**14. Launcher #2, rocket movement**

1. Какие у нас стоят две задачи (для определения движения проджектйла)?

2. Что взяли из Riffle (уже реализованный код), что мы с помощью него получаем?

3. Высчитываем вектор, вдоль которого нужно запустить снаряд

4. До этого отложено создавали ракету через UGameplayStatics, теперь по-старому

5. Передаем вектор направления в класс ракеты

6. Осталось разобраться, как снаряд будет двигаться – можно реализовать самим, а можно через встроенные возможности движка – как сделали через второй способ? Почему определяем значение скорости через поле нового компонента?

7. Что появилось в блюпринте? Как сейчас себя ведут наши ракеты, как наколдовать можно с их перемещением?

8. Проблема с акторами – какая? Как исправили? Установили параметры из предыдущего пункта через C++

1. У нас есть две задачи – высчитать направление, вдоль которого мы хотим запустить патрон, и заставить патрон двигаться вдоль данного направления.

2. Большинство кода у нас уже имеется. Из класса Riffle возьмем код для Trace Data и Make Hit:

Изображение выглядит как текст

Автоматически созданное описание

Этот код позволяет нам получить информацию о попадании (или нет) нашей ракеты.

3. Высчитаем вектор, вдоль которого нужно запустить снаряд. Конечную точку определяем по простой логике через тернарный оператор, а начальная точка соответствует положению нашего сокета дула.

Изображение выглядит как текст

Автоматически созданное описание

Имея две точки, мы можем определить вектор направления, вычтя из конечный точки начальную. Нам надо знать только направление, поэтому сделаем данный вектор единичным.

4. В предыдущем уроке мы сделали спаун ракеты через UGameplayStatics. В этот раз сделаем по старинке – через объект мира.

Разница между ними одна – функция SpawnActorDeffered возвращает указатель на конкретный класс, а UGameplayStatics – на актора (поэтому нужен бы был каст).

А далее завершаем спаун.

Просто отрефакторили код.

Изображение выглядит как текст

Автоматически созданное описание

**!Тут ошибка – должна быть функции SpawnActorDeferred (далее она тоже прослеживается – поэтому и появлялся Exception)!**

5. Вектор направления нам надо передать в наш класс ракеты, чтобы он знал, куда надо двигаться:

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

6. Вектор передали – осталось разобраться, как снаряд будет двигаться. Реализовать движение вдоль вектора достаточно просто – в функции Tick изменять положение актора на какое-нибудь значение.

Однако в UE есть уже встроенные компоненты и логика для этого.

Данный компонент изменяет положение актора на тик, в зависимости от настроенных в нем параметров. Мы уже рассматривали один компонент движения – CharacterMovementComponent – можно сказать, что они из одного семейства – умеют изменять положение в пространстве других акторов:

Изображение выглядит как текст

Автоматически созданное описание

У данного компонента множество возможностей – можно сделать самонаводящуюся ракету или определить количество отскоков снаряда. Мы рассмотрим простейшее движение с постоянной скоростью. Перейдем в cpp и добавляем его ЗФ, а также создаем этот компонент (он чисто логический, аттачитить не надо).

Все, что осталось – передать вектор скорости для данного компонента:

Изображение выглядит как текст

Автоматически созданное описание

Умножаем именно на InitialSpeed компонента (хотя могли создать свое проперти) потому, что предполагается, что вся информация о движении должна храниться в компоненте движения.

7. В блюпринте нашего проджектайла появился компонент. У него изначально установлено значение InitialSpeed в ноль. Поэтому патрон просто падает вниз.

Если поставим какое-то значение – будет падет по закону выпущенного под углом к горизонту тела (по дуге такой, как в школе на уроках физики).

Мы можем сделать так, чтобы ракета не падала вниз, изменив параметр Projectile Gravity Scale (по умолчанию 1) на 0. Тогда тело будет лететь прямолинейно. Данная переменная умножается на стандартную величину ускорения свободного падения.

8. Так же у нас наблюдается проблема – наши акторы не удаляются и их количество растет с каждым выстрелом. Поэтому на BeginPlay добавим вызов функции SetLifeSpan. А так же установим параметры для скорости и падения установим из C++:

Изображение выглядит как текст

Автоматически созданное описание

В таком случае проще увидеть значения для данных переменных, не открывая эдитор. Так же можно добавить check() на данные параметры. Таким образом, если кто-то изменяет значения в BP то мы уведомим, что такие данные не предполагались.