Руководство пользователя

Рекомендуется перед прочтением ознакомится с пользовательским соглашением

Содержание:

1. Введение

2. Общая Структура Проекта

3. Создание и Изменение Сюжета Игры (Система Узлов)

4. Управление Музыкой в Игре

5. Локализация (Перевод Текста на Разные Языки)

6. Настройка Внешнего Вида Пользовательского Интерфейса (UI)

7. Функции Сохранения и Загрузки Игры

8. Встроенные Диалоги (Всплывающие Окна)

1 Введение

1.1 Назначение Руководства

Это руководство создано для того, чтобы максимально подробно и понятно объяснить, как устроен проект текстового квеста PatternForQuest. Оно предназначено для всех, кто хочет не просто запустить приложение, но и изменять его, добавлять собственный контент, настраивать внешний вид и понимать, как работают основные механизмы игры. Независимо от вашего уровня опыта в разработке под Android, здесь вы найдете пошаговые инструкции и детальные объяснения, которые помогут вам создать свою уникальную интерактивную историю.

1.2. Обзор Функционала Проекта

Проект PatternForQuest предоставляет собой прочную и гибкую основу для создания текстовых приключений с графическим сопровождением. Его ключевые возможности включают:

Главное меню (MainActivity): Начальный экран приложения, предлагающий опции для старта новой игры, продолжения сохраненной, доступа к настройкам и выхода из приложения.

Игровой экран (GameActivity): Основной интерфейс игры, где разворачивается сюжет. Он разделен на верхнюю часть для отображения изменяющегося изображения сцены и нижнюю для текста истории.

Горизонтальная Ориентация: Все экраны приложения жестко зафиксированы в альбомной (горизонтальной) ориентации. Это обеспечивает оптимальное представление визуального и текстового контента, а также последовательный пользовательский опыт.

Система Узлов Сюжета (Node): История игры структурирована в виде взаимосвязанных "узлов". Каждый узел представляет собой дискретный фрагмент сюжета, который включает в себя специфический текст, соответствующее изображение и логику переходов к следующим узлам.

Динамические Выборы Игрока: В определенных точках сюжета игроку предоставляются варианты выбора, представленные двумя кнопками. Эти кнопки появляются только тогда, когда узел предусматривает выбор, и автоматически скрываются после того, как выбор сделан, поддерживая чистоту интерфейса.

Механизм Сохранения и Загрузки: Реализована возможность сохранения текущего прогресса игры. Игрок может вручную сохранить игру через внутриигровое меню. Загрузка сохраненной игры доступна из главного меню.

Настройки Приложения: Специальное всплывающее окно настроек предоставляет пользователю контроль над громкостью фоновой музыки и позволяет переключать язык интерфейса между русским и английским. Окно настроек автоматически масштабируется для удобства использования.

Фоновая Музыка: Единая музыкальная композиция сопровождает игрока на протяжении всего приложения, плавно переходя между экранами меню и игры, создавая атмосферу.

Полная Локализация: Весь текстовый контент приложения – сюжетные описания, надписи на кнопках, элементы интерфейса – поддерживает русский (по умолчанию) и английский языки, с простой возможностью добавления дополнительных языков.

1.3. Принципы Разработки Проекта

Проект разработан с учетом следующих принципов, которые облегчают его понимание и модификацию:

Чистый и Модульный Код: Код разделен на логически связанные модули (классы), каждый из которых выполняет определенную задачу (например, MusicManager только управляет музыкой, Node только описывает узел). Это значительно упрощает поиск нужного фрагмента кода, его изменение и отладку.

Стандартные Компоненты Android: Для построения пользовательского интерфейса и управления базовыми функциями используются стандартные компоненты и API Android (такие как Activity, Dialog, ImageView, TextView, Button, SharedPreferences,MediaPlayer). Это обеспечивает высокую надежность, хорошую производительность и совместимость с большинством Android-устройств.

Использование Ресурсов: Весь медиа- и текстовый контент (изображения, аудиофайлы, текстовые строки для разных языков, разметки UI) хранится в стандартизированных папках ресурсов Android Studio. Такой подход облегчает управление контентом, позволяет легко добавлять новые языки и оптимизировать ресурсы для устройств с разными характеристиками.

2. Структура

2.1. Ключевые Java-Классы и Их Роль

Все файлы с исходным кодом на Java, реализующие логику вашего приложения, расположены по пути: app/src/main/java/com/example/patternforquest .

2.1.1. MainActivity.java (Главное Меню)

MainActivity — это стартовая точка вашего приложения. Он является первым экраном, который видит пользователь, и служит центральным узлом для навигации по игре.

Ключевые обязанности:

Отображение Главного Меню: Загружает и отображает пользовательский интерфейс, определенный в activity\_main.xml, который включает кнопки "Начать игру", "Продолжить", "Настройки" и "Выход".

Управление Запуском Игры: Обрабатывает нажатия на кнопки "Начать игру" (запуская GameActivity с чистого листа) и "Продолжить" (запуская GameActivity с загрузкой последнего сохранения).

Интеграция с Настройками: Отвечает за вызов SettingsDialog при нажатии кнопки "Настройки". Важно, что MainActivity также реализует специальный интерфейс (SettingsDialog.SettingsDialogListener), чтобы получать уведомления от SettingsDialog об изменении языка.

Управление Музыкой в Меню: Инициализирует статический класс MusicManager при первом запуске приложения и управляет воспроизведением фоновой музыки, чтобы она играла в главном меню.

Обработка Выхода из Приложения: При нажатии кнопки "Выход" корректно завершает работу приложения и останавливает всю музыку, освобождая системные ресурсы.

Применение Языковых Изменений: Если пользователь меняет язык в настройках, MainActivity получает уведомление и вызывает метод recreate() на себе. Это приводит к перезагрузке Activity, благодаря чему весь её пользовательский интерфейс (UI) обновляется с текстами на выбранном языке.

Важные методы:

onCreate(Bundle savedInstanceState): Метод жизненного цикла Activity, вызывается при первом создании Activity. Здесь происходит начальная настройка: установка сохраненного языка, загрузка разметки (setContentView), инициализация MusicManager, поиск всех элементов UI (кнопок, фонового изображения) по их ID (findViewById) и настройка обработчиков нажатий для этих элементов.

onResume(): Метод жизненного цикла Activity, вызывается, когда Activity становится видимой и находится на переднем плане. Здесь происходит обновление состояния кнопки "Продолжить" (активна/неактивна в зависимости от наличия сохранения) и возобновляется проигрывание фоновой музыки.

onDestroy(): Метод жизненного цикла Activity, вызывается перед уничтожением Activity. Здесь происходит остановка музыки, но только в том случае, если MainActivity является последней Activity, что сигнализирует о полном завершении работы приложения.

setAppLocale(String lang): Вспомогательный приватный метод, который принимает код языка (например, "en" или "ru") и устанавливает его для всего приложения. Это влияет на то, какие файлы strings.xml будут использоваться для отображения текста.

onLanguageChanged(): Это метод, который MainActivity реализует как часть интерфейса SettingsDialog.SettingsDialogListener. Он вызывается из SettingsDialog после того, как пользователь изменил язык и закрыл диалог настроек. Единственное действие этого метода — вызов recreate(), который перезапускает MainActivity, чтобы UI обновился с новым языком.

2.1.2. GameActivity.java (Игровой Экран)

Назначение: GameActivity — это основной экран, где игрок взаимодействует с сюжетной линией текстового квеста.

Ключевые обязанности:

ОтображениеКонтента Узла: Динамически обновляет отображаемое изображение сцены (ImageView) и текст сюжета (TextView) в соответствии с данными текущего "узла" (Node) игры.

Управление Видимостью Выборов: Отображает кнопки выбора (первый и второй вариант), когда текущий узел предусматривает выбор, и автоматически скрывает их в остальное время, обеспечивая чистый интерфейс.

Обработка Взаимодействия Игрока: Реагирует на нажатия на текстовое поле (для продвижения по линейному сюжету, если нет выбора) и на кнопки выбора (для принятия сюжетных решений).

Взаимодействие с GameManager: Запрашивает у GameManager объект текущего Node для отображения его содержимого и сообщает GameManager о сделанном игроком выборе для перехода к следующему узлу.

Вызов Внутриигрового Меню: Предоставляет кнопку "Меню" в правом верхнем углу, которая вызывает InGameMenuDialog для доступа к функциям паузы, сохранения и выхода.

Обработка Конца Игры: При достижении "конечного узла" (isEndNode = true) отображает всплывающее сообщение "Конец игры!" (Toast) и затем автоматически возвращает игрока в главное меню.

Важные методы:

onCreate(Bundle savedInstanceState): Вызывается при создании Activity. Инициализирует GameManager, определяет, нужно ли загружать сохраненную игру (переданную из MainActivity) или начинать новую, находит все элементы UI и настраивает их обработчики нажатий, а затем вызывает updateUI() для первоначального отображения.

onResume(): Вызывается, когда GameActivity становится видимой. Здесь возобновляется проигрывание фоновой музыки и вызывается updateUI() для обновления экрана, например, после закрытия внутриигрового меню.

findViews(): Приватный метод, который находит все элементы UI (такие как ImageView для сцены, TextView для текста, Button для выбора и меню) в разметке activity\_game.xml по их уникальным ID и присваивает их соответствующим переменным класса.

setupListeners(): Приватный метод, который настраивает OnClickListener (обработчики нажатий) для текстового поля и кнопок выбора. Определяет, что произойдет при клике на каждый элемент.

updateUI(): Ключевой метод GameActivity. Он отвечает за:

Получение текущего Node из GameManager.

Установку изображения сцены (imageScene.setImageResource()).

Установку текста игры (textGame.setText()).

Управление видимостью контейнера кнопок выбора (layoutChoices.setVisibility()) и установку текста на этих кнопках, если узел имеет выбор.

Проверку, является ли текущий узел конечным, и, если да, отображение сообщения о конце игры и возврат в главное меню.

Если GameManager не может найти текущий узел (например, из-за ошибки в nodeId), updateUI() отображает сообщение об ошибке и возвращает в главное меню.

2.1.3. Node.java (Узел Сюжета)

Назначение: Node — это базовая модель данных, представляющая собой единичный, дискретный фрагмент вашей сюжетной линии или "сцену" в игре. Каждый объект Node инкапсулирует всю информацию, необходимую для отображения конкретной части истории и определения возможных переходов.

Ключевые обязанности:

Хранение данных сцены: Содержит всю необходимую информацию для одной сцены игры (уникальный ID, текст, изображение, данные о выборах и следующих узлах).

Предоставление данных: Обеспечивает доступ к своим внутренним данным для других классов (например, GameActivity и GameManager) через стандартные методы "геттеры" (например, getNodeId(), getTextResourceId()).

Важные свойства (переменные класса, используются при создании объекта Node):

nodeId (Тип: String, Пример: "startScene", "forestPath", "winEnding")

Подробное назначение: Это уникальный строковый идентификатор для каждого узла. Он служит как "имя" или "адрес" для конкретной сцены вашей истории.

Важность: nodeId является критически важным для навигации по сюжету. GameManager использует эти идентификаторы, чтобы найти нужный узел и определить, куда игра должна двигаться дальше.

Правила именования: Вы придумываете эти имена самостоятельно. Они должны быть уникальными в пределах всей игры, описательными, но лаконичными. Рекомендуется использовать camelCase (например, mainHall, darkCorridor) или snake\_case (например, main\_hall, dark\_corridor), но без пробелов и специальных символов, кроме подчеркивания.

textResourceId (Тип: int, Пример: R.string.textForestPath)

Подробное назначение: Это ссылка (ID) на строковый ресурс из файла strings.xml. Этот ресурс содержит текст, который будет отображаться в нижней части игрового экрана, описывая события, диалоги или мысли персонажа в данной сцене.

Как это работает: R.string.имя\_строки — это автоматически сгенерированная Android Studio константа, которая указывает на определенную текстовую запись, которую вы заранее определили в файлах strings.xml (см. раздел 5.2). Использование ресурсов строк позволяет легко локализовать игру на разные языки.

imageResourceId (Тип: int, Пример: R.drawable.zagl1)

Подробное назначение: Это ссылка (ID) на ресурс изображения из папки drawable/. Это изображение будет отображаться в верхней части игрового экрана для данной сцены, визуально дополняя текстовое описание.

Как это работает: R.drawable.имя\_картинки — это автоматически сгенерированная Android Studio константа, которая указывает на файл изображения, который вы поместили в папку drawable/ (см. раздел 5.2).

hasChoices (Тип: boolean, Пример: true или false)

Подробное назначение: Этот логический флаг служит индикатором того, является ли текущий узел точкой разветвления сюжета, где игроку необходимо принять решение.

Поведение:

Если true, то на игровом экране появятся две кнопки с вариантами выбора, ожидая действия игрока.

Если false, это означает, что узел является частью линейного повествования. Игрок просто читает текст и может нажать в любом месте текстового поля, чтобы перейти к следующей предопределенной сцене.

choice1TextResourceId (Тип: int, Пример: R.string.choiceGoLeft)

Подробное назначение: Это ссылка (ID) на строковый ресурс для надписи, которая будет отображаться на первой кнопке выбора.

Условие использования: Используется только если hasChoices узла равно true. Текст для этого ресурса должен быть заранее определен в strings.xml.

choice2TextResourceId (Тип: int, Пример: R.string.choiceGoRight)

Подробное назначение: Это ссылка (ID) на строковый ресурс для надписи, которая будет отображаться на второй кнопке выбора.

Условие использования: Используется только если hasChoices узла равно true. Текст для этого ресурса также должен быть заранее определен в strings.xml.

nextNodeIdNormal (Тип: String, Пример: "nextLinearScene")

Подробное назначение: Это nodeId следующего узла в последовательности истории.

Условие использования: Применяется, если у текущего узла нет выбора (hasChoices равно false). После того как игрок нажмет на текст на экране, игра автоматически перейдет к узлу, указанному этим nodeId.

nextNodeIdChoice1 (Тип: String, Пример: "pathA")

Подробное назначение: Это nodeId следующего узла, к которому игра перейдет, если игрок выберет первый вариант (нажмет на кнопку 1).

Условие использования: Применяется только если hasChoices узла равно true.

nextNodeIdChoice2 (Тип: String, Пример: "pathB")

Подробное назначение: Это nodeId следующего узла, к которому игра перейдет, если игрок выберет второй вариант (нажмет на кнопку 2).

Условие использования: Применяется только если hasChoices узла равно true.

isEndNode (Тип: boolean, Пример: true или false)

Подробное назначение: Этот логический флаг служит для обозначения того, является ли данный узел финальной точкой в сюжетной линии.

Поведение:

Если true, после того как этот узел будет отображен на экране, приложение покажет всплывающее сообщение "Конец игры!" (Toast) и автоматически вернет игрока в главное меню.

У конечного узла обычно нет дальнейших переходов, так как он завершает текущий игровой сеанс.

2.1.3. Три Типа Узлов (Конструкторы Node)

При создании нового узла в методе loadNodes() класса GameManager.java, вы будете использовать один из трех предопределенных способов создания объекта Node (так называемых "конструкторов"). Выбор конструктора зависит от того, какого типа узел вам нужен:

3. Система узлов

1. Конструктор для обычного линейного узла (без выбора):

new Node(String nodeId, int textResId, int imageResId, String nextNormalNodeId)

Когда использовать: Этот конструктор подходит для создания сцен, где игроку не нужно принимать никаких решений. Он просто читает текст, витрину событий или диалог, а затем нажимает на текстовое поле, чтобы продвинуться к следующей, заранее определенной сцене.

Параметры:

nodeId: Уникальный идентификатор этого узла.

textResId: ID строкового ресурса для текста сцены.

imageResId: ID ресурса изображения для сцены.

nextNormalNodeId: nodeId следующего узла, куда игра перейдет по умолчанию.

Пример использования: nodes.put("началоИгры", new Node("началоИгры", R.string.текстНачала, R.drawable.картинкаЗаглушка1, "перваяСценаЛеса"));

2. Конструктор для узла с выборами (развилка сюжета):

new Node(String nodeId, int textResId, int imageResId,

int choice1TextResId, int choice2TextResId,

String nextNodeIdChoice1, String nextNodeIdChoice2)

Когда использовать: Этот конструктор предназначен для создания моментов в истории, где игрок сталкивается с выбором между двумя различными вариантами, каждый из которых ведет к своей собственной ветке сюжета. На экране появятся две кнопки.

Параметры:

nodeId: Уникальный идентификатор этого узла.

textResId: ID строкового ресурса для текста сцены, предваряющего выбор.

imageResId: ID ресурса изображения для сцены.

choice1TextResId: ID строкового ресурса для текста на первой кнопке выбора.

choice2TextResId: ID строкового ресурса для текста на второй кнопке выбора.

nextNodeIdChoice1: nodeId следующего узла, если игрок выберет первый вариант.

nextNodeIdChoice2: nodeId следующего узла, если игрок выберет второй вариант.

Пример использования: nodes.put("перекрестокВЛесу", new Node("перекрестокВЛесу", R.string.текстПерекрестка, R.drawable.картинкаЗаглушка2, R.string.выборПойтиНалево, R.string.выборПойтиНаправо, "дорогаНалево", "дорогаНаправо"));

3. Конструктор для конечного узла (окончание игры):

new Node(String nodeId, int textResId, int imageResId, boolean isEndNode)

Когда использовать: Этот конструктор предназначен для обозначения любой концовки вашей истории. После того как этот узел отображается, игра завершается, и пользователь возвращается в главное меню.

Параметры:

nodeId: Уникальный идентификатор этого конечного узла.

textResId: ID строкового ресурса для текста, описывающего концовку (например, "Вы нашли сокровища!", "Вас съел монстр!").

imageResId: ID ресурса изображения для финальной сцены.

isEndNode: Всегда должно быть true для этого конструктора, чтобы система знала, что это конец игры.

Пример использования: nodes.put("победныйКонец", new Node("победныйКонец", R.string.текстПобеды, R.drawable.картинкаПобеды, true));

3.2. "Мозг" Сюжета: GameManager.java

Класс GameManager — это центральная часть вашей игры. Он отвечает за хранение всего сюжета и управление игровым процессом. Здесь вы будете тратить основное время на построение вашей истории.

3.2.1. Метод loadNodes(): Где Живет Ваша История

Вся ваша сюжетная линия, состоящая из взаимосвязанных Node, определена внутри приватного метода loadNodes() в классе GameManager.java.

Как это работает: При инициализации GameManager (когда игра запускается или сбрасывается), метод loadNodes() вызывается один раз. Он создает каждый Node в вашей истории и добавляет его в специальную коллекцию Map под названием nodes. Эта Map позволяет игре быстро находить любой узел по его уникальному nodeId.

Где найти: Откройте файл GameManager.java в Android Studio. Прокрутите вниз до метода private void loadNodes() { }.

Ваша задача: Внутри этого метода вы будете создавать новые узлы и добавлять их в коллекцию nodes с помощью метода nodes.put("вашNodeId", новыйОбъектNode);.

3.2.2. Пошаговое Добавление Нового Узла Сюжета

Чтобы добавить новый фрагмент истории в игру, необходимо выполнить ряд скоординированных шагов, затрагивающих несколько файлов. Следуйте этой последовательности внимательно, чтобы избежать ошибок.

3.2.2.1. Шаг 1: Планирование Нового Узла

Прежде чем начать писать код, четко определитесь с содержимым и поведением вашей новой сцены:

Содержание сцены:

Какой будет основной текст, который игрок прочитает в этой сцене? Будет ли это описание места, диалог, или внутренние мысли персонажа?

Какое изображение вы хотите, чтобы сопровождало эту сцену?

Взаимодействие игрока:

Это будет линейный узел? То есть, игрок просто читает текст и нажимает в любом месте текстового поля, чтобы перейти к следующей, заранее определенной сцене?

Или это будет узел с выбором? То есть, игроку будет предложено сделать выбор между двумя различными вариантами (с помощью кнопок), и каждый выбор приведет к своей собственной ветке сюжета?

Логика перехода:

Если это линейный узел, то какой nodeId будет у следующего узла, куда история автоматически перейдет?

Если это узел с выбором, то какие nodeId будут у двух разных узлов, куда приведет каждый из двух вариантов выбора?

Конец игры? Является ли этот узел логическим завершением одной из веток вашей истории? (Если да, установите isEndNode в true).

Уникальный Идентификатор: Придумайте уникальный и осмысленный nodeId для этого нового узла. Это должно быть короткое, но понятное имя, например: ancientTemple, hiddenPassage, finalBossFight, monsterLair.

3.2.2.2. Шаг 2: Добавление Текста Узла и Выборов в strings.xml

Весь текстовый контент вашей игры (сюжетные описания, надписи на кнопках меню и выборов, сообщения) должен быть определен в специальных XML-файлах strings.xml. Это необходимо для корректной работы системы локализации (поддержки разных языков).

1. Откройте основной файл для русского языка:

В Android Studio, в панели "Project", перейдите по пути: app -> src -> main -> res -> values -> strings.xml.

Добавьте новую строковую запись для основного текста вашего нового узла. Придумайте уникальное name (имя ресурса) для этой строки. Это имя должно быть в camelCase или snake\_case (например, textAncientTemple) и не содержать пробелов.

В app/src/main/res/values/strings.xml

<string name="textAncientTemple">Вы обнаружили древний храм, скрытый глубоко в джунглях. Вход завален огромными валунами.</string>

Если ваш новый узел имеет выбор (то есть, hasChoices будет true): Вам также необходимо добавить отдельные строковые записи для текста, который будет отображаться на каждой из двух кнопок выбора. Придумайте уникальные name для каждого варианта выбора.

В app/src/main/res/values/strings.xml

<string name="choiceClearEntrance">Попытаться расчистить вход</string>

<string name="choiceSearchPerimeter">Осмотреть периметр храма</string>

2. ОЧЕНЬ ВАЖНО: Добавьте переводы в другие языковые файлы:

Для каждого языка, который поддерживает ваша игра (например, английский, французский и т.д.), вы должны открыть соответствующий файл strings.xml. Например, для английского это app/src/main/res/values-en/strings.xml.

В этот файл вам необходимо скопировать те же самые записи <string name="..." (т.е. name должно быть идентичным: textAncientTemple, choiceClearEntrance, choiceSearchPerimeter), но перевести текст, который находится между открывающим (<string>) и закрывающим (</string>) тегами, на соответствующий язык.

В app/src/main/res/values-en/strings.xml

<string name="textAncientTemple">You discovered an ancient temple, hidden deep within the jungle. The entrance is blocked by huge boulders.</string>

<string name="choiceClearEntrance">Attempt to clear the entrance</string>

<string name="choiceSearchPerimeter">Search the temple perimeter</string>

Критическое замечание: Если вы забудете добавить текст для какого-либо языка, или допустите опечатку в name="..." в одном из языковых файлов, это может привести к тому, что приложение вылетит с ошибкой или будет отображать пустой текст на этом языке, когда он будет выбран! Всегда делайте двойную проверку.

3.2.2.3. Шаг 3: Добавление Изображения Узла в drawable/

Каждая сцена в вашей игре может иметь свое уникальное визуальное сопровождение. Эти изображения хранятся в папке drawable/.

1. Подготовьте ваш графический файл: Выберите изображение (в формате .png, .jpg или .webp), которое вы хотите использовать для вашего нового узла.

2. Правильное именование файла:

Переименуйте файл так, чтобы его имя состояло только из строчных букв, цифр и символа подчеркивания (\_).

Избегайте пробелов, дефисов (-), заглавных букв и любых других специальных символов в имени файла.

Примеры правильных имен: ancient\_temple.png, cave\_interior.jpg, jungle\_path\_day.webp.

Примеры неправильных имен: Ancient Temple.png, cave-interior.jpg, Jungle Path Day.webp.

3. Размещение файла:

В Android Studio, в панели "Project", перейдите по пути: app -> src -> main -> res -> drawable.

Просто скопируйте ваш переименованный файл изображения прямов эту папку drawable/.

4. Автоматическая регистрация: После того как вы скопируете файл, Android Studio автоматически распознает его и сгенерирует для него уникальный числовой идентификатор (ресурс ID), который вы сможете использовать в коде Java в формате R.drawable.ваше\_имя\_файла.

3.2.2.4. Шаг 4: Создание и Добавление Нового Объекта Node в GameManager.java

Теперь, когда вы подготовили текст и изображение, вы можете создать сам объект Node, который будет представлять вашу новую сцену, и добавить его в сюжетную карту игры.

1. Откройте файл GameManager.java в Android Studio.

2. Найдите приватный метод private void loadNodes() { }. Внутри этого метода вы увидите строки nodes.put(...), которые определяют текущую сюжетную линию.

3. Добавьте новую строку, которая создает новый объект Node и помещает его в коллекцию nodes. Используйте один из трех типов конструкторов Node, который соответствует вашему плану (см. раздел 3.1.2 "Три Типа Узлов (Конструкторы Node)").

Пример 1: Добавление нового линейного узла (без выбора)

Игрок просто читает текст и нажимает на него, чтобы продолжить.

Добавьте эту строку в метод loadNodes()

nodes.put("nodeAncientTemple", new Node("nodeAncientTemple", R.string.textAncientTemple, R.drawable.ancient\_temple, "nodeInsideTemple"));

Объяснение:

- "nodeAncientTemple": Это уникальный nodeId для вашей новой сцены.

- R.string.textAncientTemple: Ссылка на текст, который будет показан.

- R.drawable.ancient\_temple: Ссылка на изображение сцены.

- "nodeInsideTemple": Это nodeId следующего узла, куда игра перейдет по умолчанию. Убедитесь, что "nodeInsideTemple"

либо уже существует, либо вы создадите его позже по тем же правилам.

Пример 2: Добавление нового узла с выбором

Игрок должен выбрать один из двух вариантов, и каждый вариант ведет к своей ветке сюжета.

Добавьте эту строку в метод loadNodes()

nodes.put("nodeAncientTemple", new Node("nodeAncientTemple", R.string.textAncientTemple, R.drawable.ancient\_temple,

R.string.choiceClearEntrance, R.string.choiceSearchPerimeter,

"nodeClearedEntrance", "nodePerimeterSearched"));

Объяснение:

- "nodeAncientTemple": Уникальный nodeId для вашей новой сцены.

- R.string.textAncientTemple: Текст сцены.

- R.drawable.ancient\_temple: Изображение сцены.

- R.string.choiceClearEntrance: Текст для первой кнопки выбора.

- R.string.choiceSearchPerimeter: Текст для второй кнопки выбора.

- "nodeClearedEntrance": nodeId узла, куда игра пойдет, если игрок выберет первую кнопку.

- "nodePerimeterSearched": nodeId узла, куда игра пойдет, если игрок выберет вторую кнопку.

Убедитесь, что эти следующие узлы либо уже существуют, либо будут созданы вами.

Пример 3: Добавление нового конечного узла (завершение игры)

Этот узел обозначает одну из концовок вашей истории.

Добавьте эту строку в метод loadNodes()

nodes.put("nodeGameWonTemple", new Node("nodeGameWonTemple", R.string.textNodeWin, R.drawable.temple\_victory\_image, true));

Объяснение:

- "nodeGameWonTemple": Уникальный nodeId для этой концовки.

- R.string.textNodeWin: Текст, описывающий эту концовку (можно использовать уже существующую строку "textNodeWin" из strings.xml,

или создать новую, если нужна уникальная фраза для этой конкретной концовки).

- R.drawable.temple\_victory\_image: Изображение для этой финальной сцены.

- true: Этот флаг указывает системе, что данный узел является завершением игры.

3.2.2.5. Шаг 5: Связывание Нового Узла с Предыдущим (Обновление Ссылок)

Это ОДИН ИЗ САМЫХ КРИТИЧЕСКИ ВАЖНЫХ ШАГОВ, и его часто забывают, что приводит к ошибкам "узел не найден"! После того как вы создали новый узел, вам нужно сообщить игре, как именно к нему можно добраться из предыдущих сцен.

Определите, какой (или какие) из уже существующих узлов должен теперь вести к вашему новому узлу.

Откройте GameManager.java и найдите nodes.put(...) строку для этого предыдущего узла в методе loadNodes().

Измените соответствующее поле (nextNodeIdNormal, nextNodeIdChoice1, или nextNodeIdChoice2) этого предыдущего узла, чтобы оно теперь ссылалось на nodeId вашего нового узла.

Пример Сценария: Допустим, вы только что создали узел "nodeAncientTemple" (как в Шаге 4.2.2.4). Теперь вы хотите, чтобы из узла "nodeForestPath" (который уже существует в вашей истории) один из вариантов выбора вел к вашему новому "nodeAncientTemple".

Найдите в loadNodes() строку для nodeForestPath:

Предположим, что nodeForestPath выглядел так до изменений:

nodes.put("nodeForestPath", new Node("nodeForestPath", R.string.textNodeForestPath, R.drawable.zagl1,

R.string.choiceGoLeft, R.string.choiceGoRight, "nodeLeftPath", "nodeRightPath"));

Измените эту строку, чтобы один из его переходов (nextNodeIdChoice1 или nextNodeIdChoice2) указывал на ваш новый узел ("nodeAncientTemple"). Например, если "пойти налево" должно вести в храм:

ИЗМЕНЕННАЯ строка для nodeForestPath: теперь "nodeAncientTemple" доступен по выбору 1 ("choiceGoLeft")

nodes.put("nodeForestPath", new Node("nodeForestPath", R.string.textNodeForestPath, R.drawable.zagl1,

R.string.choiceGoLeft, R.string.choiceGoRight, "nodeAncientTemple", "nodeRightPath")); <-- Здесь изменено на "nodeAncientTemple"

Двойная проверка всех ссылок: После каждого добавления или изменения узла, критически важно убедиться, что каждый nodeId, на который вы ссылаетесь (например, "nodeAncientTemple", "nodeRightPath", "nodeWin"), действительно существует как ключ в вашей карте nodes (то есть, есть соответствующая строка nodes.put("ТОЧНО\_ТАКОЙ\_ЖЕ\_ID", ...) где-то в loadNodes()) и что он написан без единой опечатки. Неправильная ссылка приведет к ошибке "узел не найден" (errorGameNodeNotFound) в игре.

3.3. Особенности Системы Узлов

3.3.1. Разветвления и Линейные Участки

Разветвления: Создаются с использованием конструктора Node с выборами (где hasChoices равно true). Это ключевой элемент интерактивных историй, позволяющий игроку влиять на развитие сюжета и создавать несколько возможных путей.

Линейные участки: Представляют собой последовательность узлов, где hasChoices равно false. Игрок просто читает текст и нажимает на текстовое поле, чтобы перейти к следующему предопределенному узлу. Используйте линейные участки для описания событий, небольших диалогов, вступлений или концовок, где прямого выбора не требуется.

3.3.2. Конечные Узлы Игры

Любой узел, для которого в его конструкторе isEndNode установлено в true (или используется специальный конструктор для конечных узлов), является финальной точкой в игре.

После того как такой узел отображается на экране, система автоматически выводит короткое сообщение Toast "Конец игры!" (R.string.textGameEnd) и затем возвращает игрока в главное меню (MainActivity).

Вы можете создавать несколько конечных узлов, чтобы реализовать различные концовки вашей истории: победные, драматические, различные варианты поражений и т.д.

3.3.3. Что Происходит при Ошибке Ссылки на Узел

Если по какой-либо причине (nodeId написан с опечаткой, или вы забыли добавить узел в loadNodes(), или ссылка ведет на несуществующий узел), GameManager не сможет найти следующий узел, на который он ссылается. В этом случае произойдет следующее:

Всплывающее сообщение: На игровом экране (GameActivity) появится короткое всплывающее сообщение Toast с текстом: "Ошибка: узел игры не найден!" (R.string.errorGameNodeNotFound).

Возврат в меню: После показа сообщения об ошибке приложение автоматически закроет GameActivity и вернется на экран главного меню (MainActivity), чтобы предотвратить дальнейшие ошибки и дать игроку возможность начать заново.

Отладочные сообщения: В Logcat (окно отладки в Android Studio) вы, скорее всего, увидите более детальные сообщения об ошибке (например, NullPointerException или сообщение о том, что ключ не найден в Map). Это очень ценно для отладки и определения точного места проблемы.

4. Управление Музыкой в Игре

Ваш проект включает в себя систему управления фоновой музыкой, реализованную через класс MusicManager.java. В игре используется только одна фоновая композиция, которая плавно проигрывается на протяжении всего приложения.

4.1. Подготовка и Размещение Музыкального Файла (zaglmusic.mp3)

Для того чтобы приложение могло проигрывать фоновую музыку, файл с ней должен быть правильно подготовлен и размещен:

1. Формат файла: Музыкальный файл должен быть в формате .mp3 или .ogg. Оба формата широко поддерживаются Android. Рекомендуется .mp3 для максимальной совместимости.

2. Имя файла: Ваш музыкальный файл должен быть назван zaglmusic (без указания расширения, например, zaglmusic.mp3). Это критически важно, так как код приложения настроен на поиск файла именно с таким именем.

3. Размещение:

В Android Studio, в панели "Project", перейдите по пути: app -> src -> main -> res.

Убедитесь, что существует папка raw: Если папки raw нет, создайте ее: кликните правой кнопкой мыши по res -> New -> Android Resource Directory. В выпадающем списке Resource type выберите raw. Нажмите OK.

Скопируйте ваш файл zaglmusic.mp3 (или .ogg) прямо в папку app/src/main/res/raw/.

4. Автоматическая регистрация: После копирования файла, Android Studio автоматически распознает его и сгенерирует для него уникальный числовой идентификатор, который вы сможете использовать в Java-коде в формате R.raw.zaglmusic.

4.2. Функции Класса MusicManager.java

Класс MusicManager является статическим, что означает, что он существует в единственном экземпляре на протяжении всего жизненного цикла приложения, позволяя музыке продолжать играть при переключении между Activity. Вам, как правило, не нужно вручную вызывать большинство его методов, так как они уже интегрированы в MainActivity и GameActivity для автоматического управления воспроизведением.

4.2.1. Инициализация и Запуск

Инициализация: Метод MusicManager.init(getApplicationContext()) вызывается только один раз — при самом первом запуске приложения (в методе onCreate() класса MainActivity). Этот метод отвечает за:

Создание объекта MediaPlayer.

Загрузку музыкального файла zaglmusic.mp3 из ресурсов приложения.

Настройку MediaPlayer на зацикливание воспроизведения (mediaPlayer.setLooping(true);), чтобы музыка играла непрерывно.

Загрузку сохраненного уровня громкости.

Запуск/Возобновление воспроизведения: Метод MusicManager.playMusic() вызывается в следующих случаях:

При запуске MainActivity (экрана главного меню).

При запуске GameActivity (экрана игрового процесса).

При возвращении на эти экраны (например, после закрытия диалога настроек или внутриигрового меню, или после того, как приложение было свернуто и снова развернуто).

Это обеспечивает плавное и непрерывное воспроизведение фоновой музыки, поддерживая атмосферу игры.

4.2.2. Пауза и Остановка

Пауза: Метод MusicManager.pauseMusic() ставит текущую музыкальную композицию на паузу. В текущей реализации проекта этот метод не используется напрямую, но он предусмотрен для возможного расширения функционала, например, для создания кнопки "Пауза" в игре.

Полная Остановка: Метод MusicManager.stopMusic() вызывается только в одном случае — при полном выходе из приложения (то есть, когда пользователь нажимает кнопку "Выход" в главном меню). Этот метод:

Полностью прекращает воспроизведение музыки.

Освобождает все системные ресурсы, которые были заняты объектом MediaPlayer. Это критически важно для предотвращения утечек памяти и обеспечения стабильной работы приложения.

4.2.3. Зацикливание Музыки

Как упоминалось, музыкальная композиция, загруженная через MusicManager, автоматически настроена на зацикливание. Это означает, что как только трек достигает своего конца, он автоматически начинается заново с самого начала, создавая ощущение непрерывного фонового сопровождения без пауз. Эта настройка задана строкой mediaPlayer.setLooping(true); в методе MusicManager.init().

4.3. Регулировка Громкости Музыки через Окно Настроек

Игроки могут управлять громкостью фоновой музыки прямо из приложения.

4.3.1. Как Регулировать Громкость

Доступ к настройкам: Окно "Настройки" доступно из главного меню (MainActivity). Для этого нажмите на кнопку "Настройки".

Ползунок громкости: В открывшемся окне настроек вы увидите элемент пользовательского интерфейса, называемый "ползунок" (SeekBar), помеченный как "Громкость музыки".

Регулировка: Перемещая этот ползунок влево или вправо, вы можете динамически (в реальном времени) увеличивать или уменьшать громкость фоновой музыки.

4.3.2. Как Работает Сохранение Громкости

Мгновенное применение: Все изменения громкости, сделанные с помощью ползунка, применяются к воспроизводимой музыке немедленно.

Постоянное сохранение: Выбранный уровень громкости не сбрасывается после закрытия приложения. MusicManager автоматически сохраняет последний установленный уровень громкости в постоянной памяти устройства (SharedPreferences). Это означает, что при следующем запуске игры музыка будет проигрываться с тем же уровнем громкости, который был установлен в прошлый раз.

4.4. Как Заменить Фоновую Музыку на Другую

Если вы хотите использовать другую музыкальную композицию для вашего квеста, процесс замены очень прост:

1. Подготовьте новый музыкальный файл:

Убедитесь, что ваш новый музыкальный файл находится в поддерживаемом формате: .mp3 или .ogg.

Оптимизируйте его размер, чтобы не увеличивать объем приложения чрезмерно.

2. Переименуйте файл:

Очень важно: ваш новый музыкальный файл должен быть назван zaglmusic.mp3 (или zaglmusic.ogg, в зависимости от его формата). Код приложения жестко ссылается на это имя.

3. Замените файл в проекте:

В Android Studio, в панели "Project", перейдите по пути: app -> src -> main -> res -> raw.

Найдите и удалите существующий файл zaglmusic.mp3 (или .ogg).

Затем скопируйте ваш новый, переименованный файл zaglmusic.mp3 (или .ogg) прямо в эту папку raw.

4. Перезапустите приложение: Запустите ваше приложение на эмуляторе или реальном устройстве. Новая музыкальная композиция должна начать проигрываться в качестве фоновой.

5. Примечание (не рекомендуется): Если по какой-то причине вы настаиваете на использовании другого имени файла для музыки (например, my\_new\_epic\_song.mp3), вам придется вручную изменить строку кода, которая загружает музыку в MusicManager.java. В методе MusicManager.init() измените R.raw.zaglmusic на R.raw.ваше\_новое\_имя\_файла (например, R.raw.my\_new\_epic\_song). Однако, следование стандартному имени zaglmusic значительно проще.

5. Локализация

Ваш проект изначально настроен на поддержку русского и английского языков. Вся текстовая информация, отображаемая в приложении (сюжетные описания, надписи на кнопках, сообщения интерфейса), может быть легко переведена на дополнительные языки.

5.1. Принцип Работы Локализации через strings.xml

Android использует мощную систему ресурсов для управления локализацией (переводом).

Файлы strings.xml: Все текстовые фразы хранятся в специальных XML-файлах, которые называются strings.xml.

Имена ресурсов (name): Каждая текстовая фраза внутри strings.xml имеет свой уникальный идентификатор, заданный атрибутом name:

<string name="buttonStartGame">Начать игру</string>

Здесь buttonStartGame — это уникальное имя ресурса, а "Начать игру" — это текст для русского языка.

Папки языков: Для каждого поддерживаемого языка создается своя специальная папка внутри res/:

res/values/: Для языка по умолчанию (в нашем случае, русского).

res/values-en/: Для английского языка.

res/values-fr/: (Пример) Для французского языка.

В каждой такой папке находится свой файл strings.xml, содержащий одни и те же name атрибуты, но с текстом, переведенным на соответствующий язык.

Автоматический выбор: Когда приложение запускается, система Android автоматически определяет текущий язык устройства (или использует язык, выбранный пользователем в настройках игры). Затем она загружает текстовые ресурсы из файла strings.xml в папке, соответствующей этому языку. Если для текущего языка нет специального файла, используется файл из папки res/values/ (язык по умолчанию).

5.2. Файлы strings.xml для Разных Языков

app/src/main/res/values/strings.xml:

Содержимое: Этот файл содержит все текстовые фразы, используемые в приложении, на русском языке. Этот язык является языком по умолчанию для вашего приложения. Все элементы UI, которые не имеют специфического перевода для текущего языка устройства, будут отображать текст из этого файла.

app/src/main/res/values-en/strings.xml:

Содержимое: Эта папка и файл содержат те же самые текстовые фразы (имеющие те же name атрибуты), но с текстом, переведенным на английский язык.

Критическое требование: Атрибуты name="..." для каждой строки ДОЛЖНЫ БЫТЬ АБСОЛЮТНО ИДЕНТИЧНЫМИво всех файлах strings.xml для разных языков. Отличаться должен только сам текстовый контент, который находится между открывающим (<string>) и закрывающим (</string>) тегами. Это позволяет системе Android корректно "сопоставлять" тексты на разных языках.

5.3. Пошаговое Добавление Нового Языка в Игру

Добавление поддержки нового языка (например, немецкого) для вашего приложения — это простой, но важный процесс:

1. Создайте новую папку для ресурсов с кодом языка:

В Android Studio, в панели "Project" (обычно расположенной слева), кликните правой кнопкой мыши по папке app/src/main/res/.

В контекстном меню выберите New -> Android Resource Directory.

В появившемся окне "New Resource Directory":

В выпадающем списке Resource type: выберите values.

В выпадающем списке Locale: найдите и выберите код нужного вам языка, например, de - German. Android Studio автоматически добавит к имени папки префикс values- и код языка (например, values-de).

Нажмите OK. Android Studio создаст новую папку по указанному пути, например, app/src/main/res/values-de/.

2. Скопируйте базовый файл strings.xml:

Найдите файл app/src/main/res/values/strings.xml (это ваша текущая русская версия).

Скопируйте его (правой кнопкой мыши -> Copy или используйте Ctrl+C).

Вставьте скопированный файл в только что созданную папку app/src/main/res/values-de/ (правой кнопкой мыши по values-de -> Paste или используйте Ctrl+V). Убедитесь, что имя файла осталось strings.xml.

3. Переведите все строки в новом файле:

Откройте файл app/src/main/res/values-de/strings.xml.

Внутри этого файла вам необходимо перевести каждый текстовый фрагмент (т.е. тот текст, который находится между открывающим (<string>) и закрывающим (</string>) тегами) на немецкий язык.

ПОМНИТЕ: НЕ МЕНЯЙТЕ атрибуты name="..." для строк! Они должны быть абсолютно идентичными тем, что используются в русском и английском strings.xml. Меняйте только фактическое текстовое содержимое.

Пример:

В app/src/main/res/values-de/strings.xml

<string name="buttonStartGame">Spiel starten</string>

<string name="buttonContinueGame">Fortsetzen</string>

<string name="buttonSettings">Einstellungen</string>

<string name="textNodeStart">Sie stehen an der Schwelle eines mysteriösen Abenteuers. Machen Sie Ihren ersten Schritt.</string>

и так далее для всех остальных строк

4. Готово! После того как вы полностью переведете все строки и сохраните файл, ваше приложение теперь будет поддерживать немецкий язык. Пользователи смогут выбрать его в меню настроек.

5.4. Изменение Текста для Существующего Языка

Чтобы изменить уже существующий текстовый фрагмент для определенного языка:

1. Определите целевой файл strings.xml:

Если вы хотите изменить текст для русского языка, откройте файл app/src/main/res/values/strings.xml.

Если вы хотите изменить текст для английского языка, откройте файл app/src/main/res/values-en/strings.xml.

Если вы работаете с другим добавленным языком, откройте соответствующий файл в папке app/src/main/res/values-КОД\_ЯЗЫКА/strings.xml (например, values-de/strings.xml для немецкого).

2. Найдите нужную строку: В открытом файле strings.xml найдите строковую запись по ее уникальному name атрибуту. Например, если вы хотите изменить текст кнопки "Начать игру", найдите:

<string name="buttonStartGame">Начать игру</string>

3. Измените текст: Измените только текст, который находится между открывающим (<string>) и закрывающим (</string>) тегами. Ни в коем случае не меняйте name="..."!

<string name="buttonStartGame">Начать Новое Приключение!</string> (Текст изменен на более воодушевляющий)

4. Сохраните файл.

5. Применение изменений: Изменения вступят в силу, когда соответствующая Activity (экран) будет загружена или перезагружена. Например, если вы изменили текст кнопки в MainActivity, вы увидите изменение после перезапуска приложения или после переключения языка туда-обратно через настройки (если это был не текущий активный язык).

5.5. Как Происходит Переключение Языка в Приложении

Процесс переключения языка в вашем приложении автоматизирован и происходит плавно, но требует перезагрузки главного меню для полного применения изменений:

1. Выбор языка в "Настройках": Когда игрок выбирает другой язык в окне "Настройки" (SettingsDialog) — например, переключается с русского на английский — этот выбор (код языка, например, "en") сохраняется в постоянную память устройства с помощью SharedPreferences.

2. Уведомление MainActivity: SettingsDialog закрывается, но перед этим он уведомляет MainActivity (главное меню) о том, что язык был изменен. Это происходит через специальный механизм "слушателя" (SettingsDialog.SettingsDialogListener), который MainActivity реализует.

3. Перезагрузка MainActivity: MainActivity, получив уведомление об изменении языка, вызывает на себе метод recreate(). Этот метод является стандартной функцией Android, которая приводит к тому, что MainActivity полностью перезагружается (по сути, закрывается и немедленно открывается заново).

4. Применение нового языка: В самом начале метода onCreate() класса MainActivity (который вызывается при перезагрузке) приложение считывает сохраненный язык из SharedPreferences и устанавливает его для всего приложения.

5. Обновление UI: Поскольку MainActivity перезагружается, все ее элементы пользовательского интерфейса (кнопки, заголовки) и любые другие текстовые компоненты на GameActivity (если вы перейдете в игру) заново загружают свои тексты из правильного языкового файла strings.xml, соответствующего выбранному языку. Таким образом, весь интерфейс обновляется.

6. Настройка Внешнего Вида Пользовательского Интерфейса (UI)

Вы можете значительно изменить внешний вид вашего приложения, настраивая цвета кнопок, общую тему, размеры элементов и фоновые изображения. Все эти изменения производятся путем редактирования XML-файлов разметки и файлов стилей.

6.1. Обзор Файлов Разметки (.xml в папке layout/)

Каждый отдельный экран или всплывающее диалоговое окно в вашем приложении имеет свой собственный файл разметки, который описывает расположение и свойства всех его элементов UI:

activity\_main.xml: Определяет внешний вид главного меню приложения.

activity\_game.xml: Определяет внешний вид основного игрового экрана (изображение сцены, текстовое поле, кнопки выбора, кнопка меню).

dialog\_settings.xml: Определяет внешний вид всплывающего окна настроек.

dialog\_ingame\_menu.xml: Определяет внешний вид всплывающего внутриигрового меню.

Вы будете напрямую редактировать эти XML-файлы, чтобы настроить расположение, размеры и свойства элементов.

6.2. Изменение Цвета Кнопок

Цвет кнопок можно изменить двумя основными способами: глобально (для большинства кнопок сразу через тему) или индивидуально (для конкретной кнопки).

6.2.1. Глобальное Изменение Цвета Кнопок (Через Тему)

Этот метод позволяет изменить цвет большинства стандартных кнопок в вашем приложении, которые наследуют стили из вашей общей темы.

1. Определите или измените цвета в app/src/main/res/values/colors.xml:

Откройте этот файл. Здесь хранятся все цвета, которые используются в вашем приложении. Каждый цвет имеет уникальное имя (например, purple\_500, teal\_200, black).

Вы можете изменить шестнадцатеричные значения существующих цветов или добавить свои собственные. Цвет задается в формате #AARRGGBB (где AA - прозрачность, RR - красный, GG - зеленый, BB - синий компонент) или #RRGGBB (если прозрачность не указана, по умолчанию она будет 100%, т.е. FF).

Пример добавления новых цветов:

<resources>

(другие стандартные цвета)

<color name="myCustomButtonColor">#FFD4AF37</color> (Золотой цвет для фона кнопок)

<color name="myCustomButtonTextColor">#FF000000</color> (Черный цвет для текста на кнопках)

</resources>

2. Примените цвета в app/src/main/res/values/themes.xml:

Откройте этот файл. Он определяет общий стиль и цветовую палитру для всего вашего приложения.

Найдите элементы <item name="colorSecondary"> и <item name="colorOnSecondary">. Эти атрибуты часто используются для задания акцентных цветов, которые применяются ко многим виджетам по умолчанию, включая кнопки.

colorSecondary: Определяет основной цвет для акцентных элементов, часто это фон кнопок.

colorOnSecondary: Определяет цвет текста, который хорошо читается на фоне colorSecondary.

Измените их ссылки на цвета, которые вы определили в colors.xml:

<style name="Theme.PatternForQuest" parent="Theme.MaterialComponents.DayNight.DarkActionBar">

Основной цвет бренда (для верхней панели, иконок и т.д.)

<item name="colorPrimary">@color/purple\_500</item>

<item name="colorPrimaryVariant">@color/purple\_700</item>

<item name="colorOnPrimary">@color/white</item>

Вторичный акцентный цвет бренда (часто влияет на кнопки)

<item name="colorSecondary">@color/myCustomButtonColor</item> (Теперь фон кнопок будет золотым)

<item name="colorSecondaryVariant">@color/myCustomButtonColor</item> (Можно использовать тот же цвет или более темный/светлый оттенок золотого)

<item name="colorOnSecondary">@color/myCustomButtonTextColor</item> (Цвет текста на кнопках будет черным)

Цвет строки состояния (верхняя полоса с часами и уведомлениями)

<item name="android:statusBarColor">?attr/colorPrimaryVariant</item>

Дополнительные настройки темы здесь

</style>

3. Сохраните все измененные файлы и перезапустите приложение. Большинство кнопок в вашем приложении (те, которые не имеют индивидуально заданного цвета) должны изменить свой цвет в соответствии с новой темой.

6.2.2. Индивидуальное Изменение Цвета Кнопки (В XML-Разметке)

Если вам нужно задать определенный цвет только для одной конкретной кнопки (или нескольких), переопределив глобальную тему:

1. Откройте файл разметки (.xml) для экрана, где находится нужная кнопка. Например, для кнопок главного меню это activity\_main.xml, а для кнопок выбора в игре — activity\_game.xml.

2. Найдите соответствующий тег <Button> для целевой кнопки.

3. Внутри этого тега <Button>, добавьте или измените следующие атрибуты:

android:backgroundTint: Задает цвет фона кнопки.

android:textColor: Задает цвет текста, отображаемого на кнопке.

<Button

android:id="@+id/buttonStart"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="@string/buttonStartGame"

android:layout\_marginBottom="8dp"

android:backgroundTint="#FFCC0000" (Пример: красный цвет фона кнопки (шестнадцатеричный код))

android:textColor="@color/white" /> (Пример: белый цвет текста (ссылка на цвет из colors.xml))

Вы можете использовать как прямые шестнадцатеричные коды цветов (например, #FFCC0000), так и ссылки на цвета, определенные в colors.xml (например, @color/myCustomColor).

4. Сохраните файл. Изменения будут видны сразу же при просмотре макета в Android Studio и после запуска приложения.

6.3. Изменение Общей Темы Приложения (Светлая/Темная, Стиль)

Общая тема вашего приложения, определяющая его базовый внешний вид и поведение по умолчанию, задается в файле app/src/main/res/values/themes.xml.

<style name="Theme.PatternForQuest" parent="Theme.MaterialComponents.DayNight.DarkActionBar">

(определения цветов и других атрибутов)

</style>

Атрибут parent: Этот атрибут указывает на "родительскую" тему, от которой наследуются все стили и поведение вашей темы. Изменение parent позволяет кардинально изменить внешний вид приложения.

"Theme.MaterialComponents.DayNight.DarkActionBar": Это тема из библиотеки Material Design.

DayNight: Означает, что тема будет автоматически переключаться между светлым и темным режимом в зависимости от текущих системных настроек Android пользователя (если пользователь выбрал темную тему в настройках своего телефона, ваше приложение тоже станет темным).

DarkActionBar: Указывает, что верхняя панель приложения (Action Bar) будет темной.

Для изменения общей темы (примеры):

Если вы хотите, чтобы приложение всегда было светлым и не имело верхней панели (Action Bar):

Измените parent на "Theme.MaterialComponents.Light.NoActionBar".

Если вы хотите, чтобы приложение всегда было темным и не имело верхней панели (Action Bar):

Измените parent на "Theme.MaterialComponents.Dark.NoActionBar".

Исследование других тем: В Android Studio вы можете начать набирать "Theme.MaterialComponents." после parent= и IDE предложит другие доступные варианты тем из библиотеки Material Design. Попробуйте разные варианты, чтобы найти тот, который вам больше всего нравится.

Другие атрибуты темы: В этом же файле themes.xml вы можете изменять множество других атрибутов, которые влияют на общий вид:

android:statusBarColor: Определяет цвет верхней полосы уведомлений (статус-бара) на устройстве.

android:navigationBarColor: Определяет цвет нижней навигационной панели (для устройств с наэкранными кнопками).

android:windowBackground: Определяет цвет фона окна приложения по умолчанию.

textAppearanceHeadline1, textAppearanceBody1 и т.д.: Эти атрибуты позволяют настроить внешний вид типографики (шрифты, размеры текста, цвета) для различных стилей текста в приложении.

6.4. Настройка Размеров Элементов UI

Размеры элементов пользовательского интерфейса, их расположение и отступы контролируются в XML-файлах разметки (.xml).

6.4.1. Единицы Измерения (dp, sp)

Очень важно использовать правильные единицы измерения для размеров и отступов в Android, чтобы ваш UI выглядел корректно на устройствах с разными экранами:

dp (Density-independent pixels - пиксели, независимые от плотности):

Рекомендуется для всех размеров (ширина, высота, отступы, границы).

1dp приблизительно равен 1 пикселю на экране с плотностью 160 dpi.

Использование dp гарантирует, что ваш UI будет выглядеть примерно одинаково по физическому размеру на устройствах с разной плотностью пикселей и разрешением. Android автоматически масштабирует dp в зависимости от плотности экрана.

sp (Scale-independent pixels - пиксели, независимые от масштаба):

Используется ТОЛЬКО для размера текста (android:textSize).

sp масштабируются не только в зависимости от плотности экрана, но и от пользовательских настроек размера шрифта в системе Android. Это очень важно для доступности, так как пользователи с ослабленным зрением могут увеличить системный шрифт, и ваш текст также увеличится, оставаясь читаемым.

6.4.2. Изменение Процентного Соотношения Изображения и Текста на Игровом Экране

На игровом экране (activity\_game.xml) изображение сцены (imageScene) и текстовое поле (textGame) делят общую высоту экрана. Их размеры заданы в процентах от высоты родительского контейнера.

imageScene (Изображение сцены):

<ImageView

android:id="@+id/imageScene"

android:layout\_width="0dp"

android:layout\_height="0dp"

app:layout\_constraintHeight\_percent="0.7" (Текущее значение: 70% высоты экрана)

/>

Вы можете изменить значение 0.7 на другое:

Например, 0.6 (60% высоты), чтобы изображение занимало меньше места.

Или 0.8 (80% высоты), чтобы оно было больше.

textGame (Текст игры):

<TextView

android:id="@+id/textGame"

android:layout\_width="0dp"

android:layout\_height="0dp"

app:layout\_constraintHeight\_percent="0.3" (Текущее значение: 30% высоты экрана)

/>

Вы можете изменить значение 0.3 на другое:

Например, 0.4 (40% высоты), чтобы текстовое поле было больше.

Или 0.2 (20% высоты), чтобы оно было меньше.

Критическое правило: Сумма процентов для imageScene и textGame всегда должна быть равна 1.0 (что соответствует 100% высоты экрана). Это гарантирует, что эти два элемента идеально заполнят весь экран по высоте без наложений или пустых промежутков между ними.

6.4.3. Настройка Размеров и Отступов Кнопок

Вы можете точно настроить физические размеры кнопок, их внутренние и внешние отступы, а также размер текста на них.

Отступы (android:layout\_margin...):

Эти атрибуты добавляют пустое пространство вокруг элемента, отделяя его от соседних элементов.

android:layout\_marginBottom="8dp": Добавляет отступ в 8 dp снизу элемента.

android:layout\_marginStart="16dp": Отступ слева от элемента.

android:layout\_marginEnd="16dp": Отступ справа от элемента.

android:layout\_marginTop="8dp": Отступ сверху от элемента.

Вы найдете эти атрибуты в activity\_main.xml (для кнопок меню), activity\_game.xml (для кнопок выбора и кнопки меню в игре), dialog\_settings.xml и dialog\_ingame\_menu.xml. Изменяйте числовое значение dp для увеличения или уменьшения отступов.

Внутренние отступы (android:padding...):

Эти атрибуты добавляют пустое пространство внутри элемента, между его содержимым (например, текстом кнопки) и его собственными границами.

android:padding="16dp": Добавляет 16 dp отступа со всех сторон (сверху, снизу, слева, справа).

android:paddingStart, android:paddingEnd, android:paddingTop, android:paddingBottom: Позволяют задать индивидуальные внутренние отступы для каждой стороны.

Размеры кнопок (android:layout\_width, android:layout\_height):

Для кнопок выбора в activity\_game.xml используются следующие атрибуты:

<Button

android:id="@+id/buttonChoice1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:minWidth="180dp"/> (Минимальная ширина 180dp)

android:minWidth: Задает минимальную ширину, которую кнопка должна иметь. Если текст кнопки короткий, она все равно будет не уже этой величины. Вы можете изменить это значение (например, 220dp) для увеличения или уменьшения минимальной ширины кнопок выбора.

Вы также можете использовать фиксированные значения dp для android:layout\_width и android:layout\_height (например, android:layout\_width="200dp", android:layout\_height="50dp") вместо wrap\_content, чтобы задать точный размер кнопки.

Размер текста на кнопках (android:textSize):

Чтобы изменить размер текста, отображаемого на кнопке, добавьте или измените атрибут android:textSize внутри тега <Button>:

<Button

android:id="@+id/buttonStart"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="@string/buttonStartGame"

android:textSize="22sp" /> (Размер текста 22 масштабируемых пикселя (sp))

Используйте единицу sp для текста, чтобы он корректно масштабировался с настройками системного шрифта пользователя.

6.4.4. Увеличение Размера Окна Настроек (Уже Реализовано)

Окно настроек (SettingsDialog) изначально настроено так, чтобы быть значительно больше стандартного маленького диалогового окна. Это сделано для удобства взаимодействия, особенно в горизонтальной ориентации.

Как это реализовано: В файле SettingsDialog.java, внутри метода onCreate(), присутствует специальный блок кода. Этот код динамически определяет текущие размеры экрана устройства и затем устанавливает размер диалогового окна:

Ширина диалога: 80% от ширины экрана (screenWidth \* 0.8).

Высота диалога: 70% от высоты экрана (screenHeight \* 0.7).

Где изменить (если требуется дальнейшая настройка размера):

В SettingsDialog.java, метод onCreate()

WindowManager.LayoutParams lp = new WindowManager.LayoutParams();

lp.copyFrom(getWindow().getAttributes());

DisplayMetrics displayMetrics = new DisplayMetrics();

Логика получения размеров экрана, совместимая с разными версиями Android

if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.R) {

activityContext.getDisplay().getMetrics(displayMetrics);

} else {

WindowManager windowManager = (WindowManager) activityContext.getSystemService(Context.WINDOW\_SERVICE);

if (windowManager != null) {

windowManager.getDefaultDisplay().getMetrics(displayMetrics);

}

}

int screenWidth = displayMetrics.widthPixels;

int screenHeight = displayMetrics.heightPixels;

lp.width = (int) (screenWidth \* 0.8); (Здесь можно изменить процент ширины)

lp.height = (int) (screenHeight \* 0.7); (Здесь можно изменить процент высоты)

getWindow().setAttributes(lp);

Вы можете изменить коэффициенты 0.8 (для ширины) и 0.7 (для высоты) на другие значения (например, 0.9 для 90% ширины, 0.5 для 50% высоты), чтобы точнее настроить размер диалога по своему усмотрению.

6.5. Замена Фонового Изображения Главного Меню (zaglmenu.png)

Фоновое изображение главного меню легко заменить на свое собственное.

1. Подготовьте новое изображение: Выберите изображение, которое вы хотите использовать в качестве фона для главного меню. Для лучшего качества оно должно иметь разрешение, подходящее для полноэкранного отображения в горизонтальной ориентации (например, 1280x720 пикселей для планшетов или 1920x1080 пикселей для более четких экранов). Формат .png или .jpg.

2. Переименуйте: Ваш новый файл изображения должен быть назван zaglmenu.png (или zaglmenu.jpg, если у вас JPEG формат). Код приложения жестко ссылается на это имя.

3. Замените файл в проекте:

Откройте папку app/src/main/res/drawable/.

Найдите и удалите существующий файл zaglmenu.png (или .jpg).

Скопируйте ваш новый, переименованный файл zaglmenu.png (или .jpg) прямо в эту папку drawable/.

4. Перезапустите приложение: Запустите ваше приложение на эмуляторе или реальном устройстве. На экране главного меню теперь должно отображаться ваше новое фоновое изображение.

7. Функции Сохранения и Загрузки Игры

Проект включает в себя полноценную и автоматизированную систему сохранения и загрузки прогресса игры. Эта система управляется классом GameManager.java и использует SharedPreferences для хранения данных.

7.1. Принцип Работы Сохранения Прогресса (SharedPreferences)

Что такое SharedPreferences?: SharedPreferences — это стандартный механизм Android для хранения небольших объемов данных в виде пар "ключ-значение" (подобно словарю или Map). Эти данные сохраняются в постоянной памяти устройства и остаются доступными даже после закрытия приложения. SharedPreferences идеально подходят для хранения настроек пользователя, состояния игры или другой простой информации.

Что сохраняется?: В данном проекте система сохранения очень проста и эффективна: она сохраняет только уникальный идентификатор (nodeId) текущего узла, на котором находится игрок. Этого достаточно, чтобы "запомнить" прогресс игры, поскольку вся сюжетная линия (все узлы и их связи) уже жестко определена в коде GameManager.java. При загрузке игры просто считывается этот nodeId, и игра начинается с соответствующей ему сцены.

7.2. Автоматическое Сохранение Прогресса

Когда происходит: Игра автоматически сохраняет ваш прогресс при каждом переходе на новый узел. Это означает, что вам не нужно беспокоиться о потере прогресса при случайном закрытии приложения или переключении на другое.

Как это работает: В ключевом методе advanceStory() класса GameManager.java (который отвечает за вычисление и переход к следующему узлу после любого действия игрока — нажатия на текст или выбора кнопки) присутствует строка saveGame();. Эта строка вызывает метод saveGame(), который немедленно записывает nodeId текущего узла в SharedPreferences. Таким образом, после каждого шага игрока, его прогресс надежно сохраняется.

7.3. Ручное Сохранение через Внутриигровое Меню

Помимо автоматического сохранения, игрок также имеет возможность вручную сохранить свой текущий прогресс в любой момент во время игры:

1. Доступ к меню: Во время игрового процесса нажмите на кнопку "Меню", расположенную в правом верхнем углу игрового экрана (GameActivity).

2. Выбор опции: Появится всплывающее окно "Внутриигровое меню" (InGameMenuDialog).

3. Действие: Нажмите кнопку "Сохранить" в этом меню.

4. Подтверждение: После сохранения на экране на короткое время появится всплывающее сообщение Toast с текстом: "Игра сохранена!", подтверждающее успешное сохранение прогресса.

5. Возврат в игру: Диалог внутриигрового меню автоматически закроется, и игра продолжится с того места, где она была приостановлена.

7.4. Загрузка Сохраненной Игры из Главного Меню

1. Кнопка "Продолжить": На экране главного меню (MainActivity) есть кнопка с надписью "Продолжить".

2. Активность кнопки: Эта кнопка автоматически становится активной (доступной для нажатия) только в том случае, если существует сохраненный прогресс игры. То есть, если игрок уже начал игру и продвинулся дальше самого первого узла, кнопка будет активна. Если же игра никогда не запускалась, или прогресс был сброшен, кнопка останется неактивной.

3. Действие: При нажатии кнопки "Продолжить", GameManager загружает nodeId последнего сохраненного узла из SharedPreferences, и GameActivity запускается, начиная игру именно с этой сохраненной сцены.

7.5. Сброс Игры ("Начать Игру")

Кнопка "Начать игру" в главном меню (MainActivity) служит для начала нового приключения.

При нажатии на эту кнопку, MainActivity вызывает метод gameManager.resetGame().

Этот метод устанавливает текущий nodeId обратно на самый начальный узел игры (который имеет nodeId "nodeStart") и сохраняет это "сброшенное" состояние. Таким образом, вся предыдущая история и прогресс обнуляются, и игрок начинает игру с самого начала.

8. Встроенные Диалоги

Ваш проект использует два основных типа всплывающих окон (диалогов) для взаимодействия с пользователем: одно для настроек приложения и однодля внутриигрового меню. Диалоги — это компоненты пользовательского интерфейса, которые появляются поверх текущего Activity (экрана), обычно для небольших, специализированных взаимодействий, не требующих перехода на совершенно новый экран.

8.1. Окно Настроек (SettingsDialog)

Вызов: Этот диалог открывается при нажатии кнопки "Настройки" в главном меню (MainActivity).

Назначение: Предоставляет игроку интерфейс для настройки ключевых параметров приложения.

Элементы управления:

Ползунок громкости: Элемент SeekBar, который позволяет пользователю регулировать громкость фоновой музыки. Изменения применяются мгновенно и сохраняются между сессиями.

Выбор языка: Группа радиокнопок (RadioGroup) с вариантами "Русский" и "English" (и любые другие языки, которые вы добавили).

Особенности:

Автоматический размер: Код в файле SettingsDialog.java (в методе onCreate()) специально настроен для динамического увеличения размера этого диалогового окна. Оно занимает примерно 80% ширины и 70% высоты экрана устройства. Это делает диалог значительно больше стандартного и, как следствие, более удобным для взаимодействия, особенно на планшетах или в горизонтальной ориентации. Вы можете изменить эти процентные соотношения в коде, если хотите другой размер (см. раздел 6.4.4).

Сохранение настроек: Все изменения, сделанные в окне настроек (уровень громкости, выбранный язык), автоматически сохраняются в постоянную память устройства (SharedPreferences), чтобы эти настройки были применены при следующих запусках приложения.

Перезагрузка интерфейса: При изменении языка в SettingsDialog и последующем закрытии диалога, MainActivity получает уведомление и перезагружается (recreate()), чтобы применить новый язык ко всем текстовым элементам пользовательского интерфейса.

8.2. Внутриигровое Меню (InGameMenuDialog)

Вызов: Этот диалог открывается при нажатии кнопки "Меню", которая расположена в правом верхнем углу игрового экрана (GameActivity).

Назначение: Предоставляет игроку быстрый доступ к важным функциям, которые могут понадобиться во время игрового процесса, не прерывая его полностью.

Кнопки и их функции:

"Продолжить": При нажатии этой кнопки диалог просто закрывается, и игрок немедленно возвращается к игровому процессу с того места, где он остановился.

"Сохранить": При нажатии этой кнопки текущий прогресс игры (текущий nodeId) сохраняется в SharedPreferences. После сохранения на экране ненадолго появится всплывающее сообщение Toast с текстом: "Игра сохранена!", подтверждающее успешное действие. Диалог затем автоматически закрывается.

"Выйти в меню": При нажатии этой кнопки происходит два действия: сначала сохраняется текущий прогресс игры, а затем GameActivity (игровой экран) закрывается, возвращая игрока на экран главного меню (MainActivity).