**6. Компонент восприятия UAIPerceptionComponent**

1. Для чего используется данный компонент? Создали свой, от кого наследуемся?

2. Где будем создавать данный комопнент? Ввели для этого проперти. В базовом классе уже есть такой компонент? Какую функцию вводим для переопределения компонента по умолчанию и как это делаем + создание компонента? Какой ЗФ надо подключить?

3. Настраиваем наш компонент у контроллера в блюпринтах. В каком разделе панели Details и какие там настройки? (подсказка – одна главная, а затем у нее уже 7 настроили внутри)

4. Как включить дебаг режим в игре и выбрать именно категорию Perception? Какую логику запрограммируем (для «смотрения» на определенного персонажа)? В каком объекте (компонент, контроллер, персонаж)? Какую функцию вводим? Какие 4 ЗФ подключаем? Как определяем код в данной функции? Как нам понадобилось пофиксить нашу функцию получения компонента и почему и что мы обнаружили за функцию новую? Подытожим логику функции.

5. У какого объекта и в какой функции будем искать наших персонажей и выбирать между ними ближайшего и смотреть на него? Как будем заставлять персонажа смотреть (какая функция)?

1. Рассмотрим новый компонент PerceptionComponent – данный компонент позволяет NPC акторов определять других акторов в мире: видеть, слышать, реагировать на урон. Сейчас сосредоточимся на зрении.

В директории с компонентами создадим наш компонент восприятия STUAIPerceptionComponent, наследуясь от базового класса AIPerceptionComponent.

2. Данный компонент будем создавать в классе STUAIController. В базовом классе AIController уже существует данный компонент, поэтому нам надо его переопределить. Для этого объявим конструктор, в котором будем создавать компонент, и само проперти с компонентом.

Изображение выглядит как текст

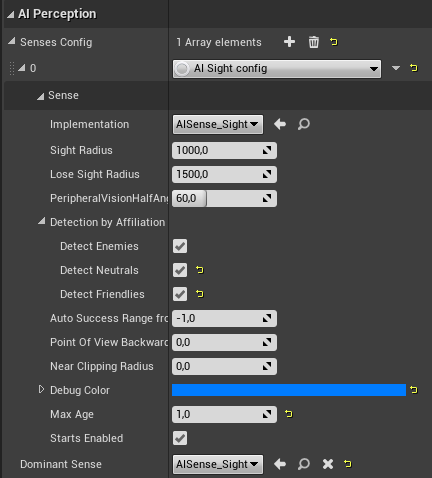
Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

В cpp-файле подключаем ЗФ компонента и определяем конструктор и единственный нюанс – нам понадобилось вызвать специальную функцию контроллера SetPerceptionComponent (передается по ссылке, поэтому нам надо разыменовать указатель).

3. Настроим наш компонент в блюпринте контроллера. Переходим в панели Details в раздел AI Perception.



Senses Config является массивом, через данный массив конфигов настраиваются чувства нашего персонажа. Список доступных конфигов:

Изображение выглядит как текст

Автоматически созданное описание

Нас интересует AI Sight config (отвечает за зрение персонажа). Нажав на треугольничек видим настройки данного конфига:

* Sight Radius – отвечает за дистанцию видимости NPC.
* Lose Sight Radius – отвечает за дистанцию, при которой NPC теряет из поля зрения объект.
* Следующий параметр с длинным названием – угол видимости.
* В Detection by Affiliation указывается кого мы видим – врагов, нейтральных персонажей или союзников. Выберем всех.
* Max Age – определяет количество времени, сколько хранится объект после того, как мы потеряли его из виду.
* Dominant Sense – с помощью него указывается приоритет конфига – если у нас несколько чувств настроено. В нашем случае мы можем оставить None, либо указать Sight.

4. Посмотрим, как это выглядит. Нажмем апостроф и клавишу «4» **на нумпаде**, чтобы увидеть дебаг-инфу о Perception:

Изображение выглядит как текст

Автоматически созданное описание

Зеленый цилиндр – Sight Radius, розовый - Lose Sight Radius.

Запрограммируем следующее – если в видимости NPC находится несколько игроков, то он должен выбрать ближайшего и развернуться в его сторону, данная механика в дальнейшей потребуется для атаки.

Открываем ЗФ нашего PerceptionComponent и объявляем функцию, которая будет возвращать указатель на ближайшего актора, которого заметил наш компонент.

Изображение выглядит как текст

Автоматически созданное описание

Потребуются 4 ЗФ:

Изображение выглядит как текст

Автоматически созданное описание

Чтобы вычислить расстояние до ближайшего актора, завели массив для всех акторов из поля видимости нашего персонажа.

Функция PerceptionComponent, которая получает всех акторов, которые были определены конкретным чувством. Принимает чувство и массив по ссылке, который она заполнит.

Далее нам надо вычислить все расстояние и найти ближайшее к нам. Для этого нужен указатель на наш паун. Получили его через контроллер (так как наш компонент находится в контроллере).

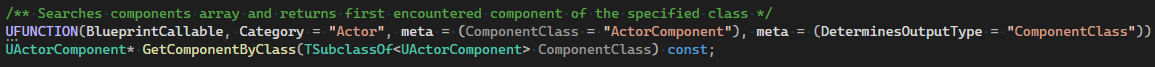
После проходим по всему массиву и находим ближайшего актора и расстояние до него. Для начала мы проверяем, жив ли персонаж, при проходе по массиву, потому что нет смысла стрелять по мертвому персонажу (+ оставим заметку на добавление разделения на данном этапе персонажей на своих и чужих).

В данный момент у нас наша функция для получения компонент была подчеркнута красным, потому что мы в качестве параметра указали APawn, а лучше было указать AActor,

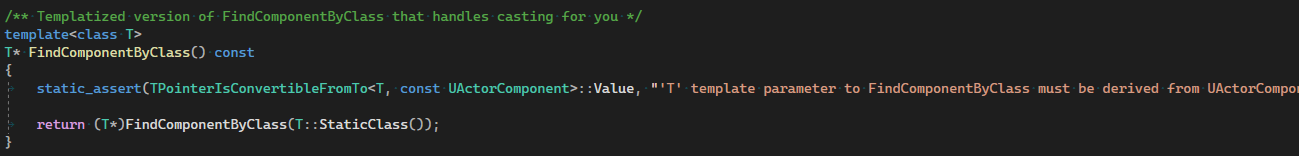
Изображение выглядит как текст

Автоматически созданное описание

потому что функция GetComponentByClass (к которой мы обращаемся внутри нашей функции) находится в AActor, а не в APawn.



Более того, если перейдем в ЗФ AActor, то можно увидеть, что уже имеется функция с нами реализованным функционалом с названием FindComponentByClass.



Если же мы все-таки все проверки прошли, то считаем расстояние и, если оно меньше, присваиваем нашей переменной и запоминаем актора. Ну и в конце возвращаем нашего актора.

Подытожим логику в данной функции:

* для начала получили всех паунов, на которых смотрит персонаж;
* после этого нам надо получить нашего персонажа, чтобы считать расстояние между ним и найденным паунами – для этого надо получить контроллер (через GetOwner(), потому что компонент является частью контроллера), а уже из него получить пауна (через его вызов GetPawn);
* дальше надо пробежаться по массиву найденных акторов и **среди живых** найти ближайшего – вот и всё.

5. Переходим в ЗФ AIController и добавим функцию Tick. Каждый фрейм будем определять ближайшего врага и поворачивать на него взор.

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

У контроллера есть специальная функция SetFocus, которая выставляет вращение контроллера таким образом, чтобы forward vector указывал на актора, которого мы передадим в качестве параметра.