УДК 681.3.068 (575.2) (04)

КОМПЬЮТЕРНЫЙ ИГРОВОЙ КОМПЛЕКС НА ОСНОВЕ ПРИНЦИПОВ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА ДЛЯ ИЗУЧЕНИЯ СЛОЖНЫХ ЛОГИЧЕСКИХ СИСТЕМ

Г.А. Десятков – докт. физ.-мат. наук, профессор,
У.К. Дегенбаев, А.А. Мохов – аспиранты,
А.В. Алексеев, А.А. Рафиев – инженеры

The paper presents game software "Kingdom Ruler" developed in KRSU. The software allows several programs to compete each other in a virtual world and view graphically a course and result of competitions. To compete successfully in such complex logical system the programs should be written using a variety of artificial intelligence techniques.

Современные средства изучения и управления сложными логическими системами основаны на применении принципов искусственного интеллекта [1]. Они позволяют принимать решения в меняющихся условиях, заранее предусмотреть которые программно невозможно из-за сложности их формализации. В связи с этим актуальным является разработка таких компьютерных средств, которые позволили бы наглядно ознакомиться с принципами создания подобных управляющих систем и приобрести навыки программирования элементов искусственного интеллекта.

В данной работе представлен программный комплекс "Kingdom Ruler", разработанный в Кыргызско-Российском Славянском университете. Комплекс позволяет организовать соревнования нескольких программ, созданных участниками, в виртуальном мире ("государстве", включающем в себя замки, рабочих и воинов) и наглядно, в графическом виде, показывать ход и результаты соревнований. Для того, чтобы программа могла успешно участвовать в создаваемой сложной логической системе, в ней должны быть использованы средства искусственного интеллекта.

Программная система "Kingdom Ruler" создана на основе Java Challenge 2004 Code-Ruler [2], с которой авторы познакомились на чемпионате мира по программированию ACM ICPC World Finals 2004 [3], и принимали участие в соревновании.

Отличительной особенностью разработанного варианта является возможность написания соревнующихся программ не только на языке Java, но и на любом другом языке программирования, способного создавать динамически подключаемые библиотеки (DLL).

Программный комплекс предназначен, в первую очередь, для студентов, специализирующихся в области программирования, а также для всех интересующихся вопросами искусственного интеллекта и использования его принципов для управления сложными логическими системами.

Состав программного комплекса. Программный комплекс состоит из трех взаимосвязанных модулей, реализующих соответствующие алгоритмы:

Arena – модуль, осуществляющий алгоритм просчета всех ходов матча и позволяющий задать состав участников и их расположение на игровом поле. Этот модуль отвечает

за соблюдение игровыми программами правил игры, следит за тем, чтобы программы не превысили предоставленный им лимит времени и памяти (соответственно, 2 секунды на ход и 1 Мб оперативной памяти). В случае нарушения программой правил игры или превышения указанных ограничений, она дисквалифицируется, и в виртуальном мире ее замки и служащие остаются без управления.

Player Loader – модуль, определяющий взаимодействие программ, управляющих игроками и модулем Arena. В его функции входит загрузка программ (DLL-файлов) в память, их инициализация, передача им всей информации о состоянии игрового поля каждого хода и получение приказов для замков и служащих. Каждая программа запускается в отдельном процессе и поэтому ошибка одной из них не повлияет на выполнение других.

BattleViewer – модуль, наглядно отображающий ход соревнования, позволяющий просматривать его как видео-ролик с возможностью отката назад, ускоренного просмотра и т.д. По окончании просмотра выводится итоговая таблица результатов с подсчетом очков, заработанных игроками.

Ниже приведены подробные сведения об этих модулях.

Модуль Arena. Интерфейс этого модуля достаточно прост и понятен. Добавлять игрока в матч можно командой **Player->Add**. Удалять – командой **Player->Remove**. Команда **Player->Shuffle** позволяет переставить игроков на игровом поле случайным образом так, чтобы они располагались по возможности симметрично. Можно изменять расположение игроков вручную, перемещая их в пределах окна при помощи указателя мыши.

Для запуска соревнования используется команда **Battle->Fight**. При этом начинается просчет ходов соревнования, о чем можно судить по индикатору, показывающему количество уже просчитанных ходов.

Интерфейс модуля Arena изображен на рис. 1.

Модуль Player Loader. Этот модуль вызывается непосредственно из модуля Arena и не имеет пользовательского интерфейса. Схема взаимодействия модулей Arena и PlayerLoader показана на рис. 2.

Модуль BattleViewer. Этот модуль предназначен для просмотра соревнования, который просчитал модуль Arena. Он запускается автоматически при окончании расчета соревнования модулем Arena, но его можно запускать и самостоятельно, передав с командной строки имя файла, содержащего протокол соревнования.

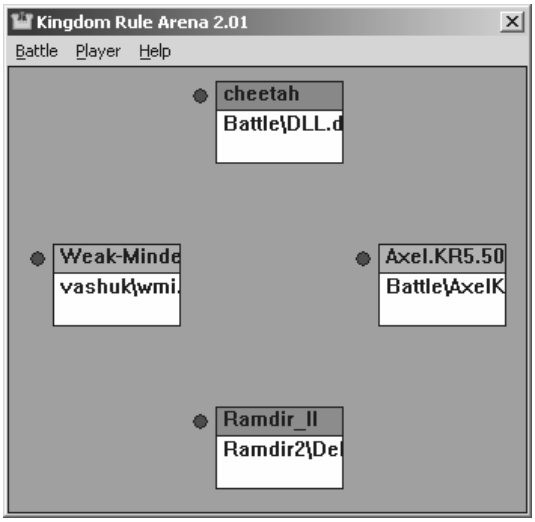

Рис.1. Интерфейс модуля Arena (выбрано четыре игрока).

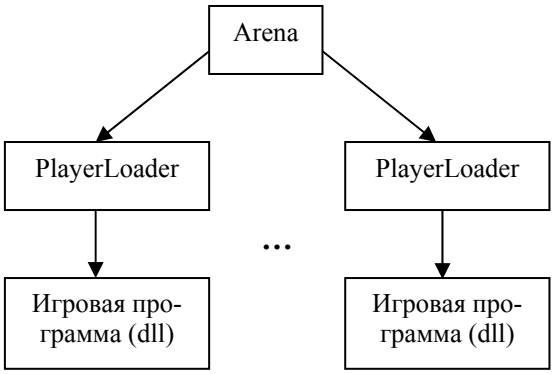

Рис.2. Схема взаимодействия Arena и PlayerLoader.

На экране отображается несколько окон. На главном окне пользователь может наблюдать детальный ход соревнования программ. Замки отображаются в виде клетки, отгоро-

женной стенами. Под центральной клеткой располагается указатель производства замка, по которому можно судить о том, что производиться в замке в данный момент и сколько времени осталось до завершения производства. На рис. 3 показана небольшая часть игрового поля. В центре находится замок, принадлежащий игроку светлого цвета, там же находится и рабочий. На данном участке видны также еще один рыцарь и рабочий. Рыцари изображены фигурками коней, а рабочие – кружочками. Разные клетки окрашены в разные цвета, что указывает на то, что они принадлежат разным игрокам.

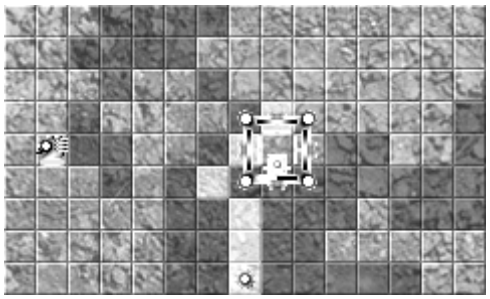

Рис. 3. Фрагмент игрового поля.

Так как на экране не помещается все игровое поле, то в правом верхнем углу экрана расположена миникарта, на которой можно видеть все игровое поле в уменьшенном виде. На ней удобно наблюдать за захватом территории игроками, так как она дает возможность представить общую картину по сравнению с основным игровым окном, на котором удобно наблюдать детали соревнования – сражения групп рыцарей, погоню их за рабочими и т.д. (рис. 4).

На миникарте видно, что игрок темного цвета потерял свой замок и у него мало земли. Скорее всего, он уже проиграл эту партию. Белый квадрат на миникарте указывает на ту область карты, которую видит в данный момент пользователь в основном окне. Этот квадрат можно перемещать указателем мыши, чтобы увидеть в главном окне ту или иную часть карты.

Под миникартой располагается окно управления просмотром битвы и текущая статистика (рис. 5). Видно, что прошло уже 210 ходов из 500. При помощи кнопок управления можно приостанавливать просмотр, возобновлять, перемещаться в разные участки битвы. Управление просмотром происходит как при обычном просмотре видео-ролика. Под панелью управления расположено окно с текущей статистикой. В нем указывается, сколько на данный момент у каждого игрока территории, замков, рыцарей, рабочих и сколько он набрал очков.

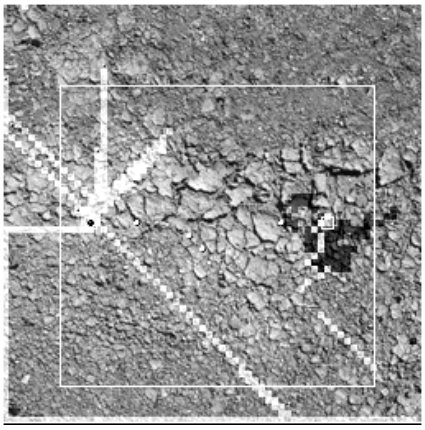

Рис. 4. Миникарта.

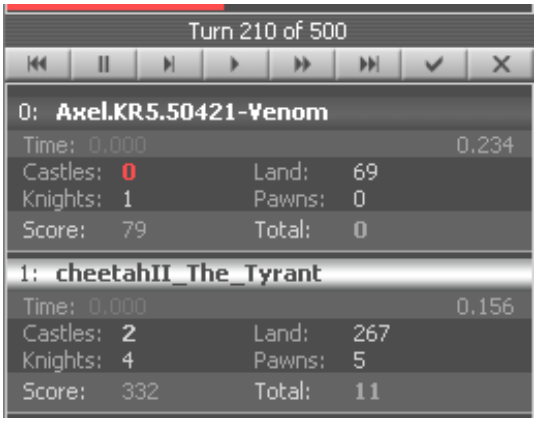

Рис. 5. Окно управления для просмотра соревнования и текущая статистика.

При завершении просмотра соревнования программа выводит сводную таблицу результатов с набранными очками игроков и общей статистикой матча (рис. 6).

Из рис. 6 видно, что в данном матче победу одержал игрок **cheetah_II_The_Tyrant** светлого цвета, набрав 23531 очко. Игрок, занявший второе место, набрал 0 очков.

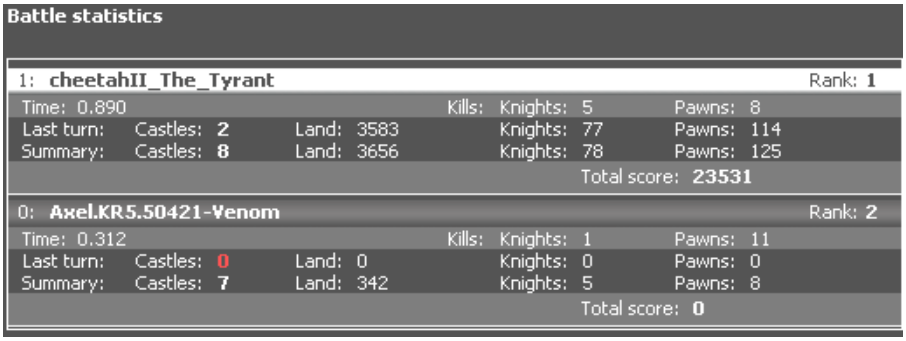

Рис. 6. Сводная таблица результатов.

Правила игры. Турнир проходит в виде серии матчей. В матче участвуют от 2 до 6 игроков, соревнующихся между собой. Каждый игрок начинает игру с одним замком, расположенным на конечной двумерной решетке (64×64), и восемью "служащими" (четырьмя рабочими и четырьмя рыцарями). На каждой клетке решетки может находиться только один служащий. Задачей игрока является отдача распоряжений служащим и замкам.

В течение матча игрок зарабатывает очки. Очки начисляются за территорию, количество замков и служащих, число поверженных врагов. Очки накапливаются с каждым ходом с весами, пропорциональными важности хода. Наиболее важным считается последний ход, наименее важным – первый. По результатам серии матчей определяется программа – победитель турнира.

Замки производят служащих, скорость производства зависит от размера захваченной территории. Территорию могут захватывать рабочие. Рыцари же могут захватывать замки других игроков, убивать рабочих и сражаться с другими рыцарями.

Игра состоит из некоторого заранее известного числа ходов (по умолчанию 500). В каждом ходе служащие могут произвести только одно действие:

- рабочие и рыцари могут переместиться на одно из восьми соседних полей;
- рабочий может захватить клетку, на которой он стоит;
- рыцарь может захватить замок, если он находится внутри замка;
- рыцарь может атаковать служащего, находящегося в одном из восьми соседних полей.

Каждый рыцарь имеет параметр, определяющий степень его здоровья. Далее этот параметр будет называться HP (hit points). Первоначально у рыцаря HP равен 100. У рабочего он равен единице. Каждая атака причиняет урон от 75% до 125% (определяется случайно) от силы удара, равного по умолчанию 30. Таким образом, рыцарь может выдержать до четырех ударов, оставшись при этом в живых; рабочий же погибает после единичной атаки рыцаря. Рыцари могут "лечиться" – восстанавливать свой HP до максимального значения. За ход рыцарь может повысить свой HP на 10 пунктов.

Для производства одного служащего замок должен накопить определенное количество ресурсов (по умолчанию 2000). Каждым ходом ресурсы всех замков игрока увеличиваются на размер территории этого игрока. Например, если игрок захватил 500 клеток поля и у него 2 замка, то каждый замок будет увеличивать свои ресурсы на 250 в ход, т.е. сможет производить по одному служащему за восемь ходов.

Создание игровой программы. Игровая программа действует в постоянно изменяющихся условиях в конкурентной борьбе с другими программами. Несмотря на то, что правила игры достаточно просты, предусмотреть все возможные игровые ситуации невозможно. Единственно приемлемое решение для создания эффективного алгоритма – применение искусственного интеллекта. (Отметим, что среди уже существующих игровых программ можно найти реализации различных методов теории искусственного интеллекта: экспертных систем, нечеткой логики, теории принятия решений, кластеризации и др.).

Каждая игровая программа должна быть скомпилирована в виде динамически подключаемой библиотеки, которая экспортирует определенные функции. Использование механизма DLL позволяет добавлять в систему новые игровые программы, без необходимости каждый раз компилировать и собирать систему. Кроме того, для создания игровой программы, это позволяет использовать языки программирования, отличные от языка программирования, на котором писалась система. Игровая программа должна содержать две функции, которые будут вызываться системой:

- Init;
- MakeTurn.

Функция Init вызывается системой перед началом соревнования, в параметрах функции передается информация о начальном состоянии игры и специальные буферы для дальнейшего взаимодействия с системой (см. таблицу).

Параметр	Описание
int **id**	Идентификатор данного игрока в системе, у всех объектов, принадлежащих данному игроку, поле lord будет равно этому числу
int **count**	Общее количество игроков
int **width**	Ширина игрового поля (64)
int **height**	Высота игрового поля (64)
Castle ***c**	Буфер обмена, содержащий информацию о замках
Servant ***s**	Буфер обмена, содержащий информацию о рабочих и рыцарях
char(***a**)[64]	Буфер обмена, содержащий информацию о территории игроков. Если клетка x,y принадлежит игроку I, то **a**[x][y] = I+1. Если она никому не принадлежит, то **a**[x][y] = 0
int **turn**	Номер хода
int **castles**	Количество замков
int **servants**	Количество служащих

Функция MakeTurn вызывается системой каждый ход, в данной функции игровая программа должна прочитать из буферов обмена информацию о новом состоянии игры и записать туда приказы для своих объектов. В параметрах функции передаются номер хода, количество замков, количество рабочих и рыцарей.

В качестве буферов обмена используются массивы специальных структур **Castle** и **Servant**:

```
struct Castle {
        int id;
        int x,y;
        int lord, wealth;
        ServantType training;
        ServantType order;
};
struct Servant {
        ServantType type;
        int id, lord;
        int x,y;
        int hp, damage, killed;
        OrderType order;
        int dir;
};
```

Виды служащих описываются перечислением **ServantType**: enum ServantType {stPawn, stKnight}.

Здесь **stPawn** обозначает рабочего, а **stKnight** – рыцаря.

Виды приказов описываются перечислением **OrderType**: enum OrderType {orAttack, orMove, orCultivate, orCapture, orNone, orNA}.

Виды приказов обозначают соответственно атаку, перемещение, захват территории, захват замка, приказ оставаться на месте и отсутствие приказа.

Приказы записываются в поле **order** соответствующего объекта. При этом для приказов перемещения и атаки необходимо задать направление при помощи поля **dir** (0 – север, 1 – северо-восток, 2 – восток, 3 – юго-восток, и т.д.).

x–1, y–1 **7**	x, y–1 **0**	x+1, y–1 **1**
x–1, y **6**	x, y	x, y–1 **2**
x–1, y+1 **5**	x, y+1 **4**	x, y+1 **3**

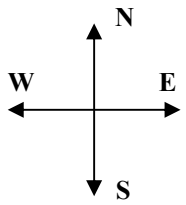

Рис. 7. Направления движения и атаки.

Выводы. Программный комплекс “Kingdom Ruler” позволяет в наглядном игровом виде изучать и совершенствовать знания в области искусственного интеллекта, направлен-

ного на управление сложными логическими системами. Комплекс уже прошел апробацию в КРСУ с участием студентов специальности “Программное обеспечение ВТ и АС”. В соревнованиях участвовали различные игровые программы, использующие разные алгоритмы управления и применяющие разные стратегии – от оборонительной до атакующей. Мы планируем проводить периодические соревнования в КРСУ, а также ежегодные соревнования всех студентов Кыргызстана.

Литература

1. http://aima.cs.berkeley.edu.
2. http://icpc.baylor.edu/jc/JC2004.pdf.
3. http://icpc.baylor.edu/icpc/finals.