УДК 004.94 DOI: 10.14529/cmse150101

ОБЗОР СИСТЕМ ПРОЦЕДУРНОЙ ГЕНЕРАЦИИ ИГР

М.Г. Меженин

Процедурная генерация контента (ПГК) является одной из наиболее актуальных задач в индустрии видеоигр. Под ПГК понимают автоматическое создание различных составляющих частей игр; особый интерес представляет проблема автоматического создания игровых правил и целых игр. В статье представлен обзор исследований, посвященных данной проблеме; описываются алгоритмы генерации игровых правил и игр различных жанров, в том числе алгоритмы на основе парадигм эволюционного моделирования и логического программирования, а также способы автоматической оценки генерируемых игр. Кратко рассмотрены первые системы генерации игр, реализованные в универсальных игровых программах. Также описаны различные форматы представления и специализированные языки для описания игровых правил в подходящем для алгоритмической обработки виде.

Ключевые слова: процедурная генерация контента, игровой дизайн, универсальные игровые программы, искусственный интеллект.

Введение

Процедурная генерация контента является одним из наиболее актуальных и активно развивающихся направлений исследований в сфере мультимедиа, в частности в индустрии видеоигр. Под *процедурной генерацией контента (ПГК)* понимают автоматическое и полуавтоматическое создание и динамическое изменение различных составляющих частей игр, в том числе игровых объектов и уровней, двумерной и трехмерной графики, эффектов, звуков, музыки, персонажей, сюжетов и др. [10]. Использование ПГК позволяет не только значительно понизить стоимость создания контента, но и решает проблему персонализации, приобретающую большую значимость в связи с увеличением количества потенциальных игроков.

В исследованиях алгоритмов ПГК особенный интерес представляет проблема процедурной генерации игровых правил, которые лежат в основе всех, в том числе и неэлектронных, игр. Решение данной задачи обладает практической значимостью, поскольку процедурная генерация правил позволит не только существенно разнообразить видеоигры за счет динамического изменения правил в процессе игры (в том числе, адаптируя игровой процесс под желания игрока), но и, возможно, создать совершенно новые игровые механики и жанры [24]. Кроме того, системы генерации игровых правил могут быть полезны исследователям алгоритмов искусственного интеллекта в сфере универсальных игровых программ, а также игровым дизайнерам для быстрого создания и тестирования различных игровых прототипов [15].

Процедурная генерация игр невозможна без языка описания игровых правил, то есть некоторого строго-определенного формата их представления в подходящем для алгоритмической обработки виде. Разработка такого языка описания обладает теоретической значимостью, поскольку ее результатом будет строгий научно-терминологический аппарат, который позволит формализовать и описать пространство возможных игровых правил [24, 25].

Процедурная генерация игровых правил является достаточно новой областью исследований. В данной статье рассмотрены ключевые работы в этой сфере, посвященные разработке систем генерации игр и языков их описания.

Статья организована следующим образом. В первом разделе рассмотрены исследования в области универсальных игровых программ. Во втором разделе описаны современные системы процедурной генерации видеоигр. В третьем разделе кратко рассмотрены различные языки описания игр. В заключении суммируются сделанные на основе данного обзора выводы и рассматриваются направления дальнейших исследований.

1. Универсальные игровые программы

В конце 1980-х годов одной из основных и наиболее актуальных задач в области искусственного интеллекта являлась задача создания компьютерной программы для игры в шахматы. С каждым годом шахматные программы становились всё умнее, и уже в 1989 году компьютерная система Deep Thought выиграла в демонстрационном матче у международного мастера по шахматам Дэвида Леви. Вместе с тем, данные системы были настолько сильно специализированы на игру в шахматы, что, по мнению некоторых ученых, достижения в данной области уже не могли использоваться для решения более общих задач искусственного интеллекта (ИИ) [24].

Для решения данной проблемы была предложена концепция *универсальной игровой программы* (УИП), то есть такой системы ИИ, которая была бы способна играть в различные игры, не обладая какими-либо знаниями о конкретной игре до ее начала. Одной из первых работ в данной области является система *METAGAME* [17], состоявшая из двух основных подсистем: модуля ИИ, способного играть в различные настольные игры, и *генератора игр*, назначение которого заключалось в генерации игр для тестирования алгоритмов ИИ.

Таким образом, METAGAME генерирует игры из многочисленного, но конечного множества «симметричных шахмато-подобных игр», удовлетворяющих следующим основным критериям:

— игры происходят на прямоугольной доске с квадратными клетками;
— в игре принимает участие двое игроков;
— начальные позиции фигур и правила симметричны для обоих игроков.

Множество возможных игр в METAGAME задается с помощью формальной *игровой грамматики*, которая определяет начальное расположение фигур, правила их движения и взятия, условия победы и др. Пример описания конкретной игры в формате METAGAME приведен на рис. 1.

Конкретные игры генерируются с помощью случайной выборки из данной грамматики, без проверки на их проходимость, поскольку в общем случае данная проблема является NP-полной. Для отсечения тривиальных игр, авторы реализовали несколько простых проверок, например, не допускающих игры, цель которых заключается в достижении фигурами своей начальной позиции.

Некоторые из параметров генератора также могут задаваться пользователем перед запуском системы. В частности, такими параметрами являются:

— сложность правил (максимальная длина описания правил движения и взятия для каждой фигуры);

— сложность предоставляемого игроку выбора (например, при генерации более простой версии шахмат игроку может не предоставляться выбор, на какую фигуру заменяется пешка при достижении последней горизонтали);
— глубина поиска (например, при меньшей глубине поиска могут рассматриваться игры с меньшим размером доски);
— локальность (максимальное расстояние, преодолеваемое фигурой за один ход).

```
DEFINE man
 MOVING
  MOVEMENT
   LEAP
   <1,1> SYMMETRY {side}
  END MOVEMENT
 END MOVING
 CAPTURING
  CAPTURE
   BY {hop}
   TYPE [{opponent} any_piece]
   EFFECT remove
                                    MOVEMENT
                                     HOP BEFORE [X = 0]
                                         OVER [X = 1]
                                         AFTER [X = 0]
                                     HOP OVER [{opponent} any_piece]
                                     <1,1> SYMMETRY {side}
                                    END MOVEMENT
                                   END CAPTURE
                                  END CAPTURING
                                  PROMOTING
                                   PROMOTE TO king
                                  END PROMOTING
                                  CONSTRAINTS continue_captures
                                 END DEFINE
```

Рис. 1. Частичное описание игры «американские шашки» на языке METAGAME

Несмотря на то, что процедурная генерация игровых правил не была основной целью проекта METAGAME, эта работа оказала большое влияние на последующие исследования в данной области. Многие последующие языки описания и системы генерации игр во многом основываются именно на результатах METAGAME.

Так, в проекте *Gala* [12] было предложено развитие концепций METAGAME с целью создания универсальной игровой программы для игр с неполной информацией. В рамках проекта был предложен одноименный декларативный язык описания игр на основе языка Prolog и алгоритмы поиска оптимальных стратегий для игр, определенных с помощью этого языка.

```
game(blind_tic_tac_toe,
 [players : [a, b],
  objects : [grid_board : array('$size', '$size')],
  params : [size],
  flow : (take_turns(mark,unless(full),until(win))),
  mark : (choose('$player', (X, Y, Mark),
            (empty(X, Y), member(Mark, [x, o]))),
          reveal('$opponent', (X, Y)),
          place((X, Y), Mark)),
  full : (\+(empty(_, _)) ->
  outcome(draw)),
  win : (straight_line(_, _, length = 3, contains(Mark)) ->
           outcome(wins('$player')))]).
```

Рис. 2. Описание игры «слепые крестики-нолики» на языке Gala

В отличие от языка METAGAME, с помощью языка Gala можно описать на порядок большее множество игр, а именно настольные игры на прямоугольной доске с полной и неполной информацией для одного, двух и более игроков. По мнению авторов, их язык описания игр более лаконичен и прост для понимания. На рис. 2 приведен пример

описания на языке Gala игры «слепые крестики-нолики», отличающейся от обычной версии тем, что во время своего хода игрок сообщает лишь клетку, на которую он сходил, не указывая, поставил ли он в нее «крестик» или «нолик».

Язык Gala позволяет задавать некоторые параметры игры в виде переменных; такими параметрами могут быть в том числе и правила движения фигур. Поскольку генерация игровых правил не была целью проекта Gala, его авторы предлагают простой программный интерфейс, благодаря которому для генерации правил на языке Gala можно использовать систему METAGAME.

2. Процедурная генерация игр

2.1. Генерация настольных игр

Исследования в области универсальных игровых программ были в первую очередь ориентированы на изучение алгоритмов ИИ, и генерация игр в них либо не рассматривалась вовсе, либо была основана на простейших алгоритмах.

В работе В. Хома и Д. Маркса [11] задача автоматического создания игр впервые была рассмотрена как отдельная актуальная проблема в области ИИ. Целью работы было создание алгоритма для генерации *сбалансированных* игр, то есть игр, в которых все игроки находятся в равных условиях и имеют одинаковые шансы выиграть. Отметим, что симметричность игры не гарантирует ее сбалансированность, поскольку даже при идентичности правил для обоих игроков игрок, ходящий первым, может обладать преимуществами над вторым игроком (и, в редких случаях, наоборот).

Авторами работы был предложен термин *автоматический игровой дизайн*, под которым понималась автоматическая корректировка баланса игры путем изменения ее правил. Отметим, что идея динамического изменения игрового баланса рассматривалась и в более ранних исследованиях, однако в них корректировка баланса достигалась путем оптимизации заранее определенных параметров игры, а не самих игровых правил.

Работа Хома и Маркса основывалась на коммерчески успешной универсальной игровой программе Zillions of Games (Zillions Development Corp., 1998), игры для которой описывались пользователями вручную на специальном декларативном языке ZRF. Система Zillions of Games предназначалась для визуализации любых настольных игр на доске, описанных на языке ZRF, с возможностью игр в них с УИП.

Предложенный алгоритм автоматического изменения баланса основан на генетическом алгоритме и заключается в следующем. На каждой итерации алгоритма из некоторого набора игр путем изменения и комбинирования их правил генерируются новые игры, в каждую из которых УИП Zillions of Games играет сама с собой 100 матчей. Каждая игра оценивается с помощью функции оценки по шкале баланса (большую оценку получают игры с минимальной разницей между количеством побед у двух игроков) и шкале разнообразия (большую оценку получают игры, наименее похожие на ранее сгенерированные результаты), после чего формируется новый набор из старых и новых игр с наибольшей оценкой и начинается следующая итерация.

Использование генетических алгоритмов и оценка результатов некоторой симуляции, а не самих особей, являются достаточно распространенными методами процедурной генерации контента; в данной работе было продемонстрировано, что эти методы могут быть успешно использованы и для динамической генерации игр.

В дальнейшем данные методы были развиты в проекте *Ludi* [1], одним из главных достижений которого является игра Yavalath — первая в мире коммерчески изданная игра, полностью сгенерированная программой в автоматическом режиме.

В рамках проекта Ludi был разработан специальный функциональный язык для описания настольных игр на доске, названный Ludi GDL. Описания на данном языке состоят из абстракций высокого уровня, что позволяет легко и лаконично описывать различные игры. Ludi GDL также является расширяемым, но для успешной работы любые дополнительные абстракции необходимо реализовывать и на низком уровне. Пример описания игры «крестики-нолики» на языке Ludi GDL приведен на рис. 3.

```
(game Tic-Tac-Toe
  (players White Black)
  (board (tiling square i-nbors)(shape square)(size 3 3))
  (end (Allwin(in-a-row 3)))
)
```

Рис. 3. Описание игры «крестики-нолики» на языке Ludi GDL

При оценивании игр в системе Ludi акцент делается на поиск наиболее интересных игр. Для этой цели по каждой сгенерированной игре в Ludi проводится некоторое количество автоматических матчей с двумя компьютерными игроками, после чего игра оценивается по 57 критериям, разделенным на три основные группы:

— *объективные* критерии для оценки качеств самих правил независимо от тестовых матчей;
— критерии *играбельности (playability)* для оценки результатов тестовых матчей;
— *качественные* критерии для оценки динамики тестовых матчей.

Авторами отмечается, что объективные критерии были наименее эффективны в поиске интересных игр. Критерии играбельности, характеризующие интересность и базовые свойства игрового процесса, в Ludi повторяют описанные выше критерии сбалансированности игр с добавлением критерия средней длины матча.

Особый интерес представляют качественные критерии, основанные на анализе *историй лидирования* в тестовых матчах, то есть данных о степени лидирования одного игрока над другим в каждый из ходов. Пример истории лидирования приведен на рис. 4.

Рис. 4. Пример истории лидирования

В данном примере перед своей победой на 27-м ходу белый игрок находится в проигрышной позиции с 18 по 22 ход. По мнению авторов, если при некотором наборе правил во многих матчах в процессе игры происходит смена лидера, данная игра является более «драматичной» и интересной.

Генерация игр в Ludi реализована на основе стандартных методов генетического программирования. В рамках тестирования системы с помощью Ludi было сгенерировано 1389 игр, из которых 19 были признаны авторами достаточно интересными, а еще 2, названные Yavalath и Pentalath, — крайне интересными. Описание игры Yavalath на языке Ludi GDL приведено на рис. 5. Правила Yavalath схожи с «крестиками-ноликами» на гексагональной доске; главное отличие — игрок побеждает, поставив четыре фигуры в ряд, но проигрывает, поставив в ряд только три фигуры.

```
(game Yavalath
  (players White Black)
  (board (tiling hex)(shape hex)(size 5))
  (end (Allwin(in-a-row 4)) (Alllose (in-a-row 3)))
)
```

Рис. 5. Описание игры Yavalath на языке Ludi GDL

Дополнительное правило может показаться излишним, но на самом деле оно добавляет игре стратегическую глубину. Например, на рис. 6 приведена позиция, в которой ход #1 белого игрока вынуждает черного игрока сделать блокирующий ход #2, и тем самым проиграть из-за нового правила. Таким образом, игроки могут в некоторой степени управлять ходами противника с помощью нетривиальных последовательностей ходов – подобное появление сложных стратегий из достаточно простых правил подтверждает высокое качество и интересность данной игры.

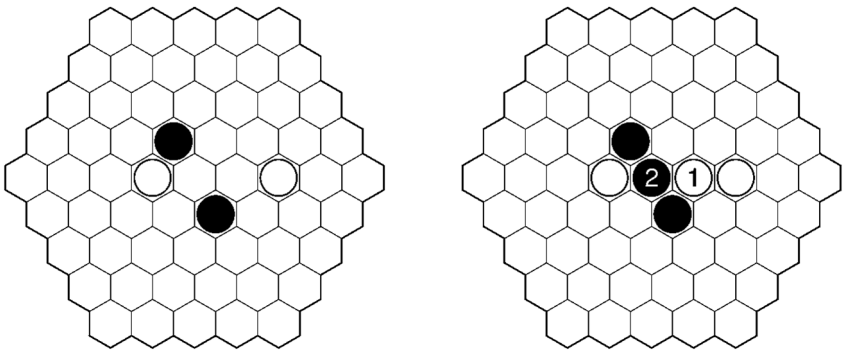

Рис. 6. Выигрышная последовательность ходов белых в Yavalath

Yavalath и Pentalath были впоследствии изданы в коммерческом виде издательством Nestorgames и являются одними из наиболее успешных игр в их каталоге.

2.2. Генерация аркадных видеоигр

В конце 2000-х годов несколько исследователей параллельно начали работу над системами процедурной генерации правил для аркадных видеоигр. Под аркадными играми мы будем понимать двумерные и трехмерные игры, игровой процесс которых заключается в движении некоторых объектов, их столкновении, появлении и исчезновении, и так далее. Данный набор действий характеризует достаточно базовые понятия, из кото-