Министерство науки и высшего образования Российской Федерации

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Санкт-Петербургский государственный университет промышленных технологий и дизайна»

|  |
| --- |
| **Институт информационных технологий и автоматизации** |

|  |  |
| --- | --- |
| Кафедра: | Цифровых и аддитивных технологий |
| Направление подготовки: | 09.03.03 Прикладная информатика |
| Профиль подготовки: | Цифровые технологии в финансах |

|  |
| --- |
| **ОТЧЕТ** |
| о прохождении учебной практики |
| тип практики: технологическая (проектно-технологическая) практика |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Руководитель  от профильной организации / структурного подразделения СПбГУПТД\*: |  |  | | | | |  |  |
| *(наименование организации)* |  | *(должность, Ф.И.О., печать организации)* | | | | |  | *(подпись, печать)* |
| Руководитель  от СПбГУПТД: |  | Доцент, к.т.н., Туркина Н.Р. | | | | |  |  |
|  |  | *(должность, ученая степень / звание, Ф.И.О.)* | | | | |  | *(подпись)* |
| Обучающийся: |  | Коровашев Дмитрий Константинович | | | | |  |  |
|  |  | *(Ф.И.О.)* | | | | |  | *(подпись)* |
| Курс |  | 2 |  | Учебная группа: |  | 2-МД-21 | |

Санкт-Петербург

2024

Министерство науки и высшего образования Российской Федерации

федеральное государственное бюджетное образовательное учреждение высшего образования

«Санкт-Петербургский государственный университет промышленных технологий и дизайна»

**Совместный рабочий график (план) проведения практики и индивидуальное задание**

Вид практики учебная практика

Тип практики технологическая (проектно-технологическая) практика

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Обучающийся | | | | | Коровашев Дмитрий Константинович | | | | | | | |
|  | | | | | *(Ф.И.О.)* | | | | | | | |
| Институт | | | информационных технологий и автоматизации | | | | | | | | | |
|  | | | | | *(наименование института)* | | | | | | | |
| Курс |  | 2 | | Учебная группа | | |  | 2-МД-21 | | Форма обучения |  | Очная |
|  |  |  | |  | | |  |  | |  |  |  |
| Направление подготовки (специальность) | | | | | | | | | 09.03.03 Прикладная информатика | | | |
|  | | | | | | | | | Прикладная информатика в дизайне | | | |
|  | | | | | | | | |  | | | |
| Сроки прохождения практики | | | | | | с 05 февраля 2024г. по 02 июня 2024г. | | | | | | |
|  | | | | | | *(по календарному учебному графику)* | | | | | | |
| Место прохождения практики | | | | | | ООО «АМЛ» | | | | | | |
|  | | | | | | *(полное наименование организации)* | | | | | | |

**Совместный рабочий график (план) проведения практики**

|  |  |
| --- | --- |
| Дата | Содержание выполняемых работ и заданий |
| Общие (типовые вопросы, изучаемые в ходе практики) | |
| 05-25.02.2024 | **Раздел 1. Исследование предметной области.** |
| Основных подходы к разработке компьютерных игр. |
| Основные принципы моделирование графических объектов. |
|  | Требования к функционированию системы "человек-компьютер". Основные подходы к формированию интерфейса пользователя. |
| 26.02-11.03.2024 | **Раздел 2. Компилируемый язык программирования общего назначения.** |
| История возникновения языка программирования. |
| Основы языка программирования. |
|  | Пользовательские функции. |
| **Выполнение индивидуального задания.** | |
| 12.03-23.05.2024 | Написание сюжетной линии на основании исследования предметной области. |
| Моделирование графических объектов (не только элементы WinForms), либо по средствам консольной графики. |
| Формирование таблицы рекордов, хранящихся в файле. |

|  |  |
| --- | --- |
|  | Разработка алгоритмов программных кодов. |
| Написание программных кодов на языке программирования. |
| Реализация возможности «выигрывать или проигрывать». |
| Включение в игру компьютерных игроков, либо мультиплеера. |
| 24.05-02.06.2024 | Формирование пояснительной записки (отчет). |
| Защита практики |

**Требования по выполнению и оформлению индивидуального задания** Пояснительная записка оформляется в соответствии с требованиямиГОСТ 7.32-2017 «Отчет о научно-исследовательской работе. Структура и правила оформления» и ГОСТ Р 7.0.100-2018 «Библиографическая запись. Библиографическое описание. Общие требования и правила составления».

**Вид (ы) отчетных материалов по практике и требования к их оформлению**

**в соответствии с индивидуальным заданием**

1. *Пояснительная записка*
2. *Презентация по материалам практики для защиты*

Руководитель практики

от СПбГУПТД \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ Туркина Н.Р.

 *(подпись, ф.и.о.)*

Принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ Коровашев Д.К.

*(подпись, ф.и.о. обучающегося)*

Дата получения обучающимся индивидуального задания 05.02.2024 г.

# РЕФЕРАТ

Пояснительная записка 72 с., 66 рис., 22 источников

ЯЗЫК ПРОГРАММИРОВАНИЯ PYTHON, СОЗДАНИЕ АРКАДНЫХ ИГР НА PYTHON, РАЗРАБОТКА ВИДЕОИГР, ИСПОЛЬЗОВАНИЕ PYTHON В ИГРОВОЙ РАЗРАБОТКЕ

Объектом исследования является разработка компьютерных игр.

Предмет исследования: создание компьютерных игр на языке программирования Python.

Цель исследования: исследовать теоретические аспекты программирования и разработать видеоигру на языке Python.

В ходе работы был проведен анализ литературы по программированию на языке Python, исследованы его возможности и преимущества для разработки компьютерных игр. На основе этого исследования была разработана полноценная аркадная игра, демонстрирующая потенциал Python в сфере создания видеоигр.

# СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 6](#_Toc168111615)

[1 ТЕОРЕТИЧЕСКИЕ АСПЕКТЫ СОЗДАНИЯ КОМПЬЮТЕРНОЙ ИГРЫ 9](#_Toc168111616)

[1.1 Особенности исследования предметной области для разработки компьютерных игр 9](#_Toc168111617)

[1.2 Основные подходы к формированию интерфейса пользователя 11](#_Toc168111618)

[1.3 Основные принципы моделирования графических объектов 13](#_Toc168111619)

[2 ВОЗМОЖНОСТИ ЯЗЫКА ПРОГРАММИРОВАНИЯ PYTHON ДЛЯ СОЗДАНИЯ КОМПЬЮТЕРНОЙ ИГРЫ 16](#_Toc168111620)

[2.1 Историческая справка о возникновении языка программирования Python 16](#_Toc168111621)

[2.2 Описание функционала языка, который использовался при разработке компьютерной игры 17](#_Toc168111622)

[2.3 Описание библиотек языка, которые использовались в создании игры 20](#_Toc168111623)

[3 ОПИСАНИЕ ПРОЦЕССА СОЗДАНИЯ КОМПЬЮТЕРНОЙ ИГРЫ 22](#_Toc168111624)

[3.1 Назначение компьютерной игры 22](#_Toc168111625)

[3.2 Сюжетная линия компьютерной игры 24](#_Toc168111626)

[3.3 Графические объекты игры и окружение 27](#_Toc168111627)

[3.4 Описание функционала таблицы рекордов 47](#_Toc168111628)

[3.5 Алгоритмы программных кодов 47](#_Toc168111629)

[3.6 Описание функционала программных кодов и представление их скриншотов 50](#_Toc168111630)

[3.7 Основные принципы оценки результата завершения игры 67](#_Toc168111631)

[3.8 Описание особенностей взаимодействия компьютерных игроков 68](#_Toc168111632)

[ЗАКЛЮЧЕНИЕ 70](#_Toc168111633)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 71](#_Toc168111634)

# ВВЕДЕНИЕ

Компьютерные игры стали важным элементом современной культуры и индустрии развлечений, предоставляя пользователям уникальные возможности для интерактивного взаимодействия с виртуальными мирами. Популярность игр обусловлена возможностью создавать захватывающие и разнообразные сюжеты, которые увлекают игроков на долгое время. Этот процесс требует использования различных языков программирования, которые обеспечивают реализацию функциональности, графики, звуковых эффектов и взаимодействия пользователя с игрой. Важно отметить, что разработка игр требует не только знаний программирования, но и понимания основ геймдизайна и пользовательского опыта.

Языки программирования являются основными инструментами для разработки игр, и каждый из них обладает уникальными характеристиками, подходящими для различных задач. Например, Swift, язык, разработанный компанией Apple, предназначен для создания приложений под iOS и macOS. Он отличается высокой производительностью, простотой в освоении и использованием современных технологий. Swift предоставляет мощные инструменты для разработки игр, включая графические и физические движки, что делает его привлекательным выбором для разработчиков, создающих игры для платформ Apple [1, 5].

C++ является одним из наиболее популярных языков программирования в игровой индустрии. Этот язык обеспечивает высокую производительность и позволяет разработчикам работать с аппаратным обеспечением на низком уровне, что особенно важно для создания сложных и ресурсоемких игр. Он используется в различных игровых движках, предоставляющих инструменты для разработки игр различных жанров. Его синтаксис и возможности управления памятью делают его идеальным выбором для создания высокоэффективных и графически насыщенных игр [2, 5].

C# получил широкое распространение среди разработчиков игр благодаря своей интеграции с игровыми движками, поддерживающими кроссплатформенную разработку. Этот язык позволяет упростить процесс создания игр, ускоряя разработку и уменьшая количество ошибок. Простота синтаксиса C# позволяет разработчикам быстро писать и отлаживать код, а встроенные возможности движков обеспечивают мощные инструменты для работы с графикой, физикой и анимацией. Это делает C# отличным выбором как для начинающих, так и для опытных разработчиков [3, 5].

Java широко используется для создания кроссплатформенных игр, особенно мобильных. Этот язык предоставляет мощные инструменты для работы с графикой, сетевыми взаимодействиями и базами данных. Java позволяет разрабатывать игры, которые могут работать на различных платформах, что делает его универсальным инструментом для разработчиков. Благодаря своей стабильности и широкому сообществу, Java остается популярным выбором для создания игр, особенно для мобильных устройств [4, 5].

Python широко признан за свою простоту и легкость чтения кода, что делает его идеальным для быстрого создания прототипов и написания скриптов для игр. Среди множества библиотек, разработанных для Python, Pygame и Arcade являются одними из самых популярных, существенно упрощая разработку как 2D, так и 3D игр. Благодаря своей интуитивной структуре и доступности, Python привлекает множество начинающих разработчиков и инди-студий. Кроме того, Python активно используется в областях машинного обучения и искусственного интеллекта, что открывает возможности для создания интеллектуальных систем и сложных игровых механик. К тому же, Python поддерживает мультиплатформенную разработку, позволяя создавать игры для различных операционных систем, что дополнительно повышает его привлекательность для разработчиков [6].

Задачи учебной практики включают исследование теоретических основ разработки компьютерных игр, изучение потенциала языка программирования Python, анализ использования игровых библиотек, создание и реализацию полноценного игрового проекта, его тестирование и документирование.

В рамках данной учебной практики была разработана компьютерная игра «Геннадий Александрович: учитель времени», целью которой является демонстрация возможностей языка программирования Python в создании интерактивного игрового процесса.

# 1 Теоретические аспекты создания компьютерной игры

## 1.1 Особенности исследования предметной области для разработки компьютерных игр

При создании компьютерных игр ключевым этапом является глубокое исследование предметной области, которое включает в себя анализ различных аспектов игровой индустрии и требований потенциальной аудитории. Рассмотрим основные особенности и методы исследования, необходимые для создания качественной игры:

1) анализ существующих игровых концепций и жанров. Первоначальным и важным этапом исследования в сфере разработки игр является тщательный анализ существующих игровых концепций и жанров. В этом процессе проводится детальный обзор основных жанров компьютерных игр, оцениваются их специфические характеристики и популярные представители. Этот этап включает в себя исследование уникальных игровых механик, сюжетных линий и визуального стиля каждого жанра. Такой подход обеспечивает более глубокое понимание особенностей каждого жанра и помогает определить основные требования и предпочтения целевой аудитории. В конечном итоге, это позволит создать продукт, который будет отвечать требованиям и ожиданиям игрового сообщества;

2) изучение целевой аудитории. Для того чтобы гарантировать успешность разработки игры, необходимо чётко и ясно определить целевую аудиторию, которая будет основным потребителем этого продукта. В этом контексте проводится основательный и глубокий анализ интересов, потребностей и предпочтений потенциальных игроков. В ходе исследования учитываются различные аспекты, включая возрастную категорию, пол, социально-культурный фон аудитории. Эти данные позволяют разработчикам создать игровой продукт, который наиболее точно и гармонично будет соответствовать ожиданиям и запросам потенциальных пользователей. Чёткое определение целевой аудитории своего продукта значительно увеличивает его потенциал в рыночной нише;

3) технологические аспекты. Исследование технологических аспектов представляет собой неотъемлемую часть разработки игр, и оно включает в себя детальный анализ современных технологий и платформ. Этот процесс включает в себя пристальное изучение различных игровых движков, фреймворков и инструментов, их особенностей и уникальных характеристик, а также оценку их применимости и эффективности для конкретного проекта. Важной составляющей является оптимизация игрового процесса и оценка требований к аппаратному обеспечению, которая включает в себя анализ производительности, скорости работы и общей эффективности игрового процесса;

4) психологические аспекты. Исследование психологических аспектов включает в себя изучение воздействия различных элементов игровой механики на общий игровой процесс и эмоциональное состояние игроков. В рамках этого исследования активно рассматриваются мотивационные аспекты, связанные с участием в игре, а также важные факторы, которые напрямую влияют на уровень увлечённости игрока игровым процессом. Кроме того, внимание уделяется актуальным проблемам игровой зависимости, исследуются возможные способы их решения и предотвращения, что позволяет сделать игровой процесс более безопасным и комфортным для игроков;

5) эстетические аспекты. Эстетический аспект охватывает всё, от дизайна персонажей, которых игроки встречают, до мира, в котором эти персонажи обитают, и архитектуры игрового пространства, которое является их домом. Важность каждой детали в дизайне игры не может быть недооценена, поскольку они все совместно работают над созданием уникальной атмосферы. Также рассматривается использование звукового и визуального контента, не только как средства для создания атмосферы, но и как мощного инструмента, который может значительно повлиять на восприятие игры игроком и его эмоциональный опыт. Различные аспекты звукового и визуального дизайна играют важную роль в том, как игра воспринимается, и какие эмоции она вызывает у игрока.

## 1.2 Основные подходы к формированию интерфейса пользователя

Создание интерфейса пользователя (UI) в компьютерных играх требует тщательного подхода к проектированию, чтобы обеспечить удобство и интуитивность взаимодействия. Важно учитывать несколько ключевых аспектов для создания эффективного и привлекательного интерфейса:

1) консистентность дизайна. Все элементы интерфейса должны быть оформлены в едином стиле, что обеспечивает визуальную целостность и облегчает ориентацию игрока в игре. Консистентный дизайн включает одинаковое оформление кнопок, меню и других элементов интерфейса. Это помогает пользователю легко распознавать и использовать элементы интерфейса, повышая общее восприятие игры. Одинаковая цветовая гамма и стиль графики также способствуют созданию гармоничного интерфейса [7];

2) визуальная иерархия. Важные элементы интерфейса должны быть легко различимы. Ключевые кнопки управления должны быть крупными и заметными, а второстепенные элементы — менее яркими. Визуальная иерархия помогает игрокам быстро находить нужную информацию и облегчает взаимодействие с игрой. Правильное использование размеров, цветов и расположения элементов помогает создать чёткую структуру интерфейса [7];

3) функциональность и обратная связь. Интерфейс должен быть функциональным, обеспечивая быструю и лёгкую навигацию. Обратная связь о действиях пользователя, например, всплывающие подсказки или появление сообщений о состоянии игры, помогает игрокам понимать, что происходит в игре. Это повышает уверенность пользователя и улучшает взаимодействие с интерфейсом. Также важно предусмотреть анимации и звуковые эффекты для подтверждения действий [8];

4) интуитивная навигация. Интерфейс должен быть простым и интуитивно понятным, чтобы игроки могли легко освоиться с управлением. Использование знакомых символов и иконок для кнопок и действий способствует быстрому освоению интерфейса. Простота и понятность UI помогают избежать перегруженности экрана и улучшают пользовательский опыт. Предусмотренные подсказки и инструкции также могут значительно облегчить начальное освоение игры [9];

5) минимализм и эффективность. Использование минималистичного дизайна позволяет избежать перегруженности экрана, что улучшает восприятие информации и снижает когнитивную нагрузку на игрока. Интерфейс должен содержать только необходимую информацию, чтобы игрок мог сосредоточиться на игровом процессе. Отказ от избыточных декоративных элементов способствует созданию чистого и ясного интерфейса [8];

6) адаптация под платформу. Важно учитывать различия в управлении на различных платформах. На компьютерах игроки обычно используют клавиатуру и мышь, тогда как на консолях управление осуществляется с помощью геймпадов, а на мобильных устройствах — через сенсорные экраны. Это требует разработки интерфейсов, адаптированных к особенностям каждой платформы, чтобы обеспечить комфортное управление и взаимодействие для всех пользователей [7];

Эти подходы помогут создать удобный и эффективный интерфейс, который улучшит общий пользовательский опыт и сделает игру более привлекательной.

## 1.3 Основные принципы моделирования графических объектов

Моделирование графических объектов включает в себя создание как 2D, так и 3D графики. При разработке игр важно учитывать специфику каждого подхода и использовать соответствующие инструменты. В этом разделе мы рассмотрим основные принципы как 2D, так и 3D моделирования.

Принципы 2D моделирования [10, 12]:

- пиксельная точность и детализация. В 2D моделировании каждый пиксель имеет значение, особенно при использовании пиксельной графики. Для создания чётких и детализированных изображений важно внимательно прорабатывать каждый элемент, избегая размытия и потерь качества. Примером программы для 2D моделирования является Aseprite, которая была использована в разработке учебного проекта. Aseprite позволяет создавать и анимировать спрайты с высокой точностью и детализацией;

- слоистая структура. Использование слоёв позволяет разрабатывать графику более эффективно, отделяя различные элементы (например, фон, персонажей, объекты) друг от друга. Это упрощает процесс редактирования и анимации. Программы, такие как Adobe Photoshop и GIMP, широко используются для создания 2D графики с применением слоёв;

- анимация кадров. В 2D моделировании анимация часто выполняется покадрово. Это означает, что каждый кадр анимации рисуется вручную, что позволяет достичь плавных и естественных движений. В Aseprite этот процесс поддерживается удобными инструментами для покадровой анимации;

- интерфейс и взаимодействие. Одним из важнейших аспектов 2D моделирования является создание интуитивно понятных и удобных интерфейсов для пользователей. Это требует глубокого понимания принципов эргономики и дизайна. Элементы интерфейса должны быть логично организованы и легко воспринимаемы, что способствует улучшению общего опыта взаимодействия пользователя с программой.

Принципы 3D моделирования [11, 12]:

- топология и полигональная сетка. В 3D моделировании правильная топология объекта играет ключевую роль. Она обеспечивает равномерное распределение полигонов по поверхности модели, что важно для последующего текстурирования и анимации. Сетка должна быть плавной и непрерывной, без пересечений и лишних граней;

- инструменты моделирования. Для создания и редактирования графических объектов используются различные инструменты, такие как масштабирование, перемещение, вращение и нарезка. Эти инструменты помогают формировать основную структуру объекта и вносить изменения в его детали. Программы, такие как Blender и Autodesk Maya, широко применяются для 3D моделирования;

- текстурирование и UV-развертка. Текстурирование — это процесс наложения изображений на поверхность 3D моделей для придания им реалистичного вида. Перед текстурированием модель должна быть подготовлена с помощью UV-развертки, которая позволяет развернуть 3D объект на плоскость для удобного наложения текстур;

- освещение и затенение. Правильное освещение играет важную роль в восприятии графических объектов. Различные типы источников света и настройки освещения помогают создать нужную атмосферу и акцентировать важные элементы сцены. Важным аспектом также является плавное затенение, которое позволяет сгладить переходы между полигонами и сделать поверхность объекта более реалистичной;

- материалы и текстуры. Использование различных материалов и текстур позволяет добиться высокой степени реализма. Важно правильно настроить параметры материалов, такие как отражение, преломление и шероховатость, чтобы свет взаимодействовал с объектом так, как это происходит в реальном мире. В программах для 3D моделирования, таких как Blender или Substance Painter, можно применять материалы PBR (Physically Based Rendering), которые учитывают физические свойства материалов и создают более реалистичные изображения.

Эти принципы помогают создать качественные и реалистичные графические объекты, которые значительно улучшают визуальное восприятие игры и делают её более привлекательной для игроков.

# 2 Возможности языка программирования Python для создания компьютерной игры

## 2.1 Историческая справка о возникновении языка программирования Python

Идея создания Python возникла у Гвидо ван Россума в конце 1980-х годов. Ван Россум стремился создать язык, который сочетал бы простоту и читаемость с мощными возможностями. Основная цель заключалась в создании языка, который был бы лёгок в изучении и использовании, но при этом подходил для решения широкого спектра задач. Вдохновением для Python послужили такие языки, как ABC, который ван Россум помогал разрабатывать, и модульная структура Unix [13].

Первая версия Python была выпущена в 1991 году и включала базовые возможности работы с данными и управления потоком выполнения. Важными аспектами Python стали динамическая типизация и автоматическое управление памятью, а также его синтаксис, который почти полностью лишён лишних символов, таких как фигурные скобки и точки с запятой, что делает код понятным даже для тех, кто только начинает изучать программирование. Благодаря этому Python стал популярным в образовательных учреждениях и среди учёных, которым нужен был инструмент для быстрого прототипирования и анализа данных [14].

Одним из ключевых факторов, способствовавших популярности Python, стала обширная стандартная библиотека, которая включает модули для выполнения различных задач, от работы с файлами и сетевыми протоколами до веб-разработки и научных вычислений. Со временем сообщество разработчиков Python значительно выросло, создавая множество сторонних библиотек и инструментов, расширяющих возможности языка [14].

Сегодня Python используется в самых различных областях, включая разработку веб-приложений, анализ данных, искусственный интеллект, машинное обучение и автоматизацию процессов. В 2019 году Python занял первое место в рейтинге языков программирования, составленном сайтом TIOBE, что подтверждает его значимость и востребованность в современной индустрии [14].

## 2.2 Описание функционала языка, который использовался при разработке компьютерной игры

Python является мощным и гибким инструментом для разработки игр благодаря множеству библиотек и фреймворков, которые расширяют его функционал и облегчают процесс создания как простых, так и сложных игровых проектов. Основные возможности Python для создания игр включают следующие области применения:

- библиотеки и фреймворки для 2D игр:

1) Pygame является одной из самых популярных библиотек для разработки 2D игр. Она предоставляет широкий спектр инструментов для работы с графикой, звуком и событиями ввода, что позволяет создавать интерактивные игры с богатым функционалом. Pygame также поддерживает анимацию и позволяет легко управлять игровыми циклами и состояниями [15];

2) Arcade — это современная библиотека для создания 2D игр, которая отличается простотой использования и хорошей документацией. Она поддерживает работу с текстурами, анимацией, физикой и взаимодействием пользователя, что делает её отличным выбором как для начинающих, так и для опытных разработчиков. Arcade предоставляет функции для управления камерой, что позволяет следить за движением игрока и создавать плавную навигацию по игровому миру. Она также поддерживает использование спрайтов и слоёв, что облегчает создание сложных сцен и уровней [16];

- библиотеки и фреймворки для 3D игр:

1) Pyglet позволяет создавать как 2D, так и 3D игры с использованием OpenGL. Эта библиотека поддерживает воспроизведение мультимедиа, работу с окнами и событиями ввода, а также анимацию и текстурирование, что делает её мощным инструментом для разработки графически насыщенных игр [17];

2) Panda3D — это полноценный игровой движок, который поддерживает разработку 3D игр. Он предоставляет инструменты для работы с графикой, физикой, звуком и анимацией, а также поддерживает интеграцию с другими библиотеками и инструментами Python [18];

- функциональные возможности для создания игрового процесса:

1) физические движки. Библиотеки, такие как Arcade и Pygame, включают физические движки, которые обеспечивают реалистичное взаимодействие объектов в игре. Это включает в себя гравитацию, столкновения и другие физические эффекты, необходимые для создания реалистичного игрового процесса [15, 16];

2) события ввода. Python обеспечивает простую обработку событий ввода с клавиатуры, мыши и других устройств. Это позволяет разработчикам легко создавать интерактивные элементы управления и реализовывать различные игровые механики;

3) управление камерой. В Arcade реализована поддержка камеры, которая позволяет следить за движением игрока и других объектов в игровом мире, создавая плавный и динамичный игровой процесс [16];

- инструменты для создания и обработки графики:

1) текстуры и спрайты. Библиотеки, такие как Arcade, предоставляют мощные инструменты для загрузки и отображения текстур и спрайтов. Это позволяет создавать высококачественные графические элементы и анимации, которые значительно улучшают визуальное восприятие игры [16];

2) работа с анимацией. Python предоставляет возможности для создания сложных анимаций персонажей и объектов. Это включает в себя поддержку покадровой анимации, а также использование скелетной анимации для более реалистичного движения;

- создание и управление игровыми уровнями:

1) редакторы уровней. Библиотеки и фреймворки Python позволяют интегрировать редакторы уровней, такие как Tiled, что облегчает процесс создания и настройки игровых уровней. Это особенно полезно для разработчиков, которые хотят создавать сложные и многоуровневые игры [19];

2) сохранение и загрузка прогресса. Python предоставляет инструменты для сохранения и загрузки состояния игры, что позволяет игрокам сохранять свой прогресс и продолжать игру с того места, где они остановились;

- аудио и звуковые эффекты. Библиотеки, такие как Pygame и Arcade, поддерживают работу со звуковыми эффектами и музыкальным сопровождением. Это включает в себя воспроизведение звуковых файлов, управление громкостью, а также создание звуковых эффектов для различных событий в игре. Аудиоэффекты играют важную роль в создании атмосферы и улучшении пользовательского опыта [15, 16];

- веб-разработка и многопользовательские игры:

1) разработка веб-игр. Python можно использовать для создания серверной части многопользовательских веб-игр. Фреймворки, такие как Django и Flask, позволяют разработчикам создавать мощные серверные приложения, которые могут взаимодействовать с клиентскими частями игр через веб-интерфейсы. Это открывает возможности для создания сложных многопользовательских систем и интеграции игр с веб-сервисами;

2) WebSocket. Для обеспечения реального времени взаимодействия между игроками можно использовать WebSocket, который поддерживается многими фреймворками Python. Это позволяет создавать динамичные и отзывчивые многопользовательские игры с минимальными задержками [20];

- машинное обучение и искусственный интеллект:

1) интеграция с машинным обучением. Python широко используется в области машинного обучения и искусственного интеллекта благодаря библиотекам, таким как TensorFlow, Keras и Scikit-learn. Эти инструменты могут быть использованы для создания интеллектуальных игровых систем, которые адаптируются к действиям игрока, анализируют его поведение и предлагают персонализированные игровые сценарии [21];

2) разработка AI для игр. Машинное обучение можно применять для создания более сложных и реалистичных игровых противников. AI может анализировать действия игроков, обучаться на их поведении и создавать стратегии, которые делают игру более интересной и непредсказуемой.

Эти функциональные возможности делают Python мощным инструментом для создания игр различных жанров и сложности. Разработчики могут использовать широкий спектр библиотек и инструментов, чтобы создавать качественные игровые проекты.

## 2.3 Описание библиотек языка, которые использовались в создании игры

При создании игры «Геннадий Александрович: учитель времени» основное внимание было уделено использованию библиотеки Arcade. В дополнение к ней была использована библиотека Math. Они значительно расширили функционал игры и упростили процесс разработки.

Arcade является одной из самых мощных и удобных библиотек для создания 2D-игр на Python. Она была специально разработана для упрощения процесса разработки игр и обладает рядом преимуществ. Среди них высокая производительность, богатый набор функций и простота использования [16]. Arcade поддерживает загрузку и отображение текстур, что позволяет легко работать с графикой. Например, в игре использовались функции arcade.load\_texture и arcade.draw\_lrwh\_rectangle\_textured для загрузки и отображения спрайтов и фонов.

Физический движок, встроенный в Arcade, позволяет моделировать гравитацию, столкновения и другие физические эффекты, что делает возможным реализацию реалистичных движений и взаимодействий объектов в игре. Это значительно упрощает создание сложных игровых механик. Кроме того, библиотека упрощает обработку событий ввода с клавиатуры и мыши. Это позволяет легко реализовать управление персонажем и взаимодействие с интерфейсом, что было реализовано через методы on\_key\_press, on\_key\_release и on\_mouse\_press.

Arcade также предоставляет инструменты для работы с камерой, позволяя следить за движением игрока и создавать плавные переходы между различными частями игрового мира. Это особенно важно для игр с большими уровнями. Поддержка тайловых карт в Arcade позволяет легко создавать и настраивать сложные игровые уровни, что было реализовано в проекте с помощью функции arcade.load\_tilemap.

В дополнение к Arcade, библиотека Math использовалась для выполнения нескольких математических операций, необходимых для расчета движений объекта в игре. Math предоставляет широкий набор математических функций, таких как синус, косинус, тангенс, логарифмы, степени и т.д. Эти функции использовались для расчета параметров движения, что обеспечило точное и предсказуемое поведение игровых элементов.

Эти библиотеки обеспечили основные графические и интерактивные функции игры, а также необходимые математические операции. Использование Arcade позволило создать игру с богатым функционалом, высококачественной графикой и реалистичной физикой, в то время как Math дополнила её и помогла в реализации равномерного движения объекта.

# 3 ОПИСАНИЕ ПРОЦЕССА СОЗДАНИЯ КОМПЬЮТЕРНОЙ ИГРЫ

## 3.1 Назначение компьютерной игры

Компьютерная игра «Геннадий Александрович: учитель времени» была создана в рамках учебной практики с целью освоения процессов разработки игр и применения теоретических знаний на практике. Выполнение такого проекта не только служит инструментом для обучения программированию, но и способствует развитию творческих навыков. Главная задача этой игры заключается в предоставлении игрокам увлекательного игрового и образовательного опыта.

Особенностью игры является интеграция образовательных элементов. Во время прохождения каждого уровня игроки будут сталкиваться с вопросами по истории, связанными с конкретной эпохой, в которой они находятся. Это требует от игроков не только умения управлять персонажем и преодолевать препятствия, но и демонстрировать свои знания в области истории. Игра способствует расширению знаний игроков, предлагая им интересный способ обучения через игру.

Игра представляет собой двухмерный платформер с элементами аркады.

Основные характеристики игры:

- игровой процесс:

1) игрок управляет персонажем, который перемещается по уровню, избегая опасностей;

2) персонаж может двигаться влево и вправо, прыгать, ускоряться и взаимодействовать с появляющимися вопросами по истории;

3) игра состоит из двух уровней, которые имеют уникальный дизайн и задачи;

4) игроку нужно дойти до конца карты и собрать осколок маски Майя, а на втором уровне отнести её через джунгли к вождю Майя;

- визуальный стиль и графика:

1) все графические элементы игры, включая персонажей и окружающую среду, были созданы или отредактированы в редакторе Aseprite;

2) уровни состоят из различных слоёв: земли, фона, смертельных текстур и объектов заднего плана, которые создают атмосферу и настроение игры;

- интерфейс и управление:

1) управление персонажем осуществляется с помощью двух вариаций клавиш клавиатуры, что делает игру удобной и доступной для игроков;

2) интерфейс игры интуитивно понятен, что позволяет игроку сосредоточиться на игровом процессе.

Игровые механики и особенности:

- персонаж игрока:

1) главный герой игры — анимированный персонаж, который может перемещаться в двухмерном пространстве;

2) персонаж обладает набором анимаций для различных действий, таких как ходьба, прыжок и бег;

3) управление персонажем осуществляется с помощью стрелок клавиатуры или клавиш W, A, D;

- вопросы по истории:

1) при прохождении определённых точек карты на экране появляется вопрос по истории той эпохи, в которой находится персонаж;

2) при правильном ответе на вопрос действия персонажа разблокируются, и он может идти дальше, а при неправильном - он погибает и ему предлагается возродиться;

- физическая модель:

1) игра использует встроенный физический движок Arcade;

2) при попадании в смертельные текстуры персонаж лишается возможности двигаться и ему приходится ожидать своей смерти на протяжении нескольких секунд.

Игра «Геннадий Александрович: учитель времени» сочетает в себе элементы развлечения и образования, создавая интерактивную среду, в которой игроки могут развивать свои навыки и расширять свои знания. Она предназначена для широкой аудитории, включая как детей, так и взрослых, желающих получить новые знания в увлекательной форме, совмещая развлекательные и образовательные аспекты.

## 3.2 Сюжетная линия компьютерной игры

Компьютерная игра «Геннадий Александрович: учитель времени» представляет собой увлекательное приключение, сочетающее элементы платформера и образовательной игры.

История начинается с экскурсии в музей, где Геннадий Александрович проводит урок истории для своих учеников. Во время демонстрации старинной маски народа Майя он случайно роняет её, и маска разламывается на несколько частей. При попытке поднять её, маска исчезает, а Геннадий Александрович оказывается в другой исторической эпохе. Народ Майя разгневан на него, и теперь, чтобы вернуться домой, ему необходимо пройти через различные периоды истории и собрать маску по частям. Игроку предстоит управлять главным героем, Геннадием Александровичем.

Игровой процесс:

- начальное меню. Игрок видит две кнопки: "Начать" и "Выйти". При нажатии кнопки "Выйти" игра закрывается, а при нажатии "Начать" показывается изображение с историей сюжета;

- история. Игрок читает рассказ о том, как Геннадий Александрович попал в Древний Египет. Далее, на экране появляются инструкции по игровому управлению с обозначением всех клавиш, необходимых для прохождения игры;

- уровни:

1) Древний Египет. Персонаж оказывается в Древнем Египте. Игрок должен провести Геннадия через карту, избегая зыбучих песков и шипов. На пути встречаются исторические вопросы, связанные с Древним Египтом, на которые необходимо правильно ответить для продвижения вперёд. В конце карты находится осколок маски, который необходимо собрать;

2) Джунгли Мексики. После сбора осколка, Геннадий переносится в джунгли Мексики к вождю Майя. Здесь игроку предстоит отвечать на вопросы про цивилизацию Майя, избегая опасностей, таких как лианы и отравленные воды. Персонаж взбирается на пирамиду к вождю, чтобы извиниться и передать найденный осколок маски;

- заключительная сцена. Вождь Майя прощает Геннадия Александровича, но предупреждает, что это не последняя их встреча. Далее следует рассказ о том, как учитель возвращается домой, восхищённый своим приключением. Появляется надпись "Продолжение следует". После этого игрок может только выйти из игры.

Функционал игры:

* управление. Игрок управляет персонажем с помощью клавиш для движения влево и вправо, прыжков и бега. Управление объясняется в начальной инструкции;
* вопросы. На каждом уровне встречаются вопросы по истории той эпохи, в которой находится персонаж. Правильные ответы необходимы для продвижения по уровню;
* сбор артефактов. Целью каждого уровня является сбор фрагментов маски, которые необходимы для завершения игры;
* опасности. Игроку предстоит избегать различных опасностей, таких как зыбучие пески, шипы, лианы и отравленные воды, чтобы не начинать уровень заново.

Игра «Геннадий Александрович: учитель времени» сочетает в себе элементы образовательной и развлекательной игры, предоставляя игрокам возможность погружаться в исторические эпохи и расширять свои знания через интерактивные задания.

Сценарий игры представлен на рисунках 1-2.

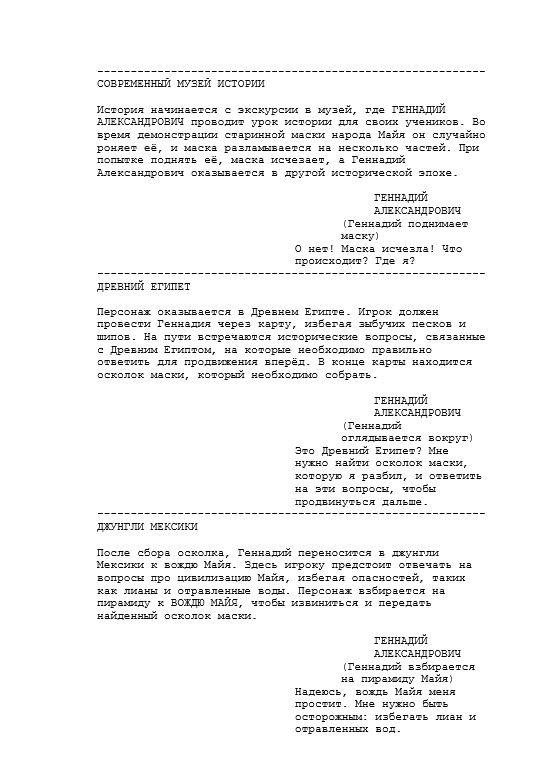


Рисунок 1 - Сценарий игры

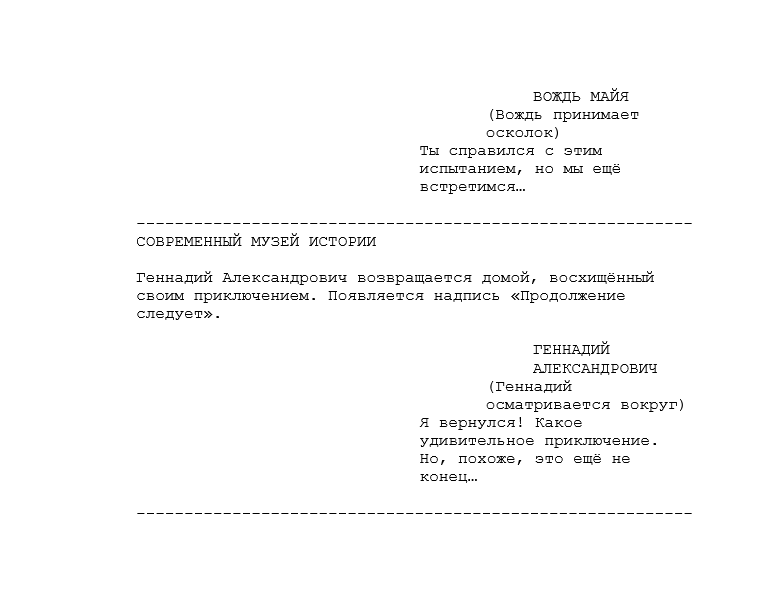


Рисунок 2 - Сценарий игры

## 3.3 Графические объекты игры и окружение

Разработка графических объектов выполнялась в редакторе изображений Aseprite, который предназначен для рисования и анимации в стиле пиксель-арт. Для изображений с вопросами и изображения с клавишами управления и ответа вождя Майя, а также приветственных меню каждого уровня применялась комбинация программ Aseprite и Adobe Photoshop. Начальное меню, изображения с историями и самое последнее изображение выполнялись только с помощью Adobe Photoshop.

Элементы слоя «ground первой карты изображены на рисунке 3.



Рисунок 3 – Элементы слоя ground первой карты

Элементы слоя ground второй карты изображены на рисунке 4.

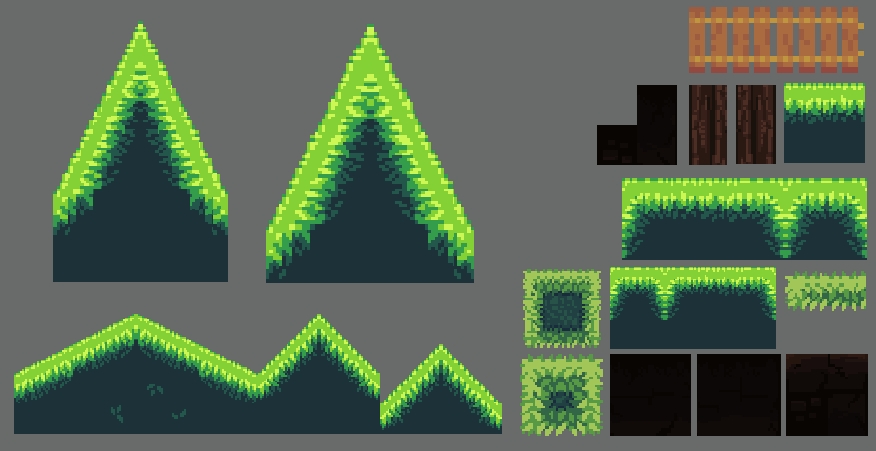


Рисунок 4 – Элементы слоя ground второй карты

Ловушки двух карт, попав в которые нужно будет возрождаться заново, представлены на рисунке 5.

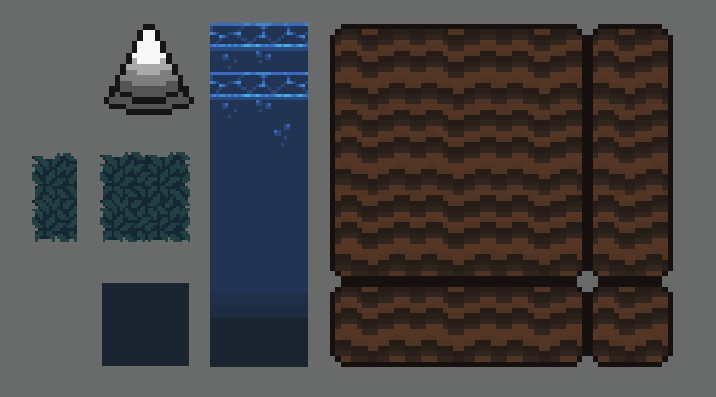


Рисунок 5 – Ловушки двух карт

Элементы заднего фона двух карт изображены на рисунке 6.



Рисунок 6 – Элементы заднего фона двух карт

Элемент слоя mask первой карты представлен на рисунке 7.



Рисунок 7 - Элемент слоя mask первой карты

Элемент слоя maya второй карты представлен на рисунке 8.



Рисунок 8 - Элемент слоя maya второй карты

Маска Майя, которая на протяжении всей игры находится в левой верхнем углу экрана вместе с счётчиком частей масок, представлена на рисунке 9.

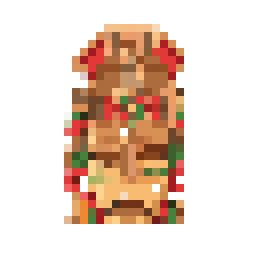


Рисунок 9 - Маска Майя в левом верхнем углу экрана

Персонаж Геннадия Александровича и его анимации (состояние простоя, прыжок, бег, ходьба, соответственно) показаны на рисунке 10.



Рисунок 10 – Персонаж

Задний фон первой карты про Древний Египет, нарисованный также в Aseprite, показан на рисунке 11.

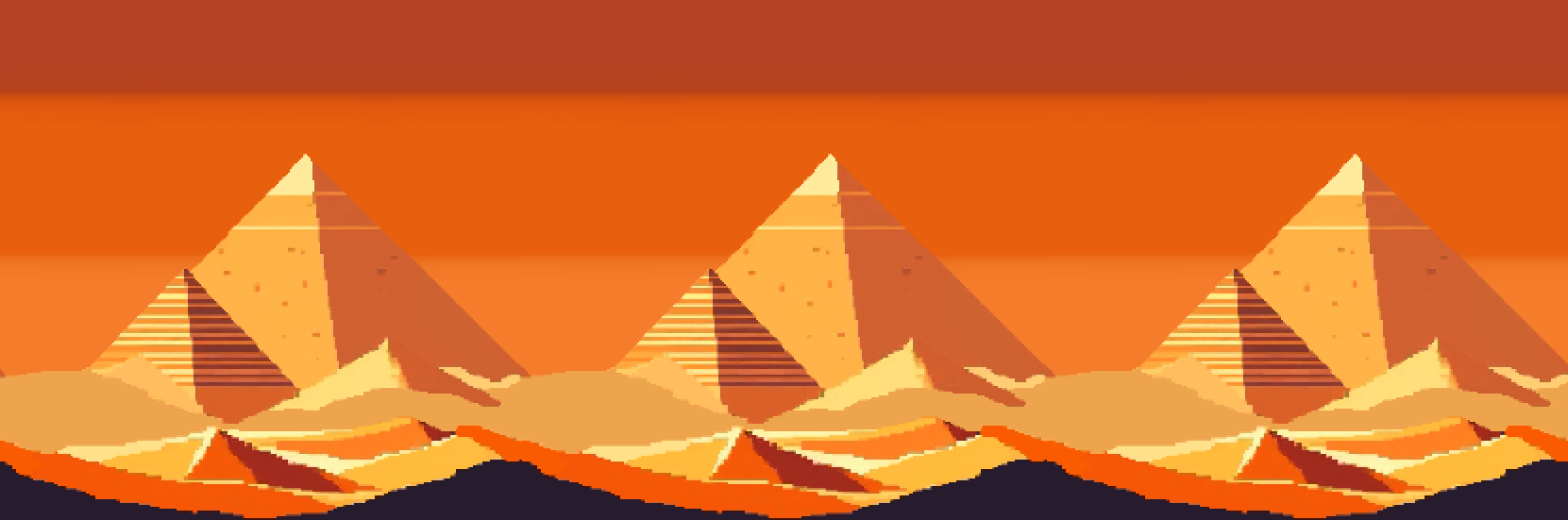


Рисунок 11 - Задний фон первой карты

Задний фон второй карты про джунгли Мексики и вождя Майя, нарисованный также в Aseprite, показан на рисунке 12.



Рисунок 12 - Задний фон второй карты

На рисунке 13 представлено начальное меню игры.



Рисунок 13 - Начальное меню

На рисунках 14 и 15 представлены сюжетные истории игры в начале и конце соответственно.

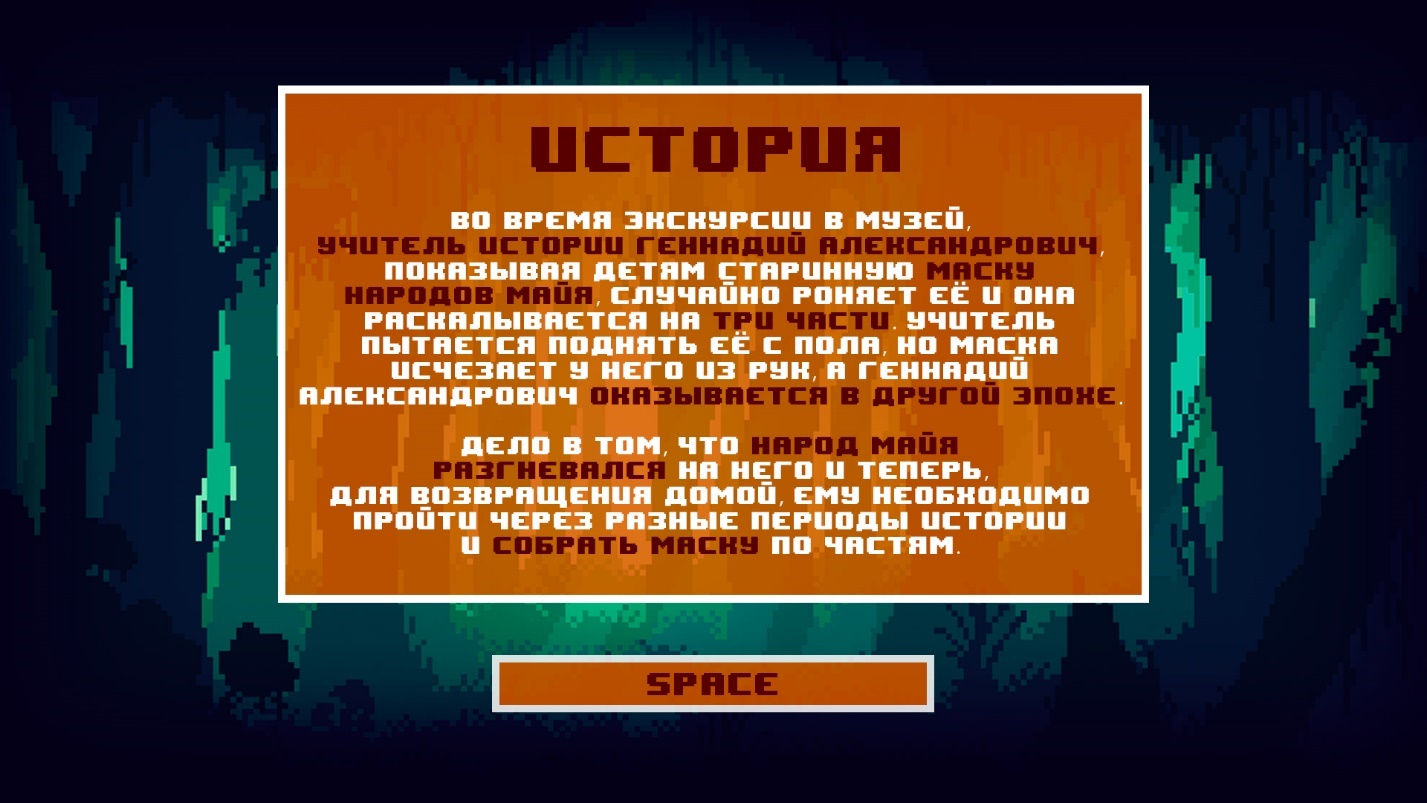


Рисунок 14 - Сюжетная история в начале

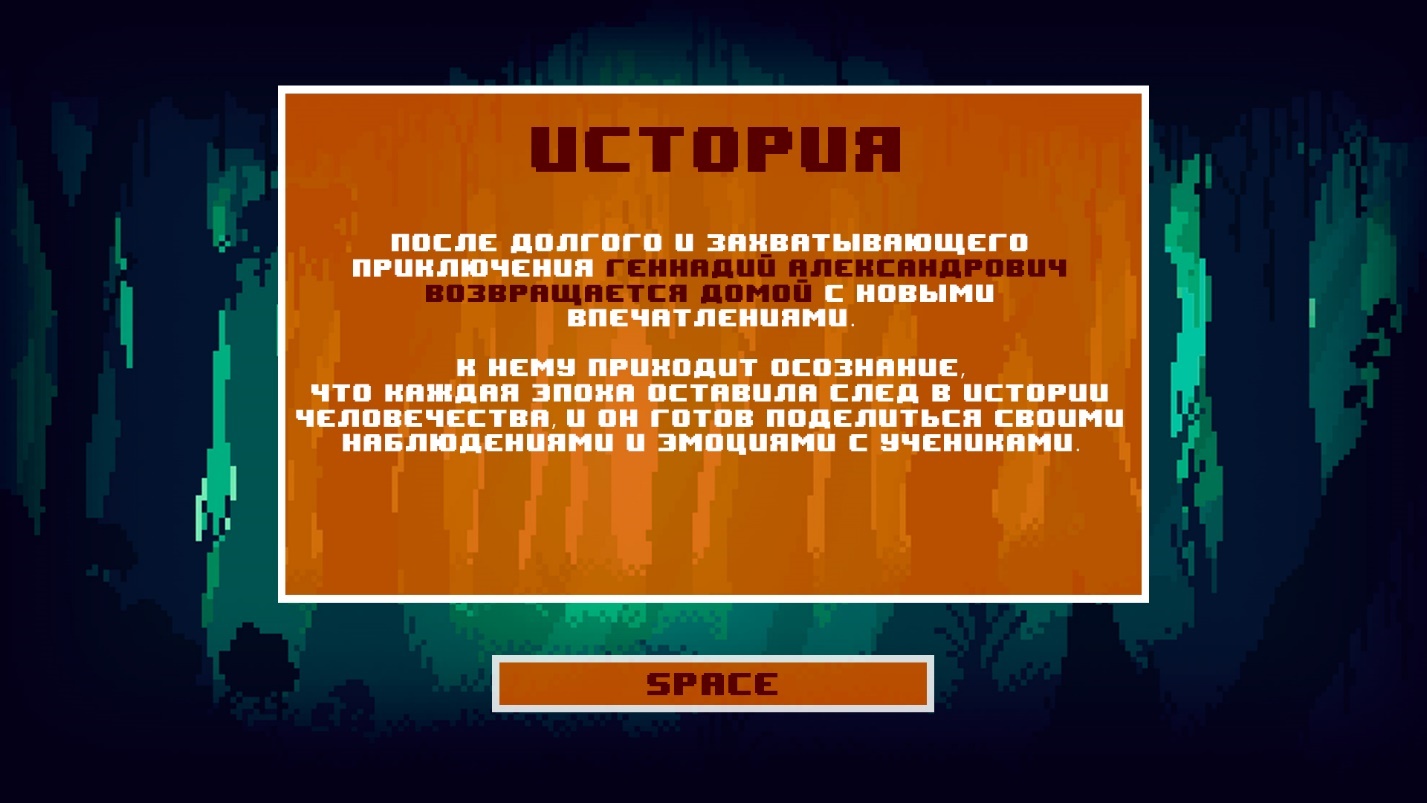


Рисунок 15 - Сюжетная история в конце

Памятка по управлению в игре показана на рисунке 16.

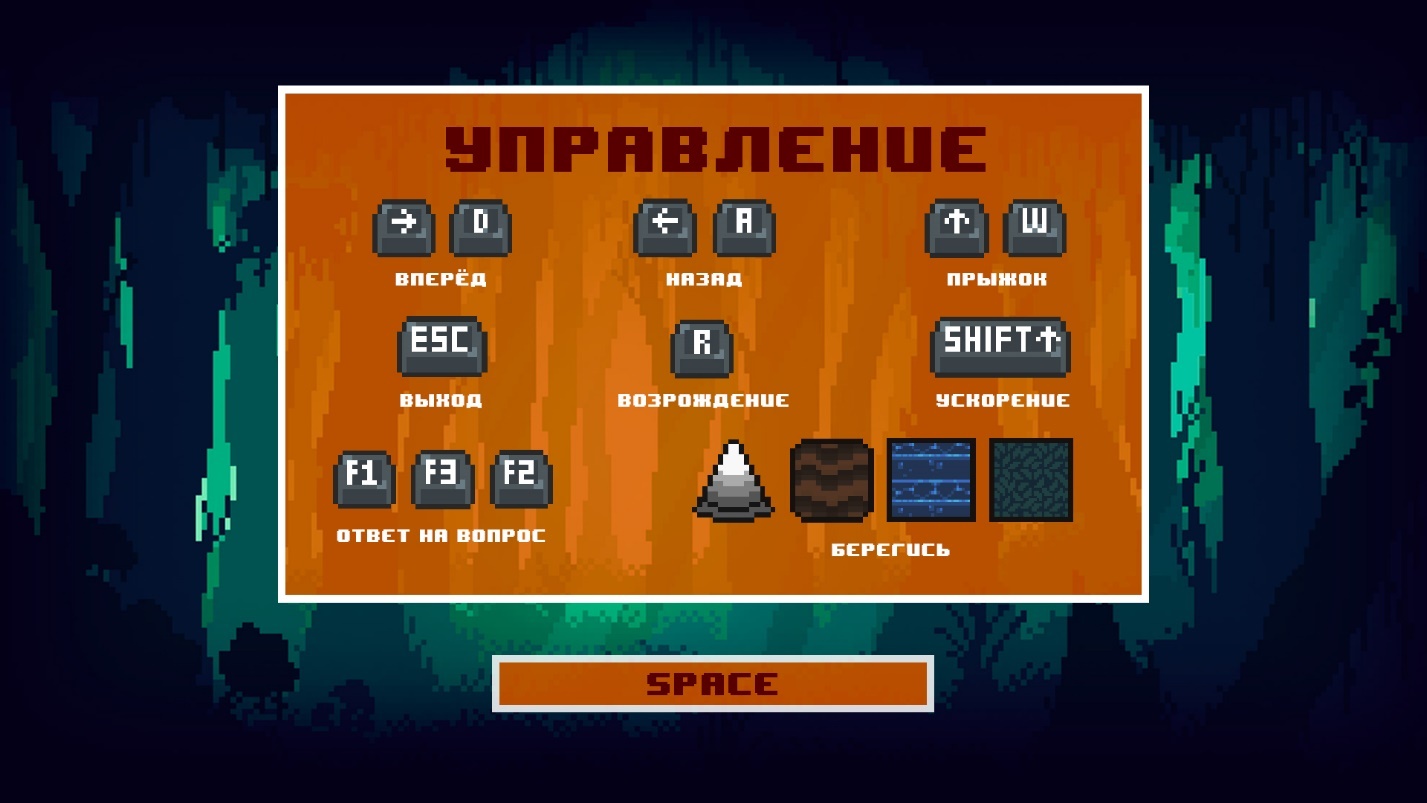


Рисунок 16 - Памятка по управлению

Начальные заставки первой и второй карт представлены на рисунках 17 и 18 соответственно.



Рисунок 17 - Начальная заставка первой карты



Рисунок 18 - Начальная заставка второй карты

Ответ вождя Майя в конце второй карты показан на рисунке 19.



Рисунок 19 - Ответ вождя Майя

Конечное изображение представлено на рисунке 20.



Рисунок 20 - Конечное изображение

Вопросы, на которые должен ответить игрок, представлены на рисунках 21-30.

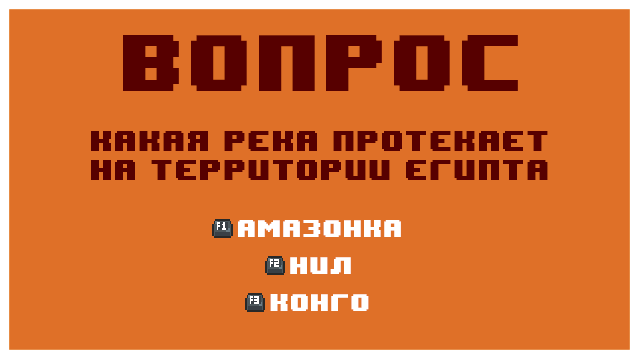


Рисунок 21 - Вопрос №1 первой карты

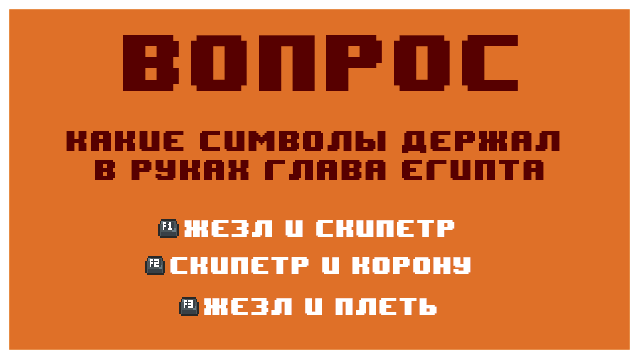


Рисунок 22 - Вопрос №2 первой карты



Рисунок 23 - Вопрос №3 первой карты



Рисунок 24 - Вопрос №4 первой карты



Рисунок 25 - Вопрос №5 первой карты

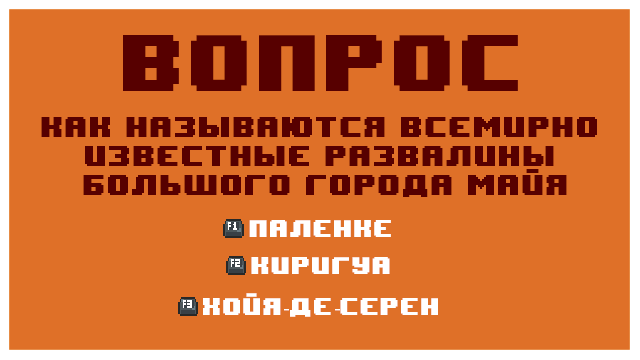


Рисунок 26 - Вопрос №1 второй карты

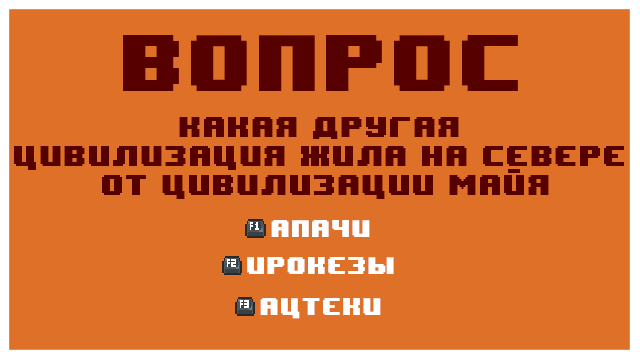


Рисунок 27 - Вопрос №2 второй карты



Рисунок 28 - Вопрос №3 второй карты

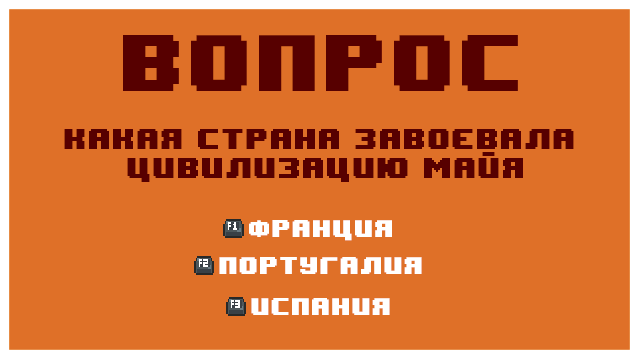


Рисунок 29 - Вопрос №4 второй карты



Рисунок 30 - Вопрос №5 второй карты

Помимо изображений, использовался текст и шрифт «8BIT WONDER(RUS BY LYAJKA)», который подходит по стилю пиксель-арт.

Начало игры с заставкой первой карты представлено на рисунке 31.



Рисунок 31 - Начало игры

После движения вперёд игроку предлагается ответить на вопрос. Если он отвечает неверно, то он возрождается по нажатию кнопки R и проходит всю карту заново. Если ответ верный, то изображение закрывается. На рисунке 32 представлен вопрос №1 на первой карте. На рисунке 33 показан исход неверного ответа на вопрос с предложением начать заново.



Рисунок 32 - Вопрос №1 первой карты



Рисунок 33 - Исход неверного ответа

При падении на шипы или попадании в зыбучие пески персонаж проваливается туда, не может двигаться и поэтому просто ждёт, пока ему предложат начать игру заново. На рисунке 34 показано падение на шипы и исход на рисунке 35. На рисунке 36 показано падение в зыбучие пески и представлен исход на рисунке 37.



Рисунок 34 - Падение на шипы



Рисунок 35 - Исход падения на шипы



Рисунок 36 - Падение в зыбучие пески



Рисунок 37 - Исход падения в зыбучие пески

На рисунке 38 представлено прохождение карты.



Рисунок 38 - Прохождение карты

На рисунке 39 представлен конец первой карты и можно увидеть часть маски. После того, как персонаж дотрагивается до неё, его переносит на вторую карту. Счётчик частей маски обновляется.



Рисунок 39 - Конец первой карты

На рисунке 40 представлена начальная заставка второй карты.



Рисунок 40 - Начальная заставка второй карты

На рисунке 41 представлен вопрос №2 на второй карте.

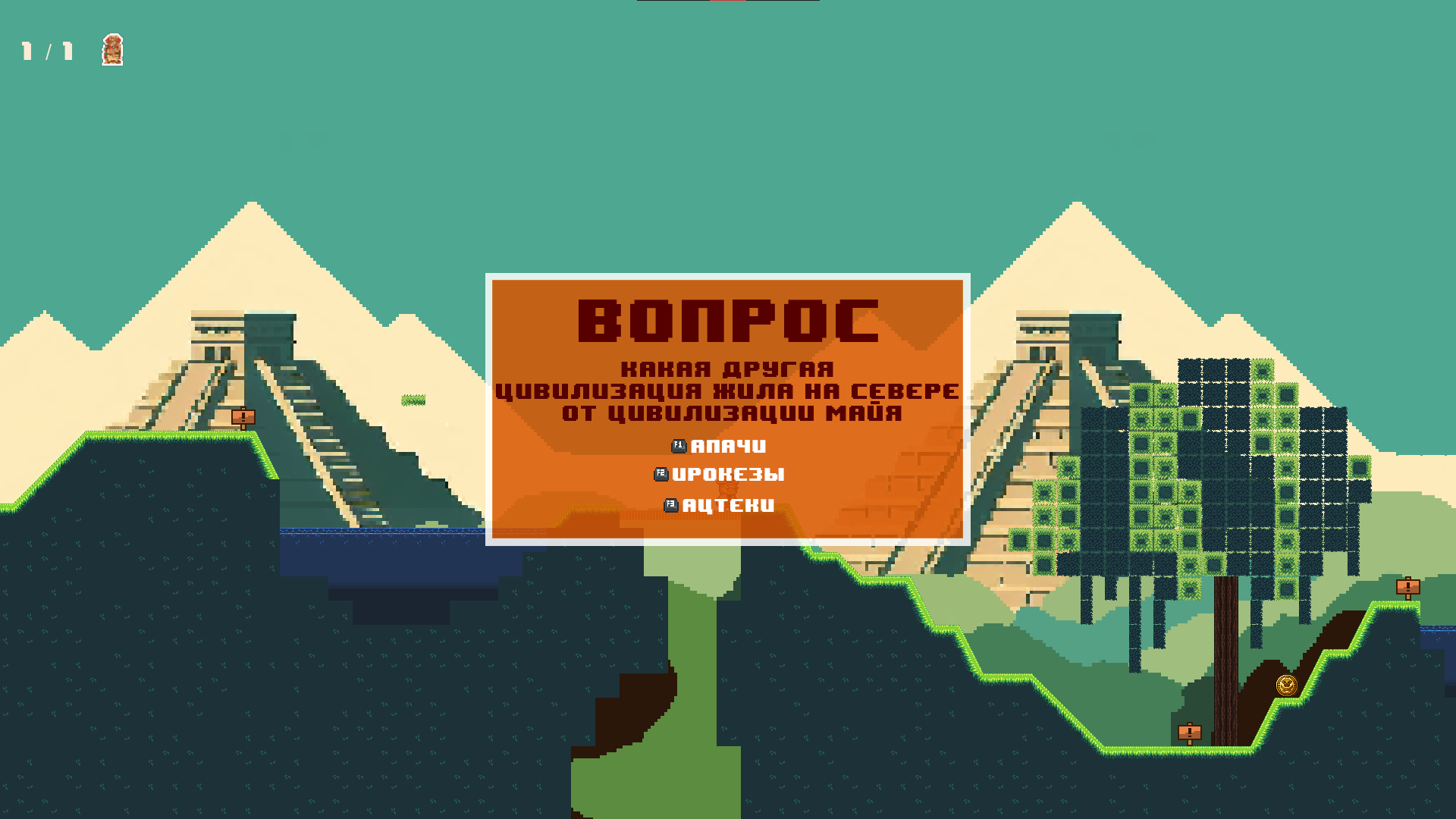


Рисунок 41 - Вопрос №2 второй карты

При падении в отравленную воду или попадании в лианы персонаж застревает там и не может двигаться и поэтому просто ждёт, пока ему предложат начать заново. На рисунке 42 показано падение в отравленную воду и исход на рисунке 43. На рисунке 44 показано попадание в лианы и представлен исход на рисунке 45.

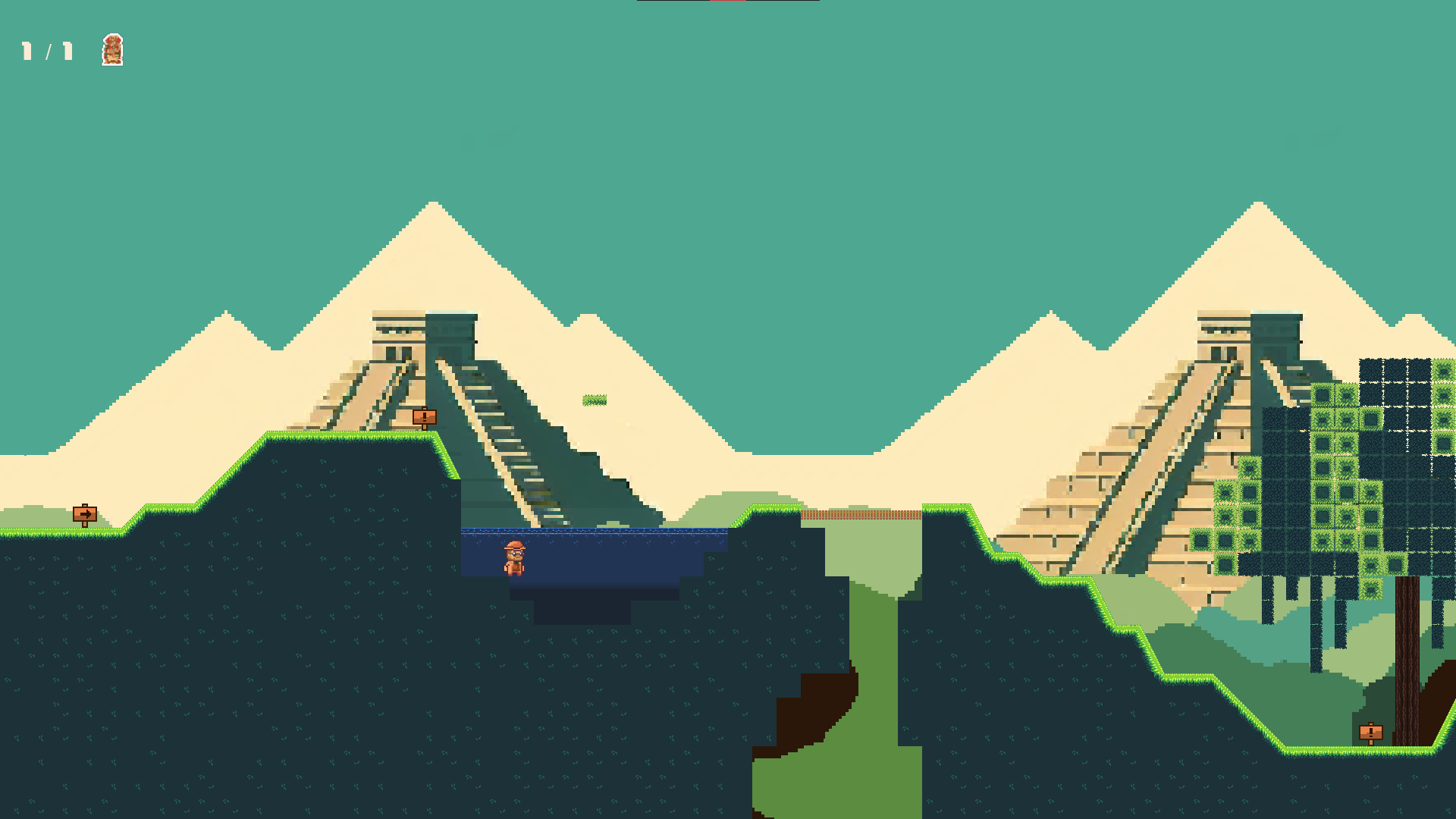


Рисунок 42 - Падение в отравленную воду

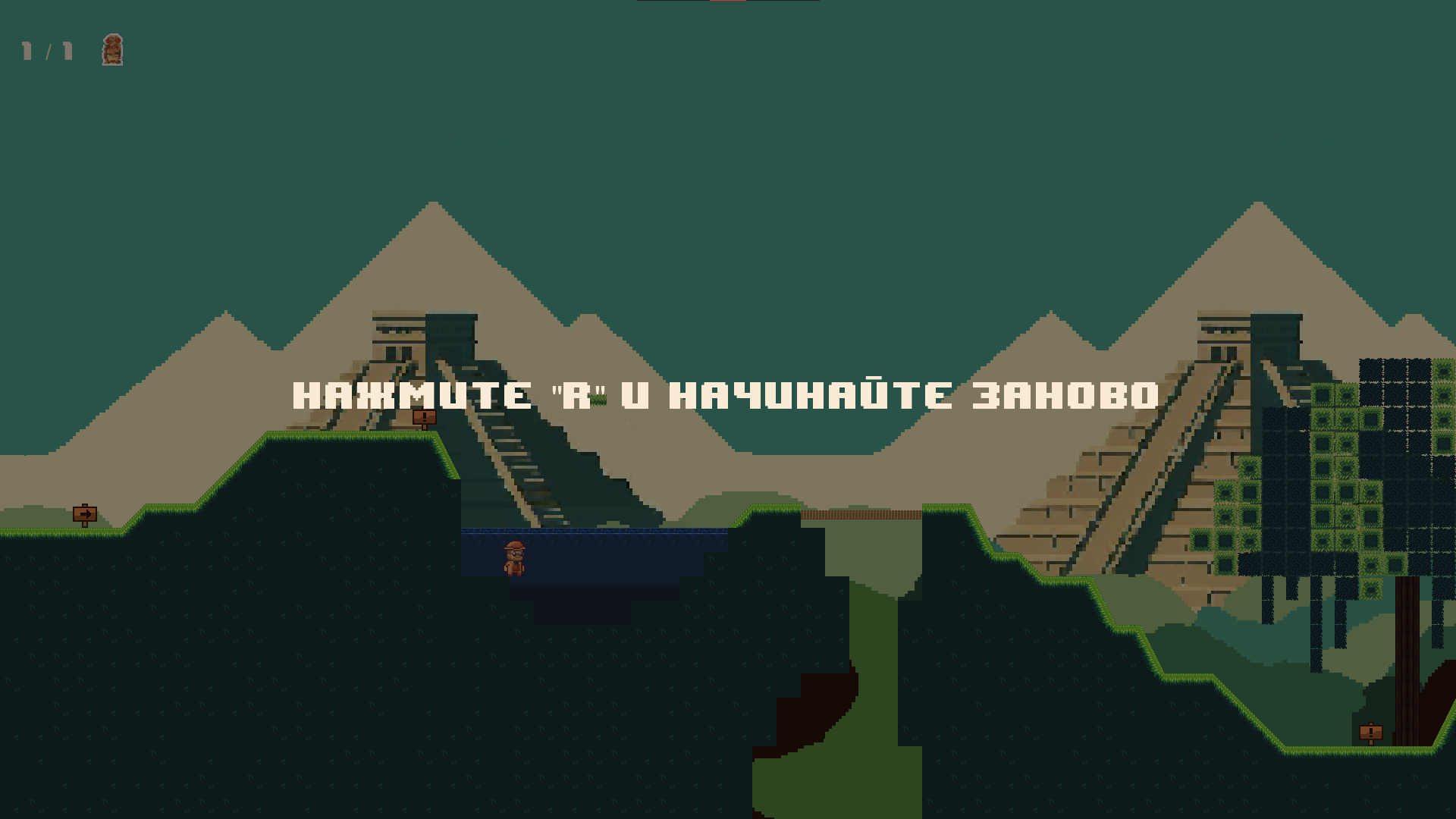


Рисунок 43 - Исход от падения в отравленную воду

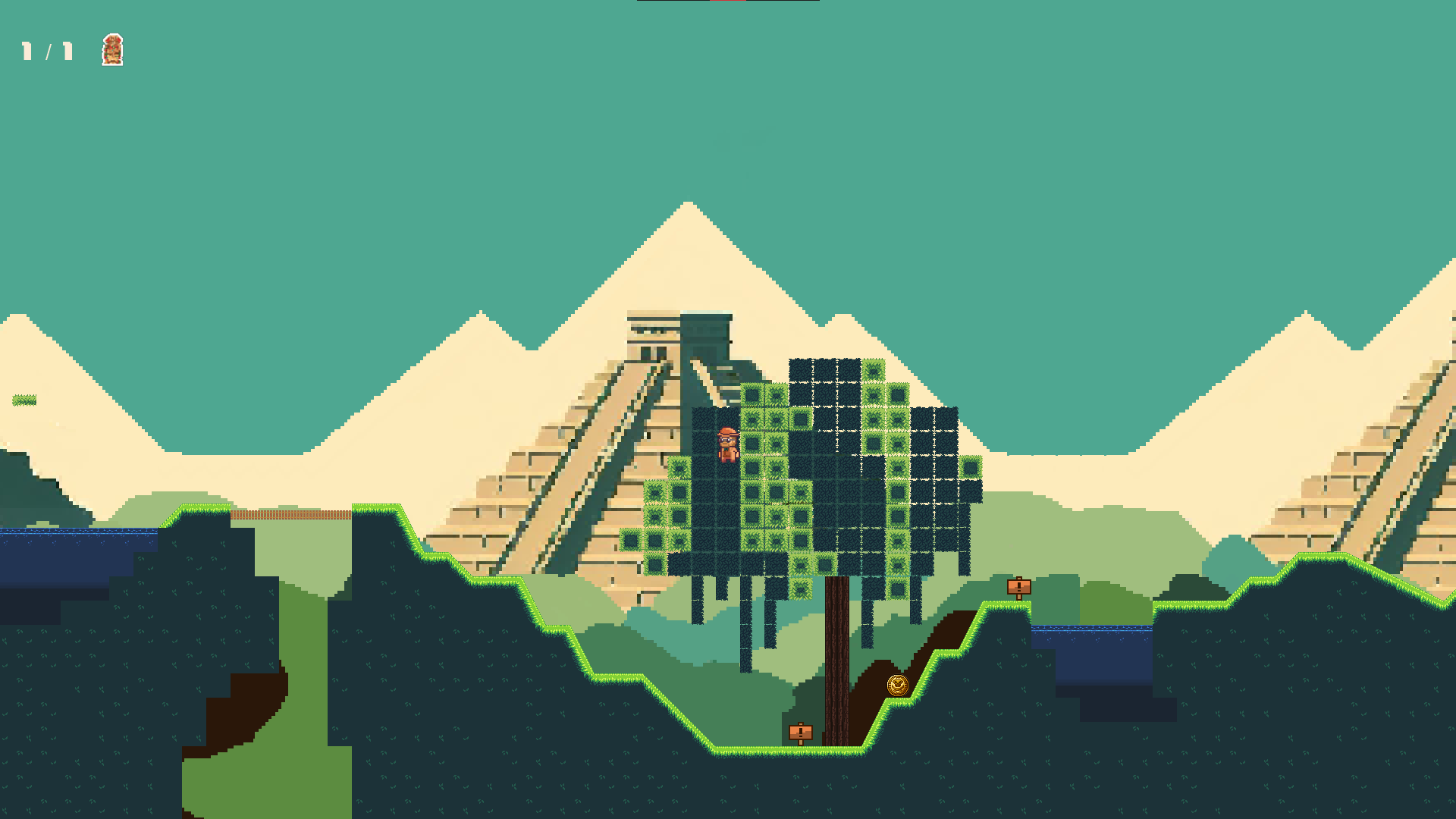


Рисунок 44 - Попадание в лианы

 Рисунок 45 - Исход от попадания в лианы

Пройдя всю вторую карту, персонаж доходит до вождя Майя, которому хочет отдать накопленные части маски. На подходе к нему появляются слова вождя Геннадию Александровичу. Они представлены на рисунке 46.



Рисунок 46 - Разговор вождя Майя и Геннадия Александровича

После нескольких слов от вождя игрок переключает изображение нажатием на клавишу Space. Ему показывается изображение с сюжетной историей на полный экран, пролистав которое, он видит конечное изображение и может только закрыть окно с игрой.

## 3.4 Описание функционала таблицы рекордов

В реализации этой игры данный функционал не предусмотрен.

## 3.5 Алгоритмы программных кодов

Для понимания логики игры с точки зрения программного кода была создана блок-схема, которая иллюстрирует ключевые аспекты взаимодействия персонажа с различными объектами в игровом мире. Эта схема помогает визуализировать основные механики игры и последовательность событий, происходящих в процессе игры.

Блок-схема, которая описывает логику игры, представлена на рисунках 47-48.

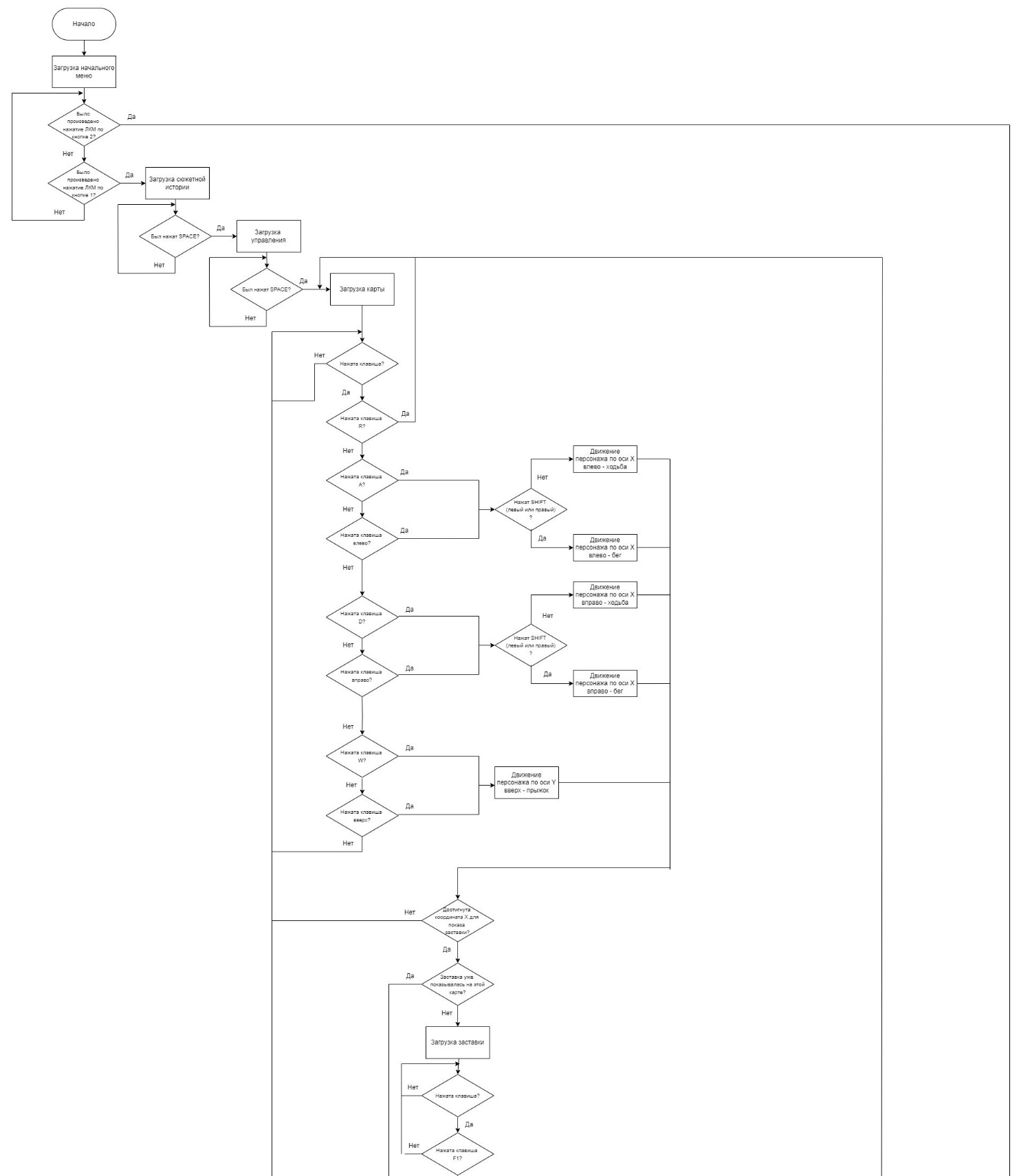


Рисунок 47 - Блок-схема логики игры

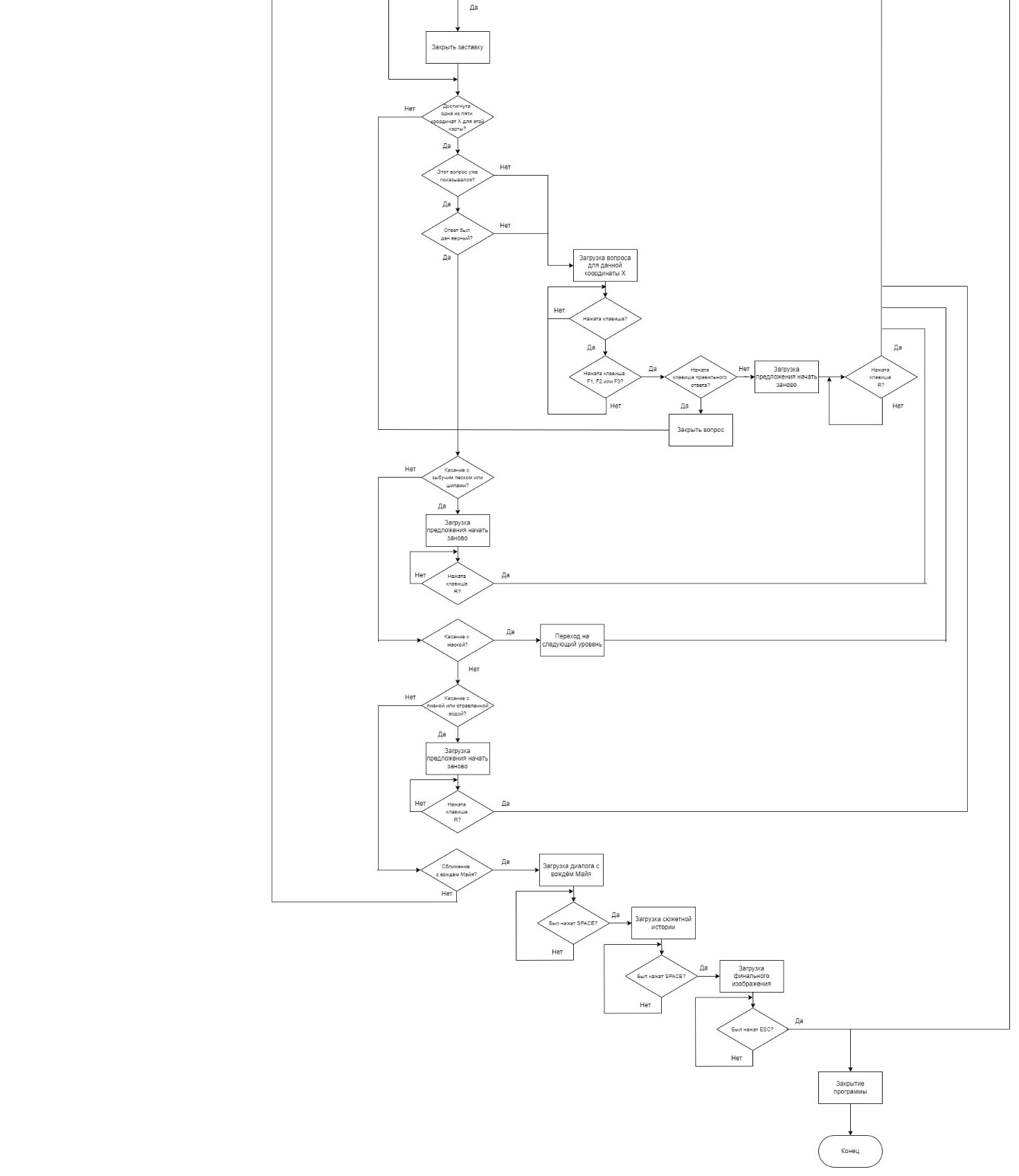


Рисунок 48 - Блок-схема логики игры

## 3.6 Описание функционала программных кодов и представление их скриншотов

С самого начала была установлена библиотека Python Arcade, после чего можно её импортировать. Сначала импортируем библиотеку Arcade с помощью команды import arcade, а также импортируем модуль math, который необходим для выполнения математических операций. Далее определены константы, такие как ширина и высота окна игры, заголовок окна игры, координаты кнопок в начальном меню, параметры персонажа (масштаб, сила гравитации, скорость прыжка и передвижения, множитель бега), время погружения в зыбучие пески и текстуры для анимации персонажа. Также определены позиции остановок в игре, каждая из которых имеет свои координаты, изображение и клавишу для взаимодействия. Фрагмент кода представлен на рисунке 49.

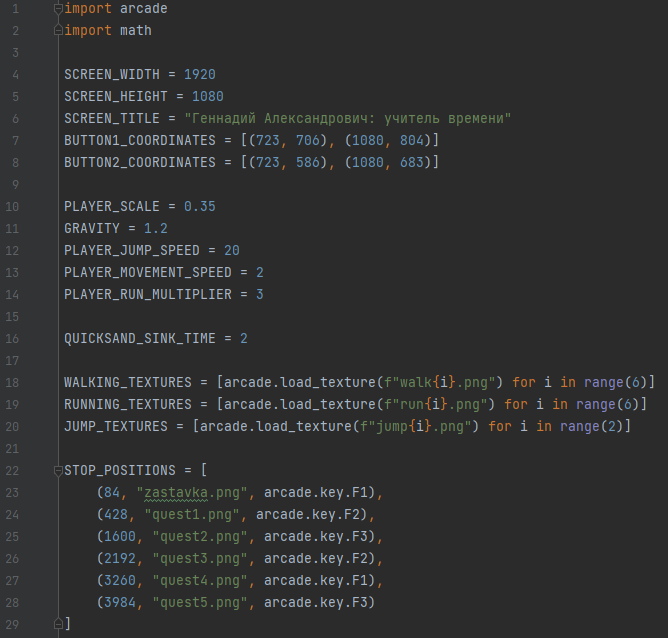


Рисунок 49 – Импорт библиотек, определение констант

Далее создаётся класс MenuView, который наследуется от arcade.View и отвечает за отображение начального меню игры. В методе \_\_init\_\_ загружается текстура фона меню с помощью функции arcade.load\_texture(«begin back.jpg»). Метод on\_draw отвечает за отрисовку экрана, используя функции arcade.start\_render() для начала процесса рендеринга и arcade.draw\_lrwh\_rectangle\_textured() для отрисовки фона на весь экран. Метод on\_mouse\_press обрабатывает нажатия мыши, проверяя координаты клика. Если нажата левая кнопка мыши и клик произошёл в области первой кнопки, создаётся и отображается вид IntroView, если же клик в области второй кнопки, то окно игры закрывается. Метод on\_key\_press реагирует на нажатие клавиш, закрывая окно игры при нажатии клавиши Escape (arcade.key.ESCAPE). Класс представлен на рисунке 50.

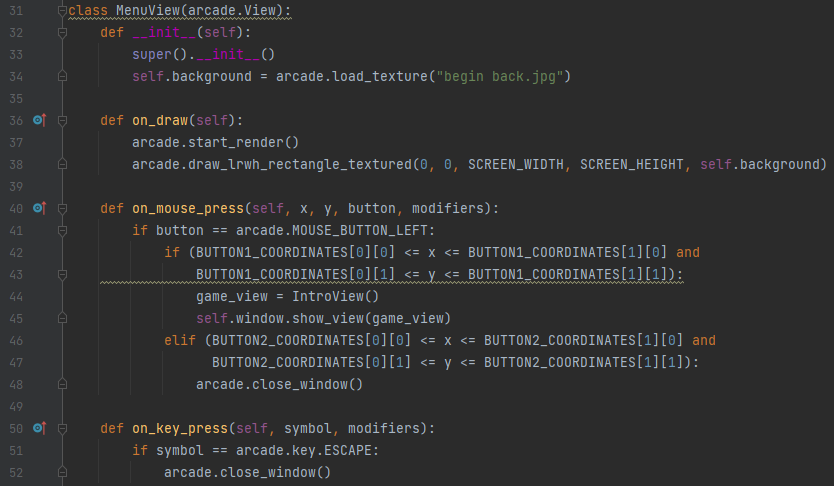


Рисунок 50 – Класс MenuView

Далее создаётся класс IntroView, который также наследуется от arcade.View и предназначен для отображения вводного экрана игры. В методе \_\_init\_\_ загружаются две текстуры для фона вводного экрана (сюжетная история и изображение с управлением) с помощью функции arcade.load\_texture(«back1.jpg») и arcade.load\_texture(«upravleniye.jpg»), и сохраняются в список intro\_images. Изначально текущий индекс изображения (current\_image) устанавливается на 0. Метод on\_draw отвечает за отрисовку экрана, начиная процесс рендеринга с помощью arcade.start\_render() и отрисовывая текущую текстуру фона через arcade.draw\_lrwh\_rectangle\_textured(), покрывая всю область экрана. Метод on\_key\_press обрабатывает нажатия клавиш: при нажатии пробела (arcade.key.SPACE) происходит переключение на следующее изображение, а если текущий индекс изображения становится равным нулю, создаётся и отображается вид GameView, представляющий основной игровой процесс. При нажатии клавиши Escape (arcade.key.ESCAPE) окно игры закрывается. Класс представлен на рисунке 51.

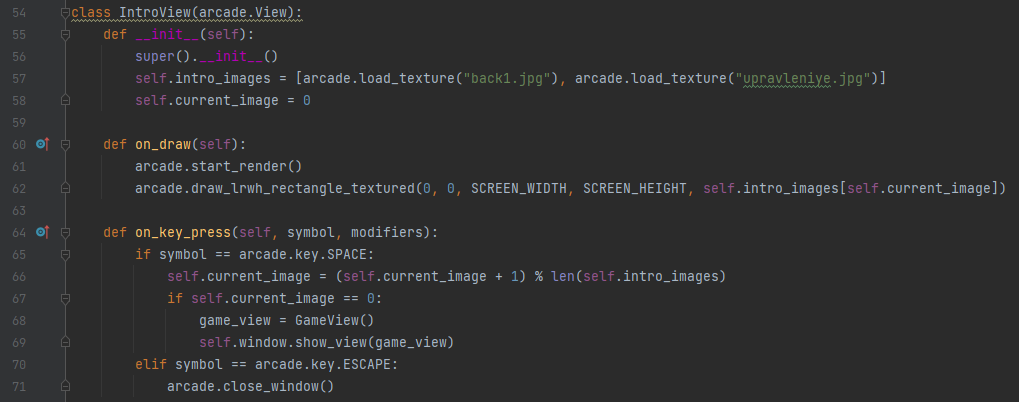


Рисунок 51 – Класс IntroView

Класс GameView наследуется от arcade.View и отвечает за основной игровой процесс. В методе \_\_init\_\_ создается спрайт игрока с текстурой «ded.png» и заданным масштабом (PLAYER\_SCALE), после чего устанавливаются его начальные координаты (center\_x и center\_y). Инициализируются две камеры (camera и gui\_camera), отвечающие за отображение игровой сцены и интерфейса соответственно. Переменные progress и max\_progress отслеживают прогресс игрока, а словарь unlocked\_stop\_positions хранит состояния остановок в игре. Загружается карта из файла «map.json» с помощью метода load\_map. Для управления физикой используется arcade.PhysicsEnginePlatformer, который обрабатывает движения игрока и взаимодействие со слоем «ground», используя заданную гравитацию (GRAVITY). Переменные состояния игрока (player\_state, player\_facing\_left, left\_pressed, right\_pressed, jump\_pressed, run\_pressed, current\_walk\_texture, current\_run\_texture, current\_jump\_texture, texture\_update\_time, texture\_update\_delta, dead, in\_quicksand, quicksand\_timer) контролируют анимацию, направление и физическое состояние игрока. Загружаются текстуры остановок (stop\_images) и устанавливаются начальные значения для состояния взаимодействия с ними (stop\_image\_shown, unlock\_keys\_pressed, current\_stop\_position). Переменные player\_facing\_left\_before\_death, current\_level, maya\_images, showing\_maya\_images, current\_maya\_image\_index, maya\_proximity\_threshold управляют состоянием игры до и после смерти игрока, текущим уровнем (количеством собранных фрагментов) и отображением изображений при подходе к слою «maya» на второй карте. Метод \_\_init\_\_ класса GameView представлен на рисунке 52.



Рисунок 52 – Метод \_\_init\_\_ класса GameView

Метод load\_map отвечает за загрузку и настройку карты игры из файла. Сначала он пытается загрузить карту с помощью функции arcade.load\_tilemap(), и в случае ошибки выводит сообщение о ней и завершает выполнение метода. Если загрузка прошла успешно, создается сцена (arcade.Scene) на основе загруженной карты, и к этой сцене добавляется спрайт игрока. Затем рассчитываются ширина и высота карты в пикселях. Списки quicksand\_list и mask\_list заполняются спрайтами ловушек и маски, соответственно.

В дальнейшем метод устанавливает начальные параметры для маски в конце первой карты, включая начальную высоту, амплитуду и период колебаний. Затем для каждого спрайта в списках ловушек и маски устанавливается хитбокс, если у спрайта есть текстура, иначе выводится сообщение об ошибке. Для обработки физики создается объект arcade.PhysicsEnginePlatformer, который связывает спрайт игрока и слоя «ground», используя заданную гравитацию. Далее, в зависимости от текущего прогресса, загружаются изображения вопросов для остановок и определяются координаты по X, которые будут провоцировать остановку на первой и второй карте, а также корректные и некорректные клавиши для каждой остановки. Словарь unlocked\_stop\_positions очищается и заполняется начальными значениями (False) для правильной работы остановок на каждой карте. Метод load\_map класса GameView представлен на рисунках 53-54.

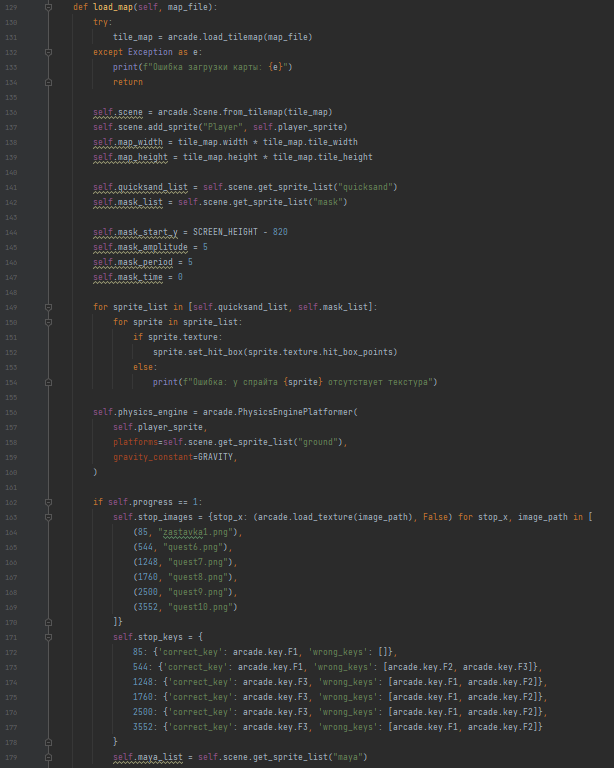


Рисунок 53 – Метод load\_map класса GameView



Рисунок 54 – Метод load\_map класса GameView

Метод on\_draw отвечает за отрисовку всех элементов на экране игры. В начале выполнения метода вызывается arcade.start\_render(), чтобы инициализировать рендеринг. Затем активируется основная камера с помощью self.camera.use(), и сцена игры отрисовывается с помощью self.scene.draw(). Если игрок не находится в слое «quicksand», то также отрисовывается спрайт игрока (self.player\_sprite.draw()). Координаты для отображения текста устанавливаются в переменных text\_position\_x и text\_position\_y. Далее используется GUI-камера (self.gui\_camera.use()) для отображения элементов интерфейса. Загружается текстура маски Майя (arcade.load\_texture(«fullmask1.png»)), которая затем отрисовывается на экране с заданными координатами и масштабом (счётчик собранных осколков). Текущий прогресс игрока отображается на экране с помощью arcade.draw\_text(). Если игрок мёртв, то рисуется затемнённый экран и отображается сообщение о необходимости перезапуска игры. Затем проверяются стоп-позиции, и, если текущая остановка должна быть показана, её изображение отрисовывается в центре экрана. Наконец, если активированы изображения maya, путём сближения с одноимённым слоем, отображается соответствующее изображение из списка self.maya\_images. Метод on\_draw класса GameView представлен на рисунке 55.

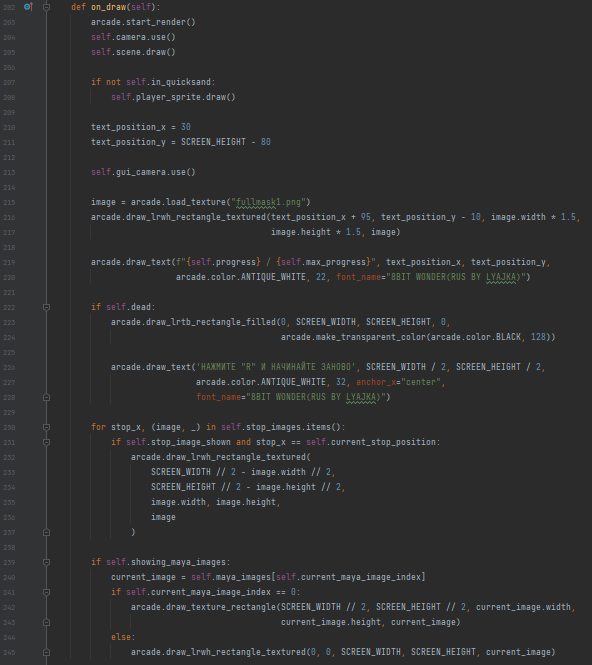


Рисунок 55 – Метод on\_draw класса GameView

Метод on\_key\_press обрабатывает нажатия клавиш и управляет различными состояниями игры в зависимости от текущей ситуации. Если в данный момент показываются изображения maya, нажатие пробела (arcade.key.SPACE) переключает на следующее изображение, если текущее не является последним. Нажатие клавиши Escape (arcade.key.ESCAPE) закрывает окно игры, если показ последнего изображения завершен, или выключает показ изображений maya, возвращаясь к игровому процессу.

Если показы изображений maya отключены, метод обрабатывает стандартные игровые клавиши. Нажатие стрелки влево (arcade.key.LEFT) или клавиши «A» активирует движение влево, а стрелки вправо (arcade.key.RIGHT) или клавиши «D» - движение вправо. Клавиша «W» или стрелка вверх (arcade.key.UP) инициируют прыжок, если это возможно. Нажатие клавиш Shift (arcade.key.LSHIFT или arcade.key.RSHIFT) включает режим бега. Клавиша "R" (arcade.key.R) позволяет возродить игрока в случае его смерти. Escape (arcade.key.ESCAPE) закрывает окно игры. Метод также обрабатывает взаимодействие со стоп-позициями на карте, проверяя правильность выбранного ответа на вопрос и обновляя состояние игры и игрока в соответствии с этим. Метод on\_key\_press класса GameView представлен на рисунке 56.



Рисунок 56 – Метод on\_key\_press класса GameView

Метод on\_key\_release обрабатывает события отпускания клавиш и обновляет состояние игры в зависимости от текущих действий игрока. Если отпущена клавиша влево (arcade.key.LEFT или arcade.key.A), устанавливается флаг self.left\_pressed в значение False. Если при этом не нажата клавиша вправо, состояние игрока переключается на «idle». Аналогичным образом, если отпущена клавиша вправо (arcade.key.RIGHT или arcade.key.D), устанавливается флаг self.right\_pressed в значение False, и если не нажата клавиша влево, состояние игрока также переключается на «idle». Если отпущена клавиша Shift (arcade.key.LSHIFT или arcade.key.RSHIFT), флаг self.run\_pressed устанавливается в значение False, и если ни одна из клавиш для передвижения не нажата, состояние игрока устанавливается в «idle».

Кроме того, метод проверяет отпущенные клавиши, связанные со стоп-позициями в игре. Если отпущена клавиша, связанная с одной из остановок (close\_key), флаг self.unlock\_keys\_pressed для соответствующей остановки устанавливается в значение False. Это гарантирует, что игрок больше не взаимодействует с остановкой после отпускания соответствующей клавиши. Метод on\_key\_release класса GameView представлен на рисунке 57.

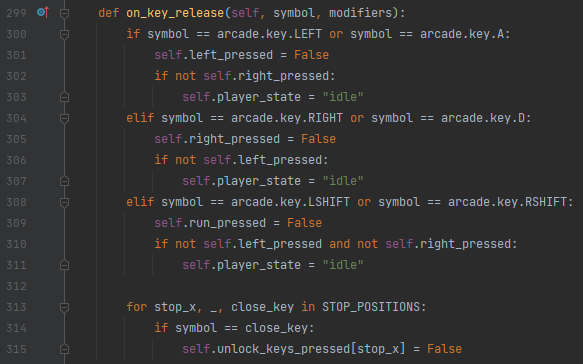


Рисунок 57 – Метод on\_key\_release класса GameView

Метод on\_update обновляет состояние игры каждый кадр, используя аргумент delta\_time для учета времени, прошедшего с последнего обновления. Сначала проверяется, не мёртв ли игрок (self.dead), и если да, выполнение метода прекращается. Затем обновляется физический движок игры (self.physics\_engine.update()), чтобы обработать все физические взаимодействия. Если позиция center\_y спрайта игрока (self.player\_sprite) становится меньше нуля, состояние игрока переключается на «idle», и он помечается как мертвый, сохраняя текущее направление взгляда.

Метод также управляет движением игрока, проверяя нажатие клавиш. Если нажаты клавиши для движения влево или вправо, изменяется скорость движения игрока (change\_x), а также проверяется, активирован ли режим бега (self.run\_pressed). Затем обновляются координаты игрока, чтобы он оставался в пределах карты. Проверка на столкновение с ловушками осуществляется путем создания списка валидных спрайтов и проверки коллизий с игроком. Если игрок входит в зону ловушек, активируется соответствующее состояние (self.in\_quicksand), и запускается таймер. Метод также управляет состоянием игрока в ловушках, вызывая метод sink\_in\_quicksand(delta\_time), если игрок застрял в них. Если игрок находится в состоянии прыжка и может прыгать, его состояние возвращается к «idle». Далее вызываются методы animate\_player(), scroll\_to\_player() и check\_stop\_positions(), чтобы обновить анимацию игрока, прокрутку камеры и проверку позиций остановок соответственно. Если игрок находится вблизи стоп-позиции, отображается соответствующее изображение с вопросом.

Последняя часть метода обрабатывает маску, обновляя её положение на основе синусоидальной функции, и проверяет коллизии с игроком. Если игрок касается маски, происходит переход на следующий уровень. Если игрок находится вблизи спрайта maya, активируется показ изображений. Метод обеспечивает непрерывное обновление состояния игры, управления игроком и проверки взаимодействий с игровыми объектами, создавая динамичный и интерактивный игровой процесс. Метод on\_update класса GameView представлен на рисунках 58-59.

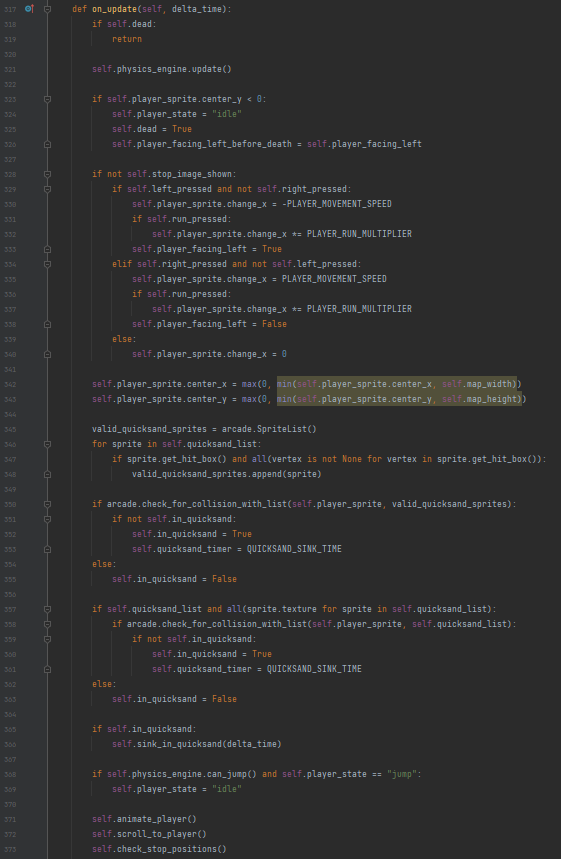


Рисунок 58 – Метод on\_update класса GameView

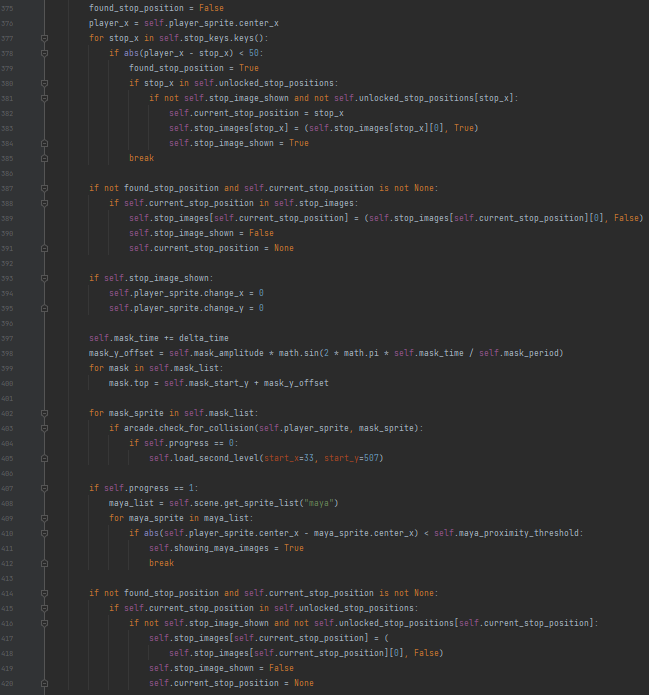


Рисунок 59 – Метод on\_update класса GameView

Метод load\_second\_level загружает второй уровень игры, обновляя текущий прогресс и перемещая игрока в начальную позицию на новом уровне. Сначала устанавливается значение прогресса (self.progress = 1), что указывает на переход на следующий уровень. Затем вызывается метод load\_map(«map1.json»), который загружает карту второго уровня из файла «map1.json». После загрузки карты координаты спрайта игрока (self.player\_sprite) устанавливаются на значения start\_x и start\_y, чтобы переместить игрока в начальную точку нового уровня. Этот метод обеспечивает плавный переход между уровнями, обновляя игровые данные и позицию игрока. Метод load\_second\_level класса GameView представлен на рисунке 60.

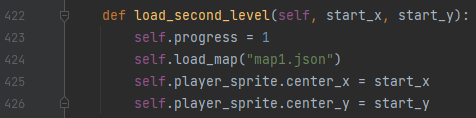


Рисунок 60 – Метод load\_second\_level класса GameView

Метод check\_stop\_positions отвечает за проверку текущих позиций остановок относительно игрока и отображение соответствующих изображений с вопросами. Сначала определяются координата игрока по оси X (player\_x) и направление его взгляда (player\_facing\_left). Если текущая остановка (self.current\_stop\_position) установлена, метод проверяет, находится ли игрок левее остановки и смотрит ли он вправо. В таком случае изображение остановки скрывается (self.stop\_image\_shown = False), и текущая остановка сбрасывается (self.current\_stop\_position = None). В противном случае метод проверяет все остановки в списке STOP\_POSITIONS, чтобы определить, находится ли игрок вблизи любой из них (на расстоянии менее 50 пикселей). Если игрок находится вблизи остановки и смотрит влево, устанавливается текущая остановка и отображается соответствующее изображение.

Методы show\_stop\_image и hide\_all\_stop\_images управляют отображением изображений остановок. Метод show\_stop\_image загружает текстуру изображения с вопросом и добавляет её в словарь self.stop\_images, если она еще не была добавлена. Метод hide\_all\_stop\_images скрывает все изображения вопросов, устанавливая соответствующее значение в словаре self.stop\_images в False. Эти методы обеспечивают корректное отображение и скрытие изображений вопросов в зависимости от положения и действий игрока, создавая интерактивный и динамичный игровой процесс.

Методы check\_stop\_positions, show\_stop\_image и hide\_all\_stop\_images класса GameView представлены на рисунке 61.

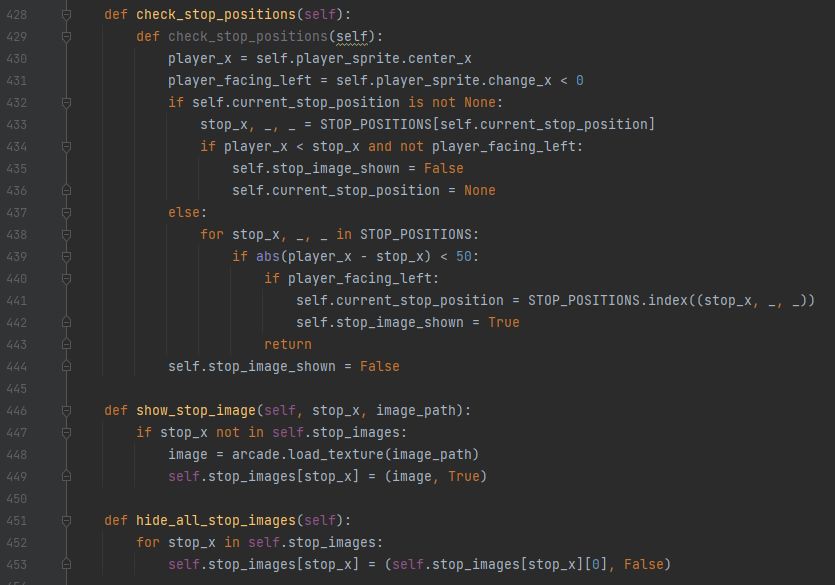


Рисунок 61 - Методы check\_stop\_positions, show\_stop\_image и hide\_all\_stop\_images класса GameView

Метод sink\_in\_quicksand управляет процессом погружения игрока в ловушки. Если таймер ловушки (self.quicksand\_timer) больше нуля, он уменьшается на значение delta\_time, указывающее время, прошедшее с последнего обновления. Во время погружения игрока скорость по оси Y (self.player\_sprite.change\_y) устанавливается в 1, и все флаги управления движением (влево, вправо, прыжок, бег) сбрасываются в False, чтобы игрок не мог двигаться, пока он погружается. Также скорость по оси X (self.player\_sprite.change\_x) сбрасывается в 0, а состояние игрока меняется на «idle». Если таймер истекает, игрок помечается как мертвый (self.dead = True), а направление взгляда игрока перед смертью сохраняется в self.player\_facing\_left\_before\_death. Метод sink\_in\_quicksand класса GameView представлен на рисунке 62.

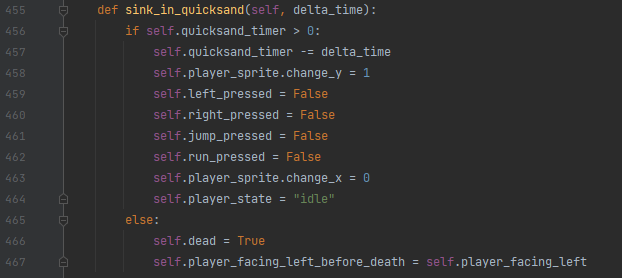


Рисунок 62 - Метод sink\_in\_quicksand класса GameView

Метод animate\_player отвечает за обновление анимации спрайта игрока в зависимости от его текущего состояния. Сначала увеличивается значение texture\_update\_delta, указывающее на прошедшее время с момента последнего обновления текстуры. Если состояние игрока «idle», текстура спрайта устанавливается на изображение «ded.png», с учётом направления взгляда игрока (flipped\_horizontally). В состоянии «walk» проверяется, если прошло достаточно времени для обновления текстуры (texture\_update\_delta >= self.texture\_update\_time), значение texture\_update\_delta сбрасывается, и индекс текущей текстуры ходьбы обновляется циклически. Затем устанавливается соответствующая текстура для спрайта игрока из списка WALKING\_TEXTURES. В состоянии «jump» аналогично обновляется текстура прыжка из списка JUMP\_TEXTURES с учетом времени. В состоянии «run» также проверяется время обновления текстуры и циклически изменяется индекс текущей текстуры бега из списка RUNNING\_TEXTURES. В каждом случае текстура спрайта игрока устанавливается с учётом направления взгляда (flipped\_horizontally). Этот метод обеспечивает плавную анимацию движений игрока, изменяя текстуры в зависимости от состояния и времени, прошедшего с последнего обновления, что делает игровой процесс более реалистичным и визуально привлекательным.

Метод animate\_player класса GameView представлен на рисунке 63.

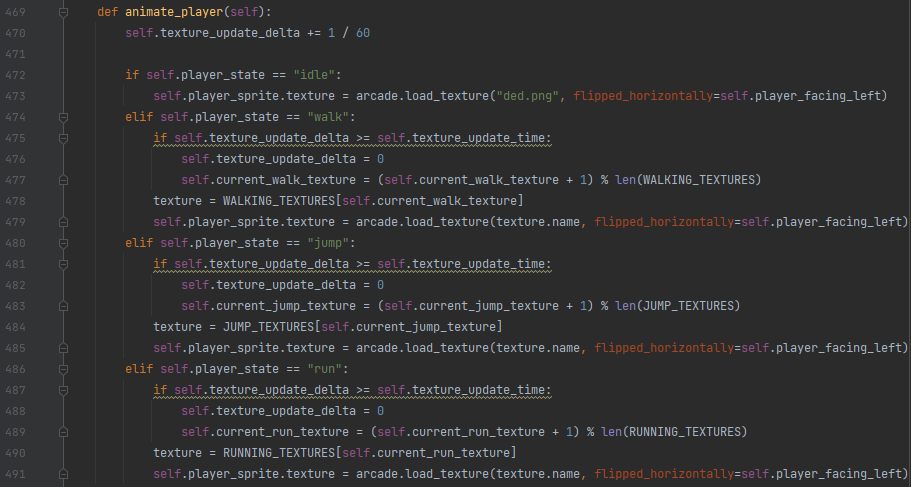


Рисунок 63 - Метод animate\_player класса GameView

Метод respawn\_player отвечает за возрождение игрока после его смерти, устанавливая его начальные координаты (center\_x = 33 и center\_y = 507). Состояние игрока обновляется: self.dead устанавливается в False, что указывает на то, что игрок больше не мёртв; self.stop\_image\_shown и self.current\_stop\_position сбрасываются, скрывая любые отображаемые стоп-позиции и обнуляя текущую. Далее проверяется, находится ли игрок в зоне ловушек при возрождении, и, если да, активируется состояние погружения и устанавливается таймер. Если нет, состояние ловушек сбрасывается. Состояние игрока устанавливается на «idle». Скорости по осям X и Y также сбрасываются до нуля, что предотвращает любое движение сразу после возрождения. Метод respawn\_player класса GameView представлен на рисунке 64.

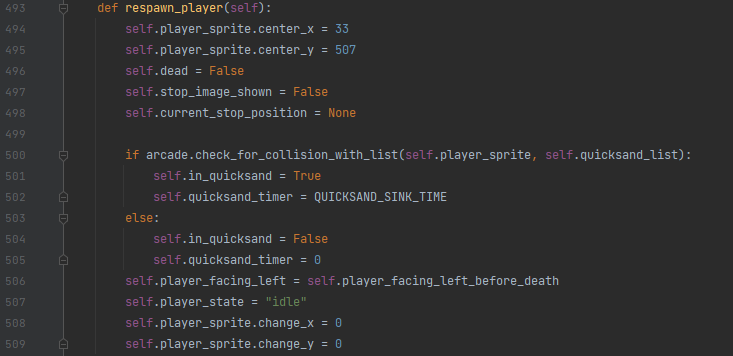


Рисунок 64 - Метод respawn\_player класса GameView

Метод scroll\_to\_player отвечает за плавное перемещение камеры, чтобы центрировать её на спрайте игрока. Сначала вычисляются координаты центра экрана (screen\_center\_x и screen\_center\_y) на основе текущего положения игрока и размеров видимой области камеры. Эти значения корректируются с помощью функций max и min, чтобы камера не выходила за пределы карты, ограничивая перемещение камеры краями карты. После вычисления конечных координат камера плавно перемещается к новому положению с заданной скоростью (0.1), используя метод move\_to. Это обеспечивает следование камеры за игроком, создавая эффект, что игрок всегда находится в центре экрана, что улучшает ориентацию игрока в игровом мире. Метод scroll\_to\_player класса GameView представлен на рисунке 65.

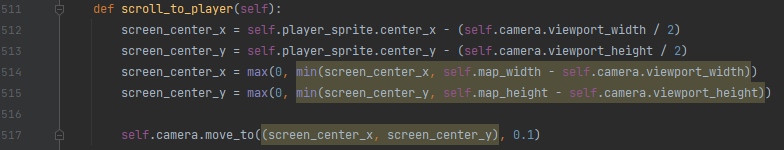


Рисунок 65 - Метод scroll\_to\_player класса GameView

Функция main является точкой входа в игру, где инициализируется основное окно и запускается игровой цикл. Сначала создается объект window класса arcade.Window с заданными параметрами ширины, высоты и заголовка окна, а также устанавливается режим полного экрана (fullscreen=True). Затем создается экземпляр класса MenuView, который отвечает за отображение главного меню игры, и этот вид отображается в окне с помощью метода show\_view(). Наконец, вызывается arcade.run(), который запускает основной цикл обработки событий и обновления экрана, позволяя игре функционировать. Если этот скрипт запускается напрямую, выполняется функция main, что обеспечивает старт игры. Код для запуска и проверки программы представлен на рисунке 66.

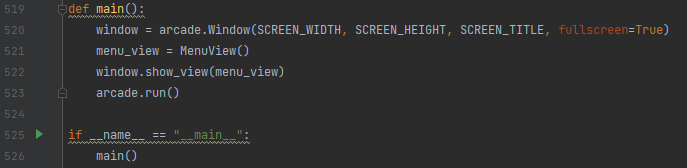


Рисунок 66 - Код для запуска и проверки программы

## 3.7 Основные принципы оценки результата завершения игры

Во-первых, успешное завершение игры определяется достижением финального уровня («Джунгли Мексики») и выполнением всех поставленных задач (сбор фрагмента маски). Игрок должен пройти через все этапы, начиная с погружения в сюжет и ознакомления с управлением, и завершая диалогом с вождём Майя. Каждый уровень представляет собой уникальную задачу, выполнение которой свидетельствует о прогрессе игрока.

Во-вторых, важным аспектом оценки является способность игрока взаимодействовать с игровыми объектами и механиками. Игрок должен правильно отвечать на вопросы по истории и преодолевать ловушки, такие как зыбучие пески, шипы, лианы и отравленные воды. Успешное использование этих элементов демонстрирует понимание игроком игровых принципов и его способность адаптироваться к меняющимся условиям игры.

В-третьих, важным критерием является количество попыток, необходимых для прохождения игры. Меньшее количество попыток указывает на высокий уровень мастерства и способность игрока быстро приспосабливаться к игровым условиям. В случае гибели персонажа или необходимости перезапуска уровня, игрок получает шанс улучшить свои навыки и попытаться пройти уровень заново.

Наконец, общее время, затраченное на прохождение игры, является значимым показателем. Быстрое прохождение с минимальным количеством ошибок свидетельствует о высоком уровне навыков и знаний по истории. Игроки, которые тратят меньше времени на прохождение уровней, демонстрируют более высокий результат, что является показателем их мастерства.

## 3.8 Описание особенностей взаимодействия компьютерных игроков

Взаимодействие игрока с игрой в «Геннадий Александрович: учитель времени» происходит через различные клавиши управления, каждая из которых вызывает определённые действия персонажа. Основные действия включают в себя: движение персонажа, прыжок, ускорение, ответы на вопросы, взаимодействие с ловушками, рестарт и закрытие игры.

При нажатии клавиш «влево» (A или LEFT) и «вправо» (D или RIGHT) главный герой перемещается в соответствующем направлении. Клавиша «вверх» (W или UP) используется для прыжков персонажа. Клавиша Shift (LSHIFT или RSHIFT) в совокупности с нажатыми клавишами движения заставляет персонажа ускоряться. Взаимодействие со всплывающими вопросами осуществляется при помощи клавиш F1, F2 и F3.

Если игрок попадает в ловушку, его персонаж начинает медленно туда погружаться, после чего гибели уже не избежать и экран вскоре затемнится и на нём появится текст: «НАЖМИТЕ «R» И НАЧИНАЙТЕ ЗАНОВО». После смерти персонажа пользователь может возродиться, нажав клавишу R, и игра будет продолжаться с начальной точки текущей карты.

Взаимодействие пользователя с программой включает следующие аспекты:

* при запуске игры происходит инициализация игрового окна и загрузка начального меню. С помощью него можно либо выйти из игры, либо начать её, погрузившись в сюжет и ознакомившись с управлением, сменив соответствующие изображения нажатием на пробел (SPACE). После чего игра будет запущена;
* управление персонажем осуществляется с помощью клавиш «влево» (A или LEFT), «вправо» (D или RIGHT), «вверх» (W или UP), Shift (LSHIFT или RSHIFT);
* взаимодействие с вопросами осуществляется при помощи клавиш F1, F2 и F3. Один из вариантов ответов будет верный и закроет изображение с вопросом, остальные - неверные, которые вызовут смерть персонажа, и игра предложит начать заново нажатием на клавишу R. Для каждого вопроса клавиша правильного ответа своя;
* в течение игры можно в любое время закрыть её, нажав на клавишу Escape (ESCAPE), и переместить персонажа на начальную точку текущей карты, нажав на клавишу R;
* на каждом уровне игрок должен избегать ловушек, чтобы достичь цели уровня. Попав в них, ему будет предложено начать заново нажатием на клавишу R;
* игра загружает следующий уровень при соприкосновении с осколком маски Майя;
* когда вторая карта пройдена, игрок начинает диалог с вождём Майя. После нажатия на пробел (SPACE) открывается изображение, повествующее о концовке сюжета, сменив его той же клавишей, пользователь видит сообщение о возможном дальнейшем продолжении игры. Закрыть игру можно только клавишей Escape (ESCAPE).

# ЗАКЛЮЧЕНИЕ

В ходе прохождения учебной (проектно-технологической) практики были изучены и применены как теоретические, так и практические аспекты разработки компьютерной игры, используя возможности языка программирования Python. В результате работы была создана компьютерная игра «Геннадий Александрович: учитель времени».

В процессе тестирования игры были проверены основные механики и функциональные возможности. Основные выводы по результатам тестирования игры «Геннадий Александрович: учитель времени»:

1. все игровые механики функционировали корректно, обеспечивая плавное и непрерывное игровое взаимодействие;
2. персонаж успешно выполнял перемещения, прыжки, ускорения и взаимодействия с объектами, что обеспечивало реалистичность игрового процесса;
3. все изображения по координатам отработали корректно;
4. слои с ловушками работали правильно;
5. по завершении всех уровней игра корректно завершалась с демонстрацией финального сообщения.

Тем не менее, в ходе тестирования были выявлены некоторые проблемы в коде: игра имела множество зависаний на втором уровне. Для решения этой проблемы были удалены лишние проверки и неактивные переменные. Помимо этого, были сокращены некоторые циклы.

Тестирование подтвердило, что игра «Геннадий Александрович: учитель времени» реализована в соответствии с первоначальной концепцией. Использование современных инструментов программирования и библиотеки Arcade позволило создать захватывающий 2D платформер с множеством графических элементов и разнообразными механиками.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Swift // Apple Developer URL: https://developer.apple.com/swift/ (дата обращения: 15.05.2024).
2. Язык программирования C++ // сравни URL: https://www.sravni.ru/kursy/info/yazyk-programmirovaniya-c/ (дата обращения: 15.05.2024).
3. Что такое C#: плюсы и минусы язык // GeekBrains URL: https://gb.ru/blog/chto-takoe-c/ (дата обращения: 15.05.2024).
4. Java // SKILLFACTORY MEDIA URL: https://blog.skillfactory.ru/glossary/java/ (дата обращения: 15.05.2024).
5. Выбираем язык программирования: какие из них популярны, для чего они нужны и с чего начать изучение // медиа нетологии URL: https://netology.ru/blog/03-2022-programming-languages (дата обращения: 15.05.2024).
6. Иванов С. С. Начнем. Python. Просто о сложном. - СПб: Издательство Наука и Техника, 2023. - 368 с.
7. 5 принципов хорошего интерфейса в играх // SkillBox Media URL: https://skillbox.ru/media/gamedev/5\_printsipov\_khoroshego\_interfeysa\_v\_igrakh/ (дата обращения: 19.05.2024).
8. Принципы разработки игрового интерфейса // DTF URL: https://dtf.ru/gamedev/70438-principy-razrabotki-igrovogo-interfeisa (дата обращения: 19.05.2024).
9. Пользовательский интерфейс // AskUsers URL: https://askusers.ru/blog/obuchenie/polzovatelskiy-interfeys/ (дата обращения: 19.05.2024).
10. Основы создания спрайтовой графики в играх // OTUS JOURNAL URL: https://otus.ru/journal/osnovy-sozdaniya-sprajtovoj-grafiki-v-igrah/ (дата обращения: 19.05.2024).
11. Основные принципы и инструменты 3D моделирования // 3D ARCMAN URL: https://arcman.ru/osnovnye-printsipy-i-instrumenty-3d-modelirovaniya/ (дата обращения: 19.05.2024).
12. Разница между 2D и 3D графикой // Соловей БЛОГ URL: https://solovey.ru/blog/raznicza-mezhdu-2d-i-3d-grafikoj/ (дата обращения: 19.05.2024).
13. Истоки Python // Хабр URL: https://habr.com/ru/articles/702458/ (дата обращения: 22.05.2024).
14. Язык программирования Python: особенности и перспективы // GeekBrains URL: https://gb.ru/blog/python/ (дата обращения: 22.05.2024).
15. Pygame // SKILLFACTORY MEDIA URL: https://blog.skillfactory.ru/glossary/pygame/ (дата обращения: 22.05.2024).
16. The Python Arcade Library // Python Arcade 2.6.17 URL: https://api.arcade.academy/en/latest/ (дата обращения: 22.05.2024).
17. 6 основных библиотек для программирования на Python // Хабр URL: https://habr.com/ru/articles/481432/ (дата обращения: 22.05.2024).
18. Обзор игрового движка Panda3D и его особенностей // DTF URL: https://dtf.ru/gamedev/108285-obzor-igrovogo-dvizhka-panda3d-i-ego-osobennostei (дата обращения: 22.05.2024).
19. Tiled Map Editor // GcUp URL: https://gcup.ru/load/utilities/tiled\_qt/4-1-0-565 (дата обращения: 22.05.2024).
20. Как работать с WebSocket в Python // skypro media URL: https://sky.pro/media/kak-rabotat-s-websocket-v-python/ (дата обращения: 22.05.2024).
21. Обзор наиболее популярных инструментов и библиотек для машинного обучения: Python, TensorFlow, Scikit-learn и пр. // DS WORKS URL: https://dsworks.ru/topic/e15a63d9-3dd2-4ee6-825b-c5cf0e4b37e6 (дата обращения: 22.05.2024).
22. Математические функции и модуль math // METANIT.COM URL: https://metanit.com/python/tutorial/6.2.php (дата обращения: 22.05.2024).