ЛЕКЦИЯ 10

Блюпринты в действии 1

ЦЕЛИ И ИТОГИ ЛЕКЦИИ

Goals

Цели этой лекции:

- Показать трансформации
- Объяснить что такое мировые координаты
- Продемонстрировать разницу между относительными и мировыми преобразованиями.
- Показать, как использовать векторные операции
- Показать некоторые векторные функции

Outcomes

К концу этой лекции вы сможете:

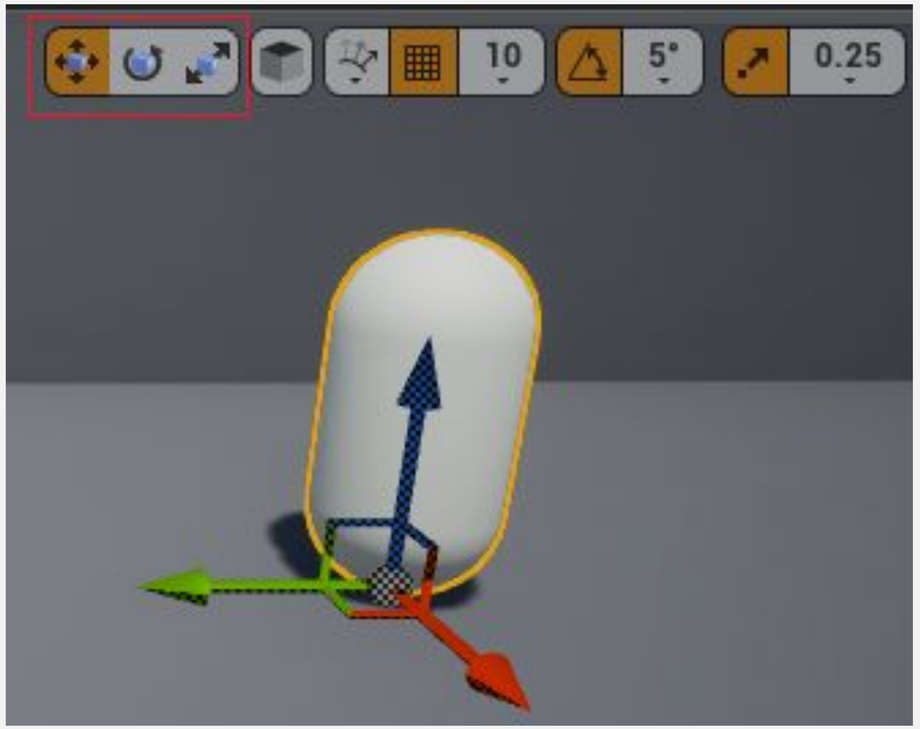
- Использовать векторы для обозначения местоположения и движения
- Различать мировые и локальные координаты
- Использовать векторные операции и функции

ТРАНСФОРМАЦИЯ

Каждый Актор имеет три свойства **Transform** которые представляют его положение в мире, вращение и масштаб, как показано на верхнем изображении справа.

Вы можете изменить трансформации Актора с помощью панели Details или с помощью гизмо в редакторе Level Editor.

В Level Editor есть три кнопки для выбора типа трансформации для применения к Актору. Картинка справа снизу показывает эти кнопки в красной рамочке.

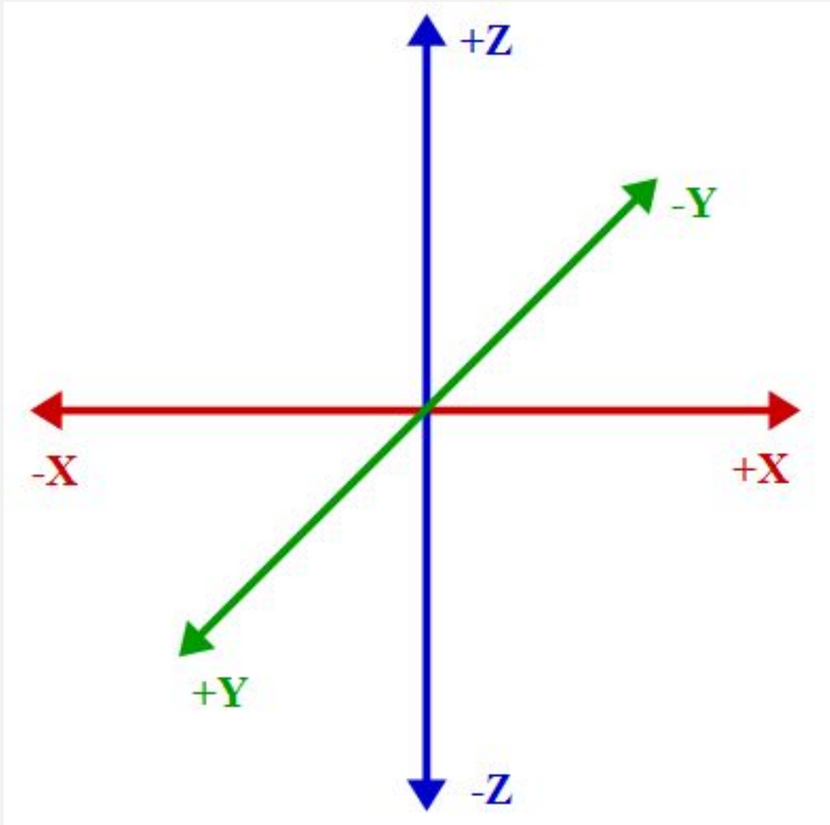


МИРОВЫЕ КООРДИНАТЫ

3D пространство представляется тремя осями: **X**, **Y**, и **Z**.

Есть разные способы организовать эти оси. Unreal Engine использует подход, показанный на рисунке справа.

Любое положение в трехмерном пространстве может быть представлено с помощью набора значений для **X**, **Y** и **Z**, указывающих положение на каждой оси. Эти значения хранятся в переменной **Location**, которая является частью структуры преобразования, и они определяют то, что известно как **местоположение в мире**.



ФУНКЦИИ LOCATION

По умолчанию единица измерения Unreal (uu) равна 1 сантиметру. Чтобы использовать значения переменной **Location** в Blueprint, можно использовать следующие функции:

- **GetActorLocation**: Возвращает текущую позицию Актора.
- **SetActorLocation**: Устанавливает новую позицию для Актора.
- **AddActorWorldOffset**: Использует значения параметра **Delta Location** для изменения текущего положения Актора.

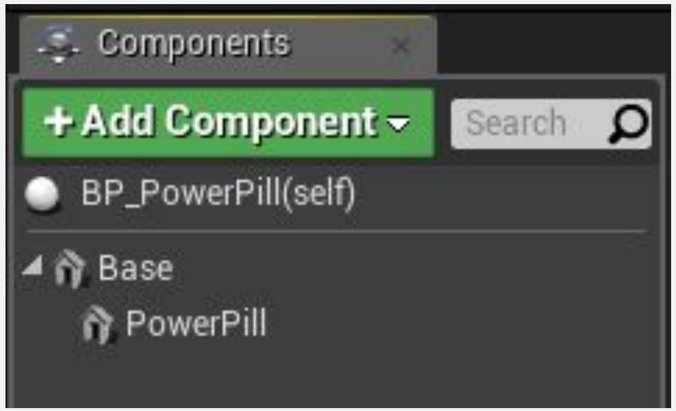
ОТНОСИТЕЛЬНАЯ ТРАНСФОРМАЦИЯ

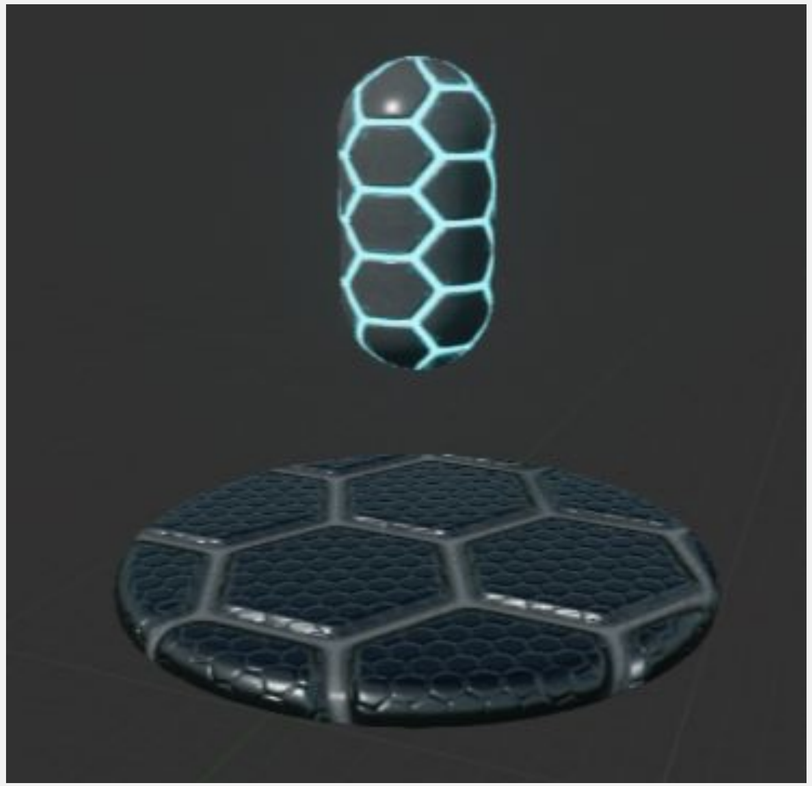
Есть еще одна концепция, известная как **относительная транформация**.

Чтобы понять эту концепцию, возьмите в качестве примера Blueprint с двумя компонентами: **базовый** компонент, который является корневым компонентом, и компонент **PowerPill**, как показано на изображении справа.

Значения **Location** компонента **PowerPill** **X** = “**0**”, **Y** = “**0**”, **Z** = “**70**”.

Положение компонента **PowerPill** относительно положения корневого компонента, поэтому при перемещении **базового** компонента компонент **PowerPill** будет следовать за движением корневого компонента, поскольку он всегда должен находиться на расстоянии 70 см от базового компонента по оси Z.





ФУНКЦИИ ОТНОСИТЕЛЬНОЙ ТРАНФОРМАЦИИ

Положение компонента **PowerPill** можно получить двумя способами: его относительное расположение, которое в этом примере равно (0, 0, 70); или его мировое положение, которое в данном случае является мировым положением родительского компонента (**базового** компонента) плюс относительное положение компонента **PowerPill**.

Положение компонента также можно определить двумя способами, используя функции **SetRelativeLocation** и **SetWorldLocation**. Функция **SetRelativeLocation** определяет новое положение компонента **PowerPill** относительно положения корневого компонента, а функция **SetWorldLocation** получает в качестве входного параметра мировую координату и определяет положение компонента **PowerPill** так, чтобы сумма положения **базового** компонента и положение компонента **PowerPill** равно указанным мировым координатам.

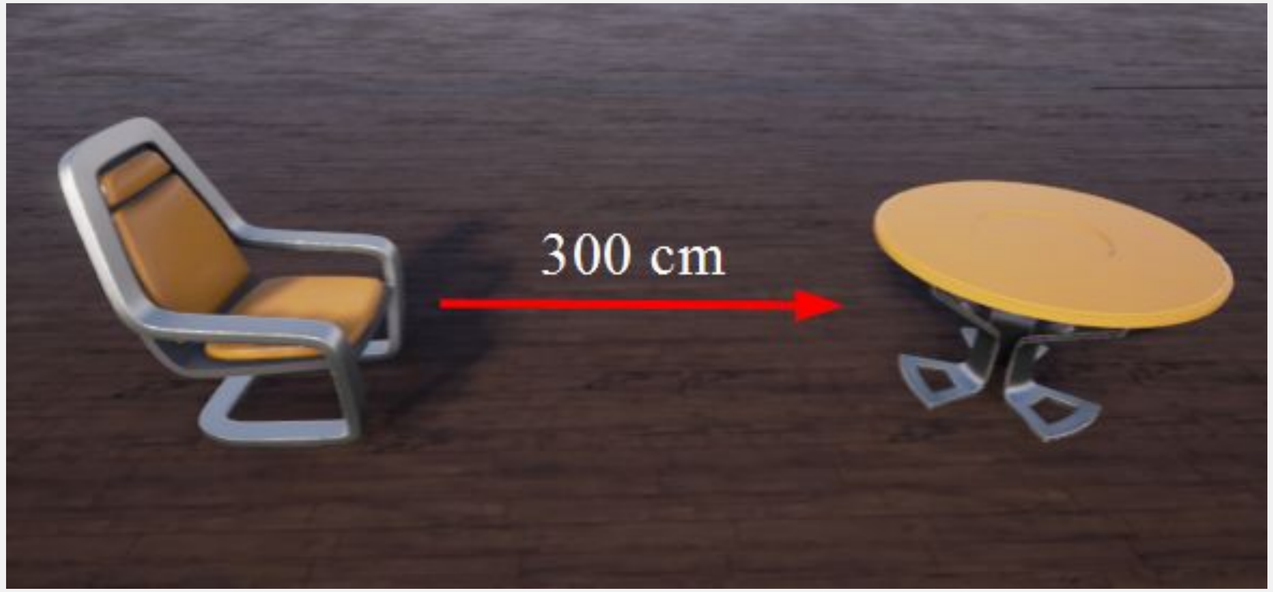
ТОЧКИ И ВЕКТОРЫ

Вектор представлен в Unreal Engine как структура, которая содержит значения с плавающей запятой X, Y и Z.

Эти значения можно интерпретировать по-разному. Один из способов использования вектора - это представление точки (или положения) в трехмерном пространстве. Например, у каждого Актора есть переменная «**Location**», которая возвращает значение вектора.

Векторы также используются для представления движения. В примере справа, чтобы направить робота от стула к столу, необходимы две части информации: направление, в котором робот должен двигаться, и расстояние.

И направление, и расстояние можно собрать в вектор: «300, 0, 0».



СЛОЖЕНИЕ ВЕКТОРОВ

Сумма двух векторов определяется сложением их элементов.

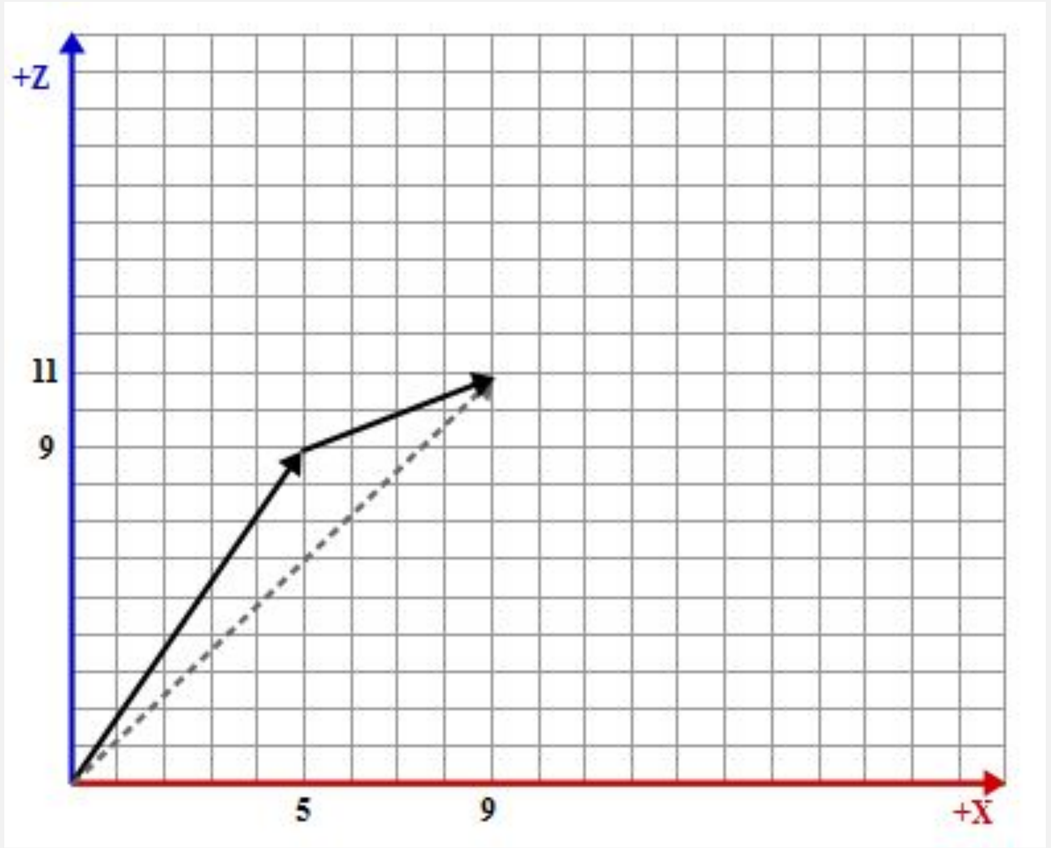
Пример:

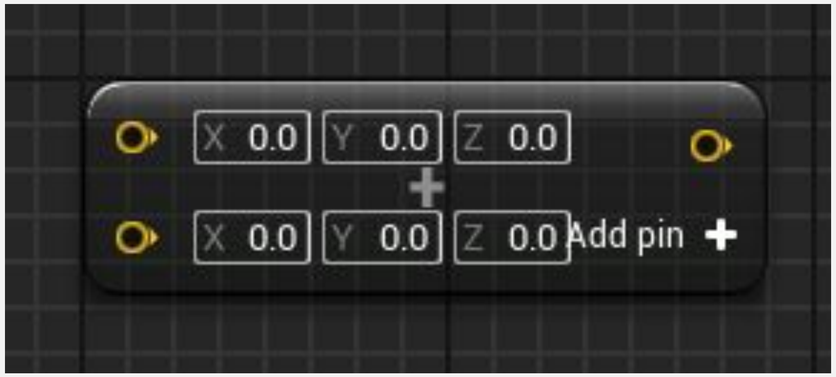
V1 = (5, 0, 9) и V2 = (4, 0, 2)

V1 + V2 = (5 + 4, 0 + 0, 9 + 2)

V1 + V2 = (9, 0, 11)

Картинка снизу справа показывает Blueprint оператор для суммы векторов.





ВЫЧИТАНИЕ ВЕКТОРОВ

Вычитание одного вектора из другого определяется вычитанием каждого элемента друг из друга:

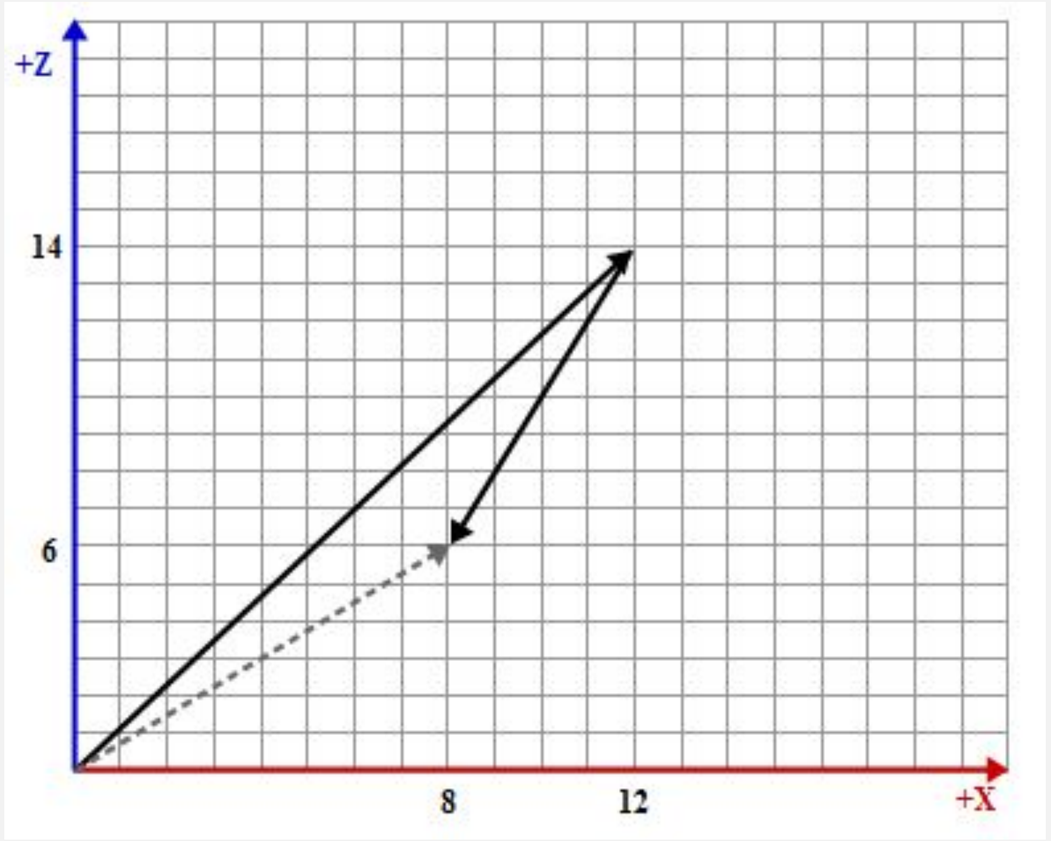
Пример:

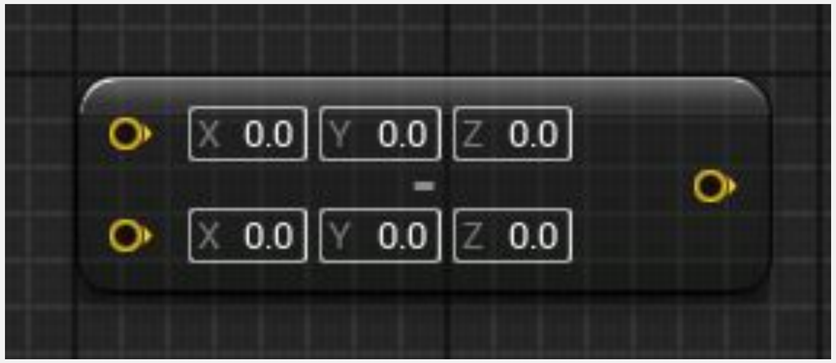
V1 = (12, 0, 14) и V2 = (4, 0, 8)

V1 – V2 = (12 – 4, 0 – 0, 14 – 8)

V1 – V2 = (8, 0, 6)

Картинка справа показывает оператор Blueprint для вычитания векторов.





ДЛИНА ВЕКТОРА

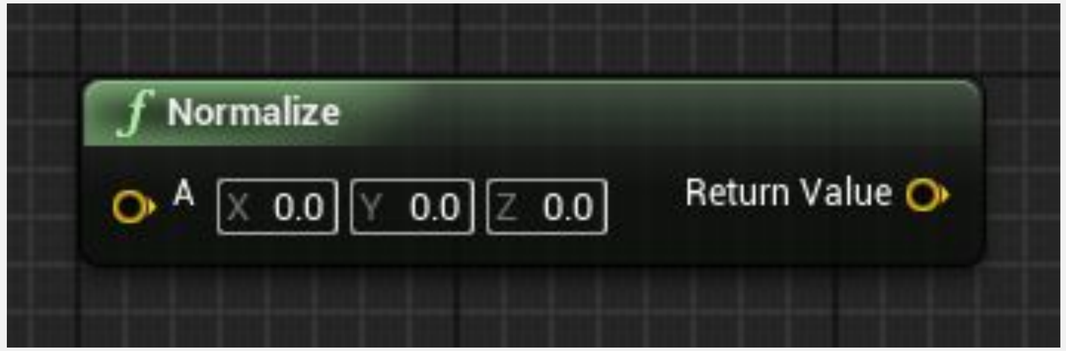
Длину или величину вектора можно рассчитать с помощью функции Blueprint, показанной на изображении справа.

Значение можно использовать для представления расстояния между двумя точками.

НОРМАЛИЗАЦИЯ ВЕКТОРОВ

Векторная нормализация используется для нахождения **единичного** вектора. Единичный вектор имеет длину, равную «1».

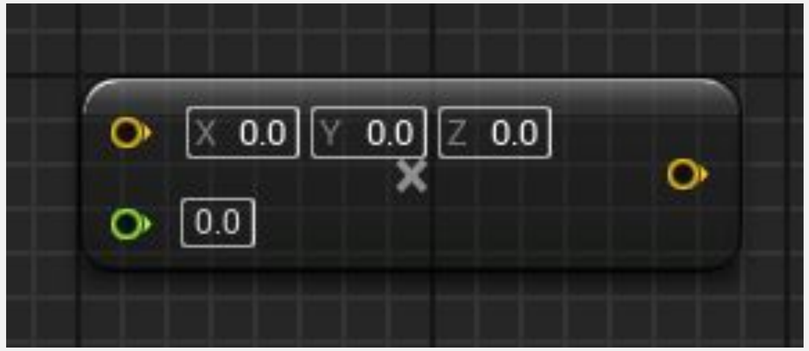
Нормализованный вектор часто используется, когда нужно указать только направление. В некоторых вычислениях следует использовать только нормализованный вектор.



СКАЛЯРНОЕ УМНОЖЕНИЕ ВЕКТОРОВ

Умножение вектора на скалярное значение выполняется путем умножения каждого из его элементов на скалярное значение.

Эта операция изменяет длину вектора.

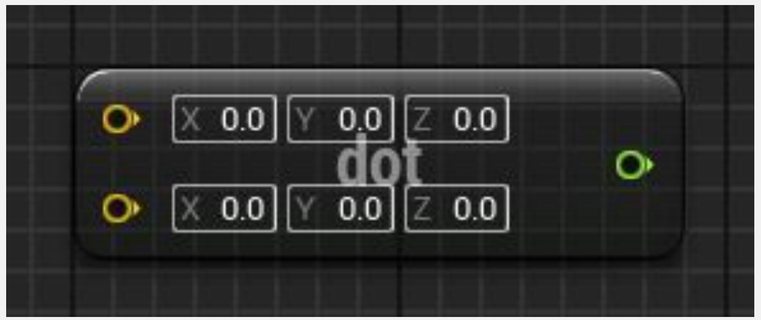


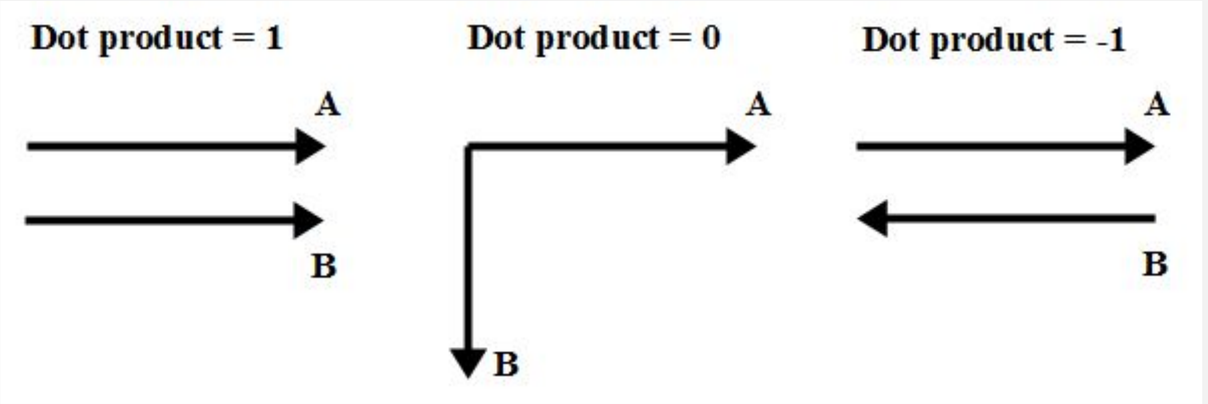
СКАЛЯРНОЕ ПРОИЗВЕДЕНИЕ

Скалярное произведение можно использовать для проверки взаимосвязи между двумя векторами, например, перпендикулярны они или параллельны.

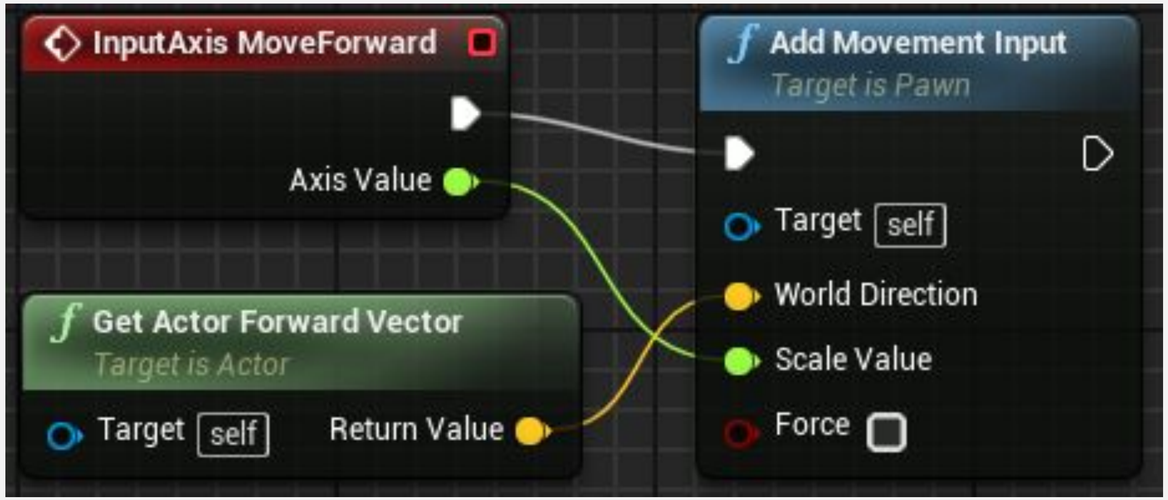
Если два вектора нормализованы, скалярное произведение равно косинусу угла, образованного между векторами, и может находиться в диапазоне от –1 до 1.

На изображении справа показаны некоторые примеры скалярного произведения двух векторов.





ФУНКЦИЯ GET ACTOR FORWARD VECTOR

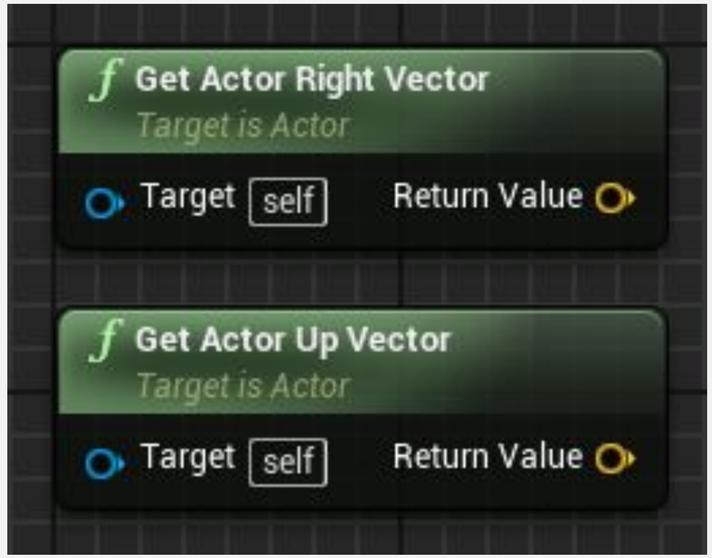


Функция **Get Actor Forward Vector** возвращает нормализованный вектор (длина = 1), который представляет направление, в котором указывает актор.

На верхнем изображении справа событие **InputAxis MoveForward** использует функцию **Get Actor Forward Vector** для перемещения игрока вперед или назад в зависимости от значения параметра **Axis Value**.

Например, используя стандартные клавиши «**WASD**» для перемещения, нажатие клавиши «**W**» устанавливает значение параметра **Axis Value** события **InputAxis MoveForward** на «**1.0**», а нажатие клавиши «**S**» устанавливает значение на **«- 1.0»**, чтобы изменить направление..

Есть две похожие функции, которые предоставляют векторы, представляющие другие направления: функция **Get Actor Right Vector** и функция **Get Actor Up Vector**, как показано на нижнем изображении.



ИТОГ

В этой лекции объяснялись преобразования, векторы и мировые координаты.

Также было показано, как использовать векторные операции и функции.