ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«МАГНИТОГОРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. Г.И. НОСОВА»

(ФГБОУ ВО «МГТУ ИМ. Г.И.НОСОВА»)

Институт энергетики и автоматизированных систем

Кафедра бизнес – информатики и информационных технологий

ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ

по дисциплине: Тестирование информационных систем

на тему: Тестирование информационной системы «Игровой движок с использованием графического API для 2D- и 3D- графики в Vulkan»

Исполнитель: Егоров Михаил Игоревич, студент 4 курса,   
группа АПИб-21-22   
Руководитель: Масленникова Ольга Евгеньевна, канд. пед. наук, доцент кафедры БИиИТ

Магнитогорск, 2024

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_bookmark0)

1. [План Тестирования 4](#_bookmark1)
2. [Создание шаблонов тест-кейсов для тестирования 7](#_bookmark2)
3. [Создание шаблона отчёта по тестированию 10](#_bookmark3)

[ЗАКЛЮЧЕНИЕ 13](#_bookmark4)

[СПИСОК ИСТОЧНИКОВ 14](#_bookmark5)

# ВВЕДЕНИЕ

Разработка игрового движка — это процесс создания программного обеспечения, предназначенного для реализации видеоигр и интерактивных приложений, обеспечивающего необходимый функционал, производительность, графические возможности и удобство использования разработчиками. Игровой движок включает в себя подсистемы для обработки графики, физики, звука, управления ресурсами и взаимодействия с пользователем, а также инструменты для создания игровых уровней и сценариев.

В рамках данного проекта объектом тестирования выступает разрабатываемый игровой движок с использованием графического API Vulkan. Vulkan представляет собой низкоуровневый интерфейс программирования, позволяющий напрямую взаимодействовать с ресурсами графического процессора (GPU). Это обеспечивает эффективное использование аппаратных возможностей и достижение высокой производительности в играх и приложениях, созданных на основе данного движка.

Целью тестирования игрового движка, разработанного с применением API Vulkan, является выявление возможных ошибок в его работе, проверка корректности основных функций, таких как обработка и отображение графики, управление ресурсами, стабильность работы при высокой нагрузке и в различных сценариях использования. Также целью тестирования является оценка удобства использования движка с точки зрения разработчиков игр и приложений.

Для реализации цели необходимо выполнить задачи, связанных с подготовкой тестирования:

1. составить план тестирования;
2. создать шаблоны тест-кейсов для тестирования;
3. создать шаблон отчёта по тестированию.

Выполнение указанных задач позволит создать основу для качественного тестирования: тест-кейсы и отчёт, которые будут использоваться в процессе проверки работы движка. С помощью созданных шаблонов появится возможность выявить основные ошибки и дефекты, что поможет улучшить качество и производительность игрового движка, повысить его надёжность и востребованность среди разработчиков и конечных пользователей.

## План Тестирования

Для организации процесса тестирования разрабатываемого игрового движка необходимо разработать детальный план тестирования. Этот документ станет руководством для специалистов по тестированию и будет содержать цели, области тестирования, выбранные стратегии, критерии оценки качества, а также необходимые ресурсы и сроки выполнения работ.

Цель тестирования игрового движка — обеспечить качество и корректность работы основных функций и подсистем, предоставляемых разработчикам игр и приложений. Важно проверить стабильность движка в различных сценариях эксплуатации и нагрузочных условиях.

Далее, чтобы определить конкретные аспекты игрового движка, которые необходимо проверить, выделим основные области, подлежащие тестированию:

1. Функциональное тестирование:

* Проверка корректности работы подсистемы рендеринга и отображения графики;
* Тестирование работы системы управления игровыми ресурсами;
* Проверка работы физического движка и системы обработки столкновений;
* Тестирование аудио-подсистемы и воспроизведения звука.

1. Тестирование интерфейса:

* Удобство и интуитивность интерфейса для разработчиков;

1. Корректность отображения элементов интерфейса на различных разрешениях и устройствах.
2. Тестирование производительности:

* Проверка стабильности работы движка при высоких нагрузках;
* Тестирование скорости обработки графических и игровых задач;
* Оценка эффективности использования аппаратных ресурсов.

1. Тестирование безопасности:

* Проверка защиты данных и ресурсов движка от несанкционированного доступа;
* Оценка устойчивости движка к сбоям и критическим ошибкам.

После определения областей тестирования, для ограничения объёма работы и концентрации ресурсов на приоритетных задачах, необходимо выделить аспекты движка, которые не будут тестироваться:

1. Тестирование совместимости с операционными системами, кроме целевых платформ;
2. Интеграция с внешними системами и сервисами, не предусмотренными в текущей версии.

Для определения способа проверки движка необходимо выбрать стратегию и подходы тестирования. Это позволит установить приоритеты задач. В данном случае для проверки функциональности приложения будет использоваться ручное тестирование с применением тест-кейсов и чек-листов. Основное внимание будет уделено функциональному тестированию ключевых сценариев работы, таких как инициализация систем и модулей движка, сценариев их использования и т.п.

Далее необходимо определить критерии тестирования движка, чтобы четко определить, когда тестирование можно считать завершённым:

1. Приемочные критерии:

* Успешное прохождение 100% тест-кейсов критического пути.
* Устранение всех критических и высокоприоритетных дефектов.

1. Критерии начала тестирования:

* Доступ к стабильной версии движка, готовой для тестирования.

1. Критерии завершения тестирования:

* Выполнение более 90% запланированных тест-кейсов.

Для обеспечения необходимых условий для успешного тестирования движка необходимо определить ресурсы тестирования:

1. Персонал - разработчик и тестировщик для выполнения тестирования и исправления выявленных дефектов.
2. Оборудование - персональные компьютеры с операционной системой Windows 10 или Linux.
3. Программное обеспечение - все необходимые инструменты и программы для выполнения тестирования, включая ПО для анализа данных, инструменты для записи ошибок и тестовые платформы.

Организация процесса тестирования и установление четких сроков для выполнения этапов требуют тщательного планирования работы и времени. Это позволит эффективно распределить ресурсы и обеспечить успешное завершение тестирования. Для этого необходимо составить расписание тестирования (таблица 1).

Таблица 1 – Расписание тестирования

|  |  |
| --- | --- |
| Этап | Даты |
| Формирование требований | 01.11.2024 – 30.11.2024 |
| Разработка тест кейсов | 01.12.2024 – 30.12.2024 |
| Тестирование и написание отчетов о дефектах | 01.01.2025 – 30.05.2024 |
| Завершение тестирования | 01.06.2025 |

Для более эффективной коммуникации и координации между участниками команды необходимо распределение ролей и ответственности. В нашем случае главными ролями будут выступать разработчик и тестировщик. Разработчик будет являться ответственным за исправление ошибок и доработку функционала. Тестировщик же будет сфокусирован на выполнении тестов по созданным тест-кейсам, а также за заполнение отчётов по тестированию.

При работе с любым продуктом всегда есть риски. Основные риски рассмотрены в таблице 2.

Таблица 2 – Оценка рисков

|  |  |  |  |
| --- | --- | --- | --- |
| Риск | Вероятность | Воздействие | Меры по устранению |
| Недостаточное время тестирования | Средняя | Высокое | Уделить внимание главным функциям |
| Недостаточное покрытие тест-кейсов | Средняя | Высокое | Применить методы автоматического тестирования |

Главной документацией, которая фигурирует в тестировании – тест-кейсы, которые будут являться основой для тестирования, а также отчёт о проведённом тестировании, который будет отражать в себе результаты, основанные на тестировании по подготовленной основе.

Для объективной оценки процесса и результата тестирования, необходимо отобрать нужные метрики, которые позволят измерить эффективность тестирования. В нашем случае нужны следующие метрики:

1. Успешное прохождение тест-кейсов вычисляется по формуле 1:

где – процентный показатель успешного прохождения тест-кейсов,

– количество успешно выполненных тест-кейсов,

– общее количество выпоненных тест-кейсов.

1. Общее устранение дефектов можно получить по формуле 2:

где – процентный показатель устранения дефектов уровня важности Level за время существования проекта,

– количество устраненных за время существования проекта дефактор уровня важности Level,

– количество обнаруженных за время существования проекта дефектов уровня важности Level.

1. Выполнение тест-кейсов вычисляется по формуле 3:

где – процентный показатель выполнения тест-кейсов,

– количество выполненных тест-кейсов,

– количество тест-кейсов, запланированных к выполнению.

1. Покрытие требований тест-кейсами по формуле 4:

где – процентный показатель покрытия требования тест-кейсами,

– количество покрытых тест-кейсами требований,

– общее количество требований.

Таким образом, был составлен подробный план тестирования игрового движка. Данный план включает цель, области тестирования, стратегию, критерии и необходимые ресурсы, что способствует эффективной организации процесса тестирования, своевременному выявлению ошибок и дефектов, а также обеспечению высокого качества конечного продукта.

## Создание шаблонов тест-кейсов для тестирования

Тест-кейсы играют ключевую роль в тестировании информационных систем. Это документ, который содержит описание условий, шагов проверки функционала системы и ожидаемых результатов. Их использование позволяет организовать процесс тестирования, повысить качество продукта и выявить ошибки на этапах разработки.

Изначально составим таблицу с общей информацией о тестировании (таблица Х)

Таблица 3 – Общая информация и тестировании

|  |  |
| --- | --- |
| **Название проекта** | Игровой движок «Lampy Engine» |
| **Номер версии** | 0.0.1a |
| **Имя тестера** | Егоров Михаил |
| **Даты тестирования** | 20.12.2024 |

После составим тест-кейсы для тестирования функциональности приложения (таблицы 4 – 15)

Таблица 4 – Test case TC\_GE\_Launch

|  |  |
| --- | --- |
| **Test Case #** | TC\_GE\_Launch |
| **Приоритет теста** | Высокий |
| **Название тестирования/Имя** | Проверка запуска игрового движка |
| **Резюме испытания** | Проверить успешный запуск приложения игрового движка (.exe) и отображение интерфейса без каких-либо ошибок или предупреждений. |
| **Шаги тестирования** | 1. Двойным кликом (или через терминал для Linux) запустить исполняемый файл engine\_launcher. 2. Дождаться загрузки движка. 3. После окончания загрузки убедиться, что отображается начальный интерфейс движка (главное меню). |
| **Данные тестирования** | * Исполняемый файл engine\_launcher (.exe или бинарный файл для Linux) * Директория с файлами движка |
| **Ожидаемый результат** | * На экране не должно появляться каких-либо ошибок, предупреждений или сообщений о некорректной работе * Главное меню движка отображается. |
| **Фактический результат** | * Сообщения об ошибках или предупреждениях отсутствуют. * Главное меню и интерфейс движка отображаются. |
| **Предпосылки** | * Персональный компьютер с установленной операционной системой (Windows 10/11 или Linux Ubuntu 20.04 и выше) * Установленная последняя версия драйверов GPU с поддержкой API Vulkan или OpenGL 4.0 * Наличие всех необходимых файлов движка в директории программы |
| **Постусловия** | Приложение движка находится в стабильном состоянии и готово к дальнейшему использованию или тестированию. |
| **Статус (Pass/Fail)** | Pass |
| **Комментарии** | Тестирование проводилось на системах с операционными системами Linux (Ubuntu 20.04) и Windows 10. В обоих случаях ошибки или проблемы не были выявлены. |

Таблица 5 – Test case TC\_GE\_Rendering

|  |  |
| --- | --- |
| **Test Case #** | TC\_GE\_Rendering |
| **Приоритет тестирования (Малый/Средний/высокий)** | Высокий |
| **Название тестирования/Имя** | Проверка корректности работы рендеринга |
| **Резюме испытания** | Проверка корректного отображения трёхмерной сцены с использованием графического API Vulkan. Объекты должны отображаться без визуальных дефектов, текстуры и освещение должны быть корректными. |
| **Шаги тестирования** | 1. Запустить движок, используя исполняемый файл engine\_launcher. 2. В открывшемся интерфейсе движка загрузить заранее подготовленную тестовую сцену с 3D-объектами. 3. Дождаться завершения процесса загрузки сцены. 4. Визуально проверить отображение всех 3D-объектов на сцене:  * Оценить корректность позиционирования объектов. * Проверить наложение текстур на объекты. * Проверить работу источников освещения и корректность отображения теней. * Внимательно изучить сцену на предмет визуальных дефектов (артефакты, мерцания, некорректное отображение). |
| **Данные тестирования** | * Тестовая сцена: 3D-модели, текстуры высокого разрешения, источники света. * Исполняемый файл движка. |
| **Ожидаемый результат** | * Все объекты сцены должны отображаться, соответствовать заранее заданному расположению. * Текстуры должны быть наложены корректно и без искажений. * Источники освещения работают корректно, тени и отражения отображаются реалистично. * Визуальные дефекты, артефакты, мерцания и ошибки рендеринга должны отсутствовать. |
| **Фактический результат** | * Сцена была загружена успешно и полностью отображена. * Позиционирование объектов, наложение текстур и отображение освещения соответствуют ожиданиям. * Визуальные дефекты отсутствуют. |
| **Предпосылки** | * ПК с установленной стабильной версией движка. * GPU с полной поддержкой Vulkan API и актуальными драйверами. * Подготовленная тестовая сцена, содержащая набор различных 3D-объектов, текстур и источников освещения. |
| **Постусловия** | Отображение всех элементов сцены |
| **Статус (Pass/Fail)** | Pass |
| **Комментарии** | Визуальная проверка проводилась на нескольких конфигурациях оборудования для подтверждения стабильности рендеринга на разных графических картах. |

Таблица 6 – Test case TC\_GE\_Interface

|  |  |
| --- | --- |
| **Test Case #** | TC\_GE\_Interface |
| **Приоритет тестирования (Малый/Средний/высокий)** | Средний |
| **Название тестирования/Имя** | Проверка интерфейса движка |
| **Резюме испытания** | Проверка удобства использования и корректности отображения элементов пользовательского интерфейса игрового движка при изменении разрешений экрана. |
| **Шаги тестирования** | 1. Проверить доступность и функциональность основных элементов интерфейса (панели инструментов, окна ресурсов, инспекторы объектов). 2. Перейти в настройки движка (меню настроек). 3. Изменить разрешение экрана на минимальное поддерживаемое движком. 4. Проверить корректность и удобство отображения интерфейса. 5. Изменить разрешение экрана на максимальное поддерживаемое движком. 6. Снова проверить корректность и удобство отображения интерфейса. 7. Повторить проверку удобства элементов интерфейса (панели, окна, инспекторы) при обоих разрешениях. |
| **Данные тестирования** | * Запущенный игровой движок (режим редактора). |
| **Ожидаемый результат** | * Интерфейс движка должен корректно адаптироваться к разным разрешениям экрана. * Все элементы интерфейса должны оставаться доступными и функциональными. * Не должно быть искажений, наложений или исчезновения элементов интерфейса. |
| **Фактический результат** | * Интерфейс корректно отображается и стабильно работает на всех протестированных разрешениях. * Элементы интерфейса легко доступны и не перекрывают друг друга. |
| **Предпосылки** | * Что и для предыдущих тестов |
| **Постусловия** | Интерфейс редактора движка функционирует стабильно и готов к дальнейшей работе. |
| **Статус (Pass/Fail)** | Pass |
| **Комментарии** | Проверка проводилась на нескольких распространённых разрешениях экрана (от 1280х720 до 3840х2160). Дефектов интерфейса не выявлено. |

Таблица 7 – Test case TC\_GE\_Performance

|  |  |
| --- | --- |
| **Test Case #** | TC\_GE\_Performance |
| **Приоритет тестирования (Малый/Средний/высокий)** | Высокий |
| **Название тестирования/Имя** | Проверка производительности движка |
| **Резюме испытания** | Проверка производительности и стабильности работы движка в режиме редактора и игрового запуска. Показатель FPS (кадры в секунду) должен стабильно превышать минимальный порог (60 FPS). |
| **Шаги тестирования** | 1. Открыть заранее подготовленную нагруженную сцену, содержащую не менее 100 объектов, текстуры высокого разрешения и несколько источников света. 2. Активировать встроенный профилировщик производительности движка или запустить стороннее ПО для мониторинга FPS. 3. Запустить воспроизведение сцены в режиме редактора и зафиксировать среднее значение FPS. 4. Переключиться в игровой режим (play mode), снова зафиксировать среднее значение FPS. 5. Провести измерения FPS в течение не менее 5 минут в обоих режимах. 6. Зафиксировать минимальные и максимальные значения FPS. |
| **Данные тестирования** | * Исполняемый файл движка. * Нагруженная сцена с большим количеством 3D-объектов, текстур и эффектов освещения. * Инструмент профилирования производительности. |
| **Ожидаемый результат** | * Среднее значение FPS во всех режимах работы движка должно стабильно превышать минимально допустимые 60 кадров в секунду. * Не должно наблюдаться резких падений производительности, зависаний или проседаний кадров ниже заданного уровня в 60 кадров в секунду. |
| **Фактический результат** | * Средний FPS стабильно превышает 60 кадров в секунду как в режиме редактора, так и в игровом режиме. * Минимальные значения не падали ниже 60 кадров, максимальные значения стабильны. * Производительность движка соответствует заявленным ожиданиям. |
| **Предпосылки** | * Что и для предыдущих тестов * Анализатор производительности (встроенный профилировщик или сторонняя программа, например, MSI Afterburner или FRAPS). |
| **Постусловия** | * Движок демонстрирует стабильную производительность, количество кадров соответствует ожиданиям (выше 60 FPS). |
| **Статус (Pass/Fail)** | Pass |
| **Комментарии** | Измерения производительности проводились на ПК с разными конфигурациями GPU. Показатели FPS стабильно сохранялись выше минимального порога. |

Таблица 8 – Test case TC\_UI\_Neg

|  |  |
| --- | --- |
| **Test Case #** | TC\_UI\_Neg |
| **Приоритет тестирования (Малый/Средний/высокий)** | Средний |
| **Название тестирования/Имя** | Проверка интерфейса при неправильном использовании |
| **Резюме испытания** | Проверка устойчивости интерфейса игрового движка к быстрым и хаотичным действиям пользователя. Интерфейс должен сохранять стабильность и корректное отображение при интенсивном взаимодействии. |
| **Шаги тестирования** | 1. В быстром режиме многократно переключаться между вкладками интерфейса («Ресурсы», «Настройки», «Сцена») не менее 20 раз. 2. Сразу после этого быстро и многократно менять различные настройки графики (качество текстур, разрешение экрана, тени, антиалиасинг и др.) не менее 10 раз. 3. Наблюдать за реакцией и стабильностью работы интерфейса на протяжении выполнения этих действий. |
| **Данные тестирования** | * Запущенный игровой движок в режиме редактора. |
| **Ожидаемый результат** | * Интерфейс игрового движка остаётся стабильным, без зависаний, искажений или других аномалий. * Все элементы интерфейса сохраняют свою функциональность и корректно отображают изменения параметров. * Движок не должен зависнуть или аварийно завершить работу в процессе теста. |
| **Фактический результат** | * Интерфейс движка сохранил стабильность во время выполнения всех действий. * Отображение элементов интерфейса корректно, зависаний или визуальных дефектов не обнаружено. * Движок продолжил стабильно функционировать после выполнения хаотичных действий пользователя. |
| **Предпосылки** | * Что и для предыдущих тестов |
| **Постусловия** | * Интерфейс и сцена движка стабильно функционируют, сохраняя готовность к дальнейшей работе. |
| **Статус (Pass/Fail)** | Pass |
| **Комментарии** | Тестирование проводилось на нескольких ПК с разными аппаратными конфигурациями для подтверждения общей стабильности интерфейса. Во всех случаях интерфейс продемонстрировал высокую устойчивость. |

Таблица 9 – Test case TC\_PHYS\_Neg

|  |  |
| --- | --- |
| **Test Case #** | TC\_PHYS\_Neg |
| **Приоритет тестирования (Малый/Средний/высокий)** | Высокий |
| **Название тестирования/Имя** | Проверка работы физического движка при экстремальных значениях параметров объектов (массы, скорости). Объекты должны корректно взаимодействовать и вести себя стабильно, без проваливаний сквозь поверхности и зависаний. |
| **Резюме испытания** | Проверка работы симуляции физики при экстремальных параметрах. |
| **Шаги тестирования** | 1. Открыть тестовую сцену, специально подготовленную для тестирования физики. 2. Выбрать объект сцены и установить экстремально высокую массу (например, 10^6 кг). 3. Выбрать другой объект и установить экстремально высокую скорость (например, 1000 м/с). 4. Запустить симуляцию физики сцены и наблюдать за поведением объектов. 5. Проверить реакцию объектов на столкновение, взаимодействие с поверхностями и другими объектами сцены. 6. Отметить любые неточности, прохождения сквозь поверхности, зависания или аномалии поведения объектов. |
| **Данные тестирования** | * Исполняемый файл движка. * Тестовая сцена с возможностью изменения параметров объектов. |
| **Ожидаемый результат** | * Объекты ведут себя стабильно и корректно. * Отсутствует прохождение объектов сквозь поверхности или друг друга. * Отсутствуют зависания или критические ошибки в работе физического движка. |
| **Фактический результат** | * Физический движок стабильно обработал экстремальные параметры. * Объекты корректно взаимодействовали с поверхностями и другими объектами сцены. * Ошибки физического взаимодействия (проваливание, зависание) отсутствуют. |
| **Предпосылки** | * Что и для предыдущих тестов |
| **Постусловия** | * Физическая модель мира корректно отрабатывает заданные экстремальные условия без дефектов и ошибок. |
| **Статус (Pass/Fail)** | Pass |
| **Комментарии** | Тестирование подтвердило стабильную работу физического движка при экстремальных нагрузках. Тесты проводились неоднократно с различными комбинациями экстремальных параметров. |

Таблица 10 – Test case TC\_Res\_Load

|  |  |
| --- | --- |
| **Test Case #** | TC\_Res\_Load |
| **Приоритет тестирования** | Высокий |
| **Название тестирования/Имя** | Проверка загрузки игровых ресурсов |
| **Резюме испытания** | Проверка корректности работы системы ресурсов игрового движка, включая горячую перезагрузку (hot reload) текстур, моделей, шейдеров и аудиофайлов. |
| **Шаги тестирования** | 1. Создать новую сцену и убедиться, что она открыта и работает корректно. 2. Загрузить в движок набор тестовых ресурсов: 3. Текстуры различных форматов и размеров. 4. 3D-модели с разным уровнем детализации. 5. Аудиофайлы разного формата и продолжительности. 6. Несколько вариантов шейдеров. 7. Вставить ресурсы в сцену и убедиться, что каждый тип ресурса отображается или воспроизводится корректно. 8. Изменить ресурсы внешним образом (например, редактировать текстуру и перезаписать её файл на диске). 9. Проверить работу системы «горячей» перезагрузки, убедиться, что ресурсы обновляются в движке автоматически без необходимости его перезапуска. 10. Оценить время и корректность перезагрузки каждого ресурса. |
| **Данные тестирования** | * Примеры текстур (.png, .jpg). * 3D-модели (.obj, .fbx). * Аудиофайлы (.mp3, .wav). * Шейдеры (.glsl, .spv). |
| **Ожидаемый результат** | * Все ресурсы загружаются и отображаются корректно без каких-либо искажений и ошибок. * Горячая перезагрузка работает стабильно и без задержек. Изменения внешних ресурсов отображаются на сцене немедленно после сохранения изменений. |
| **Фактический результат** | * Все ресурсы загружены и корректно отображены в сцене. * Горячая перезагрузка ресурсов функционирует правильно и без задержек. * Ошибок и искажений не выявлено. |
| **Предпосылки** | * Что и для предыдущих тестов * Набор внешних ресурсов (3D-модели, текстуры, аудиофайлы, шейдеры), доступных для загрузки и замены во время работы движка. |
| **Постусловия** | * Система управления ресурсами движка корректно функционирует. * Горячая перезагрузка работает стабильно и соответствует ожиданиям. |
| **Статус (Pass/Fail)** | Pass |
| **Комментарии** | Тестирование проводилось с использованием нескольких наборов ресурсов разного формата и размера. Проблем или сбоев обнаружено не было. |

Таблица 11 – Test case TC\_Shader\_Correctness

|  |  |
| --- | --- |
| **Test Case #** | TC\_Shader\_Correctness |
| **Приоритет тестирования** | Высокий |
| **Название тестирования/Имя** | Проверка работы шейдеров на сцене |
| **Резюме испытания** | Проверить, что движок правильно отображает все визуальные эффекты, связанные с использованием шейдеров (освещение, тени, отражения) в сцене. |
| **Шаги тестирования** | 1. В интерфейсе движка открыть подготовленную тестовую сцену с заранее установленными шейдерами. 2. Запустить режим отображения сцены и убедиться, что все объекты и шейдеры корректно подгружены. 3. Визуально проверить отображение освещения на объектах сцены. 4. Проверить отображение теней от различных источников освещения. 5. Проверить корректность работы шейдеров отражения на поверхностях объектов. 6. Провести тщательную проверку на отсутствие графических артефактов (мерцания, «рваных» текстур, некорректных бликов или теней). |
| **Данные тестирования** | * Исполняемый файл движка. * Тестовая сцена с шейдерами (GLSL/SPIR-V), включающая:  1. Объекты с отражающими поверхностями. 2. Источники освещения (точечные, направленные, и рассеянные). 3. Шейдеры для обработки теней и отражений. |
| **Ожидаемый результат** | * Все объекты сцены корректно отображают шейдеры. * Освещение соответствует заданным параметрам, тени выглядят естественно и правильно расположены. * Шейдеры отражения работают корректно, отражения выглядят реалистично. * Визуальные дефекты и артефакты отсутствуют. |
| **Фактический результат** | * Шейдеры отображаются и функционируют корректно. * Освещение сцены, тени и отражения отрабатывают в соответствии с заданными параметрами. * Визуальных дефектов, мерцаний, артефактов или сбоев в работе шейдеров не зафиксировано. |
| **Предпосылки** | * Что и для предыдущих тестов * Подготовлена тестовая сцена с набором объектов и шейдеров (освещение, тени, отражения). |
| **Постусловия** | * Все шейдеры корректно применяются и отображаются на объектах сцены. * Движок продолжает стабильно функционировать и готов к следующим этапам тестирования. |
| **Статус** | Pass |
| **Комментарии** | Тестирование проводилось на нескольких различных GPU от AMD и NVIDIA с актуальными версиями драйверов для Vulkan API. Во всех случаях результат оказался положительным. |

Таблица 12 – Test case TC\_Audio\_Playback

|  |  |
| --- | --- |
| **Test Case #** | TC\_Audio\_Playback |
| **Приоритет тестирования (Малый/Средний/высокий)** | Низкий |
| **Название тестирования/Имя** | Проверка воспроизведения звука |
| **Резюме испытания** | Проверка корректности работы аудиоподсистемы движка при воспроизведении звуковых файлов в режиме реального времени. |
| **Шаги тестирования** | 1. Открыть заранее подготовленную тестовую сцену с аудиоисточниками. 2. Запустить воспроизведение сцены. 3. Проверить воспроизведение каждого звукового эффекта и фоновой музыки: 4. Запустить звуки через соответствующие действия на сцене. 5. Внимательно прослушать каждый звуковой эффект до конца. 6. Убедиться, что звуки проигрываются асинхронно, без наложений, искажений или пропуска частей аудио. 7. Повторить воспроизведение для звуков разной длительности и форматов. 8. Проверить корректность остановки и повторного запуска звуков. |
| **Данные тестирования** | * Тестовая сцена с несколькими аудиоисточниками (фоновые звуки, эффекты взаимодействия). * Аудиофайлы форматов .wav и .mp3 разной длительности (от 1 до 60 секунд). |
| **Ожидаемый результат** | * Все звуки воспроизводятся полностью, без прерываний и искажений. * Отсутствуют задержки или преждевременное завершение воспроизведения звука. |
| **Фактический результат** | * При воспроизведении звуковых эффектов выявлено, что некоторые аудиофайлы не проигрываются полностью и обрываются за 2-3 секунды до завершения. * Проблема повторяется с файлами формата .mp3 длительностью более 30 секунд. * Короткие файлы (.wav) воспроизводятся без проблем. |
| **Предпосылки** | * Что и для предыдущих тестов * Установленные актуальные аудио-драйверы. * Подготовленные аудиофайлы различных форматов (WAV, MP3, OGG). |
| **Постусловия** | * Дефект в системе воспроизведения аудио нуждается в исправлении и повторном тестировании. |
| **Статус (Pass/Fail)** | Fail |
| **Комментарии** | Проблема воспроизводится только с длительными файлами (.mp3). Рекомендуется дополнительная проверка аудиосистемы и обработчика аудио в движке. |

Таблица 13 – Test case TC\_Memory\_Leak

|  |  |
| --- | --- |
| **Test Case #** | TC\_Memory\_Leak |
| **Приоритет тестирования (Малый/Средний/высокий)** | Средний |
| **Название тестирования/Имя** | Проверка движка на утечки памяти |
| **Резюме испытания** | Проверка стабильности работы движка и отсутствия утечек памяти в процессе длительной непрерывной работы с нагруженной сценой. |
| **Шаги тестирования** | 1. Открыть заранее подготовленную нагруженную сцену. 2. Запустить сцену в режиме воспроизведения (Play mode). 3. Зафиксировать начальный объём используемой памяти в системе (RAM, VRAM). 4. Оставить движок работать непрерывно на протяжении длительного времени (от 1 до 5 часов). 5. Каждые 30 минут проверять текущий объем используемой памяти, фиксируя показатели и отклонения. 6. По завершении теста (через 5 часов) сравнить финальный объём используемой памяти с первоначальным. 7. Завершить работу движка и проверить, полностью ли освободилась память после закрытия программы. |
| **Данные тестирования** | * Запущенная стабильная версия движка. * Нагруженная тестовая сцена (свыше 500 объектов с различными материалами и текстурами). |
| **Ожидаемый результат** | * Использование памяти движком должно стабилизироваться через небольшой промежуток времени после запуска сцены (10–15 минут). * Дальнейший рост объема занимаемой памяти не должен превышать 5% от первоначального значения за весь период тестирования (1–5 часов). * После закрытия движка вся занимаемая память должна быть полностью освобождена системой. |
| **Фактический результат** | * За время тестирования (4 часа) потребление памяти стабилизировалось после первых 10 минут и далее сохранялось практически неизменным (рост менее 2% за весь период наблюдений). * После завершения теста и закрытия движка память освободилась полностью. |
| **Предпосылки** | * Что и для предыдущих тестов * Подготовленная нагруженная сцена с большим количеством объектов, текстур и активных компонентов. |
| **Постусловия** | * Утечек памяти не выявлено. * Движок стабильно работает с памятью, правильно высвобождает ресурсы. |
| **Статус (Pass/Fail)** | Pass |
| **Комментарии** | Тестирование проводилось на системах с различными аппаратными конфигурациями и не выявило никаких утечек или проблем с управлением памятью. |

Таблица 14 – Test case TC\_ TC\_Crash\_Recovery

|  |  |
| --- | --- |
| **Test Case #** | TC\_Crash\_Recovery |
| **Приоритет тестирования (Малый/Средний/высокий)** | Высокий |
| **Название тестирования/Имя** | Проверка восстановления после сбоя движка |
| **Резюме испытания** | Проверка способности игрового движка корректно восстанавливать данные и проектное состояние после аварийного или принудительного завершения работы программы. |
| **Шаги тестирования** | 1. В процессе загрузки сцены (на этапе загрузки ресурсов или инициализации сцены) принудительно завершить работу движка (например, через диспетчер задач или командой принудительного завершения в терминале). 2. Повторно запустить движок. 3. Проверить, запускается ли движок корректно после принудительного завершения. 4. Убедиться в наличии или отсутствии потери проектных данных, ресурсов или настроек. 5. Оценить корректность автоматического восстановления состояния сцены и интерфейса. |
| **Данные тестирования** | * Исполняемый файл движка. * Тестовая сцена с большим набором объектов и параметров (освещение, ресурсы, настройки). |
| **Ожидаемый результат** | * После повторного запуска движок должен автоматически корректно восстановить проектные данные и состояние сцены. * Интерфейс движка должен быть полностью функционален, элементы интерфейса корректно отображаются без потери данных и искажений. * Восстановленные данные должны полностью соответствовать последнему стабильному состоянию до аварийного завершения. |
| **Фактический результат** | * Движок успешно восстановил своё состояние после принудительного завершения работы. * Сцена и проектные данные были корректно восстановлены без каких-либо потерь. * Интерфейс движка отображается корректно, ошибок и искажений не выявлено. |
| **Предпосылки** | * Что и для предыдущих тестов * Автоматическое сохранение и система восстановления данных движка активированы. |
| **Постусловия** | * Движок продолжает стабильно функционировать после восстановления. * Потери проектных данных отсутствуют. |
| **Статус (Pass/Fail)** | Pass |
| **Комментарии** | Проверка была проведена несколько раз с разными точками аварийного завершения. Во всех случаях движок корректно восстанавливал состояние без потерь данных. |

Таблица 15 – Test case TC\_UI\_Scaling

|  |  |
| --- | --- |
| **Test Case #** | TC\_UI\_Scaling |
| **Приоритет тестирования** | Средний |
| **Название тестирования/Имя** | Проверка масштабируемости интерфейса |
| **Резюме испытания** | Проверка корректности и удобства отображения интерфейса редактора движка при изменении разрешения и масштабирования элементов. |
| **Шаги тестирования** | 1. Перейти в меню настроек графики и изменить разрешение экрана на минимальное доступное (например, 1280x720). 2. Проверить корректность и удобство отображения всех внутренних элементов интерфейса редактора (панели, инспекторы, меню и инструменты). 3. Изменить разрешение на максимальное доступное для монитора (например, 3840x2160). 4. Повторно проверить отображение всех элементов интерфейса, убедиться, что они корректно масштабировались и не имеют искажений или наложений. 5. Дополнительно проверить возможность ручного изменения размера панелей интерфейса и их реакцию на масштабирование. 6. Проверить, что текст интерфейса читаем и понятен при любом из протестированных разрешений и масштабов. |
| **Данные тестирования** | * Запущенный интерфейс редактора движка. |
| **Ожидаемый результат** | * Интерфейс редактора должен корректно адаптироваться под любое установленное разрешение экрана. * Все элементы интерфейса должны оставаться функциональными, отображаться чётко и без искажений. * Не должно возникать наложений, сдвигов или перекрытия элементов интерфейса. |
| **Фактический результат** | * Интерфейс движка масштабируется корректно при минимальном и максимальном разрешениях. * Элементы интерфейса адаптируются без искажений, все панели и окна остаются функциональными. * Текст и элементы интерфейса читаемы и понятны. |
| **Предпосылки** | * Что и для предыдущих тестов |
| **Постусловия** | * Интерфейс движка стабильно функционирует, сохраняя удобство при работе на разных разрешениях экрана. |
| **Статус** | Pass |
| **Комментарии** | Тестирование проводилось на нескольких экранах с различным разрешением и соотношением сторон (Full HD, 2K, 4K). В каждом случае интерфейс демонстрировал стабильную работу и масштабируемость без визуальных дефектов. |

Разработаны тест-кейсы, которые позволяют организовать процесс тестирования, определить основные сценарии использования системы и проверить её функциональность. Эти тест-кейсы обеспечивают последовательное выполнение тестов, фиксацию результатов и выявление ошибок, что способствует улучшению качества разрабатываемого продукта.

## Создание шаблона отчёта по тестированию

Отчёт по тестированию является основным документом, отражающим результаты проверки информационной системы. Он необходим для анализа качества продукта, выявления проблем и оценки их влияния на систему. Отчёт помогает разработчикам определить степень готовности системы к эксплуатации, а также спланировать дальнейшие улучшения функциональности, стабильности и безопасности для будущих обновлений.

Первый раздел отчёта включает краткое описание процесса тестирования, содержащее информацию о сроках проведения, проценте выполненных тест-кейсов, количестве зарегистрированных баг-репортов и других ключевых показателях.

Во втором разделе будет представлена информация о команде, непосредственно участвовавшей в тестировании. (таблица 16)

Таблица 16 – Команда

|  |  |  |
| --- | --- | --- |
| Имя | Должность | Роль |
| Егоров Михаил | Разработчик | Ответственный за тест-кейсы |
| Максим Козин | Тестировщик | Проводит тестирование на основе тест-кейсов |

Третий раздел отчёта посвящён описанию процесса тестирования, который служит для систематизации и документирования всех действий, связанных с проверкой информационной системы. Это описание помогает участникам проекта понять, каким образом были получены результаты, и становится основой для принятия решений о доработке или запуске продукта.

Четвёртый раздел включает расписание тестирования, устанавливающее временные рамки для выполнения основных этапов. Оно позволяет определить сроки ключевых задач, эффективно распределить ресурсы и организовать взаимодействие между членами команды. (таблица 17)

Таблица 17 – Расписание

|  |  |  |  |
| --- | --- | --- | --- |
| Имя | Дата | Деятельность | Продолжительность |
| Егоров Михаил | 1.12 – 15.12 | Составление плана тестирования | 30 |
| Максим Козин | 15.12 – 20.12 | Проведение тестирования | 10 |

Затем следует подготовить статистику выявленных дефектов, которая необходима для оценки качества тестируемой системы и эффективности проведённого тестирования. Кроме того, необходимо составить подробный список обнаруженных дефектов. (таблица 18 – 19)

Таблица 18 – Статистика дефектов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | **Важность** | | |
| **Статус** | **Количество** | **Низкая** | **Средняя** | **Высокая** |
| Найдено | 1 | 1 | 0 | 0 |
| Исправлено | 0 | 0 | 0 | 0 |
| Проверено | 0 | 0 | 0 | 0 |
| Отклонено | 0 | 0 | 0 | 0 |

Таблица 19 – Список дефектов

|  |  |  |
| --- | --- | --- |
| **ID** | **Важность** | **Описание** |
| TC\_Audio\_Playback | Низкая | Звуки не влияют на общее использование движка |

В результате был разработан шаблон отчёта по тестированию, который предоставляет полное представление о процессе и итогах проверки информационной системы. Этот документ служит основой для анализа выполненной работы над продуктом.

# ЗАКЛЮЧЕНИЕ

В рамках индивидуального домашнего задания объектом тестирования выступал разрабатываемый игровой движок с использованием API Vulkan — инструмент, предназначенный для создания высокопроизводительных игр и графических приложений. Движок ориентирован на разработчиков, предоставляя удобные средства управления ресурсами, рендеринга и взаимодействия с трехмерной графикой.

В процессе выполнения ИДЗ были выполнены следующие задачи:

* составлен план тестирования игрового движка;
* созданы шаблоны тест-кейсов для проверки ключевых функций и подсистем движка;
* разработан шаблон отчета по результатам тестирования.

Решение указанных задач сформировало основу для последующего тестирования движка: подготовленные тест-кейсы и отчёты будут использоваться при дальнейших проверках и испытаниях. Применение созданных шаблонов позволит эффективно выявить критические ошибки и дефекты, что существенно повысит качество движка. Это, в свою очередь, увеличит его надёжность и востребованность среди разработчиков игр и конечных пользователей.

# СПИСОК ИСТОЧНИКОВ

1. Аграновский, А. В. Тестирование веб-приложений : учебное пособие / А. В. Аграновский. — Санкт-Петербург : ГУАП, 2020. — 155 с. — ISBN 978-5-8088-1515-5. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/216533 — Режим доступа: для авториз. пользователей.
2. Елизаров, Д. А. Тестирование программных продуктов : учебно- методическое пособие / Д. А. Елизаров. — Омск : ОмГУПС, 2023 — Часть 1 — 2023. — 25 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/419267 — Режим доступа: для авториз. пользователей.
3. Иванова, Г. С. Методы обработки данных и оценки программ : учебное пособие / Г. С. Иванова, Т. Н. Ничушкина, Е. К. Пугачев. — Москва : МГТУ им. Н.Э. Баумана, 2020. — 74 с. — ISBN 978-5-7038-5409-9. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/205670 — Режим доступа: для авториз. пользователей.
4. Дукельский, К. В. Управление качеством программного обеспечения : учебное пособие / К. В. Дукельский, И. Б. Бондаренко. — Санкт-Петербург : СПбГУТ им. М.А. Бонч-Бруевича, 2021. — 52 с. — Текст : электронный // Лань : электронно- библиотечная система. — URL: https://e.lanbook.com/book/279632 — Режим доступа: для авториз. пользователей.
5. Аниче, М. Эффективное тестирование программного обеспечения / М. Аниче

; перевод с английского А. Н. Киселева. — Москва : ДМК Пресс, 2023. — 370 с. — ISBN 978-5-97060-997-2. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/314930 — Режим доступа: для авториз. пользователей.