**Слайд 1**. Здравствуйте, меня зовут Аухадиев Александр, группа 9381, я представляю выпускную квалификационную работу на тему «разработка программы планирования действий в стратегической игре».

**Слайд 2.** Игры жанра стратегии в реальном времени очень популярны среди игроков по всему миру, но ИИ в этих играх очень предсказуем и легко обыгрывается даже игроками-любителями.

Большинство алгоритмов, которые работают в традиционных играх, таких как шахматы, не могут быть применены к играм жанра RTS или просто неэффективны в вычислительном отношении. Данная область исследований интересна и с практической точки зрения, потому что в ней существуют подзадачи, которые связаны с задачами в реальной жизни. Управление производством в играх RTS напоминает проблемы промышленного производства в реальном мире. Алгоритмы, изобретенные для управления движением юнитов в играх RTS, позже могут быть распространены на управление дорожным движением в реальных городах.

Игра этого жанра StarCraft предоставляет удобный интерфейс программного взаимодействия для создания собственного внутриигрового ИИ. Она стала испытательным полигоном для разных методов его создания, а между разными его реализациями проходят ежегодные соревнования.

Поэтому в рамках ВКР я решил создать ИИ для игры StarCraft, решающий задачу планирования действий.

**Слайд 3.** Планирование действий заключается в создании и выполнении плана – последовательности действий, позволяющей достичь поставленной цели, то есть перейти из начального состояния в целевое.

Разрабатываемая программа должна на основе текущей игровой ситуации выбирать актуальную цель и из доступного множества действий выполнять план, обладающий двумя свойствами:

1. Корректностью, то есть при успешной генерации плана выполнение его шагов всегда приводит к целевому состоянию

2. Полнотой: алгоритм должен находить план, если тот существует.

**Слайд 4**. Для выбора метода решения были изучены самые популярные методы создания игрового ИИ и разные реализации для стратегий в реальном времени.

В результате обзора методов было установлено, что стандартные методы планирования не применимы к играм данного жанра, так как те обладают слишком большим пространством состояний, а количество возможных действий по ходу игры непрерывно увеличивается, из-за чего невозможно вовремя формировать актуальный план.

Сравнение аналогов показало такие недостатки как отсутствие полноценной реализации, способной провести полную игровую сессию, зависимость от экспертных знаний и отсутствие логики принятия решений на стратегическом уровне, то есть на уровне долгосрочных игровых целей. Также можно отметить, что все аналоги способны обучаться на игровых записях.

**Слайд 5.** Из всех рассмотренных методов наиболее подходящим стали деревья поведения, так как они решают все описанные недостатки и к тому же:

1. Обладают гибкой наглядной структурой

2. Легко масштабируемы и модифицируемы

3. Поддерживают машинное обучение

4. Поддерживают принятие решений как тактическом, так и на стратегическом уровне

Недостатки:

1. Реактивное поведение без анализа последствий

2. Однообразное поведение при выполнении определённых условий без учёта дополнительных факторов или непредвиденных событий

Вывод: при решении задачи планирования действий деревья поведения можно использовать для объединения действий в готовые последовательности, выполняющие некоторые подцели. Затем из этих последовательностей можно будет сформировать план. Таким образом, деревья поведения позволят сократить пространство поиска при генерации плана.

**Слайд 6.** На основе деревьев поведения была разработана математическая модель, описывающая параметризируемое состояние среды и множества действий и условий, с помощью которых можно воздействовать на текущее состояние.

Внутренние узлы дерева поведения выполняют запуск логики дочерних узлов и, в зависимости от своего типа, определяют успешность/неуспешность своего выполнения. Так, например, селекторы будут выполнять запуск дочерних узлов, пока хотя бы один из них не будет выполнен успешно, или все они не завершат выполнение неудачей, в то время как узел последовательности будет запускать поведение дочерних узлов, пока либо все они не будут выполнены успешно, либо пока хотя бы один из них не завершит выполнение неудачей.

Листовые узлы заключают в себе условия и действия.

Таким образом, выполняя обход дерева поведения в глубину и запуская логику его узлов, можно менять текущее состояние.

**Слайд 7.** Разработанная архитектура подразумевает создание деревьев поведения из базы данных или напрямую из программного кода и их запуск. В процессе работы узлы деревьев поведения могут сохранять и считывать необходимую информацию с так называемой доски объявлений (BlackBoard).

Взаимодействие с API игры происходит через специальные обработчики, позволяющие отделить логику деревьев поведения от реализации конкретных игровых действий.

Также это позволит при наличии у других стратегических игр подобного API использовать логику деревьев поведения для реализации в них ИИ.

**Слайд 8.** Для создания программы были реализованы классы для деревьев поведения и их узлов разных типов, класс BlackBoard для сохранения информации и статические классы для взаимодействия с игровой средой. В процессе разработки использовались шаблоны проектирования одиночка, чтобы для каждого дерева поведения создавался только один экземпляр, и чтобы к доске объявлений можно было обратиться из любой части программы.

При реализации классов, описывающих узлы, использовался шаблон компоновщик, позволяющий одинаково обращаться к любому потомку класса Node, как к отдельным узлам, так и к их комбинациям.

Статические классы соответствуют шаблону фасад, так как упрощают интерфейсы BW API, образуя унифицированную подсистему и предотвращают появление сильных связей между классами, отвечающими за игровую логику, и классами из BW API.

**Слайд 9.** Основной алгоритм программы подразумевает создание игрового клиента из запущенной игровой сессии и получение объекта класса Game, через который происходит всё взаимодействие с игрой.

Затем происходит запуск деревьев поведения, которые с помощью своих узлов через статические утилиты взаимодействуют с игрой, получая информацию об успешности/неудаче выполнения и сохраняя необходимую информацию на доску BlackBoard.

По ходу игры возникают некоторые игровые асинхронные события, отмеченные на диаграмме как onEvent, которые обрабатываются в главном классе программы – классе Bot, который может на основе сгенерированного события сохранить некоторую информацию в BlackBoard, чтобы узлы поведения смогли ей воспользоваться.

Если после завершения выполнения дерева поведения игра ещё не окончена, дерево запускается снова.

**Слайд 10.** Для проверки работоспособности программы были написаны сценарии ожидаемого поведения в разных игровых ситуациях, которым выполняемая логика программы должна удовлетворять.

Проверялись такие типичные для всех стратегий в реальном времени ситуации, как развитие базы и наём юнитов, защита, разведка и атака соперника. Каждый из этих сценариев имеет одну или несколько глобальных игровых целей, для достижения которых и должен генерироваться и выполняться план.

В ходе проведения эксперимента не было обнаружено появление программных ошибок или неожиданное поведение игровых сущностей, никак не обусловленное реализованной логикой. В некоторых сценариях поведения были выявлены сильные отличия наблюдаемого поведения от ожидаемого. Все обнаруженные несоответствия решаются путём усложнения реализованной логики, то есть добавления новых узлов деревьев поведения, что подтверждает тезис о простоте их модернизации.

**Слайд 11.** Таким образом, был реализован искусственный интеллект на основе деревьев поведения для компьютерной стратегической игры в реальном времени StarCraft, решающий задачу планирования действий.

Разработанную программу можно использовать для изучения возможностей исследуемого метода, участия в соревнованиях между другими реализациями и при создании ИИ для новых игр данного жанра.

В качестве направлений дальнейшей разработки возможна реализация графического фреймворка для создания и наглядного отображения деревьев поведения, а также добавление в программу планировщика, способного генерировать план с учётом последствий каждого действия и допускающего перепланирование при возникновении непредвиденных ситуаций.

**Слайд 12.** На этом всё, спасибо за внимание, я готов ответить на ваши вопросы.

Реализованы пять деревьев поведения:

1. GlobalBT. Основное дерево, выбирающее текущую цель и запускающее подходящее для её достижения дерево поведения

2. DefaultBT. Дерево поведения, выполняющее стандартный набор действий на каждой итерации GlobalBT

3. ScoutBT. Дерево поведения для разведки вражеских позиций и сохранения информации о зданиях противника

4. ProtectBaseBT. Дерево поведения для защиты базы при атаке противника, запускающее у боевых юнитов поведение атаки на нападающих, у мирных – поведение бегства от опасности

5. AttackEnemyBaseBT. Дерево поведения для выбора набора юнитов и их атаки на противника.