Слайд

Здравствуйте. Меня зовут Приходько Евгений, моя работа называется «Исследование методов моделирования принятия решений на примере ботов в компьютерной игре».

Слайд

В настоящее время широко распространены системы, принимающие решения. Они основаны на модели интеллектуального агента, имеющего датчики для получения информации, решающее устройство и исполнительные механизмы для воздействия на среду. Решающее устройство может быть реализовано с помощью различных методов и алгоритмов принятия решений, а значит есть потребность в сравнении этих методов для обоснования их выбора при создании новых систем.

Слайд

Таким образом, цель моей работы – сравнить эффективность методов моделирования принятия решений. Для достижения этой цели нужно рассмотреть существующие методы, реализовать агентов, используя эти методы, и сравнить их по различным критериям оценки эффективности.

Слайд

Сравнивать методы необходимо на примере конкретной задачи. В данной работе рассматривается разработка ботов для компьютерной игры в жанре topdown shooter. В игре два персонажа двигаются по арене и стреляют друг в друга. Попадания уменьшают количество очков здоровья у персонажей. Если они уменьшаются до нуля, игрок проигрывает. Стены блокируют обзор, укрытия блокируют часть выстрелов, а в углах расположены аптечки – предметы, восстанавливающие очки здоровья.

Под принятием решения имеется в виду выбор определенной стратегии поведения, которая даст максимальное игровое преимущество. Эффективность метода принятия решений определяется по способности реализованного агента хорошо играть, т.е. по количеству выигрышей и проигрышей.

Слайд

Перейдем к рассмотрению методов. Первый метод – это конечные автоматы. Состояниями автомата являются принимаемые решения, в данном случае – это стратегии поведения агента. С учетом правил игры были определены 4 стратегии: поиск противника, атака, защита и отступление. В качестве условий переходов используется состояния среды, при которых агент должен изменить стратегию поведения. На слайде приведен граф итогового конечного автомата. Условия переходов выбирались на основании поведения реального игрока, а затем корректировались в ходе функциональных тестов, т.е. игр бота против живого игрока.

Конечные автоматы простоты в разработке для систем с небольшим числом состояний, однако при увеличении количества состояний, переходов или усложнении модели среды их становится сложно поддерживать из-за большого количества связей между компонентами.

Слайд

Следующий метод – деревья поведения. Возможны два подхода к проектированию деревьев поведения применительно к рассматриваемой задаче. Первый вариант – это дерево на основе селектора. В нем каждая ветвь реализует отдельную стратегию поведения, а само дерево осуществляет выбор нужной ветви. Второй подход – дерево на основе последовательности. В этом случае ветви реализуют отдельные компоненты системы, а дерево осуществляет их последовательный обход. В данной работе был выбран второй подход, так как в противном случае возникало дублирование элементов из-за того, что стратегии имели общую структуру. Стратегии были декомпозированы на отдельные компоненты: управление стрельбой, управление движением и анализ местности для выбора наиболее выгодной позиции.

Слайд

К преимуществам деревьев поведения можно отнести модульность. Независимость узлов дерева позволяет использовать их в разных ветвях одного дерева, в разных деревьях и даже в разных проектах. Декомпозиция стратегий на отдельные компоненты позволила уменьшить связность объектов, а значит облегчить разработку и поддержку системы. Добавление новых компонентов потребует только добавления новых ветвей без изменения уже реализованных.

К недостаткам можно отнести меньшее быстродействие из-за того, что для принятия решения в общем случае требуется произвести полный обход дерева в глубину.

Слайд

Следующий метод – нечеткая логика. Для реализации системы, использующей нечеткую логику, требуется определить входные и выходные лингвистические переменные, а также правила нечеткого вывода. Количество входных переменных влияет на количество правил вывода, которые требуется определить при создании агента. Большое количество правил вывода значительно усложняет разработку, поэтому требуется выделить небольшое количество важных переменных.

В рассматриваемой игре к таким параметрам относятся количество очков здоровья агента и противника, а также видимость противника. Первые два важны, так как влияют на выбор стратегии поведения. Например, если очков здоровья у агента много, а у противника мало, то можно вести себя более агрессивно, а если наоборот – лучше предпочесть защиту. Третий параметр важен, так как без видимости противника многие стратегии просто бессмысленны.

Для каждой переменной были определены термы, т.е. лингвистические значения, которые они могу принимать. Для очков здоровья это «мало», «средне» и «много». Для функций принадлежности этих термов выбраны трапеции для простоты проектирования и вычислений. Для видимости противника используются два синглтона, так как эта переменная аналогична булевой логической переменной.

Слайд

Аналогичным образом определены выходные переменные. Их количество не влияет на количество правил вывода, поэтому можно использовать любое их количество, которое требуется для управления агентом. В ходе исследования были выделены следующие переменные, представленные на слайде. Дистанция до противника: в случае атаки следует сохранять близкую дистанцию, чтобы нанести больше урона, при защите - среднюю, чтобы уменьшить количество получаемого урона, а при отступлении – дальнюю. Следующая важная переменная - радиус поиска точки для движения. При атаке и защите требуется чаще менять позицию и двигаться непредсказуемо, поэтому точка выбирается вблизи агента и чаще обновляется, а при поиске противника или отступлении выбор точки осуществляется по всей арене. Остальные переменные определяют веса различных свойств позиций на арене. Например, при атаке больший вес имеют точки с прямой видимостью противника, при защите – точки за укрытиями, а при отступлении – точки за пределами обзора противника.

Слайд

Общий вид правила вывода представлен на слайде. Для удобства правила представлены в табличном виде. Для каждой комбинации термов входных переменных даются выходные значения. Для определенных ранее входных переменных требуется 18 правил вывода, однако за счет объединения правил, в которых не используется одна из входных переменных, общее количество удалось уменьшить до 12.

Представленные на слайде правила были определены вручную на основе представлений о желаемом поведении агента, а затем были скорректированы в ходе тестирования против живого игрока и других агентов.

Слайд

Нечеткая логика позволяет описывать поведение агента с помощью естественного языка. Однако для реализации агента нужно определить большое количество правил вывода. Для рассмотренного случая потребовалось 18 правил. Если добавить еще одну переменную, то потребуется уже 54 правила. Если еще одну – 162 правила. Это значит, что нечеткая логика применима только к системам с простой моделью среды, имеющих небольшое количество переменных.

Слайд

Следующий метод – нейронные сети. В качестве архитектуры был выбран многослойный перцептрон. Входные переменные аналогичны с рассмотренными для нечеткой логики. Выходные переменные – это веса возможных стратегий поведения агента. При принятии решения выбирается стратегия с наибольшим весом.

Слайд

Для обучения сети использовалось множество из 200 записей. Из них 100 непосредственно для обучения и 100 для проверки качества обучения. Под качеством обучения имеется в виду доля проверочных записей, для которых ожидаемый результат совпал с фактическим.

На качество обучения могут влиять различные параметры, перечисленные на слайде. Можно представить процесс обучения в виде функции, аргументы которой – это параметры обучения, а результат – это качество обучения. Экспериментально было установлено, что эта функция гладкая, монотонно возрастает при увеличении количества повторов обучения и для разных сечений по количеству повторов имеет один максимум. Это значит, что можно применить численные методы и найти оптимальные параметры обучения.

Слайд

Расчеты показали, что для рассматриваемой задачи лучше подходит двухслойный перцептрон, так как он имеет более высокое качество обучения при меньшем количестве повторов обучения.

Слайд

Нейронные сети позволяют аппроксимировать неизвестную функцию принятия решений по нескольким известным решениям. Нейронные сети сложно отлаживать, так как они не объясняют, почему было принято то или иное решение. Полученное поведение агента сложно корректировать, так как для этого требуется переобучение. Также сложно реализовать реакцию на редкое событие, так как в обучающем множестве для него будет мало записей.

Нейронные сети позволяют получить реакции агента на ситуации, которые не были учтены или были не известны при разработке, однако такие реакции могут быть не адекватными. Например, в ходе исследования возникали ситуации, когда агент предпочитал атаковать цель даже если ее не было видно.

Слайд

Игра и рассмотренные агенты были реализованы с помощью игрового движка Unity на языке C#. Игрок и агенты управляют персонажами через одинаковый интерфейс. Агенты используют компоненты Shooting и Navigation для работы с этим интерфейсом и компонент TerrainReasoning для оценки позиций на арене и выбора наиболее выгодной. Контроллеры реализуют правила, рассмотренные ранее.

Слайд

Для сравнения методов принятия решений проводились игры реализованных агентов друг против друга. Необходимые данные собирались в базу данных SQLite, а затем обрабатывались с помощью SQL и MATLAB.

Критерии сравнения можно разделить на основные и дополнительные. Основные критерии – это способность методов решать поставленную задачу лучше других и быстродействие. В данном случае критерием качества решения задач является количество побед и поражений.

Дополнительные метрики позволяют оценить качество принимаемых решений с точки зрения предметной области. Количество нанесенного и полученного урона дают информацию о стратегиях связанных с атакой и защитой, а доля игр, в которых была взята аптечка – о стратегиях, связанных с отступлением.

Слайд

Из полученных графиков видно, что деревья поведения имеют наибольшее количество побед и наименьшее количество поражений. Нечеткая логика имеет быстродействие ниже на два порядка так как в нем используется численное интегрирование. Другие методы имеют одинаковый порядок быстродействия.

Слайд

Деревья поведения так же имеют наибольшее значение среднего нанесенного урона и наибольшую долю игр, в которых была взята аптечка.

Их полученных данных можно сделать вывод, что для рассматриваемой задачи наиболее эффективным методом принятия решений являются деревья поведения.

Слайд

Результаты работы представлены на слайде. На этом у меня все, спасибо за внимание.