Правительство Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Санкт-Петербургский государственный университет»

Кафедра информатики

Дудкин Евгений Владимирович

Создание игры Реверси

Курсовая работа

Научный руководитель:
ст. преподаватель Литвинов Ю.В.

Санкт-Петербург

2015

Оглавление

Введение

В современном мире прикладываются большие силы для создания различных игр, за которыми люди будут проводить свой досуг. В наши дни, когда почти у каждого есть компьютер, с довольно частой периодичностью выпускается множество разнообразных, продвинутых компьютерных игр. Но, как это ни странно, востребованными остаются также логические и настольные мини-игры. Целью нашей работы было реализовать одну из таких игр - реверси.

В игре реверси используется квадратная доска размером 8 × 8 клеток (все клетки могут быть одного цвета) и 64 специальные фишки, окрашенные с разных сторон в контрастные цвета, например, в белый и чёрный. Клетки доски нумеруются от верхнего левого угла: вертикали — латинскими буквами, горизонтали — цифрами. Один из игроков играет белыми, другой — чёрными. Делая ход, игрок ставит фишку на клетку доски «своим» цветом вверх.

В начале игры в центр доски выставляются 4 фишки: чёрные на d5 и e4, белые на d4 и e5.

Первый ход делают чёрные. Далее игроки ходят по очереди.

Делая ход, игрок должен поставить свою фишку на одну из клеток доски таким образом, чтобы между этой поставленной фишкой и одной из имеющихся уже на доске фишек его цвета находился непрерывный ряд фишек соперника, горизонтальный, вертикальный или диагональный (другими словами, чтобы непрерывный ряд фишек соперника оказался «закрыт» фишками игрока с двух сторон). Все фишки соперника, входящие в «закрытый» на этом ходу ряд, переворачиваются на другую сторону (меняют цвет) и переходят к ходившему игроку.

Если в результате одного хода «закрывается» одновременно более одного ряда фишек противника, то переворачиваются все фишки, оказавшиеся на всех «закрытых» рядах.

Игрок вправе выбирать любой из возможных для него ходов. Если игрок имеет возможные ходы, он не может отказаться от хода. Если игрок не имеет допустимых ходов, то ход передаётся сопернику.

Игра прекращается, когда на доску выставлены все фишки или когда ни один из игроков не может сделать хода. По окончании игры проводится подсчёт фишек каждого цвета, и игрок, чьих фишек на доске выставлено больше, объявляется победителем. В случае равенства количества фишек засчитывается ничья.[4]

Поскольку реализаций на компьютере, в отличие от мобильных телефонов, задача реализации реверси для компьютера представляет интерес. Кроме того, игра реверси может служить площадкой для реализации и апробации алгоритмов искусственного интеллекта, что интересно как учебная задача.

Постановка задачи

Целью было реализовать для компьютера игру Реверси.

Чтобы реализовать то, что мы задумали, потребовалось решить следующие задачи:

- провести обзор уже существующих решений
- реализовать ядро игры
- реализовать пользовательский интерфейс
- реализовать искусственный интеллект, обладающий разными уровнями сложности
- провести апробацию полученного результата

1. Обзор

1.1. Существующие решения

Самым популярным проектом, код которого находится в открытом доступе, является программа Reversi Майка Холла. Игра обладает довольно “приятным” пользовательским интерфейсом. Понравилось, что добавлена довольно неплохая анимация при перекрашивании “чужих” фишек. Также в данном приложении ведется подробная статистика игры. Из-за кнопок Undo Move и Redo Move можно возвращаться в любой момент партии. В целом, программа выполнена профессионально и это не удивительно, так как она разрабатывалась около 2 лет.

Но в этом приложении помимо явных плюсов, есть и недостатки. У пользователя есть огромный выбор различных функций. Благодаря этому игрок может настроить игру под себя. Например, если пользователю надоели контрастные цвета фишек (белый,черный), то он может выбрать другие. То же самое касается цвета доски. Но все эти функции находятся в панели Menu, вверху окна игры. Из-за большого количества такого рода функций затрудняется поиск стандартных процедур. То есть потребуется время, чтобы найти кнопки, отвечающие за сложность игры или право первого хода. Также программа очень долго “думает” на уровне сложности Expert.

Еще один проект, который находится в открытом доступе, - Reversi, права на который принадлежат Ness-2009. Игра обладает поиском оптимального хода, реализованного с помощью алгоритма альфа-бета отсечения. Но в то же время, пользовательский интерфейс довольно прост (рис. 1). Также пользователю не предоставлен выбор цвета. Помимо этого, был обнаружен очень существенный недочет. Приложение никак не реагирует, когда у одного из игроков заканчиваются ходы. Это правило реверси создатели явно упустили.

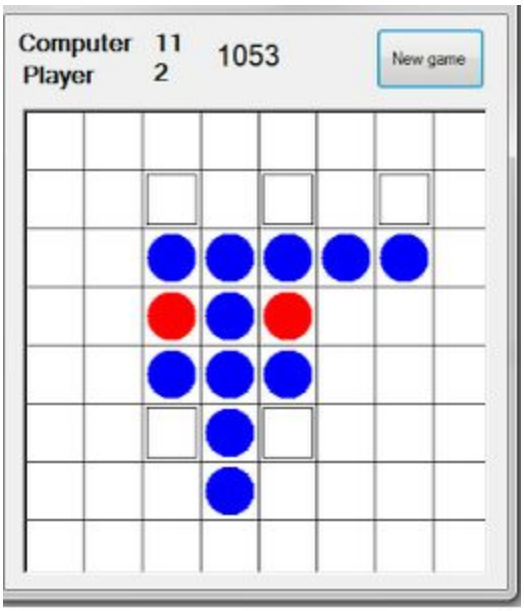

рис. 1

Кроме представленных есть и другие реализации этой игры. Но они не находятся в открытом доступе.

1.2. Методы поиска наилучшего хода

Если мы хотим наделить наше приложение искусственным интеллектом, то нам нужно “научить” компьютер делать только правильные ходы. Для этого потребовалось разобраться с алгоритмами поиска наилучшего хода. В основном в программах такого рода, где можно построить дерево всевозможных вариантов развития партии, используются алгоритм минимакс, алгоритм альфа-бета отсечения и эвристика.

Минимакс (рис. 2) - правило принятия решений, используемое в теории игр для минимизации возможных потерь из тех, которые лицу, принимающему решение, нельзя предотвратить при развитии событий по наихудшему для него сценарию [2]. Алгоритм минимакс обеспечивает способ смотреть на несколько ходов вперед, путем максимизации количества фишек на доске искусственного интеллекта и минимизации фишек игрока (рис. 2).

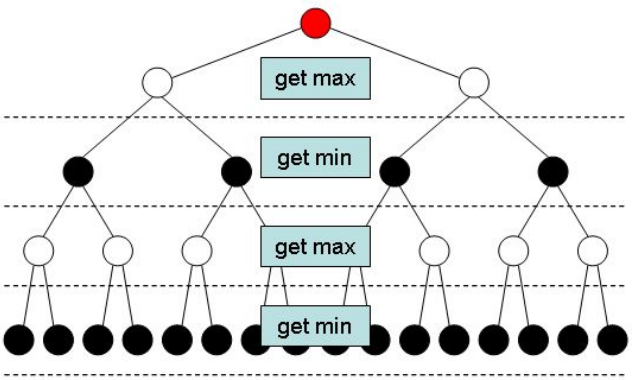

рис. 2

Альфа-бета отсечение - алгоритм поиска, стремящийся сократить количество узлов, оцениваемых в дереве поиска алгоритмом минимакса. В основе алгоритма лежит идея, что оценивание ветви дерева поиска может быть досрочно прекращено, если было найдено, что для этой ветви значение оценивающей функции в любом случае хуже, чем вычисленное для предыдущей ветви [1]. Преимущество данного алгоритма заключается в том, что некоторые из ветвей подуровней дерева поиска могут быть исключены после того, как хотя бы одна из ветвей уровня рассмотрена полностью. Так как отсечения происходят на каждом уровне вложенности, эффект может быть весьма значительным [3].

Для успешной игры нашего искусственного интеллекта нужно также учитывать некоторые эвристики игры реверси. Регионы 3 и 5 (рис. 3) желанные позиции на игровой доске реверси. Если занять эти поля(особенно регион 5), то положение противника сразу становится затруднительным. Эти два региона позволят легко завладеть большим количеством фишек противника сразу. Таким образом, наш искусственный интеллект должен занимать эти позиции, если это возможно.

По причинам перечисленным выше регионы 2 и 4 автоматически являются рискованными, потому что противник получает доступ к регионам 3 и 5. Поэтому не рекомендуется занимать эти две области.[5]

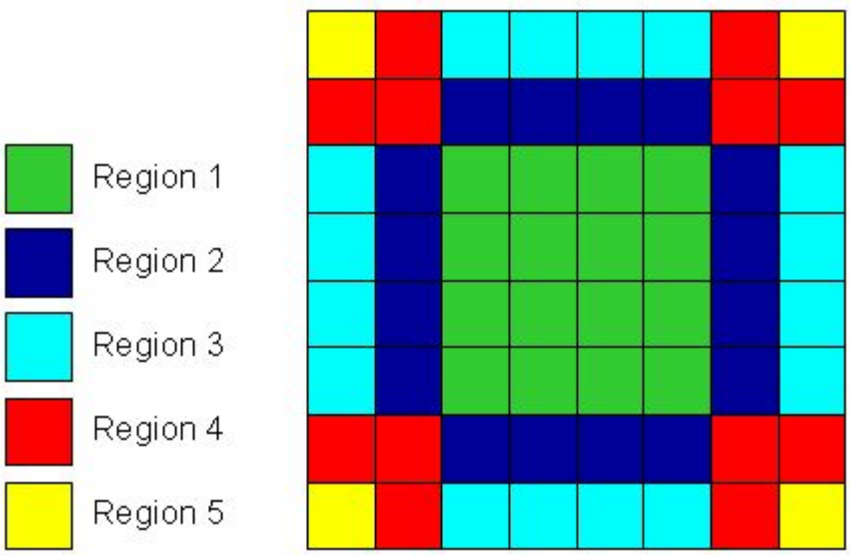

рис. 3

2. Описание решения

2.1. Архитектура

Ниже (рис. 4) представлена диаграмма классов нашей программы. По данной схеме можно получить представление об архитектуре приложения, реализующего игру реверси.

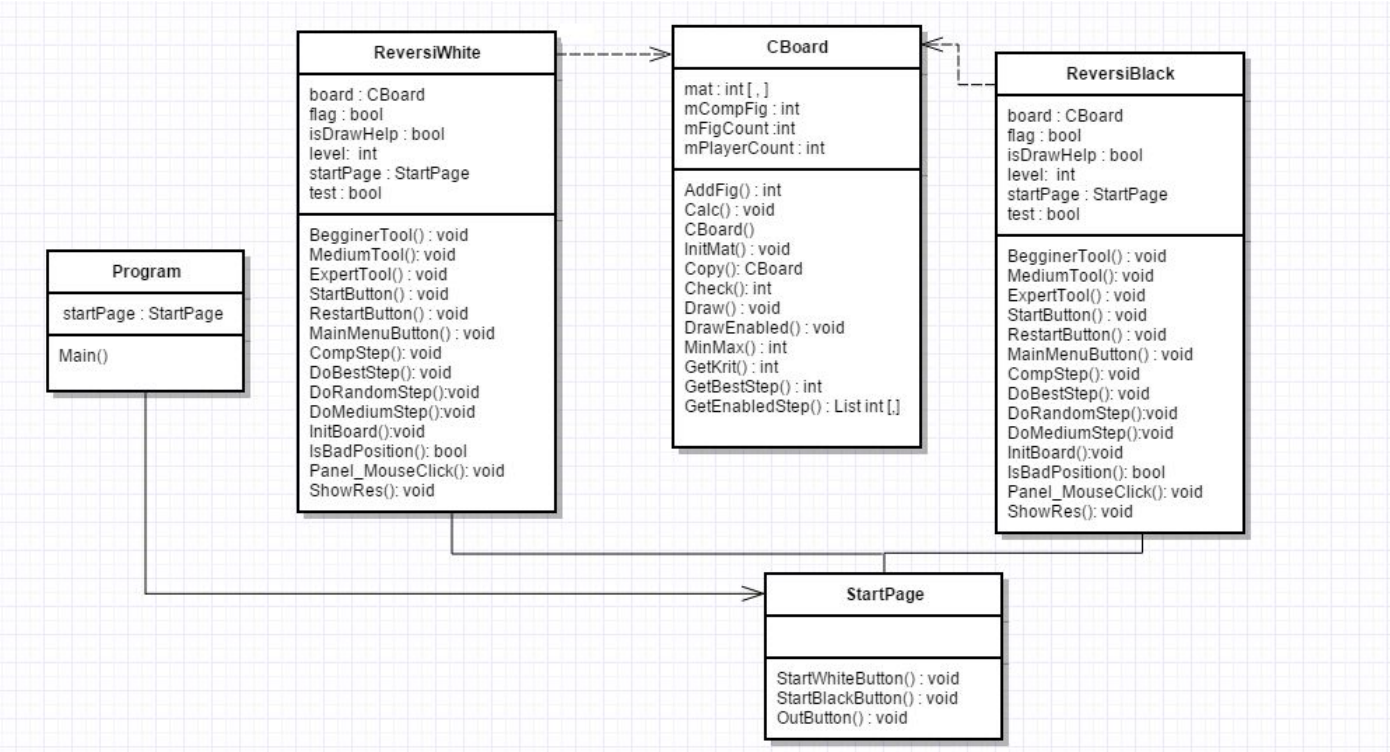

рис. 4

При запуске приложения появляется окно StartPage.Именно в этом окне игрок делает выбор своего цвета. Из него мы можем попасть в окна, где уже реализована сама игра.

2.2. Реализация

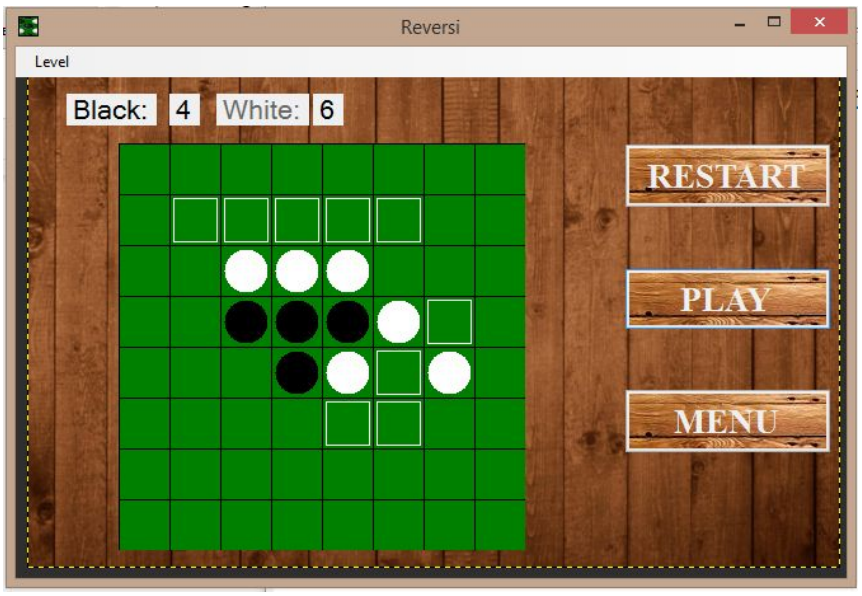

рис. 5

Выше(рис. 5) представлен скриншот нашей игры. Самым сложным, на мой взгляд, оказалось реализовать игровую панель. Необходимо было как-то связать доску, с которой взаимодействует пользователь, с ядром программы (в частности с матрицей этой доски). В итоге, игровая панель заполняется 64 квадратами. При нажатии пользователя на один из таких квадратов, чтобы осуществить свой ход функция Panel_MouseClick принимает координаты точки нажатия и генерирует уже координаты нашего маленького квадрата (делением координат x и y на длину стороны маленького квадрата). Получив эти координаты мы можем занести их в матрицу игры (получили строку и столбец матрицы) и работать с таким абстрактным представлением доски.

Также пользователь может выбрать уровень сложности, который ему больше всего подходит. Это делается при помощи кнопки Level. Нами было реализовано 3 уровня сложности: новичок, средний и эксперт. Новичок делает случайный ход из всех возможных. Средний же своим ходом перекрашивает максимальное число фишек противника. Про реализацию уровня сложности эксперт будет сказано ниже.

2.3. Реализация компьютерного оппонента

Самый умный бот был создан при помощи алгоритма минимакса, описанного выше. Была реализована функция GetBestStep, которая возвращает число, оценивающее все возможные ходы компьютера. Это число - результат оценочной функции GetKrit, которая по матрице игры определяет выигрышность позиции относительно компьютера. Чем это число больше, тем ход лучше. Для хода выбирается, конечно же, самый лучший. Так как доказано, что человек не может смотреть на 3 хода вперед, то глубина перебора ограничена именно таким количеством ходов.

Также были учтены эвристики игры. Мы научили компьютерного оппонента избегать нежелательных позиций на доске с помощью функции IsBad.

3. Апробация

Для оценки результатов были проведены эксперименты, позволяющие узнать уровень приложения. В первую очередь были проведены игры нашего искусственного интеллекта с человеком, который до этого момента не имел опыта в игре реверси. “Эксперту” он проиграл с разгромным счетом, а вот “Среднего” с трудом, но победил. Следующим тестирование приложения проходил человек, который умеет играть в реверси. Этот игрок довольно часто практикуется и тренируется со своими друзьями. Конечно же неудивительно, что “Средний” уровень сложности показался ему довольно простым. “Эксперта” же он обыгрывал только в 50% случаев. По словам игрока, компьютер действует достаточно по-умному, избегает ловушек и при первой же возможности занимает углы. В целом, игроки, которые участвовали в эксперименте остались довольны игрой, выделили интерфейс, который показался им нестандартным и приятным.

Следующим этапом стала игра между нашим приложением и искусственным интеллектом на телефоне. Соревновались боты на своих максимальных уровнях сложности. Было видно, что на телефоне реализован точно такой же алгоритм поиска оптимального хода. Боты старались не занимать слабых позиций. Но наш компьютерный оппонент первый допустил ошибку, так как других ходов у него не осталось. В итоге, телефон легко довел партию до конца и стал победителем. При смене цветов история повторилась с точностью до наоборот. Победу одержал уже компьютер.

Заключение

В итоге, была создана игра реверси, которая обладает тремя уровнями сложности и, по мнению опрошенных пользователей, хорошим и удобным интерфейсом. Также был реализован искусственный интеллект, который был протестирован. Результаты оказались довольно хорошими, так как наш бот побеждает даже у опытных игроков и наравне борется с уже существующими приложениями.

Таким образом, были решены все задачи, которые ставились перед нами в самом начале работы.

Список литературы

[1] URL: https://ru.wikipedia.org/wiki/Альфа-бета-отсечение/ дата обращения 20.09.2015

[2] URL: https://ru.wikipedia.org/wiki/Минимакс/ дата обращения 19.09.2015

[3] Стюарт Рассел, Питер Норвиг — Искусственный интеллект. Современный подход [2-е издание, 2007, Вильямс], с. 241, 266

[4] URL: https://ru.wikipedia.org/wiki/Реверси/ дата обращения 19.09.2015

[5] URL: http://mnemstudio.org/ дата обращения 19.09.2015