Инструкция по использованию диалоговой системы (без использования сетевой части)

ОГЛАВЛЕНИЕ

1. Структура пакета

1.1 Модель

1.1.1 Перечисления

CheckType

```
[System.Serializable]
public enum CheckType
{
   Equlas = 0,
   NotEqulas = 1,
   Greate = 2,
   Less = 3
}
```

Тип проверки для узла условий. Любой параметр можно проверить *равно/не равно*, а для параметров типа int доступна также проверка *больше/меньше.*

DialogueAnimType

```
public enum DialogueAnimType
{
   Talk = 0,
   Yes = 1,
   No = 2,
   Nervously = 3,
   InSurprise = 4,
   Quastion = 5,
   Fear = 6,
}
```

Тип анимации. Устанавливается в узле реплики. Если диалоговая зона использует анимации, то она задаётся эмоцией (как персонаж говорит реплику)

DialogueState

```
public enum DialogueState
{
   Disactive = 0,
   TalkReplic = 1,
   WaitChoose = 2,
   WaitEvent = 3,
   ChooseComplete = 4,
   EventComplete = 5,
   LastClick = 6
}
```

Используется диалоговой зоной для контроля состояний во время хода диалога.

ParameterType

```
[System.Serializable]
public enum ParameterType
{
   Bool,
   Int
}
```

Параметры используются для хранения данных между диалогами и помогают изменять сюжет без выбора игрока.

1.1.2 Классы

AnswerUI

```
[System.Serializable]
public class AnswerUI
{
   [Tooltip("Панель для варианта")] public GameObject variantButton;
   [Tooltip("Текст варианта")] public Text variantText;
}
```

Данный класс используется для того, чтобы в диалоговую зону можно было передать не просто кнопку варианта ответа, но и текст варианта ответа. Это используется для случаев, когда они разделены иерархически.

Добавление этих частей в один класс поможет позже представить их в виде единого массива.

variantButton – Объект, который будет появляться как кнопка выбора. Не обязательно это может быть сама кнопка (например, панель подложка).

variantText – Текст, на котором будет выводиться текст ответа перед выбором.

DialogueActiorPointItem

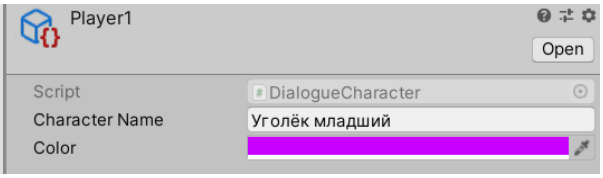
```
[Serializable]
public class DialogueActorPointItem
{
   [Tooltip("Позиция персонажа в диалоге")] public Transform
actorPoint;
   [Tooltip("Роль персонажа в диалоге")] public DialogueCharacter
actorRole;

   public DialogueActorPointItem(Transform point)
   {
      actorPoint = point;
   }
}
```

Данный класс используется для того, чтобы точки, куда встают персонажи в диалоге, можно было помещать в массиве диалоговой зоны в любом порядке. Достаточно лишь выбрать роль участника диалога и установить для него точку в пространстве с помощью пустого объекта.

При старте диалога по роли будет найден соответствующий персонаж и установлен на точку.

DialogueCharacter

```
[System.Serializable]
[CreateAssetMenu(menuName = "IgoGoTools/DialogueCharacter")]
public class DialogueCharacter : ScriptableObject
{
   public string characterName;
   public Color color;
}
```



Та самая роль персонажа. Помимо уже упомянутого использования в установке персонажа на точку, используется для установки субтитрам определённого персонажа. Этот же цвет используется и в редакторе.

ScenarioParameter

```
[System.Serializable]
public class ScenarioParameter
{
    public ParameterType type;
    public string parameterName;
    public int intValue;
    public bool boolValue;
}
```

Этот класс используется для хранения различных переменных, которые позволяют менять ход диалога без участия игрока. К примеру, можно запоминать, делал ли игрок какие-то действия или учитывать отношения с конкретным персонажем.

DialogueNode

Основной класс системы. Все диалоги состоят из узлов. Типы узлов бывают разные, но из-за особенностей хранения все они объединены в одном классе. Просто в зависимости от выбранного типа используются разные свойства класса.

Здесь может хранится текст реплики с указанием роли говорящего, роль и выборы для какой-то развилки, условие, которое будет менять ход диалога в зависимости от значения параметра, диалоговое событие, или ссылка на другой узел.

1.2 Контроллеры

1.2.1 Файл диалога (DialogueSceneKit)

Этот класс наследуется от ScriptableObect. В нём хранится лист диалоговых узлов. Они имеют ссылки друг на друга образуя диалоговые ветки. Конкретно этот класс интересен тем, что предоставляет ряд методов для добавления и удаления узлов с корректным переназначением ссылок. Всё это используется скрыто от пользователя.

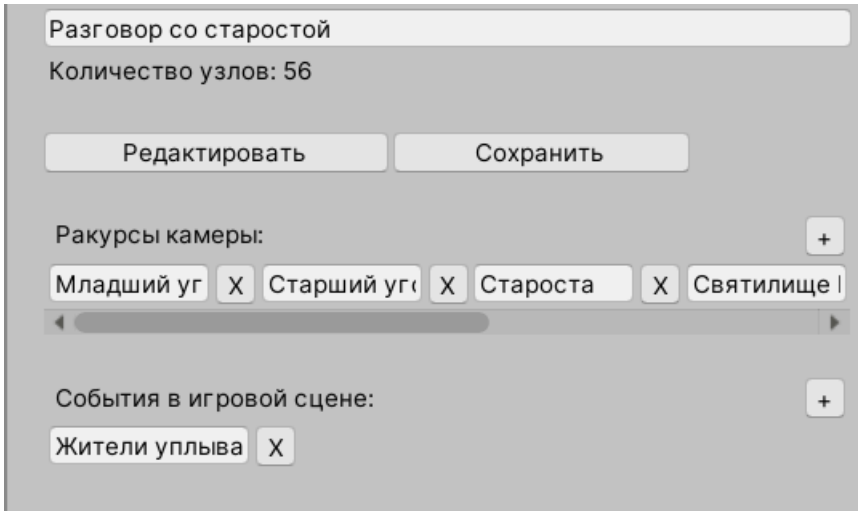
1.2.2 Пакет диалоговых параметров (ParameterPack)

Этот класс также наследуется от ScriptableObject. Он также используется визуальной частью, поэтому подробно разбираться не нужно. Также он имеет ряд методов, которые помогают корректно менять параметр, узнавать его тип и значение, искать по названию и т.д.

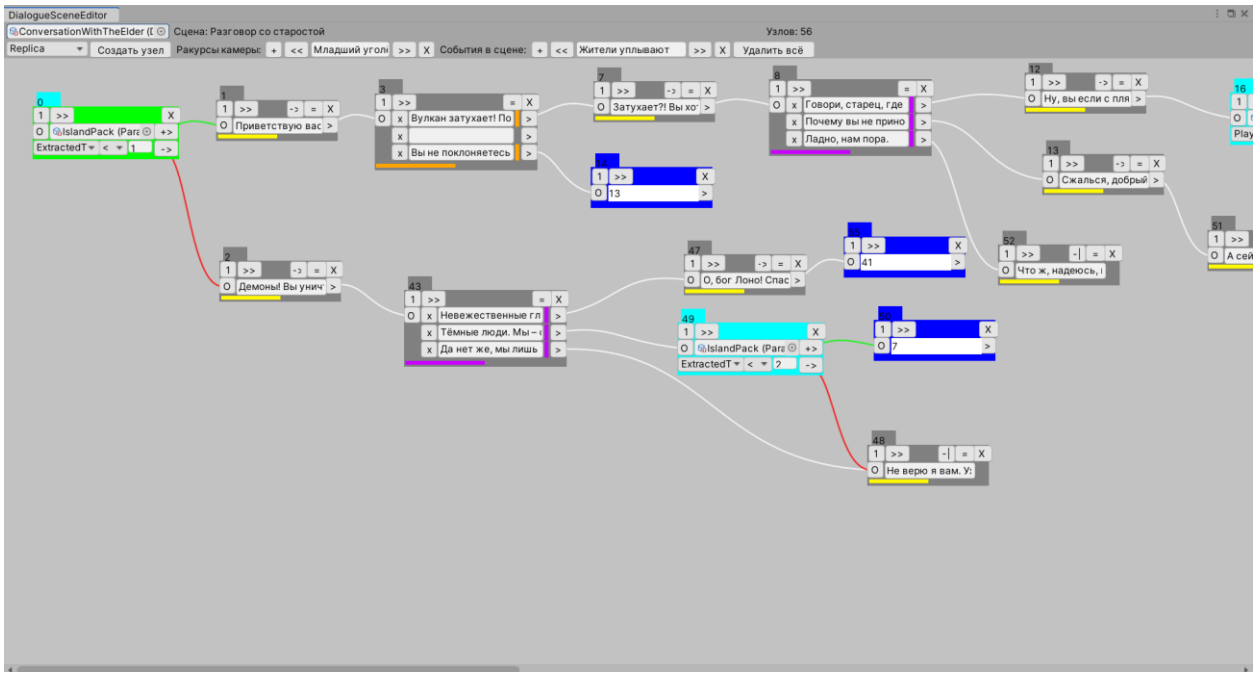
1.3 Представления (Editor)

1.3.1 Инспектор файла диалога

Для класса DialogueSceneKit написан кастомный инспектр. Там можно задать развёрнутое название диалога, чтобы легче было вспоминать сцену, указать возможные ракурсы камеры и события, сохранять и редактировать.

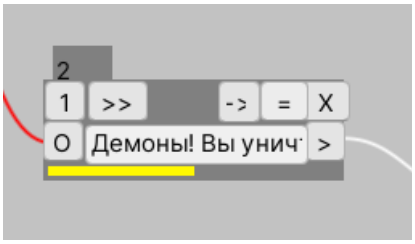


1.3.2 Редактор диалогов

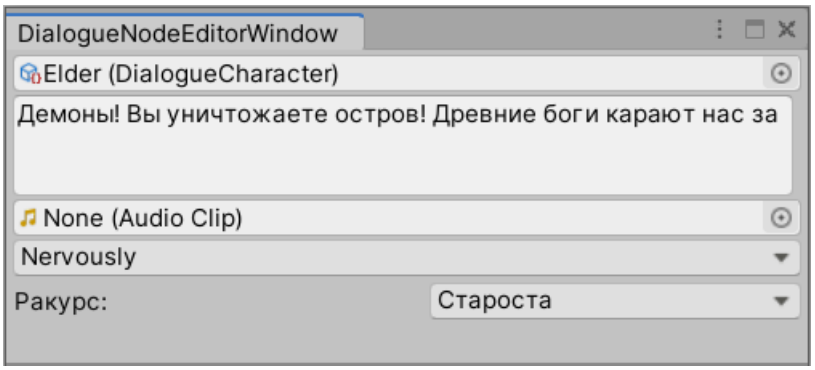


Класс DialogueSceneEditor наследуется от EditorWindow. Здесь можно прописывать логику конкретной диалоговой сцены, о чём подробно будет написано в разделе 3.

1.3.3 Редактор диалогового узла

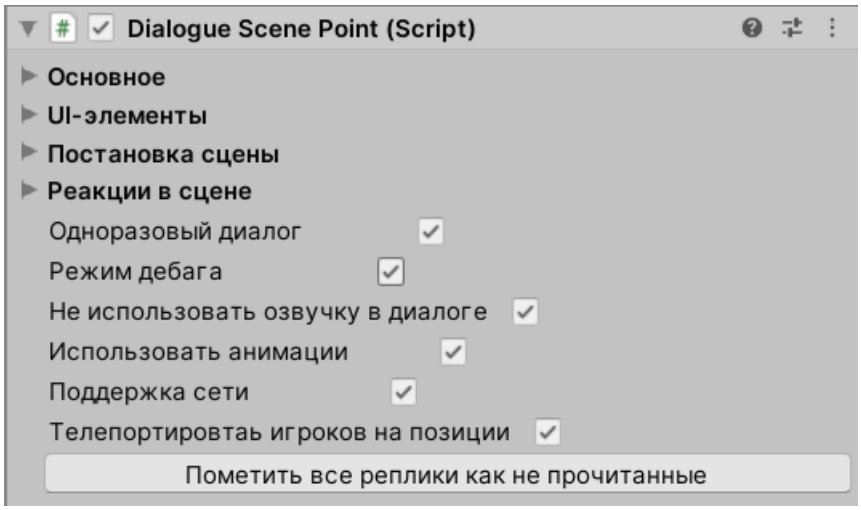


Если у узла имеется кнопка «=», то можно открыть окно расширенных настроек. Оно будет отличаться в зависимости от типа узла. К примеру, для реплики оно будет выглядеть следующим образом:



1.3.4 Инспектор диалоговой зоны

Для класса DialogueScenePoint написан кастомный инспектор. Подробнее о заполнении диалоговой зоны будет описано в разделе 4.



1.4 Части поддержки

1.4.1 Диалоговая зона

Класс DialogueScenePoint используется в конкретной игровой сцене. Туда требуется предавать части интерфейса, конкретных персонажей, точки для позиций игрока, ракурсов камеры и т.д.

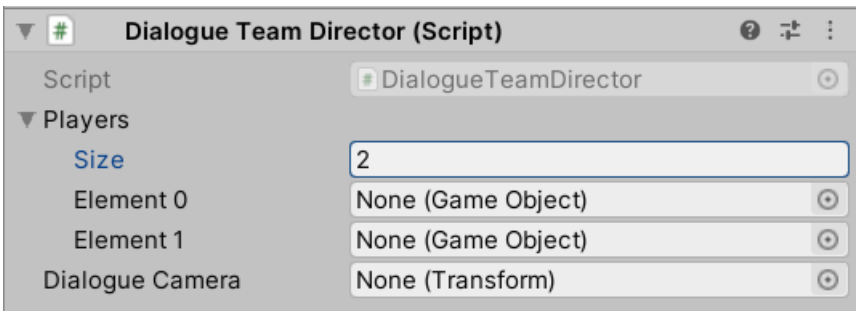
1.4.2 Контроллер персонажа для диалогов

Этот класс позволяет носителю участвовать в диалогах. Причём, не имеет значения, управляется персонаж игроком или является NPC. Однако для первого случая требуется под каждую игру дописывать функционал для перехода в состояние диалога. Также данный класс автоматически добавляет аниматор. Если в диалоговой зоне используются анимации, то команды, отдающиеся диалогом в виде типа DialogueAnimType будут преобразованы в конкретные анимации. Сам аниматор должен быть построен конкретным образом. Пример есть в пакете.

PlayerTransform – компонент Transform персонажа используется для того, чтобы поставить его на нужную точку в диалоге.

DialogueCharacter – роль персонажа.

1.4.3 Скрипт переключения геймплей-диалог



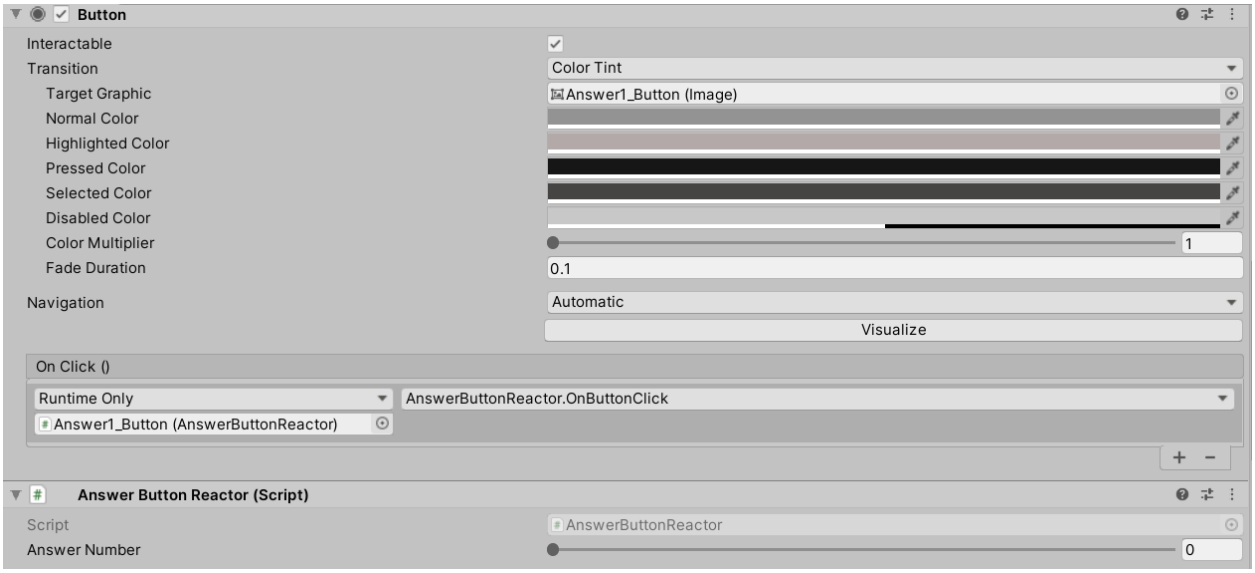
Данный класс используется для перевода управляемых игроком персонажей в состояние диалога. Сюда же передаётся игровая камера, которая потом перемещается по ракурсам в диалогах. Для конкретный игры функционал по отключению возможностей управления игроком и камерой и возврата назад нужно прописывать самостоятельно.

1.4.4 Диалоговое событие

```
public abstract class DialogueEventReactor : MonoBehaviour
{
    public abstract void OnEvent();
}
```

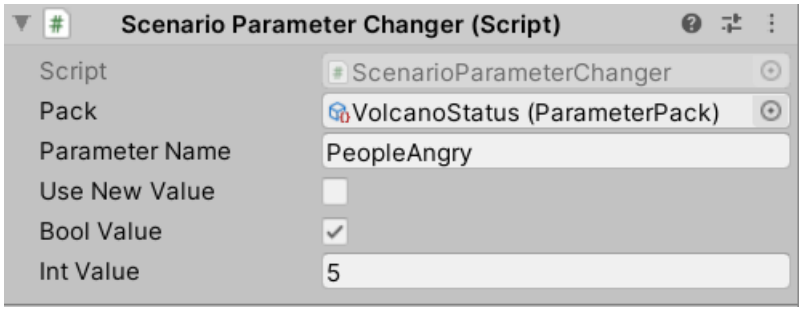
Данный класс является базовым для всевозможных реакций на события, испускаемые диалогом. В принципе, если в игре используется система модулей, то можно создать один класс, который будет запускать другие модули по команде из диалога. Пример есть в классе DialogueEventReactorSample

1.4.5 Скрипт прослойка для событий кнопок вариантов ответов



Данный класс используется для того, чтобы не приходилось для каждой новой сцены диалога добавлять обработчики события кнопки вручную. Вместо этого для каждой кнопки, отвечающий за выбор в диалоге требуется указать обработчиком этот скрипт. Также в нём нужно указать номер ответ. Каждая диалоговая зона будет сама искать и задавать обработчики.

1.4.6 Модуль изменения значения сценарного параметра

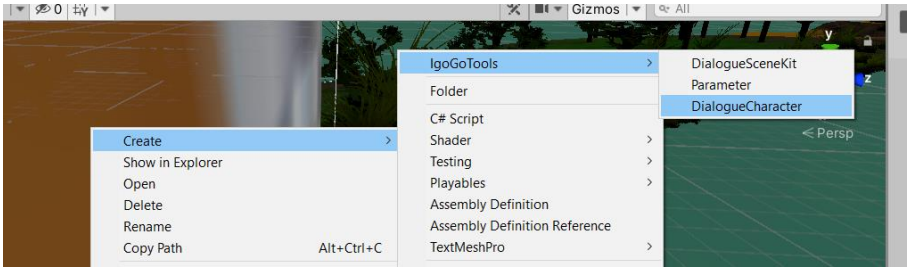


С помощью этого скрипта можно в любое время менять значения сценарных параметров. Достаточно вызвать метод Use().

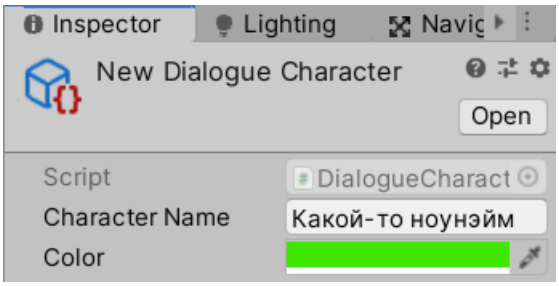
При наведении на названия полей можно увидеть подробные подсказки.

2. Инструкция по созданию ролей

Каждая роль – это отдельный персонаж во всей игре. Для создание новой роли нужно использовать контекстное меню окна Project.

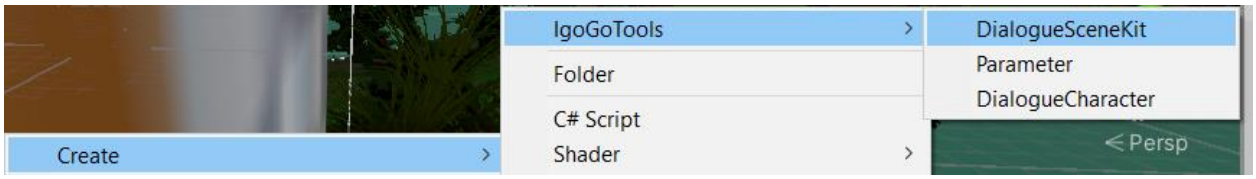


После этого для созданной роли можно задать развёрнутое имя. Цвет указанный в этом файле, будет использован при показе реплик этого персонажа.

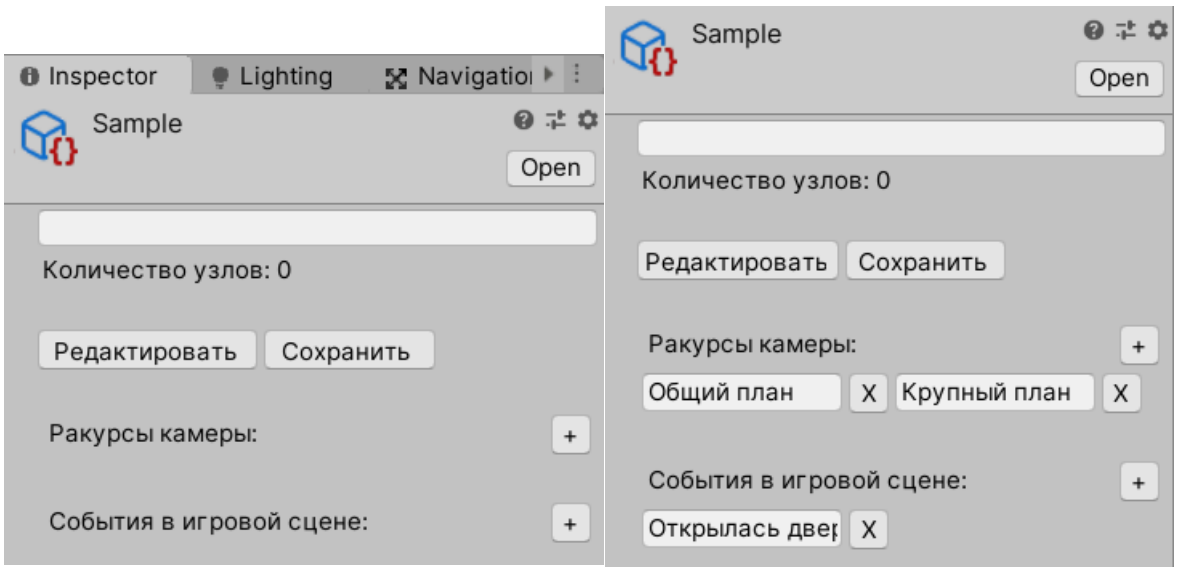


3. Инструкция по созданию файла диалога

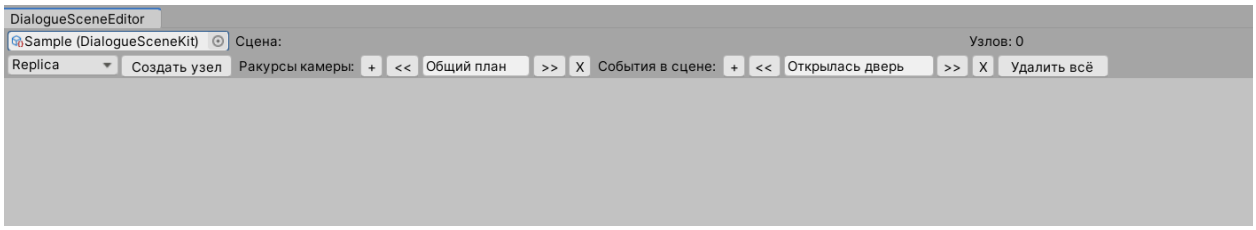
Для создания диалоговой схемы нужно в окне Project воспользоваться контекстным меню.



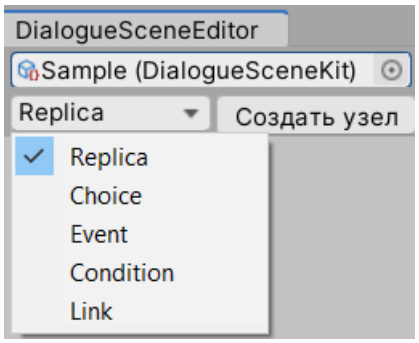
После чего будет доступен инспектор этой схемы. Там можно добавить ракурсы камеры, которые будут использоваться в диалоговой сцене и события, которые она будет испускать в игровую сцену.



Кнопка «Редактировать» открывает редактор диалоговой схемы. Сверху там доступно меню с теми же функциями добавления и удаления ракурсов и событий. Однако здесь доступно также создание самих диалоговых узлов.



Чтобы создать новый узел, нужно выбрать его тип в выпадающем меню, а потом нажать «Создать узел».

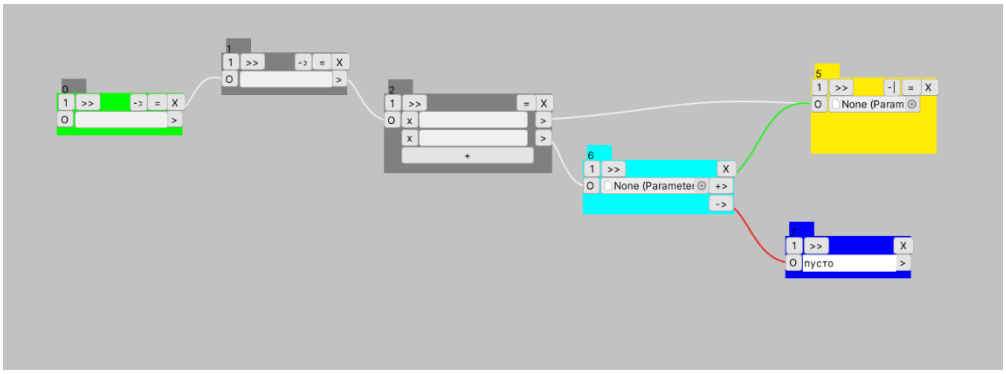


Выбранный узел появится в центре окна. Его можно перемешать, если навестись на верхнюю часть узла и зажать левую кнопку. Из-за особенностей класса Event перемещение отображается совсем не плавно. Вероятно, вы отпустите кнопку, и только чуть позже узел переместиться. Иногда помогает прокрутка колёсика мыши.

Узел, с которого будет начинаться диалог, будет выделен зелёным цветом. Вы можете назначить первым другой узел с помощью кнопки «1» на каждом узле.

Чтобы установить связь от узла к узлу нужно сначала нажать кнопку «>» одного узла, а потом «o» другого. Связь установится автоматически.

Вы можете создать узлы 5 типов. Каждый из них будет иметь номер. На рисунке:



0 – реплика, стартовый узел;
1 – реплик;
2 – выбор;
3 – событие;
4 – условие;
5 – ссылка.

У всех узлов есть кнопки:

«1» - назначение первым узлом;

«>>»/«<<» - смена направления узла (помогает вести линейные ветки змейкой, чтобы экономить место)

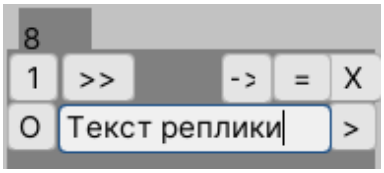
«->»/«-|» - помечает узел, как имеющий продолжение/конечный. У конечного будет недоступен выход.

«X» - удалить узел

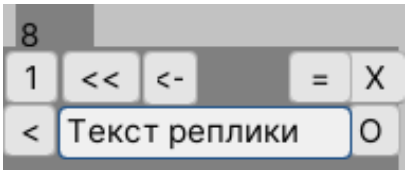
«o» - вход узла

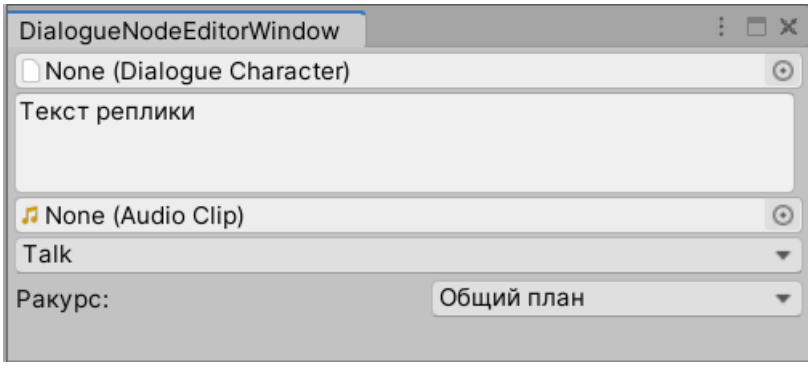
«>»/«<» - выход узла (в зависимости от направления). У условия немного видоизменено, чтобы обозначить разные варианты исходов.

Для некоторых узлов доступна кнопка «=». Она открывает окно расширенных настроек узла. Далее каждый узел будет разобран подробно.



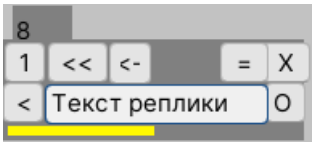
У реплики есть текстовое поле для ввода текста реплики и кнопка вызова расширенной настройки.



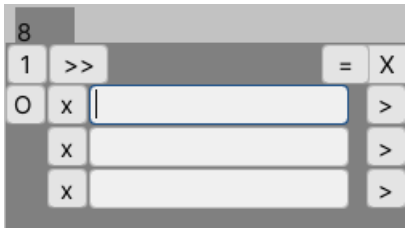


В окне расширенных настроек можно задать роль – говорящего, ракурс камеры. Если игра использует озвучку, сюда передаётся аудиофайл реплики. Также, если используются анимации, можно выбрать эмоцию.

После того, как вы укажите роль говорящего, узел начнёт отображать его цвет. Это поможет вам легко искать те узлы, у которых вы не задали роль.



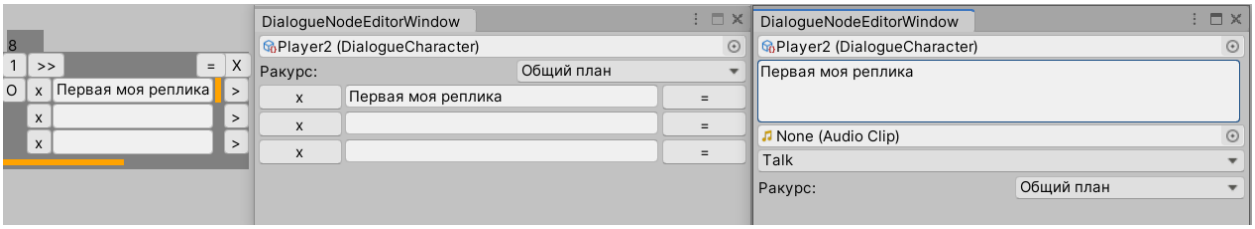
Суть данного узла в том, что игроку, которого вы укажите ролью в качестве говорящего, будет предоставляться выбор из нескольких реплик. С помощью кнопки «+» можно добавлять варианты.



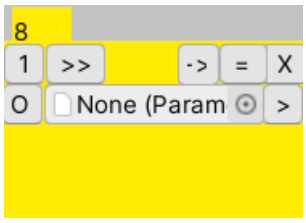
Каждый новый вариант представляет из себя всё ту же реплику. Но расширенные настройки распространяются именно на окно выбора.

Пока лимит вариантов = 3. Но это можно легко изменить в файле Editor/DialogueSceneEditor.cs

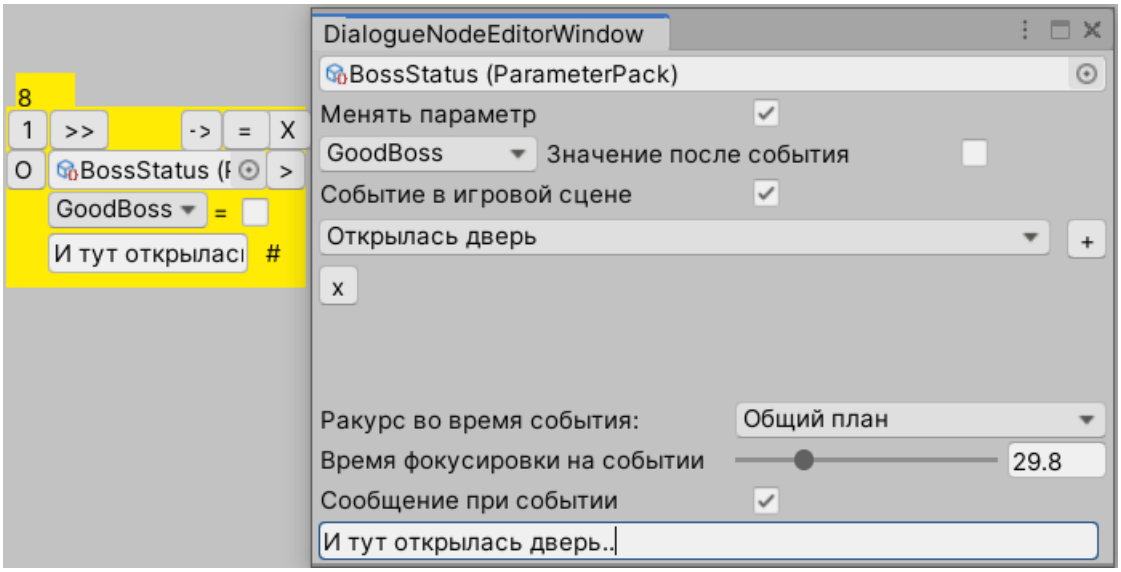
```
public class DialogueSceneEditor : EditorWindow
{
    private const int limitAnswerCount = 3;
```



Вызов окна расширенных настроек для узла «выбор» откроет окно, где можно указать роль выбирающего, ракурс камеры и также работать с вариантам. Как уже было сказано, каждый из вариантов является репликой, поэтому для него следует вызвать своё окно расширенных настроек, чтобы задать роль говорящего. Как правило, это тот же персонаж, кто делает выбор. Если все роли указаны, то цвет будет указан не только внизу узла, но и при каждом из вариантов.



Работать с узлом «событие» без окна расширенных настроек нет смысла.

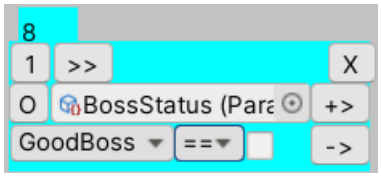


Данный узел может выполнять три функции: менять значения диалоговых параметров, информировать подписчиков о конкретных событиях и давать команду о показе какого-то текстового сообщения.

Для изменения параметра требуется указать в первое поле пакет параметров. В выпадающем списке ниже («менять параметр») будут все параметры этого пакета. А вариант изменения значения будет определяться в зависимости от типа параметра.

Для события в игровой сцене требуется, чтобы эта диалоговая схема содержала хотя бы одно событие. Его и выбирают. Можно также указать фиксацию на событии в каком-то ракурсе камеры. К примеру, по событию у вас запускается открытие двери, и кто-то входи в комнату. Вы хотите показывать это игроку в течение почти 30 секунд, а потом пойти дальше по диалогу. При использовании события сцены на узле в схеме будет показан знак «#».

Текстовое сообщение будет показано в специальной части интерфейса, отведённой под него в течение определённого количества времени, а потом исчезнет.



В узел условия помещают пакет параметров, выбирают нужный и устанавливают условие, которое будет проверяться (==, !=, >, <). Если условие выполняется, диалог идёт по ветке «+>» иначе – «->».

Узел ссылки используется, если в ходе диалога нужно вернуться к какому-то узлу, но прямая ссылка с отрисованной связью будет плохо читаться на схеме. Достаточно просто связать выход этого узла с

любым другим и установится невидимая связь. Сам узел будет показывать номер узла, на который он ссылкается.

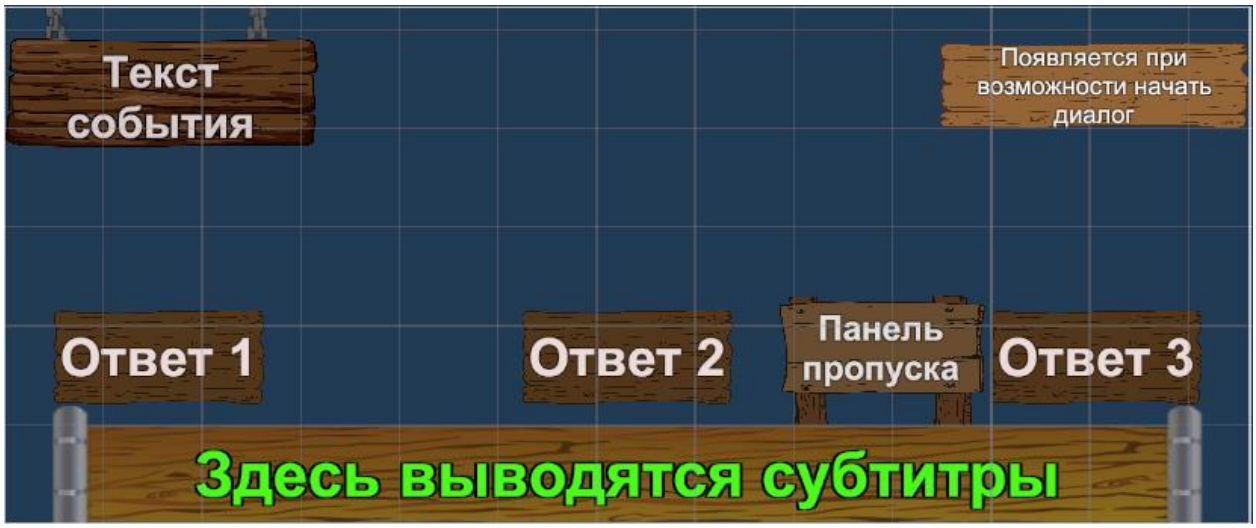
4. Инструкция по интеграции созданного диалога и ролей в игру

4.1 Подготовка интерфейса

Интерфейс для диалога должен содержать следующие элементы:

- Панель с субтитрами. Любой контейнер (с картинкой-подложкой или пустой) и текст внутри. Сама строка субтитров передаётся из диалога.
- Кнопки с вариантом ответа (о подключении специальных скриптов было написано в пункте 1.4.5). Сами строки вариантов ответа передаются из диалога. Если для варианта в узле выбора схемы диалога указана пустая строка, то эта кнопка не будет отображаться. Так можно делать симметричное расположение кнопок.
- Панель пропуска реплики – любой контейнер, в котором будет уже заранее прописанный текст. Эта панель появляется, когда даётся возможность пропустить реплику по нажатию клавиши.
- Текст события – любой контейнер с текстом внутри. Появляется, когда диалоговое событие содержит текст. Отображается 5 секунд, после чего пропадает. Строка текста события передаётся из диалога.
- Панель-подсказка – любой контейнер с текстом. Эта панель появляется, когда персонажу даётся возможность начать диалог. Строка текста подсказки передаётся из диалога.

Ниже представлен пример интерфейса.



4.2 Подготовка персонажей

Если нужно, чтобы персонажи как-то подкрепляли свои реплики движениями, нужно снабдить их правильным деревом анимаций и специальным скриптом DialogueController. Пример корректного дерева анимаций есть в пакете. Работает он по следующему принципу:

У вашего персонажа (игрового или NPC) есть какие-то состояния, которые не относятся к диалогу. В примере это состояние Stay. При старте диалога компонент DialogueController будет давать аниматору сигнал TalkStatus=1, что переведёт персонажа в состояние анимации, когда он просто стоит в диалоге.

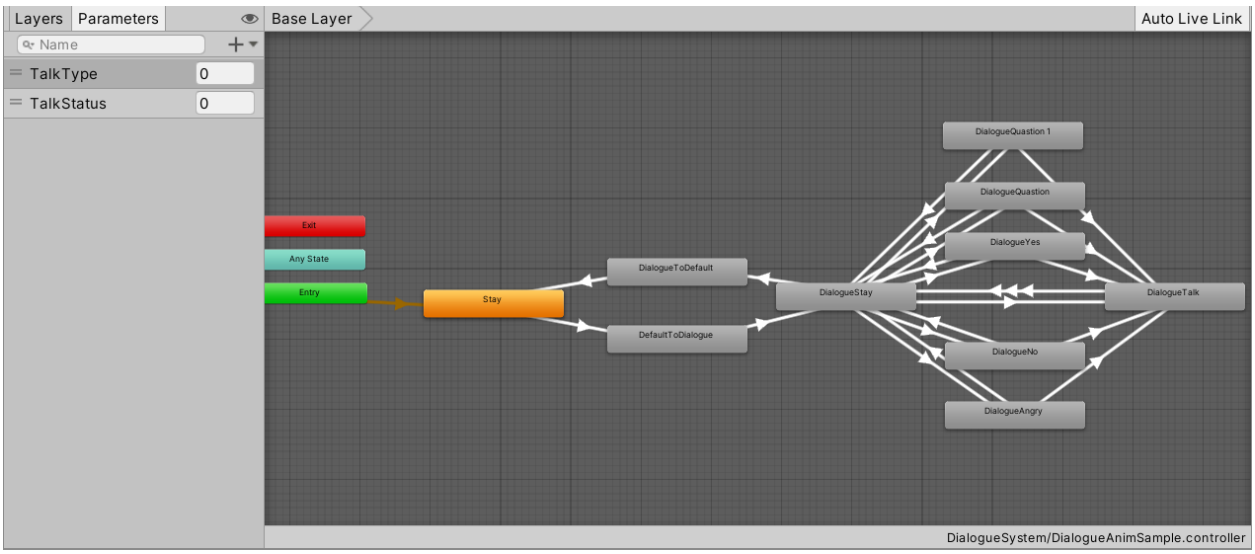
По мере хода диалога каждая реплика этого персонажа будет снабжаться командой TalkStatus=2 и TalkType = №_эмоции_в_узле_реплики. Эмоции уже названы. На данный момент там 6 тех, которые были приняты за базовые. Вы можете спокойно добавить новую анимацию в дерево и в Enum DialogueAnimType.

Базовой при разговоре считается состояние DialogueTalk. Это просто анимация того, как ваш персонаж что-то говорит. Каждый раз, когда используется анимация другой эмоции, после окончания её клипа идёт переход в обычный разговор, клип которого уже зацикливается, пока реплика не будет закончена.

Когда реплика персонажа закончена, какой бы ни была его эмоция, его состояние возвращается к стандартному диалоговому, когда он просто стоит безмолвно.

Прелесть в том, что это лишь схема. Сами анимации могут быть уникальными для каждого персонажа. Скрипт DialogueController не использует никаких событий анимации, но если они понадобятся, рекомендация писать обработчики либо в этот скрипт, либо делать его наследника – вариацию уже с обработчиками под конкретного персонажа.

Ниже представлен приме дерева анимаций.



4.3 Подготовка специальных объектов

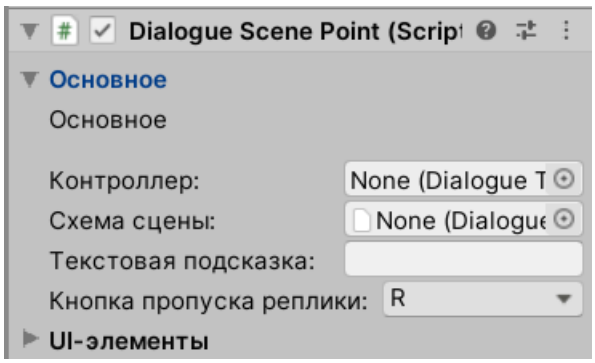
Если в диалоге будут события, отправляемые в игровую сцену (# на узле события в схеме), вам потребуются обработчики в игровой сцене.

То, какой функционал будет у этих обработчиков, зависит от потребностей проекта, но чтобы создать обработчик, нужно на объект повесить скрипт, который наследуется от DialogueEventReactor. Подробнее в пункте 1.4.4. Также в проекте есть скрипт DialogueEventReactorSample, где представлен один из способов соединения диалоговых событий с другими модулями логики.

4.4 Подготовка зоны диалога

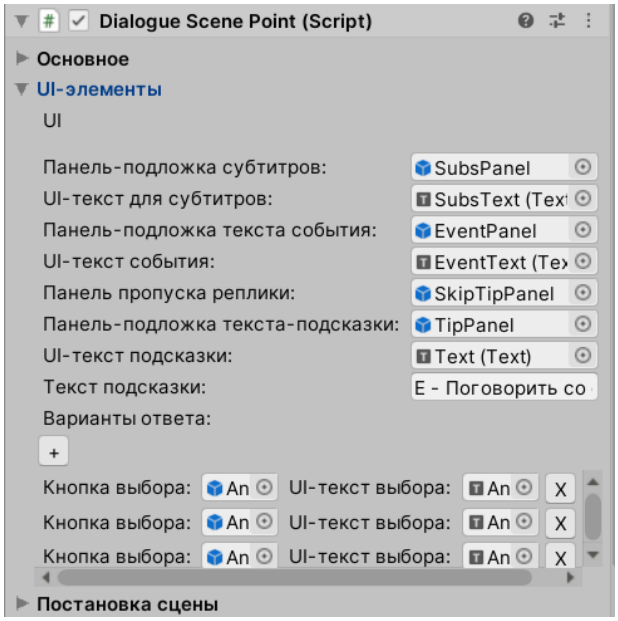
Диалоговая зона представляет собой пустую зону с триггером. Поэтому рекомендация сразу создать пустой объект с компонентом BoxCollider, помеченным ключом IsTrigger. Можно также поставить тэг DialogueScenePoint. И при входе в зону давать возможность игроку нажать кнопку и начать диалог, а при выходе, забирать. Но то, как запускать диалог – дело ваше. В коде это метод DialogueScentPoint.StartDialogue().

На созданный пустой объект с триггером следует добавить компонент DialogueScenePoint. Далее пройдёмся подробнее по его заполнению. Там есть несколько групп:



В основном нужно указать контроллер DialogueTeamDirector. Он всегда один на сцену и управляет по большей части переключением режима игры с обычного геймплея на диалог для всех игровых персонажей и камеры. Подробнее смотрите в самом скрипте.

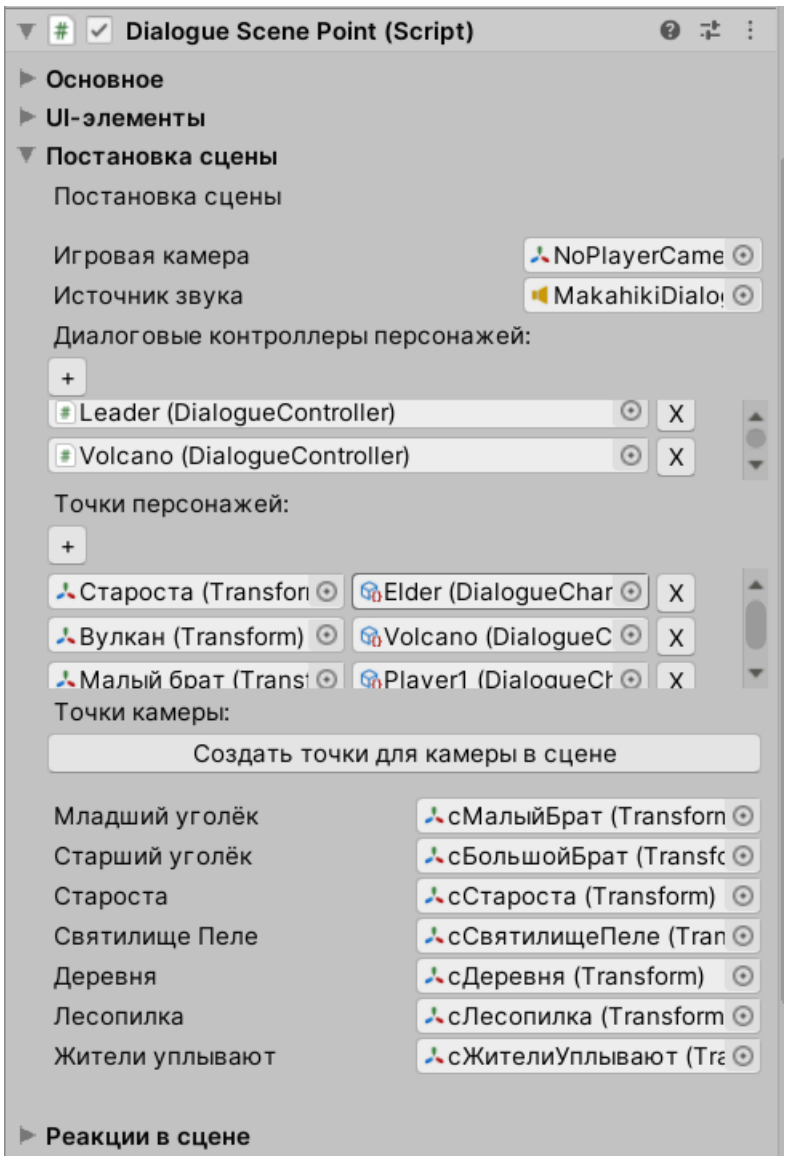
Сюда же передаётся схема диалоговой сцены, которая создавалась в редакторе, текстовая подсказка, о которой говорилось в части 4.1, выбирается кнопка для пропуска реплики. НЕ ВЫБИРАЙТЕ SPACE или ENTER. Эти кнопки срабатывают также на выделенный UI. Это будет означать что в первый раз вы сделаете выбор, кликните на кнопку. Она останется выделенной, пойдёт реплика. Когда вы пропустите реплику нажатием, к примеру, space в некоторых случаях может возникнуть ситуация, когда это нажатие будет воспринято системой событий Unity как клик по уже выделенной кнопке.



Следующая часть – это пользовательский интерфейс во время диалога. Всё это подробно было разобрано в пункте 4.1.

Стоит обратить внимание, что для верности были разделены панель подложка и сам текст.

Вы всегда передаёте объект, который должен появиться, например, панель субтитров. Предполагается, что текст с самими субтитрами уже там. Никто не мешает вам передать сам gameobject текст субтитров и сделать их без панели.



Следующая часть относится к постановке сцены.

Игровая камера будет летать по всем ракурсам, которые вы подготовили для диалога.

Источник звука используется, если диалог предполагает озвучку.

Диалоговые контроллеры – персонажи, которые участвуют в диалоге. Они, как правило находятся в сцене, если это NPC, либо должны подгружаться в скрипт программно (List actors в скрипте зоны).

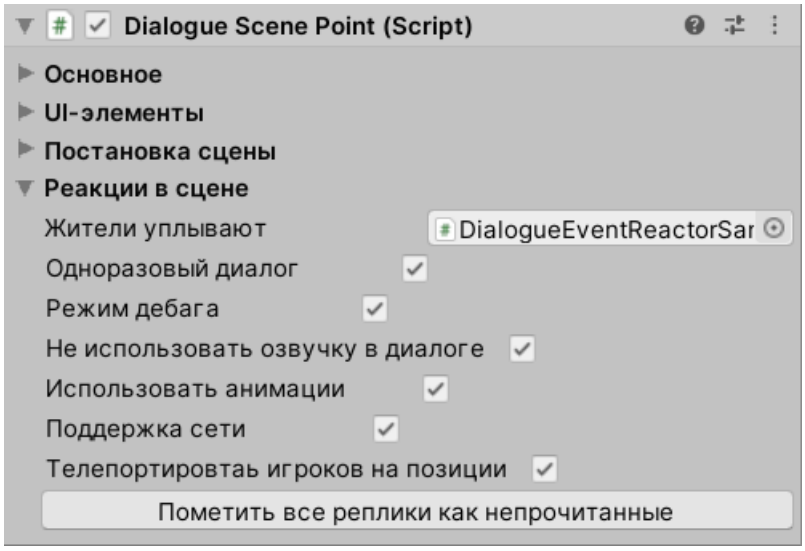
Точки персонажей – это те позиции, куда будут вставать персонажи во время диалога. По факту – пустые объекты, у которых берётся transform. То, какой персонаж куда будет вставать, вы задаёте диалоговыми ролями.

С точками камеры всё примерно также. Это пустые объекты, позиция и поворот которых берётся для того, чтобы правильно выставить камеру. На всякий случай, во время принятия какого-то из ракурсов камера встраивается в соответствующий объект как дочерний. Написав несложные скрипты для движения этих объектов или же запуска соответствующих анимаций и подключив их к диалоговым событиям, установив какое-то время фиксации, вы вполне можете сделать сцены с движением камеры, двигая объекты ракурса. Либо же просто двигать этот объект с помощью timeline как часть ролика на движке.

В разделе с реакциями всё просто. Вы в диалоговой схеме прописывали строкой названия каких-то событий. Здесь вы подключаете на эти события обработчики. О них уже говорилось подробнее в разделе 1.4.4.

Есть также ряд настроек типа назначения диалога одноразовым. В таком случае диалог сразу после окончания будет отключать свой коллайдер (чтобы не давать возможности снова запустить диалог). А через 6 секунд и вовсе

удалится (чтобы позволить отработать тексту события и снова спрятать его), чтобы не засорять вашу локацию.



Режим дебага просто включает отрисовку позиций персонажей и ракурсов камеры, чтобы было удобнее их расставлять.

Вы можете не использовать озвучку в диалоге, тогда перелистывание репликами будет происходить по клавише.

«Использовать анимации» подключает сигналы аниматорам.

Поддержка сети просто включает испускание событий, о которых будет описано в отдельном руководстве по подключению сети.

Телепортировать игроков на позиции – эта функция будет расставлять персонажей по тем позициям, которые вы им подготовили. В противном случае они останутся на своих местах. Функция специфическая, но иногда может пригодиться для сцен с репликами персонажей, которых игроки не видят, а ракурсы показывают какие-то другие места.

Кнопка «пометить все реплики как непрочитанные» очищает все реплики. Дело в том, что в систему встроен контроль над тем, в первый ли раз игроку показана реплика. Если реплика повторяется, то для случая с озвучкой можно не дослушивать и пропустить реплику, как в режиме без озвучки, по кнопке. Так вот эта кнопка снова помечает все реплики непрочитанными.

Заключение

На этом руководство подходит к концу. Помните, что данную систему можно дополнять, у нас вполне получалось внедрять в неё ролики на движке, выполненные с помощью timeline. Да и про хитрости с наездом камеры уже упомянуто. Не забывайте про использование диалоговых параметров.

Приятного пользования.