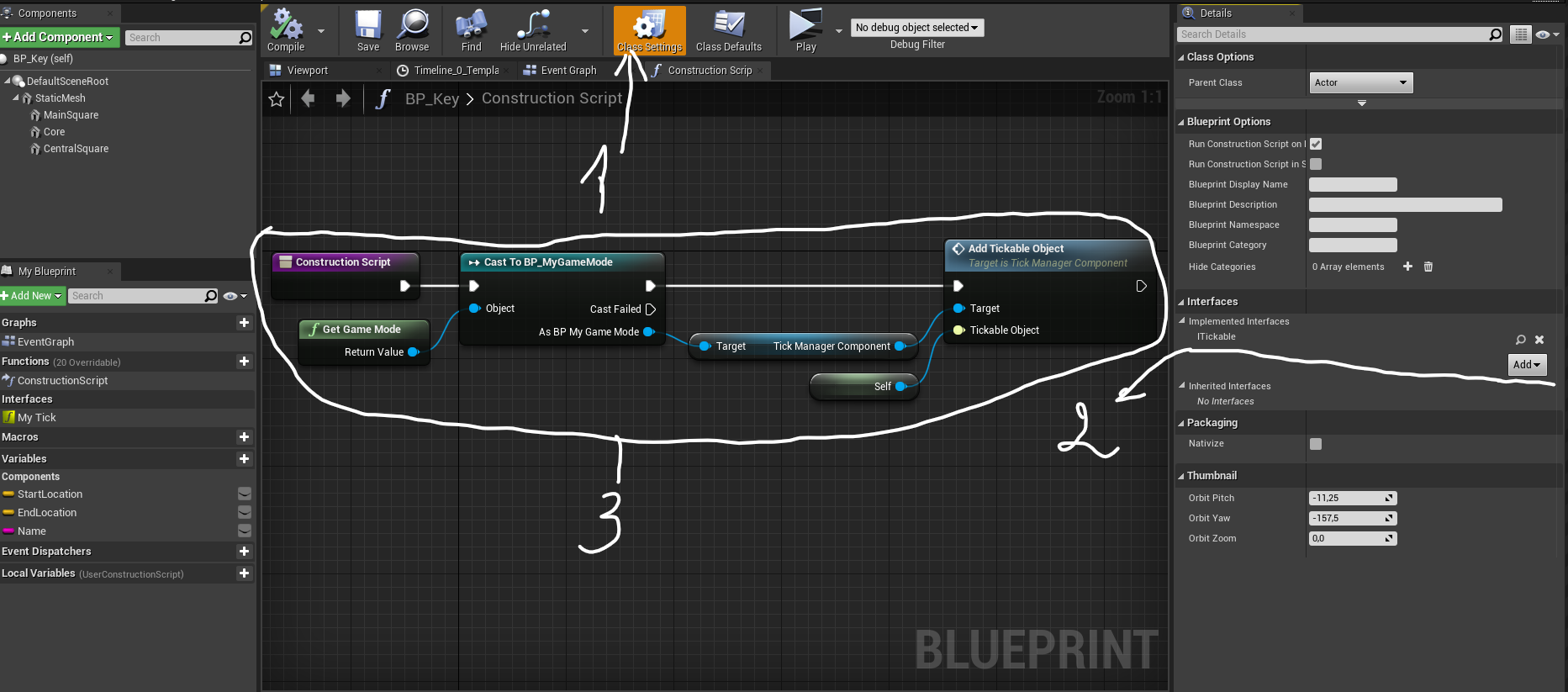
**TextColor – не трогать**

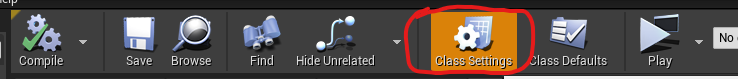
**TextColor – ожидает создания**

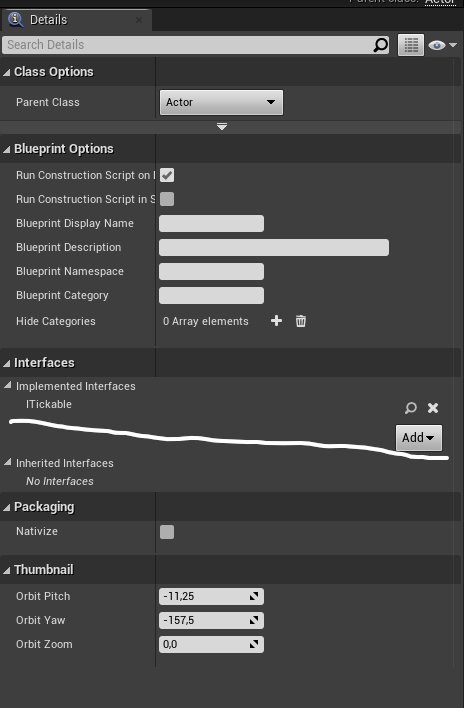
**TextColor – под вопросом**

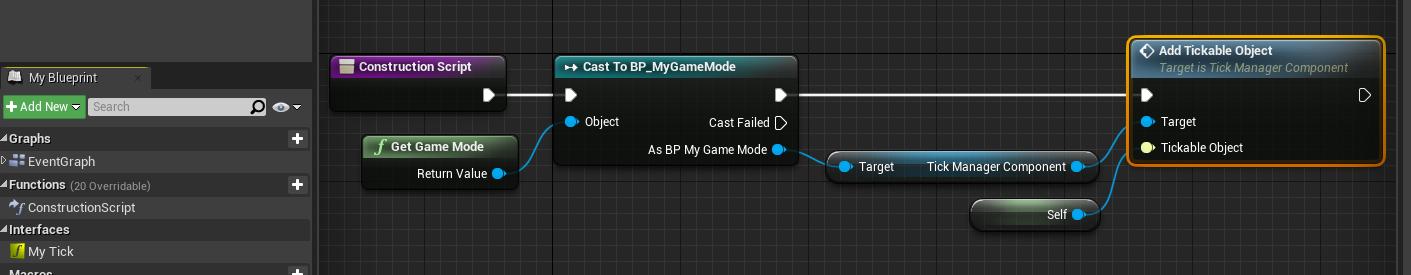
**TickManager –** организация всех объектов, которые используют метод Tick. Для того чтоб избежать повторного использования Tick на обьектах. Один Tick на всю игру.

Хранит в себе список всех обьектов которые могут выполняться каждый кадр. Поэтому тот Blueprint, который нужно чтоб выполнялся каждый кадр, в нем нужно реализовать интерфейс ITickable, так же нужно добавить этот Blueprint в массив **ITickableGroup** с помощью метода **AddTickableObject** (Желательно делать это в ConstructionScript (на этапе создания объекта) или в Event BeginPlay, если объект не создается динамически)

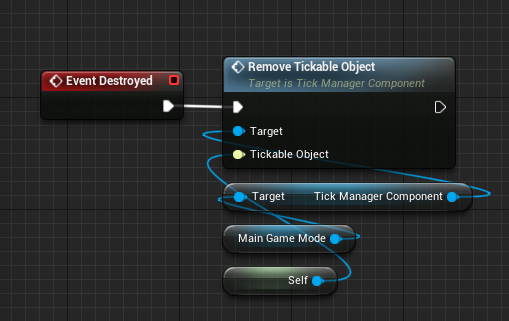
****

****

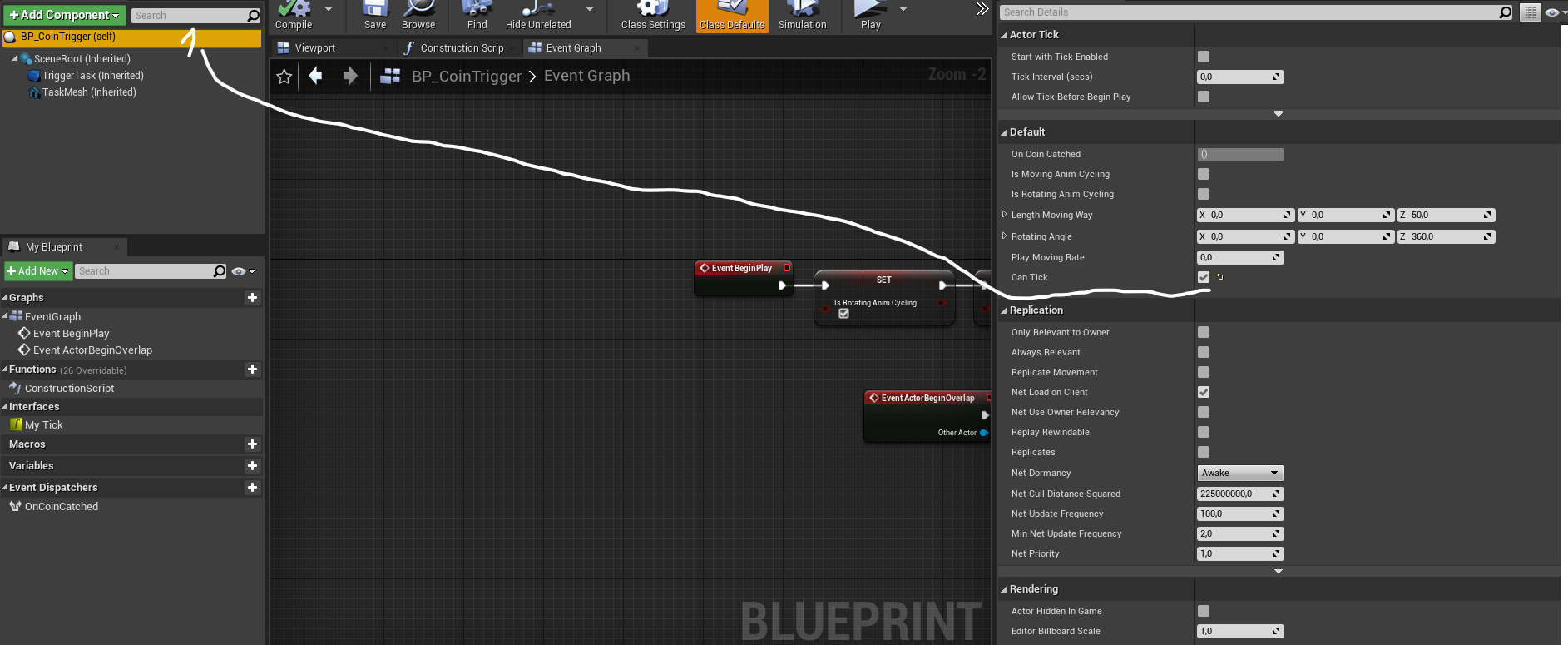
****

****

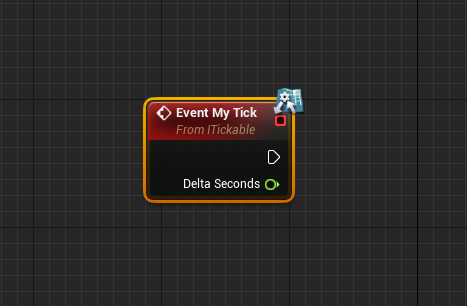
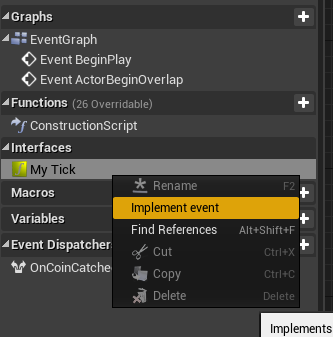
И удалить из массива **ITickableGroup** с помощью метода **RemoveTickableObject, (**перед тем как объект будет удален со сцены) и указать текущий обьект, который удаляется, чтоб не получать доступ к объекту, который не существует больше

****

**(ОБНОВЛЕНО) BP\_BaseActor** хранит в себе всю эту логику (не нужно каждый раз добавлять и удалять из массива объект). Просто на объекте, который мы спавним и в нем нужно чтоб срабатывал **Tick**, нужно поставить галочку **CanTick**



и имплементировать событие **MyTick**, которое создаст ноду **MyTick**



**Методы:**

**Category [Default]:**

**public:**

* **AddTickableObject (TickableObject) –** публичный метод, который добавляет объект, который реализует интерфейс ITickable в список **ITickableGroup.**
* **RemoveTickableObject (TickableObject) -** публичный метод, который удаляет объект, который реализует интерфейс ITickable из списка **ITickableGroup.**

**Переменные:**

**Category [Default]:**

* **ITickableGroup –** группа объектов, которые реализуют интерфейс ITickable

**Первый элемент в массиве это персонаж BP\_MyCharacter.**