1. В данном уроке мы с вами создадим таймер, проинициализируем его на BeginPlay. Он у нас будет срабатывать каждые N секунд и на каждое срабатывания таймера у нас будет меняться цвет нашего материал. Проведем небольшую подготовительную работу перед созданием таймера. Вструктуре GeometryData давайте создадим новую переменную типа float. Назовем её TimerRate. Данная переменная будет отвечать за частоту срабатывания нашего таймера. TimerRate будет измеряться в секундах. Добавим макрос uproperty сверху переменной, делаем ей expose в editor:

Изображение выглядит как текст

Автоматически созданное описание

2. Далее создаем дескриптор таймера - переменную, которая будет однозначно идентифицировать наш таймер. Данная переменная имеет тип FTimerHandle. Назовем ее TimerHandle. С помощью данного дескриптора мы будем иметь доступ к таймеру - можем его поставить на паузу, или остановить, например. Создаем функцию OnTimerFired:

Изображение выглядит как текст

Автоматически созданное описание

Данная функция будет вызываться каждый раз, когда будет срабатывать таймер.

3. Создаем также тело функции в .cpp файле. Последний подготовительный момент. В .cpp файле наверху добавляем заголовочный файл, который называется TimerManager:

Изображение выглядит как текст

Автоматически созданное описание

4. Переходим в функцию BeginPlay. и в самом низу пишемGetWorldTimerManger. Данная функциявозвращает объект, который называется таймер менеджер. Мы воспользуемся функцией SetTimer объекта таймер менеджер. Данная функция проинициализирует нам наш таймер. Первый аргумент функции - это ссылка на дескриптор таймера. Мы передаем туда нашу переменную TimerHandle. Второй аргумент - это указатель на объект, на котором мы хотим вызывать функцию каждый раз, когда срабатывает таймер. Мы будем вызывать функцию нашего актора, поэтому указываем this. Thisв C++ означает указатель на текущий объект. Третьим аргументом идёт ссылка на функцию, которая будет вызываться каждый раз, когда срабатывает таймер. Мы указываем нашу функцию OnTimerFired. Четвертый аргумент - это частота срабатывания таймера в секундах. Мы воспользуемся переменной из GeometryData - TimerRate. И последний аргумент. Он означает зацикленный таймер или нет. Поставим true, если бы мы поставили false, то таймер сработал бы один раз и остановился:

Изображение выглядит как текст

Автоматически созданное описание

Для некоторых из вас может данная конструкция показаться сложной с точки зрения синтаксиса. Я под лекцией оставлю ссылку на биндинг функций в C++. Можете самостоятельно почитать про это. Если говорить простым языком, то таймер будет вызывать функцию нашего объекта, когда прошло указанное нами количество секунд.

5. Переходим в функцию OnTimerFired и запрограммируем в ней смену цвета материала. Для этого будем использовать уже существующую функциюSetColor. Передадим туда переменную NewColor и создадим данный цвет рандомно. Создаем данную переменную. Она будет иметь тип FLinearColor. Сгенерировать рандомный цвет в unreal engine очень просто. За нас уже все запрограммировали. Просто вызываем статическую функцию FLinearColor, которая называется MakeRandomColor. Давайте также сгенерированныйцвет выведем в консоль. У FLinearColor (как и у большинства типов unreal engine) имеется функция ToString, которой мы и воспользуемся при выводе нашего цвета:

Изображение выглядит как текст

Автоматически созданное описание

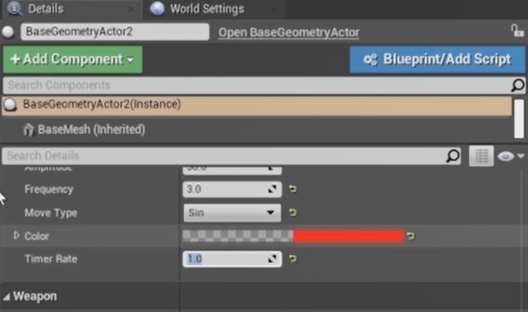
6. Рассмотрим небольшой момент. В нашей функции SetColor имеется вызов CreateAndSetMaterialInstanceDynamic. Мы можем безопасно её вызывать сколько угодно раз. Давайте провалился внутрь, нажмем F12. Сначала данная функция проверяет не существует ли уже динамический материал, если он существует, она просто возвращает указатель на него:

Изображение выглядит как текст

Автоматически созданное описание

То есть в нашем вызове SetColor по таймеру нет никакой утечки памяти. При первом вызове функции SetColor динамический материал создастся, а в последующих вызовах будет использоваться указатель на него.

7. Итак мы готовы посмотреть, что у нас получилось. Запускаем editor. Давайте удалим один actor со сцены,чтобы не путаться где чей цвет. Переходим в details panel оставшегося актора и давайте поменяем TimerRate. Поставим одну секунду:



Запускаемся. Наш код работает. Актор каждую секунду меняет цвет и данный цвет логируется в output лог.

8. Давайте будем останавливать таймер после того как он отработал N раз. Cоздадим два дополнительных члена класса. Первый член – это константа. Она будет отвечать за количество раз, сколько нужно отработать нашему таймеру перед тем, как мы его остановим. А второй – это счетчик срабатываний нашего таймера. Назовем его TimerCount и присвоим значение равное нулю:

Изображение выглядит как текст

Автоматически созданное описание

9. В функции OnTimerFired проверим условие, что TimerCount меньше максимального количества срабатываний таймера. Также сразу добавим префиксный инкремент TimerCount. При данном условии мы будем менять цвет нашего материала, поэтому перенесём весь код в блок if. И дополнительно залогируем TimerCount. Если значение переменной TimerCount стало больше, чем значение MaxTimerCount, то мы будем останавливать наш таймер. Для этого также воспользуемся глобальным таймер менеджером и вызовем его функцию, которая называется ClearTimer, которая принимает дескриптор таймера - TimerHandle. Давайте также добавим в лог остановку таймера. Запускаем едитор:



Нажимаем кнопку play и смотрим в лог. После пяти раз таймер остановился и цвет у нас больше не меняется.