**12. WIDGET COMPONENT / HEALTH BAR**

1. Добавим полоску жизней для противников. Для начал создадим BP-виджет WBP\_HealthBar. Что добавляем? Как протестили через BP то, как работает Widget? Чтобы было видно со всех сторон здоровье? Создадим класс для HealthBar от UserWidget, называем STUHealthBarWidget. Что создаем в нем (4 проперти и одну функцию, как определяем ее поведение – три пункта надо сделать в ней)?

2. Что создаем у AI-персонажа (проперти и две функции, одна из них – функция, которую сделали виртуальной в базовом персонаже – для чего)? Как настроили наше новое проперти в конструкторе? Как настроили BeginPlay (просто проверка) и виртуальную функцию?

3. Перейдя в эдитор, что увидим у AI-персонажа? Как настраиваем? Все неплохо, но есть одна проблемка – какая и как пофиксили (добавили проперти и две функции AI-персонажу)? Последнее – когда мы погибаем, мы становимся наблюдателем, поэтому как изменяем вызов одной функции?

1. Добавим полоску жизней для противников. Для начал создадим BP-виджет WBP\_HealthBar. Добавляем один элемент – ProgressBar.

Для проверки мы добавили нашему AI-персонажу данный виджет с помощью элемента Widget в блюпринте (данный элемент рендерит выбранный Widget), выбрали наш WBP\_HealthBar. Однако, если мы заходили за персонажа, элемент было не видно. Можно сделать вращение данного компонента в зависимости от вращения нашей камеры, но у Widget есть проперти Space, которая устанавливается либо в World, либо в Screen. Если выбрать Screen, то виджет будет отображаться в слое интерфейса, а не 3D-пространства. Даже если объект заграждает противника.

Создадим класс для HealthBar от UserWidget, называем STUHealthBarWidget. Создаем проперти для прогресс бара, процент жизни, ниже которого прогресс бар будет отображаться, процент жизни, при котором меняется цвет прогресс бара и два проперти для цветов хелс бара и функцию для обработки установки цвета и процентов прогресс бара:

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Функция для установки заливки Healthbar’а называется SetFillColorAndOpacity. И в конце устанавливаем наш новый процент.

2. У AI-персонажа создадим WidgetComponent, который мы создавали в блюпринтах. Функцию BeginPlay и функцию, которая вызывается при изменении процента здоровья. Такая у нас есть в Base-персонаже, она называется OnHealthChanged, мы ее сделаем виртуальной и в секции protected. Переопределим ее поведение в нашем классе.

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описаниеДля того, чтобы получить объект UserWidget, который мы устанавливаем в блюпринтах, у WidgetComponent есть специальная функция GetUserWidgetObject – возвращает указатель на UserWidget, мы сразу делаем каст. Если он не нулевой – передаем в виджет процент жизни нашего персонажа.

3. Перейдя в эдитор увидим, что у нашего AI-персонажа создался WidgetComponent. Указываем WidgetClass – WBP\_HealthBar. Далее надо сделать reparent нашему WBP\_HealthBar и переименовываем ProgressBar.

Все неплохо, только когда мы отдаляемся, виджет выглядит гигантским. Сделаем так, что, если мы находимся достаточно далеко от противника, виджет так же становится невидимым.

Изображение выглядит как текст

Автоматически созданное описание

В Tick функции будем определять расстояние до PlayerController (он у нас единственный) и, если данное расстояние больше определенного значения, то будем делать виджет невидимым. В целях оптимизации это можно делать с помощью отдельного таймера, чтобы проверять видимость не каждый фрейм, а раз в секунду, например.

Изображение выглядит как текст

Автоматически созданное описание

Функция получения указателя на PlayerController находится в UWorld – GetFirstPlayerController, оттуда уже берем пауна и его локацию. Определяем расстояние между положением персонажа и AI-персонажа. В начале функции надо проверить, что все указатели не нулевые.

Изображение выглядит как текст

Автоматически созданное описание

И последнее – когда мы погибаем и становимся наблюдателем, функция GetPawn будет возвращать nullptr и Visibility устанавливаться не будет – она останется той, какой она была в последний момент. Функция GetPawn – функция базового класс AController, она возвращает Pawn. У PlayerController дополнительно имеется SpectatorPawn с указателем на Spectator, поэтому у него есть специальная функция GetPawnOrSpectator – она возвращает указатель на паун, если он не нулевой, либо на спектейтор в противном случае.