МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

**ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ**

**Кафедра технологий программирования**

**Искусственный интеллект в компьютерных играх**

Курсовая работа

|  |  |
| --- | --- |
| Котусева Дмитрия Дмитриевича  студента 3 курса,  специальность  «Прикладная  информатика»  Научный руководитель:  кандидат физико-математических наук,  доцент В. В. Горячкин |  |

Минск - 2019

**РЕФЕРАТ**

**!!!Курсовой проект**, 21 с., 8 рис., 10 источников, 1 приложение.

**Ключевые слова:** АГЕНТ, ИИ.

**Объект исследования** — организация работы искусственного интеллекта в сфере компьютерных игр.

**Предмет исследования** — характеристические особенности методов построения эффективной модели игрового искусственного интеллекта.

**Цель работы** — исследовать различные схемы работы искусственного интеллекта в играх, адаптировать один из методов при реализации собственного приложения, написать работающий прототип.

**Задачи**:

1. аналитический обзор типов компьютерных игр, где применим искусственный интеллект, и способов организации работы искусственного интеллекта;
2. анализ и разбор математических алгоритмов, используемых при разработке искусственного интеллекта;
3. построение собственной модели искусственного интеллекта;
4. разработка работающего прототипа.

**Методы исследования** — изучение теоретических материалов, построение логических схем и моделей, проверка теоретических данных в игровом движке.

**!!!Полученные результаты:**

1. подготовлен общий аналитический обзор сфер и способов применения инерциальных датчиков;
2. построена общая математическая модель трёхосевого МЭМС-акселерометра;
3. программная реализация полученной математической модели.

**Область применения** — компьютерные игры.

**РЕФЕРАТ**

**!!!Курсовой проект**, 21 с., 8 рис., 10 источников, 1 приложение.

**Ключевые слова:** АГЕНТ, ИИ, ИГРОВОЙ ДВИЖОК.

**Объект исследования** — организация работы искусственного интеллекта в сфере компьютерных игр.

**Предмет исследования** — характеристические особенности методов построения эффективной модели игрового искусственного интеллекта.

**Цель работы** — исследовать различные схемы работы искусственного интеллекта в играх, адаптировать один из методов при реализации собственного приложения, написать работающий прототип.

**Задачи**:

1. аналитический обзор типов компьютерных игр, где применим искусственный интеллект, и способов организации работы искусственного интеллекта;
2. анализ и разбор математических алгоритмов, используемых при разработке искусственного интеллекта;
3. построение собственной модели искусственного интеллекта;
4. разработка работающего прототипа.

**Методы исследования** — изучение теоретических материалов, построение логических схем и моделей, проверка теоретических данных в игровом движке.

**!!!Полученные результаты:**

1. подготовлен общий аналитический обзор сфер и способов применения инерциальных датчиков;
2. построена общая математическая модель трёхосевого МЭМС-акселерометра;
3. программная реализация полученной математической модели.

**Область применения** — компьютерные игры.

**РЕФЕРАТ**

**!!!Курсовой проект**, 21 с., 8 рис., 10 источников, 1 приложение.

**Ключевые слова:** АГЕНТ, ИИ.

**Объект исследования** — организация работы искусственного интеллекта в сфере компьютерных игр.

**Предмет исследования** — характеристические особенности методов построения эффективной модели игрового искусственного интеллекта.

**Цель работы** — исследовать различные схемы работы искусственного интеллекта в играх, адаптировать один из методов при реализации собственного приложения, написать работающий прототип.

**Задачи**:

1. аналитический обзор типов компьютерных игр, где применим искусственный интеллект, и способов организации работы искусственного интеллекта;
2. анализ и разбор математических алгоритмов, используемых при разработке искусственного интеллекта;
3. построение собственной модели искусственного интеллекта;
4. разработка работающего прототипа.

**Методы исследования** — изучение теоретических материалов, построение логических схем и моделей, проверка теоретических данных в игровом движке.

**!!!Полученные результаты:**

1. подготовлен общий аналитический обзор сфер и способов применения инерциальных датчиков;
2. построена общая математическая модель трёхосевого МЭМС-акселерометра;
3. программная реализация полученной математической модели.

**Область применения** — компьютерные игры.

Тут должно быть содержание

**Перечень условных обозначений**

ИИ искусственный интеллект

# Введение

В эпоху стремительно развивающихся информационных технологий достижения прогресса проникают во всё большее количество сфер жизнедеятельности человека. Не исключением стала и индустрия развлечений.

Мировая индустрия развлечений аккумулирует миллиарды долларов и является значимой частью экономики большинства стран. Она включает большое количество различных направлений, одним из которых является направление компьютерных игр, получившее развитие именно благодаря укреплению в нашей жизни компьютерных технологий.

Компьютерная игра представляет собой не что иное как обычное приложение, предназначенное для развёртывания на целевой аппаратной платформе. Список этих платформ может варьироваться от обычных в классическом понимании компьютеров до встроенных в автомобили вычислительных устройств и «умных» наручных часов.

Существует огромное количество самых различных жанров компьютерных игр на любой вкус и цвет: шутеры, симуляторы, стратегии, приключения, музыкальные игры, ролевые игры и др. Стоит отметить, что чётких границ в классификации игр на данный момент не существует и зачастую чётко определить принадлежность игры тому или другому жанру крайне затруднительно.

Следует понимать, что почти никто из разработчиков игр не начинает разработку своего творения с абсолютного нуля. Все они используют ту или иную стартовую платформу ― игровой движок. Игровой движок выполняет базовые функции, необходимые в любой игре: трёхмерный рендеринг, обработка сигнала устройств ввода-вывода, обработка звуков, отрисовка элементов индикации и т. д. Также он предоставляет разработчикам необходимые программные интерфейсы для взаимодействия со встроенными возможностями движка и организации собственной логики приложения. Таким образом, конечный продукт является результатом работы как создателей движка, так и разработчиков самой игры.

Как было упомянуто выше, игровые движки берут на себя реализацию базового функционала, без которого никак не обойтись при разработке любой игры. Несмотря на это, на пути создания игры перед разработчиками встаёт немалое количество задач, которые им приходиться решать самим. Одной из таких задач является разработка искусственного интеллекта.

Стоит отметить, что в рамках игровой индустрии под искусственным интеллектом понимается не аналог интеллекта естественного (интеллекта человека), а его симуляция. Основных причин такого подхода можно выделить три:

1. Дороговизна разработки полноценного искусственного интеллекта.
2. Нехватка вычислительных мощностей для обеспечения должного уровня игрового опыта для игрока.
3. Разработка полноценного искусственного интеллекта зачастую является избыточной в рамках поставленной задачи и обойтись можно куда меньшими силами.

То есть при разработке игрового искусственного интеллекта перед разработчиками стоит задача наиболее правдоподобной имитации поведения человека с использованием минимального количества вычислительных ресурсов. Реализация искусственного интеллекта сильно влияет на геймплей, системные требования и бюджет игры, и разработчики балансируют между этими требованиями, стараясь сделать интересный и нетребовательный к ресурсам искусственный интеллект малой ценой. Поэтому подход к игровому искусственному интеллекту серьёзно отличается от подхода к традиционному искусственному интеллекту — 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 [источник 2 (страница википедии про ии)]. В дальнейшем при упоминании искусственного интеллекта будет пониматься именно имитация интеллекта реального, а не его аналог.

Целью данной работы является изучение и анализ существующих на данный момент методов организации работы искусственного интеллекта, типичных проблем, с которыми сталкиваются разработчики в процессе создания, а также реализация одного или нескольких из них в виде работающего прототипа.

Актуальность данной работы подкрепляется постоянно растущим спросом игроков на всё более интересные и сложные игры, создание которых требует разработки всё более сложных моделей искусственного интеллекта, а также растущей популярностью компьютерных игр, как способа досуга во всё более широких слоях населения.

# Игровой искусственный интеллект

## Обзор и классификация технологий и методов искусственного интеллекта

В любой игре, где необходимо наличие искусственного интеллекта (далее ИИ), перед этим ИИ ставится задача по управлению «интеллектуальными агентами», где агент является игровым персонажем, транспортным средством, ботом, а иногда и чем-то более абстрактным: целой группой сущностей или даже цивилизацией. Во всех случаях перед агентом стоят следующие задачи:

* Sense: получить информацию о внешней среде.
* Think: на основе полученных данных принять решение о дальнейших действиях.
* Act: собственно выполнение принятого на предыдущем шаге решения.

По завершении выше описанного цикла агент попадёт в новую ситуацию, поэтому цикл придётся проделать заново.

Существует достаточно большое количество методов (алгоритмов) организации работы ИИ.

Одним из простейших алгоритмов является дерево решений. Дерево решений состоит из узлов двух типов:

* Узлы принятия решений: выбор между двумя альтернативами на основе проверки некоторого условия, где каждая альтернатива представлена в виде отдельного узла.
* Конечные узлы: действие для выполнения, представляющее окончательное решение.

В качестве примера можно провести знаменитую игру Pong, в которой перед игроком и ИИ ставится одинаковая задача — отбить мяч с помощью платформы в сторону оппонента (рисунок 1.1).

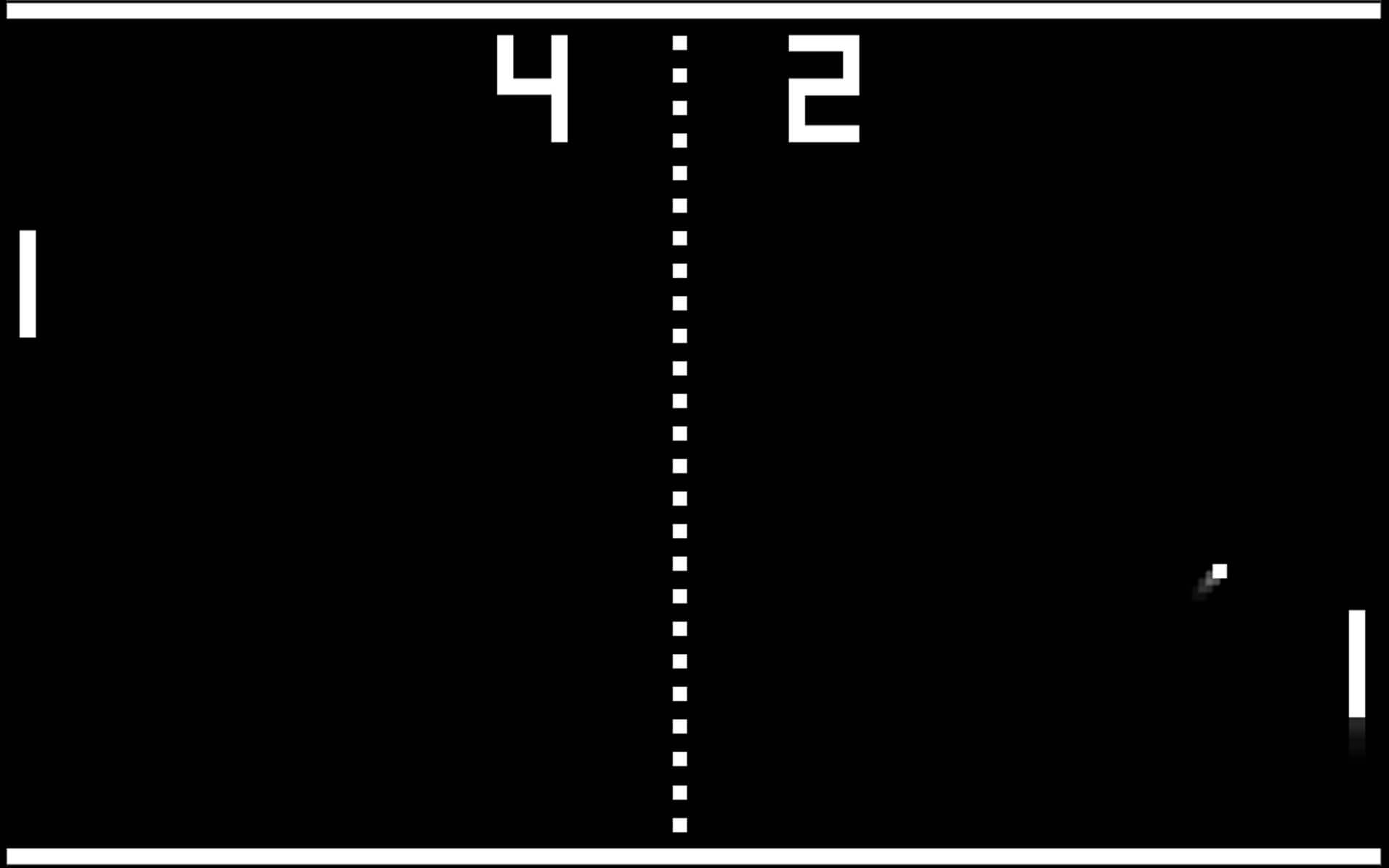


Рисунок 1.1 Интерфейс игры Pong

Более точно перед ИИ стоит задача ― решить, в каком направлении переместить платформу, чтобы отбить мяч. Блок-схема дерева решений такого ИИ может выглядеть следующим образом (рисунок 1.2):

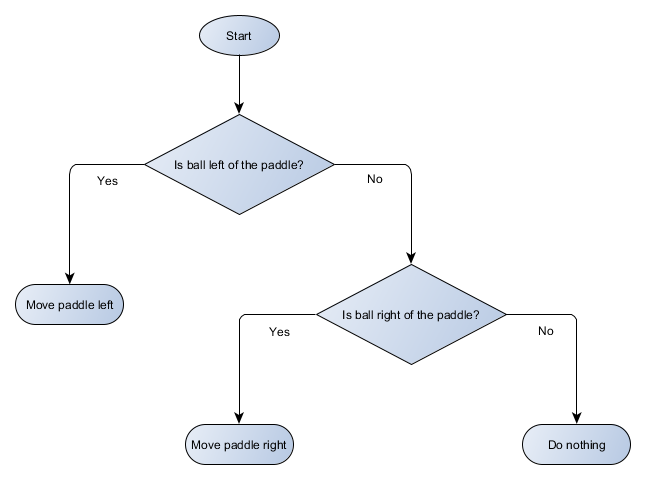


Рисунок 1.2 Дерево решений для ИИ игры Pong

В приведённой выше блоке-схеме нетрудно видеть, что получение данных о внешней среде (игра знает где мяч и где платформа, поэтому ИИ обращается к ней за этой информацией) и принятие решений происходит в узлах принятия решений, выполнение же решений происходит в конечных узлах.

Дерево решений неплохо подходит для реализаций простых ИИ, способных выполнять небольшой спектр действий. Но иногда существует слишком много условий, чтобы эффективно представить их в дереве решений или скрипте. Иногда нужно заранее оценивать, как изменится ситуация, прежде чем принимать решение о следующем шаге. Для решения этих проблем нужны более сложные подходы [ссылка на источник 3].

Ещё одним интересным подходом к организации ИИ является конечный автомат (finite state machine или FSM). Конечный автомат предполагает, что наш агент всегда находится в одном из конечного множества состояний. В качестве примера разберём агента (воина) со следующим конечным автоматом (рисунок 1.3):

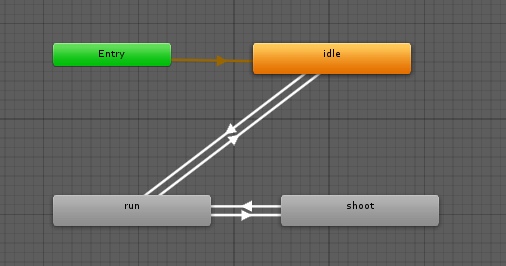


Рисунок 1.3 Пример конечного автомата



Рисунок 1.4 Игровой агент

На рисунке 1.3 нетрудно различить 3 состояния, в которых может находиться агент:

* Состояние покоя (именно в нём он находится на рисунке 1.4).
* Состояние бега за игроком.
* Состояние стрельбы в игрока.

И снова мы можем посмотреть на эту систему через призму цикла Sense/Think/Act. Sense воплощается в данных, используемых логикой перехода. Think — переходами, доступными в каждом состоянии. А Act осуществляется действиями, совершаемыми периодически в пределах состояния или на переходах между состояниями [ссылка на источник 3]. Основной вопрос при реализации такой логики ИИ вызывает механизм проверки условий переходов между состояниями. Зачастую логику проверок условий переходов выносят за пределы автомата. Автомат, в таком случае, получает данные условий переходов через обработчики событий, генерируемых внешним кодом. Таким образом, этап Sense происходит вне автомата во внешнем коде.

Конечно же, описанный выше пример является крайне простым. В более сложных системах количество состояний и связей между ними может значительно возрасти. В этом случае может возникнуть большое количество одинаковых переходов к схожим по логике состояниям. Например, при организации разных типов состояний атаки (атака ближнего боя и стрельба). Автомат такого типа крайне тяжело расширять и отлаживать из-за неоправданно большого количества схожих связей. Решить данную проблему поможет преобразование автомата из обычного одноуровневого в иерархический конечный автомат (hierarchical finite state machine или HFSM), где каждое состояние может быть оболочкой для группы схожих по логике состояний.

В качестве примера модернизируем автомат нашего воина (рисунки 1.5 и 1.6):

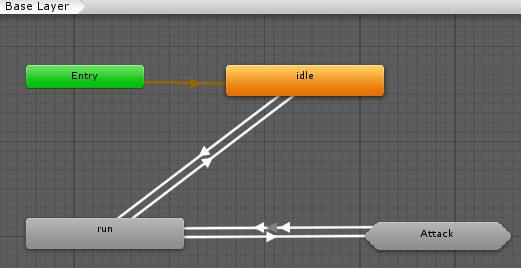


Рисунок 1.5 Верхний уровень иерархического конечного автомата

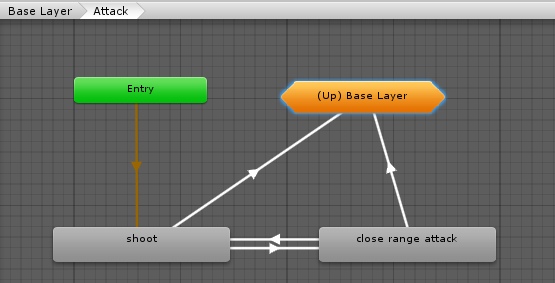


Рисунок 1.6 Внутренняя логика состояния атаки

В этом примере мы добавили новый тип атаки (атака ближнего боя) и объединили его вместе с состоянием стрельбы в общее состояние атаки, которое и осталось на верхнем уровне иерархии.

Таким нехитрым способом возможно минимизировать количество дублированных переходов в конечных автоматах, что значительно улучшает общую расширяемость автомата.

Ещё одним достаточно популярным методом реализации логики ИИ является дерево поведений (behavior tree). Данный подход стал популярным после успешного его применения в игре Halo.

В случае дерева поведений логика агента определяется специальным графом, который представляет собой дерево (отсюда первая половина названия метода). Структура графа начинается с единственного корневого узла. В дереве все узлы можно разделить на три типа (рисунок 1.7):

* Composite: узлы, имеющие два и более дочерних узлов.
* Decorator: узлы, имеющие лишь один дочерний узел.
* Leaf: узлы на концах дерева, не имеющие дочерних узлов и обозначающие конкретное действие.

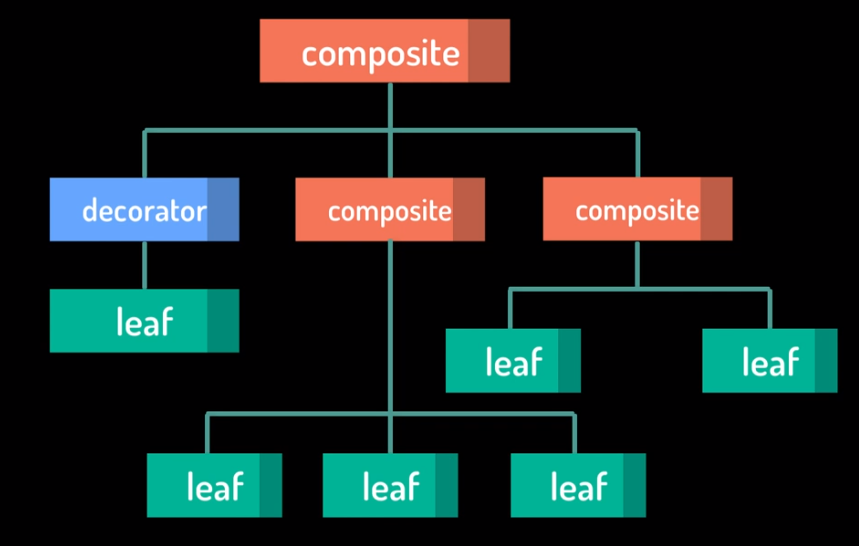


Рисунок 1.7 Структура дерева поведений

В свою очередь узлы типа composite делятся на два подтипа:

* Selector: сигнализирует о том, что только один дочерний узел будет посещён (рисунок 1.8).
* Sequence: обеспечивает, что каждый дочерний узел посещается в порядке следования (рисунок 1.9).

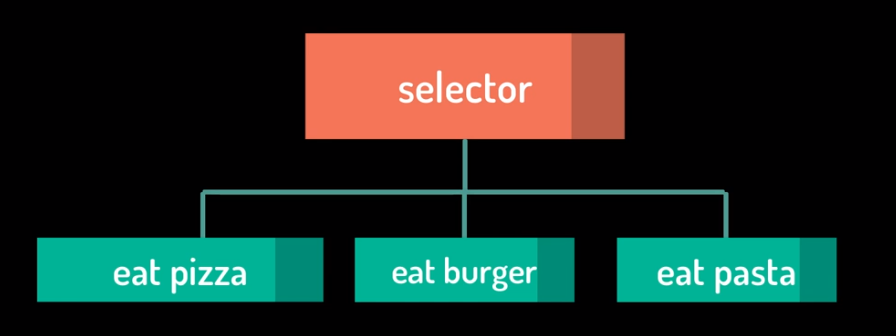


Рисунок 1.8 Логика selector’а такова, что он выберет одну из опций в целях утоления голода

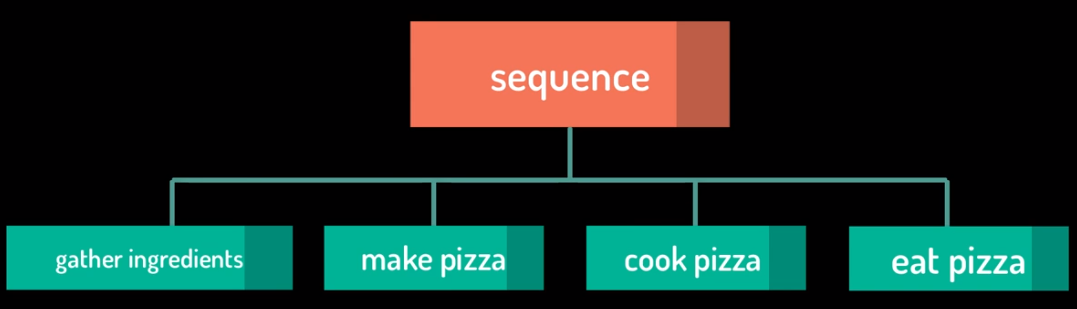


Рисунок 1.9 Представлена последовательность действий, которую должен выполнить агент в целях достижения поставленной цели (утоление голода)

Каждый узел возвращает одно из трёх значений:

* Succeded: действие узла завершилось успешно.
* Failed: действие узла завершилось неудачно.
* Running: узел в данный момент в процессе выполнения.

Это лишь общая схема работа дерева поведений. Деревья поведений сложны — есть много способов их составить, да и найти правильное сочетание декораторов и составных узлов может быть проблематично [источник 3]. По этой причине существует большое количество всевозможных реализаций данного метода.

Последним по порядку, но не по значению является алгоритм под общим названием utility-based system (основанная на полезности система). Этот метод предоставляет несколько другой подход к организации выбора решений. Вместо того, чтобы иметь чёткий набор выборов или дерево возможных действий, алгоритм предлагает изучить все действия и выбрать самое подходящее в данный момент на основе какой-либо численной характеристики полезности данного действия, где полезность — произвольная мера того, насколько важно или желательно выполнение этого действия для агента. В общем случае назначается диапазон возможных значений полезности с нижним и верхним пределами. На значение полезностей разных решений могут влиять различные значения внешней среды в данный момент времени. Например, если агент долго не ел и счётчик голода достиг критической отметки, полезность действия «найти пищу» будет стремиться к максимальной из всех возможных. Однако в случае нахождения рядом врагов действие «найти безопасное место» также может заиметь очень высокую меру полезности. Что выберет агент, зависит от реализованной программистом логики, но очевидно, что оба действия в данном примере будут конкурирующими друг с другом и выбор любого из них с точки зрения человека покажется логичным.

Описанные выше подходы не являются единственными и зачастую могут применяться с некоторыми модификациями. Однако при моделировании и реализации нашего собственного ИИ отталкиваться мы будем именно от них.

**Проектирование игрового приложения с искусственным интеллектом**

Обзор современных компьютерных игр и их классификация