УДК 004.42

РАЗРАБОТКА АЛГОРИТМОВ ПОВЕДЕНИЯ ПРОТИВНИКА С ИСПОЛЬЗОВАНИЕМ МОДЕЛЕЙ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Дужий А.А., Терещенко Т.М.

DEVELOPMENT OF AN ALGORITHM OF THE ENEMY'S BEHAVIOR USING MODELS OF ARTIFICIAL INTELLIGENCE

Duzhiy A., Tereshchenko T.

В статье описан процесс разработки алгоритма поведения персонажа-противника с использованием моделей искусственного интеллекта в среде Unity3D. Разработаны скрипты механики поведения на языке программирования C#. Разработанная модель поведения может быть использована в прототипе игрового приложения.

Ключевые слова: *игровое приложение, скрипт, алгоритм, , сцена, unity3d, C#, язык программирования, ИИ, искусственный интеллект.*

Введение. Индустрия компьютерных игр - это сектор экономики, связанный с разработкой, продвижением и продажей компьютерных игр. На данном этапе развития он очень активно совершенствуется. Индустрия компьютерных игр охватывает большое количество специальностей, по которым работают тысячи людей по всему миру. Все большее количество людей интересуются этим направлением, а параллельно с этим активно растет спрос на хорошие качественные игры. Важными задачами в разработке компьютерных игр являются разработка алгоритмов поведения противника с использованием моделей искусственного интеллекта.

Цель статьи – описать процесс разработки алгоритма поведения противника с использованием моделей искусственного интеллекта для последующего его использования в прототипе игрового приложения.

Игровой искусственный интеллект — это набор программных методик, которые используются в компьютерных играх для создания иллюзии интеллекта в поведении персонажей, управляемых компьютером. Игровой ИИ, помимо методов традиционного искусственного интеллекта, включает также алгоритмы теории управления, робототехники, компьютерной графики и информатики в целом.

Реализация ИИ сильно влияет на геймплей, системные требования и бюджет игры, и разработчики балансируют между этими требованиями, стараясь сделать интересный и нетребовательный к ресурсам ИИ малой ценой. Поэтому подход к игровому ИИ серьёзно отличается от подхода к традиционному ИИ — широко применяются разного рода упрощения, обманы и эмуляции. Например: с одной стороны, в шутерах от первого лица безошибочное движение и мгновенное прицеливание, присущее ботам, не оставляет ни единого шанса человеку, так что эти способности искусственно снижаются. С другой стороны — боты должны делать засады, действовать командой и т. д., для этого применяются «костыли» в виде контрольных точек, расставленных на уровне.

Персонажей компьютерных игр, управляемых игровым искусственным интеллектом, делят на:

- неигровые персонажи (англ. Non-player character, NPC) — как правило, эти персонажи являются дружественными или нейтральными к человеческому игроку.
- боты (англ. Bot) — враждебные к игроку персонажи, приближающиеся по возможностям к игровому персонажу; против игрока в любой конкретный момент сражаются небольшое количество ботов. Боты наиболее сложны в программировании.
- мобы (англ. Mob) — враждебные к игроку «низкоинтеллектуальные» персонажи. Мобы убиваются игроками в больших количествах ради очков опыта, артефактов или прохождения территории.

Согласно дизайну игры, персонаж - противник должен в течение всей игры непрерывно преследовать игрока и наносить ему повреждения, если подойдет слишком близко. Также он должен уметь телепортироваться через препятствия, если их невозможно обойти.

Для создания полного алгоритма поведения противника работу было решено разделить на этапа:

- Написание скрипта преследования игрока

- Написание скрипта телепортации
- Создание алгоритма нанесения повреждений игроку

Создание целостной модели поведения из нескольких отдельных скриптов позволит облегчить процесс разработки, а также даст возможность более детально реализовать каждую часть. Разработанные алгоритмы поведения персонажа-потивника можно увидеть на Рис.1 и Рис.2

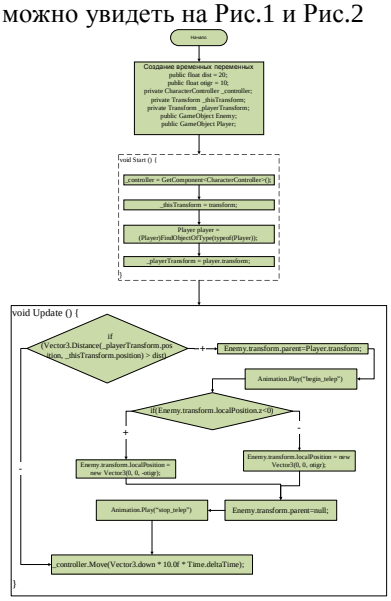

Рис.1 Алгоритм телепортации

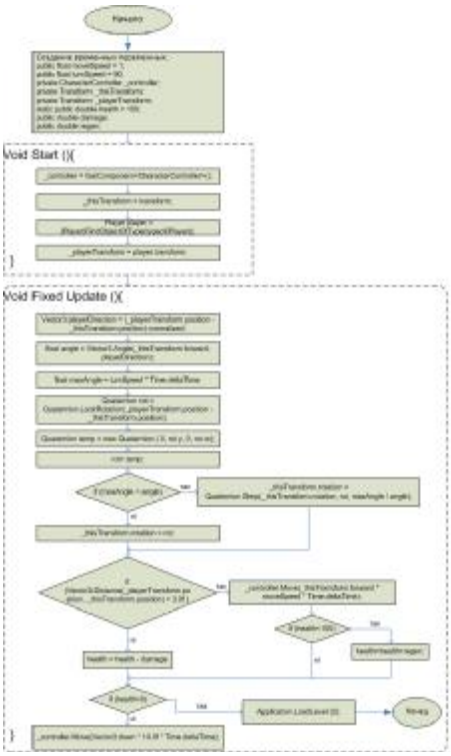

Рис.2 Общая модель поведения противника

После создания всех алгоритмов необходимо написать код скриптов. Код был написан на языке C#, поскольку для данного движка он наиболее удобен и имеет высокую производительность.

Для начала был создан скритп, который получает положение игрока на карте и его координаты. Для этого использовалась Unity-функцию «Transform», которая хранит в себе необходимые нам координаты. Созданный скрипт Player с помощью Drag & Drop был перенесен на нашего игрока, который, для удобства, также был назван Player.

После этого был реализован алгоритм преследования игрока противником. Для этого создали скрипт для врага, который обращался к скрипту Player, и позволили ему использовать функции и переменные, находящиеся внутри скрипта Player. Для этого использовалась функция «FindObjectOfType(typeof())», которая позволяет настроить общения скриптов друг с другом. После мы заставляли врага двигаться, используя метод «.Move», но перед этим сделав проверку с помощью оператора «if{}» , чтобы расстояние между врагом и игроком была более трех единиц, только тогда выполнять движение. С помощью функции «Quaternion», которая позволяет реализовать поворот, и функции Vector3.Angle, которая вычисляет угол поворота, осуществляли поворот врага к игроку, если вектор его движения направлен не к персонажу. Также к скрипту был добавлен небольшого размера код, который отвечает за гравитацию. Реализуется он с помощью функции .Move(Vector3.Down*10), которая, как мы видим, выполняет движение, направленное вниз с силой в 10 единиц.

Следующей задачей было написание скрипта телепортации врага. Он нужен для того, чтобы враг мог преодолевать препятствия на карте а также чтобы расстояние между игроком и врагом не была слишком большой, ведь для удобства геймплея мы делаем скорость движения игрока значительно больше, чем скорость врага.

Для начала, как и в предыдущем скрипте, нам нужно обратиться к функциям скрипта Player («FindObjectOfType(typeof())») и получить расположение игрока («Transform»). Также с помощью функции Vector3 вычислить вектор направления к игроку, а с помощью ее метода .Distance - расстояние между врагом и игроком. Далее мы создаем две переменные, которые будут отвечать за максимальное расстояние, на которое игрок может отойти от врага (dist), и расстояние от игрока, на которое будет перемещаться враг (otigr). Задаем им свойство public, чтобы можно было менять их прямо в движке Unity. Далее, с помощью оператора "if{}", проверяем: расстояние между игроком и врагом больше, чем значение переменной dist? Если так, то перемещаем врага к игроку на расстояние, равное переменной otigr с помощью метода ".Parent" функции " transform" и оператора «if{}». Скрипт телепортации готов.

Переходим к следующему уровню взаимодействия врага и игрока - добавление в игру показателя здоровья игрока и зависимость его пере-

менной от действий врага. Для этого можно редактировать старый скрипт Enemy, если нет желания создавать новый. Итак, добавляем в скрипт три новые переменные: health - текущий показатель здоровья, damage - уровень наносимых повреждений игроку и regen - скорость восстановления здоровья игрока. Теперь с помощью оператора «if{}» и проверок задаем взаимодействие врага и переменной здоровья игрока. Если расстояние между врагом и игроком равна 3 единицы, показатель здоровья уменьшается на значение, равное переменной damage, если расстояние больше 3 единиц - увеличивается на значение, равное переменной regen. Для большей атмосферности игры задаем значение damage вдвое больше, чем regen. Чтобы процесс восстановления здоровья не был бесконечный, добавим еще одну проверку: if(health<100){health=health+regen;}. Чтобы увидеть, как меняется уровень нашего здоровья, используем функцию Debug.Log(health), которая будет выводить значение переменной health в рабочую консоль (Рис. 3).

Рис. 3 Консольный отчет нанесения урона игроку противником

Когда значение переменной health падает до отметки 0, игра считается проигранной. Таким образом, скрипт полностью готов, осталось только откомпилировать его.

Выводы. В результате работы был разработан алгоритм поведения игрового персонажа-противника с использованием моделей искусственного интеллекта. Эта модель может использоваться в прототипе компьютерной игры.

Для использования в релизных версиях игры необходимо усовершенствовать алгоритм и добавить больше возможных реакций окружающего мира на действия игрока.

Литература

1. http://unity3d.com/ - официальный сайт компании Unity
2. http://docs.unity3d.com/Documentation/ScriptReference/ - библиотека функций Unity
3. http://www.unity3dstudent.com/ - уроки Unity3D
4. http://unity3d.ru – русскоязычное сообщество Unity3D.
5. http://www.digitaltutors.com/training/unity-tutorials - уроки Unity3D
6. http://habrahabr.ru/post/128711/ - статья про скрипты Unity3D

References

1. http://unity3d.com/ - Unity official website - oficial'nyj sajt kompanii Unity
2. .http://docs.unity3d.com/Documentation/ScriptReference/ - biblioteka funkcij Unity
3. http://www.unity3dstudent.com/ - uroki Unity3D
4. http://unity3d.ru – russkojazychnoe soobshhestvo Unity3D.
5. http://www.digitaltutors.com/training/unity-tutorials - uroki Unity3D
6. http://habrahabr.ru/post/128711/ - stat'ja pro skripty Unity3D

Дужий А.О., Терещенко Т.М. Розробка алгоритмів поведінки супротивника з використанням моделей штучного інтелекту

У статті описано процес розробки алгоритму поведінки персонажа-супротивника з використанням моделей штучного інтелекту в середовищі Unity3D. Розроблено скрипти механіки поведінки на мові програмування C#. Розроблена модель поведінки може бути використана в прототипі ігрового додатку.

Ключові слова: ігровий додаток, скрипт, алгоритм, сцена, unity3d, C#, мова програмування, ШІ, штучний інтелект.

Duzhiy A., Tereshchenko T. Development of an algorithm of the enemy's behavior using models of artificial intelligence

This article describes the creation process of an algorithm for enemy's behavior in Unity3D environment, using models of artificial intelligence. Scripts of mechanical behavior, based on previously developed algorithm, were written on the programming language C#. The developed model of an enemy's behavior can be used in the prototype of game.

Keywords: game application, scripts, algorithm, scene, unity3d, C#, AI, artificial intelligence, models.

Дужий Андрій Олександрович. Студент п'ятого курсу кафедри комп'ютерних наук Східноукраїнського національного університету імені Володимира Даля. Email: dronaut@gmail.com

Терещенко Тетяна Михайлівна. Доцент, к.т.н. кафедри комп'ютерних наук Східноукраїнського національного університету імені Володимира Даля. Email: whiteopal@yandex.ru

Рецензент: Заслужений діяч науки і техніки України, д.т.н., проф. Ульшин В.О.

Стаття подана 22.04.2014