**Лабораторная работа №8. Игровые скрипты**

***Цель работы:*** Разработка, добавление и настройка игровых скриптов.

***1.1. Общие теоретические сведения***

Разработка, добавление и настройка игровых скриптов в Unity являются важным аспектом создания игр. С помощью скриптов можно определять поведение объектов, обрабатывать пользовательский ввод, управлять анимацией, реализовывать логику игровых механик и многое другое. Вот более развернутые теоретические сведения по этой теме:

**Разработка игровых скриптов:**

* **Язык программирования:** Unity поддерживает несколько языков программирования, но наиболее распространенным является C#. Вы можете использовать C# для написания скриптов для управления поведением игровых объектов.
* **Основы программирования:** Для разработки игровых скриптов необходимо иметь базовые знания программирования, такие как понимание переменных, условий, циклов, функций и т.д.
* **API Unity:** Unity предоставляет множество встроенных классов и методов, которые облегчают создание игровых скриптов. Это включает в себя классы для управления игровыми объектами, физикой, анимацией, звуком и многим другим.
* **Использование компонентов:** Скрипты в Unity обычно добавляются к игровым объектам в качестве компонентов. Вы можете создать скрипт и прикрепить его к объекту, чтобы определить его поведение.

**Добавление игровых скриптов в Unity:**

* **Создание нового скрипта:** Для создания нового скрипта выберите "Assets" -> "Create" -> "C# Script" в главном меню Unity. Затем введите имя для своего скрипта.
* **Редактирование скрипта:** Откройте созданный скрипт в вашей любимой интегрированной среде разработки (IDE), такой как Visual Studio или Visual Studio Code. В этом скрипте вы определите логику, которую хотите реализовать.
* **Написание кода:** Ваш код должен определять методы и переменные, которые определяют поведение вашего игрового объекта. Например, вы можете написать методы для обработки пользовательского ввода, перемещения объекта, взаимодействия с окружающим миром и т.д.
* **Добавление скрипта к игровому объекту:** Чтобы добавить скрипт к игровому объекту, просто перетащите скрипт из окна проекта Unity на объект в сцене или в окне "Inspector".

**Настройка игровых скриптов в Unity:**

* **Параметры скрипта в окне "Inspector":** После добавления скрипта к игровому объекту, вы можете настроить его параметры в окне "Inspector". Некоторые параметры могут быть открыты для изменения непосредственно из редактора Unity.
* **Отладка скриптов:** Unity предоставляет инструменты для отладки скриптов. Вы можете использовать методы отладки, такие как вывод на консоль (Debug.Log), точки останова (breakpoints), а также отслеживание переменных и состояний объектов.
* **Тестирование и итерация:** После настройки скрипта важно протестировать его в режиме игры, чтобы убедиться, что он работает правильно. При необходимости вносите коррективы в код и продолжайте итерации до достижения желаемого результата.

Разработка, добавление и настройка игровых скриптов в Unity требует понимания основ программирования и API Unity, а также умения адаптировать логику для конкретных потребностей вашей игры.

***1.2 Написание скрипта для управления персонажем от первого лица***

Вот пример простого скрипта на языке C#, который реализует управление от первого лица (FPS) для персонажа в Unity. Этот скрипт обрабатывает ввод от клавиатуры и мыши для управления перемещением и поворотом персонажа:

using UnityEngine;

public class FPSController : MonoBehaviour

{

public float movementSpeed = 5f;

public float mouseSensitivity = 2f;

private CharacterController characterController;

private Camera playerCamera;

private float verticalRotation = 0f;

void Start()

{

characterController = GetComponent<CharacterController>();

playerCamera = GetComponentInChildren<Camera>();

Cursor.lockState = CursorLockMode.Locked;

Cursor.visible = false;

}

void Update()

{

// Поворот персонажа по горизонтали (вокруг оси Y)

float mouseX = Input.GetAxis("Mouse X") \* mouseSensitivity;

transform.Rotate(0f, mouseX, 0f);

// Поворот камеры по вертикали (вокруг оси X)

float mouseY = Input.GetAxis("Mouse Y") \* mouseSensitivity;

verticalRotation -= mouseY;

verticalRotation = Mathf.Clamp(verticalRotation, -90f, 90f);

playerCamera.transform.localRotation = Quaternion.Euler(verticalRotation, 0f, 0f);

// Движение персонажа

float horizontalMovement = Input.GetAxis("Horizontal") \* movementSpeed;

float verticalMovement = Input.GetAxis("Vertical") \* movementSpeed;

Vector3 movement = new Vector3(horizontalMovement, 0f, verticalMovement);

movement = transform.rotation \* movement;

characterController.Move(movement \* Time.deltaTime);

}

}

В этом скрипте используются два основных ввода: мышь для управления поворотом персонажа и клавиатура для управления его движением. Помимо этого, ввод по оси Y открывает или закрывает замок курсора, что позволяет игроку свободно двигать мышью в окне игры без ограничений.

Чтобы использовать этот скрипт, просто добавьте его к игровому объекту, представляющему вашего персонажа в сцене Unity. Затем настройте параметры скрипта, такие как скорость движения и чувствительность мыши, в инспекторе объекта, к которому прикреплен скрипт.

***1.3. Пример выполнения задания***

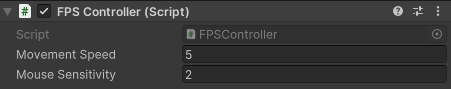


Рис 1.1 Скрипт на игровом объекте

**Лабораторная работа №9. Физические свойства объектов**

***Цель работы:*** Организация взаимодействия объектов за счет добавления гравитации и массы к объекту.

* 1. ***Общие теоретические сведения***

Организация взаимодействия объектов в Unity за счет добавления гравитации и массы - это важный аспект создания реалистичной физики в играх.

**Гравитация в Unity:**

* **Физический движок Unity:** Unity использует встроенный физический движок, который автоматически учитывает гравитацию. Это позволяет объектам в вашей игре реагировать на гравитационное поле и падать вниз, когда ничего не поддерживает их.
* **Настройка гравитации:** Гравитация в Unity представляется величиной, указанной в параметрах проекта. По умолчанию, ось Y направлена вниз, что соответствует стандартной земной гравитации. Однако вы можете настраивать это значение в зависимости от потребностей вашей игры.
* **Воздействие на объекты:** Гравитация воздействует на все объекты в вашей сцене, которые имеют компонент Rigidbody или являются объектами с физическим телом. Это позволяет им свободно падать вниз и взаимодействовать с другими объектами, окружающими их.

**Масса объекта и ее влияние:**

* **Определение массы:** Масса объекта в Unity определяется с помощью компонента Rigidbody. Этот компонент также определяет другие физические характеристики объекта, такие как инерция и сопротивление движению.
* **Взаимодействие с гравитацией:** Масса объекта влияет на его поведение под воздействием гравитации. Чем больше масса, тем больше усилий требуется, чтобы изменить его скорость и направление движения. Это приводит к более реалистичному поведению объектов с разной массой в игровом мире.
* **Коллизии и столкновения:** Масса также влияет на столкновения и коллизии объектов в сцене Unity. Объекты с большей массой будут оказывать большее воздействие при столкновении и могут перемещать или поворачивать более легкие объекты.

**Организация взаимодействия объектов:**

* **Управление параметрами физических тел**: В Unity можно настраивать различные параметры физических тел объектов, такие как их масса, трение, отскок и другие характеристики. Это позволяет создавать разнообразные эффекты во время взаимодействия объектов в игровом мире.
* **Применение сил и импульсов:** Unity позволяет программно применять силы и импульсы к объектам в сцене. Это может быть использовано для симуляции различных физических эффектов, таких как толчки, удары или гравитационное притяжение между объектами.
* **Физические материалы:** Unity также позволяет назначать физические материалы объектам, которые определяют их поведение при коллизиях и столкновениях. Эти материалы могут изменять параметры физического взаимодействия, такие как трение, упругость и т.д.

Организация взаимодействия объектов в Unity за счет добавления гравитации и массы представляет собой важный аспект создания реалистичной физики и поведения объектов в игровом мире. Понимание этих концепций позволяет разработчикам создавать увлекательные и реалистичные игровые сценарии.

***1.2 Добавление гравитации и массы для объекта***

**Компонент Rigidbody:**

* Rigidbody - это компонент, который добавляется к объектам для симуляции их физического поведения в Unity.
* Для добавления Rigidbody к объекту выберите этот объект в сцене, а затем в окне "Inspector" нажмите кнопку "Add Component" и выберите "Physics" -> "Rigidbody".

**Настройка параметров Rigidbody:**

* После добавления Rigidbody можно увидеть набор параметров, включая "Use Gravity" (Использовать гравитацию). Нужно убедиться, что этот параметр включен, чтобы объект подчинялся гравитации.
* Также можно настроить другие параметры Rigidbody, такие как масса, трение, отскок и т.д., чтобы дополнительно настроить физическое поведение объекта.

**Проверка настройки гравитации**:

После того, как был добавлен Rigidbody и включен параметр "Use Gravity", объект должен начать подчиняться гравитации и падать вниз.

**Добавление массы:**

Масса объекта влияет на его взаимодействие с другими объектами в сцене. Объекты с большей массой будут иметь большее влияние на окружающие объекты, и им будет тяжелее подняться или передвинуться.

**Настройка параметра массы в Rigidbody:**

* В компоненте Rigidbody есть параметр "Mass" (Масса). Установите значение этого параметра в соответствии с желаемым весом объекта.
* Чем больше значение массы, тем тяжелее будет объект, и тем сильнее он будет влиять на другие объекты в сцене.

**Дополнительные настройки:**

Помимо массы, можно настроить другие параметры, такие как "Drag" (Сопротивление движению) и "Angular Drag" (Сопротивление вращению), чтобы дополнительно управлять физическим поведением объекта.

**Проверка воздействия массы:**

После настройки массы объекта в его компоненте Rigidbody можно наблюдать, как он взаимодействует с другими объектами в сцене, учитывая свою массу и физические свойства.

Добавление гравитации и массы для объекта в Unity позволяет создавать более реалистичные и интерактивные игровые миры, где объекты ведут себя подобно их реальным аналогам. Это особенно полезно при создании симуляций физических объектов, игровых персонажей и многих других элементов игры.

***1.3. Пример выполнения задания***

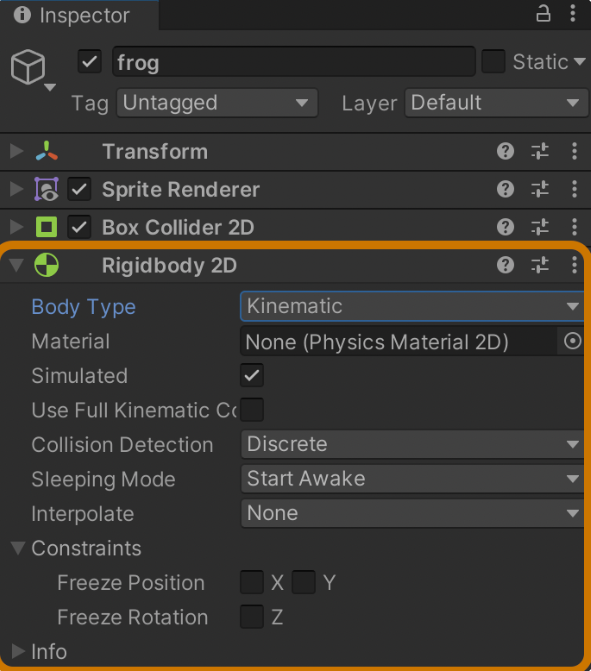


Рис 1.1 Компонент Rigidbody на объекте

**Лабораторная работа №10. Обработка препятствий в виртуальных мирах**

***Цель работы:*** Добавление столкновения объектов с поверхностью.

* 1. ***Общие теоретические сведения***

Добавление столкновений объектов с поверхностью в Unity является важным аспектом создания реалистичной физики и интерактивного поведения в игре.

**Коллизии в Unity:**

**Что такое коллизии:** В игровой среде коллизии происходят, когда два объекта взаимодействуют друг с другом, соприкасаясь или пересекаясь в пространстве. В Unity коллизии позволяют объектам реагировать на другие объекты и окружающую среду.

**Компонент Collider:** Для того чтобы объекты могли взаимодействовать с другими объектами в сцене, им необходим компонент Collider. Collider - это компонент, который определяет геометрию объекта и используется для обнаружения столкновений.

**Добавление столкновений с поверхностью:**

**Выбор подходящего коллайдера:** В Unity существует несколько типов коллайдеров, каждый из которых подходит для различных типов объектов и сценариев.

* **Box Collider:** Используется для объектов с прямоугольной формой.
* **Sphere Collider:** Подходит для сферических объектов.
* **Capsule Collider:** Применяется для объектов, имеющих форму капсулы (например, персонажей).
* **Mesh Collider:** Позволяет использовать собственную геометрию модели объекта для обнаружения столкновений.

**Добавление коллайдера к объекту:** Чтобы добавить коллайдер к объекту, выберите объект в сцене, затем в окне "Inspector" нажмите кнопку "Add Component" и выберите соответствующий тип коллайдера из списка.

**Настройка коллайдера:** После добавления коллайдера вы можете настроить его параметры в окне "Inspector". Это может включать в себя изменение размера, формы и ориентации коллайдера для лучшего соответствия геометрии объекта.

**Тестирование столкновений:** После добавления коллайдера к объекту вы можете проверить, как он взаимодействует с другими объектами в сцене, перемещая или вращая объект в режиме редактирования или запуская игру для тестирования столкновений в реальном времени.

Добавление столкновений объектов с поверхностью в Unity позволяет создавать реалистичное поведение объектов в игровом мире. Правильное использование коллайдеров и настройка их параметров позволяют создавать точные и надежные столкновения, что важно для создания интересной и увлекательной игровой среды.

***1.2 Добавление коллайдеров***

**Добавьте компонент Collider:**

В окне "Inspector" выбранного объекта нажмите кнопку "Add Component".

В поисковой строке введите название нужного коллайдера:

* Например, если если необходимо добавить коллайдер прямоугольной формы, введите "Box Collider".
* Выберите соответствующий компонент коллайдера из списка результатов. Коллайдер будет добавлен к выбранному объекту.

**Настройте параметры коллайдера:**

* После добавления коллайдера можно настроить его параметры в окне "Inspector". Это может включать в себя изменение размера, формы и ориентации коллайдера для лучшего соответствия геометрии объекта.
* Некоторые параметры, такие как "Is Trigger" (является триггером), позволяют объектам проходить сквозь друг друга, что может быть полезным для определенных типов взаимодействия.

**Повторите для других объектов:**

Повторите шаги 1-3 для других объектов, которые должны сталкиваться с поверхностью. Может потребоваться добавить коллайдеры ко всем объектам, которые должны участвовать в столкновениях.

**Тестирование столкновений:**

После добавления коллайдеров к объектам можно проверить, как они взаимодействуют друг с другом в сцене. Можно перемещать объекты, чтобы увидеть, как они взаимодействуют при столкновениях, или запустить игру для более точного тестирования.

Добавление столкновений объектов с поверхностью в Unity позволяет создавать интересные и реалистичные сценарии в игре. Правильное использование коллайдеров и настройка их параметров помогает создавать убедительные и точные столкновения между объектами.

***1.3. Пример выполнения задания***

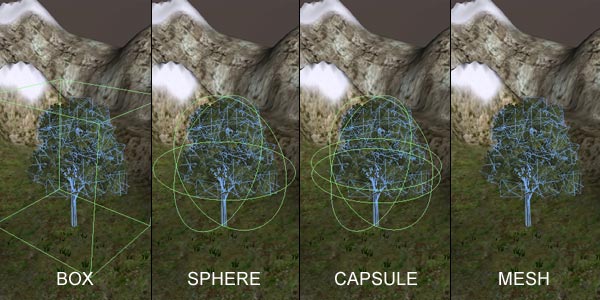
******

Рис 1.1 Разные типы коллайдеров

**Лабораторная работа №11 Консоль в среде разработки**

***Цель работы:*** Сделать анализ скриптовых ошибок.

* 1. ***Общие теоретические сведения***

**Консоль в Unity** - это инструмент, который играет ключевую роль в процессе разработки игр, особенно при анализе скриптовых ошибок. Давайте разберемся более подробно, как именно консоль помогает разработчикам в этом процессе:

**1. Отслеживание ошибок скриптов:**

Когда разработчики пишут скрипты на языке программирования, поддерживаемом Unity (например, C#), ошибки могут возникать из-за синтаксических ошибок, ошибок в логике или неверного использования API Unity. Консоль позволяет быстро обнаруживать и отображать эти ошибки, указывая на конкретные строки кода, в которых они произошли.

**2. Анализ предупреждений:**

Помимо фатальных ошибок, консоль также выводит предупреждения, которые указывают на потенциальные проблемы в коде. Это могут быть предупреждения о неоптимальном использовании ресурсов, неправильном использовании переменных и других аспектах, которые могут негативно повлиять на работу игры.

**3. Отладка и исправление ошибок:**

Консоль предоставляет разработчикам информацию о месте и типе возникших ошибок, что позволяет им быстро перейти к соответствующему участку кода и исправить проблему.

Кроме того, консоль также позволяет разработчикам выводить собственные отладочные сообщения и журналы, которые могут помочь в процессе отладки сложных частей кода.

**4. Улучшение качества кода:**

Путем анализа сообщений об ошибках и предупреждений, выводимых в консоль, разработчики могут улучшить качество своего кода, исправив потенциальные проблемы и обеспечив более эффективную работу скриптов.

**5. Взаимодействие с другими инструментами:**

Консоль в Unity также интегрирована с другими инструментами разработки, такими как инспектор, где разработчики могут просматривать переменные в реальном времени, что облегчает процесс отладки.

**6. Управление и фильтрация сообщений:**

Консоль также предоставляет различные фильтры для отображения только определенных типов сообщений (например, только ошибок или только предупреждений), что делает процесс анализа ошибок более удобным и эффективным.

Консоль в Unity является мощным инструментом для анализа скриптовых ошибок и предупреждений, который помогает разработчикам быстро обнаруживать и исправлять проблемы в коде, улучшая качество и эффективность процесса разработки игр. Отслеживание ошибок в консоли является неотъемлемой частью разработки игр в Unity и способствует созданию более стабильных и высококачественных игровых продуктов.

***1.2 Консоль и анализ скриптовых ошибок***

**1. Открытие консоли:**

* Консоль в Unity можно открыть, выбрав в верхнем меню пункт "Window" -> "General" -> "Console", или просто нажав сочетание клавиш Ctrl + Shift + C (Cmd + Shift + C на macOS).
* После этого откроется окно консоли, где будут отображаться сообщения об ошибках, предупреждениях и другой отладочной информации.

**2. Анализ скриптовых ошибок:**

* После того, как вы запустите вашу игру или сцену в редакторе Unity, консоль будет отображать любые скриптовые ошибки, которые возникли в процессе компиляции или выполнения.
* Скриптовые ошибки будут отмечены красным цветом и содержат сообщения о том, какую проблему они обнаружили и в каком файле и на какой строке она произошла.

**3. Интерпретация ошибок:**

* Чтение сообщений об ошибках в консоли поможет вам определить, какие именно проблемы возникли в вашем коде. Ошибки могут быть вызваны синтаксическими ошибками (например, неправильным использованием синтаксиса языка программирования), ошибками компиляции (например, неверным импортом библиотек) или ошибками выполнения (например, попыткой обращения к несуществующей переменной).
* Каждое сообщение об ошибке будет содержать информацию о том, что именно вызвало ошибку, и часто будет указано конкретное место в коде, где произошла проблема.

**4. Исправление ошибок:**

* После того, как вы определили причину ошибки, вам нужно исправить соответствующий участок кода.
* После внесения изменений перезапустите игру или сцену, чтобы убедиться, что ошибка больше не возникает, и проверьте консоль на предмет других ошибок.

**5. Управление выводом:**

Консоль Unity также предоставляет возможность фильтрации вывода сообщений. Вы можете фильтровать сообщения по типу (ошибка, предупреждение, сообщение) и по ключевым словам, что упрощает анализ и поиск нужной информации.

**6. Дополнительные средства анализа:**

Помимо самой консоли, Unity предоставляет другие инструменты анализа ошибок, такие как редактор кода, инспектор и отладчик, которые могут быть использованы для более подробного анализа и исправления проблем в коде.

В целом, использование консоли в Unity и анализ скриптовых ошибок является необходимым этапом при разработке игр. Это помогает разработчикам быстро и эффективно обнаруживать и исправлять проблемы в их коде, обеспечивая более стабильную и качественную игровую среду.

***1.3. Пример выполнения задания***

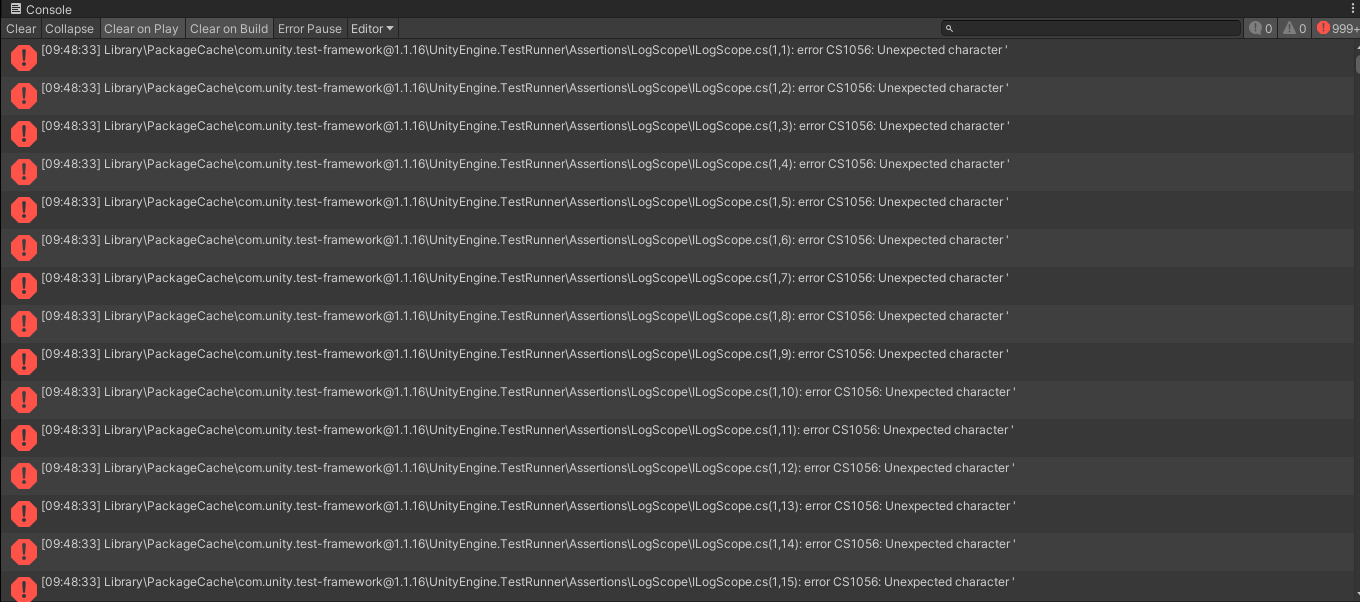


Рис 1.1 Консоль в Unity

**Лабораторная работа №12 Оптимизация проекта**

***Цель работы:*** Изучить методы оптимизации проектов и применить их в существующем проекте.

* 1. ***Общие теоретические сведения***

Оптимизация проекта в Unity является ключевым аспектом разработки игр, особенно при работе с компьютерными играми и мобильными приложениями. Эффективная оптимизация позволяет улучшить производительность, снизить нагрузку на оборудование, повысить качество графики и обеспечить плавный игровой процесс. В этом реферате мы рассмотрим различные методы и приемы оптимизации проекта в Unity.

**1. Оптимизация графики:**

* Уменьшение полигональности моделей: Используйте модели с минимальным количеством полигонов для уменьшения нагрузки на графический движок.
* Оптимизация текстур: Используйте сжатые текстуры с разумным разрешением и форматом, чтобы снизить использование памяти.
* Управление уровнем детализации (LOD): Реализуйте систему уровней детализации для моделей, чтобы автоматически снижать качество отображения при удалении от камеры.
* Оптимизация шейдеров: Используйте простые шейдеры с минимальным числом вычислений для улучшения производительности.

**2. Оптимизация кода:**

* Устранение узких мест: Идентифицируйте и устраните узкие места в коде, которые могут замедлять производительность игры.
* Оптимизация циклов: Пересмотрите циклы и операции внутри них, чтобы уменьшить количество вычислений и итераций.
* Пулы объектов: Используйте пулы объектов для повторного использования ресурсов и снижения нагрузки на сборщик мусора.
* Асинхронные операции: Используйте асинхронные методы и загрузку ресурсов для распределения нагрузки и улучшения отзывчивости приложения.

**3. Оптимизация анимации:**

* Ограничение количества анимаций: Используйте только необходимые анимации и уменьшите количество одновременно проигрываемых анимаций.
* Уменьшение детализации анимации: Сократите количество ключевых кадров и используйте упрощенные версии анимаций для улучшения производительности.

**4. Оптимизация аудио:**

* Сжатие аудиофайлов: Используйте сжатые форматы аудиофайлов с минимальным битрейтом для уменьшения использования памяти и нагрузки на процессор.
* Предзагрузка звуков: Загружайте аудиофайлы заранее, чтобы избежать задержек во время их проигрывания.

**5. Оптимизация управления памятью:**

* Минимизация использования памяти: Используйте механизмы сборки мусора, чтобы минимизировать утечки памяти и освободить неиспользуемые ресурсы.
* Память в реальном времени: Используйте объекты с низкой стоимостью выделения