Санкт-Петербургский государственный университет

Фундаментальная информатика и информационные технологии
Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

Тен Ен Ми

Генерация динамического контента для коллекционной карточной игры на базе методов машинного обучения

Магистерская диссертация

Научный руководитель:
Ст. преподаватель Салищев С. И.

Рецензент:
Глава отдела разработки Каскевич Д. Е.

Санкт-Петербург
2017

SAINT-PETERSBURG STATE UNIVERSITY

Fundamental Informatics and Information Technology
Mathematical and Software Support for Computers, Computer Systems and Networks

En Mi Ten

Dynamic content generation for collectible card game using machine learning methods

Master's Thesis

Scientific supervisor:
Senior Lecturer S.I. Salishchev

Reviewer:
Lead Developer D.E. Kaskevich

Saint-Petersburg
2017

Оглавление

Введение

В последние годы интерес к генерации игрового контента на основе существующего посредством методов машинного обучения, а также значимость генерируемых данных в игровой индустрии значительно выросли. С этим связана и популярность этого направления в области научных исследований: постоянно изучаются методы и новые возможности [12], [17], [19] создания высококачественного игрового контента (любая составляющая игры: уровни, музыка, правила, атмосфера, текстуры, герои, сюжеты и т.п.) с участием или без участия человека [11].

Автоматическая генерация игрового контента - актуальная задача в области разработки игр [1], [16]. Она не только добавляет играм разнообразие, подчеркивает красоту и многогранность дизайна [15], но и значительно ускоряет, облегчает процесс разработки [25]. Таким образом, механизмы генерации контента привлекают большое количество новых игроков, при этом снижая общую стоимость разработки и поддержки игр.

Одной из наиболее популярных областей применения технологий генерации динамического игрового контента являются элекронные коллекционные карточные игры, имеющие на сегодняшний день значительную популярность по сравнению с играми других жанров. Решение этой задачи практически значимо, так как карты - основной и главный элемент в таких играх. Разнообразие и количество карт прямо пропорционально успеху, популярности и прибыльности проекта, поэтому карточные стратегии постоянно требуют обновления, добавления новых уникальных элементов. В данном контексте автоматическая генерация новых карт является полезным и важным инструментом для игровых дизайнеров. Также стоит отметить, что каждая коллекционная карточная игра использует уникальную механику и накладывает ряд ограничений на процесс выбора метода генерации. В этой работе рассматривается технология генерации карт в недавно выпущенной коллекционной карточной стратегии Mighty Party [27], которая предоставляет разнообразный набор механик, взаимодействий и существ. Раз-

работанный алгоритм обрабатывает все множество карт и пополняет его уникальными элементами, не нарушающими игровой баланс. Описанное решение способно не только генерировать уникальный контент на основе имеющегося, но и реагировать на добавление новых игровых механик.

1. Обзор

1.1. Коллекционные карточные игры

Коллекционные карточные игры — это разновидность карточных игр, впервые появившихся в 1993 году и состоящий из специально разработанных наборов игральных карт. Популярные и успешные игры этого жанра обычно имеют больше тысячи уникальных карт. Один из первых и наиболее успешных таких проектов является игра Magic: The Gathering (больше 16000 карт) [18].

Как правило, карточные стратегии начинаются со стартовой колоды, которая может быть модифицирована с помощью бустеров, которые содержат случайные карты разной редкости, обычно от 5 до 15 карт [8]. Одна из карт в бустере — редкая или уникальная карта, получить которую гораздо труднее, чем остальные карты, и чаще всего имеет более высокую ценность, чем остальные. Эти значения могут меняться: карты могут быть недоступными в некоторых игровых форматах или мета игры может в корне измениться с появлением новых карт [26]. В любом случае, имея достаточное количество существ, игрок может создавать колоды для битв.

Помимо невероятно популярной Magic: The Gathering, среди успешных проектов — Pokémon, Legend of the Five Rings, Lord of the Rings и т.д. сейчас огромную популярность завоевывают цифровые коллекционные карточные игры, чему способствовал успех игры Hearthstone (рис. 1), выпущенной во всем мире в 2014 году.

Hearthstone основывается на информации уже существующей вселенной Warcraft, использует те же элементы, персонажей и реликвии. Игра представляет собой пошаговую карточную игру между двумя игроками с колодами из тридцати карт вместе с выбранным варлордом с уникальными способностями. Игроки используют очки маны для произнесения заклинаний или призыва существ на поле с главной целью снизить здоровье противника до нуля. За выигранные бои игрок может получить золото, награды в виде новых карт и другие внутриигровые

Рис. 1: Популярная карточная стратегия Hearthstone от компании Blizzard.

призы. Также игроки могут покупать наборы новых карт через золото и микротранзакции, чтобы улучшить свои колоды. Есть несколько режимов игры, включая случайные и ранговые бои, а также ежедневные и еженедельные квесты, которые помогут заработать больше золота и карт. Новый контент для игры включает в себя добавление новых наборов карт и геймплэя, в виде однопользовательских приключений, которые вознаграждают игрока коллекционными карточками по завершении.

Сейчас игры такого жанра невероятно популярны, и это показывают не только отчеты и графики больших аналитических сервисов, но и тот факт, что на самой популярной платформе онлайн вещания Twitch, где основным контентом являются игры и киберспорт, в топе самых просматриваемых — карточные стратегии.

Количество игроков, предпочитающих такой жанр, имеет невероятный рост, так, например, в апреле 2016 количество игроков в Hearthstone превысило отметку в 50 миллионов, а 1 мая 2017 разработчики заявили, что уже более 70 миллионов людей играют в эту карточную стратегию [5]. Неудивительно, что такая популярность жанра способствует тому,

Рис. 2: Недавно выпущенная карточная игра Faeria.

что с каждым днем количество новых карточных стратегий увеличивается: за последний месяц на популярной игровой платформе Steam произошел релиз таких игр, как Faeria (рис. 2), Shardbound, KROSMAGA (рис. 3), Mighty Party и это еще не весь список.

Рис. 3: Недавно выпущенная карточная игра KROSMAGA.

Как было указано выше, основным и главным элементом коллекционных карточных игр являются карты, от разнообразия, количества, уникальности и подобных характеристик которых зависит успех и популярность проектов. Это объясняет тот факт, что карты в этих играх — явление динамическое: разработчики, дизайнеры постоянно должны поддерживать интерес пользователя к игре, модифицируя различными способами ее контент (карты). Именно поэтому частые обновления карточных стратегий привычны для поклонников жанра, к примеру, с момента релиза (январь 2017) в Mighty Party вышло порядка 6 апдейтов с добавлением, удалением существ, изменением способностей у бойцов и т. д.

Неудивительно, что генерация контента для игр именно этого жанра - неотъемлемый элемент разработки.

1.2. Генерация динамического контента

Машинная генерация игрового контента становится все более популярным инструментом как в области разработки игр, так и в области исследований [17], [24]. Воспроизведение новых элементов игры посредством методов машинного обучения применяется для повышения основных игровых метрик, снижения производственных затрат и усилий, экономии места для хранения и т. д. Научные исследования в этой области решают вышеуказанные проблемы, а также изучают новые возможности применения этой технологии, например, внедряя генерацию игрового контента, который может адаптироваться к игроку. В 2014 году во всем мире вышла игра Middle-earth: Shadow of Mordor [20], просто перевернув игровую индустрию своей системой Nemesis.

В этой игре невозможно встретить двух одинаковых противников, так как они генерируются в зависимости от действий игрока, поэтому и уникальны по целому набору характеристик: внешность, речь, взаимоотношения с другими орками, способности, звание в орде, и т.д.. Таким образом, система Nemesis создание вокруг игрока динамичный мир с индивидуальными врагами, на формирование и развитие которых влияет он сам.

Именно благодаря инструменту генерации динамического контента и по сей день не угасает интерес игроков к таким продуктам, как Minecraft и No Man's Sky [6].

Итак, причины использования генерации контента в играх:

- Низкая себестоимость;
- Регулируемый объем генерируемого контента;
- Разнообразие;
- Адаптация;
- Возможность изменения генерируемого контента с учетом пользовательских предпочтений.

Генерация контента с помощью методов машинного обучения определяется в этой работе, как создание игрового контента с применением моделей, которые были обучены на уже существующем контенте [2]. Эти модели могут обучаться с использованием разных алгоритмов обучения [14], [22], [13], включая нейронные сети, вероятностные модели, деревья решений и т. д. Генерация также может быть разных типов.

Виды генерируемого контента:

- 3D-модели,текстуры, эффекты;
- Анимация;
- Звуки и музыка;
- Уровни, ландшафты, здания;
- Предметы и существа;
- Поведение и ИИ;
- Сюжет и задания и т. д.

Эта работа сфокусирована на функциональном игровом контенте, то есть, на элементах, изменение которых могут повлиять на внутриигровые действия игрока, в данном случае — это карты.

Интересные карты - это то, что делает коллекционные карточные игры уникальными между собой, а разнообразие и количество карт значительно повышают интерес и удовольствие от игрового процесса, как и в описанных примерах игр другого жанра.

Но создание новых произвольных карт — процесс нелегкий и небыстрый. Обычно над этим работает команда геймдизайнеров [3], которая продумывает и создает наборы этих новых карт-существ с уникальными способностями, силой, здоровьем, принадлежащих той или иной расе, группе, учитывая текущий игровой баланс и многие другие факторы. Поэтому для многих известных карточных игр задача создания "виртуального геймдизайнера", который будет способен создавать

неограниченное количество интересных произвольных карт, то есть задача генерации контента, считается актуальной.

Генерация игрового контента сильно отличается от генерации контента в других областях [4], ведь игры динамичны, и большинство игровых материалов имеют структурные ограничения, проверить которые можно и нужно, только играя в них. Уровень, который структурно не позволяет игрокам завершить его, не будет хорошим результатом, даже если он визуально привлекателен.

1.3. Mighty Party

Mighty Party — быстрая коллекционная карточная стратегия с дуэльной PvP боевкой на поле (грид 3x4).

Рис. 4: Сбор боевой армии в Mighty Party.

Рис. 5: Сражение в Mighty Party.

На момент проведения исследований по работе насчитывалось около

130 карт существ (на рис. 4 и 5 представлены армии игрока и типичное сражение).

Каждая карта относится к одной из трех фракции: Order, Nature, Chaos и к одному из трех классов: Melee (может быть выставлено на любую клетку только в первые два ряда), Archer (в последние два ряда), Building (могут находится на любой клетке всех трех рядов).

Конечно же помимо принадлежности к тому или иному классу и фракции, существо уникально своими способностями (призыв других существ, возвращение атаки, пробивание, блок, кража атаки и т.д.), которые могут проявляться в разное время хода: при появлении на поле, в конце хода, при атаке и в начале хода; также уровнем здоровья и атаки (которые в свою очередь повышаются при наличии определенного количества таких же карт в твоей армии).

2. Актуальность темы

- Популярность карточных коллекционных игр. По результатам исследования статистического сервиса SuperData коллекционные карточные игры (Magic the Gathering, Hearthstone, Shadowverse, Faeria и т. д.) заработают $ 1.4 млрд в 2017 году. Цифровые игровые аккаунты игроки составляют 61% от аудитории ККИ благодаря большей доступности компьютерных карточных стратегий, чем их физические аналоги. Hearthstone: Heroes of Warcraft заработали примерно в четыре раза больше, чем его ближайший конкурент в 2016 году и просто перевернул рынок коллекционных карточных игр. Shadowverse, запущенный в середине 2016 года, заработал в том же году $ 100.1 млн.;

- Необходимость постоянного обновления/добавления контента для разнообразия игры, поддержания пользовательского интереса, следовательно и привлечения новой аудитории;

- Упрощение и ускорение работы геймдизайнеров, как в инди-проектах, так и в крупных компаниях. Генерация является интересным альтернативным вариантом трудоемкого ручного создания контента.

3. Постановка задачи

Основные задачи:

- Исследование области;
- Изучение существующих решений, методов и систем;
- Анализ достоинств и недостатков существующих решений, методов и систем;
- Формирование теоретической основы будущего исследования;
- Программная реализация;
- Внедрение полученной системы и проверка ее работы на реальных данных;
- Анализ полученных результатов.

Таким образом, целью этой работы является реализация системы, позволяющей генерировать новых карт на базе имеющегося игрового контента.

Требования к системе:

- “Среднее” быстродействие: поскольку генерация производится не “на лету”, то вопрос быстродействия является второстепенным, в то же время от этого напрямую зависит объемы сгенерированного контента;
- Корректность и правдоподобность генерируемого контента: полученные карты не должны нарушать основные принципы игровой механики или игрового баланса;
- Уникальность и разнообразие: полученный контент должен быть максимально разнообразен. Похожие (определяются с помощью применения определенной метрики) карты должны быть исключены;

- Формирование теоретической основы будущего исследования;
- Адаптивность, гибкость и управляемость: добавление новых механик или изменения баланса должно учитываться при генерации контента.

4. Основная часть

4.1. Рекуррентные нейронные сети

Сущесвует множество различных классов искусственных нейронных сетей, отличающихся друг от друга архитектурой, форматом, размером входных параметров и т.д. Для задачи, рассматриваемой в данной работе, был выбран класс рекуррентных нейронных сетей.

Рекуррентная нейронная сеть - это модификация стандартной нейронной сети прямого распространения, которая позволяет моделировать последовательности данных. В каждый момент времени сеть получает данные на вход, обновляет скрытое состояние и выдает информацию, предсказание на выход. Это менее популярный тип нейронных сетей, чем, например, многослойный перцептрон, но при этом умеющий работать с динамическими процессами.

Значительную роль в выборе технологии играет ограничение других классов нейронных сетей в размере данных: такие сети принимают вектор фиксированной длины в качестве входных данных и выдают вектор фиксированного размера в качестве результата. Также многие модели ограничены фиксированным количеством вычислительных шагов, к примеру, слоев. Основное отличие и преимущества рекуррентных сетей [7] - возможность работать с последовательностями векторов: как на входе, так и на выходе или, как чаще всего происходит - и на входе, и на выходе.

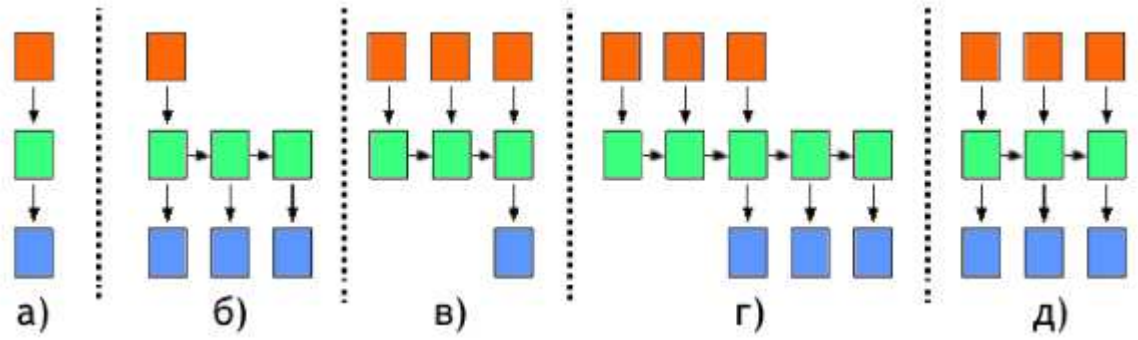

Рис. 6: Модели нейронных сетей.

На рисунке 6 схематично представлены несколько моделей архи-

тектуры нейронных сетей, где прямоугольник - это вектор, а стрелки представляют функции. Оранжевым цветом выделены входные векторы, синим - выходные, а зеленые прямоугольники скрытые слои рекуррентных нейросетей. Слева направо:

- (а) Модель обработки без рекуррентных нейросетей с фиксированным размером данных для входа и выхода (например, классификация изображений);
- (б) Последовательности на входе (например, анализ настроения текста, когда на вход подается предложение, которое впоследствии классифицируется, как выражение положительных или отрицательных чувств);
- (в) Последовательности на выходе (например, на вход подается изображение и выводит предложение слов);
- (г) Последовательности на входе и на выходе (например, машинный перевод: рекуррентная нейросеть читает предложение на одном языке, а затем выводит предложение на другом языке);
- (д) Синхронный ввод и вывод последовательностей (например, классификация видео, то есть, пометка каждого кадра видео).

Стоит отметить, что в каждом из этих случаев размер последовательностей заранее не зафиксирован, так как определенное рекуррентное преобразование (зеленые прямоугольники) может применяться сколько угодно раз.

В качестве примера предположим, что имеется словарь только из четырех возможных букв: ”s”, ”k”, ”i”, ”l”. Задача - подготовить сеть на обучающей последовательности ”skill”. На самом деле эта обучающая последовательность - источник четырех отдельных учебных примеров:

1. Символ «k» следует вернуть с учетом имеющегося контекста «s»;
2. Символ «i» следует вернуть в контексте имеющегося «sk»;

3. Символ «l» следует вернуть в контексте имеющегося «ski»;

4. И, наконец, символ «l» должен быть следующим, учитывая имеющийся контекст «skil».

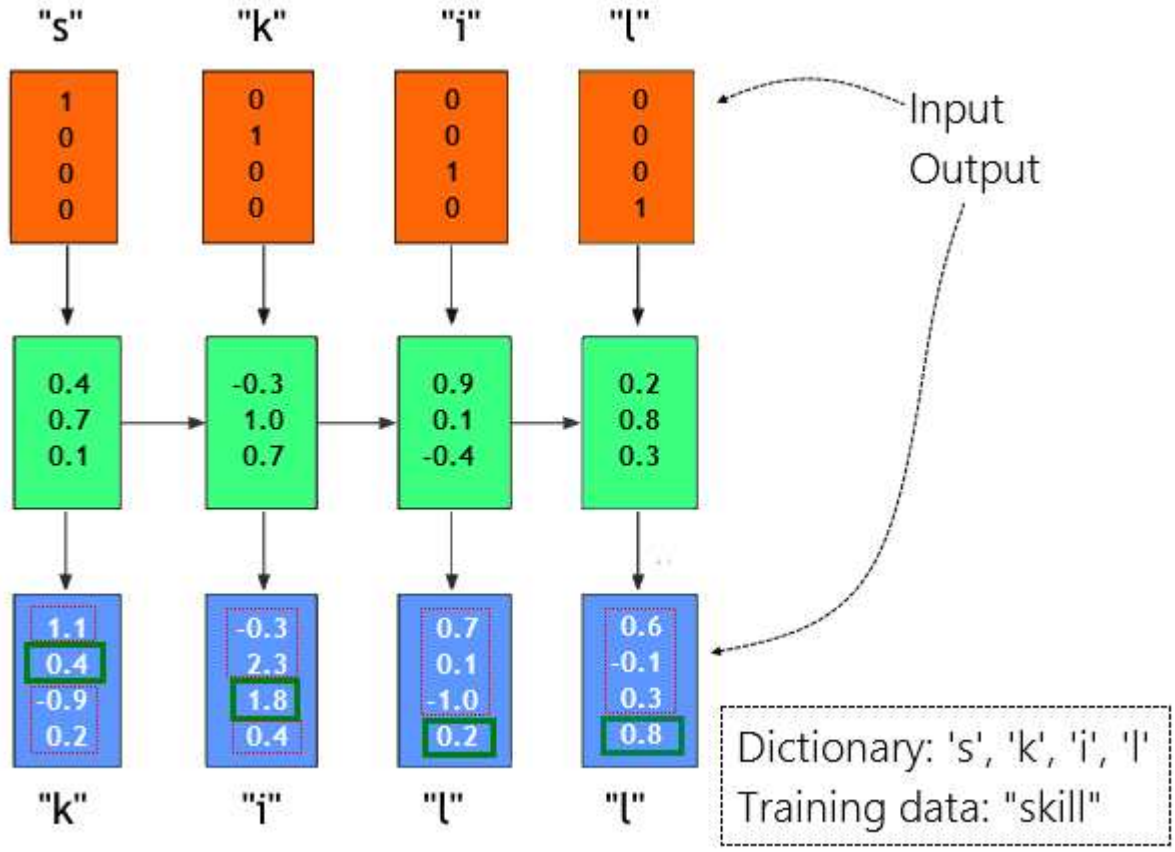

Рис. 7: Рекуррентная нейронная сеть с 4-мерными входными и выходными слоями и скрытым слоем из 3 нейронов.

Например, по рис. 7 видно, что в первый момент времени, когда сеть распознала символ ”s”, в качестве появления следующим символом ему было присвоено значение ”1.1”, ”0.4” - букве ”k”, ”-0.9” - букве ”i” и ”0.2” - букве ”l”. Зная из нашей обучающей выборки (”skill”), что следующий правильный символ - ”k”, хочется увеличить степень этого символа (выделено зеленым цветом) и понизить вероятность появления остальных букв (отмечено красным). Аналогично в каждый момент времени есть целевой символ с желательно наибольшим значением вероятности появления.

В общем случае, каждый символ представлен, как n-мерный вектор, состоящего из всех нулей, кроме единственного значения, который соответствует индексу символа в словаре, и передается в сеть по одному с шаговой функцией. Далее рассматривается последовательность выходных n-мерных (n - количество символов в словаре) векторов, которые интерпретируются, как вероятности появления следующим, которую на текущем шаге рекуррентная нейросеть присваивает каждому из символов.

Конечно, технологии с последовательной обработкой данных мощнее, чем сети с зафиксированным количеством вычислительных шагов и, следовательно, более привлекательны при создании более интеллектуальных систем. Более того, рекуррентные сети могут подавать на вход вектор и состояние (тоже вектор) с определенной функцией создания нового состояния. С точки зрения программирования это может быть интерпретировано, как запуск определенной программы с заданными входными параметрами и несколькими внутренними переменными. Таким образом, рекуррентные нейроные сети могут описывать программы. Более того, рекуррентные сети считаются Тьюринг-полными, так как с соответствующими весами могут имитировать произвольные программы.

Важно понимать, что сформулировать и обучить мощные модели, которые научатся последовательно обрабатывать данные возможно, даже если они на входе/выходе не представлены в виде последовательностей фиксированной длины. Как, например, это применялось при обучении считывать номера домов слева направо с фотографий Google Maps [9] или генерации изображений, где к холсту последовательно добавляется цвет [14].

4.2. Формальное представление

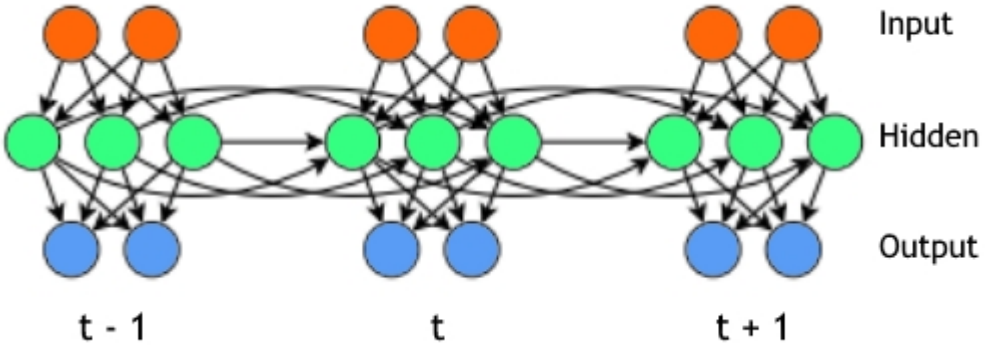

Рис. 8: Архитектура рекуррентной нейронной сети. Высокая размерность скрытого слоя и нелинейность активационной функции, используемой единицами этого слоя, являются источником богатой динамики рекуррентных нейросетей.

Стандартная рекуррентная нейронная сеть формализуется следующим образом (рис. 8): при заданной последовательности входных векторов $(x_1, ..., x_T)$ сеть вычисляет последовательность скрытых состояний $(h_1, ..., h_T)$ и последовательность выходов $(o_1, ..., o_T)$ путем итерации следующих уравнений для t от 1 до T:

$$h_t = \tanh(W_{hx}x_t + W_{hh}h_{t-1} + b_h) \quad (1)$$

$$o_t = W_{oh}h_t + b_o \quad (2)$$

В этих уравнениях W_{hx}, W_{hh}, W_{oh} - весовые матрицы (input-hidden, hidden-hidden, hidden-output, соответственно), векторы b_h и b_o - смещения, tanh применяется поэлементно. Неопределенное выражение $W_{hh}h_{t-1}$ в момент времени $t = 1$ заменяется специальным начальным вектором смещения h_{init}.

Градиенты рекуррентных сетей несложно вычислить посредством метода обратного распространения ошибки во времени (Backpropagation Through Time, BPTT, [21]), поэтому можно предположить, что они легкообучаемы с помощью алгоритма градиентного спуска. Но взаимосвязь между параметрами и динамикой сети - величина крайне нестабильная, что исключает метод градиентного спуска из числа эффек-

тивных. Доказано, что градиент затухает (или, реже, взрывается) экспоненциально по мере обратного распространения во времени, что говорит о невозможности использования градиентного спуска для обучения. К тому же, случайная тенденция обратного градиента к экспоненциальному раздутию значительно увеличивает дисперсию градиентов и делает обучение неустойчивым. В какой-то момент времени эти теоретические результаты и эмпирическая трудность обучения рекуррентных нейронных сетей привели практически к полному отказу от исследований в этой облайти. Один из способов побороть неспособность градиентного спуска к изучению долгосрочных временных структур в стандартных рекуррентных сетях заключается в модификации модели: добавление модулей "памяти", специально предназначенных для хранения информации в течение длительных периодов времени. Эта технология получила название "Долгая краткосрочная память" ("Long-Short Term Memory", [23]). Долгая краткосрочная память позволяет обрабатывать наборы данных, которые требуют долгосрочного запоминания и отзыва, но даже на этих наборах данных технология лучше, чем технология стандартных рекуррентных нейросетей, обучаемых с помощью HF-оптимизатора [10].

Для решения задачи генерации текстов часто используются рекуррентные нейронные сети с некотрыми модицфикациями для повышения производительности [7]. Динамика скрытого слоя зависит от входных данных и матрицы весов hidden-hidden. В стандартных рекуррентных сетях входные данные x_t после преобразований посредством матрицы W_{hx}, подаются на вход текущего скрытого слоя как аддитивное смещение. Модификация стандартных методово заключалась в том, чтобы входной символ помимо этого смещения, определял всю hidden-hidden матрицу с ее нелинейной динамикой.

Этот подход [7] представляет рекуррентную сеть как модель неограниченного дерева, каждый узел которого - вектор скрытого состояния, а ребра помечены символом, определяющим, как родительский узел порождает дочерний узел. Такое представление подчеркивает сходство рекуррентных сетей с марковской моделью, которая хранит знакомые

символьные строки в дереве, при этом ясно, что дерево рекуррентных нейронный сетей потенциально мощнее марковской модели, поскольку распределенное представление узла допускает разные узлы для обмена знаниями. К примеру, символьная строка "ing" вполне вероятна после "fix", а также вполне вероятна после "break". Если векторы скрытого слоя, получив "fix" и "break", пониммают, что это может быть основы глаголов, тогда на это представление может быть выдан символ "i", чтобы сформировать скрытое состояние, которое предскажет символ "n". Для того, чтобы предсказание было хорошим, нам требуются и знания о глагольных олсновах в предыдущем скрытом состоянии, и символ "i". Связанность этих параметров и стала основой подхода мультипликативного взаимодействия. Эти модификации рекуррентной сети формально выглядят так:

$$h_t = \tanh(W_{hx}x_t + W_{hh}^{(xt)}h_{t-1} + b_h) \tag{3}$$

$$o_t = W_{oh}h_t + b_o \tag{4}$$

То есть, матрица весов hidden-hidden функция текущего входного параметра x_t.

Они идентичны выражениям (1) и (2), за исключением того, что W_{hh} заменяется на $W_{hh}^{(xt)}$, позволяя входным символам задавать разные hidden-hidden матрицы.

Пусть имеется M матриц, $W_{hh}^{(1)}, ., , , W_{hh}^{(M)}$, где M - число размерностей x_t, тогда $W_{hh}^{(xt)}$ можно представить, как:

$$W_{hh}^{(xt)} = \sum_{m=1}^{M} x_t^{(m)} W_{hh}^{(m)} \tag{5}$$

где $x_t^{(m)}$ - m-я координата x_t.

Цель языковой генерации - предсказать следующий символ в последовательности. Более формально, учитывая последовательность обучения $(x_1, ..., x_T)$, нейронная сеть использует последовательность своих выходных векторов $(o_1, ..., o_T)$ для получения последовательности прогнозирующих распределений $P(x_{t+1}|x_{\leq T}) = softmax(o_t)$, где распреде-

ление softmax определяется как:

$$P(softmax(o_t) = j) = \exp(o_t^{(j)}) / \sum_k \exp(o_t^{(k)})) \tag{6}$$

Суть моделирования языка - максимизировать общую логарифмическую вероятность обучающей последовательности $\sum_{t=0}^{T-1} \log P(x_{t+1}|x_{\leq t})$, что подразумевает обучение сети распределениями вероятности по последовательностям. Хотя скрытые элементы являются детерминистическими, мы можем стохастически производить образцы с помощью построенной сети, так как выходные элементов определяют условное распределение $P(x_{t+1}|x \leq t)$. Мы можем использовать экземпляры из этого условного распределения для получения следующего символа в сгенерированной строке, а также предоставлять его в качестве следующего на вход в рекуррентной нейронной сети.

4.3. Описание данных

Данные представлены в формате XML-файла (31348 символов, около 5000 слов), который содержит полное описание героев коллекционной карточной игры Mighty Party.

Герой или юнит в "Mighty Party" - это набор уникальных способностей ("Пробивание", "Регенерация", "Увеличение атаки" и т.д.), а также признаки, характеризующие каждое существо: уровень здоровья, уровень атаки, название, а также идентификатор, используемый непосредственно разработчиками.

Описания каждого героя или юнита представлены в виде XML-узла. Ниже в качестве примера приведено описание существующего героя "Defender" или "Защитник", содержащегося в обучающей выборке:

```
<unit>
  <main>
        <title>Defender</title>
        <id>17</id>
        <rarity>1</rarity>
        <health>13</heatlh>
        <attack>2</attack>
  </main>
  <skills>
        <skill>
                <id>Block</id>
                <value>2</value>
        </skill>
  </skills>
</unit>
```

4.4. Эксперименты

В качестве тестируемой нейронной сети была выбрана рекуррентная нейронная сеть со следующей архитектурой - 3 слойная рекуррентная нейронная сеть с 256 скрытыми узлами на каждом слое. На вход нейронная сеть получала данные в вышеписанном формате с описанием существующих карт в виде одного файла размером в 1.2 MB. Получив символ, сеть выдает распределение по тому, какие символы, вероятно, стоят после него. Получив это распределение и отправив его обратно, выдается следующая буква. Этот процесс повторяется, таким образом, формируется текст. Ниже представлены примеры после нескольких часов первых экспериментов, сеть работала, обучаясь на описании порядка 80 существ. В целом среди результатов было несколько правильно сформулированных экземпляров карт, в частности:

```
<unit>
  <main>
        <title>Toad Prince</title>
        <id>6302</id>
        <rarity>2</rarity>
        <health>10</heatlh>
        <attack>1</attack>
  </main>
  <skills>
        <skill>
                <id>Return</id>
                <value>4</value>
        </skill>
  </skills>
</unit>
```

Правда, тяжело адекватно оценить полученное существо с точки зрения баланса. Аналогичным образом среди результатов встречались и карты с некорректным описанием, как, например, в нижепредставленном фрагменте можно заметить "Вампиризм" с нулевым значением, что по сути никакого смысла и не несет.

```
<unit>
  <main>
```

```
        <title>Void</title>
        <id>394</id>
        <rarity>3</rarity>
        <health>5</heatlh>
        <attack>6</attack>
  </main>
  <skills>
        <skill>
                <id>Vampire</id>
                <value>0</value>
        </skill>
  </skills>
</unit>
```

Видно, что даже на представленных фрагментах сгенерированного результата у "новых" существ есть идентификатор, который создавался случайным образом в процессе генераци, но позднее правильно составленные карты получали инкрементирующий идентификатор относительно "реальных", поэтому средства оценки корректности карт был просто необходим.

4.5. Оценка генерируемых элементов

Чаще всего у карточных коллекционных игр несложные правила и механика, эта особенность присутствует и в новой стратегии Mighty Party. Сложность таких игр возникает засчет наличия в них огромного количества уникальных существ, что также усложняет или даже делает невозможным процесс нахождения идеального баланса для их создателей: вводя новых героев, геймдизайнеры не могут обозначить одну карту, как "очень редкая", а другую - как "очень-очень редкая", ведь есть четкая, заранее определенная градация, которой нужно придерживаться.

Среди подзадач, на которые был разделен процесс исследования, значимую роль играет изучение, реализаия механизма, который смог бы оценить сгенерированный результат с учетом текущего баланса или который также смог бы найти недо- или пере- оцененные карты, то есть, например, карты, способные на "большее" по сравнению с другими картами данной группы. Такой анализ дает возможность оценить "мощность" карт как с точки зрения их самостоятельных действий, так и, учитывая силу взаимодействия героев на "столе". То есть, необходимо найти модель такой системы, а также выбрать основную характеристику для сравнения.

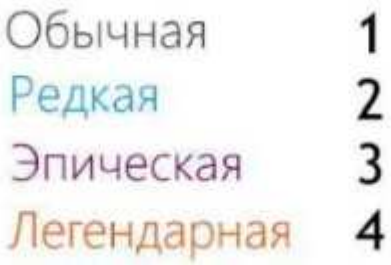

Рис. 9: Оценка карт Mighty Party по редкости выпадения в сундуках.

В карточных стартегиях в качестве оценочного элемента можно взять стоимость призыва, силу существ и т.п., по которым удобно их группировать для дальнейшего сравнения. В системе, разарботанной в ходе исследования для игры Mighty Party параметром-оценкой является редкость выпадения существ со значениями, представленными на рис. 9.

Рис. 10: Два существа разной редкости без дополнительных способностей.

Рассмотрим на примере двух существ: "Воина" обычной редкости и легендарного "Принца" (рис. 10), у которых нет дополнительных способностей. Наша задача - выяснить, какой герой мощнее и эффективнее.

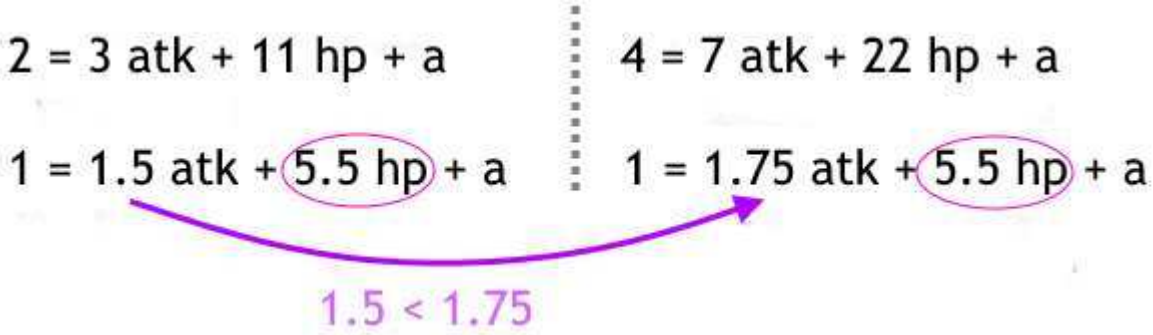

Рис. 11: результат: "Принц" мощнее "Воина".

Этот случай (рис. 11) кажется довольно примитивным в силу отсутствия у существ каких-либо самостоятельных эффектов или способностей при взаимодействии с другими: пробивание, двойная атака, исцеление и т.д.

$$\text{Rarity} = \sum_{i=0}^{n} (\text{coef}_i * \text{ability}_i)$$

Рис. 12: Общая формула для вычисления "мощи" существа.

На рис. 12 представлена общая модель, также видно, что мощность

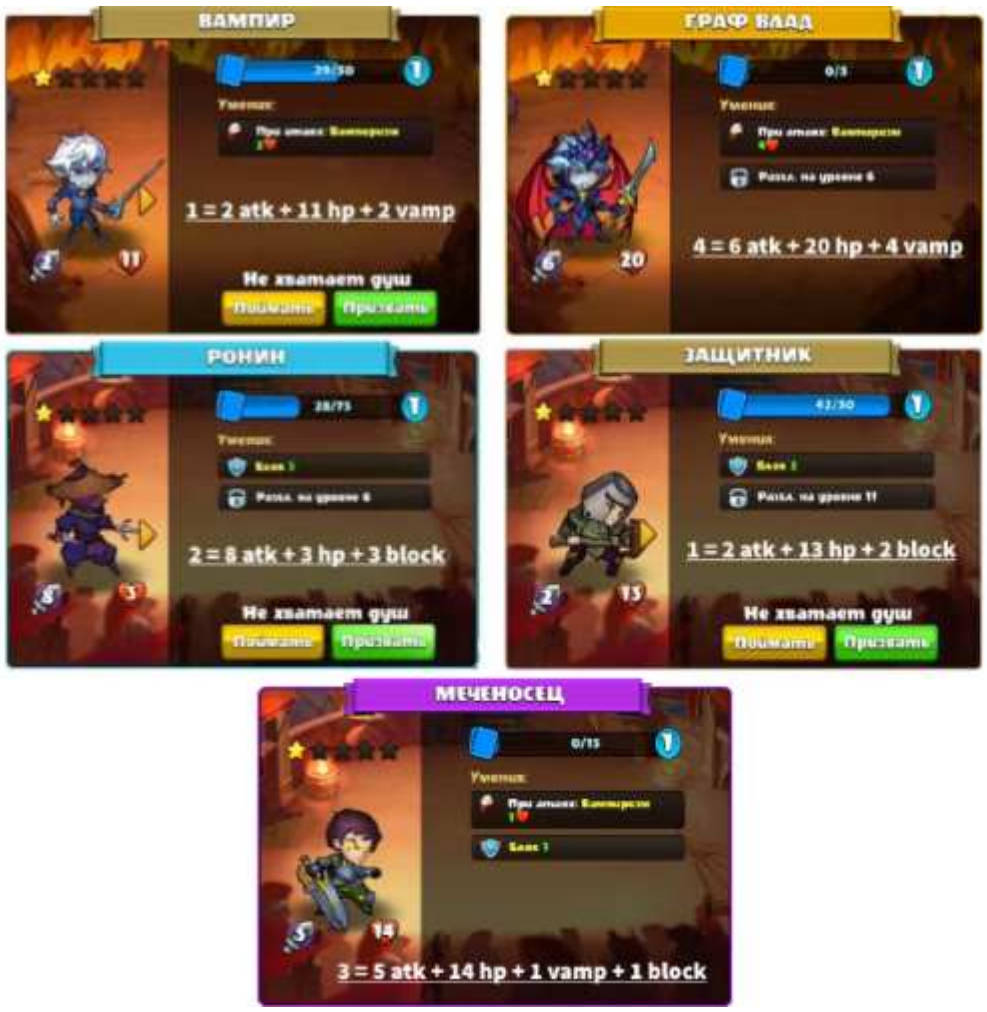

Рис. 13: Некоторые существа с "Блоком" и "Вампиризмом" и составленные для них уравнения.

существа вычисляется, как сумма оценок его характеристик, удовлетворяя следующим правилам:

- Мощность существа напрямую зависит от ее редкости.
- Мощность существа напрямую зависит от ее уровня.
- Мощность существ возрастает линейно.
- Дополнительный эффект - величина всегда постоянная, то есть, к примеру, "Божественный щит" или "Немота" вычисляются всегда одинаково, независимо от существа, редкости и прочих параметров.

Используя вышеописанную модель, рассмотрим более сложный пример: 5 существ разной редкости, обладающие способностями ”Блок” и ”Вампиризм” (рис. 13).

Rarity	Attack	Health Power	Vampire	Block	a
1	2	11	2	0	1
4	6	20	4	0	1
2	8	3	0	3	1
1	2	13	0	2	1
3	5	14	1	1	1

atk = 0.5505
hp = 0.1743
vamp = -0.3853
block = -0.5596
a = -1.2477

Рис. 14: Оценка некоторых способностей существ.

Применив метод наименьших квадратов для полученной системы уравнений, получаем коэффициенты (рис. 14), что означает, что теперь можно оценивать или сравнивать карты с ”Вампиризмом” и ”Блоком”.

Рис. 15: Боец дальнего боя ”Афина”.

Стоит отметить, что пример показывает только принцип работы механизма, так как для работы с реальной системой пяти карт недостаточ-

но. В частности, исследование проводилось на 135 картах уникальных существ. Как можно заметить эта модель несложная, и отлично применима в случае оценивания простых существ, но в игре также присутствуют и ”непростые” существа, способности которых тесно связаны с существами, находящимися в поле или которые только будут разыгрываться и которые оценивать посредством такой схемы было бы некорректно.

Athena Size	Rarity Value
0 / 0	2.83
5 / 3	3.57
10 / 6	4.12
15 / 9	5.97
20 / 12	7.59
25 / 15	9.79

Рис. 16: Разные случаи использования ”Афины”.

Например, в случае с легендарной ”Афиной” (рис. 15), сила которой зависит от нескольких факторов: положения существа на поле, количество окружающих существ и здоровья противника, нужно строить модель, описанную выше, для всех возможных случаев призыва.

Рис. 17: Статистика сыгранных ”Афины” в реальной жизни.

На рис. 16 представлены результаты и видно, что только в случаях,

когда 3 и больше клеток вокруг "Афины" свободны, существо сыграно эффективно. Примененные коэффициенты рассчитаны для 135 существ с около 25 разных способностей.

В реальной жизни это одна из самых популярных карт, 75% топовых игроков Mighty Party имеют "Афину" в своих колодах, и это вполне оправдано. Собрав статистику (рис. 17) около 1500 боев, в которых эта карта была выставлена на поле, можно определить "реальную" редкость (мощность) этого существа и оценить его. Получили мощность этой карты, в среднем равную 5.1, что говорит нам о том, что использование "Афины" в битве в основном эффективно, поэтому значительная часть игроков судя по форумам всячески пытается получить эту карту.

На рисунке 18 показан график с недооцененными (синие точки) и переоцененными (красные точки) юнитами, созданными командой геймдизайнеров. Можно заметить, что "Афина" не единственная недооцененная карта. Немало в игре и переоцененных героев, редкость которых геймдизайнерам "Mighty Party" можно было бы снизить.

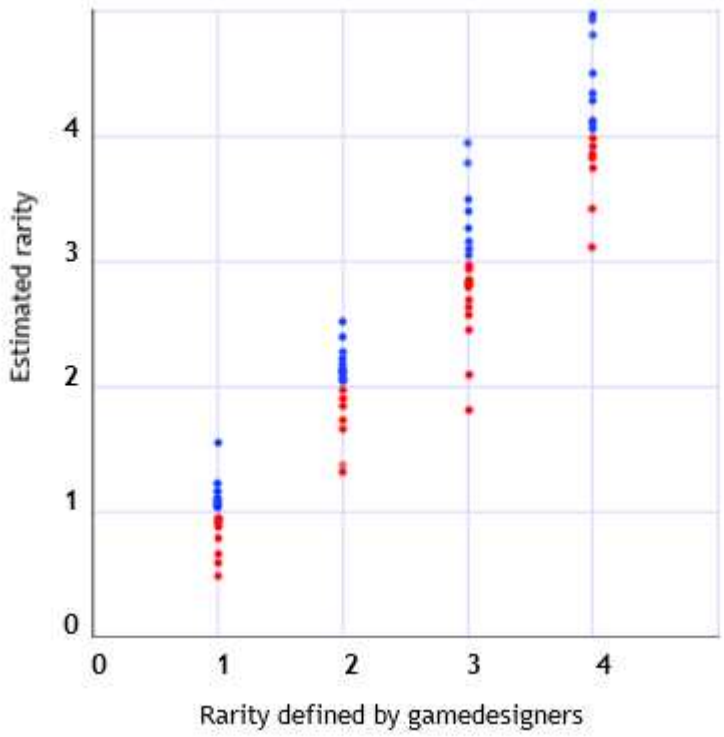

Рис. 18: Результаты оценивания, существующих в игре, карт с помощью получившегося механизма.

Такой же анализ был проведен и для сгенерированных нейросетью карт. Так как построенная нейронная сеть обучалась с помощью этого механизма оценивания карт, результаты (рис. 19) выглядят лучше, так как на графике нет явно выраженных выбросов, недооцененных или переоцененных карт.

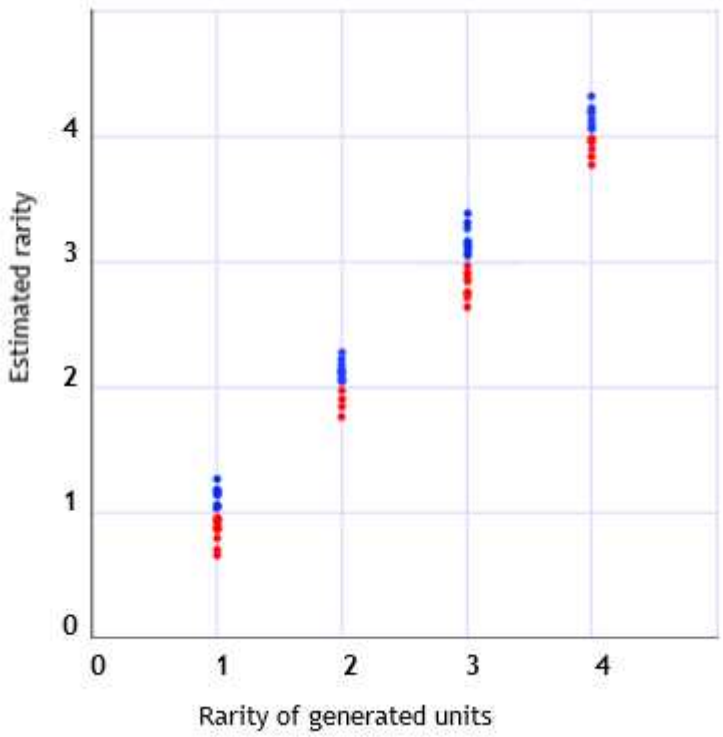

Рис. 19: Результаты оценивания сгенерированных карт.

5. Результаты

В результате исследования была разработана система, включающая в себя глубокую рекуррентную нейронную сеть и набор вспомогательных систем для фильтрации входных и выходных данных. Вспомогательные инструменты играют очень важную роль, так как негенерируемый игровой контент (обучающая выборка) имеет недостаточный размер для корректной работы нейронной сети.

Полученная система удовлетворяет всем вышеописанным требованиям:

- Корректность генерируемого контента: в ходе предварительного анализа входных данных разработана вспомогательная система, позволяющая оценивать корректность генерируемого контента. Данная система полностью исключает вероятность нарушения генерируемым контентом механики и балансом рассматриваемой игры Это в свою очередь позволяет использовать полученную систему полностью автоматически, без участия человека;

- Уникальность и разнообразие: также была разработана вспомогательная система, позволяющая оценивать различие между элементами генерируемого контента. Настройки данной вспомогательной системы позволяют гибко управлять общей уникальностью генерируемого контента;

- Адаптивность и гибкость: данное требование выполнено благодаря реализации на основе искусственной нейронной сети, так нейросети могут адаптировать свои синаптические веса к изменениям окружающей среды (например, добавление нового элемента игровой механики). Таким образом, при изменении условий данная система может легко переучиваться или “доучиваться”.

Список литературы

[1] A. Doull E. Key M. Toy R. Evans T. Adams. Game-designer interviews. — 2014.

[2] A. M. Smith M. Mateas. Answer set programming for procedural content generation: A design space approach. — 2011.

[3] Boyer B. My generation: How Indie Game Makers are Embracing Controlled Chaos. — 2009.

[4] Doull A. The Death Of The Level Designer: Procedural Content Generation in Games. — 2008.

[5] Entertainment Blizzard. 70 Million Players! — 2017.

[6] Games Hello. No Man's Sky. — 2016.

[7] I.Sutskever J.Martens G.Hinton. Generating Text with Recurrent Neural Networks. — 2011.

[8] J. M. Font T. Mahlmann D. Manrique J. Togelius. A Card Game Description Language. — 2013.

[9] J.Ba V.Mnih K.Kavukcuoglu. Multiple Object Recognition with Visual Attention. — 2014.

[10] J.Martens I.Sutskever. Learning Recurrent Neural Networks with Hessian-Free Optimization. — 2011.

[11] J.Schmidhuber. Deep learning in neural networks: An overview. — Elsevier, 2014.

[12] J.Schmidhuber J.Togelius. An Experiment in Automatic Game Design. — 2008.

[13] J.Togelius G.N.Yannakakis K.O.Stanley C.Browne. Search-based Procedural Content Generation: A Taxonomy and Survey. — 2011.

[14] K. Gregor I. Danihelka A. Graves D. J. Rezende D. Wierstra. DRAW: A Recurrent Neural Network For Image Generation. — 2015.

[15] M. Shaker N. Shaker J. Togelius. Ropossum: An Authoring Tool for Designing, Optimizing and Solving Cut the Rope Levels. — 2013.

[16] M.Guzdial M.Riedl. Game Level Generation from Gameplay Videos. — Elsevier, 2014.

[17] M.J.Nelson N.Shaker J.Togelius. Procedural Content Generation in Games: A Textbook and an Overview of Current Research. — 2016.

[18] Magic: The Gathering. — W. of the Coast, 1993.

[19] McFerran D. More than a million super mario maker levels have been uploaded in a week. — 2015.

[20] Moss R. Seven uses of procedural generation that all developers should study. — 2016.

[21] P.Werbos. Backpropagation through time: What it is and how to do it. — 1990.

[22] Rossignol J. Indie Game Developers Enlist Algorithms to Do the World-Building for Them. — 2008.

[23] S.Hochreiter J.Schmidhuber. Long short-term memory. — 1997.

[24] Software Introversion. Procedural Content Generation. — 2008.

[25] T.Coleman. Opinion: Procedural Generation Cost Analysis. — 2011.

[26] V. Hom J. Marks. Automatic Design of Balanced Board Games. — 2007.

[27] Обзор игры Mighty Party. — MMO13, 2017.

Примечание

Несколько карт, разработанных геймдизайнерами (из начальной выборки):

- Beholder
  End Turn: Summon Tree 0/1 in front of it
  Appear: +1 Attack to random Chaos ally (unlocks on lvl 6)

- Chiropteran
  Vampire 2 HP
  Attack: Steal 1 Attack

- Wood Spirit
  Start Turn: Buff +1 HP to all allies

- Mr. Toad
  Appear: Give Block 2 to random ally
  Return 2 Damage

- Driada
  Melee allies +1 HP
  Nature allies +1 HP (unlocks on lvl 11)

- Cat Lady
  Heal +6 HP to your Warlord

Особенности сгенерированных карт, нарушающие игровой баланс:

- Red Death
  Start Turn: Summon Tree 0/0 behind front of it

- Void
  Mental Shield
  Vampire 13 HP

- Son of Defender
  Chaos allies: -10Attack
  Give all enemies Double Attack

- Tesla Life
  Pierce 10 Damage

Сгенерированные карты, нарушающие игровую механику:

- Vlad Sir
  Atta +

- Mother Prince
  Attacked: Double
  Block 4

Сгенерированные карты:

- Mother Death
  6 Damage to Melee allies

- Lord Ronin
  Pierce 3 Damage
  Double Attack