1. **СТРУКТУРНОЕ ПРОЕКТИРОВАНИЕ**

После анализа всех требований к разрабатываемому проекту, мы переходим к разбиению системы на функциональные блоки. Этот метод позволяет создавать гибкую архитектуру приложения, что в свою очередь позволяет изменять существующие и добавлять новые функциональные блоки, не затрагивая общую работу системы.

Структура проекта состоит из следующих блоков:

* блок игровой логики;
* блок игрового контроллера;
* блок игрового интерфейса;
* блок управления персонажем;
* блок игрового меню;
* блок чтения аудиофайла;
* алгоритм анализа аудиосэмплов;
* алгоритм вычисления музыкального ритма.

Взаимосвязь основных блоков проекта отображена на структурной схеме ГУИР.400201.304 С1.

* 1. **Блок игровой логики**

Данный блок содержит в себе базовую логику игровых правил и взаимодействий игрока с игровыми объектами. В блоке игровой логики описываются основные логические элементы игры, такие как условия проигрыша, подсчёт очков игрока. В данный блок так же входит установление взаимосвязей между другими блоками данного проекта. Для реализации данного блока будет использован язык визуального программирования Blueprints.

* 1. **Блок игрового контроллера**

Данный блок отвечает за всевозможные взаимодействия игроком, с игровым миром, через игрового персонажа. В данном блоке описываются основные принципы передвижения персонажа, взаимодействия с предметами, реакции персонажа на события и так далее. Для реализации данного блока, также будет использован язык визуального программирования Blueprints.

* 1. **Блок игрового интерфейса**

Блок игрового интерфейса отвечает за интерфейс главного меню игры, интерфейса для взаимодействия с аудиофайлами, а так же за интерфейс самого игрового процесса, счётчика очков и других элементов индикации и вывода информации пользователю. Для реализации данного блока будет использован язык визуального программирования Blueprints.

* 1. **Блок управления персонажем**

Для осуществления управления игровой пешкой необходимо при считывании нажимаемых клавиш, создавать события, которые в дальнейшем будут передаваться в блок игрового контроллера. Нажатия должны считываться как с клавиатуры, так геймпада. В данном блоке должна быть реализована реакция на нажатие определённых, заранее установленных разработчиком клавиш, отвечающих за исполнение игровой пешкой конкретных действий. Для реализации данного блока будет использован язык визуального программирования Blueprints, а также настройки проекта Unreal Engine 4.

* 1. **Блок игрового меню**

Данный блок представляет собой реализацию игрового меню и окон для взаимодействия с игроком, созданных с помощью внутренних инструментов Unreal Engine 4, для создания графических пользовательских интерфейсов. Логика взаимодействия пользователя с интерфейсом описывается на языке визуального программирования Blueprints.

* 1. **Блок чтения аудиофайла**

Блок алгоритма осуществляющего чтение заголовка и блока данных аудиофайла. Первоначальной задачей алгоритма является определение формата файла, валидация событий, приводящих к возникновению ошибок при чтении и открытие файла в режиме чтения. Аудиофайл должен соответствовать формату .wav, так как данный формат файла используется для хранения несжатого аудиосигнала в импульсно-кодовой модуляции. После открытия файла, главными задачами блока будут являться, чтение заголовка, инициализация структуры для сохранения данных из заголовка аудиофайла, таких как количество каналов, частоту дискретизации, аудиоформат, наличие и тип кодировки, количество байт для хранения одного сэмпла, а так же общий размер файла без учета первых 16 байт. Данная информация необходима для корректного чтения блока данных конкретного аудиофайла. Так же полученная информация будет использована для анализа композиции в блоке анализа аудиофайла. Данный алгоритм будет реализован с использованием языка программирования С++.

* 1. **Алгоритм анализа аудиосэмплов**

Блок алгоритма анализа аудиосэмплов отвечает за обработку и анализ блока данных. Используя информацию, полученную в заголовке файла в алгоритме чтения файла, производится быстрое преобразование Фурье с использованием окна Гаусса. Использование окна Гаусса позволит избавиться от возможного появления шумов, после применения БПФ функции. Входными значениями алгоритма являются массив с амплитудно-временными значениями, а так же размер данного массива. Выходным значением является массив с амплитудно-частотными значениями, отображающими перепады амплитуд конкретных диапазонов частот в определённый промежуток времени. На основе данных значений производится анализ и вычисление ритма музыкальной композиции, который в дальнейшем отправляется в блок игровой логики. Данный алгоритм будет реализован с использованием языка программирования С++.

* 1. **Алгоритм вычисления музыкального ритма**

В данном алгоритме, исходя из полученных данных алгоритмов чтения аудиофайла, а также из алгоритма анализа аудиосэмплов, происходит вычисление музыкального ритма композиции. В учёт берутся количество каналов, а так же пиковые амплитудно-частотные значения. Данный алгоритм будет реализован с использованием языка программирования С++.