Черты (Traits)

Черты - низкоуровневые блоки данных (компоненты) в составе сущностей. В отличие от деталей, черты являются простыми UStruct данными. Плагином они управляются вручную кэш-эффективным способом и главным образом направлены на runtime-производительность.

Создание черт

Все Unreal структуры типа UStruct должны быть доступными в Apparatus'е автоматически. Если вы не можете найти свою структуру в списке черт, дважды кликните на неё в Content-браузере, чтобы она загрузилась. В целом вы должны сделать это только один раз, потому что она будет загружена автоматически, если где-либо есть на неё ссылки.

Организация в C++

Вы, главным образом, обращайтесь к официальному Unreal Engine мануалу по созданию UStruct-ов.

По существу, вы создаёте хеадер (.h) файл и (опциально) файл-ресурс (.cpp). Пример трейта, объявленного с помощью только заголовочного файла:

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

#pragma once

#include "CoreMinimal.h"
#include "MyTrait.generated.h"

/**
 *
 */
USTRUCT(BlueprintType)
struct MY_API FMyTrait
{
    GENERATED_BODY()

  public:

    UPROPERTY(BlueprintReadWrite, EditAnywhere)
    float VelocityX = 0;

    UPROPERTY(BlueprintReadWrite, EditAnywhere)
    float VelocityY = 0;
};
```

Вы можете опустить спецификации UPROPERTY, но с указанными ключевыми словами у вас появляется возможность использовать `MyTrait` как в C++ коде, так и в блупринтах.

Пара слов о Garbage Collection

Как уже упоминалось раньше, Apparatus использует свою собственную низкоуровневую модель памяти, чтобы хранить трейты эффективно. Это достигается некоторой ценой. Вам приходится вручную гарантировать безопасность ссылок на другие U-Объекты (на Actor-ы, Component-ты и т.д.) внутри своих трейтов, потому что последние должны были бы быть подсчитаны сборщиком мусора, который в рассматриваемом контексте недействителен.

К счастью, вы можете сделать это достаточно просто при помощи ссылок на эти же объекты в определённых ассетах или объектах через некоторый GC-зависимый инстанциированный UObject, тем самым задерживая объекты в памяти. Эта глобальная инстанция также может быть добавлена в корень (added to root), чтобы не стать собранной коллектором. Вы можете также добавить объекты, на которые ссылаетесь в своих трейтах, к корню, чтобы достичь того же результата.

Ссылки на остальные сущности (Subjects) через хэндлеры (Subject Handles) - прекрасное решение, поскольку управляются самим плагином. Просто запомните, что эти хэндлеры - как слабые ссылки (Handles are like weak references). Они не поддерживают ссылаемую сущность, просто становятся невалидными, когда сущность была уничтожена (удалена).