|  |  |
| --- | --- |
|  | МИНОБРНАУКИ РОССИИ  федеральное государственное бюджетное образовательное учреждение  высшего образования  **«Балтийский государственный технический университет «ВОЕНМЕХ» им. Д.Ф. Устинова»**  **(БГТУ «ВОЕНМЕХ» им. Д.Ф. Устинова»)** |
| БГТУ.СМК-Ф-4.2-К5-01 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Факультет |  | И |  | Информационные и управляющие системы |
|  |  | шифр |  | наименование |
| Кафедра |  | И9 |  | Системы управления и компьютерные технологии |
|  |  | шифр |  | наименование |
| Дисциплина |  | Представление знаний в информационных системах | | |

Отчет по практической работе №3

|  |
| --- |
| Поиск на игровых деревьях |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Выполнил студент группы | | | | | |  | И508Б |
| Кабиров К.Р. | | | | | | |
| Фамилия И.О. | | | | | | |
| **РУКОВОДИТЕЛЬ** | | | | | | |
|  | |  | |  | | |
| Фамилия И.О. | | | Подпись | | | |
| Оценка |  | | | | |  |
| «\_\_\_\_\_» |  | | | | | 2023 г. |

САНКТ-ПЕТЕРБУРГ

2023 г.

1. Игровое поле.

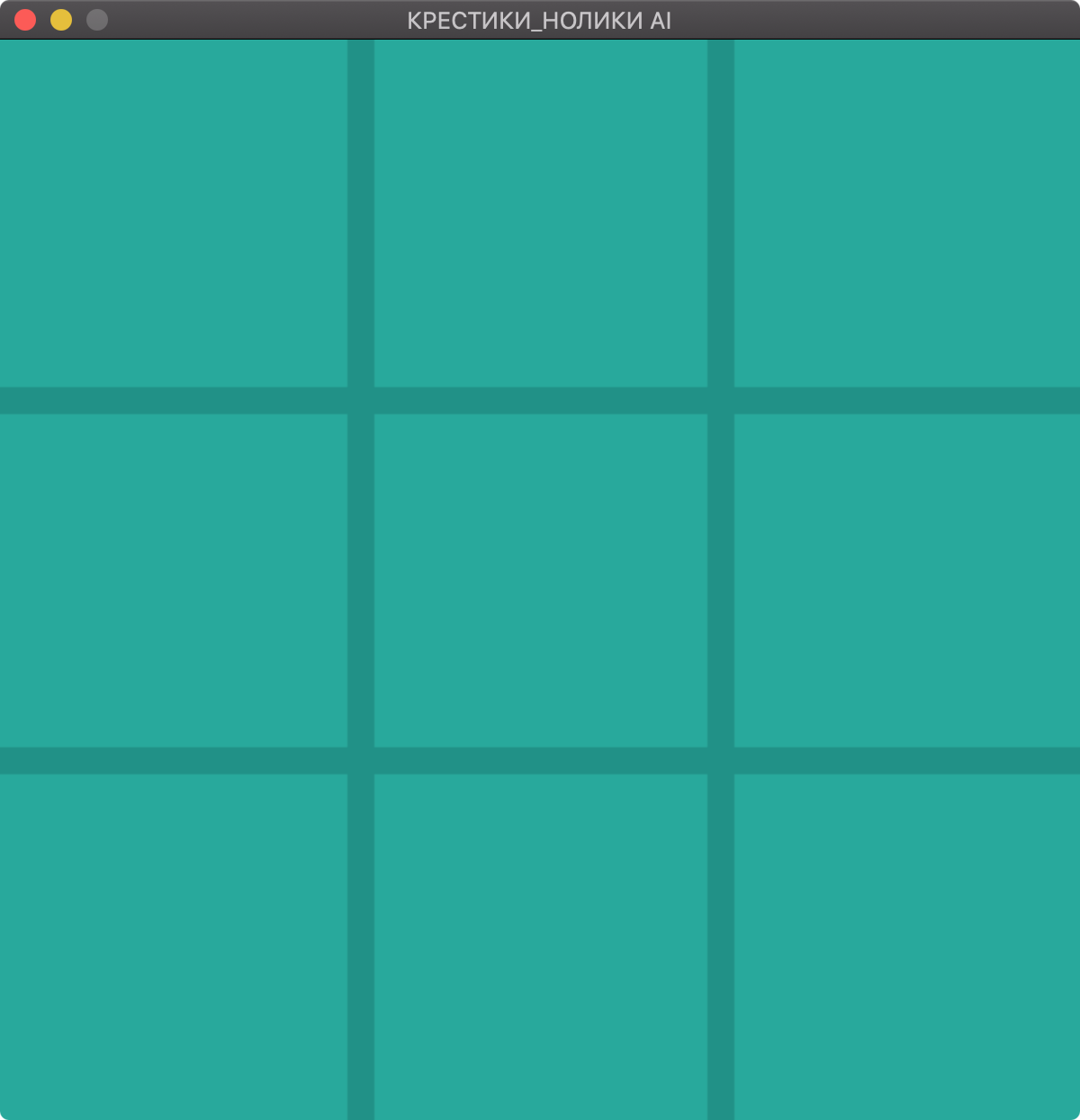


Рисунок 1 – Игровое поле в начале игры

1. Протокол поиска:

Для минимаксного алгоритма:

После хода пользователя Минимаксный алгоритм вычисляет выигрышный ход за 18920 итераций, в этом можно убедиться, посмотрев протокол поиска, представленный на рисунке 2.

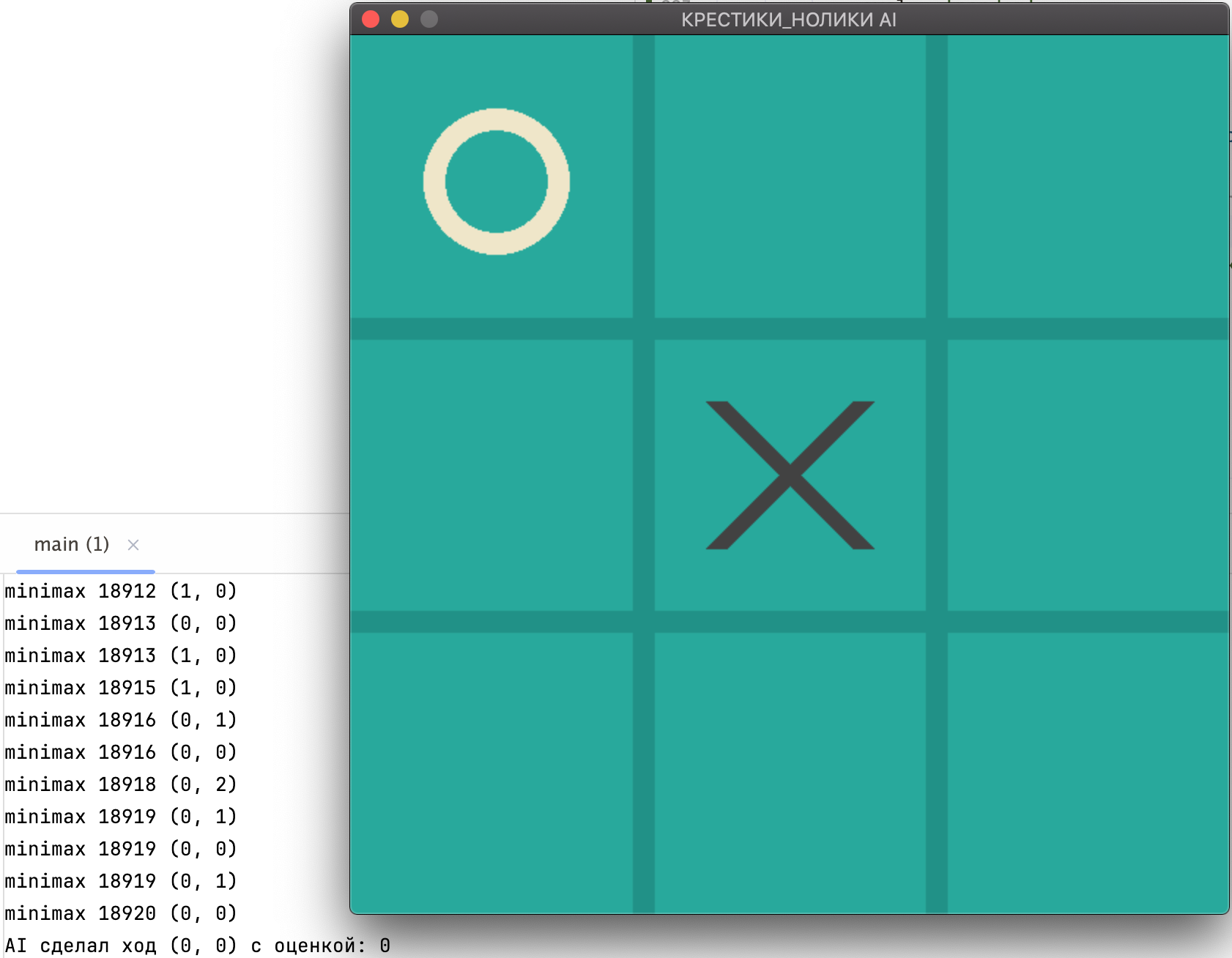


Рисунок 2 – Протокол поиска и ход ИИ (минимаксный алгоритм)

После каждого следующего хода количество итераций будет уменьшаться из-за того, что количество возможных ходов уменьшается. Игровое поле и протокол поиска после нескольких ходов представлены на рисунке 3.

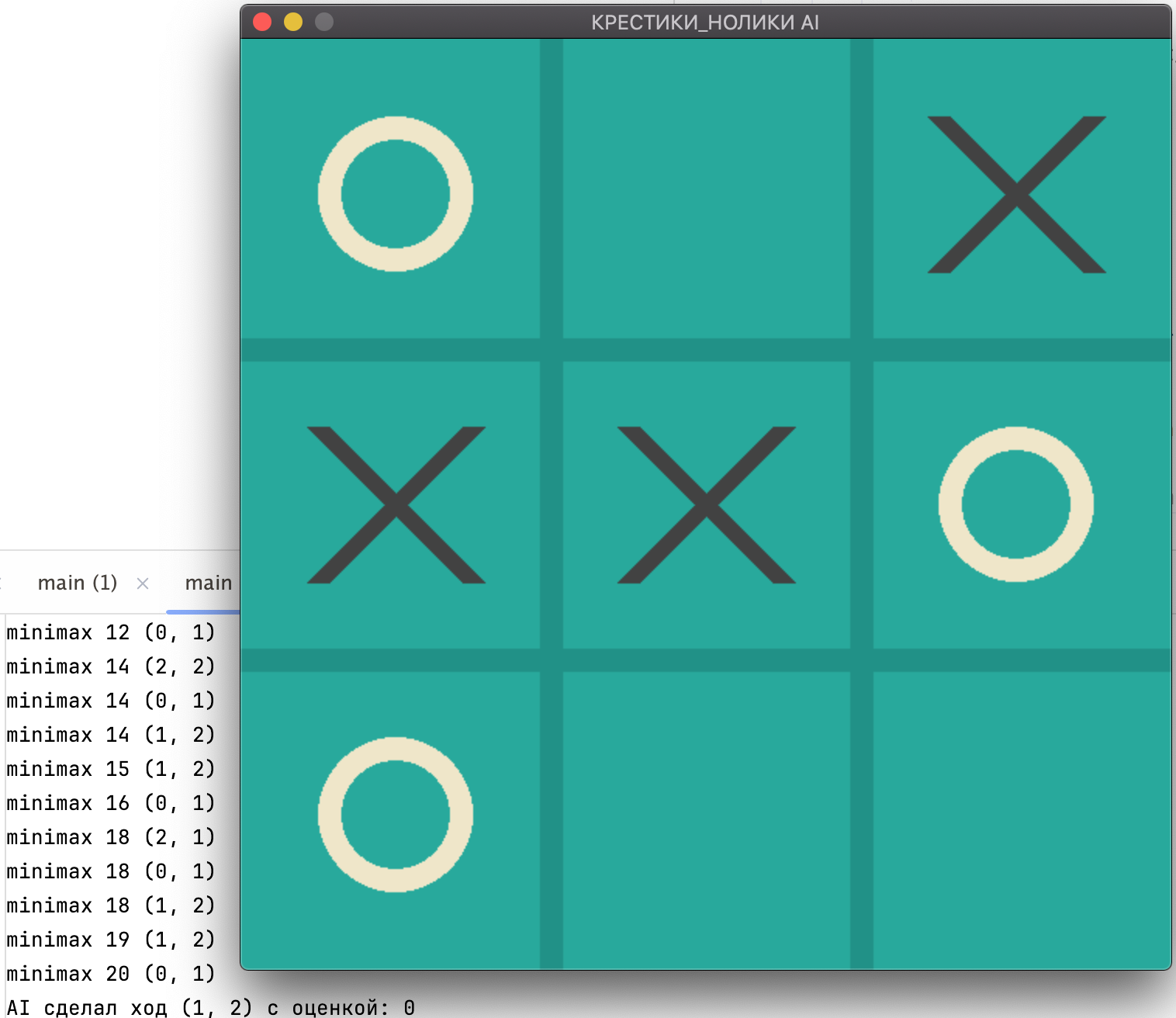


Рисунок 3 – Игровое поле и протокол поиска после нескольких ходов

Для алгоритма с отсечениями:

После хода пользователя алгоритм с отсечениями вычисляет выигрышный ход за 4455итераций, в этом можно убедиться, посмотрев протокол поиска, представленный на рисунке 4.

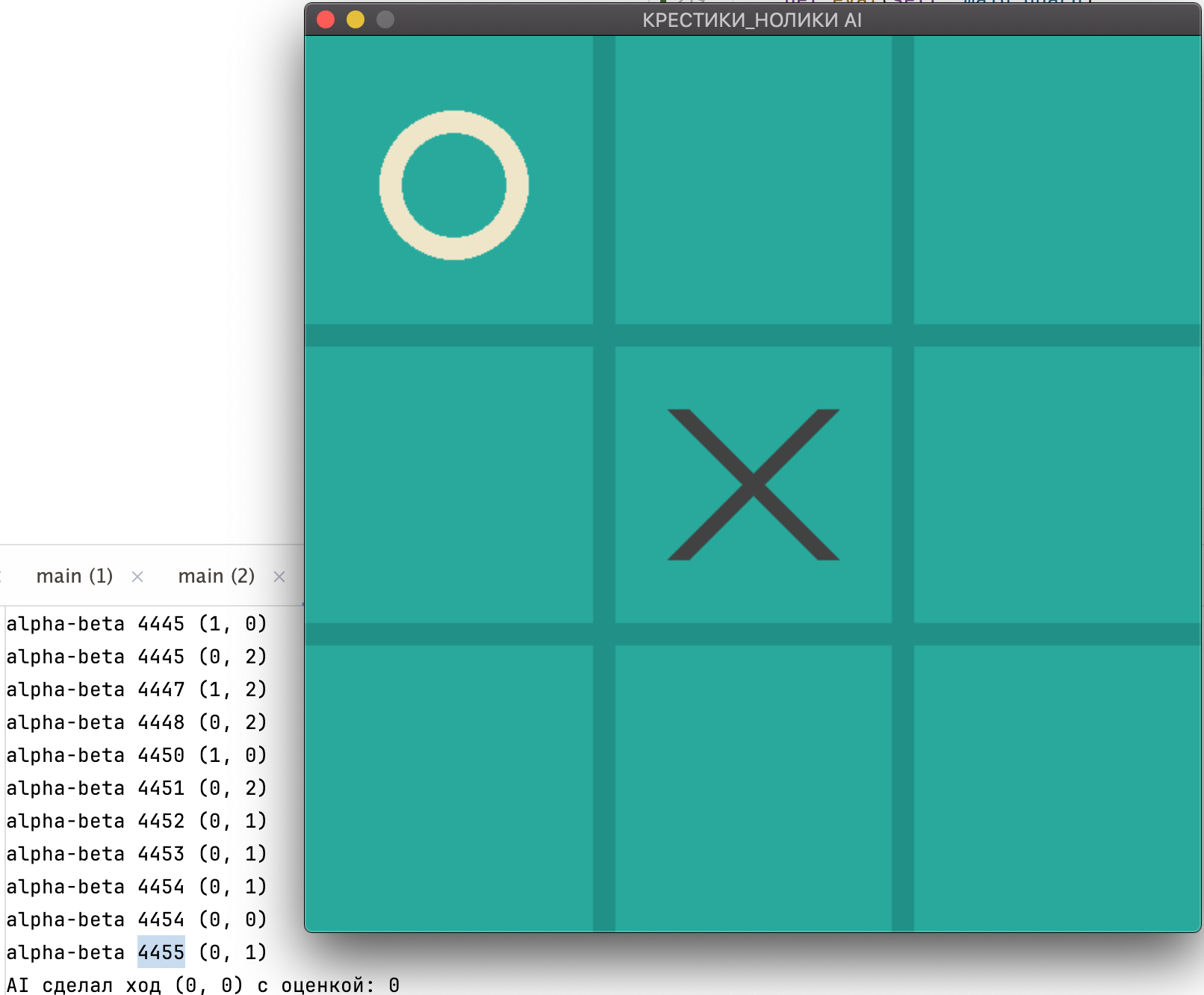


Рисунок 4 – Протокол поиска и ход ИИ (алгоритм с отсечениями)

После каждого следующего хода количество итераций будет уменьшаться из-за того, что количество возможных ходов уменьшается. Игровое поле и протокол поиска после нескольких ходов для алгоритма с отсечениями представлены на рисунке 5.

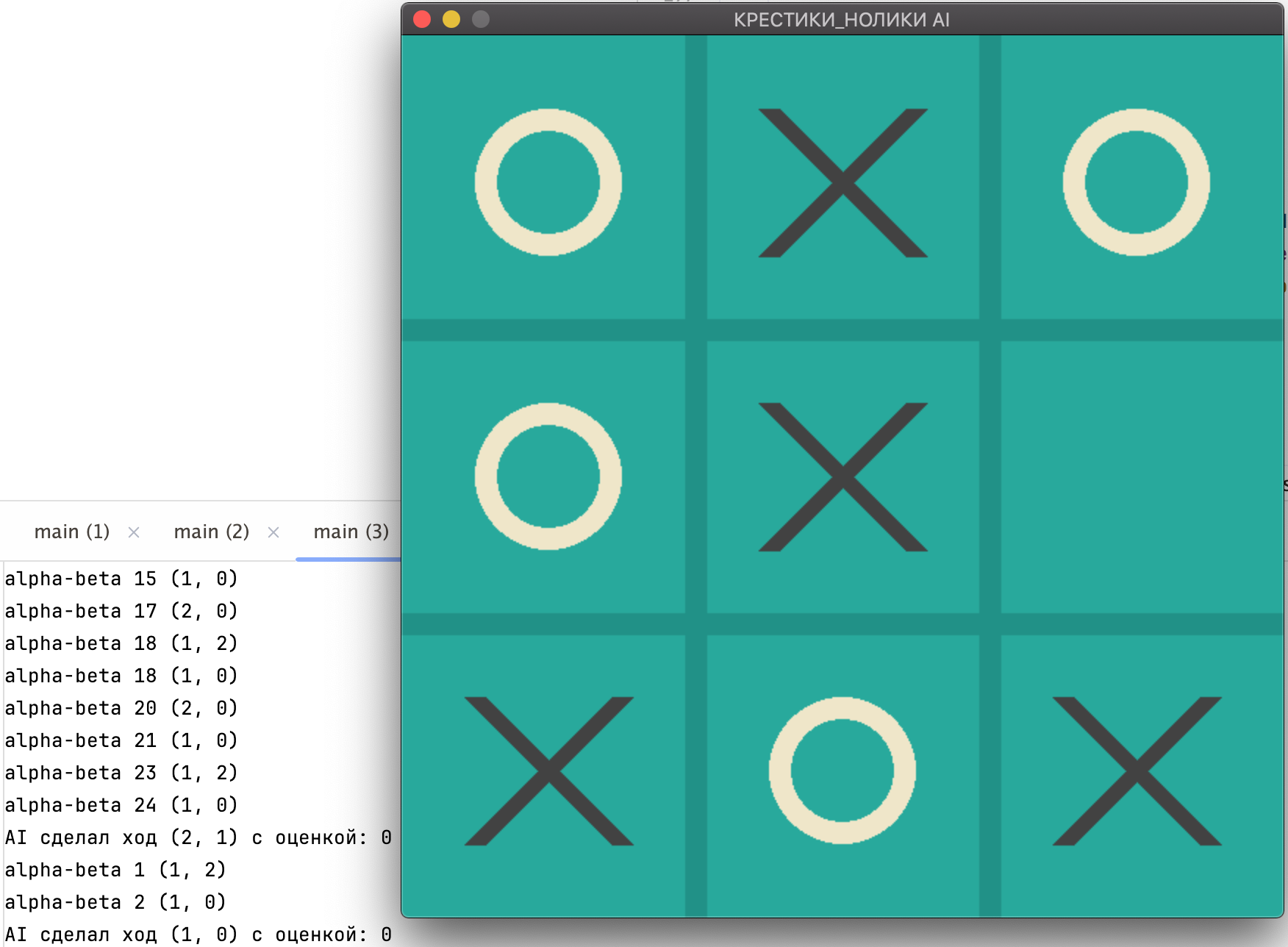


Рисунок 5 – Игровое поле и протокол поиска после нескольких ходов

Для минимаксного алгоритма:

def minimax(self, board, maximizing):

case = board.final\_state()

# player 1 побеждает

if case == 1:

return 1, None # eval, move

# player 2 побеждает

if case == 2:

return -1, None

elif board.isfull():

return 0, None

if maximizing:

self.count += 1

max\_eval = -100

best\_move = None

empty\_sqrs = board.get\_empty\_sqrs()

for (row, col) in empty\_sqrs:

temp\_board = copy.deepcopy(board)

temp\_board.mark\_sqr(row, col, 1)

eval = self.minimax(temp\_board, False)[0]

if eval > max\_eval:

max\_eval = eval

best\_move = (row, col)

print("minimax", self.count, best\_move)

return max\_eval, best\_move

elif not maximizing:

min\_eval = 100

best\_move = None

empty\_sqrs = board.get\_empty\_sqrs()

for (row, col) in empty\_sqrs:

temp\_board = copy.deepcopy(board)

temp\_board.mark\_sqr(row, col, self.player)

eval = self.minimax(temp\_board, True)[0]

if eval < min\_eval:

min\_eval = eval

best\_move = (row, col)

return min\_eval, best\_move

Для алгоритма с отсечениями:

def alpha\_beta(self, board, maximizing, alpha, beta):

case = board.final\_state()

# player 1 побеждает

if case == 1:

return 1, None # eval, move

# player 2 побеждает

if case == 2:

return -1, None

elif board.isfull():

return 0, None

if maximizing:

max\_eval = -100

best\_move = None

empty\_sqrs = board.get\_empty\_sqrs()

for (row, col) in empty\_sqrs:

self.count += 1

temp\_board = copy.deepcopy(board)

temp\_board.mark\_sqr(row, col, 1)

eval = self.alpha\_beta(temp\_board, False, alpha, beta)[0]

if eval > max\_eval:

max\_eval = eval

best\_move = (row, col)

print("alpha-beta", self.count, best\_move)

# alpha-beta pruning

if max\_eval > beta:

return max\_eval, best\_move

alpha = max(alpha, max\_eval)

return max\_eval, best\_move

elif not maximizing:

min\_eval = 100

best\_move = None

empty\_sqrs = board.get\_empty\_sqrs()

for (row, col) in empty\_sqrs:

temp\_board = copy.deepcopy(board)

temp\_board.mark\_sqr(row, col, self.player)

eval = self.alpha\_beta(temp\_board, True, alpha, beta)[0]

if eval < min\_eval:

min\_eval = eval

best\_move = (row, col)

# alpha-beta pruning

if min\_eval < alpha:

return min\_eval, best\_move

beta = min(beta, min\_eval)

return min\_eval, best\_move