**3. Class AHUD, crosshair**

1. HUD – что за класс и для чего предназначен? Какой создаем класс, от кого наследуемся, как называем, помещаем в папочку

2. В родительском классе ищем функцию, какую (для отрисовки HUD’а)? Когда она вызывается? Что позволяет делать? Переопределяем ее в нашем классе и вводим еще одну функцию для рисования прицела

3. Функция HUD’а, которая позволяет рисовать линии на экране, что принимает? Каким воспользуемся дополнительным объектом и из какого ЗФ? Какие у него нужны нам поля? Новый тип, в котором храним значения? Как рисуем прицел?

4. Переопределяем дефолтный HUDClass

5. Смещаем камеру и убираем здоровье над персонажем – как в BP и в коде?

1. HUD – специальный класс, который отвечает за отрисовку интерфейса, который будет накладываться поверх игры. Мы можем унаследоваться от данного класса и нарисовать все необходимые нам элементы с помощью данного класса.

Собственно, создадим наш собственный класс, унаследуемся от HUD и назовем STUGameHUD и поместим в папку UI.

Добавляем наш путь в Build.cs.

2. В заголовочном файле нашего HUD проваливамся в AHUD (от которого наследовались) и найдем функцию DrawHUD().

Данная функция отвечает за отрисовку и вызывается каждый фрейм – в ней мы можем рисовать линии, прямоугольники, выводить текстуры, текст и т.д. Переопределим ее в нашем классе и добавим дополнительную функцию для инкапсуляция кода отрисовки прицела.

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

3. Функция HUD’а, которая позволяет рисовать линии на экране, называется DrawLine() и принимает начальные и конечные точки на экране по X и Y (4 параметра), цвет линии и толщину линии.

Для того, чтобы нарисовать наш прицел по центру экрана, нам надо узнать, где этот центр экрана. В этом нам поможет объект Canvas из ЗФ Engine/Canvas.h.

В данном объекте происходит рисовка всех графических примитивов и именно в нем находятся размеры **текущего окна** игры.

Данный класс содержит две переменные – SizeX и SizeY, которые содержат размеры экрана (ширину и высоту).

Для вычисления центра экрана воспользуемся переменной **шаблонного** типа TInterval – он представляет собой отрезок и хранит два элемента. Передаем туда наши размеры, умноженные на 0,5 – чтобы получить центр экрана.

У него есть два поля Min и Max, которые и содержат те параметры, которые мы передали.

Так же нам нужны три переменные. Первая для половины размера линии, так как мы будем рисовать из центра и будем откладывать по половинке вверх и вниз.

Вторая для толщины линии.

Последняя – для цвета.

Так как рисуем две линии, надо вызвать два раза функцию для отрисовки:

Изображение выглядит как текст

Автоматически созданное описание

4. Последнее, что надо сделать – переопределить дефолтный HUDClass:

Изображение выглядит как текст

Автоматически созданное описание

5. Так как при такой позиции камеры крестик выглядит странно, поменяем позицию камеры, а также уберем здоровье над персонажем.

Скрывать здоровье будем хитрым способом: у HealthTextComponent есть два проперти Owner No See и Only Owner See – из названия понятно, что они делают – с помощью них можно управлять рендерингом компонента в зависимости оттого, владеем мы им или нет. Поставим Owner No See – тогда мы будем видеть у других персонажей, а у нас – нет.

А так же не забудем сбросить Rotation, чтобы он был повернут к нам над другим персонажем.

А так же поставим настройки оффсета SpringArmComponent и Owner no See через код:

Изображение выглядит как текст

Автоматически созданное описание