**13. EQS CONTEXT**

1. Создадим генератор точек вокруг \_ и NPC будет перемещаться с помощью этого генератора. Что создаем для начала, как назвали и как определяем в нем генератор?

2. Далее нам нужно получить нужную точку из нашего генератора. Как настроили это? Однако сейчас точки точки генерируются вокруг текущего пауна, нам бы хотелось же их генерировать вокруг нашего врага. Что для этого существует в EQS? Где он находится? Какой по умолчанию? Какой для начала создадим и от кого наследуясь? Что умеет возвращать данный класс? Какие 4 функции можно переопределить? Какую мы переопределили и как настроили ее? Возвращаемся в EQS и как выбираем нужный нам контекст?

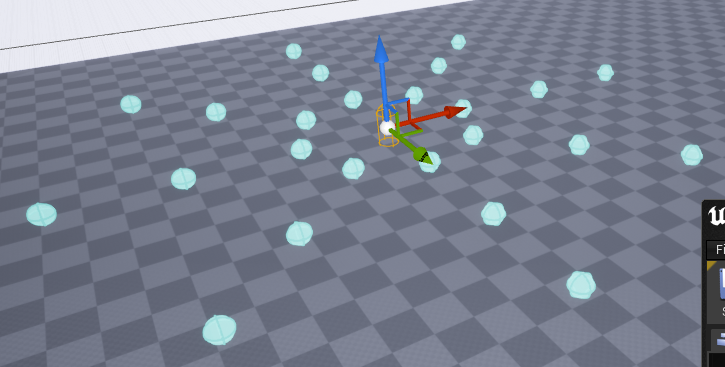
3. Теперь создадим C++-контекст, который как будет работать, от кого наследуемся? Какая есть функция в контексте по умолчанию в EQS? Как она работает? Как мы ее переопределяем в нашем классе + проперти добавили? Из какого ЗФ и как работает функция для получения блекборда? Что надо не забыть сделать при получении актора врага?

4. Теперь переходим в редактор и настраиваем наш EQS и BT. Как? Как отображать в дебаге текущий EQS?

1. Создадим генератор точек вокруг противника и NPC будет перемещаться с помощью этого генератора.

Создали новый EQS\_NextToEnemyLocation. Поменяем нашему TestPawn EQS Template на этот EQS.

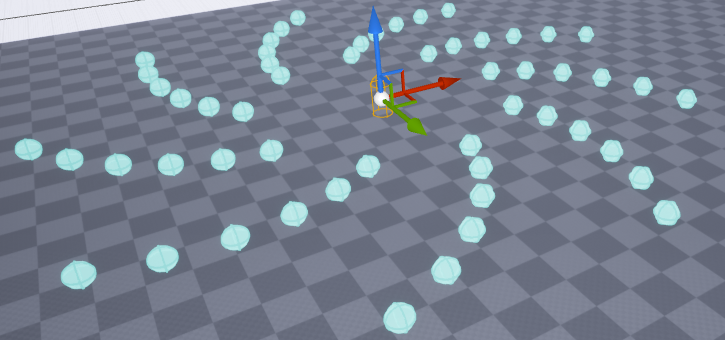
Открываем EQS, выбираем генератор Donut, его паттерн похож на бублик.



* В Projection Data выставили Trace Mode в Navigation;
* Inner Radius установили в 450;
* Outer Radius через Data Binding Random Number от 1000 до 1500;
* Number of Rings так же от 3 до 8;
* Points Per Ring так же от 4 до 20;
* А так же включим чекбокс Use Spiral Pattern – лучи будут немного закручены.

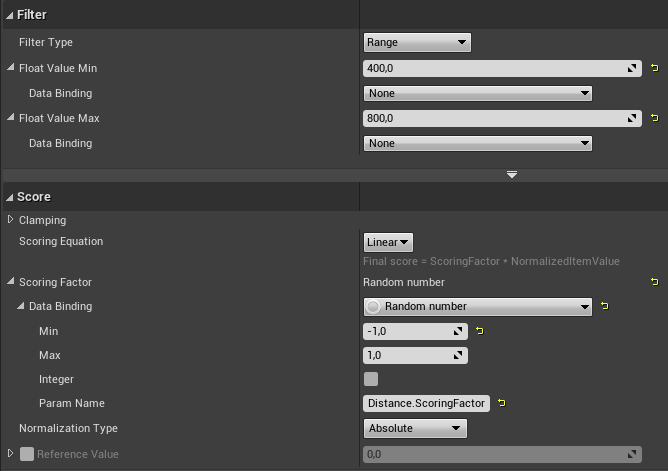
Изображение выглядит как текст

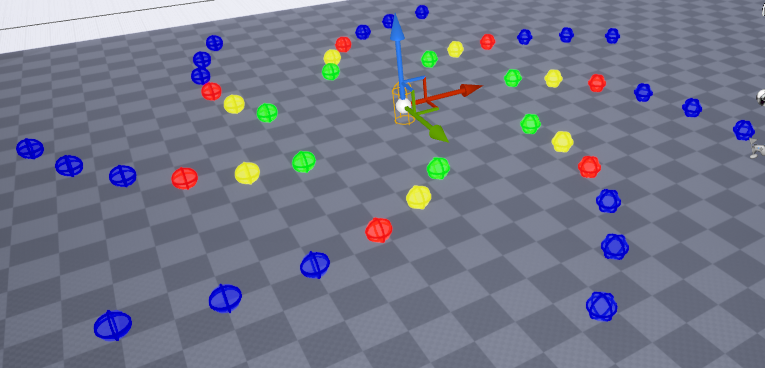
Автоматически созданное описание



2. Далее нам нужно получить нужную точку из нашего генератора. Добавляем тест Distance. В Filter Type установим Range, минимальное и максимальное значение от 400 до 800.

Сейчас максимальный вес получает точка, наиболее удаленная от центра, если же изменить Scoring Factor с 1 на -1, будет наоборот. Его можно так же сделать Random Number, поставим значения от -1 до 1 – таким образом максимальный вес будут иметь случайные локации.





Данные точки генерируются вокруг текущего пауна, нам бы хотелось же их генерировать вокруг нашего врага. Для этого в EQS существует специальный класс Context. Нажмем на наш генератор и найдем проперти Center. Контекст по умолчанию Querier выдает пауна, который владеет данный EQS – мы можем это поменять, создать свой собственный контекст, который будет возвращать актора нашего врага.

Изображение выглядит как текст

Автоматически созданное описание

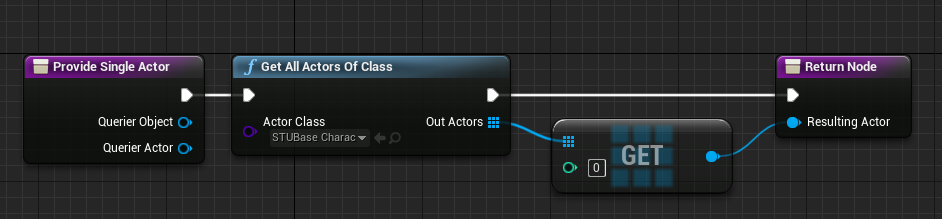
Для быстрого теста создали BP-класс контекста, наследуясь от EnvQueryContext\_BlueprintBase и назвав его EQSContextSTUCharacter – он будет возвращать самого первого персонажа со сцены.

Контекст умеет возвращать как акторов, так и локации. Нам необходимо переопределить одну из предложенных функций:

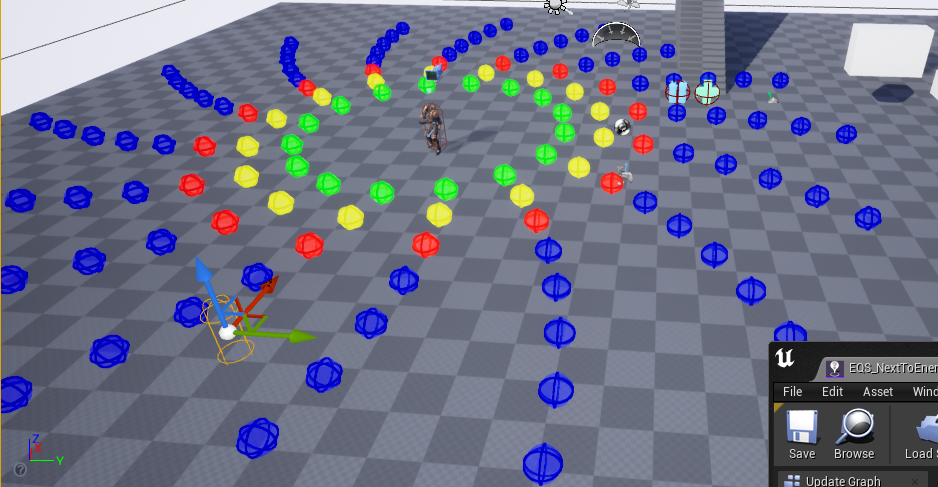
Изображение выглядит как текст

Автоматически созданное описание

Можно вернуть массив акторов, массив локаций, одного актора или одну локацию. Мы воспользуемся функцией ProvideSingleActor:



Получим всех акторов класса нашего персонажа и вернем первый элемент из данного массива. Возвращаемся в EQS и в проперти Center выбираем наш контекст у нашего генератора, а так же в проперти Distance To нашего теста, который определяет, относительно какого актора в нашем тесте высчитывается расстояние. Теперь все считается относительно нашего персонажа.



3. Посмотрим, как это работает в самой игре. Откроем BT и заменяем Template в RunEQSQuery на наш новый EQS. Все работает.

Теперь создадим C++-контекст, который будет возвращать указатель на врага, который находится в нашем блекборде, назовем его STUEnemyEnvQueryContext, наследуемся от EnvQueryContext.

Прежде чем разбирать нами созданный контекст, рассмотрим, как выглядит контекст, который используется EQS по умолчанию:

Изображение выглядит как текст

Автоматически созданное описание

Вычисление нужного актора происходит в функции ProvideContext – здесь просто вычисляется актор Owner для данного EQS и указатель на данный актор передается в функцию SetContextHelper.

В нашем же контексте нам нужно получить указатель на вражеского актора из Blackboard и воспользоваться той же функций SetContextHelper. Переопределим функцию ProvideContext в нашем классе, добавив еще проперти для имени ключа вражеского актора, который мы используем в Blackboard:

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

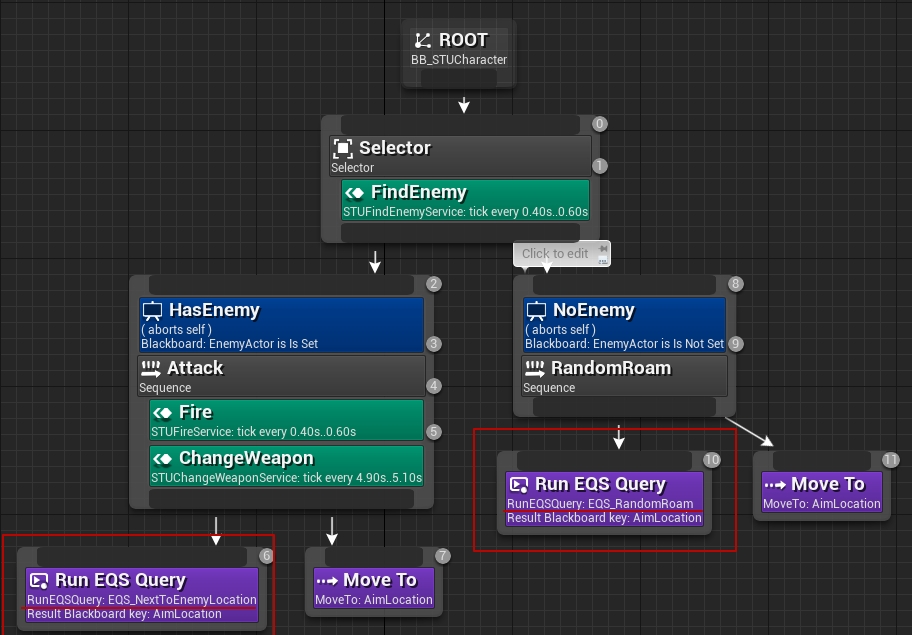
Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Подключили при этом 4 ЗФ. QueryOwner – это наш персонаж, который владеет данным EQS, через него получим указатель на BlackboardComponent с помощью функции GetBlackboard, которая внутри просто кастит передаваемый актор к пауну, получает его контроллер и у него получает компонент. Ну а далее просто получаем нашего EnemyActor в ContextActor и не забываем сделать каст, так как у нас GetValueAsObject возвращает UObject.

4. Теперь переходим в редактор и у нашего EQS заменяем везде контексты и обновим наш BT с логикой EQS:



Чтобы в дебаге отображался текущий EQS, надо нажать на «\*».