1. Рассмотрим основной строковый тип данных в UE – Fstring.

2. Для начала создадим свою собственную категорию логирования. Для этого в самом начале файла BaseGeometryActor.cpp вводим макрос DEFINE\_LOG\_CATEGORY\_STATIC – данный макрос отвечает за создание локальной категории логирования (то есть такой категории, которая будет доступна только в рамках данного cpp файла).

3. Данный макрос принимает следующие три аргумента:

* название категории;
* 2 и 3 аргументы означают максимально допустимый уровень логирования. Временно введем Error, Error.

Эти Verbosity – это специальный enum, в котором разные уровни логирования в определенном порядке. Так как мы записали Error, а он второй по счету, те что выше – нельзя (сообщения не будут выводиться). Третий параметр более строгий – он относится ко времени компиляции – поэтому тело нашего лога не будет добавлено в код программы, компилятор его проигнорирует.

Если напишем All, All, можно будет все.

4. В UE есть несколько типов строк FString, FName, FText. Начнем с первого. К FString можно относиться как к аналогу std::string – это контейнер символов. В UE используется свой контейнер символов TCHAR.

Однако UE\_LOG не умеет напрямую работать с FString, он умеет работать с массивом символов. Чтобы получить указатель на массив символов нужно применить оператор разыменования перед строкой:

UE\_LOG(LogBaseGeometry, Display, TEXT(“Name: %s”), \*Name);

Сигнатура оператора зведочки:

FORCEINLINE const TCHAR\* operator\*() const { ... }

Для конкатенации строки с int используется FString::FromInt(int):

FString WeaponNumStr = ... + FString::FromInt(WeaponsNum);

Для конкатенация строки с float уже надо FString::SanitizeFloat(float):

FString HealthStr = ... + FString::SanitizeFloat(Health);

Для bool:

FString IsDeadStr = ... + FString(IsDead ? “true” : “false”);

Сконструируем новую строку на основе предыдущих через FString::Printf():

FString Stat = FString::Printf(TEXT(“ \n == All Stat == \n %s \n %s \n %s ”), \* WeaponNumStr, \*HealthStr, \*IsDeadStr);

И выведем это дело в лог:

UE\_LOG(LogBaseGeometry, Warning, TEXT(“%s”), \*Stat);

Теперь выведем сообщение на экран уже прямо поверх самой игры. Делается это через указатель на глобальный объект движка (GEngine), для этого надо подключить ЗФ Engine/Engine.h. Делается это с помощью функции AddOnScreenDebugMessage().

* первый аргумент – уникальный ключ сообщения (не дает сообщениям с одинаковыми ключами выводится на экран).
* второй – время, сколько секунд будет сообщение на экране.
* дальше цвет – через FColor::ключевое слово или через RGB-конструктор FColor(R, G, B).
* дальше строка, которую хотим вывести (при этом тут не надо разыменовывать).
* два оставшихся можно оставить по умолчанию – первый обозначает порядок вывода – новые сообщения сверху выводятся или нет, последний параметр – scale – можем поменять размер нашего сообщения.

GEngine->AddOnScreenDebugMessage(-1, 3.0f, FColor::Red, Name);

GEngine->AddOnScreenDebugMessage(-1, 5.0f, FColor::Green, Stat, true, FVector2D(1.5f, 1.5f));

Ну и перенесем все в функцию printStringTypes():

// ...

void printStringTypes();

// ...

void printStringTypes() {

}