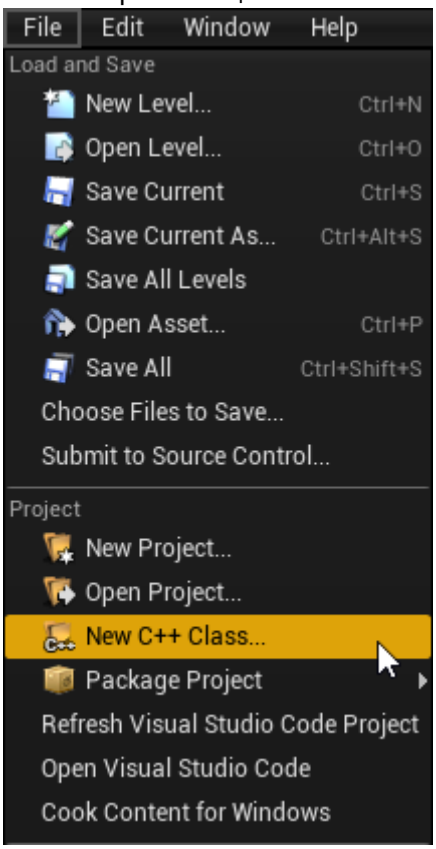
# Тип "механический"

ECS явным образом отделяет данные и логику, оперирующую этими данными. Эта логика, в свою очередь, обычно выполняется покадрово (то есть повторяется). Apparatus имплементирует эту функциональность через концепт, называемый *Mechanical* (механический тип). Механические объекты - как многогранный организм, включают несколько механик (Mechanics), которые выполняются внутри них.

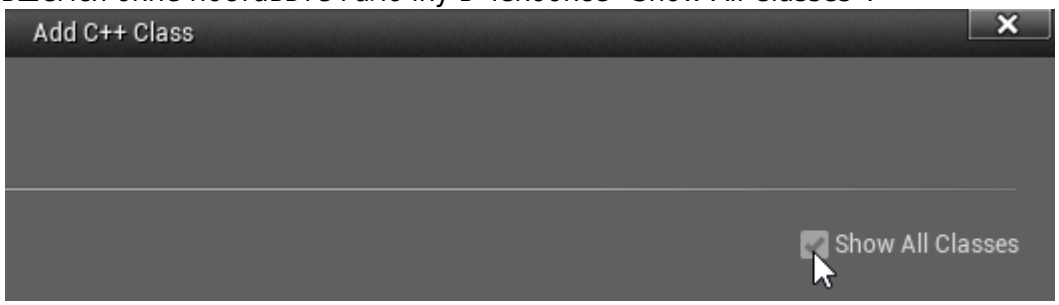
## Организация в C++

Если вы намерены идти C++ путём, создание собственного механического типа проходит примерно так.

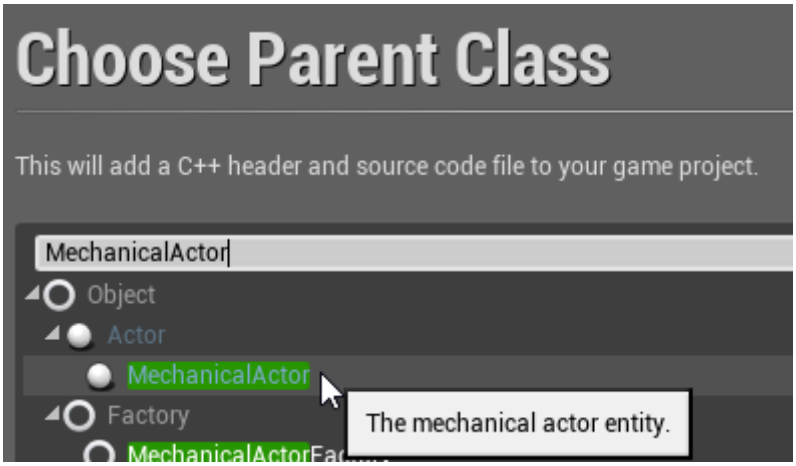
1. Откройте главное UE File-меню и выберете опцию “New C++ Class...”:



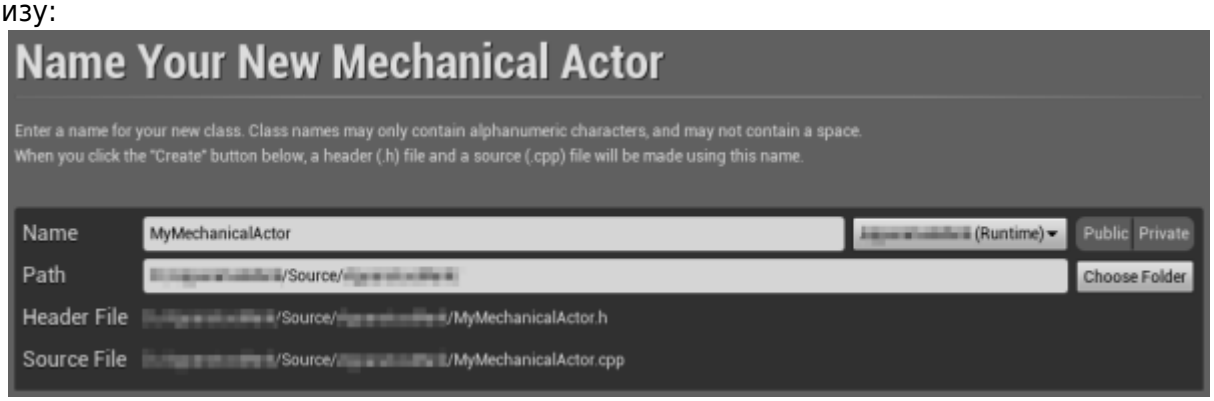

2. В открывшемся окне поставьте галочку в чекбоксе “Show All Classes”:



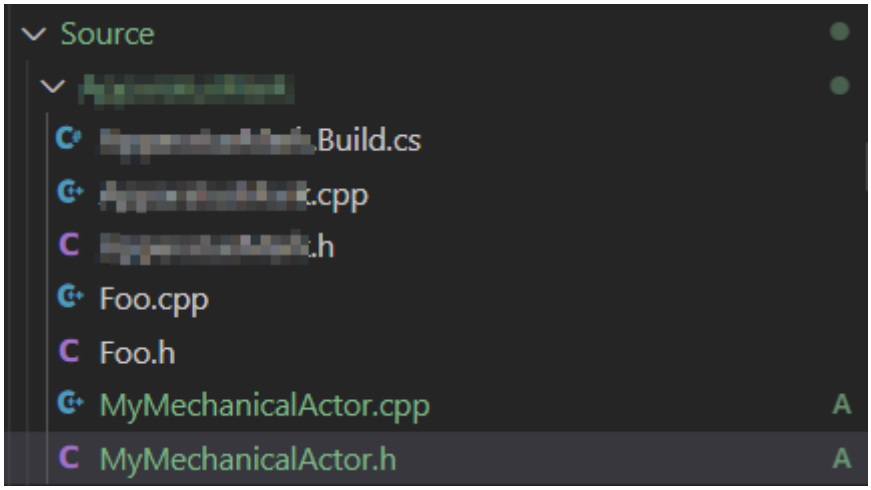

3. Теперь вы можете выбрать любой доступный базовый класс, включая классы плагина. Выберете “Mechanical Actor” в качестве базового:



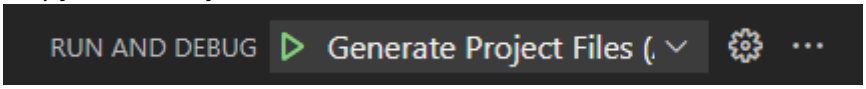

4. Нажмите “Next”, и вы должны увидеть диалоговое окно выбора имени. Назовите создаваемый класс, как надо, и продолжите, кликнув зелёную кнопку “Create Class” внизу:




5. Новый класс создастся как комбинация хеадера (.h) и файла-ресурса (.cpp). Всё находится в (под)папке Source вашего проекта. Вы должны увидеть файлы в выбранной IDE:




6. Заметим, что вам, возможно, надо будет перекомпилировать проект и/или перезапустить редактор после этого. Не пугайтесь некоторых возможных ошибок, ещё раз соберите IDE-проект, скомпилируйте и запустите вновь.




7. Наполнение соответствующих файлов должно выглядеть так:
   - `MyMechanicalActor.h`:

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

```
#pragma once

#include "CoreMinimal.h"
#include "MechanicalActor.h"
#include "MyMechanicalActor.generated.h"

/**
 *
 */
UCLASS()
class MY_API AMyMechanicalActor : public AMechanicalActor
{
    GENERATED_BODY()
};
```

- MyMechanicalActor.cpp:

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


#include "MyMechanicalActor.h"
```

8. Теперь вы можете перегрузить один (или несколько) Tick-методов и реализовать вашу механику, как обычно это делается в C++...
   - ... в заголовочном файле:

```
void Tick(float DeltaTime) override;
```

   - ... в .cpp-файле:

```
void AMyMechanicalActor::Tick(float DeltaTime)
{
  // Ваш код механики
}
```