|  |  |  |
| --- | --- | --- |
|  |  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных технологий

**Отчет по практическим работам №5-8**

по дисциплине «Системная и программная инженерия»

|  |  |
| --- | --- |
| **Выполнили:**  Студенты группыИВБО-01-22 | Горохов М.П. Караваев И.И.  Лобода А.С.  Андреев Д.А. |
| **Проверил:** | Туманова М.Б. |

МОСКВА 2025 г.

СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 2](#_Toc195646661)

[ПРАКТИЧЕСКАЯ РАБОТА №5 4](#_Toc195646662)

[5.1 Задание 4](#_Toc195646663)

[5.2 Выполнение работы 4](#_Toc195646664)

[5.2.1 Построение структурных диаграмм 4](#_Toc195646665)

[5.2.2 Диаграмма классов 4](#_Toc195646666)

[5.2.3 Диаграмма объектов 6](#_Toc195646667)

[5.2.4 Построение процессов проекта в нотации IDEF0 6](#_Toc195646668)

[5.3 Итог 10](#_Toc195646669)

[6.1 Задание 12](#_Toc195646670)

[6.2 Выполнение работы 12](#_Toc195646671)

[6.2.1 Построение диаграммы в нотации DFD 12](#_Toc195646672)

[6.2.2 Словесное описание взаимодействия компонентов системы 14](#_Toc195646673)

[6.2.2.1 Обработка действий игрока 14](#_Toc195646674)

[6.2.2.2 Обработка физики объектов 14](#_Toc195646675)

[6.2.2.3 Обработка состояния игры 15](#_Toc195646676)

[6.3. Логическая модель базы данных 15](#_Toc195646677)

[6.4 Итог 17](#_Toc195646678)

[ПРАКТИЧЕСКАЯ РАБОТА №7 18](#_Toc195646679)

[7.1. Задание 18](#_Toc195646680)

[7.2 Архитектура системы 18](#_Toc195646681)

[7.3 Архитектурная диаграмма 19](#_Toc195646682)

[7.4 Матрица требований 20](#_Toc195646683)

[ПРАКТИЧЕСКАЯ РАБОТА №8 30](#_Toc195646684)

[8.1 Задание на практическую работу 30](#_Toc195646685)

[8.2 Ход выполнения практической работы 30](#_Toc195646686)

[8.2.1 Выбор ГОСТа для разработки технического задания 30](#_Toc195646687)

[8.2.2 Техническое задание по ГОСТу 33](#_Toc195646688)

[8.3 Вывод 38](#_Toc195646689)

[ЗАКЛЮЧЕНИЕ 39](#_Toc195646690)

[СПИСОК ИСТОЧНИКОВ 40](#_Toc195646691)

**ПРАКТИЧЕСКАЯ РАБОТА №5**

**5.1 Задание**

1. Построить структурные диаграммы своего проекта.

1.1. Диаграмма классов.

1.2. Диаграмма объектов.

1. Разобрать процесс работы внутри проекта в нотации по выбору, IDEF0 или BPMN 2.0. Построить собственный процесс в выбранной нотации.

**5.2 Выполнение работы**

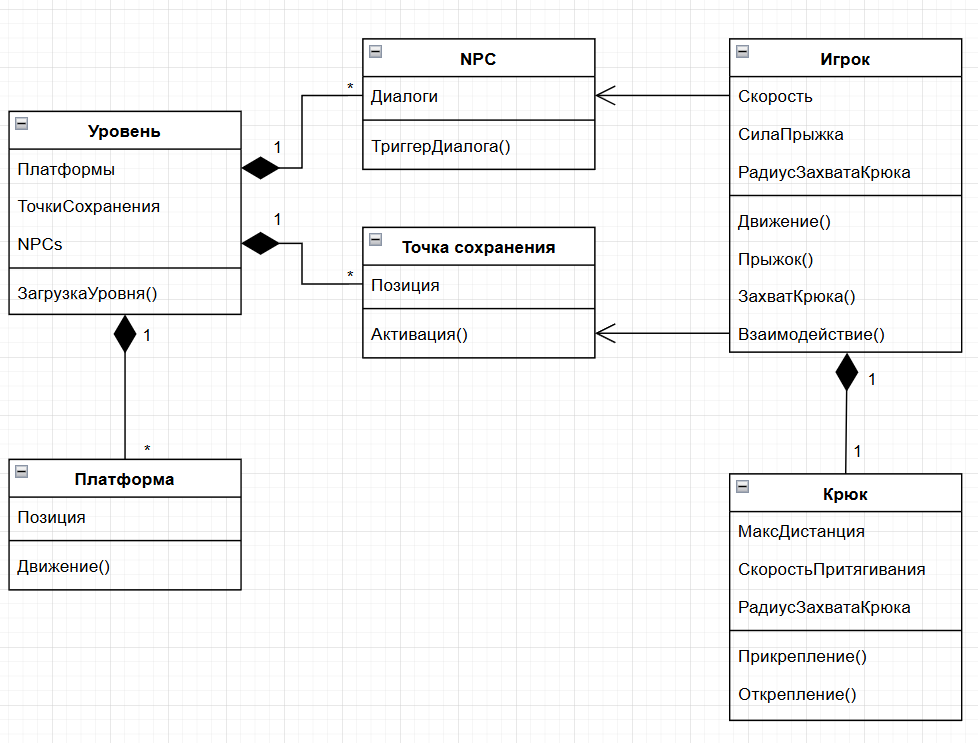
**5.2.1 Построение структурных диаграмм**

Основное назначение структурных диаграмм заключается в графическом представлении состава статистических совокупностей, характеризующихся как соотношение различных частей каждой из совокупностей. Структурная диаграмма – это инструмент модульного дизайна сверху вниз, построенный из квадратов, представляющих различные модули в системе и соединяющие их линии. Линии представляют связь и / или право собственности между видами деятельности и вспомогательными видами деятельности, как они используются в организационных диаграммах.

**5.2.2 Диаграмма классов**

Диаграмма классов – структурная диаграмма языка моделирования UML, демонстрирующая общую структуру иерархии классов системы, их коопераций, атрибутов (полей), методов, интерфейсов и взаимосвязей между ними. Широко применяется не только для документирования и визуализации, но также для конструирования посредством прямого или обратного проектирования.

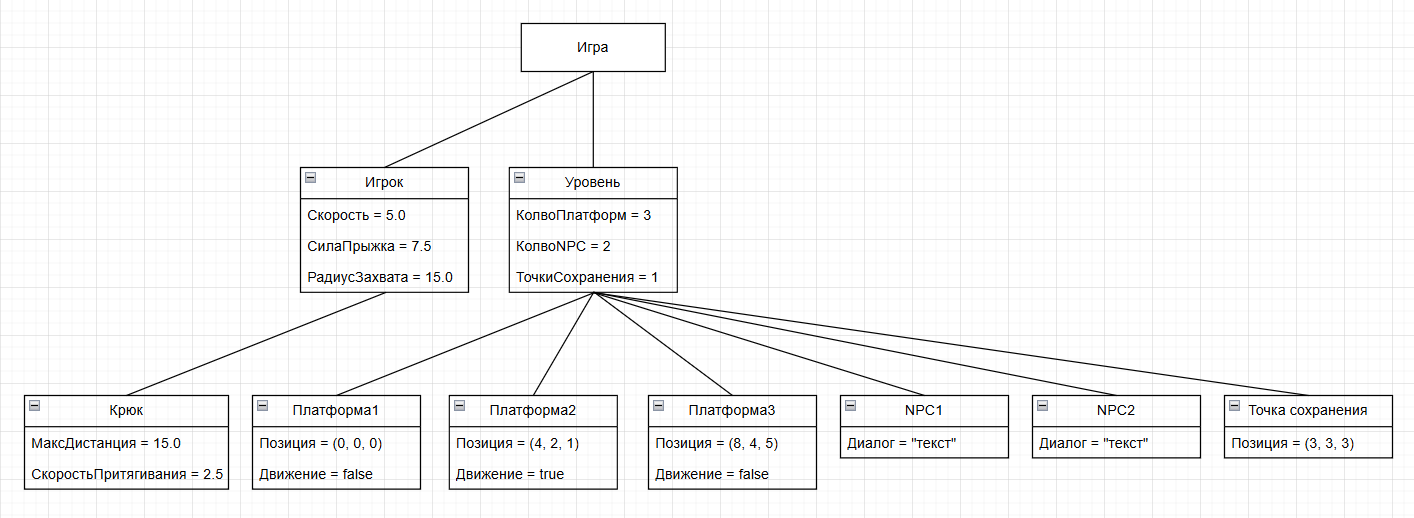
Разработка диаграмм классов и объектов для игры на движке Unreal Engine 5 начинается с анализа требований и выделения ключевых компонентов системы. На этапе проектирования диаграммы классов определяются основные сущности, такие как Игрок, NPC, Точка сохранения, Крюк, Уровень и Платформа. Для каждого класса задаются атрибуты (например, скорость игрока, количество платформ на уровне, диалоги NPC) и методы (например, движение игрока, загрузка уровня, прикрепление крюка). Затем устанавливаются взаимосвязи между классами: ассоциации (игрок взаимодействует с NPC и точкой сохранения), композиции (уровень содержит платформы). Диаграмма классов служит основой для понимания архитектуры игры и разделения системы на логические модули.



**Рисунок 5.1 – Диаграмма классов**

**5.2.3 Диаграмма объектов**

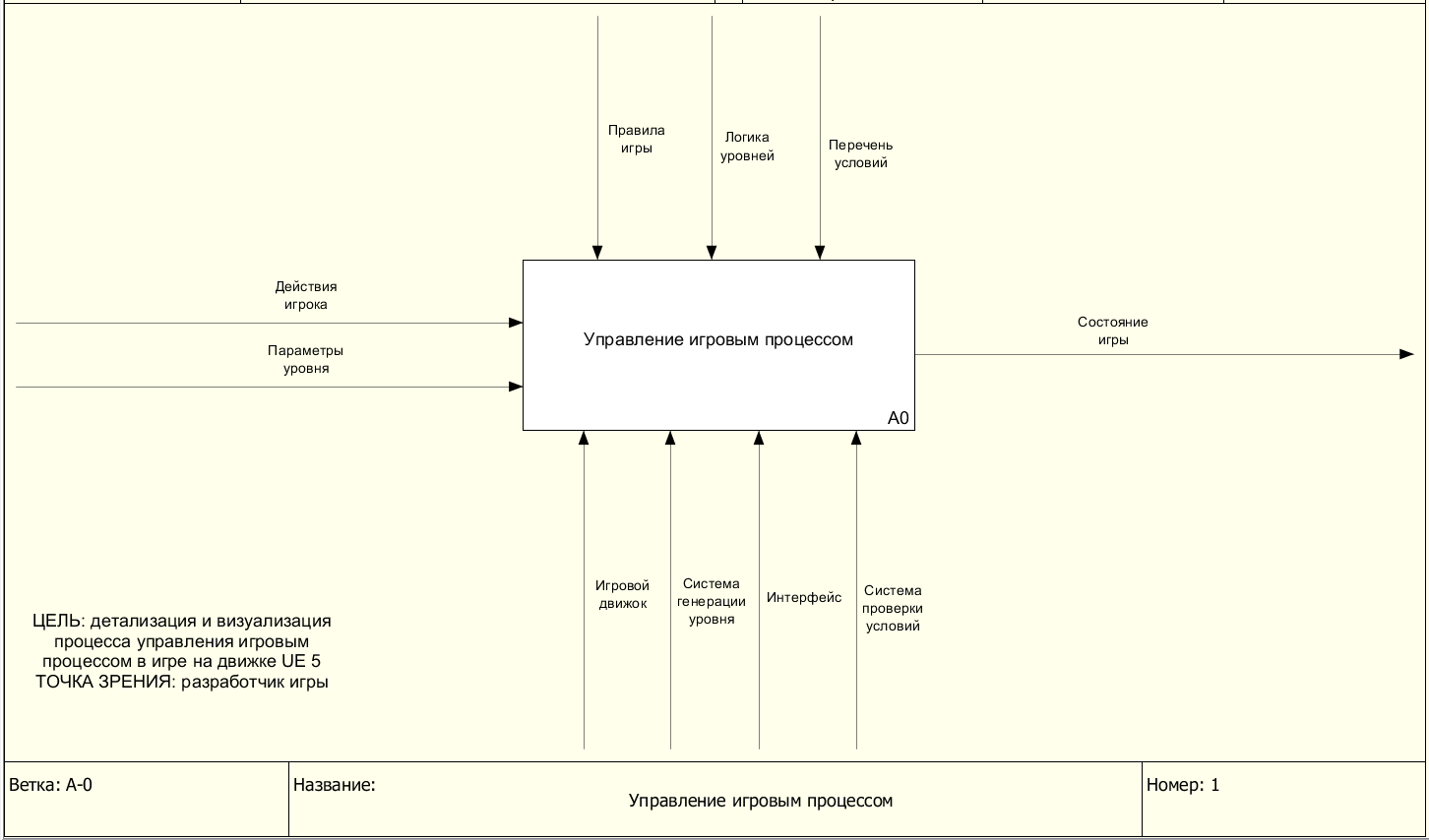
Диаграмма объектов в языке моделирования UML предназначена для демонстрации совокупности моделируемых объектов и связей между ними в фиксированный момент времени. Диаграмма объектов описывает конкретные экземпляры объектов и напрямую соотносится с диаграммой классов, которая дает общее представление о конфигурации системы. Она используется для документирования структур данных и создания статических снимков состояний объектов принимая во внимание реальные экземпляры или прототипы. Динамику поведения объектов обычно изображают в виде последовательности таких диаграмм.



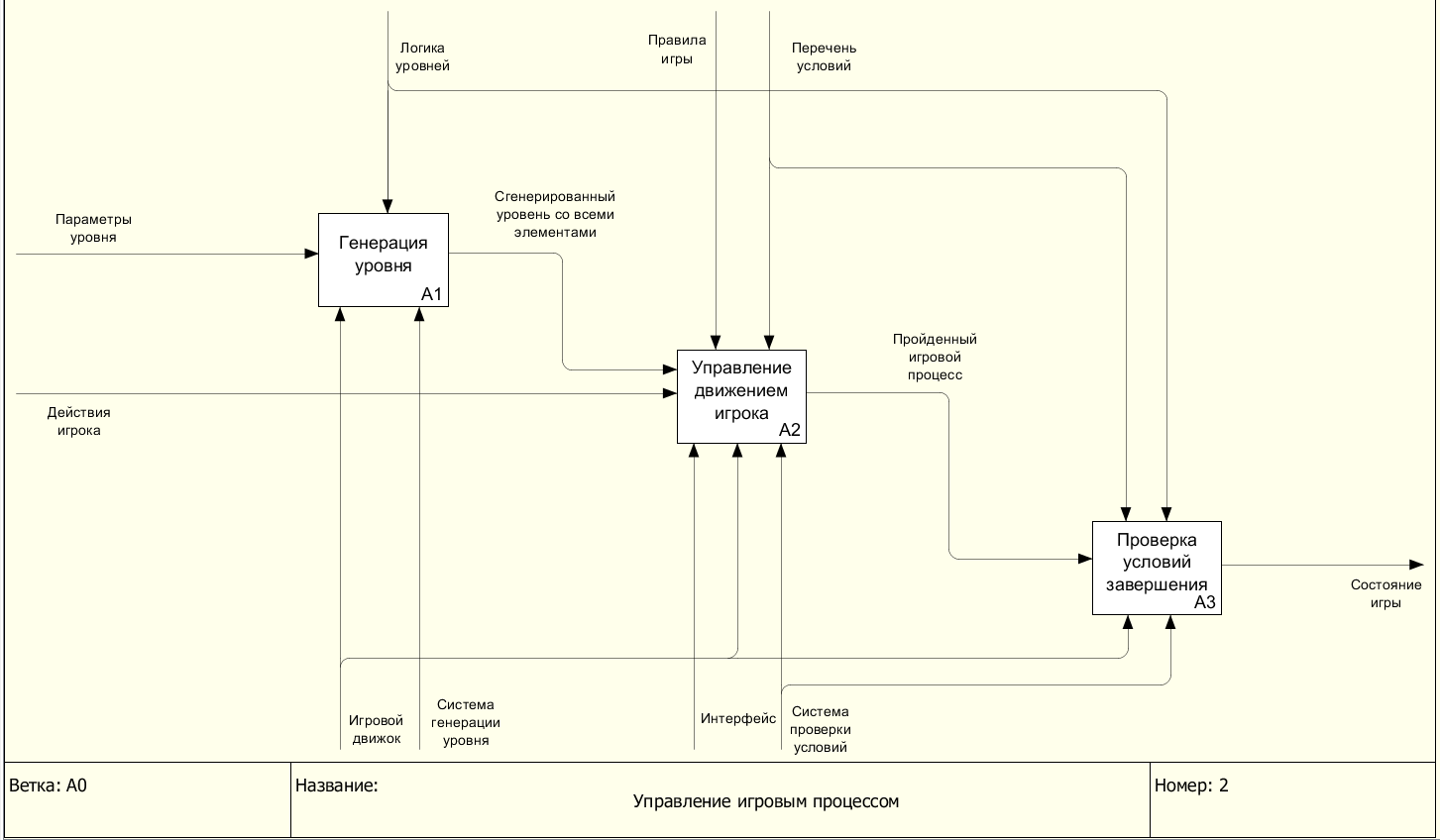
**Рисунок 5.2 – Диаграмма объектов**

**5.2.4 Построение процессов проекта в нотации IDEF0**

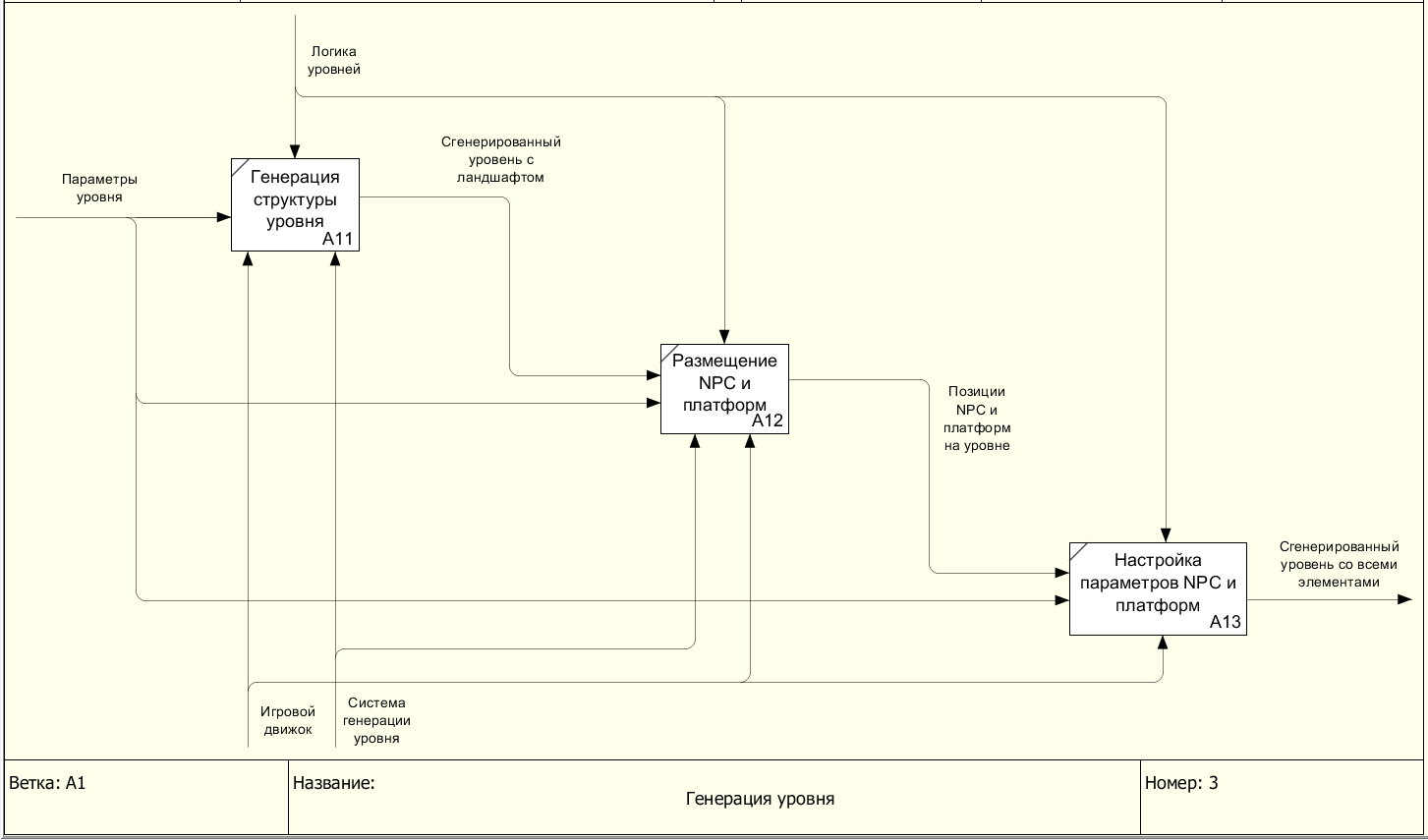
Разработка процесса работы внутри игры на движке Unreal Engine 5 с использованием нотации IDEF0 начинается с описания общего процесса управления игрой на контекстной диаграмме. Основная функция — "Управление игровым процессом". Этот уровень задает общее представление о том, как работает игра, и служит основой для дальнейшей декомпозиции. На следующем этапе процесс разбивается на три основных подпроцесса: "Генерация уровня", "Управление движением игрока" и "Проверка условий завершения игры". Каждый из этих процессов детализируется с помощью декомпозиции. Итоговая иерархия диаграмм IDEF0 включает контекстную диаграмму, диаграммы основных процессов и их подпроцессов. Это позволяет охватить все аспекты работы игры, начиная от генерации уровней и заканчивая завершением игры. Такой подход обеспечивает четкое понимание архитектуры игры и взаимодействия её компонентов, что особенно важно для разработчиков, тестировщиков и других участников проекта.



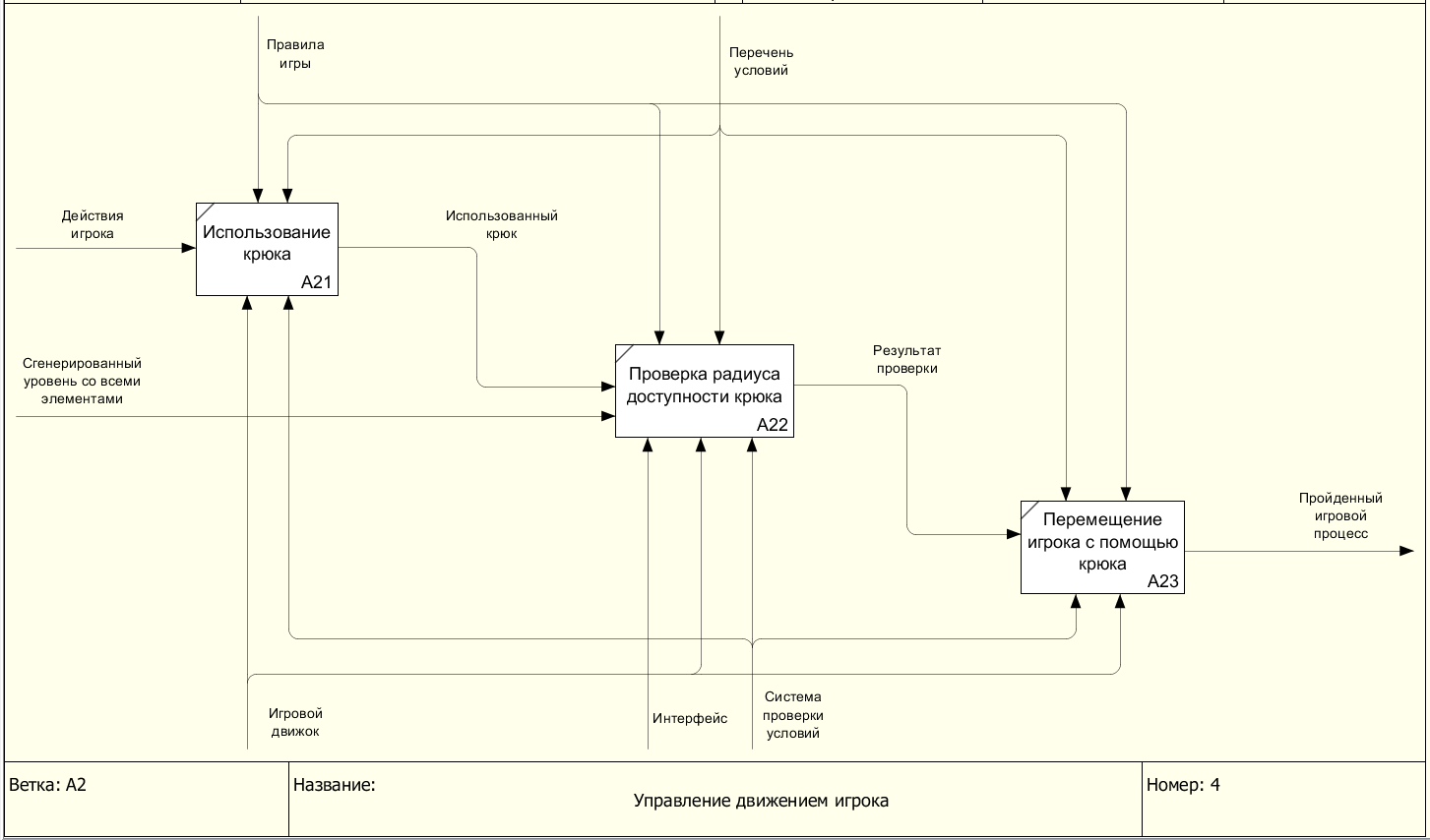
**Рисунок 5.3 – Процесс «Управление игровым процессом» от лица разработчика игры**



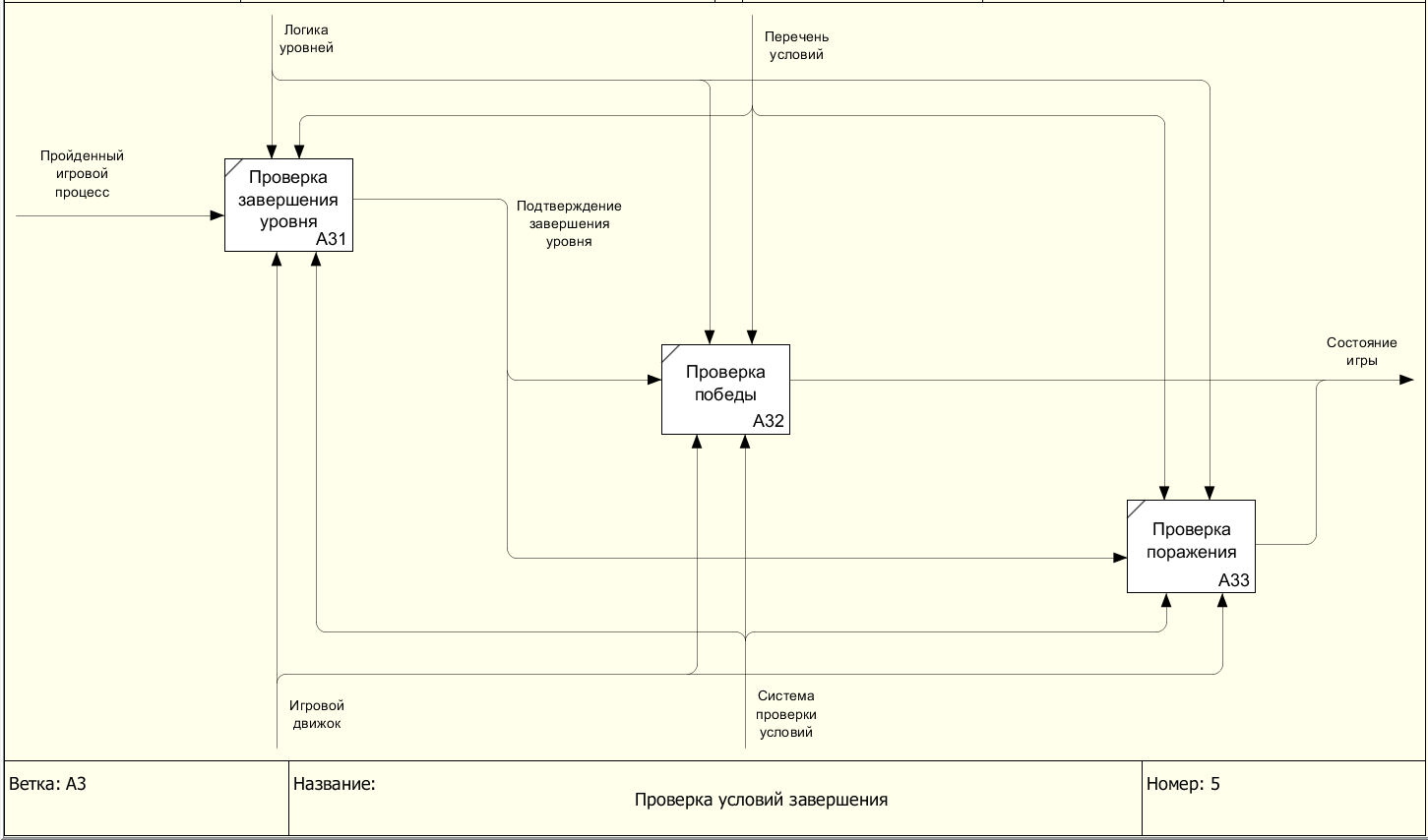
**Рисунок 5.4 – Декомпозиция процесса «Управление игровым процессом»**



**Рисунок 5.5 – Декомпозиция процесса «Генерация уровня»**



**Рисунок 5.6 – Декомпозиция процесса «Управление движением игрока»**



**Рисунок 5.7 – Декомпозиция процесса «Проверка условий завершения»**

**5.3 Итог**

В ходе выполнения данной практической работы были разработаны структурные диаграммы и модели процессов, которые позволили детально описать архитектуру и логику игры. С помощью диаграмм классов и объектов удалось визуализировать ключевые компоненты игры, такие как игрок, уровень, платформы, NPC, крюк и точка сохранения, а также их взаимосвязи. Это помогло чётко определить структуру системы и взаимодействие между её элементами. Моделирование процессов в нотации IDEF0 позволило детально описать игровой процесс, начиная с генерации уровня и заканчивая завершением игры. Были выделены ключевые процессы, такие как управление движением игрока, а также проверка условий победы или поражения. Каждый процесс был разбит на подпроцессы, что помогло определить входы, выходы, управления и механизмы для каждого этапа игры. Это обеспечило глубокое понимание логики игры и упростило проектирование и тестирование.

**ПРАКТИЧЕСКАЯ РАБОТА №6**

**6.1 Задание**

1. Посредством анализа прошлых диаграмм и изучив материал практики, выполнить построение диаграммы в нотации DFD для своего проекта.
2. Словесно описать информационное взаимодействие компонентов системы.
3. Построить нормализованную логическую модель базы данных собственного проекта.

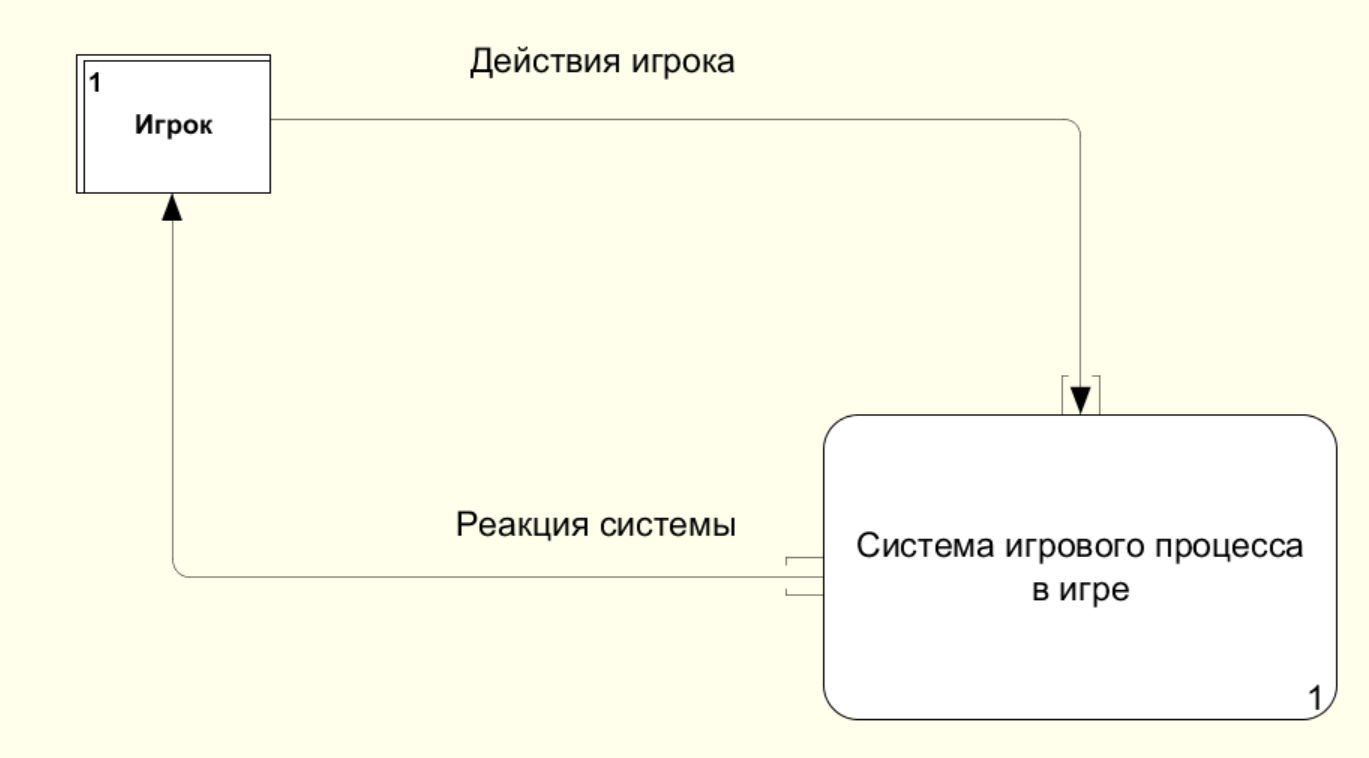
**6.2 Выполнение работы**

**6.2.1 Построение диаграммы в нотации DFD**

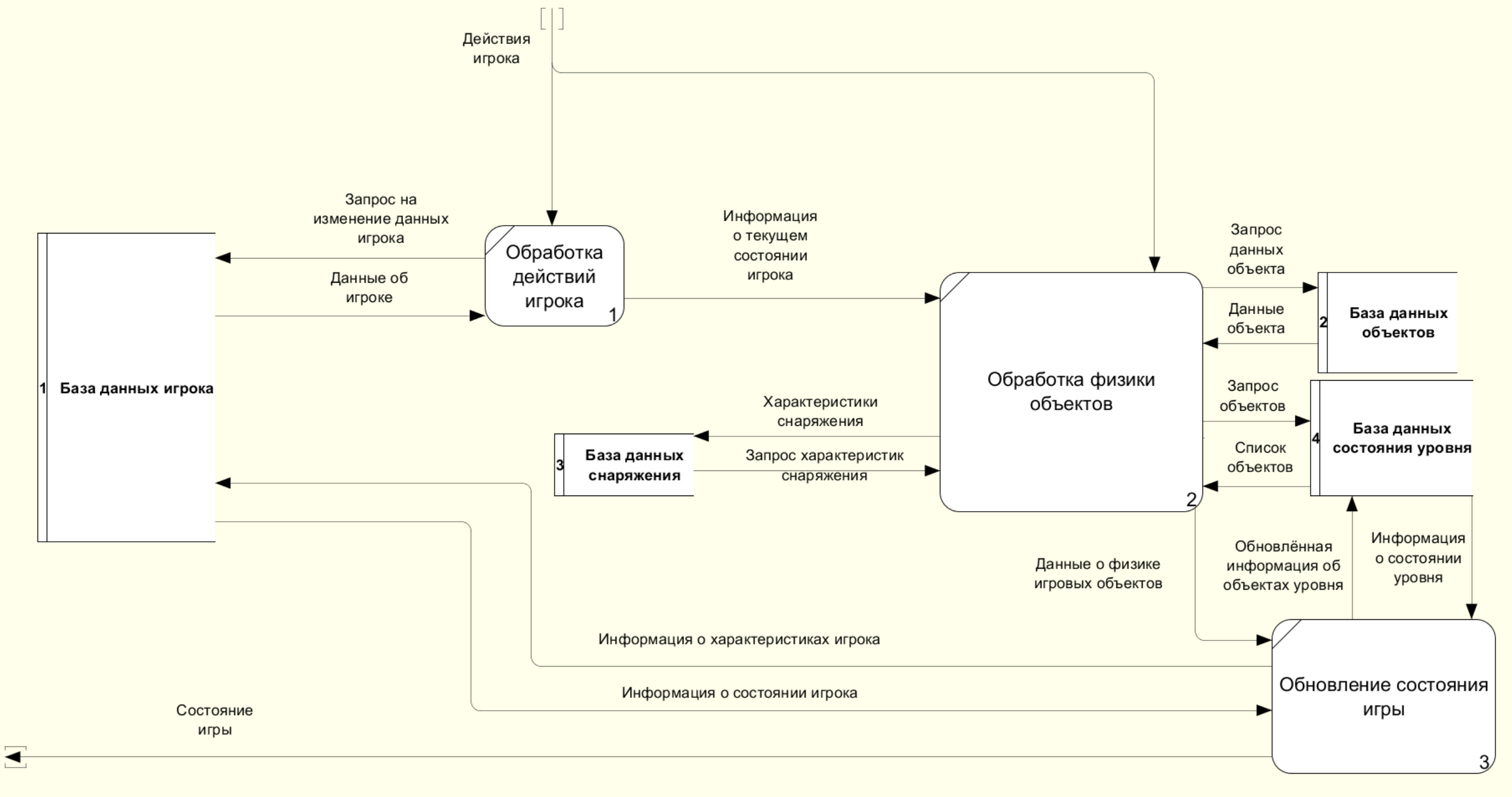
Диаграмма потоков данных позволяет представлять процессы, данные и их потоки наглядным образом, что позволяет упростить разработку сложных систем.

В данном случае DFD диаграмма позволяет визуализировать взаимодействие игрока с игрой совместно с внутренними процессами системы.

На диаграмме потребуется отобразить взаимодействие игрока с персонажем и уровнем, изменение состояний объектов и передачу информации о состоянии игры игроку.



**Рисунок 6.1 – Верхний уровень диаграммы**

****

**Рисунок 6.2 – Декомпозиция диаграммы**

**6.2.2 Словесное описание взаимодействия компонентов системы**

Каждый этап взаимодействия компонентов системы, обработка действий игрока, обработка физики объектов, обновление состояния игры, включает в себя интеракцию между системой, базами данных и игроком, для корректного игрового процесса.

**6.2.2.1 Обработка действий игрока**

1. Игрок меняет снаряжение:

Запрос на смену снаряжения отправляется в систему для дальнейшей обработки.

2. Обработка смены снаряжения:

Запрос отправляется в базу данных игрока, если данное снаряжение присутствует у игрока, то система изменяет экипированное, в ином случае смены не происходит, в конце возвращаются все данные о игроке.

3. Передача информации о состоянии игрока:

Система проверяет был ли запрос на смену снаряжения, если его не было, тогда она запрашивает его текущее состояние.

**6.2.2.2 Обработка физики объектов**

1. Запрос характеристик снаряжения:

Из базы данных снаряжения системой запрашиваются характеристики экипированного снаряжения.

2. Изменение физики объектов:

После того, как пришли данные снаряжение, система вычисляет новую скорость игрока и в зависимости от его действий изменяет вектор движения, затем система запрашивает данные обо всех активных объектах на карте из базы данных состояния уровня и обновляет их данные, если игрок взаимодействовал с объектом на карте, то система запрашивает данные об объекте и в зависимости от его свойств (можно ли присоединиться к нему, притягивается ли объект и т.д.) обновляет их физику.

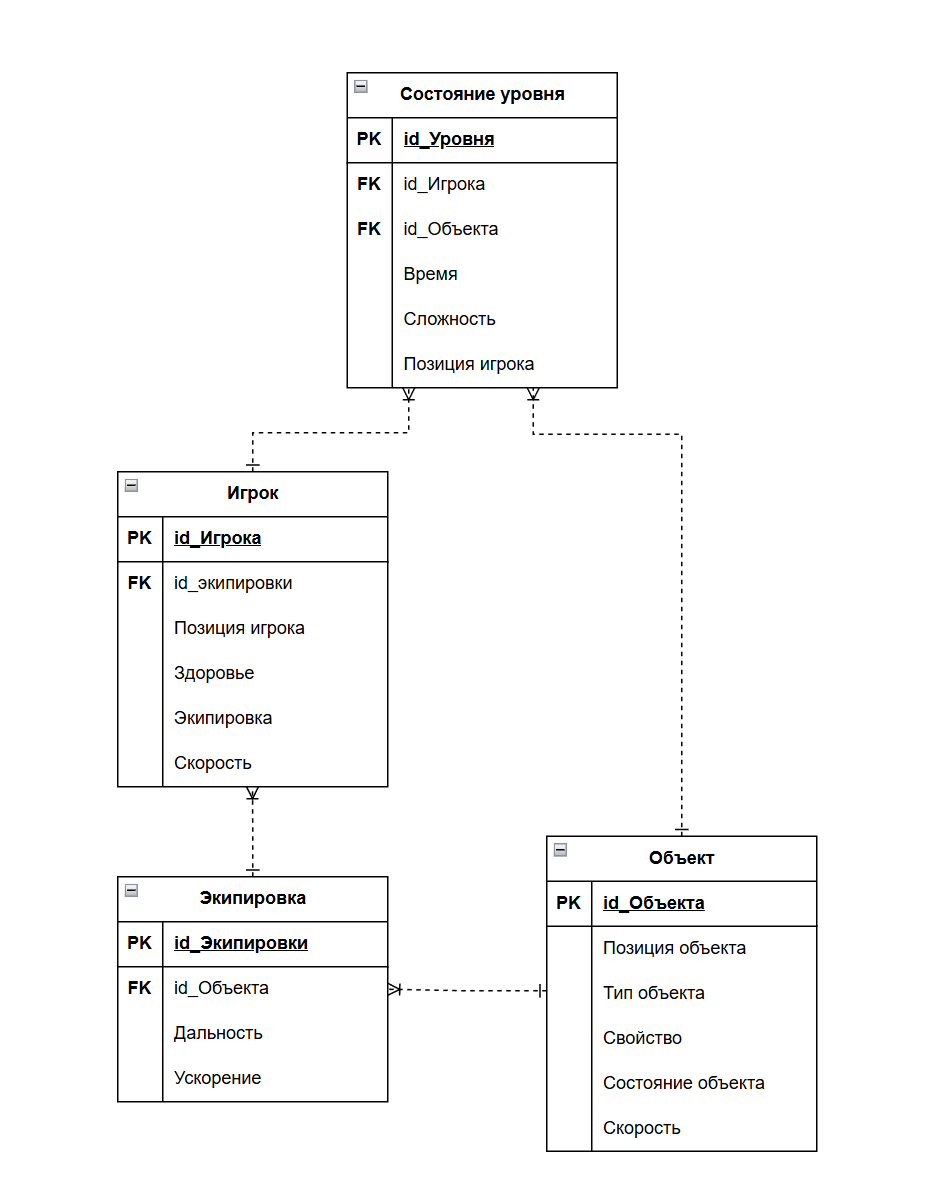
**6.2.2.3 Обработка состояния игры**

1. Обновление данных:

После обновления физики система передаёт обновлённые данные (новую скорость, направление и здоровье) в базу данных игрока, где они сохраняются. Также система заносит обновлённые данные об объектах (новые активные объекты или уже неактивные объекты) в базу данных состояния уровня.

**6.3. Логическая модель базы данных**

Логическая модель базы данных для нашей игры представляет собой структурированное описание данных, необходимых для функционирования игры. Основные сущности модели включают состояние уровня, игрок, объект, экипировка. Каждая сущность имеет свои атрибуты и связи с другими сущностями, что позволяет эффективно управлять данными и обеспечивать корректную работу игры.



**Рисунок 6.3 – Диаграмма объектов**

Состояние уровня отслеживает текущие объекты, время, проведённое на уровне, его сложность и позицию игрока. Состояние уровня связано с игроком и объектами на уровне. Каждый объект имеет свои определённые свойства, скорость, позицию и текущее состоянии (сломан, горит и т.д.).

Игрок, являясь ключевой сущностью, хранит информацию о своём местоположении, здоровье, скорости и экипировке, он связан с текущей экипировкой. Каждая экипировка имеет ускорение, крюк также имеет информацию о дальности своего применения, экипировка может быть связана с объектом, если они контактируют.

**6.4 Итог**

В ходе выполнения практической работы были выполнены задачи по созданию информационной диаграммы системы и разработке логической модели базы данных для игры. Построение диаграммы потоков данных (DFD) позволило визуализировать ключевые процессы, хранилища данных и взаимодействие между компонентами системы. Это обеспечило четкое понимание того, как информация перемещается в системе, что является важным этапом для дальнейшей разработки и оптимизации игрового процесса. Кроме того, словесное описание взаимодействия компонентов системы помогло глубже понять логику работы игры, включая обработку действий игрока, физики игры и обновление состояния уровня. А разработанная логическая модель базы данных обеспечивает целостность данных и минимизирует избыточность, что важно для эффективной работы системы. Полученные результаты могут быть использованы для дальнейшей реализации проекта, а также для анализа и улучшения процессов в игре.

# **ПРАКТИЧЕСКАЯ РАБОТА №7**

7.1. Задание

1. Необходимо описать предлагаемую архитектуру системы и обосновать выбор определенных программных решений для реализации ее компонентов. Программные решения должны включать в себя языки реализации будущей системы, планируемые к применению фреймворки, движки, базы данных и т.д.

2. Построить архитектурную диаграмму своей разработки.

3. Отразить в матрице требований соответствие каждого требования компоненту архитектуры, в котором оно реализуется.

**7.2 Архитектура системы**

Для реализации игры была выбрана концептуальная архитектура из-за её гибкости, которая является важным критерием выбора архитектуры при реализации игровых проектов.

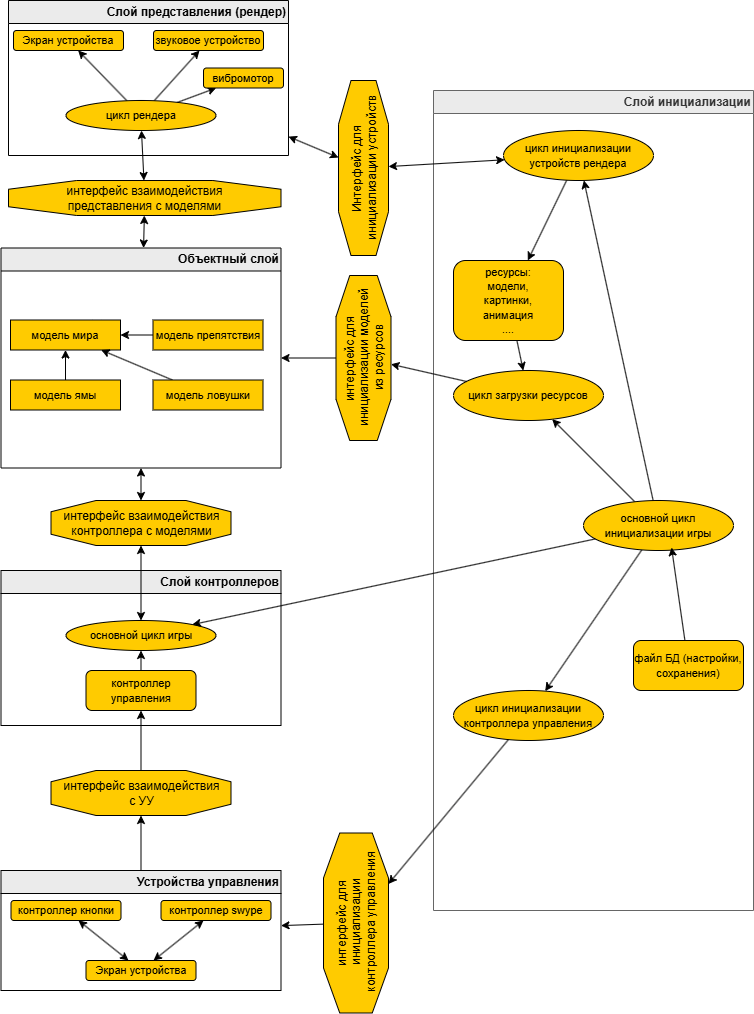
Разработка игры начинается с выбора игрового движка, который определяет дальнейшие ограничения при выборе стека технологий. В качестве игрового движка для разработки видеоигры был выбран Unreal Engine 5. Главные причины выбора – простота и многофункциональность. Основной язык написания скриптов (которые обеспечивают функционирование игры) – C++.

Для создания моделей для игры будет использоваться blender, модели из которого будут переноситься в игровой движок через файлы формата “.fbx”.

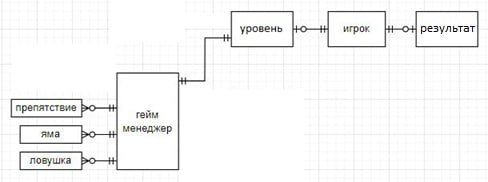
База данных будет использоваться для хранения сохранений игрока. Для реализации будут использоваться текстовые файлы, которые будут хранить данные в зашифрованном виде на устройстве игрока.

**7.3 Архитектурная диаграмма**

Была составлена диаграмма архитектуры проекта, изображенная на рисунке 7.1.



**Рисунок 7.1 Архитектурная диаграмма**



**Рисунок 7.2 Слой логики**

**7.4 Матрица требований**

Созданная ранее матрица требований была дополнена.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № | Требование | | Суть | | Автор | | | Ссылки | | Критерий проверки | Компоненты архитектуры |
| 1 | Разработка управления игры. | | | | | | | | | |  |
| 1.1 | Разработка системы управления | | Необходимо найти подходящее управление под программу | | Горохов М.П. | | | https://jmeter.apache.org/ | | Проверка на нескольких пользователях | Input System (Enhanced Input) – система обработки ввода в UE5, Gameplay Ability System (GAS) – для сложных взаимодействий |
| 1.2 | Изменение управления | | Необходимо добавить функцию смены управления | | Горохов М.П. | | | https://atnsoft.ru/keymanager/ | | Изменение управления и тестирование | Input Mapping Context (Enhanced Input) – динамическая переконфигурация управления, Widget Blueprint – меню настроек |
| 2 | Разработка интерфейса игры | | | | | | | | | |  |
| 2.1 | Проверка доступности | | Оценка доступности системы для пользователей с цветовой слепотой. | | Караваев И.И. | | | https://dtf.ru/gamedev/68075-kak-daltonizm-vliyaet-na-opyt-igrokov-osnovnye-problemy-i-ih-resheniya | | Тестирование с включенным фильтром цветовой слепоты. | UMG (Unreal Motion Graphics) – система UI с поддержкой accessibility, Post Process Materials – фильтры коррекции цвета |
| 2.2 | Проверка адаптивности интерфейса | | Тестирование работы интерфейса на разных мониторах. | | Караваев И.И. | | | https://gamedev.stackexchange.com/questions/73823/why-do-games-ask-for-screen-resolution-instead-of-automatically-fitting-the-wind | | Проверка корректного отображения на мониторах различного разрешения. | Canvas Panel + Anchors – адаптивная верстка в UMG, DPI Scaling – автоматическое масштабирование |
| 2.3 | Тестирование пользовательского опыта | | Оценка удобства использования и логики интерфейса | | | Караваев И.И. | | https://www.crazyegg.com/ | | Проведение тестирования с реальными пользователями, сбор отзывов. | Analytics (Unreal Insights, Google Analytics) – сбор данных, Widget Navigation System – логика переходов |
| 3 | Создание уровней игры | | | | | | | | | |  |
| 3.1 | Создание цепочки уровней | | Необходимо добавить сюжетные уровни для игроков | | | Горохов М.П. | | https://habr.com/ru/articles/274483/ | | Проверка наличия уровней в игре | Level Streaming – динамическая подгрузка уровней, Data Tables – хранение параметров уровней |
| 3.2 | Выбор уровней | | Добавить возможность выбора уровня после его открытия | | | Горохов М.П. | | https://stackoverflow.com/questions/75935599/how-to-make-a-level-selection-page-and-levels-act-as-individual-with-unity-2d | | Выбор уровня внутри игры | Level Selector Widget (UMG) – меню выбора, SaveGame System – запись прогресса |
| 3.3 | Добавление оценки за прохождение | | На основании нескольких факторов | | | Лобода А.С. | | https://create.roblox.com/docs/tutorials/use-case-tutorials/scripting/basic-scripting/score-points | | Прохождение уровней несколько раз | Scoring System (Blueprint/C++) – логика подсчета очков, Leaderboard (Online Subsystem) – таблица рекордов |
| 4 | Механика передвижения | | | | | | | | | |  |
| 4.1 | Уникальный способ передвижения | | Добавить уникальную механику передвижения | | | Андреев Д.А. | | https://dtf.ru/indie/221174-kak-hudozhniku-pridumat-mehaniki-dlya-svoei-igry-baza-geimdizaina | | Проверка работоспособности механики | Character Movement Component (Custom) – кастомная физика, Animation Blueprint – анимации |
| 4.2 | Базовые способы передвижения | | Добавить стандартный для игр от первого лица способ передвижения | | | Андреев Д.А. | | https://habr.com/ru/articles/164489/ | | Проверка передвижения внутри игры. | First Person Template (UE5) – готовая реализация FPS, Collision System – обработка столкновений |
| 5 | Производительность и надёжность | | | | | | | | | |  |
| 5.1 | Оптимизация под железо | | Тестирование игры на оборудование указанном в минимальных требованиях | | Караваев И.И. | | | https://tproger.ru/articles/vyjavlenie-i-sbor-trebovanij-k-po-ultimate-guide | | Тестирование с подходящим оборудованием | LODs (Level of Detail) – динамическая детализация, Nanite – для статичных объектов |
| 5.2 | Проверка на вылеты | | Проверить работоспособность приложения на длительном промежутке времени | | Лобода А.С. | | | https://dev.epicgames.com/documentation/en-us/unreal-engine/crash-reporting-in-unreal-engine | | Запуск приложения на несколько часов под нагрузкой. | Crash Reporter (Unreal Engine) – автоматический сбор логов, Profiler (Unreal Insights) – мониторинг производительности |
| 5.3 | Работоспособность приложения | | Проверка работоспособности приложения после скачивания. | | Караваев И.И. | | | https://metanit.com/sharp/aspnet6/18.1.php | | Запустить приложение. | Packaging Tool (UE5) – сборка релизной версии, Platform-specific Settings – настройки под Android/iOS |
| 5.4 | Проверка на ошибки | | Найти различные баги и исправить их | | Андреев Д.А. | | | https://vagon.io/blog/best-bug-tracking-testing-and-management-tools-for-games | | Длительное тестирование игры с нестандартным подходом. | Automated Testing (Gauntlet, Unreal Automation) – автоматизированные тесты, Debug Tools – отладка в редакторе |
| 6 | Сохранение прогресса | | | | | | | | | |  |
| 6.1 | Добавить сохранения | | Добавление опции сохранения прогресса внутри игры | | | | Андреев Д.С. | | https://dev.to/hexblit/save-your-game-json-dictionaries-a-godot-game-engine-tutorial-135c | Проверка работоспособности сохранений. | SaveGame System (UE5) – сериализация данных, JSON/Binary Serialization – формат хранения |
| 6.2 | Выбор сохранения | | Добавить возможность выбора сохранения | | | | Андреев Д.С. | | https://www.kodeco.com/418-how-to-save-and-load-a-game-in-unity | Выбрать сохранение и отследить изменения | SaveGame Slots – система слотов сохранений, UMG (Load Game Menu) – интерфейс выбора |
| 7 | Контроль сроков разработки | | | | | | | | | |  |
| 7.1 | Планирование этапов | | | Разработка плана работ с выделением ответственных лиц и сроков. | | | Горохов М.П. | | https://www.monitask.com/en/blog/how-to-write-a-work-plan | Наличие плана-графика в проектном инструменте. | Jira/ClickUp Integration – трекинг задач, Unreal Project Version Control (Perforce/Git LFS) – управление версиями |
| 7.2 | Отчетность по задачам | | | Каждая задача должна иметь статус, ответственного и дедлайн | | | Горохов М.П. | | https://aptien.com/en/kb/articles/how-to-manage-tasks | Регулярное обновление статусов задач | Scrum/Agile Boards – визуализация прогресса, Slack/MS Teams Notifications – уведомления |
| 7.3 | Мониторинг выполнения | | | Оценка прогресса проекта и принятие корректирующих мер | | | Лобода  А.С. | | https://ru.smartsheet.com/ | Наличие отчетов о ходе выполнения работ | Burndown Charts (Jira) – аналитика, Unreal Insights (Performance Metrics) – метрики сборки |
| 8 | | Коммуникация внутри команды | | | | | | | | |  |
| 8.1 | | Проведение собраний | Регулярные встречи команды для обсуждения прогресса и проблем | | Горохов М.П. | | | | https://www.zoom.com/ru | Фиксация результатов собраний | Zoom/Google Meet Integration – видеоконференции, Confluence/Notion – документация |
| 8.2 | | Документирование решений | Вся информация о проекте должна фиксироваться и быть доступной для команды | | Андреев Д.А. | | | | https://github.com/ | Наличие актуальной документации | GitHub Wiki / Markdown Docs – хранение документации, Unreal Engine Documentation Tool – встроенная система |
| 8.3 | | Обратная связь | Сбор и анализ обратной связи от команды | | Горохов М.П. | | | | https://workspace.google.com/intl/ru/products/docs/ | Проведение опросов внутри команды | Google Forms / Typeform – сбор фидбека, Retrospective Meetings – разбор итераций |

**ПРАКТИЧЕСКАЯ РАБОТА №8**

* 1. **Задание на практическую работу**

1. Необходимо обосновать выбор ГОСТа для разработки технического задания, почему был выбран тот или иной стандарт, по каким разделам данный ГОСТ подходит больше и т.д.
2. Составить в соответствии с выбором техническое задание по ГОСТу или дизайн документ (диздок).
   1. **Ход выполнения практической работы**
      1. **Выбор ГОСТа для разработки технического задания**

Для разработки технического задания (ТЗ) был выбран ГОСТ 34.602- 2020, так как он наиболее полно охватывает требования к автоматизированным системам (АС), что соответствует характеру проекта. Игра является программным продуктом, который требует детального описания функциональных требований, технических характеристик и порядка разработки.

ГОСТ 34.602-2020 — это государственный стандарт, который регламентирует требования к разработке технического задания (ТЗ) на автоматизированные системы (АС). Этот стандарт применяется для создания документации, которая определяет цели, задачи, требования и порядок разработки автоматизированных систем.

Основные разделы ГОСТ 34.602-2020:

* Общие сведения — содержит информацию о системе, заказчике, разработчике, сроках выполнения работ и источниках финансирования.
* Назначение и цели создания системы — описывает, для чего создается система и какие цели она должна достичь.
* Характеристика объектов автоматизации — содержит описание объектов, которые будут автоматизированы, и условий их эксплуатации.
* Требования к системе — включает требования к функциональности, надежности, безопасности, эргономике и другим аспектам системы.
* Состав и содержание работ по созданию системы — описывает этапы разработки, сроки и ответственных за выполнение работ.
* Порядок контроля и приемки системы — определяет, как будет проводиться тестирование и приемка системы.
* Требования к документированию — перечень документов, которые должны быть разработаны в процессе создания системы.
* Источники разработки — список документов и материалов, на основе которых разрабатывается ТЗ.

Ниже приведены основные причины выбора этого стандарта:

* + - 1. **Соответствие проекта критериям автоматизированной системы**

Игра представляет собой программный продукт, который автоматизирует процесс взаимодействия пользователя с игровым миром. Игра включает в себя множество автоматизированных процессов, таких как:

* + - * + Генерация уровней.
        + Обработка действий игрока (Перемещение игрока с помощью крюка между платформами, сбор предметов).
        + Сохранение и загрузка игрового прогресса.

Все эти процессы требуют четкого описания в техническом задании, что соответствует требованиям ГОСТ 34.602-2020.

* + - 1. **Полнота и структурированность стандарта**

ГОСТ 34.602-2020 предлагает детализированную структуру технического задания, которая охватывает все аспекты разработки автоматизированной системы. Это позволяет:

* + - * + Четко определить цели и задачи проекта.
        + Указать требования к функциональности, надежности и безопасности системы.
        + Описать порядок разработки, контроля и приемки системы.

Для нашей игры важно, чтобы все аспекты разработки были задокументированы, включая требования к интерфейсу, игровым механикам и техническому обеспечению.

* + - 1. **Гибкость стандарта**

ГОСТ 34.602-2020 допускает адаптацию разделов под специфику проекта. Это означает, что можно:

* + - * + Исключить разделы, которые не относятся к проекту (например, требования к транспортабельности для подвижных систем).
        + Добавить дополнительные разделы, если это необходимо (например, описание игровых механик или сюжета).

Опять же, для нашей игры — это особенно важно, так как проект имеет уникальные особенности, такие как процедурная генерация уровней и система передвижения персонажа с помощью крюк-кошкой.

* + - 1. **Учет требований к безопасности и надежности**

Стандарт включает разделы, посвященные требованиям к безопасности и надежности системы. Для Игры это важно, так как:

* + - * + Необходимо обеспечить защиту данных пользователя (например, прогресс игры).
        + Система должна быть устойчива к сбоям и быстро восстанавливаться после ошибок.
      1. **Соответствие современным стандартам разработки**

ГОСТ 34.602-2020 является актуальным стандартом, который учитывает современные требования к разработке автоматизированных систем. Это важно для игры, так как она должна соответствовать современным технологическим стандартам (например, поддержка C++, Blueprints, JavaScript).

* + 1. **Техническое задание по ГОСТу**

1. **Общие сведения**

Игра "Hook-Game" (условное обозначение CHC-25) разрабатывается по заказу РТУ МИРЭА (г. Москва, Проспект Вернадского, д. 78) силами компании-разработчика ООО "D.I.M.A." (г. Москва, Проспект Вернадского, д. 78). Основанием для разработки является технико-экономическое обоснование (ТЭО) от 10.02.2025 и договор на разработку №11 от 12.02.2025.

Плановые сроки выполнения работ: начало разработки — 12.02.2025, окончание разработки — 28.05.2025. Источником финансирования является бюджет заказчика. Результаты работ оформляются в виде технической документации и передаются заказчику по окончании каждого этапа разработки. Приемка работ осуществляется по актам сдачи-приемки.

1. **Назначение и цели создания системы**

Игра "Hook-Game" предназначена для предоставления пользователям интерактивного игрового опыта, в котором игрок управляет персонажем, перемещающимся по платформам с помощью крюк-кошки. На каждом шаге игроку предоставляется выбор на какую платформу переместится и с каким НПС поговорить(договорится), чтобы продолжить свой путь, ведь от каждого такого выбора зависит отношение других НПС на игрока. Благодаря этому создается элемент неопределенности и стратегического планирования.

Целями создания системы являются:

* + Создание увлекательной и интерактивной игры с элементами стратегии и случайности.
  + Обеспечение удобного и интуитивно понятного интерфейса для пользователя.
  + Достижение высокой производительности и отзывчивости системы.
  + Обеспечение кросс-платформенной поддержки (ПК, мобильные устройства).

1. **Характеристика объектов автоматизации**Объектом автоматизации в проекте является игровой процесс, который включает перемещение игрока по платформам, сбор предметов, преодоление препятствий (ловушки), а также сохранение и загрузку игрового прогресса. Игра должна функционировать на современных компьютерах. Минимальные требования к устройству пользователя: операционная система - Windows 10 (64-разрядная); процессор - четырёхъядерный Intel или AMD с частотой 2,5 ГГц; оперативная память - 8 ГБ; графический процессор- DirectX 11 или DirectX 12, совместимая видеокарта; место на диске - 50 ГБ SSD.
2. **Требования к системе**

Система должна удовлетворять минимальным системным требованиям.

К функциональным требованиям относятся:

* + Функция выбора пути на каждом шаге.
  + Функция сбора предметов и преодоления препятствий.
  + Функция сохранения и загрузки игрового прогресса.
  + Функция отображения текущего прогресса

Требования к видам обеспечения включают:

* + Программное обеспечение: Использование современных технологий (C++, Blueprints, JavaScript).
  + Техническое обеспечение: Поддержка работы на устройствах с разрешением экрана не менее 1920x1080.
  + Информационное обеспечение: Использование базы данных для хранения прогресса пользователей.
  + Организационное обеспечение: Разработка руководства пользователя и технической документации.

1. **Состав и содержание работ по созданию системы**

Разработка системы будет осуществляться в несколько этапов:

1. Анализ требований и проектирование системы (12.02.2025 - 17.02.2025). На этом этапе будут определены основные требования к системе, разработана архитектура и интерфейс.
2. Разработка игрового движка и интерфейса (18.02.2025 - 10.03.2025). На этом этапе будет реализован основной функционал игры, включая генерацию платформ, обработку действий игрока и сохранение прогресса.
3. Тестирование и отладка (11.03.2025 - 24.03.2025). На этом этапе будет проведено функциональное, производительное и безопасное тестирование системы.
4. Внедрение и запуск (31.03.2025 - 07.04.2025). На этом этапе система будет передана в эксплуатацию.

По окончании каждого этапа будут предоставлены соответствующие документы: техническое задание, проектная документация, отчеты по тестированию и руководство пользователя.

1. **Порядок контроля и приемки системы**

Контроль и приемка системы будут осуществляться на основе результатов испытаний, которые включают функциональное тестирование, тестирование производительности и тестирование безопасности. Результаты испытаний оформляются в виде отчетов и передаются заказчику для утверждения. Приемка работ осуществляется по актам сдачи-приемки.

1. **Требования к документированию**

В процессе разработки системы должны быть подготовлены следующие документы:

* + Техническое задание.
  + Документация разработчика.
  + Документация пользователя.
  + Отчеты по тестированию.
  + Проектная документация.

Все документы должны соответствовать стандартам ЕСКД (единая система конструкторской документации) и быть предоставлены в электронном и печатном виде.

1. **Источники разработки**

В качестве источников разработки использовались следующие документы и материалы:

* + Отчеты по аналогичным проектам.
  + Научные статьи и исследования.
  + Информационные материалы по игровым движкам и технологиям.
  + Техническая документация по используемым технологиям.

**Приложения**

Приложение 1. Расчет ожидаемой эффективности системы.

Ожидаемое количество пользователей: 25 в месяц. Ожидаемый доход от монетизации: 100 рублей в месяц.

Приложение 2. Оценка научно-технического уровня системы. Использование современных технологий (C++, Blueprints, JavaScript). Поддержка кросс-платформенности (ПК, мобильные устройства).

* 1. **Вывод**

В ходе выполнения практической работы было разработано техническое задание на создание игры "Hook-Game" в соответствии с требованиями ГОСТ 34.602-2020. Данный стандарт был выбран благодаря его полноте и структурированности, что позволяет охватить все аспекты разработки автоматизированных систем, включая функциональные, технические и организационные требования.

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения практических работ №5-8 были разработаны структурные диаграммы, которые помогли визуализировать основные элементы игры. Это позволило четко определить взаимодействие между объектами и логику игрового процесса. Особое внимание было уделено проектированию архитектуры системы. Ну и важным этапом стало формирование технического задания, которое было составлено в соответствии с ГОСТ 34.602-2020. Полученные результаты станут основой для дальнейшей реализации проекта, а также помогут избежать ошибок на этапе разработки. Проект "Hook-Game" имеет все шансы стать интересной и качественной игрой, которая привлечет внимание игроков своей уникальной механикой и атмосферой.

**СПИСОК ИСТОЧНИКОВ**

* + - 1. https://docs.unrealengine.com/5.0/en-US/
      2. <https://store.steampowered.com/app/278360/A_Story_About_My_Uncle/>
      3. Методические указания по практическим работам №5-8 по предмету «Системная и программная инженерия» [Электронный ресурс] — URL: https://online-edu.mirea.ru/mod/assign/view.php?id=445056 (Дата обращения: 24.02.2025).
      4. ГОСТ 34.602-2020. Информационные технологии. Техническое задание на создание автоматизированной системы. – Дата введения 2022-01- 01. – Москва: Российский институт стандартизации, 2022. – 15 с. (Дата обращения: 24.02.2025).