Just Logic

Введение

Just Logic – это расширение визуального программирования для Unity. С его помощью Вы можете создать логику для игры прямо в инспекторе. Just Logic может обращаться к другим скриптам и объектам, а также использовать любые другие плагины и dll библиотеки.

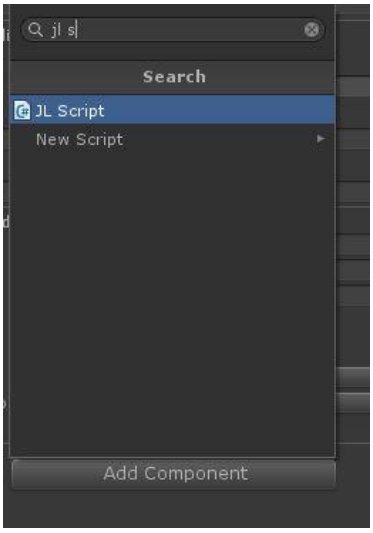
Платформы

Поддерживаются Standalone, Web Player, iOS и Android. На остальных платформах корректная работа JustLogic возможна, но не гарантируется.

Установка

Импортируйте JustLogic.unitypackage , используя меню «Assets/Import Package/Custom Package».

Блоки



Чтобы создать новый *JL Script* создайте новый объект (Game Object/Create Empty) и добавьте на него компонент *JL Script* (для этого нажмите кнопку «Add Component» в инспекторе и введите название компонента «JL Script» в окно поиска).

Скрипт строится из блоков – *действий* и *выражений*.

Каждый блок имеет свои настройки - параметры. Обычно сначала вычисляется значения параметров – *аргументы* – а после этого уже выполняется само действие. В инспекторе не обязательные параметры показаны в квадратных скобках.

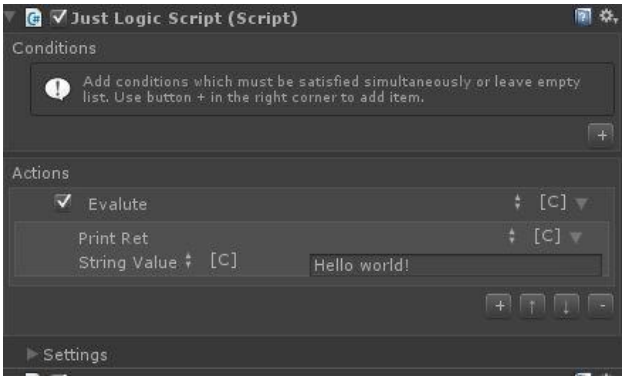
Выражение отличается от *действия* тем, что возвращает некое значение – результат своей работы. Это значение может быть тут же использовано в качестве *аргумента* для другого блока.

Примеры *действий*: переместить объект, загрузить уровень, замедлить время.

Примеры *выражений*: найти главную камеру, сравнить два числа, узнать длину строки.

	Действие	*Выражение*
Возвращает значение	-	+
Параметры могут содержать другое *действие*	+	-
Параметры могут содержать другое *выражение*	+	+
Может приостанавливать выполнение (таймеры)	+	-

Перед запуском скрипта выполняется проверка *условий*, указанных в списке *Conditions* в инспекторе. Запуск не будет произведен, если хотя бы одно из заданных *условий* не соблюдено.



Каждое *условие* представляет собой выражение, которое возвращает значение типа *bool* («да/нет» или True/False). Когда все *условия* вернули

значение True («да»), начинается выполнение скрипта.

Действия задаются в разделе *Actions* и выполняются последовательно сверху вниз. Последовательность *действий* можно объединить с помощью блока *Sequence*.

*Hint: с*лева от названия действия находится флажок, с помощью которого его можно временно отключить.

Чтобы поменять *выражение* или *действие* щелкните по его названию и выберите подходящий вариант в открывшемся окне (для удобства его можно закрепить). Если нужное выражение отсутствует в списке, возможно, его тип не соответствует параметру, значение которого Вы устанавливаете.

*Hint: б*лок можно скопировать, для этого выберите соответствующий пункт в меню блока (нажмите правой кнопкой).

Действия не доступны внутри *выражений*. Говоря математически, *выражение* – это функция, которая принимает параметры. Каждый параметр может быть другой функцией или константой, но он обязательно должен иметь значение. *Действие* не возвращает значение, поэтому его нельзя передать в параметр.

Если необходимо выполнить *выражение* без использования возвращаемого им значения (как будто это *действие*), следует упаковать его в специально предназначенное *действие Evalute* (вычислить).

События

Сам по себе скрипт запуститься не может. Обычно его запускает *обработчик* определенного *события*. Каждый *обработчик* имеет собственный задаваемый набор *условий*, что позволяет сразу проверять свойства *события*.

Общий порядок запуска скрипта:

1. Срабатывание *обработчика события*
2. Проверка *условий* для события
3. Начало запуск скрипта
4. Проверка *условий* для скрипта
5. Выполнение *действий* скрипта

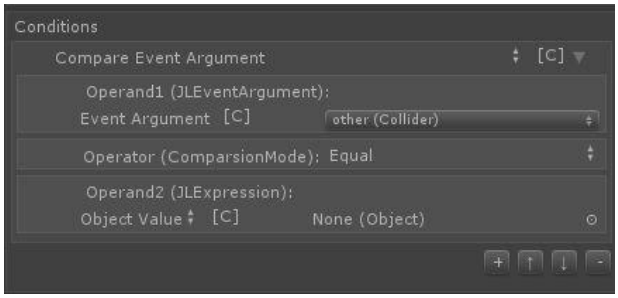
Обработчик события – компонент «JL Event Handler», который может находиться на том же объекте, что и сам скрипт, или на любом другом объекте сцены.

Параметр *Event* задает событие, при котором обработчик начнет проверку условий.

Подробнее узнать о событиях в Unity можно на сайте http://docs.unity3d.com/Documentation/Manual/EventFunctions.html

Описание основных событий доступно по адресу http://docs.unity3d.com/Documentation/ScriptReference/MonoBehaviour.html

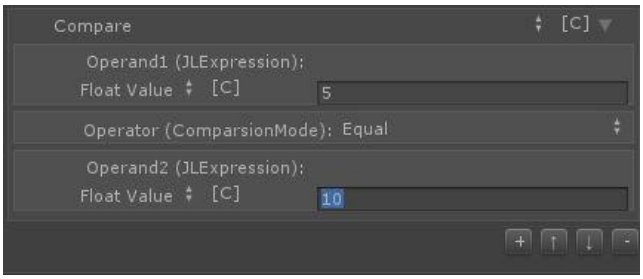
Список выражений *JLScripts* задает скрипты, которые необходимо запустить. В это поле можно перетащить и объекты, на которых находится скрипты.



В списке *Conditons* можно задать условия *обработчика*. Только если все условия соблюдены, скрипт будет запущен. Рекомендуется сначала проверять работу скрипта без указания *условий*, а только потом добавлять их.

Свойство *события* (например, other в OnTriggerEnter) можно сравнить с любым значением при помощи *выражения Event/Compare Event Argument.*

Выражение: *Compare*



Предназначено для сравнения двух значений. Возвращает *bool* значение: *True* или *False*.

От названного ранее *Compare Event Argument* это выражение отличается только тем, что его параметр Operand1 доступен для изменения.

Там где это возможно, преобразование типов происходит автоматически. Вы можете сравнить Game Object и Component, добавленный к Game Object.

Возвращаемое значение	*bool*
Параметры Operand1 и Operand2	Два сравниваемых *выражения*
Параметр Operator	Тип сравнения: равенство, неравенство, больше, меньше, больше или равно, меньше или равно.

Действие: *Branch / If*

If принимает *bool* аргумент и в зависимости от его значения выполняет одно из двух *действий*.

Hint: используйте *действие Sequence*, чтобы вместо одного действия выполнить последовательность.

Параметры:

Value	*bool выражение,*
Then	выполняется, если значение Value равно *True*,
Else	выполняется, если значение Value равно *False.*

Выражения: группа *Logical*

Описанные ниже блоки этой группы предназначены для сравнения аргументов типа *bool* (*True или False*). Сам результат сравнения так же возвращается в виде *bool* значения.

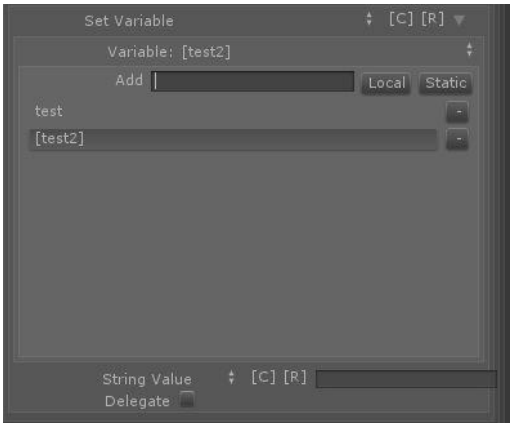
Приведенные ниже *выражения* возвращают *True*, если

And	все аргументы равны *True* или аргументы не указаны,
Or	хотя бы один из аргументов равен *True*,
Xor	только один из аргументов равен *True*,
Not	единственный аргумент равен *False* (инвертирует значение).

В остальных случаях эти *выражения* возвращают *False*.

Выражение IIf похоже на *действие If*, оно возвращает один из двух аргументов в зависимости от значения bool-аргумента Value.

Переменные



Переменные позволяют временно сохранить некоторые данные. Вы можете записать значение, чтобы использовать позже в своем скрипте. Выражение *Variable / Set* записывает данные в *переменную*, а *Variable / Get* считывает.

Hint: *Variable / Set* является выражением, поэтому только что записанное значение можно сразу передать в качестве аргумента другому блоку.

Кроме обычных переменных, которые хранят значение только во время выполнения скрипта, существует еще два вида переменных:

- *Статические переменные* сохраняют свое значение после завершения скрипта и могут быть прочитаны при его следующем запуске.

  В инспекторе название *статической переменной* отображается в квадратных скобках.

- *Глобальные переменные* работают так же, но при этом они являются общими для всех скриптов сразу. С их помощью можно передавать данные другим скриптам.

Чтобы проверить, содержит ли *переменная* значение, используйте *bool-выражение Variable / Is Set*.

При установке значения *переменной* есть возможность отложить вычисление значения до момента, когда *переменная* будет прочитана. Для этого включите опцию *Delegate* в настройках *Variable / Set*. Каждый раз при считывании значения *переменной* установленное *выражение* будет вычисляться заново. Обратите внимание, что для *глобальных переменных* эта опция не действует.

Сохраненное числовое значение переменной можно увеличивать или уменьшать на единицу с помощью выражений *Variable / Increment* и *Variable / Decrement* соответственно. Эти выражения возвращают новое значение переменной.

Действия: группа Time

Блок *SetTimeScale* устанавливает скорость игрового времени (по умолчанию 1).

Остальные *действия* этой группы позволяют приостановить выполнение скрипта на некоторое время:

Wait for game time	время ожидания игровое (указано в секундах),

Wait for real time	реальное время ожидания (указано в секундах),
Wait for next frame	ждет следующий кадр (*событие Update*),
Wait for next fixed update frame	ждет следующий кадр вычисления физики (*событие FixedUpdate*).

Блоки: группа External

Блоки этой группы предназначены для обращения к другим скриптам и плагинам.

Действие TriggerScript запускает указанный JL скрипт (независимо, так же, как и при срабатывании *обработчика события*). При этом выполнение текущего скрипта будет продолжено даже в том случае, если запускаемый скрипт начнет ожидание по таймеру (Wait For…).

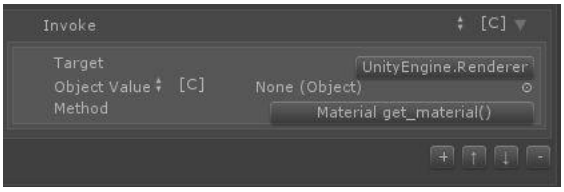
Действие Call JL Script (scene or prefab) отличается от предыдущего тем, что запускаются только *действия* указанного скрипта (под управлением выполняемого). Выполнение текущего скрипта не продолжится, пока вызываемый скрипт не будет завершен (например, в случае ожидания по таймеру). Проверка условий вызываемого скрипта вообще не вызывается.

Действие Call JL Script (asset) позволяет выполнить скрипт, сохраненный в виде ассета. Такой скрипт находится вне сцены и может вызываться из нескольких разных сцен. Чтобы создать такой скрипт, используйте меню Assets/Create/Just Logic Script.

Аналогично *выражение Call Expression (asset)* позволяет запустить выражение, сохраненное в виде ассета (создается с помощью Assets/Create/Just Logic Expression). Возвращаемое значение может быть использовано.

Выражение *Call Expression (scene or prefab)* запускает выражение, хранящееся на сцене (компонент JL Expression).

***Выражение** Invoke* позволяет вызвать любой метод указанного объекта. Так же можно указать конкретный тип и вызвать статический метод (такие методы в списке имеют квадратные скобки вокруг аргументов).



***Действие** SendMessage* вызывает метод на объекте или скрипте. Подробнее о SendMessage можно узнать здесь: http://docs.unity3d.com/Documentation/ScriptReference/Component.SendMessage.html

Блоки: группа Branch

Return	Останавливает выполнение скрипта
Return Script	Останавливает выполнение ассета JL Script
Noop	Ничего не делает
If	Действие в зависимости от значения выражения.
IIf	Возвращает одно из двух значений в зависимости от значения выражения.
Sequence	Объединяет последовательность *действий*
Evalute Expression	Выполняет *выражение*
Группа Exception	Блоки для обработки исключений

Действия: группа Loop

Блоки данной группы позволяют выполнять заданное действие несколько раз подряд.

SimpleFor	Выполняет блок указанное количество раз
For	Выполняет блок, устанавливая значение *переменной* от From до To
ForEach	Выполняет блок, заполняя значение *переменной* элементами заданного списка
While	Выполняет блок, пока соблюдены заданные условия
Continue	Переходит к следующей итерации блока группы Loop (для последовательностей)
Break	Останавливает выполнение блока группы Loop

Другие блоки

Предназначение остальных блоков соответствует их названию. Обычно документация Unity содержит описание функции с тем же названием, что и у блока. Информацию о таких функциях можно найти в документации по Unity: http://docs.unity3d.com/

Группа блоков	**Описание**
Value	*Выражения*, используемые для задавания значений заранее, в противоположность их вычислению во время выполнения скрипта.

Рекомендации по использованию

- Всегда обращайте внимание на Console. Даже если внешне все работает отлично, в консоли может быть полезная для Вас информация.
- Осторожно со ссылками на объекты (*ObjectValue*)! Если передать в выражение уничтоженный или отсутствующий объект, скорее всего, будет сгенерировано исключение и выполнение всего скрипта остановится. Все исключения появляются в окне Console.
- Некоторые объекты могут самоуничтожаться со временем. Чтобы обнаружить такие объекты, просмотрите Ваш скрипт в инспекторе во время тестирования игры.
- Во время тестирования Вы можете вручную запустить скрипт из инспектора. Для этого нажмите правой кнопкой на заголовке скрипта и выберите в контекстном меню пункт Start Execution.
- Читайте документацию Unity, включая раздел Scripting. С помощью выражения *Invoke* Вы можете использовать любые функции Unity. Отличается только сам способ создания скрипта.

Автор: Влад Таранов
Skype: vbprogr
YouTube: http://www.youtube.com/user/aqlasolutions
Web: http://www.aqla.net