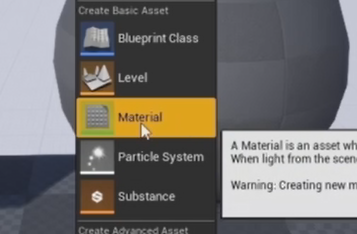
1. В данном уроке мы создадим наш первый материал. Зададим ему цветовой параметр и научимся в C++ менять данный параметр динамически. Для того чтобы создать новый материал в Unreal Engine, в контент браузере кликаем правой кнопкой мыши и в меню выбираем материал:



Зададим имя нашему материалу M\_BaseColor. Он будет отвечать за цвет нашего меша.

2. Кликаем два раза на нашем материале, открывается material editor - это специализированный редактор в Unreal Engine, который отвечает за обработку параметров материала.

3. Материал - это специальный вид ассета в unreal engine, который отвечает за визуальную часть: как будут выглядеть static mesh, skeletal mesh, система частиц и другие объекты. Если быть точнее, то материал это shader - мини подпрограмма, которая работает на GPU и вычисляет цвета и положение пикселей в нашей сцене.

4. Давайте добавим параметр цвета нашему материалу. Для этого в произвольной области блюпринта кликаем правой кнопкой мыши и в поиске вводим vector. Листаем список наверх. В категории constant выбираем вектор с четыремя компонентами:

Изображение выглядит как текст

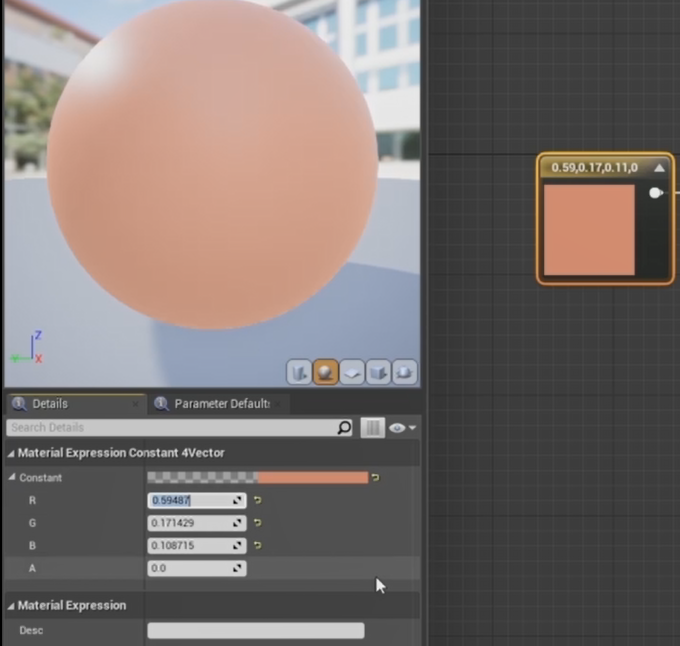
Автоматически созданное описание

Данная нода отвечает за цвет. Первые 3 компонента это rgb и четвертый - альфа. Соединяем ниточку блюпринта с Base Color:

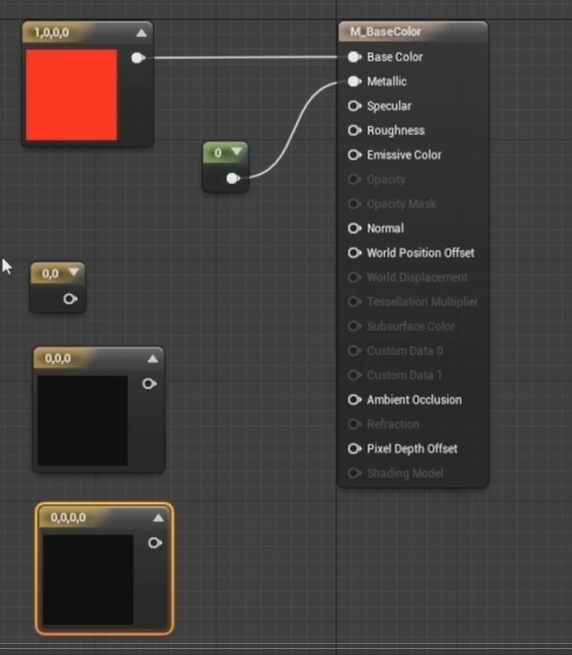
Изображение выглядит как текст, электроника, микроволновая печь, кухонный прибор

Автоматически созданное описание

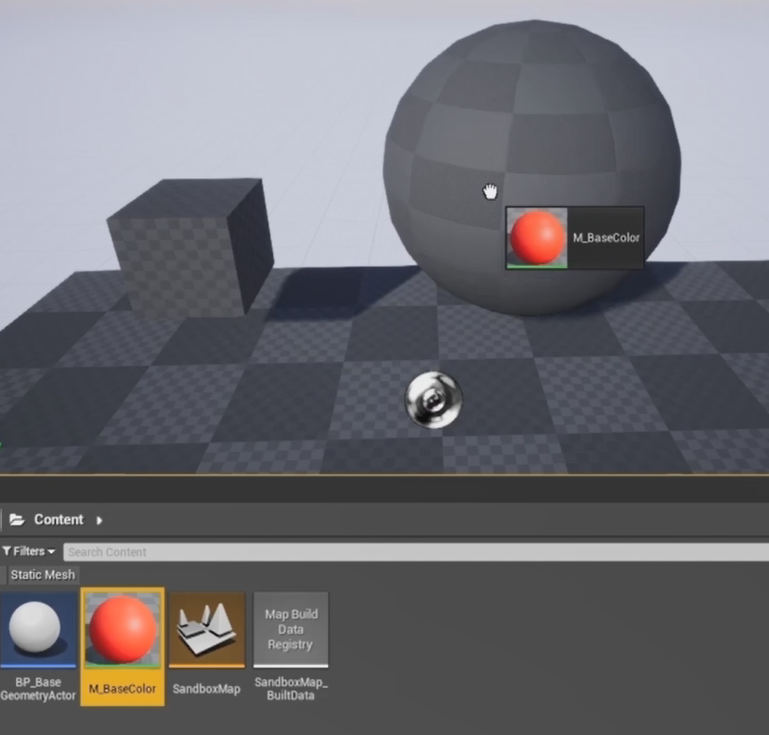
В details panel мы можем настроить цвет, кликнув на constant. Открывается color picker. И давайте выберем оттенок красного. Либо мы можем редактировать компоненты вектора через input поля:

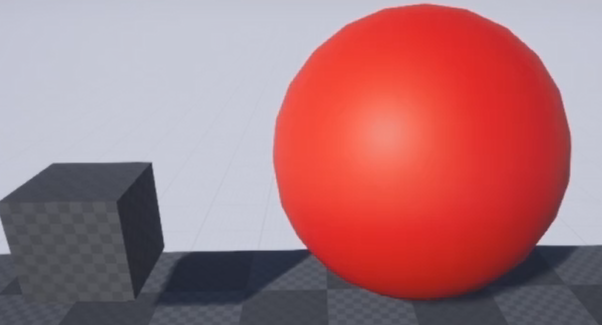


Чтобы быстро добавить константу или векторные ноды на сцену, мы можем использовать клавиатуру. Зажимаем единицу на клавиатуре и кликаем в произвольной области блюпринта. Добавляется константная нода. Мы её, например, можем подключить в металлик. Если кликаем с зажатой двойкой, то вектор с двумя компонентами, с тройкой с тремя и с четверкой появляется наша 4-х компонентная нода:

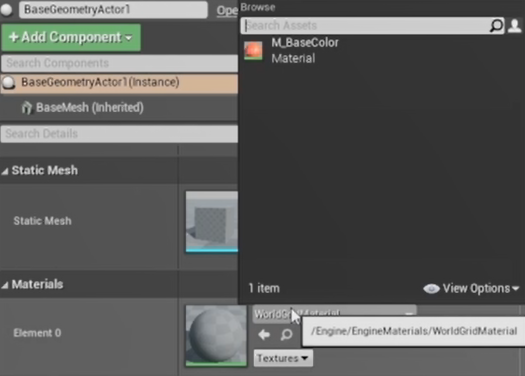


5. Нажимаем apply, save и возвращаемся в карту. Мы уже можем пользоваться нашим материалом. Для этого достаточно drag&amp;drop перетащить его на какой-либо объект на сцене, например, на наш static mesh:





Либо кликнув на акторе, в категории materials выбрать наш материал:



6. Давайте наш цвет превратим в параметр. Для этого на ноде кликаем правой кнопкой мыши и в меню выбираем convert to parametr:

Изображение выглядит как текст, табло, снимок экрана

Автоматически созданное описание

Называем наш параметр color, нажимаем apply, save и возвращаемся в карту.

7. Давайте создадим другой вид асета, который называется materialinstance. Для этого на нашем материале кликаем правой кнопкой мыши и выбираем create material instance:



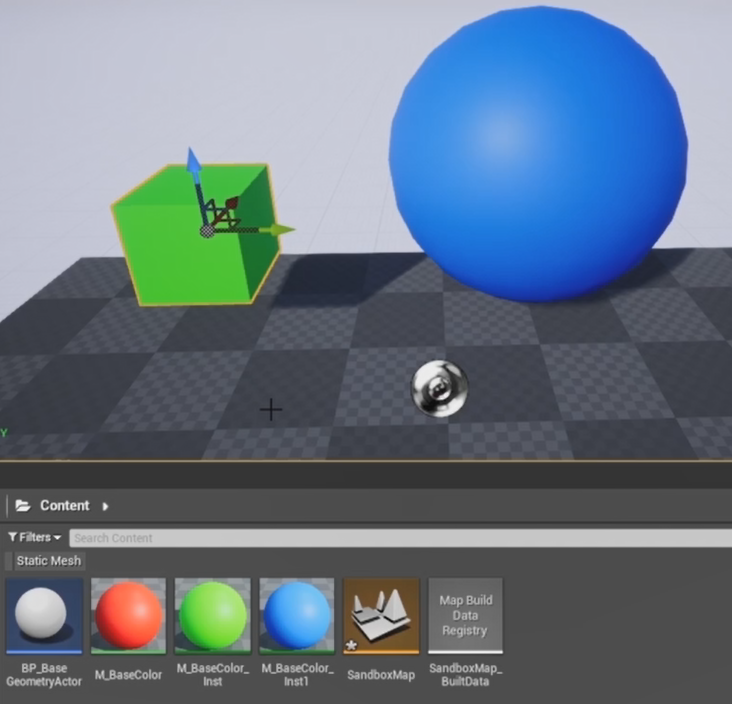
Кликаем два раза на данном ассете. Открывается редактор материал инстанса. Данный вид ассета является экземпляром материала. Он создан для того, чтобы могли настроить параметры материала, которые выведены наружу. В данном случае в панели details у нас только один параметр, который мы создали – color:

Изображение выглядит как текст, снимок экрана, монитор

Автоматически созданное описание

Кликаем на него и меняем цвет на зеленый. Сохраняем инстанс.

8. Мы можем продублировать данный asset, кликнув на нем и нажав Ctrl-W. Данная комбинация работает со всеми блюпринтами. Открываем наш очередной инстанс и меняем ему цвет на голубой. Таким методом мы можем создать набор цветов, используя лишь один материал:



9. Но данным подходом мы, конечно же, пользоваться не будем. Мы с вами научимся получать доступ к материалу в C++ и устанавливать цвет, который мы захотим. Возвращаемся к коду. Давайте получим доступ к нашему материалу, который засетапили в едиторе и поменяем значение его параметра color. Делается это крайне просто. Нам понадобится всего лишь две строчки кода. Мы воспользуемся функцией, которая есть у каждого примитивного компонента, от которого наследуется наш статик меш компонент. Она называется CreateAndSetMaterialInstanceDynamic. Функция принимает один аргумент - индекс материала. У нас с вами один материал у статик меша, поэтому указываем 0. На основе обычного материала она создает материал инстанс, устанавливает его компоненту и возвращает на него указать. По сути аналог того, что мы с вами делали в едиторе. Создаем указатель на объект, который возвращает данная функция. Указатель имеет тип UMaterialnstanceDynamic. Назовем его DynMaterial:

Изображение выглядит как текст

Автоматически созданное описание

Также подключим заголовочный файл, где объявлен данный тип #include "Materials/MaterialInstanceDynamic.h":

Изображение выглядит как текст

Автоматически созданное описание

10. Далее нам нужнопроверить валидность нашего указателя, создался объект или нет. Проверяем на null пишем if(DynMaterial) И если указатель у нас нулевой, то воспользуемся функцией класса UMaterialInstanceDynamic, которая называется SetVectorParameterValue. Данная функция имеет два аргумента. Первый аргумент имя параметра в материале. Мы с вами назвали его color и второй параметр эта структура FLinearColor. Данная структура также отвечает за определение цвета. Мы с вами уже использовали FColor, когда печатали сообщение на экране. Отличие данных типов в том, что FLinearColor это 32-битный цвет, а FColor 8-битный. У FLinearColor также есть предопределенные цвета:

Изображение выглядит как текст

Автоматически созданное описание

Если мы нажмём с вами f12, то увидим данные список:

Изображение выглядит как текст, табличка

Автоматически созданное описание

Воспользуемся жёлтым цветом.

11. Сохраняемся и запускаем editor. Нажимаем play. Каждому материалу установился желтый цвет, вне зависимости от динамических материалов, которые на них сейчас находятся. Мы в коде проставляем значение параметра color = yellow:

Изображение выглядит как мяч

Автоматически созданное описание

12. Давайте немного доработаем код. Создадим property, которая будет отвечать за цвет нашего статик меша по умолчанию. Создаем новую переменную в нашей структуре типа FLinearColor. Назоваем её color. И присвоим ей значение по умолчанию FLInearColor::Black. Делаем переменную color property. Копируем макрос UPROPERTY сверху переменной и меняем название категории на дизайн:

Изображение выглядит как текст

Автоматически созданное описание

13. Переходим в .cpp файл. И в SetVectorParameterValue меняем второй параметр с yellow на значение переменной из нашей структуры BaseGeometryData:

Изображение выглядит как текст

Автоматически созданное описание

Давайте оформим весь наш код в функцию. Для этого создаем в .h файле функцию SetColor. Она будет иметь один параметр FLinearColor по константной ссылке. Мы будем передавать цвет нашего материала через этот параметр:

Изображение выглядит как текст

Автоматически созданное описание

Нажимаем правой кнопкой мыши на функции. Выбираем quick actions and refactoring, create declaration/definition. В .cpp файле создалось тело функции. Переносим код, который мы написали на BeginPlay в данную функцию. Заменим второй параметр функции SetVectorParameterValue на просто color - параметр нашей функции SetColor:

Изображение выглядит как текст

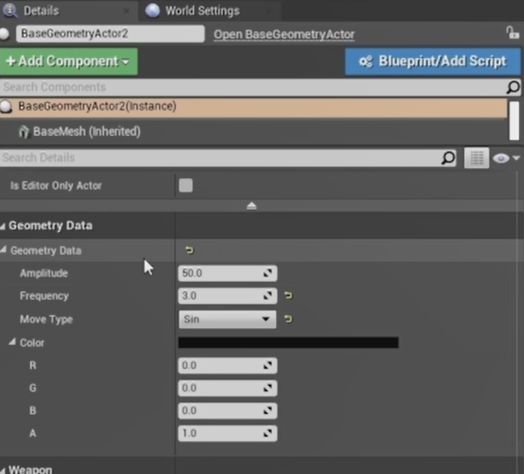
Автоматически созданное описание

И вызываем на BeginPlay функцию SetColor с параметром GeometryData.Color:

Изображение выглядит как текст

Автоматически созданное описание

14. Запускаем build, компилируем наш проект. Нажимаем play. Статик меши у нас по умолчанию черные, потому что значение параметра color - black. Но теперь мы можем контролировать данный параметр. Переместимся в категорию GeometryData нашего актора. Переменная color отображается также как в материале:



Мы можем покомпонентно редактировать данный вектор, либо нажав на него вызвать color picker и проставить нашим акторам цвета. Запускаемся. Цвета у статик мешей поменялись на те, которые мы указали в property.