

Государственное бюджетное общеобразовательное учреждение
гимназия № 73 «Ломоносовская гимназия» Выборгского района Санкт-Петербурга

Стратегии принятия решений в антагонистических играх на примере игры калах

Работу выполнил: Борович Андрей Алексеевич, 6.3 класс;

Руководитель: Серов Юрий Михайлович

Санкт-Петербург 2015 год

Оглавление

Оглавление	2
Введение.....	3
Цели работы	4
Задачи работы.....	4
Актуальность темы	4
Анализ литературы и источников	4
Антагонистические игры.....	5
История игры	6
Правила игры калах	7
Среда разработки программы.....	8
Интерфейс программы.....	8
Описание модулей программы.....	8
Оценка позиции	11
Версии программы.....	13
Статистика.....	14
Заключение.....	15
Приложения	17
Приложение 1. Библиография	17
Приложение 2. Иллюстрации	18
Приложение 3. Листинг программы.....	21

Введение

В ходе предыдущей научной работы я познакомился с шифрами эпохи Возрождения и написал программу, реализующую алгоритм шифрования Виженера. Это пробудило во мне интерес к программированию, и я решил написать программу для какой-нибудь настольной игры. Шахматы на тот момент показались мне слишком сложной задачей. Тогда же я нашёл описание игры калах в книге Чарльза Уэзерела «Этюды для программистов»¹.

В книге вкратце описан алгоритм поиска верного хода, правила игры в калах и дано задание по написанию программы. На первом этапе я создал программу для игры человека с человеком. Такая программа умела делать ходы в соответствии с правилами игры, но не умела выбирать наилучший собственный ход. На следующем этапе я наделил программу «искусственным интеллектом», чтобы она могла играть против человека и выбирать наилучший ход из возможных.

Очевидно, что игра калах не такая сложная, как шахматы. В то же время калах сильно отличается от крестиков-ноликов разнообразием в правилах и количеством игровых позиций. Все эти игры (шахматы, калах, крестики-нолики) – антагонистические², и алгоритмы поиска лучшего хода в них очень похожи, хотя внешне они сильно отличаются. Так появилась идея научной работы, в которой будут рассмотрены разные варианты принятия решения при выборе хода.

¹ Чарльз Уэзерел «Этюды для программистов», изд. Мир, 1982.

² Антагонизм – от древнегреческих слов «противник» и «борьба».

Цели работы

1. Рассмотреть алгоритмы принятия решений в антагонистических играх: Minimax, Negamax.
2. Создать информационную модель, позволяющую смоделировать и реализовать процесс принятия решения в игре калах.

Задачи работы

1. Познакомиться с игрой калах.
2. Познакомиться с методами программирования антагонистических игр.
3. Написать программу для игры в калах, реализующую алгоритм поиска лучшего хода.
4. Выяснить, какие оценочные функции являются наиболее эффективными.
5. Собрать статистику по игре.

Актуальность темы

Навыки, полученные при создании данной программы, можно применить в дальнейшем, как для программирования более сложных игр, так и для решения других задач. Алгоритмы поиска лучшего хода тесно связаны со стратегией принятия решений в экономике, и во второй половине прошлого века теория игр стала важным инструментом моделирования для решения экономических задач.³

Анализ литературы и источников

³ Васин А.А., Морозов В.В. «Теория игр и модели математической экономики». – МАКС Пресс, 2005

В ходе моей работы главными источниками информации были: книга Е.Н. Корнилова «Программирование шахмат и других логических игр»⁴ и книга Ч. Уэзерела «Этюды для программистов».

Книга Уэзерела послужила толчком к началу работы и дала основную идею. Книга Корнилова помогла мне с созданием «искусственного интеллекта». Кроме того, я ознакомился с целым рядом статей по данной теме. Все они перечислены в Приложении «Библиография».

Антагонистические игры

Антагонистической игрой в теории игр называется игра, в которой участвуют два игрока, выигрыши которых противоположны. Если один игрок выигрывает в какой-либо позиции, то другой игрок проигрывает. Из-за этого антагонистические игры называют играми с нулевой суммой. Это существенно отличает антагонистические игры от других игр.

Неантагонистические игры иногда называют играми с непротивоположными интересами. Такие игры бывают кооперативными и некооперативными.

Сложность антагонистической игры определяется числом терминальных (конечных) позиций⁵, на которых заканчивается игра.

«Крестики-нолики» – простой пример антагонистической игры, в ней не более 512 терминальных позиций. После каждого хода на игровом поле возникает новая позиция. Последовательно перебирая возможные ходы, можно построить график развития для всех партий. Такой график называют деревом принятия решений.⁶ Полное число узлов в дереве принятия решений для игры крестики-нолики составляет $9! = 326880$.

Шахматы и го – наверное, самые сложные варианты антагонистических игр.

В шахматах число терминальных позиций оценивается числом 10^{43} , а число уникальных партий $\sim 10^{118}$ (так называемое «Число Шеннона»)⁷. В игре го оценка для числа терминальных позиций составляет 10^{171} .

⁴ Корнилов Е.Н. «Программирование шахмат и других логических задач». – БХВ-Петербург, 2005.

⁵ Статья Некооперативная игра (https://ru.wikipedia.org/wiki/Некооперативная_игра)

⁶ Горьков Алексей «Об ИИ в интеллектуальных играх» (<http://habrahabr.ru/post/143552/>)

⁷ Статья Число Шеннона (https://ru.wikipedia.org/wiki/Число_Шеннона)

Калах же, как уже было сказано, находится в промежуточном положении по сложности – для него количество терминальных позиций можно оценить, как 10^{19} , а количество узлов в дереве игры $\sim 10^{27}$ (в описанном ниже варианте правил).

История игры

Калах – африканская настольная логическая игра семейства манкала.⁸

Манкала (искажённое арабское *maqala* – «перемещение») – это большая семья игр с перекладыванием камней, её история насчитывает от пяти до семи тысяч лет. Находки археологов свидетельствуют о том, что калах по возрасту превосходит и шахматы, и домино, и маджонг.

Манкала была популярна у кочевых народов благодаря своей простоте: для неё даже не обязательно иметь доску и фишки, достаточно сделать в земле лунки, взять горсть камней или кофейных зёрен – и можно играть. До сих пор подобные игры сохранились в традициях народов Африки и Азии.

Доски для игры в калах находили в различных районах Азии и Африки – вырезанные из дерева, вылепленные из глины, высеченные из камня или слоновой кости. Изображения поля для игры в калах встречаются на египетских саркофагах, на ступенях афинских храмов, на стелах у обочин караванных путей (см. Приложение 3 «Иллюстрации»).

Считается, что игра калах является не только развлечением, но и содержит память народов: «движение по кругу символизирует цикличность года и смену сезонов, раскладывание камешков – одновременно посев зерна и сбор урожая, а пустые лунки – голод и недород. Здесь же прячутся древние гадательные ритуалы и самые первые счётные доски»⁹.

⁸ Статья Манкала (<https://ru.wikipedia.org/wiki/Манкала>)

⁹ Дмитрий Скирюк, Калах: древнейшая игра в камешки (http://letidor.ru/article/kalakh_drevneyshaya_igra_v_kam_24400)

В средние века калах был почти забыт, но в конце XIX века о нём вспомнили. В 1893 году на Чикагской выставке появилось описание игры, «в которую когда-то играла половина человечества», и оттуда попало в многочисленные журналы. В 1905 году с этой игрой познакомился Вильям Чемпион. Именно он, будучи уже стариком, в 1940 году, наладил производство наборов для игры в калах. Благодаря ему, игра стала популярной в Америке.

Правила игры калах

Поле для игры представляет доску, состоящую из двух рядов лунок, по 6 лунок в каждом ряду. Принадлежащие игроку лунки находятся в ближнем к нему ряду. Другой ряд принадлежит сопернику. Помимо этого, у каждого игрока есть особенная большая лунка – калах, – находящаяся справа от его лунок.

В начале игры в лунки помещаются камни (фишки). В каждую лунку изначально кладётся 6 камней.¹⁰

Игроки ходят по очереди. В свой ход игрок берёт все камни из одной своей лунки и раскладывает эти камни в последующие лунки против часовой стрелки: сначала слева-направо в своём ряду, затем в свой калах, затем справа-налево в лунки соперника. При раскладывании пропускается калах соперника и снова слева-направо в свои лунки. В каждую лунку он кладет ровно по одному камню до тех пор, пока не разложит все взятые камни.

Если последний (из взятых) камень игрок кладёт в свой калах, то он ходит ещё раз. В других случаях ход переходит к сопернику.

Если последний (из взятых) камень игрок кладёт в свою пустую лунку (не в калах) и противоположная лунка (в ряду соперника) не пуста, то игрок

¹⁰ Есть варианты игры, когда в каждую лунку изначально кладётся другое число камней.

«захватывает» (забирает) все камни из лунки соперника и вместе со своим камнем, который положил последним, перекладывает в свой калах.¹¹

Игра заканчивается, когда один из игроков не может походить, так как все его лунки пусты. В таком случае второй игрок переносит все оставшиеся в его лунках камни в свой «калах», и производится подсчет разности камней в «калах» игроков. Тот, кто набрал больше камней, побеждает.

Среда разработки программы

Программа разработана в среде разработки Lazarus.

Lazarus – свободная среда разработки программного обеспечения на языке программирования Object Pascal для компилятора Free Pascal.

Интерфейс программы

Интерфейс программы для игры в калах включает в себя:

- 12 одинаковых кнопок, которые обозначают лунки игроков;
- 2 больших кнопки, которые обозначают калахи игроков;
- кнопка «Новая игра» начинает новую партию;
- слайдер «Сложность», меняет алгоритм поиска лучшего хода и регулирует глубину просчёта дерева решений;
- чек-бокс «Записывать лог-файл» позволяет создавать текстовые файлы, в которых записываются сделанные ходы.

Описание модулей программы

Программу можно условно разделить на несколько частей:

- Обработка действий игрока (пользователя)

¹¹ Есть вариант игры, где игрок забирает свой последний камень, положенный в пустую лунку в своём ряду, даже если противоположная лунка соперника не содержит ни одного камня.

- Процедуры обработки нажатия кнопок-лунок, вызывающие процедуру «button_click», с соответствующим номеру кнопки-луноки параметром
- Процедура «button_click», отвечающая за ход игрока и компьютера содержащая
 - Вызов процедуры «turn_checking» с параметром номера лунки, проверяющей возможен ли ход из этой лунки
 - Если ход возможен, программа вызывает процедуру «turn» с параметром «х» (номером лунки), которая изменяет основной массив «а», описывающий игровую позицию, в соответствии с правилами и переданным номером лунки
 - Вызов процедуры «writing_in_file», записывающей текущую позицию в файл
 - Вызов процедуры «refresh_captions», изменяющей показываемое состояние поля в соответствии с массивом «а»
 - Если не произошло попадания игроком в калах, проводится расчёт наилучшего хода для компьютера функцией «analysis». О ней будет подробнее рассказано далее
 - Далее выполняются схожие с действиями игрока действия компьютера с учётом рассчитанного наилучшего хода
- Виртуальные действия компьютера (программы)
 - Процедура «analysis», проводящая следующие действия
 - В цикле, последовательно перебирающем лунки
 - Если в лунке есть камни, то вызывается функция «virtual_turn», производящая изменение виртуального массива, вначале совпадающего с основным массивом «а», так как если бы мы ходили из лунки соответствующей текущей итерации цикла

- Если при этом в виртуальном массиве не произошло попадания в калах, вызывается функция «rotate», переворачивающая доску (далее и в случае если попадание в калах произошло, выполнение функции будет различаться лишь в знаке рекурсивного вызова, и эта часть будет опущена)
- Если переменная-счётчик «round_counter» больше единицы и в виртуальном массиве не терминальная позиция, то происходит инверсированный рекурсивный вызов процедуры «analysis»
- Если полученное значение меньше, то есть лучше для стороны, за которую функция рассчитывает оценку, чем все полученные ранее, то есть на более ранних итерациях цикла, то переменной «mark_turn1», отвечающей за передачу значения оценки на верхние уровни рекурсии, присваивается полученное значение функции «analysis»
- Если же переменная-счётчик «round_counter» больше единицы, но в виртуальном массиве достигнута терминальная позиция, то переменной «mark_turn1» присваивается значение оценочной функции для виртуального массива
- Если значение переменной-счётчика «round_counter» равно значению «depth», переменной, которая показывает глубину рекурсии, то есть мы находимся на самом верхнем (первом) уровне рекурсии, то номер лунки, для которой значение «analysis» самое лучшее присваивается переменной «best_turn»
- Если же значение переменной-счётчика «round_counter» равно 1, то есть мы находимся на

глубочайшем уровне рекурсии, то переменной «mark_turn1» присваивается значение оценочной функции для виртуального массива

- Функция «virtual_turn», изменяющая передаваемый в качестве параметра массив в соответствии с правилами и переданным номером лунки.
- Функция «virtual_win_checking», проверяющая массив, передаваемый в качестве аргумента, на терминальную позицию
- Функция «evaluate_position», возвращающая значение оценки текущей позиции.

Оценка позиции

Нобелевский лауреат, Клод Шеннон¹², известный американский математик и инженер, занимавшийся программированием игр (в основном шахмат), выделял два типа стратегий поиска лучшего хода:

- Тип А – перебор всех возможных ходов на фиксированную глубину и использование в конце оценочной функции.
- Тип В – только выборочное расширение определённых строк используя накопленные знания игры, чтобы «подрезать» ненужные ветви дерева решений.

Основными алгоритмами для оценки позиции в антагонистических играх являются:

- MiniMax (тип А по Шеннону)
- NegaMax (тип А)
- Alpha-beta отсечение (тип В)
- Метод Монте-Карло (тип В)

¹² Claude Shannon, Programming a Computer for Playing Chess

MiniMax использует две оценочных функции. Одна используется для минимизации оценки (поиска лучшего хода для одного игрока), а другая для максимизации оценки (поиска лучшего хода для другого игрока). Алгоритм NegaMax похож на MiniMax, но использует инвертированную оценку и компактнее в реализации.

Оба этих алгоритма основываются на переборе всех ходов. Чаще всего перебор идёт в функции, рассматривающей все возможные ходы из текущей позиции и рекурсивно¹³ вызывающей саму себя. В конце (на самом глубоком уровне) вызывается оценочная функция для всех позиций, получившихся на нижнем уровне построенного дерева решений. Для калаха наиболее простой оценочной функцией является разность камней в калахах соперников:

$$\text{evaluate} = K_h - K_c.$$

Этот вариант предлагает использовать в своей книге Ч. Уэзерелл. Однако, такая оценочная функция обладает явными недостатками – например, она не учитывает количество камней, находящихся в лунках игроков (в конце игры камни из лунок переносятся в калахи игроков). Чтобы получить более точную оценку позиции, в моей программе используется оценочная функция такого вида:

$$\text{evaluate} = (K_h - K_c) * 100 + N_h - N_c,$$

где K_h – количество камней в калахе игрока; K_c – количество камней в калахе программы; N_h – сумма камней в лунках игрока; N_c – сумма камней в лунках программы.

Изначально в программе был реализован анализ по полным раундам. При этом в функции «search_best_turn» сначала оценивалась позиция для одного игрока (на основе разбора возможных дальнейших позиций для каждого

¹³ Рекурсия - вызов функции из неё же самой, непосредственно (простая рекурсия) или через другие функции (сложная или косвенная рекурсия) (<https://ru.wikipedia.org/wiki/Рекурсия>)

возможного хода) и делался наилучший для него виртуальный ход, а затем проводились аналогичные расчёты для ответов другого игрока. Функция «search_best_turn» рекурсивно вызывала себя заданное число раз и в конце использовала приведённую выше функцию оценки позиции для определения наилучшего хода на самом глубоком уровне рекурсии.

В дальнейшем анализ по полным раундам был заменён на анализ по полураундам, где функция «analysis» рассчитывала лучший ход для одного игрока, виртуально переворачивала доску, и рекурсивно вызывала себя для оценки позиции другого игрока. Это позволило оптимизировать и сократить код программы.

Оценочная функция очень важный элемент программы. В дальнейшем можно попробовать другие её варианты. Например, доктор физико-математических наук Г. С. Цейтин предлагал использовать такую формулу:¹⁴

$$\text{evaluate} = (K_c + 17,3 / (37 - K_c) - 40 / D_c) - (K_h + 17,3 / (37 - K_h) - 40 / D_h) \pm 1,72;$$

где добавлены значения активности поля D_h и D_c (для игрока и программы соответственно). Активность поля принимается равной количеству камней, попадающих в свои лунки при ходе из этой лунки, минус количество камней, попадающих в лунки соперника, плюс 7.

Версии программы

В ходе работы программа претерпела многочисленные изменения, которые были сохранены по версиям.

В первой версии программа годилась только для игры двух человек, так как не умела выбирать ход самостоятельно. В этой версии отсутствовали правила повторного хода и захвата камней. Существовала проверка окончания игры. Позже была реализована примитивная процедура выбора хода – программа

¹⁴ Побожий Ю. «Машины играют в калах» // Наука и жизнь, №12 1971

делала ход из первой своей непустой лунки. Это позволило сыграть первую партию с компьютером.

Затем появилась первая оценочная функция (разность камней в калахах) и функция «search_best_turn» стала выбирать ход осмысленно. В очередной версии программы оценочная функция была улучшена, а функция «search_best_turn» стала вызываться рекурсивно.

Для анализа работы программы была реализована запись в текстовый файл ходов партии. Также добавилась ещё одна процедура имитирующая ход игрока. Теперь компьютер играл с воображаемым соперником, который делал ход из первой своей непустой лунки. Интересно, что направление поиска непустой лунки (слева-направо или справа-налево) оказывает большое влияние на успешность игры.

На следующем этапе обрёл новый вид интерфейс программы. Функция «analysis» заменила функцию «search_best_turn» и теперь программа просчитывает позицию по полураундам. Новая функция перебирает лунки от калаха и проверяет новые позиции «на победу».

Статистика

В тестировании приняли участие 11 игроков.

Игрок	Уровень 1		Уровень 2		Уровень 3	
	Игр	Побед	Игр	Побед	Игр	Побед
АИ	1	1	2	1	7	1
КШ	1	1	2	1	2	0
АБ	1	1	1	1	1	1
ВК	3	3	3	1	3	0
СР	1	0	1	0	1	0

СБ	3	3	1	1	4	0
БА	1	1	2	1	2	1
СК	2	1	1	0	2	1
БР	2	2	2	1	1	0
ЕЕ	1	0	1	1	1	0
ЕИ	1	1	3	0	0	0

На самом сложном уровне из 24 партий программа выиграла 20, то есть доля выигрышей составляет 83%.

	Игры	Победы игроков	Процент побед
Уровень 1	17	14	82%
Уровень 2	19	8	42%
Уровень 3	24	4	17%

Можно говорить о том, что программа является сильным «игроком», обыгрывающим взрослого человека с высшим образованием.

Заключение

В этой работе я рассмотрел стратегии принятия решений в антагонистических играх и познакомился с игрой калах, по которой написал программу. Исполняемый файл программы выложен для скачивания и ознакомления на сайте <http://andrey.borevich.ru>.

На основе изложенной выше информации можно сделать следующие выводы:

1. Калах – довольно простой пример антагонистической игры, но алгоритм, использованный при поиске наилучшего хода, очень похож на алгоритм, используемый в других антагонистических играх, поэтому написание алгоритма для другой игры потребует внесения лишь некоторых изменений в оценочной функции.

2. Практически все простые (то есть опирающиеся только на оценку текущей позиции и не учитывающие ответы соперника) алгоритмы поиска наилучшего хода не дают точности в просчете, достаточной для того, чтобы в большинстве случаев побеждать сильного соперника-человека.
3. Созданная мной программа выигрывала у людей с высшим образованием в 83% случаев. Можно говорить о том, что программа является довольно сильным «игроком», легко обыгрывающим взрослого человека.
4. Большинство алгоритмов поиска лучшего хода, опирающиеся на полный (до определённого уровня) перебор дерева позиций вглубь, не дают хорошего соотношения результата и быстродействия, и требуют для быстрого и эффективного функционирования дополнительных алгоритмов (alpha-beta отсечение¹⁵, метод Монте-Карло и другие). Данные алгоритмы я рассмотрю в следующей работе.

Тема программирования игры калаш кажется мне интересной для дальнейшей разработки, так как имеются большие возможности для улучшения и усовершенствования алгоритма принятия решений, а также на основе приложения для Windows можно разрабатывать приложения для on-line игры и приложения для мобильных устройств.

¹⁵ Данный алгоритм описан в статьях: Donald E. Knuth and Ronald W. Moog «Анализ альфа-бета отсечений» и Дубнер П.Н. «Описание программы, играющей в калаш». – URL: algolist.manual.ru.

Приложения

Приложение 1. Библиография

Богатырев А. «Игра в Калах». – URL: <http://samlib.ru>.

Васин А.А., Морозов В.В. «Теория игр и модели математической экономики». – МАКС Пресс, 2005.

Горьков А. «Об ИИ в интеллектуальных играх». – URL: <http://habrahabr.ru/post/143552>

Дубнер П.Н. «Описание программы, играющей в калах». – URL: algolist.manual.ru.

Корнилов Е.Н. «Программирование шахмат и других логических задач». – БХВ-Петербург, 2005.

Скирюк Д. Калах: древнейшая игра в камешки. – URL: http://letidor.ru/article/kalakh_drevneyshaya_igra_v_kam_24400

Степан TheHorse «Минимакс на примере игры в зайца и волков» – URL: habrahabr.ru/post/146088/.

Побожий Ю. «Машины играют в калах» // Наука и жизнь, №12 1971.

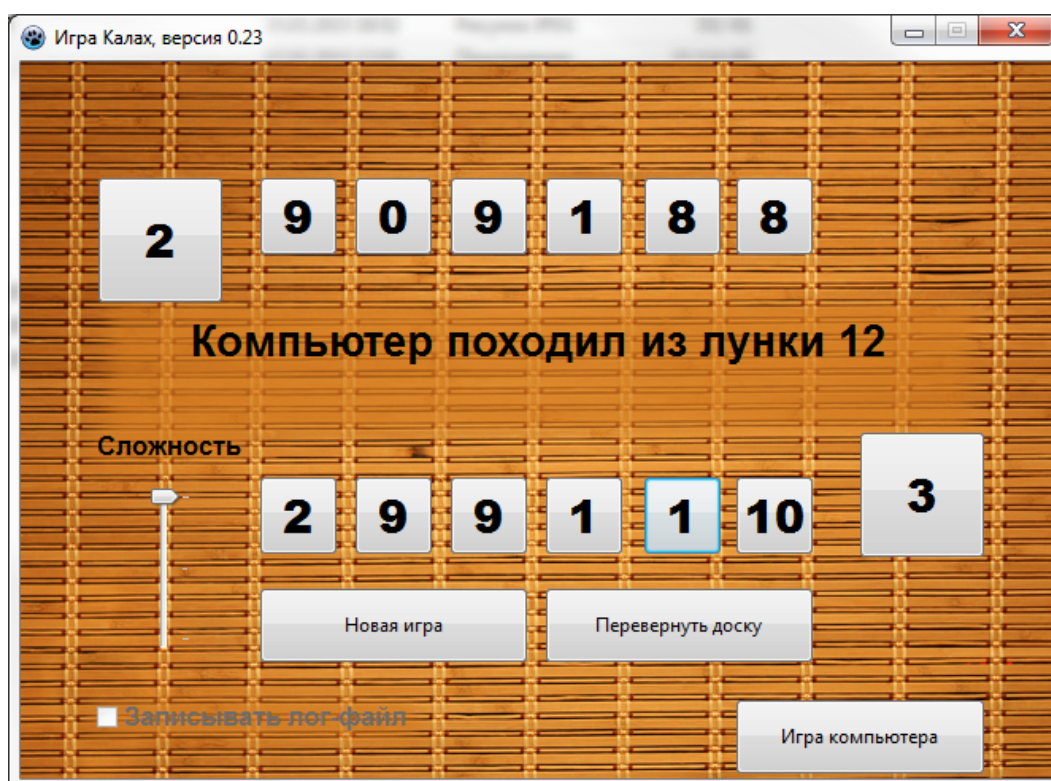
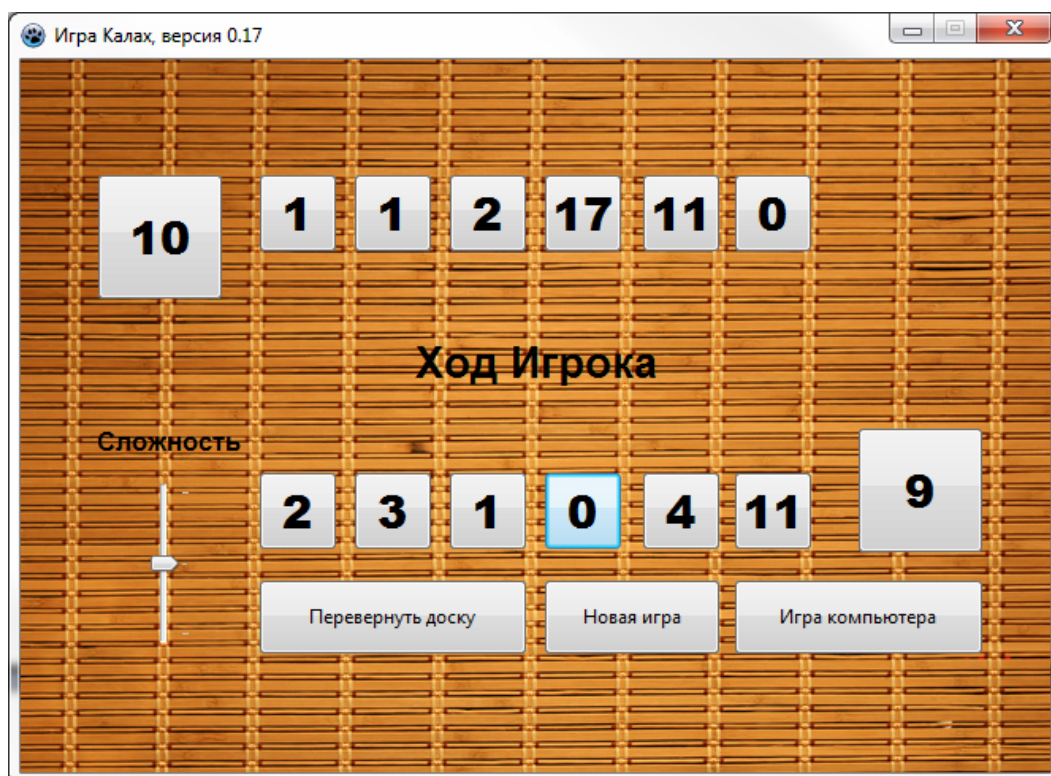
Уэзерелл Ч. «Этюды для программистов». – Мир, 1982.

Статьи Википедии: Калах, Антагонистические игры, Альфа-Бета отсечение, Рекурсия, Математическая экономика, Число Шеннона.

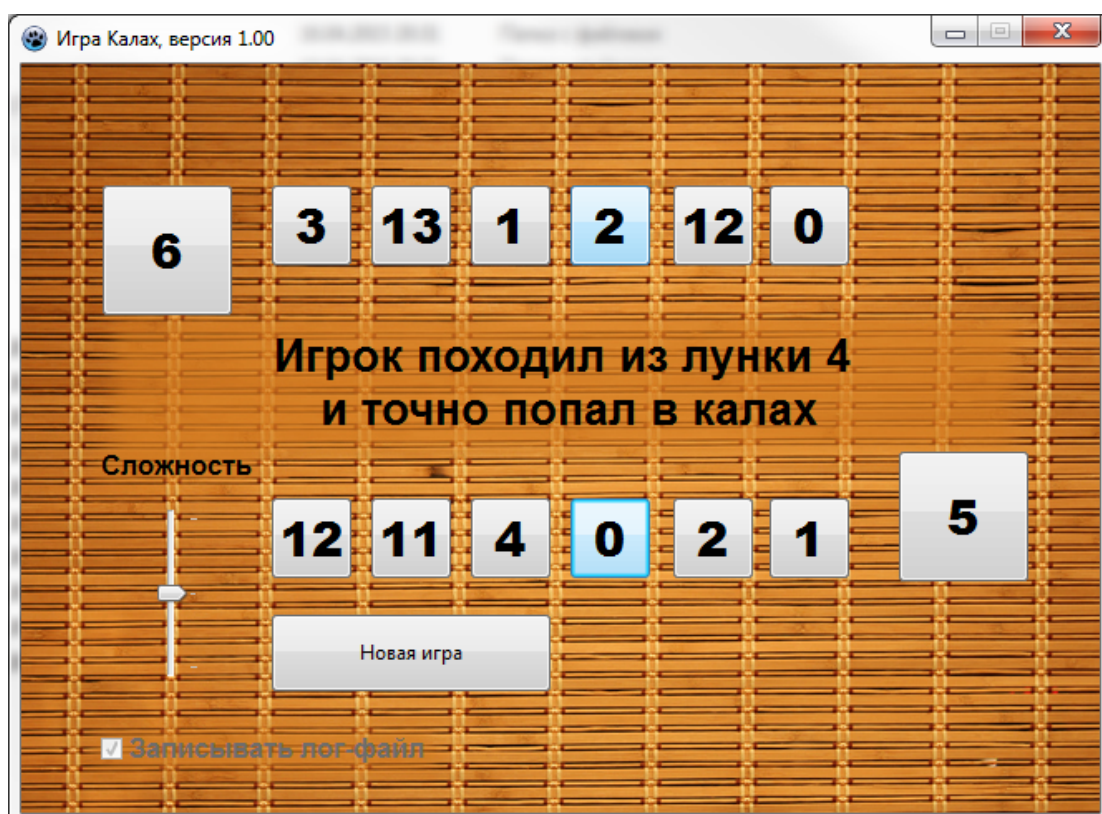
Donald E. Knuth and Ronald W. Moor «Анализ альфа-бета отсечений» (перевёл П.Н. Дубнер). – URL: algolist.manual.ru.

Приложение 2. Иллюстрации

Внешний вид ранних версий программы



Внешний вид программы



Как играют в калах



Доска для игры в калах

