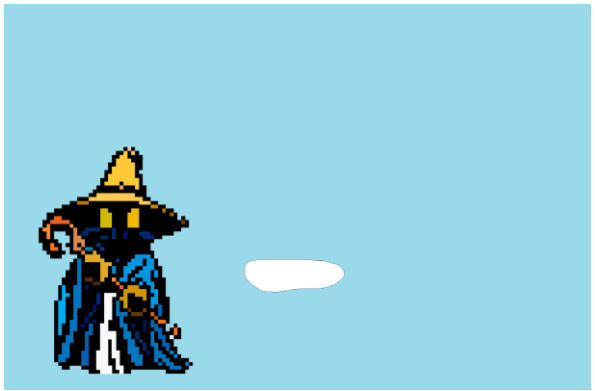


рис. 17 - Акт III, локация №2 арифметическая битва 3 акта

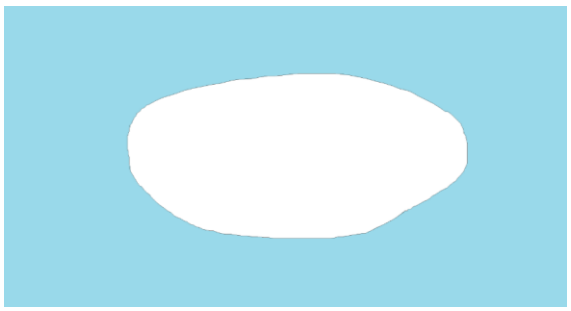


рис. 18 - Акт III, локация №3 - комната босса 3 акта

7.2 Дизайн игрока и неигровых персонажей

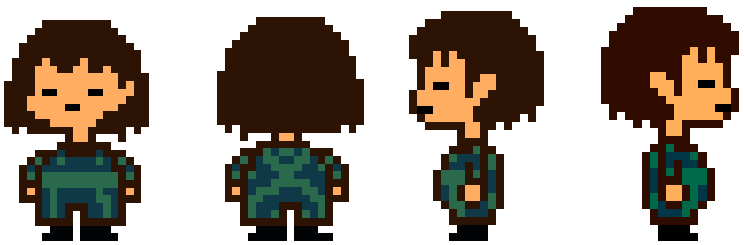


рис. 19 - изображения игрока со всех сторон

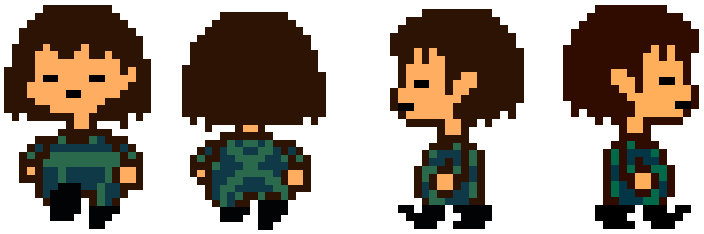


рис. 20 - анимация для перемещения

Дизайн персонажа был вдохновлен главным героем игры Undertale - Фриск

рис. 21 - Рыцарь, защищающий босса 3 акта

рис. 22 - Странник, помогающий игроку в 3 акте

рис. 23 - Маг математики, также босс 3 акта

рис. 24 - Маг природы, также босс 2 акта

рис. 25 - Маг физики, также босс 1 акта

рис. 26 - Маг русского языка, атакует игрока в мини-игре “увернись от букв”

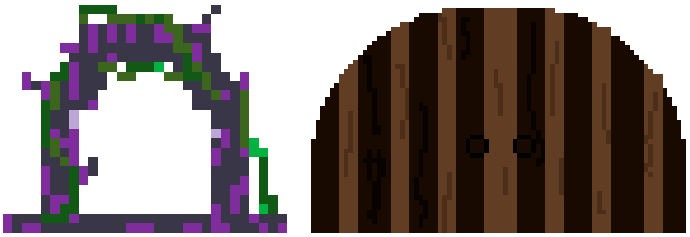


рис. 27 - двери 1 акта

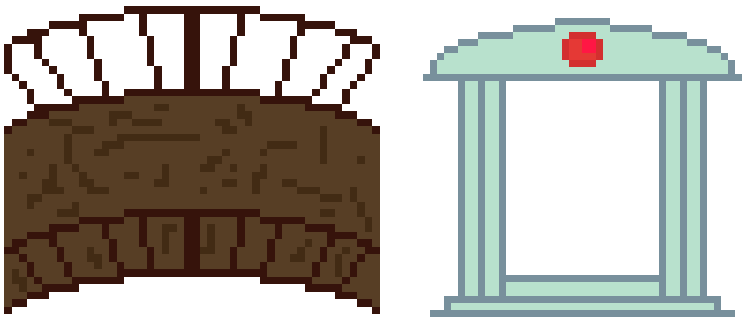


рис. 28 - дверь 2 акта (справа), дверь 3 акта (слева)

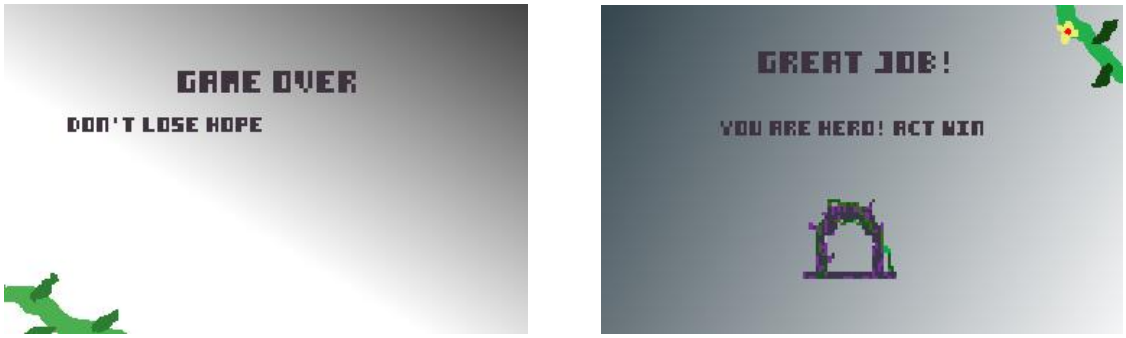


рис. 29 - экраны окончания игры (слева) и победы над боссом (справа).

Все эти изображения находятся в папке data в соответствующих каталогах. Также там есть изображения различных игровых объектов, заставки каждого акта, кнопки для головоломок и мелодии для локаций.

Весь дизайн создавался при помощи сайта www.pixilart.com.

Также мы создали сайт для установки игры:

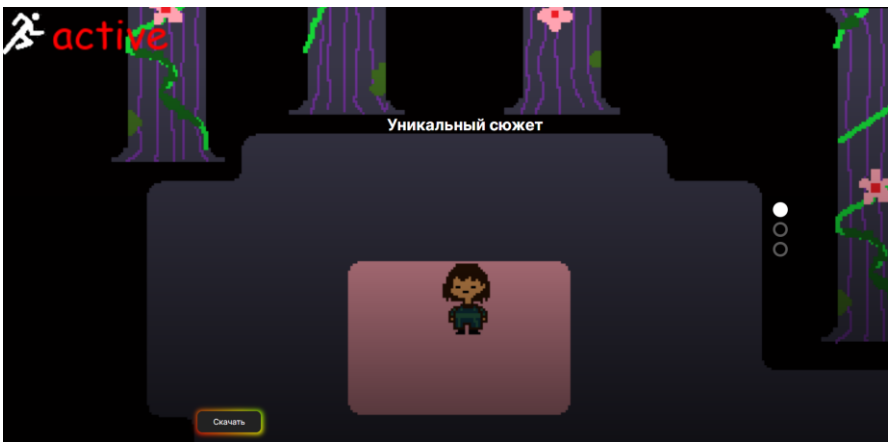


рис. 30 - дизайн сайта

8. Реализация игры

После того, как мы завершили создание всех необходимых элементов дизайна, настало время приступить к реализации игры.

8.1 реализация геймплея

АКТ I

При запуске акта отображается экран, соответствующий данному акту. Игрок появляется на карте первой локации (рис. 4). При движении игрока меняется изображение его спрайта в зависимости от направления (рис. 19 и 20). Коллизия со стенами реализована через проверку цветов пикселей вокруг игрока.

В конце комнаты располагается спрайт двери акта (рис. 27 слева). Проходя через эту дверь, игрок перемещается в следующую комнату (рис. 5). В этой комнате находится маг, повёрнутый спиной (рис. 26 справа). При приближении игрока маг убегает в другую комнату. Сверху находится дверь (рис. 27 справа), ведущая в третью локацию (рис. 6), где реализована мини-игра "Уверниись от букв".

Сначала всё спокойно, но затем начинают лететь буквы алфавита из случайных позиций. В конце комнаты стоит этот же маг, но повёрнутый лицом на игрока (рис. 26 слева). Также там находится дверь, ведущая в комнату арифметической битвы (рис. 7).

В этой комнате игрок встречает Мага математики (рис. 23), который не позволит ему пройти дальше, пока игрок не ответит на арифметический вопрос. Ответ на вопрос всегда лежит в диапазоне от 0 до 9. Ответ вводится через клавиши цифр на клавиатуре. При неправильном ответе появляется экран поражения, на котором выводится время жизни игрока (рис 29 слева). При правильном ответе игрок появляется в комнате (рис. 8), ведущей к боссу акта.

В этой комнате также располагаются 6 табличек, при соприкосновении с которыми игрок будет видеть текст, рассказывающий ему о боссе. Если игрок коснется двери в конце данной комнаты, то он попадёт в комнату босса 1 акта (рис. 9).

Боссом акта является Маг физики (рис. 25). Битва с боссом состоит из трех фаз: шипы, огонь и снова шипы:

1 фаза — это шипы, которые вылезают из-под земли. Перед этим в месте их появления создаётся предупредительный спрайт. На нём нарисован восклицательный знак, а сам он окрашен в красный цвет, чтобы предупредить об атаке. После пары секунд на месте предупредительного знака появляются шипы;

2 фаза — это атака огнем. После шипов маг начинает атаковать игрока огнём вправо. Спрайты огня появляются справа и летят налево в игрока. Они появляются в случайных местах. После этой атаки начинается 3 фаза;

3 фаза является повторением 1 фазы.

После прохождения всех фаз игрок переходит к двери выхода, ведущей во второй акт. По прохождении через дверь игрок видит победное меню (рис. 29 справа).

АКТ II

Игрок появляется в первой локации второго акта (рис. 10). Отсюда он может попасть в лабиринт и в комнату с игрой в ассоциации. На карте есть речка, которую игрок может пересечь, найдя монетку и принеся её в приемник.

Сначала игрок должен попасть в комнату с игрой в ассоциации (рис. 11), чтобы получить ключ. В комнате игрок выбирает изображение, соответствующее данному органу.

После успешного выбора игрок получает ключ. Затем игрок исследует лабиринт (рис. 12), где находит сундук.

По прохождении лабиринта игроку предстоит перейти появившийся мост через реку (рис. 13), где он может встретится с Магом математики на арифметической битве второго акта (рис. 14).

Арифметическая битва здесь аналогична битве с первого акта. Затем игрок сразу переходит к боссу акта (рис. 15) - Магу природы (рис. 24). Битва с боссом состоит из трёх фаз.

1 фаза состоит из спрайтов лиан, которые вылетают справа налево по 2 штуки за раз. Они расположены горизонтально и вылетают из случайных позиций.

2 фаза проходит похоже на 1, но здесь лианы летят сверху. Они больше горизонтальных лиан и медленнее.

3 фаза представляет собой атаку горизонтальных лиан, которые летят с двух разных сторон.

После победы игрок переходит к двери, ведущей в третий акт. По прохождении через дверь игрок видит сообщение о прохождении акта.

АКТ III

В третьем акте игрок появляется в первой локации (рис. 16). Эта локация огромна. Перейдя через мост на другое облако, игрок обнаруживает рыцаря (рис. 21), защищающего мага математики. Чтобы обойти его, игроку нужно пойти направо и найти радугу, по

которой можно спуститься на землю. Здесь, у реки, находится странник (рис. 22), который предложит помощь игроку в обмен на 5 яблок, расположенных под 3 яблонями.

Собрав все яблоки, игрок получает возможность использовать волшебную воду из реки, становясь невидимым. Таким образом, он может обойти рыцаря и попасть к магу математики для арифметической битвы (рис. 23).

После этого игрок сталкивается с финальным боссом акта, Магом Математики. Битва с финальным боссом состоит из 10 фаз и 3 типов атак.

1 тип характеризуется необычной атакой, когда поле боя окрашивается в красный, а появляется восклицательный знак, предупреждающий об атаке. В случайном месте карты появляется облако, которое может спасти игрока.

Во 2 типе игроку задается арифметический вопрос, на который нужно ответить, выбрав из 4 вариантов. Если ответ не дан в течение 5 секунд, игра заканчивается.

3 тип представляет атаку ветром. Он выбрасывает в игрока прямоугольники сжатого воздуха, которые становятся невидимыми при слиянии с фоном.

В самой битве фазы чередуются данными атаками:

1 фаза - 1 тип атаки;

2 фаза - 2 тип атаки;

3 фаза - снова 1 тип атаки;

4 фаза - 3 тип атаки;

5 фаза - снова 1 тип атаки;

6 фаза - аналогично 2 фазе атаки;

7 фаза - такая же, как и 4 фаза;

8 фаза - снова 1 тип атаки;

9 фаза - 2 тип атаки;

10 фаза - 3 тип атаки;

После успешного завершения всех фаз, игрок переходит к двери, ведущей к победе в акте (рис. 28 справа). По прохождении через неё игрок увидит сообщение о завершении акта и начнутся титры.

Завершая последний акт, игрок осуществляет свое путешествие через мир игры и достигает финальной части своего приключения.

Также по ходу игры программа создает автосохранения, реализованные через базу данных db.sql располагающийся в папке data. При помощи них игрок всегда может вернутся к месту ближайшего сохранения, а не проходить все с нуля.

В каждой локации игры есть музыкальное сопровождение. Если игрок захочет убавить или выключить музыку, то он сможет сделать это через меню, управляя стрелками клавиатуры.

Также в этом меню игрок может просмотреть клавиши управления игрой и их назначения (рис. 2) и выйти в загрузочное меню для выбора сохранения.

8.2 Создание инсталлятора

Мы разработали инсталлятор для игры, который обеспечивает простоту и удобство установки игры на компьютер пользователя. Инсталлятор автоматически устанавливает все необходимые файлы и компоненты, позволяя пользователям начать игру всего лишь в несколько кликов.

8.3 Создание сайта

Для удобства пользователей мы разработали сайт (рис. 30), где они могут просматривать изображения из игры и быстро загружать инсталлятор игры. На сайте представлены качественные изображения игрового процесса, персонажей и локаций, а также кнопка для загрузки инсталлятора. Это обеспечивает пользователям быстрый доступ к установочному файлу игры, что делает процесс начала игры еще более удобным.

Эти шаги были предприняты с целью сделать нашу игру более доступной и удобной для пользователей, обеспечивая им простой доступ к информации о проекте и возможность быстрой установки игры на свои устройства.

Заключение

В заключение, этот проект представляет собой великолепное сочетание креативности, технического мастерства и страсти к созданию захватывающего игрового опыта. Он был инициирован с целью создания уникальной игры, которая была бы увлекательной и привлекательной для игроков всех возрастов.

Задачи, поставленные перед командой, были выполнены с изысканностью и профессионализмом. От разработки основной структуры проекта до проработки каждой детали дизайна и игровой механики - каждый этап разработки проходил с тщательным вниманием к деталям и высоким качеством исполнения.

Одной из ключевых целей проекта было создание неповторимой атмосферы и визуальной привлекательности игрового мира. Благодаря умелому использованию дизайна карт, персонажей и анимаций нам удалось достичь этой цели, захватывая игрока в увлекательное путешествие сквозь различные локации и акты игры.

Технические аспекты игры, такие как реализация игровых механик, управление персонажем, босс-битвы и мини-игры, были тщательно проработаны и реализованы с использованием передовых методов программирования и инструментов разработки.

Система автосохранения, интегрированная в игру, обеспечивает удобство и комфорт для игрока, позволяя ему в любой момент вернуться к ближайшему сохранению и продолжить игровой процесс без необходимости начинать сначала.

Целью нашего проекта было создание увлекательного и захватывающего игрового опыта, который бы приносил радость и удовлетворение игрокам. Мы уверены, что наша работа достигла этой цели, и надеемся, что игра принесет много радости и удовольствия всем, кто попробует её сыграть.

Список литературы

https://www.pixilart.com - для создания дизайна в пиксельной стилистике.

https://github.com - для совместной работы над проектом и возможности делится им.

https://github.com/mario12508/Entangled-Tale-pygame-project - сам проект на github для возможности просмотра кода и файлов игры.

https://lyceum.yandex.ru - Яндекс Лицей для обучения python.

https://124476.github.io/Entangled-Tale/ - Ссылка сайта игры для установки