**15. Launcher #3, apply radial damage**

1. Что реализуем в данном уроке? Какие 4 компонента сразу определили в классе ракеты?

2. Подключаем два ЗФ для рисования сфер и нанесения ущерба, а также донастроим наш компонент коллизии (две функции), что они делают?

3. Как нам реагировать на момент столкновения снаряда с актором?

4. Какую логику реализовали при столкновении, какие функции использовали? Для функции нанесения урона нужно получить нашего персонажа и контроллер – как мы это сделали?

5. Последний момент – как мы указали компоненту коллизии, каких акторов мы хотим игнорировать во время движения? Зачем это сделали?

1. Задача – научить ракету наносить урон при столкновении с каким-нибудь объектом.

Параметры для урона мы укажем в классе ракеты. Можно сделать и по-другому: за урон полностью отвечает лаунчер и все параметры передаются в класс ракеты (аналогично ShotDirection).

Сразу создадим необходимые переменные:

а) радиус поражения;

б) количество урона;

в) нанесение или нет полного урона в сфере;

г) а также исправим ранее захардкоденную константу для времени уничтожения персонажа.

Изображение выглядит как текст

Автоматически созданное описание

2. Подключаем два ЗФ для рисования сфер и GameplayStatics для нанесения ущерба.

Донастроим еще наш CollisionComponent:

Изображение выглядит как текст

Автоматически созданное описание

Благодаря первой настройке у нас не будет физических взаимодействий – будут только оповещения о столкновениях у данной коллизии.

Вторая настройка устанавливает то, как будет данная коллизия будет реагировать с другими объектами, у которых обрабатываются коллизии. Мы установили одинаковую реакцию для всех каналов. Наш проджектайл будет блокироваться всеми доступными каналами.

3. Подпишемся новый делегат OnComponentHit – он вызывается тогда, когда происходит столкновение с актором или другим компонентом:

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Через F12 нажимаем на наш делегат, чтобы определить его сигнатуру для нашей функции. При этом из нашей сигнатуры делегата мы почему-то убрали первый компонент (тут есть еще UPrimitiveComponent):



4. Логика при столкновении следующая:

Изображение выглядит как текст

Автоматически созданное описание

Первое, что мы сделали – остановили движение – функция StopMovementImmediately – она устанавливает движение в ноль. Мы делали что-то похожее, когда наш персонаж умирал и мы вызывали функцию Disable Movement у CharacterMovementComponent.

Далее необходимо нанести радиальный урон - функцию урона взяли ту же, что и для DevDamageActor, и настроили ее под наш случай.

Для передачи в массив игнорируемых акторов вернемся в cpp файл нашего лаунчера и определим хозяина ракеты – нашего персонажа, так как мы установили Owner’ом нашего оружия персонажа:

Изображение выглядит как текст

Автоматически созданное описание

Добавили функцию для указания контроллера. Данная функция возвращает указатель на контроллер персонажа.

Изображение выглядит как текст

Автоматически созданное описание

Эта функция – задел на будущее, так как позже при подсчете того, кто кого подстрелил, мы будем использовать указатель на данный контроллер.

После осталось только уничтожить актор.

Ну и в конце нарисуем сферу, которая будет визуализировать наш взрыв.

5. Последний момент – мы можем указать компоненту коллизии, каких акторов мы хотим игнорировать во время движения (второй аргумент указывает, нужно его игнорировать или нет):

Изображение выглядит как текст

Автоматически созданное описание

Мы это делаем из-за того, что скорость полета снаряда может быть настолько медленной, что мы можем его догнать.