ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Департамент программной инженерии

СОГЛАСОВАНО
Приглашенный преподаватель
кафедры менеджмента инноваций

Novikov Eugen VWPKI 31FE642550364FDC
Digital unterschrieben von Novikov Eugen VWPKI 31FE642550364FDC
Datum: 2022.05.15 19:41:45 +02'00'

_____ Е.А. Новиков
«____» _____ 2022 г.

УТВЕРЖДАЮ
Академический руководитель
образовательной программы
«Программная инженерия»
профессор департамента программной
инженерии, канд. техн. наук

_____ В.В. Шилов
«____» _____ 2022 г.

БЭКЕНД КОНСТРУКТОРА ВИЗУАЛЬНЫХ НОВЕЛЛ

Пояснительная записка

ЛИСТ УТВЕРЖДЕНИЯ

RU.17701729.04.08-01 81 01-1-ЛУ

Исполнитель
студент группы БПИ 202
_____/Д.Ю.Фёдоров/
«14» мая 2022 г.

Москва 2022

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

УТВЕРЖДЕН
RU.17701729.04.08-01 81 01-1-ЛУ

БЭКЕНД КОНСТРУКТОРА ВИЗУАЛЬНЫХ НОВЕЛЛ

Пояснительная записка

RU.17701729.04.08-01 81 01-1

Листов 36

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

Москва 2022

АННОТАЦИЯ

Данный документ является пояснительной запиской к программному проекту «Бэкенд конструктора визуальных новелл», реализующему внутренню часть программы для создания игр в жанре "Визуальная новелла".

Раздел «Введение» содержит наименование программы и документ, на основании которого ведётся разработка.

Раздел «Назначение и область применения» содержит функциональное и эксплуатационное назначение программы.

Раздел «Технические характеристики» содержит следующие подразделы: постановка задачи на разработку программы, описание алгоритма и функционирования программы с обоснованием выбора, описание и обоснование выбора метода организации входных и выходных данных, описание и обоснование выбора состава технических и программных средств.

Раздел «Ожидаемые технико-экономические показатели» содержит предполагаемую потребность и экономические преимущества разработки по сравнению с отечественными и зарубежными образцами или аналогами.

Данный документ разработан в соответствии с требованиями:

1) ГОСТ 19.101-77 Виды программ и программных документов [1]
2) ГОСТ 19.102-77 Стадии разработки [2]
3) ГОСТ 19.103-77 Обозначения программ и программных документов [3]
4) ГОСТ 19.104-78 Основные надписи [4]
5) ГОСТ 19.105-78 Общие требования к программным документам [5]
6) ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом [6]
7) ГОСТ 19.404-79 Пояснительная записка. Требования к содержанию и оформлению [7]

Изменения к Пояснительной записке оформляются согласно ГОСТ 19.603-78 [8], ГОСТ 19.604-78 [9].

СОДЕРЖАНИЕ

ГЛОССАРИЙ

Проект – итоговый продукт (игра), состоящий из сцен.

Сцена – созданная пользователем последовательность игровых событий.

Персонаж – объект сцены, обладающий именем и изображением.

Визуальная новелла – видеоигра в жанре «Визуальный роман».

Десктопное приложение – программа, которая устанавливается на ПК пользователя.

Граф проекта – граф, состоящий из входного узла и узлов сцен, связанных между собой через предопределенные пользователем соединения.

Граф сцены – граф, состоящий из узлов, указанных ниже, связанных между собой через предопределенные пользователем соединения.

1) **E-Узел (Entry)** – начальный узел сцены.
2) **CBG-Узел (Change Background)** – узел, при активации которого меняется фон текущей сцены.
3) **CD-Узел (Change Dialog)** – узел, при активации которого меняется текст выбранного персонажа.
4) **CN-Узел (Change Character Name)** – узел, при активации которого меняется имя персонажа, говорящего в данный момент.
5) **PS-Узел (Play Sound)** – узел, при активации которого воспроизводится выбранный звуковой файл.
6) **SC-Узел (Show Character)** – узел, при активации которого на сцене появляется выбранный персонаж.
7) **HC-Узел (Hide Character)** – узел, при активации которого на сцене пропадает выбранный персонаж.
8) **SS-Узел (Start Scene)** – узел, при активации которого воспроизводится выбранная сцена.

Переход – один из двух типов соединения узлов графа, указанных ниже.

1) **SMPL-Переход (Simple)** – моментальный переход.
2) **CLK-Переход (Click)** – переход, если игрок нажал на левую клавишу мыши.

Синглетон – порождающий шаблон проектирования, гарантирующий, что в приложении будет единственный экземпляр некоторого класса, и предоставляющий глобальную точку доступа к этому экземпляру.

Фронтенд – клиентская сторона пользовательского интерфейса программы.

Бэкенд – часть программы, отвечающая за функционирование её внутренней части. Сюда входят функции перемещения данных между фронтендом и бэкендом; функции для взаимодействия графов, узлов, переходов; функции для сохранения/загрузки данных.

1. ВВЕДЕНИЕ

1.1. Название программы

1.1.1. Название программы на русском языке

«Бэкенд конструктора визуальных новелл»

1.1.2. Название программы на английском языке

«Back-end of the visual novel constructor»

1.2. Документы, на основании которых ведется разработка

Учебный план подготовки бакалавров по направлению 09.03.04 «Программная инженерия» и утвержденная академическим руководителем программы тема курсового проекта.

2. НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ

2.1. Функциональное назначение

Программа предоставляет специализированный инструментарий для создания, запуска и распространения игр в жанре «визуальная новелла».

2.2. Эксплуатационное назначение

Эксплуатационным назначением данного приложения является облегчение процесса разработки визуальных новелл. Для достижения данной цели пользователи могут использовать концепцию визуального программирования.

3. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

3.1. Постановка задачи на разработку программы

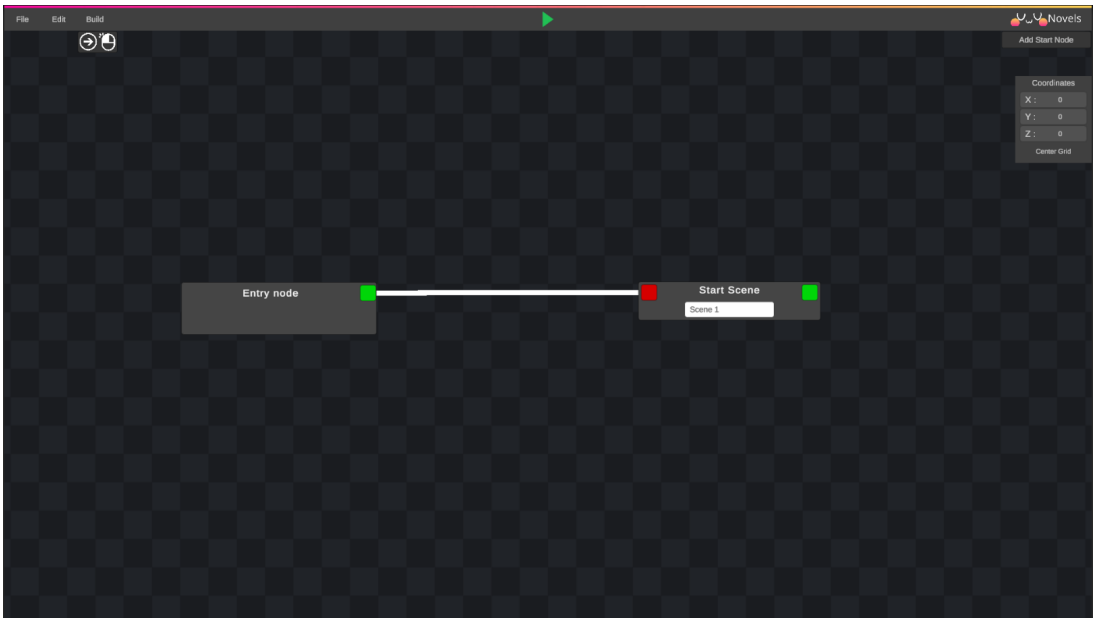
Программа была разработана в соответствии с техническим заданием «Бэкенд конструктора визуальных новелл».

3.2. Описание алгоритма и функционирования программы с обоснованием выбора

3.2.1. Визуальное программирование

Конструктор использует концепцию визуального программирования. Благодаря данной концепции пользователи могут создавать игры в жанре 'визуальная новелла' без написания кода. Чтобы добавить какое-либо действие в игру, пользователю требуется создать подходящий узел и объединить его с помощью переходов. Данный этап происходит на фронтенде программы.

В качестве зарекомендовавшего себя примера визуального программирования, мы рассматривали систему Blueprint в игровом движке Unreal Engine 4. Подробнее об этой системе можно узнать в официальной документации Unreal Engine 4 [10]



Фронтенд окна редактирования проекта.

Рис. 1.

3.2.2. Сущность визуальной новеллы (Класс Novel)

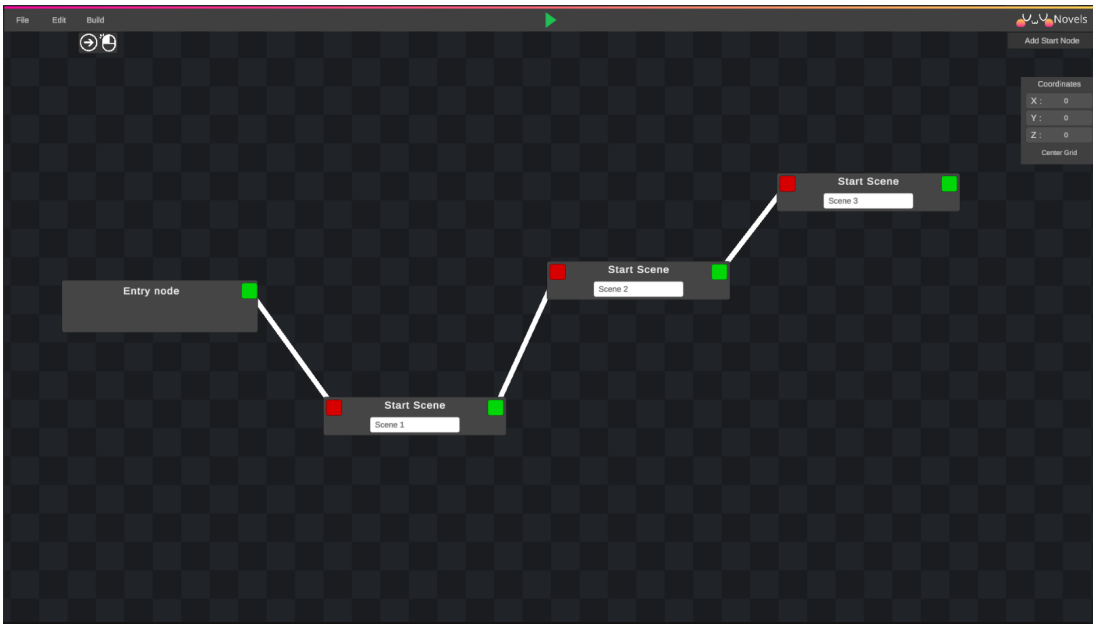
Задача бэкенда программы - это создать абстрактную сущность визуальной новеллы, используя данные из фронтенда. Сущность визуальной новеллы можно сохранять/загружать с помощью сериализации. Сущность новеллы содержит все требуемые ресурсы игры или пути к ним и является входным файлом для запуска игры.

3.2.3. Графы (Класс Graph)

В первую очередь, сущность визуальной новеллы хранит в себе графы проекта. Графы представлены в виде одностороннего списка вида (... -> УЗЕЛ -> ПЕРЕХОД -> УЗЕЛ -> ...). Эта цепочка является последовательностью действий игры. Есть два типа графов: граф игры и граф сцены.

3.2.3.1. Граф игры (Класс NovelGraph)

Граф игры - это главный граф новеллы. Когда пользователь запускает игру, то в первую очередь запускается данный граф. Он состоит из двух типов узлов: начального узла (E-Узел) и узла запуска сцены (SS-Узел). Когда игрок проходит все узлы графа игры, то игра считается завершённой.



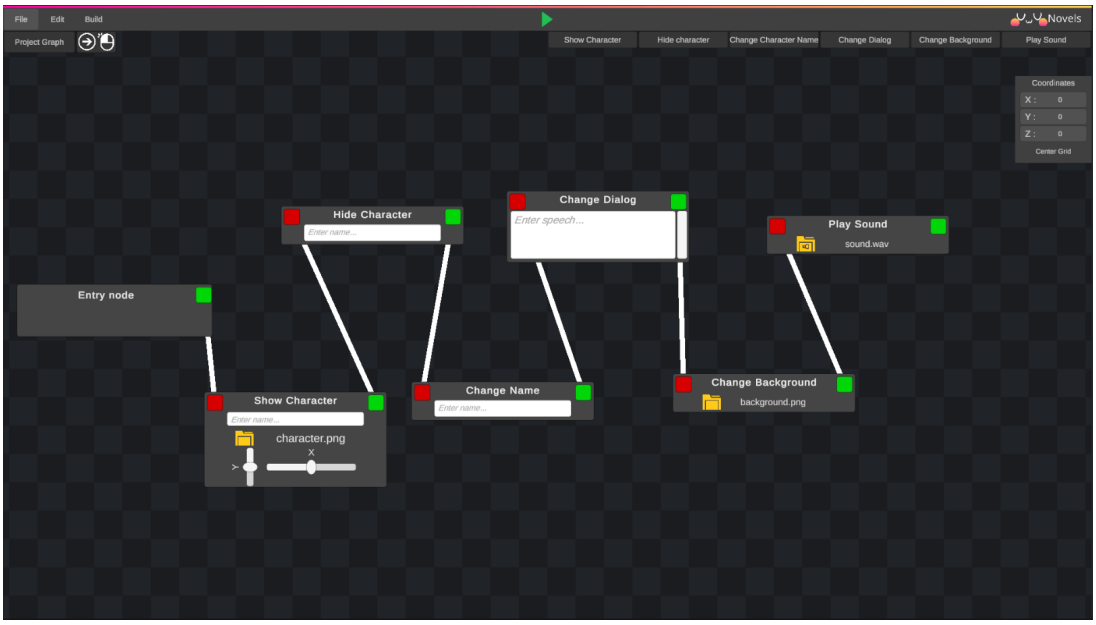
Визуальное представление графа игры

Рис. 2.

3.2.3.2. Граф сцены (Класс SceneGraph)

Когда срабатывает узел запуска сцены (SS-Узел) в графе игры, то запускается последовательность действий сцены. При этом последовательность действий в графе игры

останавливается. Данный граф может иметь шесть типов узлов: E-Узел, SC-Узел, HC-Узел, CN-Узел, CD-Узел, CBG-Узел, PS-Узел. Когда игрок проходит все узлы графа сцены, то сцена считается завершённой. Тогда игрок возвращается обратно на граф игры.



Визуальное представление графа сцены

Рис. 3.

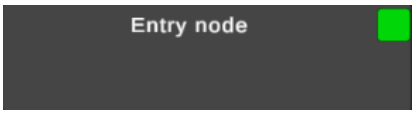
3.2.4. Узлы (Класс Node)

Узел - это блок, входящий в граф. Может быть соединён с другими узлами через переходы. Каждый узел можно активировать. Когда игрок идёт по графу проекта, то активирует каждый узел на пути, когда проходит через него.

У каждого узла есть метод void Run(), который срабатывает каждый раз, когда игрок активирует этот узел.

3.2.4.1. E-Узел (Entry) (Класс EntryNode)

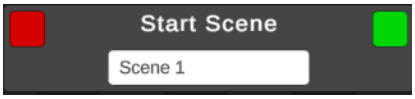
Начальный узел сцены. При активации ничего не делает. Нужен для того, чтобы указать, где находится начало графа (любого). В отличие от всех остальных узлов, не может иметь входных переходов.



Визуальное представление
E-Узла
Рис. 4.

3.2.4.2. SS-Узел (Start Scene) (Класс StartSceneNode)

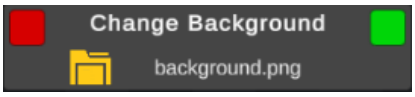
При активации обращается к игровому менеджеру, чтобы запустить граф сцены. Имеет поле с названием сцены. Может быть создан только в графе игры.



Визуальное представление
SS-Узла
Рис. 5.

3.2.4.3. CBG-Узел (Change Background) (Класс ChangeBackground Node)

При активации обращается к игровому менеджеру, чтобы изменить картинку фона. Имеет поле с путём к картинке в формате PNG. Может быть создан только в графе сцены.



Визуальное представление
CBG-Узла
Рис. 6.

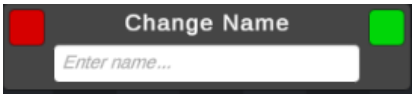
3.2.4.4. CD-Узел (Change Dialog) (Класс ChangeSpeechDialogue Node)

При активации обращается к игровому менеджеру, чтобы изменить текст диалогового окна. Имеет поле с текстом. Может быть создан только в графе сцены.

Визуальное представление
CD-Узла
Рис. 7.

3.2.4.5. CN-Узел (Change Name) (Класс ChangeNameDialogue Node)

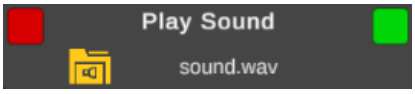
При активации обращается к игровому менеджеру, чтобы изменить поле имени персонажа в диалоговом окне. Имеет поле с текстом. Может быть создан только в графе сцены.



Визуальное представление
CN-Узла
Рис. 8.

3.2.4.6. PS-Узел (Play Sound) (Класс PlaySoundNode)

При активации обращается к игровому менеджеру, чтобы воспроизвезди аудио. Имеет поле с путём к аудио WAV. Может быть создан только в графе сцены.

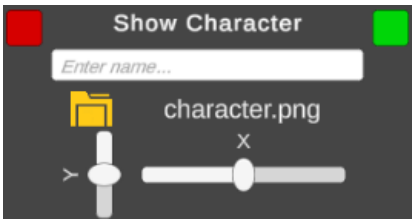


Визуальное представление
PS-Узла
Рис. 9.

3.2.4.7. SC-Узел (Show Character) (Класс ShowCharacterNode)

При активации обращается к игровому менеджеру, чтобы отобразить персонажа. Имеет поле с путём к картинке в формате PNG. Имеет ползунки, чтобы настроить положение персонажа по осям X и Y. Имеет поле для имени персонажа. Может быть

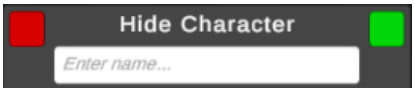
создан только в графе сцены.



Визуальное представление
CBG-Узла
Рис. 10.

3.2.4.8. HC-Узел (Hide Character) (Класс HideCharacterNode)

При активации обращается к игровому менеджеру, чтобы скрыть персонажа. Имеет поле для имени персонажа, которого нужно скрыть. Может быть создан только в графе сцены.



Визуальное представление
HC-Узла
Рис. 11.

3.2.5. Переходы (Класс Transition)

Переходы - это условия перехода игрока от одного узла к другому. Всего есть два перехода: SMPL-переход и CLK-переход.

У переходов есть поле bool CanMoveNext. Если оно true, то совершается переход. Иначе нет.

3.2.5.1. SMPL-Переход (Simple) (Класс SimpleTransition)

Моментальный переход. Не обладает условиями. Переведёт игрока на следующий узел, когда наступит следующий кадр.

У такого перехода поле bool CanMoveNext всегда возвращает true.

Визуальное представление
SMPL-перехода

Рис. 12.

3.2.5.2. CLK-Переход (Click) (Класс ClickTransition)

Переход сработает, когда игрок нажмёт на экран.

У такого перехода поле bool CanMoveNext возвращает true, если игрок нажимает на экран. Условие проверяется каждый кадр.

Визуальное представление
CLK-перехода

Рис. 13.

3.2.6. Строение в Unity

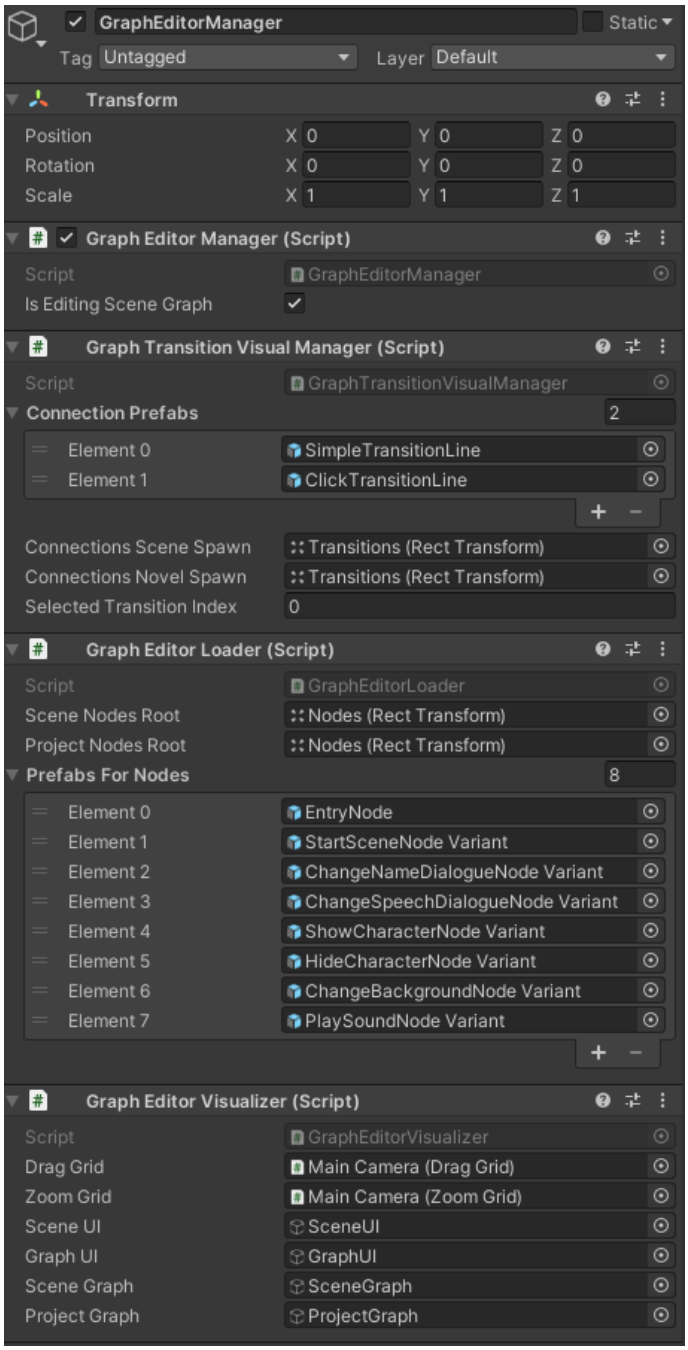
В Unity существуют две сцены: сцена редактора и сцена игры.

3.2.7. Менеджеры

Менеджер - это объекты в Unity, использующие шаблон проектирования синглетон. Подробнее о менеджерах можно узнать в статье на хабре «Использование Singleton в Unity3D» [11]. Через менеджеры абстрактная часть бэкенда может взаимодействовать с пользователем. Существуют три менеджера: менеджер редактора, глобальный менеджер, менеджер игры.

3.2.7.1. Менеджер редактора (Объект GraphEditorManager)

Менеджер, существующий только в сцене редактора.



Объект GraphEditorManager в Unity

Рис. 14.

Имеет следующие важные функции для взаимодействия бэкенда и фронтенда:

1) List<Node> ConvertEditorNodesIntoNodes(Transform nodesParent)

Конвертирует каждый узел на фронтенде (класс EditorNode) в абстрактный узел бэкенда (класс Node), вызывая геттер Node у каждого EditorNode. На вход функции принимается объект Unity, дети которого являются узлами на фронтенде. Возвращает список узлов бэкенда.

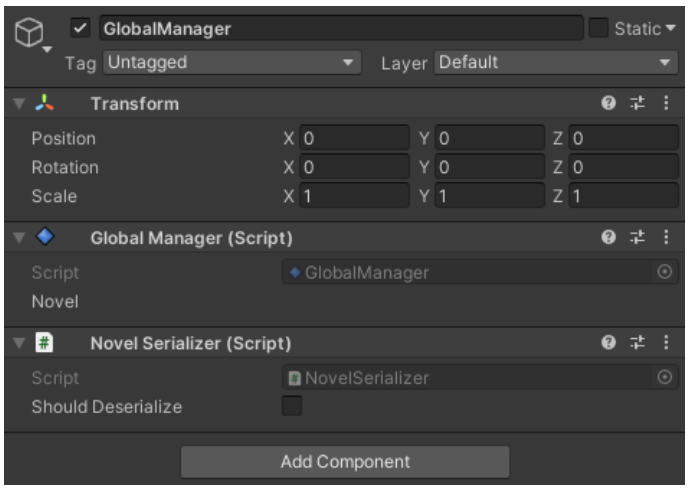
2) Dictionary<Node, EditorNode> SpawnEditorNodesFromNodes(List<Node> nodes,

Transform parent)

Конвертирует каждый абстрактный узел бэкенда (класс Node) в узел фронтенда (класс EditorNode). Все узлы бэкенда появляются как дети переданного объекта Unity. На вход функции принимается список абстактных узлов бэкенда и объект Unity, где должны появится ноды фронтенда. Возвращает словарь, где ключ - это узел бэкенда, а значение - узел фронтенда.

3.2.7.2. Глобальный менеджер (Объект GlobalManager)

Менеджер, существующий и в сцене редактора, и в сцене игры. Содержит сущность новеллы (класс Novel), который сейчас редактируется или проигрывается в качестве игры. Имеет методы для сериализации/десериализации. Имеет логику продвижения по графам.

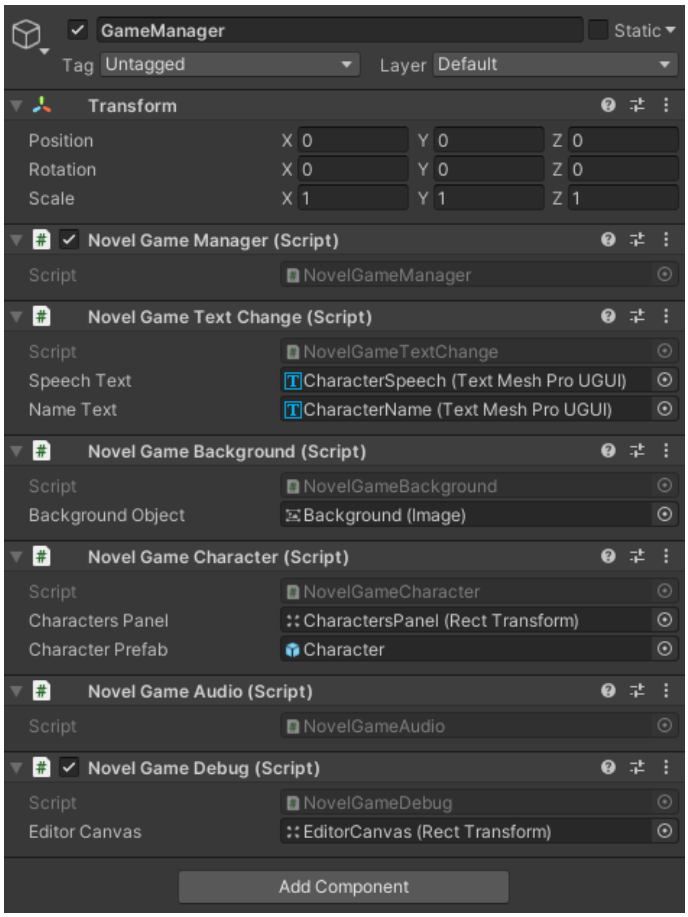


Объект GlobalManager в Unity

Рис. 15.

3.2.7.3. Менеджер игры (Объект GameManager)

Менеджер, существующий только в сцене игры. Содержит все нужные функции для вызова из бэкенда программы, чтобы отображать действия узлов. Например, проигрывает аудио, отображает картинки, меняет текст игры.



Объект GameManager в Unity

Рис. 16.

3.2.8. Схема бэкенда

Чтобы обобщить роль бэкенда, можно построить такую схему (рис. 15). Слева находятся сущности сцены редактора, а справа - сущности сцены игры. Снизу сущности фронтенда, а сверху - бэкенда.

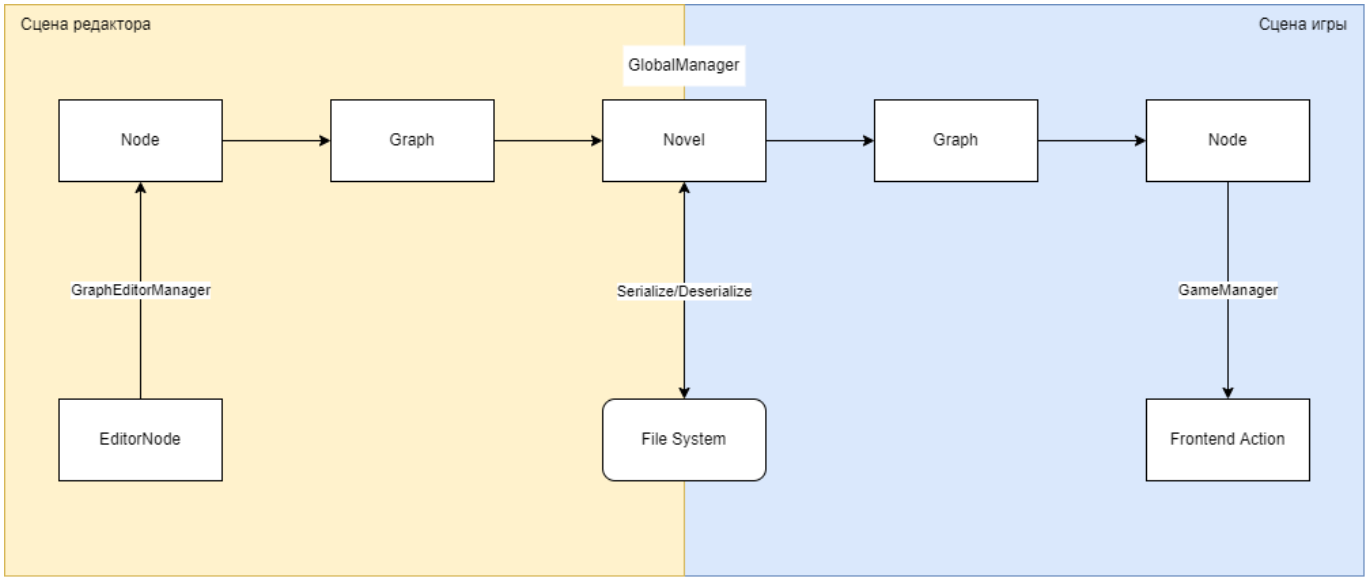
В редакторе пользователь через фронтенд добавляет узел (класс EditorNode). С помощью методов менеджера GraphEditorManager узел фронтенда преобразуется в абстрактный узел бэкенда (класс Node). Далее узел попадает в соответствующий граф (класс Graph), далее граф попадает в сущность визуальной новеллы (класс Novel), которая хранится в оперативной памяти в менеджере GlobalManager.

Через GlobalManager мы можем сохранить сущность визуальной новеллы на диск. Или же наоборот, загрузить сущность визуальной новеллы с диска.

Когда пользователь запускает новеллу в сцене игры, то сущность новеллы достаётся из GlobalManager. Если пользователь создавал новеллу, а теперь хочет её проверить, то сущность новеллы достаётся напрямую из сцены редактора. Если пользователь запустил

новеллу в релизном режиме, то сущность новеллы должна попадать в GlobalManager через файловую систему.

Далее пользователь ходит по графам новеллы, активируя узлы. Узлы взаимодействуют с пользователем через GameManager, который имеет команды для работы с фронтендом.



Объект GameManager в Unity

Рис. 17.

3.3. Описание и обоснование выбора метода организации входных и выходных данных

3.3.1. Входные данные

3.3.1.1. Изображения

Изображения используются в двух узлах:

1) CBG-Узел
2) SC-Узел

Для CBG-Узла изображение используется для создания фона в сцене.

Для SC-Узла изображение используется для создания образа персонажа.

Все изображения подаются в формате "PNG". Данный формат содержит альфа-

канал, благодаря которому можно настраивать прозрачность картинки. Также данный формат поддерживает большое количество цветов и высокое разрешение.

3.3.1.2. Звуковые файлы

Все звуковые файлы подаются в формате "WAV". Данный формат обладает высоким качеством звука.

Используется в PS-Узле для воспроизведения звуков в сцене.

3.3.1.3. Текстовые данные

Текстовые данные используются в следуюших узлах:

1) SS-Узел
2) CD-Узел
3) CN-УЗел
4) SC-Узел
5) HC-Узел

Для SS-Узла пользователь вводит название сцены.

Для CD-Узла пользователь вводит новый диалог.

Для CN-Узла пользователь вводит имя персонажа, говорящего в данный момент.

Для SC-Узла пользователь вводит имя персонажа.

Для HC-Узла пользователь вводит имя персонажа, которого хочет скрыть.

3.3.2. Выходные данные

1) Отображение (запуск) созданной игры
2) Json-файл, содержащий игру
3) Json-файл, содержащий проект

3.4. Описание и обоснование выбора состава технических и программных средств

3.4.1. Unity

Основная причина выбора платформы Unity заключается в том, что он кроссплатформенный, что позовляет запускать конструктор и созданные на нем игры на таких операционных системах как macOS, Windows, Linux. При этом приложения на Unity всегда обладают одинаковым интерфейсом вне зависимости от операционной системы, так как используется собственный рендерер. Также Unity имеет большой инструментарий и

свою библиотеку готовых ассетов.

3.4.2. C#

Unity использует C# как основной язык программирования. Данный язык мы изучали на первом курсе, поэтому смогли применить полученные знанения на практике. Язык обладает требуемым функционалом.

4. ОЖИДАЕМЫЕ ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ

4.1. Предполагаемая потребность

Данный продукт предоставляет возможность создать и запустить новеллу людям, увлекающимся данным жанром игр, чтением книг, писательством или рисованием, но по тем или иным причинам, не желающим заниматься программирования. По результатам Customer Development были получены следующие ответы на вопрос о том, почему заинтересованные в создании визуальных новеллах люди отказываются их создавать.

4.2. Ориентировочная экономическая эффективность

Использование данного продукта позволит разработать новеллу, не используя языки программирования, а пользуясь интуитивно-понятной системой логических узлов, тем самым исключая потребность в деятельности программистов. Это предоставляет гейм-дизайнеру полную свободу действий, так как не требуется ждать завершения работы со стороны программистов.

4.3. Экономические преимущества разработки по сравнению с отечественными и зарубежными аналогами

Прямыми конкурентами являются «Episode» [12], «Twine» [13], «RenPy» [14].

Преимущества по сравнению с Episode:

1) Возможность загружать свои изображения.
2) Наличие десктопной версии.

Преимущества по сравнению с Twine:

1) Наличие графической составляющей, а не только текстовые квесты.
2) Не требуется писать скрипты внутри текста.

Преимущества по сравнению с RenPy:

1) Не требует навыков программирования.

ПЕРЕЧЕНЬ ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ 19.101-77. ЕСПД. Виды программ и программных документов. — Москва : Стандартинформ, 2005.
2. ГОСТ 19.102-77. ЕСПД. Стадии разработки. — Москва : Стандартинформ, 2005.
3. ГОСТ 19.103-77. ЕСПД. Обозначение программ и программных документов. — Москва : Стандартинформ, 2005.
4. ГОСТ 19.104-78. ЕСПД. Основные надписи. — Москва : Стандартинформ, 2005.
5. ГОСТ 19.105-78. ЕСПД. Общие требования к программным документам. — Москва : Стандартинформ, 2005.
6. ГОСТ 19.106-78. ЕСПД. Требования к программным документам, выполненным печатным способом. — Москва : Стандартинформ, 2005.
7. ГОСТ 19.404-79. ЕСПД. Пояснительная записка. Требования к содержанию и оформлению. — Москва : Стандартинформ, 2005.
8. ГОСТ 19.603-78. ЕСПД. Общие правила внесения изменений. — Москва : Стандартинформ, 2005.
9. ГОСТ 19.604-78. ЕСПД. Правила внесения изменений в программные документаы, выполненные печатным способом. — Москва : Стандартинформ, 2005.
10. Цикл статей о инструменте визуального программирования Blueprint в движке Unreal Engine 4:
https://docs.unrealengine.com/4.27/en-US/ProgrammingAndScripting/Blueprints.
11. Статья «Использование Singleton в Unity3D» от пользователя CodeBits:
https://habr.com/ru/post/341830/.
12. Платформа для создания визуальных новелл Episode:
https://www.episodeinteractive.com/.
13. Инструмент для создания текстовых квестов, визуальных новелл Twine:
https://twinery.org/.
14. Движок для создания визуальных новелл RenPy:
https://www.renpy.org/.

ПРИЛОЖЕНИЕ 1

Описание и функциональное назнаение классов

Таблица 1.

Класс	Назначение
Novel	Сущность визуальной новеллы. Она создаётся в конструкторе и используется в игре.
NovelConfiguration	Базовые данные о новеллы. Содержит имя и версию новеллы.
UISettings	Данные о графическом представлении игры.
NovelGlobalData	Данные новеллы, которые используются во время игры.
ConstantData	Данные новеллы, которые не изменяются во время игры.
RuntimeData	Данные новеллы, которые изменяются во время игры.
NovelScene	Данные сцены. Содержит имя сцены и граф сцены.
NovelSerialzer	Функции для работы с сериализацией/десериализацией Novel.
Graph	Представление графа и функций для работы с ним.
NovelGraph	Граф проекта.
SceneGraph	Граф сцены.
GraphEditorManager	Менеджер редактора. Содержит функции для передачи данных между фронтендом и бэкендом.
GlobalManager	Глобальный менеджер. Содержит сущность новеллы и функции для работы с ней.
NovelGameManager	Менеджер игры. Содержит функции для взаимодействия бэкенда с фронтендом.
Transition	Базовый класс для всех переходов. Содержит функцию «возможен ли переход?».
SimpleTransition	SMPL-Переход.
ClickTransition	CLK-Переход.
Node	Базовый класс для всех узлов. Содержит функцию активации.
EntryNode	E-Узел. Точка входа.
PlaySoundNode	PS-Узел. Проигрывает выбранное аудио.
StartSceneNode	SS-Узел. Запускает сцену.
ChangeNameDialogue Node	CN-Узел. Ставит имя персонажа, который сейчас говорит.
ChangeSpeechDialogue Node	CD-Узел. Добавляет диалог.
ShowCharacterNode	SC-Узел. Показывает персонажа.
HideCharacterNode	HC-Узел. Скрывает персонажа.
ChangeBackground Node	CBG-Узел. Меняет задний фон.

ПРИЛОЖЕНИЕ 2

Описание и функциональное назнаение полей, методов и свойств классов

Таблица 2.

Novel			
Свойства			
Имя	Модификатор	Тип	Назначение
Configuration	public	NovelConfiguration	Базовая информация о проекте
UISettings	public	UISettings	Данные интерфейса новеллы
GlobalData	public	NovelGlobalData	Данные проекта
Graph	public	NovelGraph	Граф проекта

Таблица 3.

NovelConfiguration			
Свойства			
Имя	Модификатор	Тип	Назначение
NovelName	public	string	Имя новеллы
NovelVersion	public	string	Версия проекта

Таблица 4.

NovelGlobalData			
Свойства			
Имя	Модификатор	Тип	Назначение
ConstantData	public	ConstantData	Данные, которые не изменяются во время игры
RuntimeData	public	RuntimeData	Данные, которые изменяются во время игры

Таблица 5.

ConstantData				
Свойства				
Имя	Модификатор	Тип	Назначение	
Scenes	public	List<NovelScene>	Список всех сцен	
Методы				
Имя	Модификатор	Тип	Аргументы	Функция
GetSceneByName	public	NovelScene	string	Находит сцену с заданным именем. Если не нашло, то создаёт такую сцену.

Таблица 6.

NovelScene				
Свойства				
Имя	Модификатор	Тип	Назначение	
Name	public	string	Имя сцены	
Graph	public	SceneGraph	Граф сцены	
Методы				
Имя	Модификатор	Тип	Аргументы	Функция
GetGlobalScene-Index	public	int	void	Находит индекс сцены в глобальном списке сцен.

Таблица 7.

NovelSerializer				
Свойства				
Имя	Модификатор	Тип	Назначение	
ShouldDeserialize	public	bool	Нужно ли пытаться десериализовать новеллу при старте игры	
Поля				
Имя	Модификатор	Тип	Назначение	
_globalManager	private	GlobalManager	Глобальный менеджер	
Settings	private	JsonSerializer-Settings	Настройки для сериализатора	
Методы				
Имя	Модификатор	Тип	Аргументы	Функция
SerializeNovel	public	void	string	Сериализует новеллу и помещает её по переданному пути
DeserializeNovel	public	void	string	Десериализует новеллу из переданного пути

Таблица 8.

Graph				
Свойства				
Имя	Модификатор	Тип	Назначение	
EntryRoot	public	Node	Входная точка игры	
CurrenNode	public	Node	Точка игры, на которой остановился игрок	
AllNodes	public	List<Node>	Все узлы сцены	
Поля				
Имя	Модификатор	Тип	Назначение	
_allNodes	private	List<Node>	Все узлы сцены	
Методы				
Имя	Модификатор	Тип	Аргументы	Функция
TryToMoveNext	public	Transition	void	Продвигает граф на 1 шаг, если это возможно. Возвращает переход, который произошёл.
IsGraphEnd	public	bool	void	Достигнут ли конец графа?
AddNodeNext	public	void	Node, Node, Transition	Добавляет в граф узел, подключенный к другому узлу с помощью переданного перехода

Таблица 9.

GraphEditorManager				
Свойства				
Имя	Модификатор	Тип	Назначение	
NovelScene	public	NovelScene	Сцена, которую сейчас редактируют	
Поля				
Имя	Модификатор	Тип	Назначение	
isEditingSceneGraph	public	bool	Сейчас пользователь редактирует сцену?	
_graphEditorLoader	public	GraphEditorLoader	Загрузчик данных из бэкенда	
_graphEditorVisualizer	public	GraphEditorVisualizer	Функции для работы с фронтендом	
Методы				
Имя	Модификатор	Тип	Аргументы	Функция
CompileSceneGraphIntoNovel	public	void	void	Передаёт граф сцены из фронтенда в бэкенд
CompileNovelGraphIntoNovel	public	void	void	Передаёт граф проекта из фронтенда в бэкенд
LoadSceneGraph	public	void	NovelScene	Передаёт граф сцены из бэкенда в фронтенд
LoadProjectGraph	public	void	void	Передаёт граф проекта из бэкенда в фронтенд

Таблица 10.

GlobalManager				
Свойства				
Имя	Модификатор	Тип	Назначение	
Novel	public	Novel	Новелла, которая сейчас или редактируется, или играет. К ней все имеют доступ.	
Методы				
Имя	Модификатор	Тип	Аргументы	Функция
StepGraphNovel	public	void	void	Делает шаг в графе проекта. Если граф закончился, то оповещает об этом
StepGraphScene	public	void	NovelScene	Делает шаг в графе сцены. Если граф закончился, то делает шаг в графе проекта

Таблица 11.

Transition				
Свойства				
Имя	Модификатор	Тип	Назначение	
InNode	public	Node	Узел, присоединённый к началу перехода	
OutNode	public	Node	Узел, присоединённый к концу перехода	
Методы				
Имя	Модификатор	Тип	Аргументы	Функция
CanMoveNext	public	bool	void	Можно ли в данный момент пройти по переходу?
GetName	public	string	void	Имя перехода

Таблица 12.

Simple Transition				
Свойства				
Имя	Модификатор	Тип	Назначение	
InNode	public	Node	Узел, присоединённый к началу перехода	
OutNode	public	Node	Узел, присоединённый к концу перехода	
Методы				
Имя	Модификатор	Тип	Аргументы	Функция
CanMoveNext	public	bool	void	Можно ли в данный момент пройти по переходу? Всегда да
GetName	public	string	void	Имя перехода

Таблица 13.

Click Transition				
Свойства				
Имя	Модификатор	Тип	Назначение	
InNode	public	Node	Узел, присоединённый к началу перехода	
OutNode	public	Node	Узел, присоединённый к концу перехода	
Методы				
Имя	Модификатор	Тип	Аргументы	Функция
CanMoveNext	public	bool	void	Можно ли в данный момент пройти по переходу? Спрашивает у фронтенда, нажал ли пользователь на экран
GetName	public	string	void	Имя перехода

Таблица 14.

Node				
Свойства				
Имя	Модификатор	Тип	Назначение	
EditorPosition	public	Vector2	Позиция узла в редакторе	
TransitionsIn	public	List<Transition>	Переходы, входящие в узел	
TransitionsOut	public	List<Transition>	Переходы, выходящие из узла	
Методы				
Имя	Модификатор	Тип	Аргументы	Функция
Run	public	void	void	Активация узла

Таблица 15.

EntryNode				
Свойства				
Имя	Модификатор	Тип	Назначение	
EditorPosition	public	Vector2	Позиция узла в редакторе	
TransitionsIn	public	List<Transition>	Переходы, входящие в узел	
TransitionsOut	public	List<Transition>	Переходы, выходящие из узла	
Методы				
Имя	Модификатор	Тип	Аргументы	Функция
Run	public	void	void	Активация узла. Ничего не делает

Таблица 16.

PlaySoundNode				
Свойства				
Имя	Модификатор	Тип	Назначение	
EditorPosition	public	Vector2	Позиция узла в редакторе	
TransitionsIn	public	List<Transition>	Переходы, входящие в узел	
TransitionsOut	public	List<Transition>	Переходы, выходящие из узла	
Audio	public	Audio	Аудио, которое требуется проиграть	
Методы				
Имя	Модификатор	Тип	Аргументы	Функция
Run	public	void	void	Активация узла. Проигрывает аудио

Таблица 17.

StartSceneNode				
Свойства				
Имя	Модификатор	Тип	Назначение	
EditorPosition	public	Vector2	Позиция узла в редакторе	
TransitionsIn	public	List<Transition>	Переходы, входящие в узел	
TransitionsOut	public	List<Transition>	Переходы, выходящие из узла	
Scene	public	NovelScene	Сцена, которую требуется показать	
Методы				
Имя	Модификатор	Тип	Аргументы	Функция
Run	public	void	void	Активация узла. Запускает новую сцену

Таблица 18.

ChangeNameDialogueNode				
Свойства				
Имя	Модификатор	Тип	Назначение	
EditorPosition	public	Vector2	Позиция узла в редакторе	
TransitionsIn	public	List<Transition>	Переходы, входящие в узел	
TransitionsOut	public	List<Transition>	Переходы, выходящие из узла	
NameToShow	public	string	Имя, которое нужно отобразить	
Методы				
Имя	Модификатор	Тип	Аргументы	Функция
Run	public	void	void	Активация узла. Ставит выбранное имя в диалоговое окно

Таблица 19.

ChangeSpeechDialogueNode				
Свойства				
Имя	Модификатор	Тип	Назначение	
EditorPosition	public	Vector2	Позиция узла в редакторе	
TransitionsIn	public	List<Transition>	Переходы, входящие в узел	
TransitionsOut	public	List<Transition>	Переходы, выходящие из узла	
SpeechToShow	public	string	Текст, который нужно отобразить	
Методы				
Имя	Модификатор	Тип	Аргументы	Функция
Run	public	void	void	Активация узла. Ставит выбранный текст в диалоговое окно

Таблица 20.

ShowCharacterNode				
Свойства				
Имя	Модификатор	Тип	Назначение	
EditorPosition	public	Vector2	Позиция узла в редакторе	
TransitionsIn	public	List<Transition>	Переходы, входящие в узел	
TransitionsOut	public	List<Transition>	Переходы, выходящие из узла	
Character	public	Character	Персонаж, которого нужно показать	
Pos	public	Vector3	Позиция персонажа	
Методы				
Имя	Модификатор	Тип	Аргументы	Функция
Run	public	void	void	Активация узла. Отображает персонажа. Если персонаж уже на сцене, то меняет его позицию

Таблица 21.

HideCharacterNode				
Свойства				
Имя	Модификатор	Тип	Назначение	
EditorPosition	public	Vector2	Позиция узла в редакторе	
TransitionsIn	public	List<Transition>	Переходы, входящие в узел	
TransitionsOut	public	List<Transition>	Переходы, выходящие из узла	
Character	public	Character	Персонаж, которого нужно показать	
Методы				
Имя	Модификатор	Тип	Аргументы	Функция
Run	public	void	void	Активация узла. Скрывает персонажа

Таблица 22.

ChangeBackgroundNode				
Свойства				
Имя	Модификатор	Тип	Назначение	
EditorPosition	public	Vector2	Позиция узла в редакторе	
TransitionsIn	public	List<Transition>	Переходы, входящие в узел	
TransitionsOut	public	List<Transition>	Переходы, выходящие из узла	
BackgroundPath	public	string	Путь к картинке, которую нужно показать	
Методы				
Имя	Модификатор	Тип	Аргументы	Функция
Run	public	void	void	Активация узла. Ставит выбранный задний фон

Лист регистрации изменений

Номера листов (страниц)					Всего листов (страниц) в докум.	№ документа	Входящий № сопроводительного докум. и дата	Подп.	Дата
Изм.	измененных	замененных	новых	аннулированных					