Деталь

Детали - это главные строительные блоки в Apparatus-е. Это высокоуровневые сущности (в отличие от трейтов), которые поддерживают некоторую дополнительную ECS+ функциональность, например, мульти-итерацию и наследование.

Детали наследуются от класса UObject и являются субъектом сборщика мусора и Unreal-овской модели памяти (в то время как трейты используют собственную организацию памяти).

Если надо изменить какие-то составляющие детали, то достаточно обратиться к ним напрямую через оператор обращения к полю → или не-const метод самой детали. Нету нужды копировать данные детали, применять к ним изменения и устанавливать в сущностный объект.

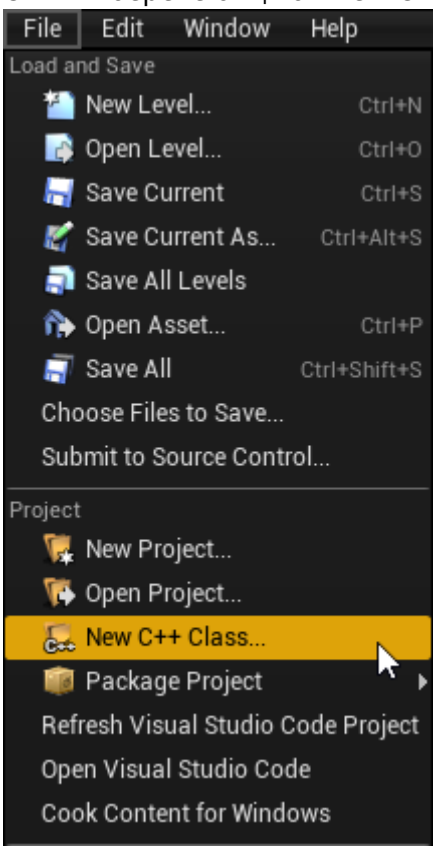
В качестве оптимизации для некоторой внутренней логики детали не могут быть легко удалены с сущностных объектов. Они могут быть только выключенными, т.е. *disabled*. По сути это то же самое, что и удаление, потому что фильтры следят за состоянием enabled/disabled у детали в контексте отбора сущностей для оперирования.

Создание деталей

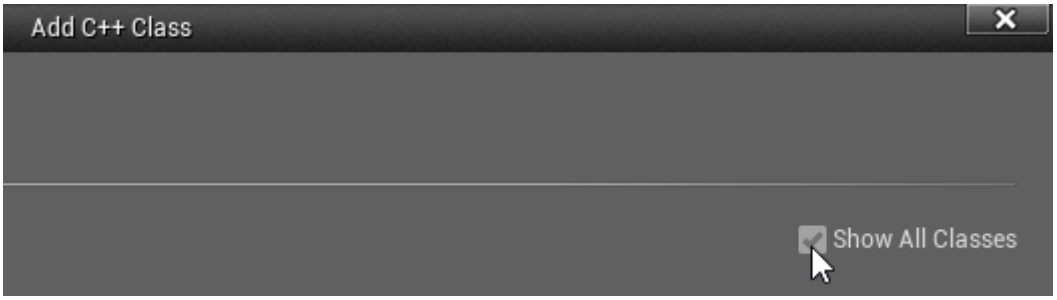
Используя C++

Чтобы создать деталь, видимую в вашем C++ коде, вам следует сделать следующее:

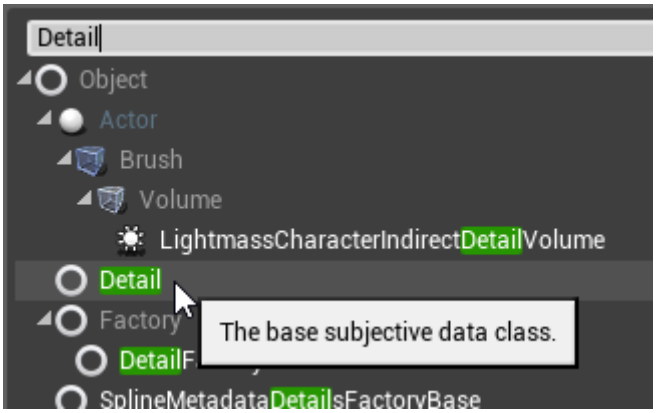
1. Откройте главное File-меню в UE и выберете опцию «New C++ Class...» :



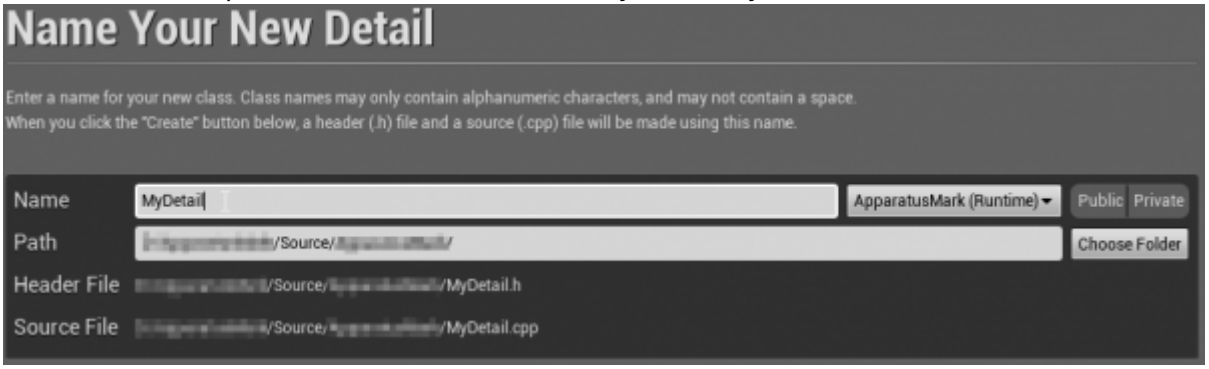
2. В открывшемся окне пометьте «Показывать все классы» («Show All Classes»):



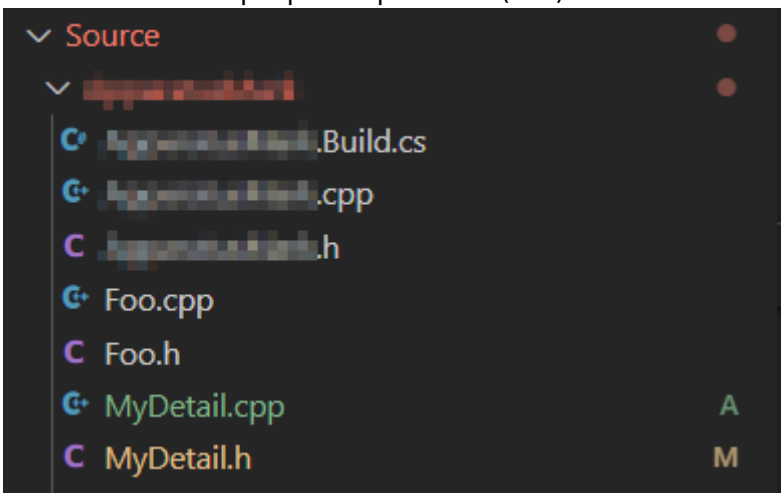
3. Теперь вы можете выбрать любой доступный базовый класс, включая классы Apparatus-a. Выберете «Detail» в качестве базового класса:



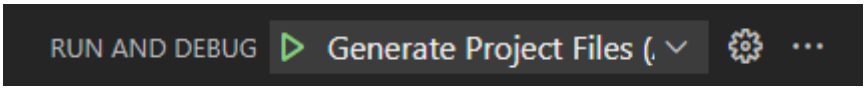
4. Кликните «Next» и вы должны увидеть диалог выбора имени. Назовите создаваемый класс как надо и продолжите, нажав на зелёную кнопку «Create Class»:



5. Новый класс будет создан как комбинация заголовочного файла (.h) и файла-ресурса (.cpp). Всё будет помещено в «Source» (под)папку вашего проекта. Теперь вы должны увидеть их в выбранной системе программирования (IDE):



6. Заметим, что вы, возможно, должны будете рекомпилировать проект и/или перезапустить редактор после этого. Не волнуйтесь за некоторые возможные ошибки, вновь сгенерируйте проект IDE, скомпилируйте и запустите.



7. Содержимое соответствующих фалов должно быть таким:
   - MyDetail.h:

```
// Fill out your copyright notice in the Description page of
Project Settings.

#pragma once

#include "CoreMinimal.h"
#include "Detail.h"
#include "MyDetail.generated.h"

/**
 *
 */
UCLASS()
class ME_API UMyDetail : public UDetail
{
    GENERATED_BODY()
};
```

   - MyDetail.cpp:

```
// Fill out your copyright notice in the Description page of
Project Settings.


#include "MyDetail.h"
```

8. Теперь вы можете добавлять какие-нибудь поля данных в класс, как обычно это делается, прямо в заголовочном файле C++:

```
float X = 0;
float Y = 0;
```

9. Возможно вы также захотите опубликовать свои поля в качестве свойств класса, чтобы получить к ним доступ через блупринты и даже поменять их начальные значения через пользовательский интерфейс (для подробностей, пожалуйста, посетите документацию по Свойствам):

```
UPROPERTY(BlueprintReadWrite, EditAnywhere)
float X = 0;

UPROPERTY(BlueprintReadWrite, EditAnywhere)
float Y = 0;
```

10. Ваша C++ деталь готова к использованию. Пожалуйста, проверьте API документацию для дополнительной информации.