7 Лабораторная работа № 7. Настройка поведения искусственного интеллекта для объектов.

Цель работы: Изучение базовых алгоритмов искусственного интеллекта в Unreal Engine и их применение.

Порядок выполнения работы

1 Изучить основные теоретические положения, сделав необходимые выписки в конспект.
2 Получить задание у преподавателя, выполнить типовые задания.
3 Оформить отчет.

Требования к отчету

1 Цель работы.
2 Постановка задачи.
3 Результаты исследования.
4 Выводы.

Основные теоретические положения

Введение

В играх под AI(Artificial Intelligence, искусственный интеллект) понимается реализация интеллектуальных (кажущихся интеллектуальными) поведений для различных агентов. Типы AI могут существенно различаться:

- Как AI обрабатывает задачи;
- Как AI получает или воспринимает информацию о мире;
- Как AI интерпретирует геометрическое пространство мира;
- Как AI пытается повторять поведение, приближенное к человеческому поведению.

Главным компонентом любого игрового AI является формальная система принятия решений - по сути мозг AI. Этот «мозг» выполняет определенные действия, оценивая окружающую обстановку.

Создание AI включает в себя следующие шаги:

1) Создание класса неигрового персонажа
2) Создание AI контроллера
3) Добавление и настройка навигации
4) Создание дерева поведения

Создание неигрового персонажа и контроллера

Класс AI контроллер является классом, в котором принято писать специфическую для неигрового персонажа логику. Часть нод, например, ноды, связанные с созданием дерева поведения, можно создать только в классе AIController и наследуемых от него. Чтобы создать новый контроллер для персонажа, необходимо выбрать класс AIController в окне создания блупринт-класса(рисунок 7.1).

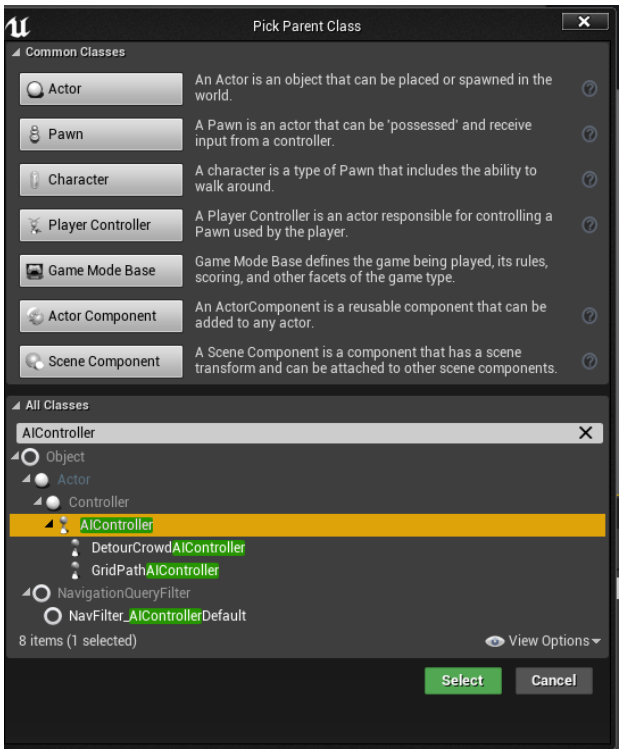

Рисунок 7.1 – Создание контроллера

После того, как контроллер создан, можно переходить к созданию персонажа. Обычно для объектов управляемого AI используются классы Pawn и Character. Класс Character наследуется от Pawn, и есть смысл в его использовании, если ваш агент — это классический гуманоидный персонаж. Для создания класса персонажа необходимо выбрать пункт Character в окне создания блупринт-класс(рисунок 7.2).

Рисунок 7.2 – Создание класса персонажа

После этого необходимо настроить меш персонажа. Для примера будет использоваться стандартный манекен. Настройка меша представлена на рисунке 7.3.

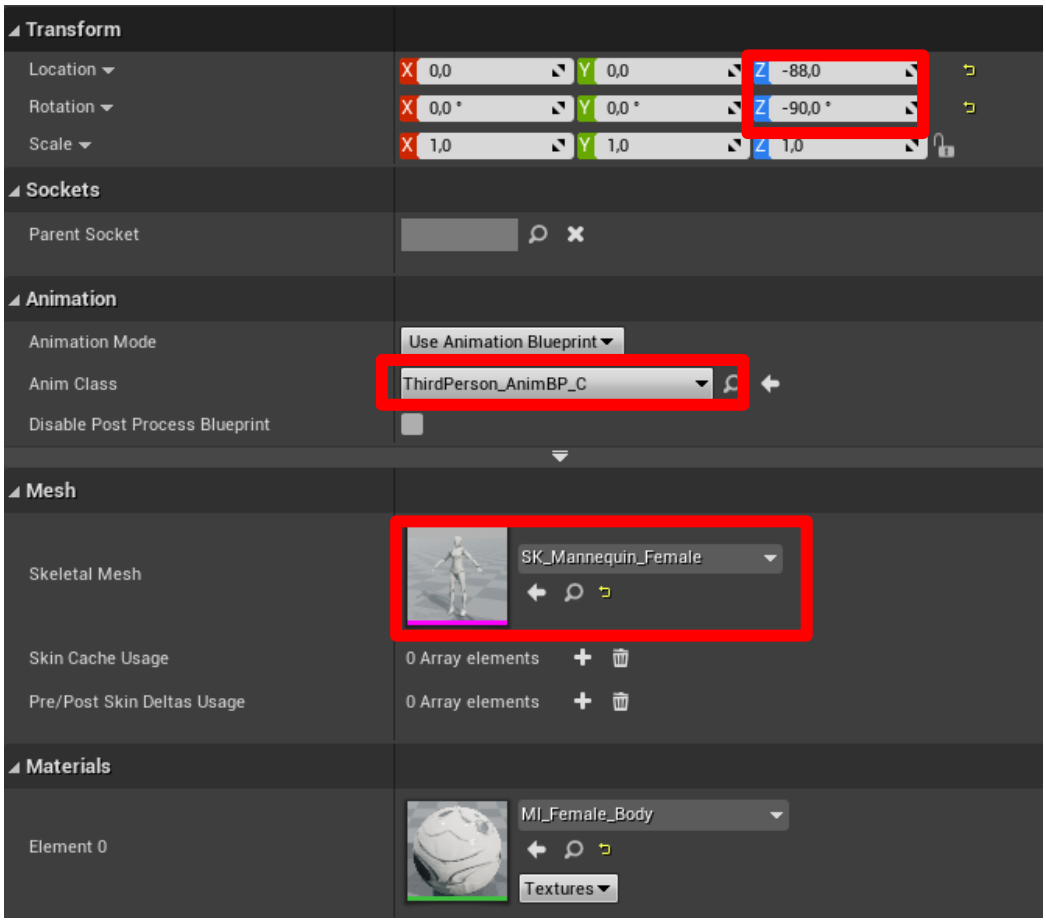

Рисунок 7.3 – Настройка меша

Последней настройкой персонажа является замена стандартного AIController на созданный вручную. Для этого необходимо выбрать нужный класс свойства AIController Class в общих настройках класса персонажа(рисунок 7.4). После этого можно разместить созданного персонажа на сцену.

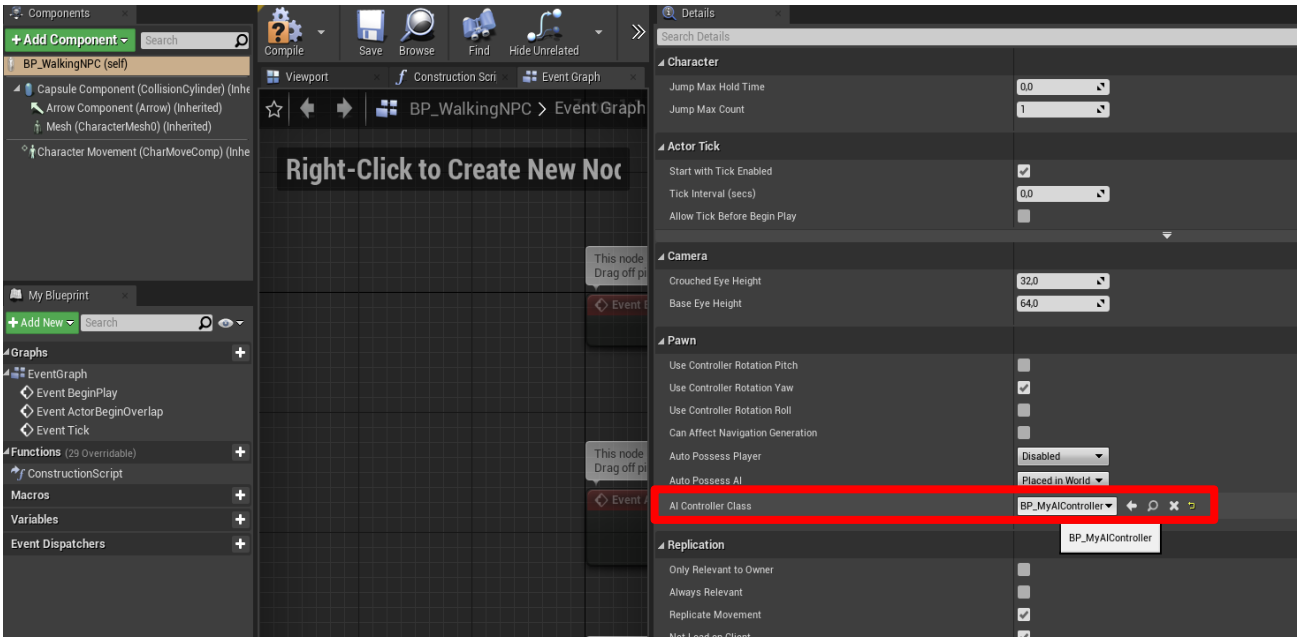

Рисунок 7.4 – Настройка контроллера для персонажа

Добавление навигации

AI может перемещаться по миру, только если он понимает, как это делать. AI воспринимает мир вокруг себя в очень упрощенном формате. Процесс идентификации путей по локации называется Pathfinding. Три основных типа навигационных систем это — Navigation Points, Navigation Grids и Navigation Meshes.

Для того, чтобы добавить навигацию на уровень, необходимо поместить туда Nav Mesh Bounds Volume(рисунок 7.5). Вы можете изменить масштаб актора, чтобы покрыть нужную площадь уровня.

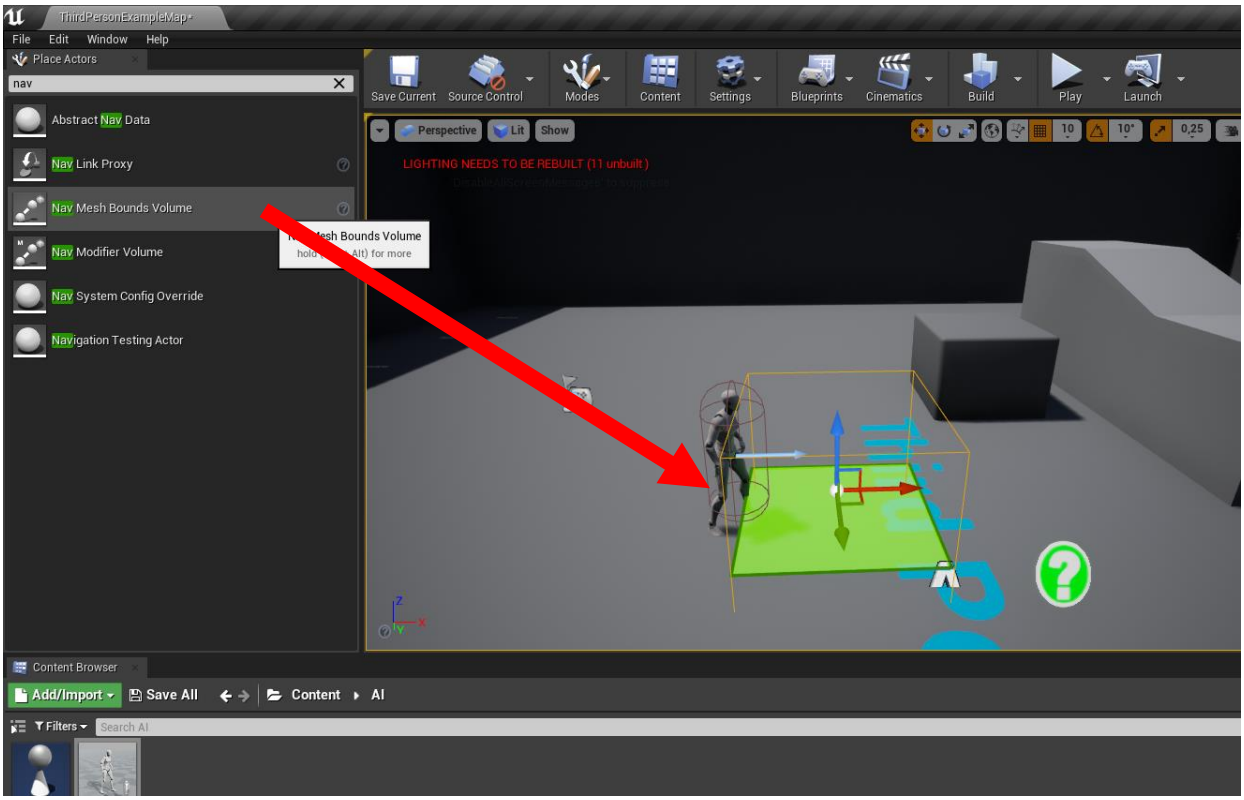

Рисунок 7.5 – Добавление Nav Mesh Bounds Volume

При нажатии на клавишу P будет подсвечена область Nav Mesh. Персонажи, управляемые AI, смогут перемещаться только по зеленой зоне. Также можно отобразить область Nav Mesh в окне «Show», выбрав пункт Navigation(рисунок 7.6).

Рисунок 7.6 – Отображение навигации

Behavior Tree и Blackboard

После создания персонажа с контроллером и добавления навигации можно переходить к настройке поведения неигрового персонажа. Основным инструментом для этого является Behavior Tree(дерево поведения). Оно представляет собой комбинацию из двух ассетов – Blackboard и непосредственно сам Behavior Tree.

Blackboard является памятью ИИ, которая содержит в себе переменные для использования их в дереве поведения. Для создания Blackboard и Behavior Tree необходимо выбрать соответствующие пункты в окне создания ассетов во вкладке Artificial Intelligence(рисунок 7.7).

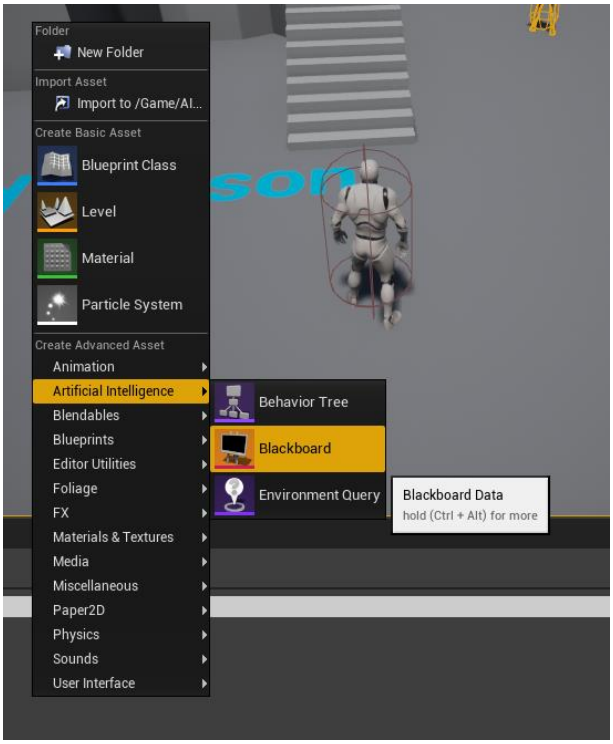

Рисунок 7.7 – Создание Blackboard и Behavior Tree

Для добавления новых переменных в Blackboard необходимо нажать кнопку New Key и выбрать необходимый тип переменной(рисунок 7.8). Стоит заметить, что все переменные, добавленные таким образом, будут иметь значение по умолчанию, которое невозможно изменить внутри Blackboard. Настройка таких переменных происходит вручную в блупринтах.

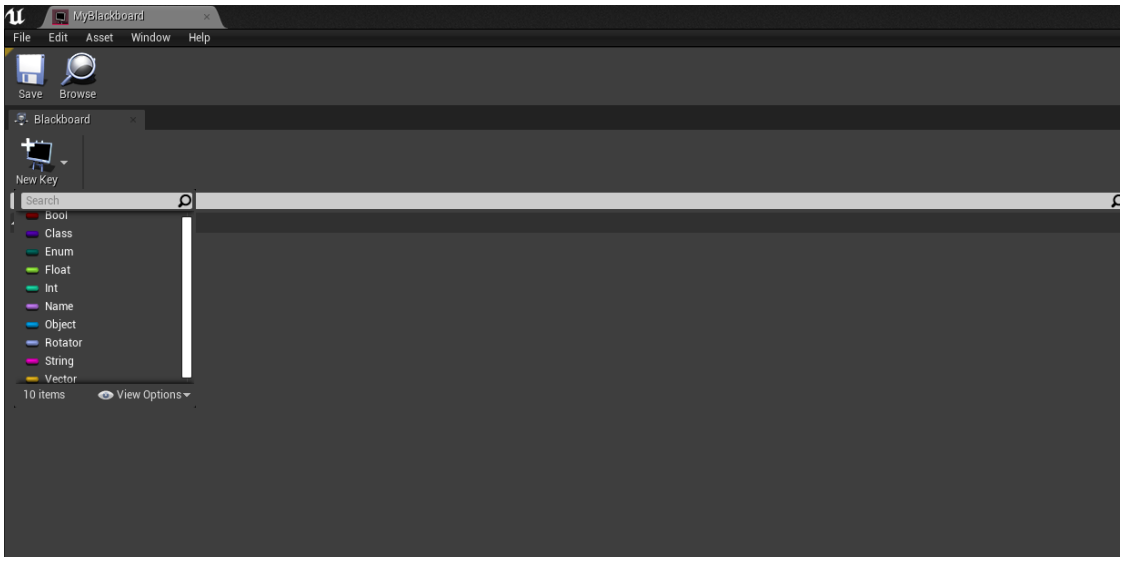

Рисунок 7.8 – Добавление переменных в Blackboard

Как только будет создан Behavior Tree необходимо назначить ему соответствующий Blackboard. В дереве поведения используется пять типов блоков:

1) **Root.** Выражение Root уникальное в Behavior Tree и является отправной точкой. Оно может иметь только одно соединение, и вы не можете подключить Decorators или Services к нему. Выражение Root не несет в себе никаких свойств, но выбрав это выражение вы увидите свойства Behavior Tree в окне Details, где вы можете задать Blackboard ассет вашего Behavior Tree. Нод Root можно подключать только к нодам типа Composite.
2) **Composite.** Эти выражения определяют основания ветви и базовые правила, которые определяют, как эта ветвь выполняется.
3) **Decorator.** Выражения также известные как “условные”. Их прикрепляют к другим выражениям и создают выполняемые решения для ответвления в дереве.
4) **Task.** Непосредственно инструкции поведения.
5) **Service.** Эти выражения прикрепляют к Composite выражениям, и выполняют с определенной частотой до тех пор, пока ветка выполняется. Они часто используются для создания проверок и обновления Blackboard.

В Unreal Engine можно использовать как имеющиеся Tasks, Services и Decorators, так и создавать новые с помощью соответствующих кнопок на панели Toolbar(рисунок 7.9).

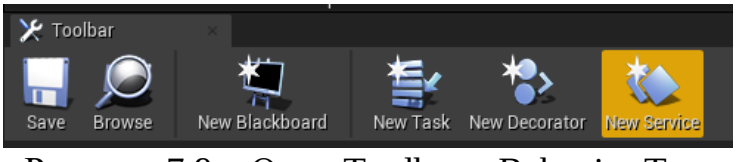

Рисунок 7.9 – Окно Toolbar в Behavior Tree

Для того, чтобы добавить композитные ноды или таски необходимо в пустом месте Behavior Tree нажать ПКМ(рисунок 7.10).

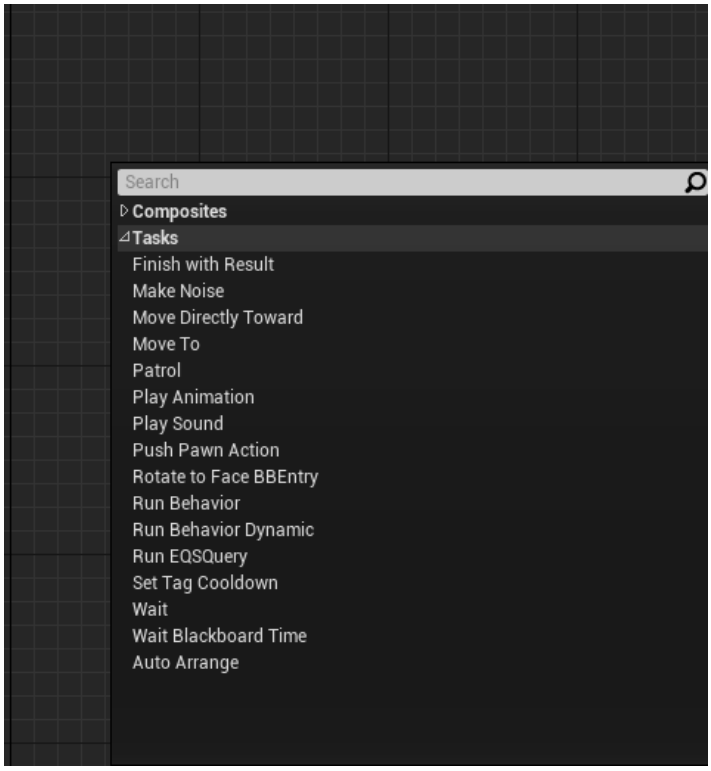

Рисунок 7.10 – Добавление нод в дерево поведения

После создания и настройки дерева поведения для неигрового персонажа, необходимо запустить выполнение дерева в созданном AIController. Функция Run Behavior Tree запускает выполнение дерева поведения, а функция Use Blackboard подключает использование конкретного блэкборда(рисунок 7.11).

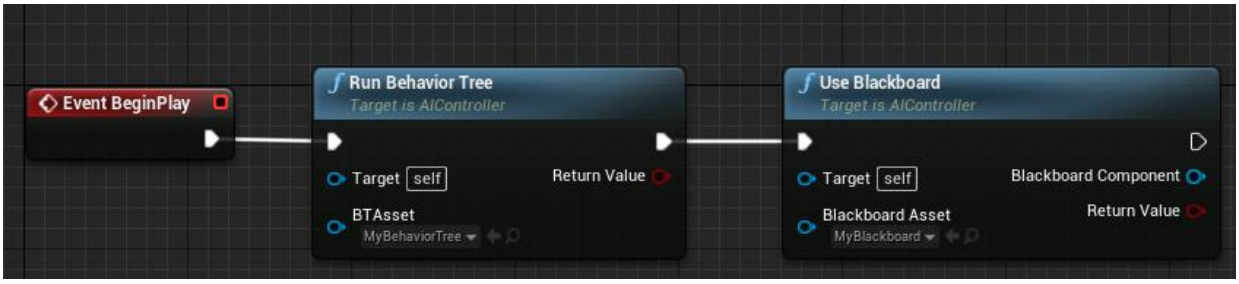

Рисунок 7.11 – Запуск дерева поведения в AIController

Пример 1. Создать неигрового персонажа и дерево поведения. Необходимая модель поведения – патрулирование(движение к различным случайным точкам в определенном радиусе), следование за главным персонажем при его приближении и возвращение к патрулированию при его отдалении.

Решение

Для начала необходимо добавить необходимые переменные в созданный Blackboard. В данном случае понадобятся две переменные: PlayerReference типа Object и IsPlayerNear типа Bool(рисунок 7.12). Для переменной PlayerReference дополнительно можно указать BaseClass и установить значение ThirdPersonCharacter.

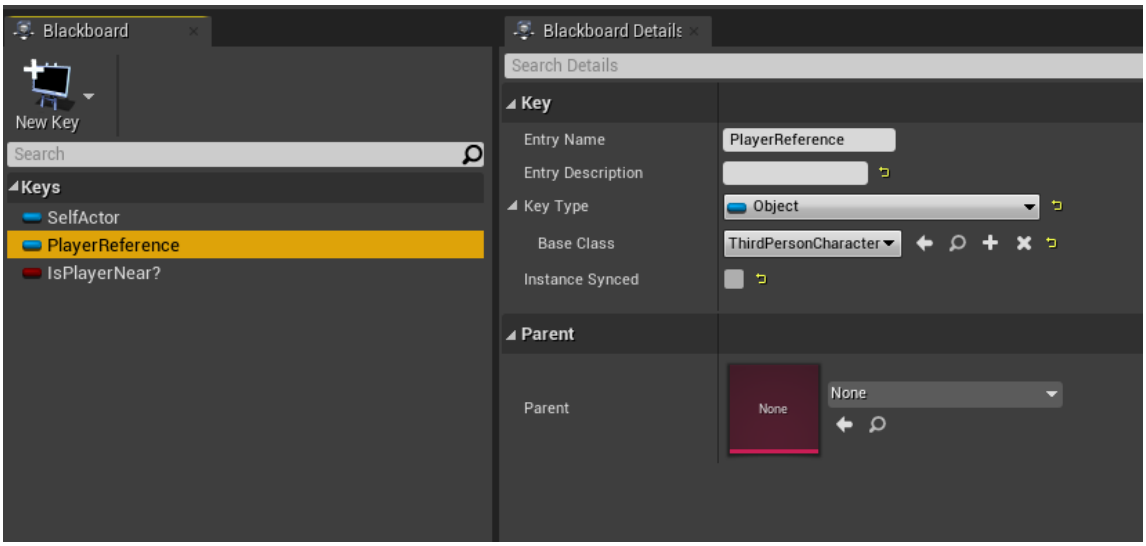

Рисунок 7.12 – Добавление необходимых переменных в Blackboard

После того, как необходимые переменные добавлены, можно переходить к настройке дерева поведения. Для логики патрулирования создадим новую таску. По умолчанию она создастся в той же папке, где находится дерево поведения, со стандартным названием. Переименуем ее в PatrolTask. После этого она появится в списке доступных для добавления нод. Добавим ее в дерево поведения и присоединим к композитной ноде Sequence, добавим стандартную таску Wait, чтобы движение происходило с определенными промежутками времени(рисунок 7.13).

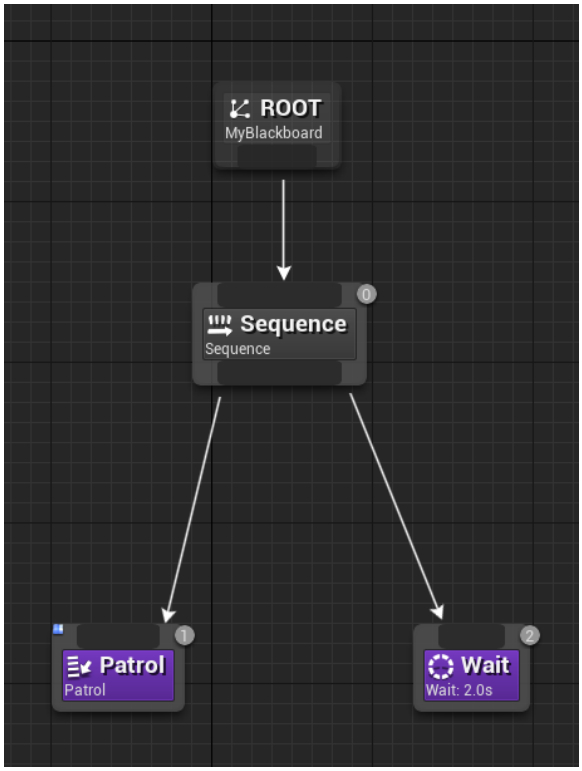

Рисунок 7.13 – Добавление ноды Patrol

Теперь можно перейти к наполнению таски. Для ее открытия необходимо дважды нажать на нее в дереве поведения или в Content Browser. В таске есть возможность добавить автоматически срабатываемые события, которые будут выполняться в случае захода в данную ветвь дерева, похожие по своей сути на Event BeginPlay, Event Tick в блупринтах обычных Actor’ов.

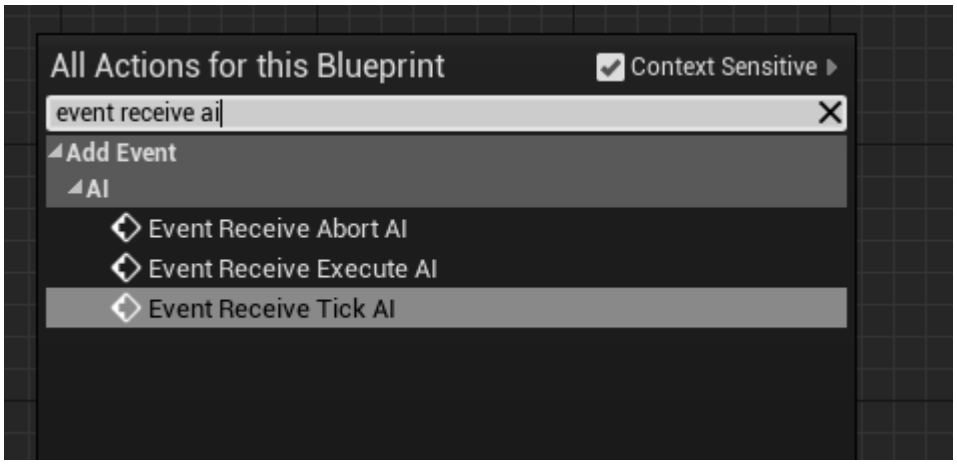

Рисунок 7.14 – События в таске

Для нашего примера необходимо добавить событие Event Receive Execute AI, которое выполнится один раз при вызове этой таски в дереве поведения. Движение персонажа происходит с помощью функции AI MoveTo, которая может переместить его либо к определенной точке, либо к конкретному объекту, если существует навигация, позволяющая добраться до цели. Необходимая локация, к которой необходимо проследовать, будет задаваться случайно и находиться с помощью функции GetRandomReachablePointInRadius с заданным радиусом и центром в месте текущего нахождения персонажа(функция GetActorLocation). Как только движение закончится, необходимо сообщить дереву поведения, что таска выполнилась окончательно и можно переходить к выполнению следующей. Для этого используется функция FinishExecute, которая подсоединяется к исполняемому пину On Success ноды AI MoveTo. Таска Patrol представлена на рисунке 7.15.

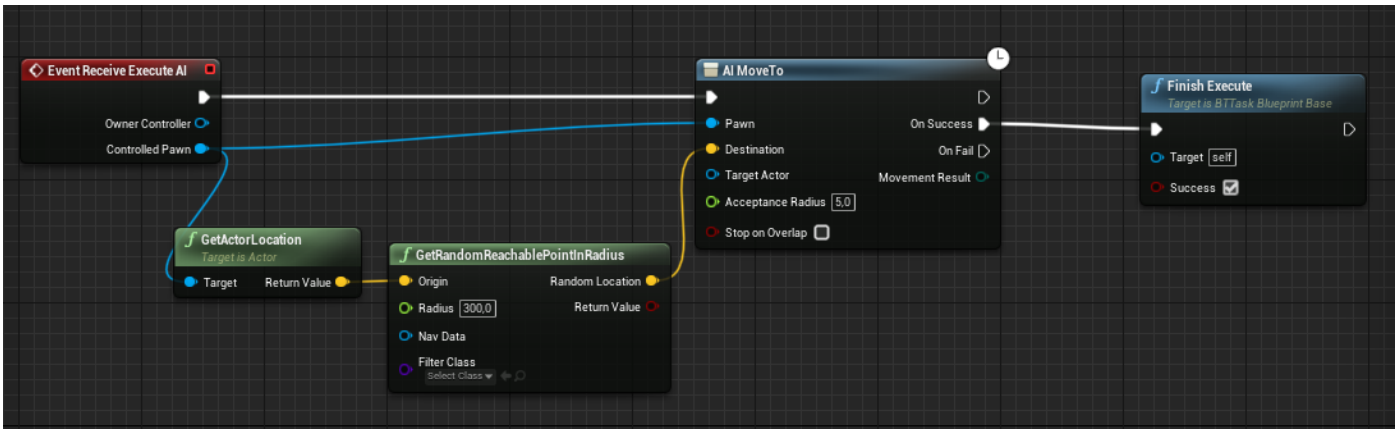

Рисунок 7.15 – Таска Patrol

Поскольку патрулирование должно заканчиваться в тот момент, когда рядом окажется главный персонаж, добавим в созданный AICharacter компонент SphereCollision. При попадании главного персонажа в область триггера необходимо устанавливать значение переменной IsPlayerNear, созданной в Blackboard, на True и обратно на False при выбегании из зоны триггера. Для этого используется функция Set Value As Bool. Данной функции необходимы ссылка на конкретный Blackboard, которую можно получить с помощью функции GetBlackboard, и имя переменной, которое будет передаваться с помощью функции MakeLiteralName(рисунок 7.16).

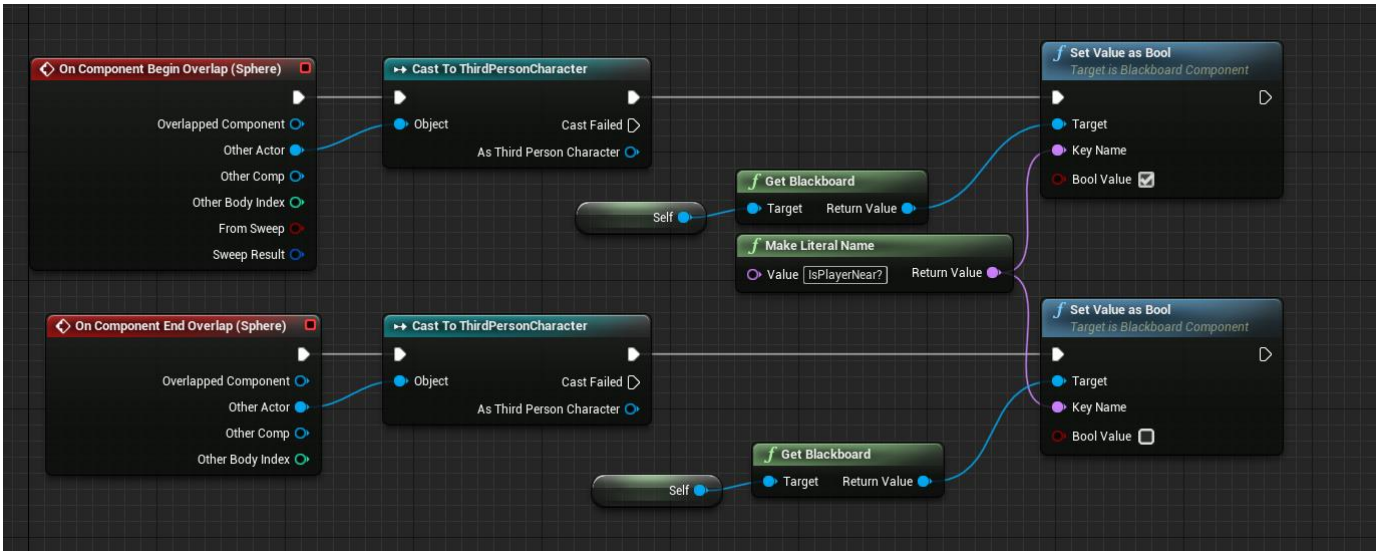

Рисунок 7.16 – Изменение переменной при попадании персонажа в триггер

Теперь модифицируем дерево, добавив проверку на состояние переменной IsPlayerNear. Для этого можно использовать стандартный Decorator типа Blackboard. Чтобы добавить проверку, необходимо нажать ПКМ на нужный блок(в нашем случае Sequence) и выбрать соответствующий декоратор(рисунок 7.17).

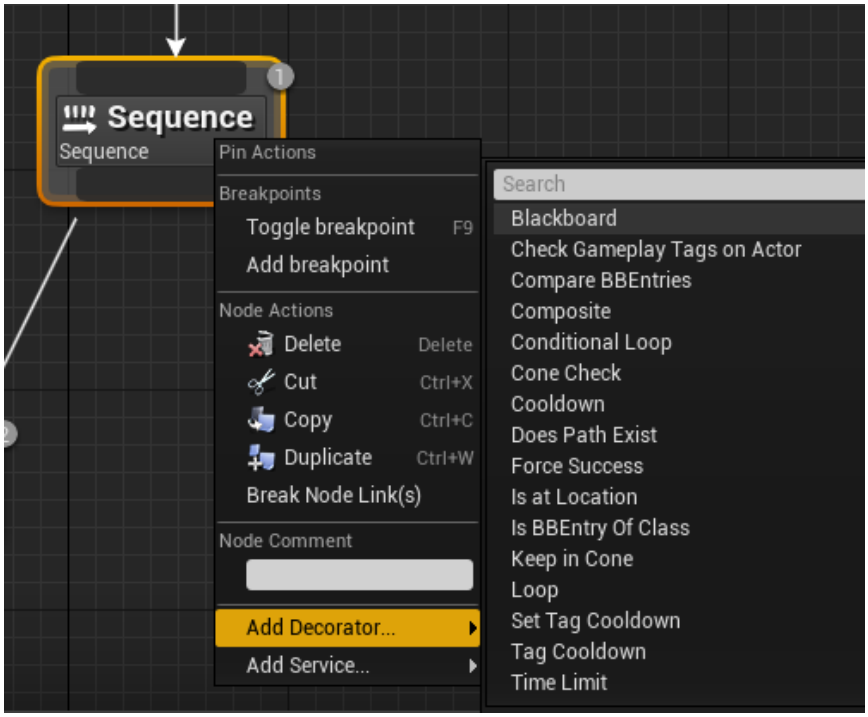

Рисунок 7.17 – Добавление Decorator

В окне деталей данного декоратора следует выбрать нужную переменную и установить свойство KeyQuery на Is Not Set, поскольку блок патрулирования должен срабатывать при значении False переменной IsPlayerNear.

Чтобы добавленный Decorator работал корректно, перед Sequence необходимо дополнительно установить блок Selector. Модернизированное дерево поведения представлено на рисунке 7.18.

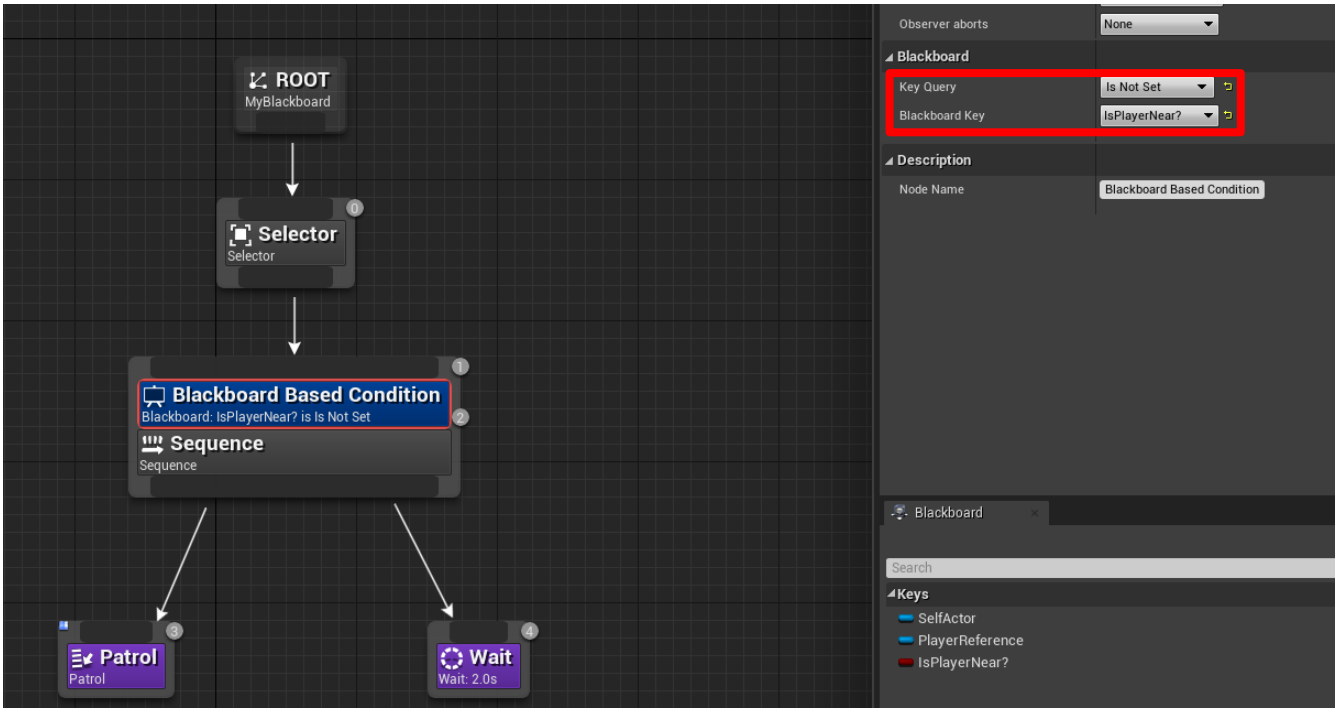

Рисунок 7.18 – Дерево поведения после добавления проверки

После этого можно перейти к созданию логики следования за персонажем. Добавим в дерево поведения еще одну ветвь. Для передвижения можно использовать стандартную таску MoveTo, в которой устанавливается значение Blackboard Key на созданную в Blackboard переменную PlayerReference, которая будет хранить в себе ссылку на главного персонажа(рисунок 7.19).

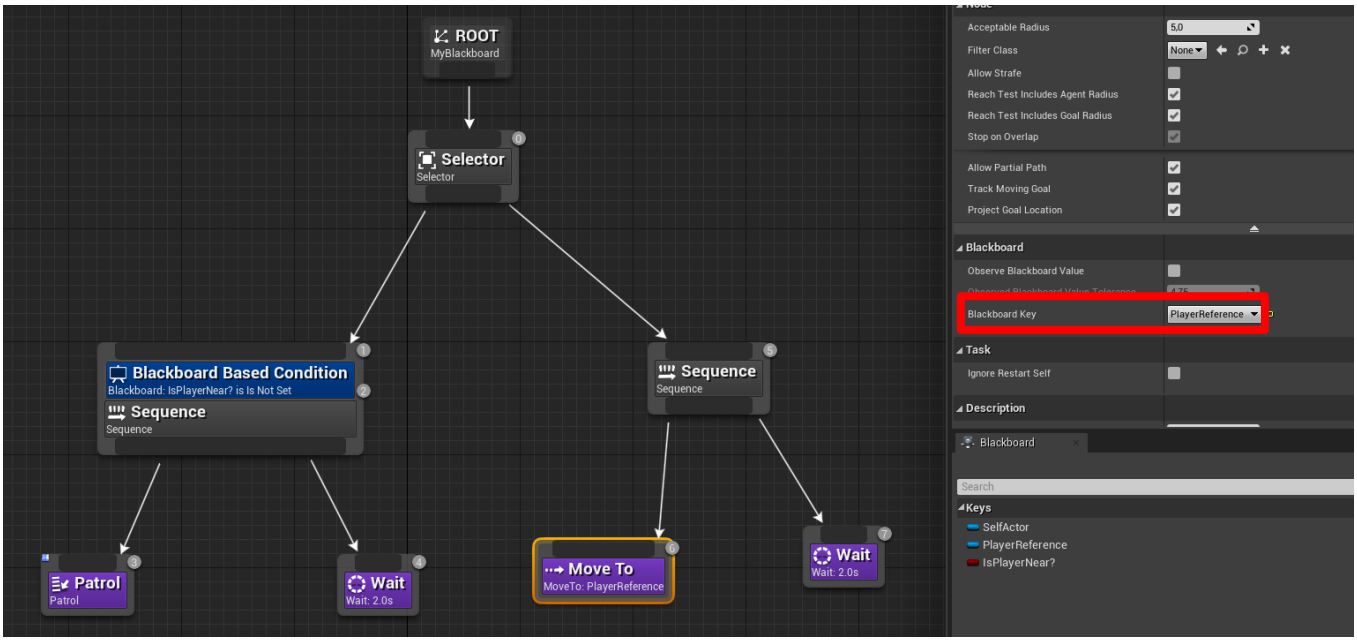

Рисунок 7.19 – Добавление ветви следования

Последнее, что осталось сделать, это установить значение переменной-ссылке на персонажа. Сделать это можно при помощи функции Set Value As Object в классе AIController. Для получения ссылки на главного персонажа можно использовать функцию GetPlayerCharacter.

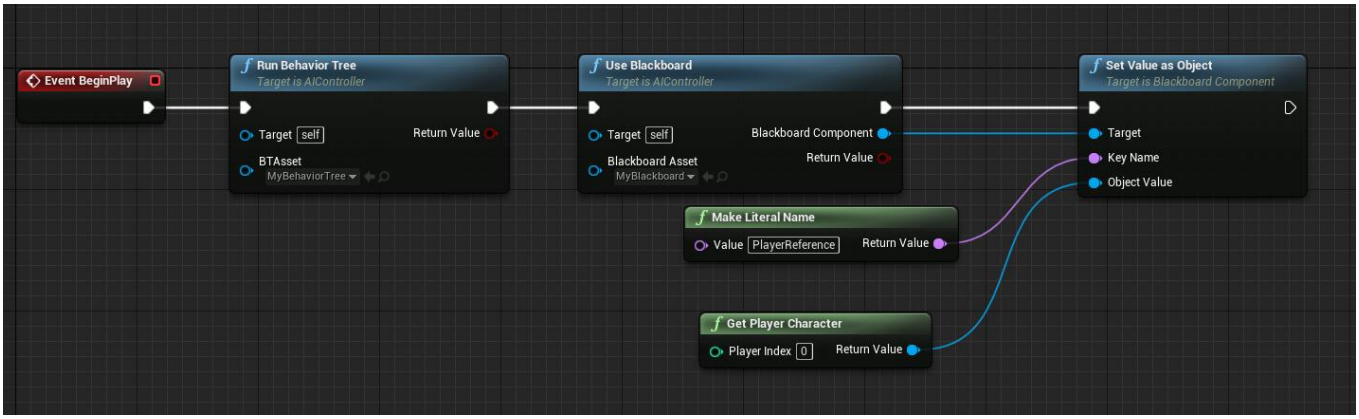

Рисунок 7.20 – Установка значения переменной в AIController

Контрольные вопросы и задания

Задание. Выполнить пример 1.