1. Создадим наш первый C++ класс, будем наследоваться от класса Actor – это базовый класс движка, который можно добавить на сцену.

2. Класс можно создать несколькими способами: в директории с C++ классами через ПКМ, либо через вкладку File в главном меню.

3. При создании класса открывается popup – здесь мы видим наиболее часто используемые классы компоненты, но можно открыть все классы и проследить древо наследований в анриале.

4. Выбираем класс Actor, далее задаем имя, путь оставляем тот же. Воспользуемся дополнительной возможностью и нажмем кнопку public – будут созданы две дополнительные директории Public и Private, в первую положат хедер, во второй – cpp-файл. Все ЗФ будут создаваться в папке Public, а все cpp – в private. Нажимаем create class и VS обновляет проект, скомпилирует файлы, а editor подгрузит все библиотеки.

5. В VS обновился solution и добавился только что созданный класс. В solution следующая структура – Engine – код движка, Games – пользовательские проекты.

6. Рассмотрим, что сгенерировал нам VS:

* #include "CoreMinimal.h" - ЗФ, в котором определены базовые типы, константы, инструменты для логирования.
* #include "GameFramework/Actor.h" - мы наследуемся публично от Actor, поэтому нам надо указать ЗФ, где он объявлен.
* #include "BaseGeometryActor.generated.h" - автоматически сгенерированный хедер, движок генерирует его для каждого актора, там находится метаинфа о классе и он должен идти самым последним в списке всех инклюдов
* UCLASS(), GEOMETRYSANDBOXCLEAR\_API, GENERATED\_BODY() – это макросы, которые рассмотрим позже – это некоторый обвес вокруг класса, который позволяет пользоваться возможностями UE, например, чтобы данный класс был доступен в блюпринтах.

7. В Content Browser теперь есть наш класс в папке C++ Classes. Мы можем его драг-н-дропнуть на вьюпорт, он появится в world outliner, но у него еще не заданы никакие параметры, поэтому у него нету даже Gizmo (но мы скоро это исправим).

8. Функции, которые UE сгенерировал внутри нашего класса:

ABaseGeometryActor(); - конструктор класса по умолчанию;

virtual void BeginPlay() override; - вызывается один раз, либо когда объект спавнится, либо когда началась игра, если актор на сцене;

virtual void Tick(float DeltaTime) override; - вызывается каждый фрейм, у нее есть параметр DeltaTime – это разница по времени между предыдущим и текущим тиком.

9. В cpp файле идет подключение нашего нового хедера. В теле конструктора мы видим структуру:

PrimaryActorTick.bCanEverTick = true;

У нее есть поле bCanEverTick, установленное в данном случае в true, что означает, что тик нашего актора будет вызываться каждый фрейм. Эта переменная служит для оптимизации и иногда нам не надо, чтобы данная функция вызывалась каждый фрейм.

В методе BeginPlay() мы видим ключевое слово Super – это alias на имя базового класса (AActor):

Super::BeginPlay(); ( == AActor::BeginPlay(); )