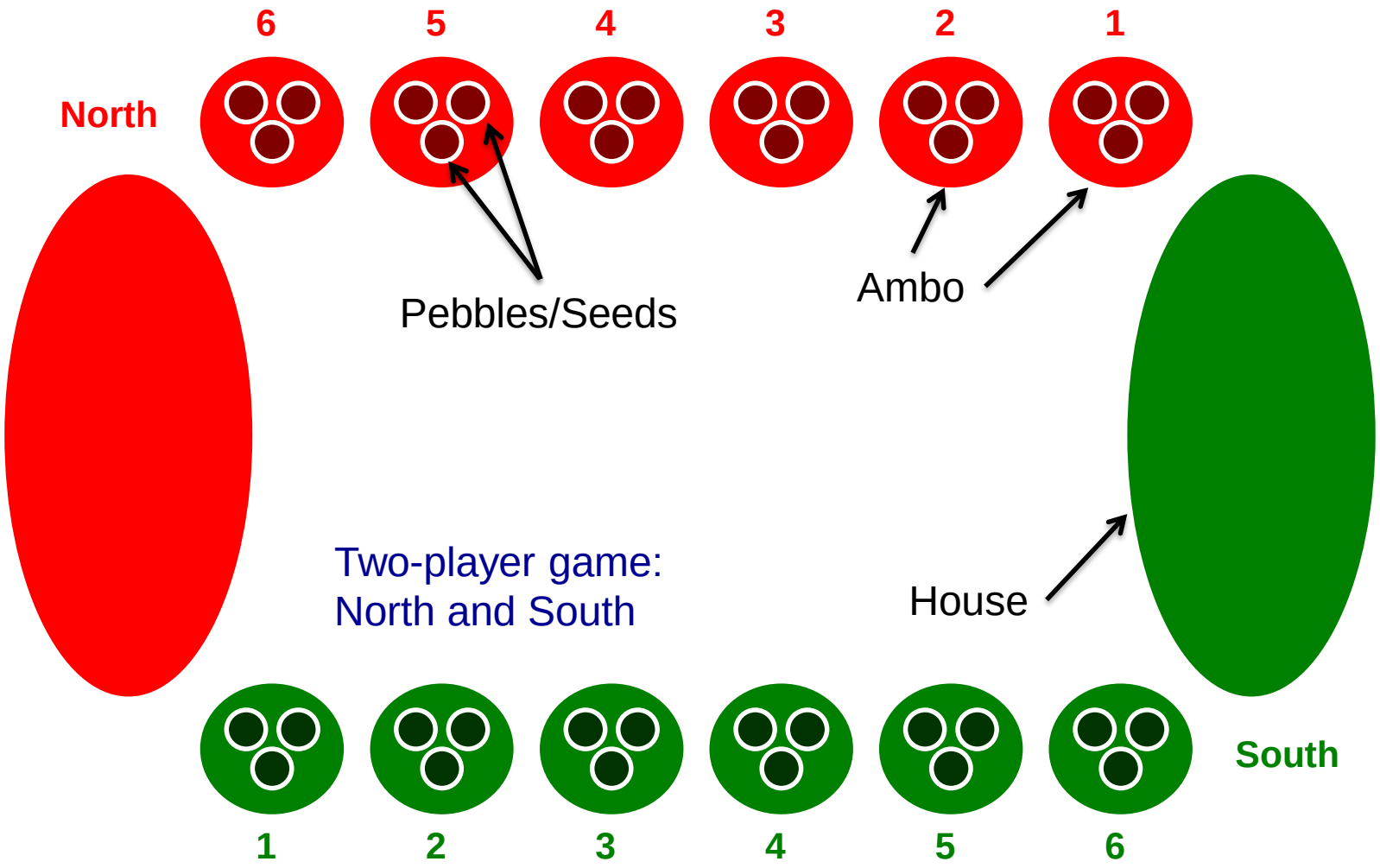
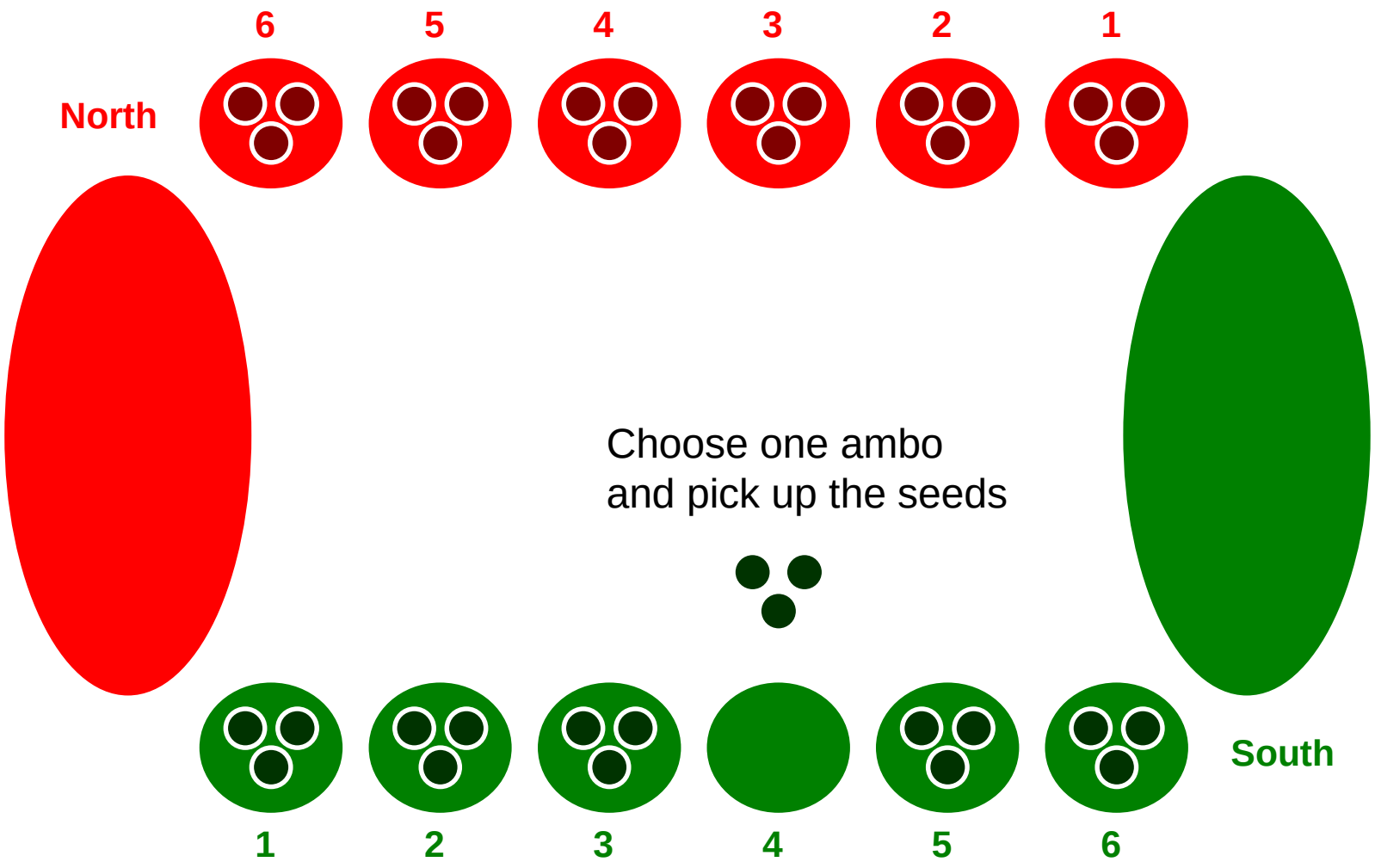
ASSIGNMENT 2 - KALAH

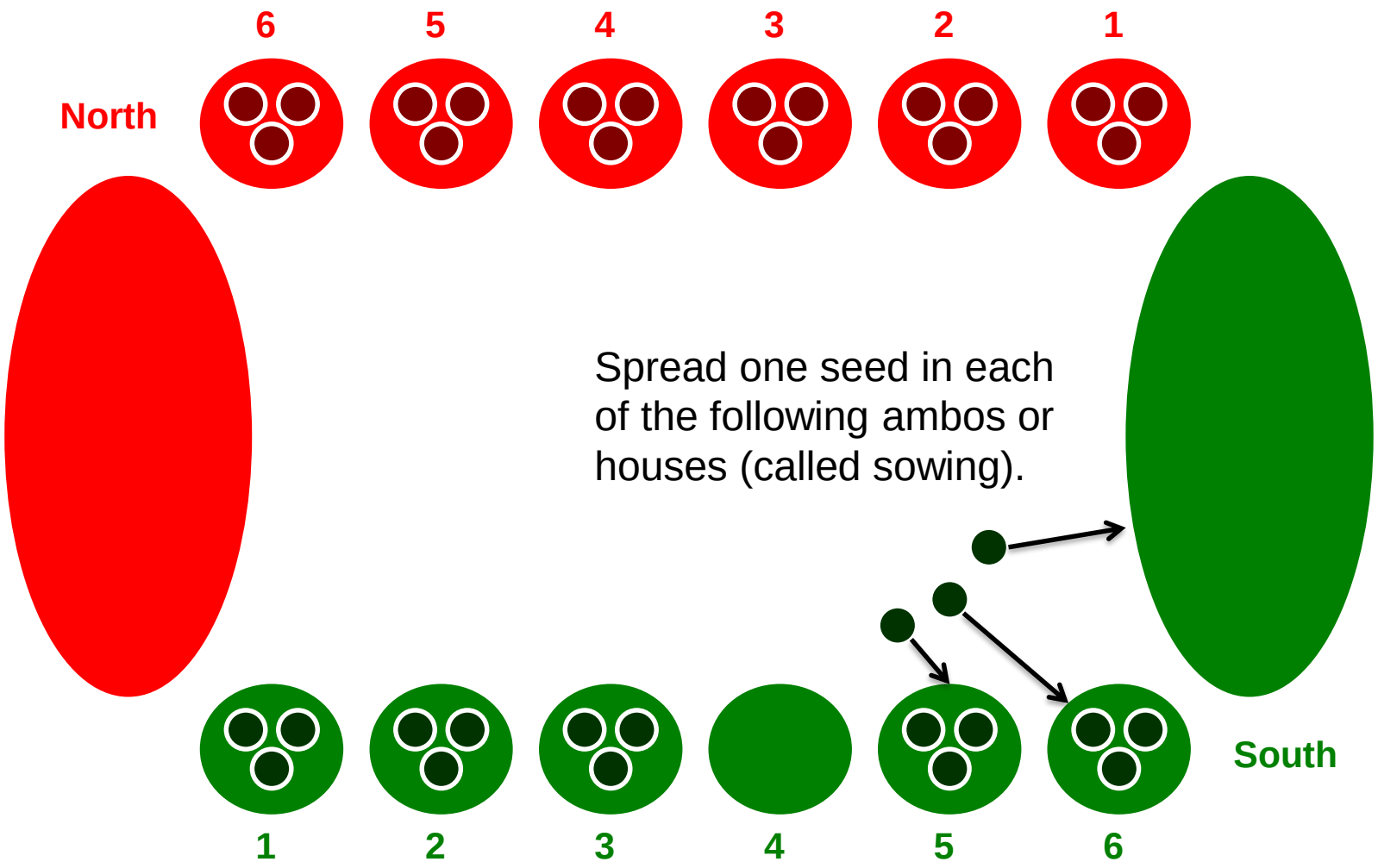
Kalah


6
5
4
3
2
1
North
Pebbles/Seeds
Ambo
House
Two-player game:
North and South
South
1
2
3
4
5
6

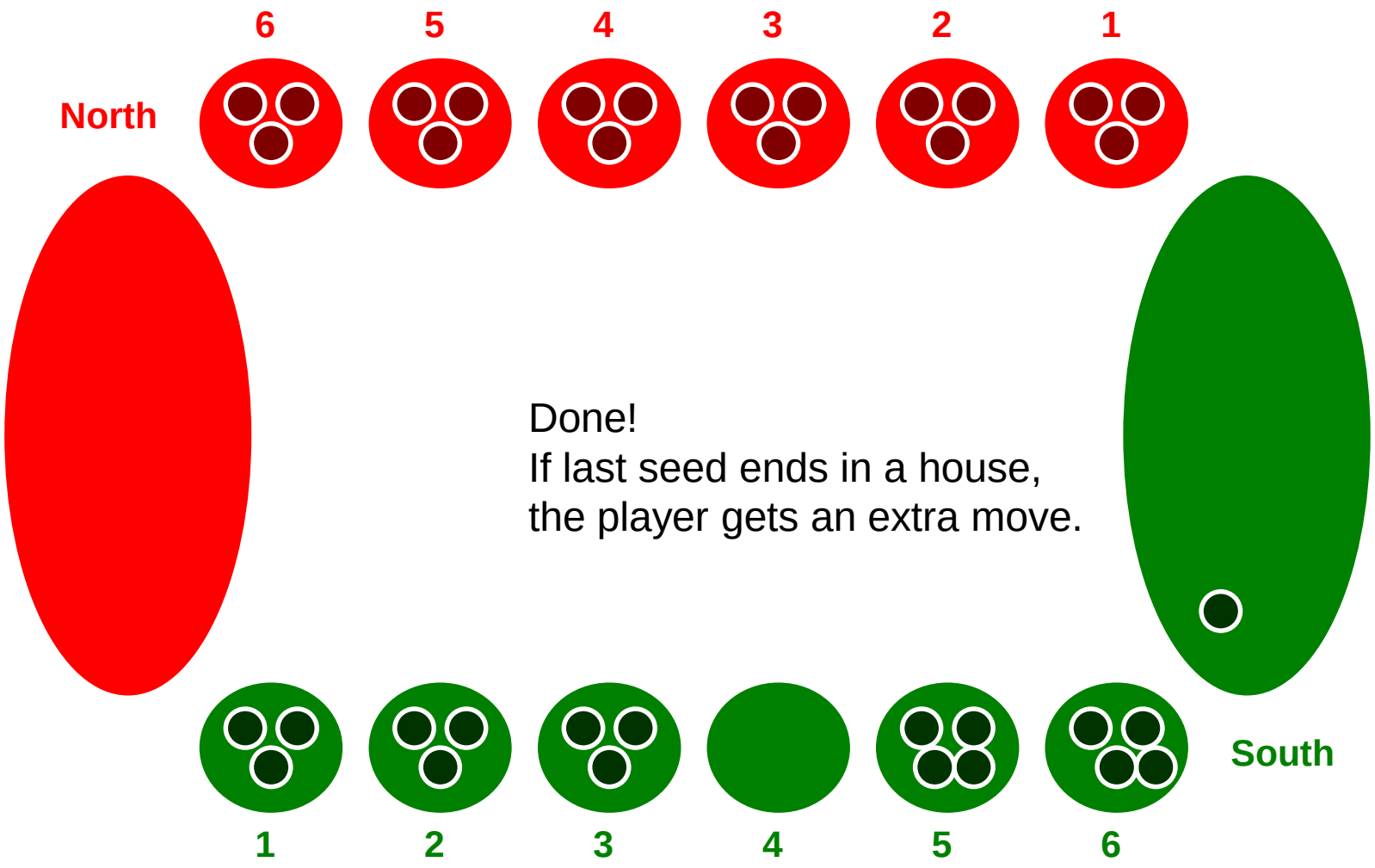
South’s move


6
5
4
3
2
1
North
Choose one ambo
and pick up the seeds
South
1
2
3
4
5
6

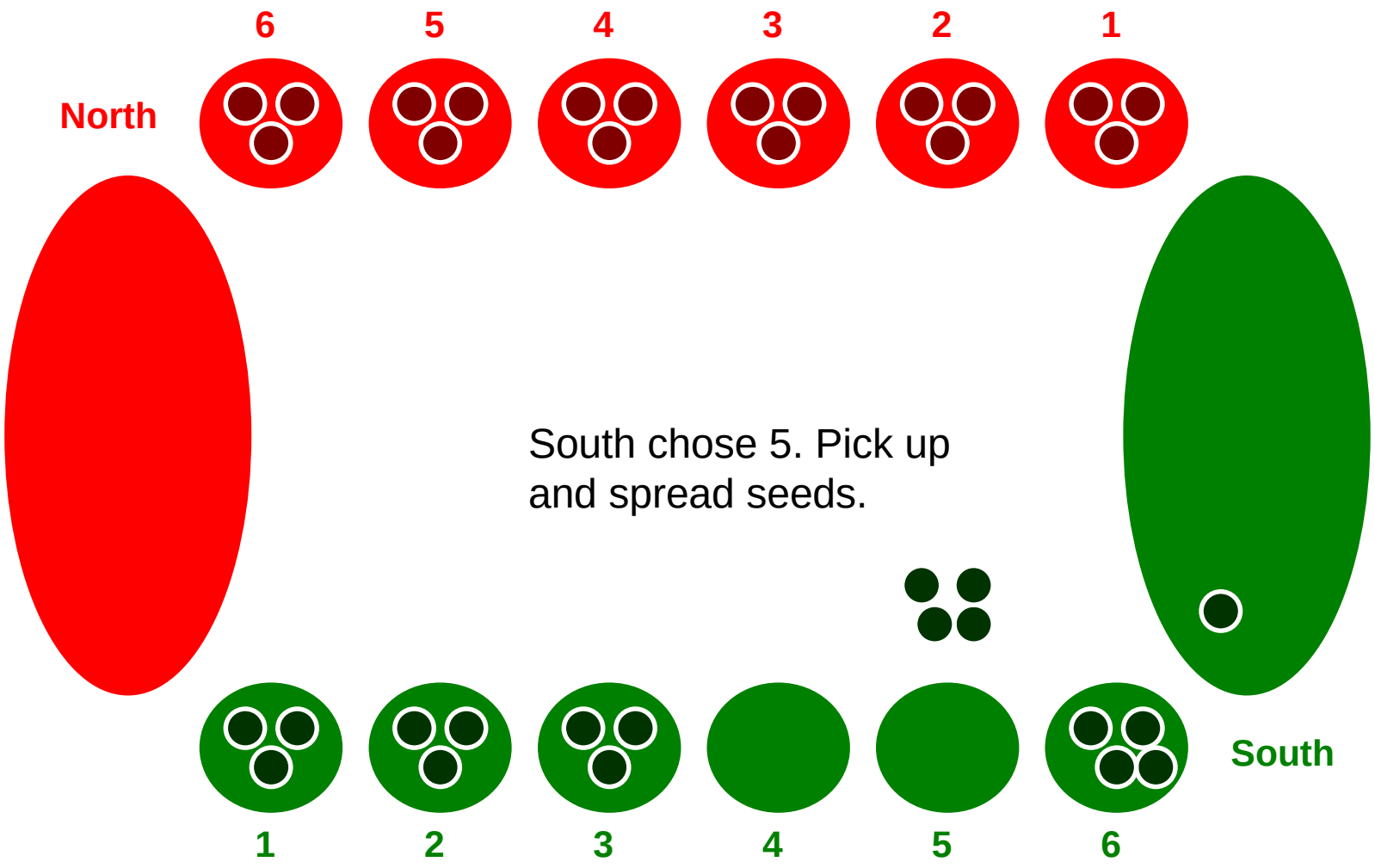
South's move


6
5
4
3
2
1
North
Spread one seed in each of the following ambos or houses (called sowing).
South
1
2
3
4
5
6

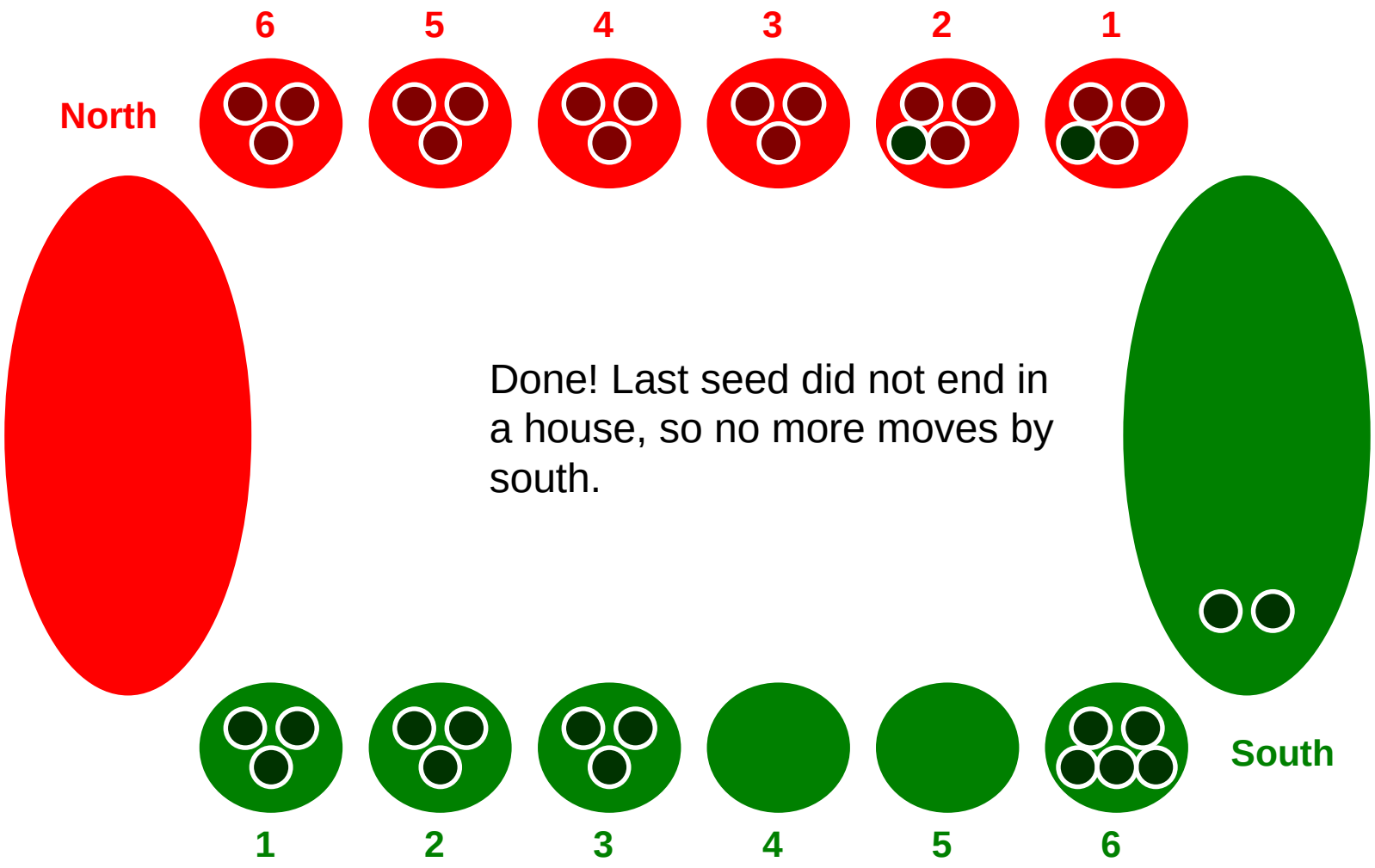
South’s move


6
5
4
3
2
1
North
Done!
If last seed ends in a house,
the player gets an extra move.
South
1
2
3
4
5
6

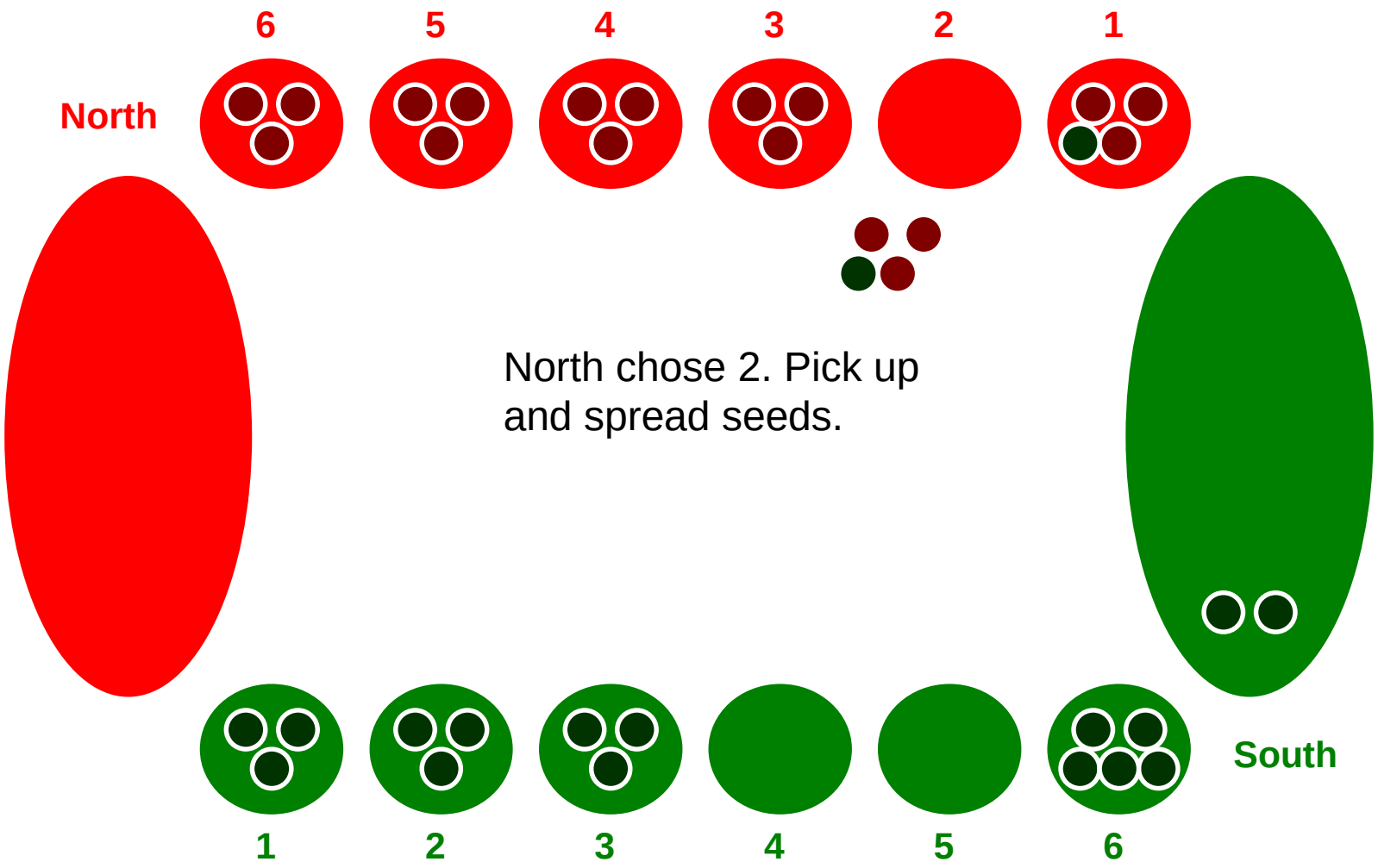
South’s move


6
5
4
3
2
1
North
South chose 5. Pick up and spread seeds.
South
1
2
3
4
5
6

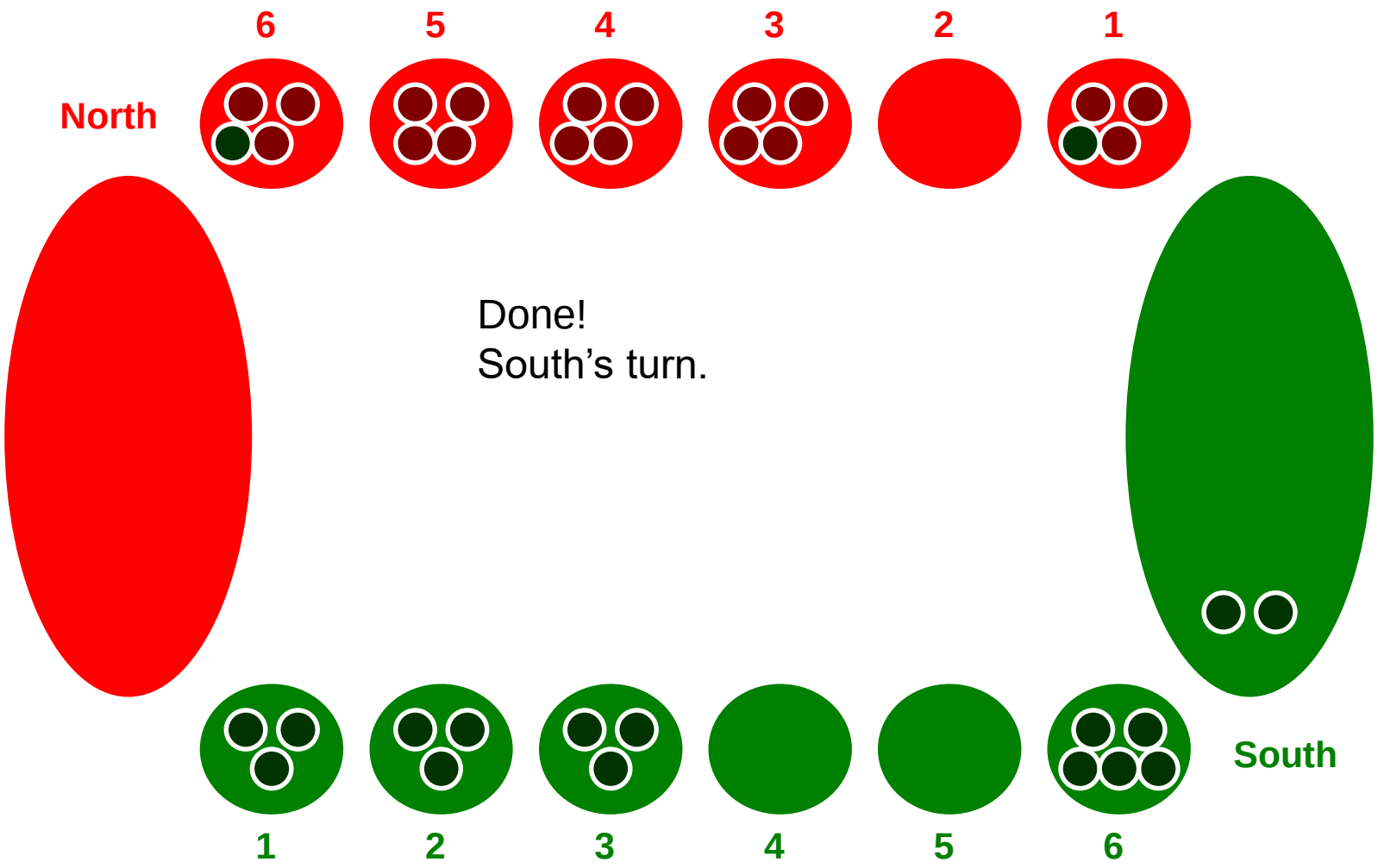
South’s move


6
5
4
3
2
1
North
Done! Last seed did not end in a house, so no more moves by south.
South
1
2
3
4
5
6

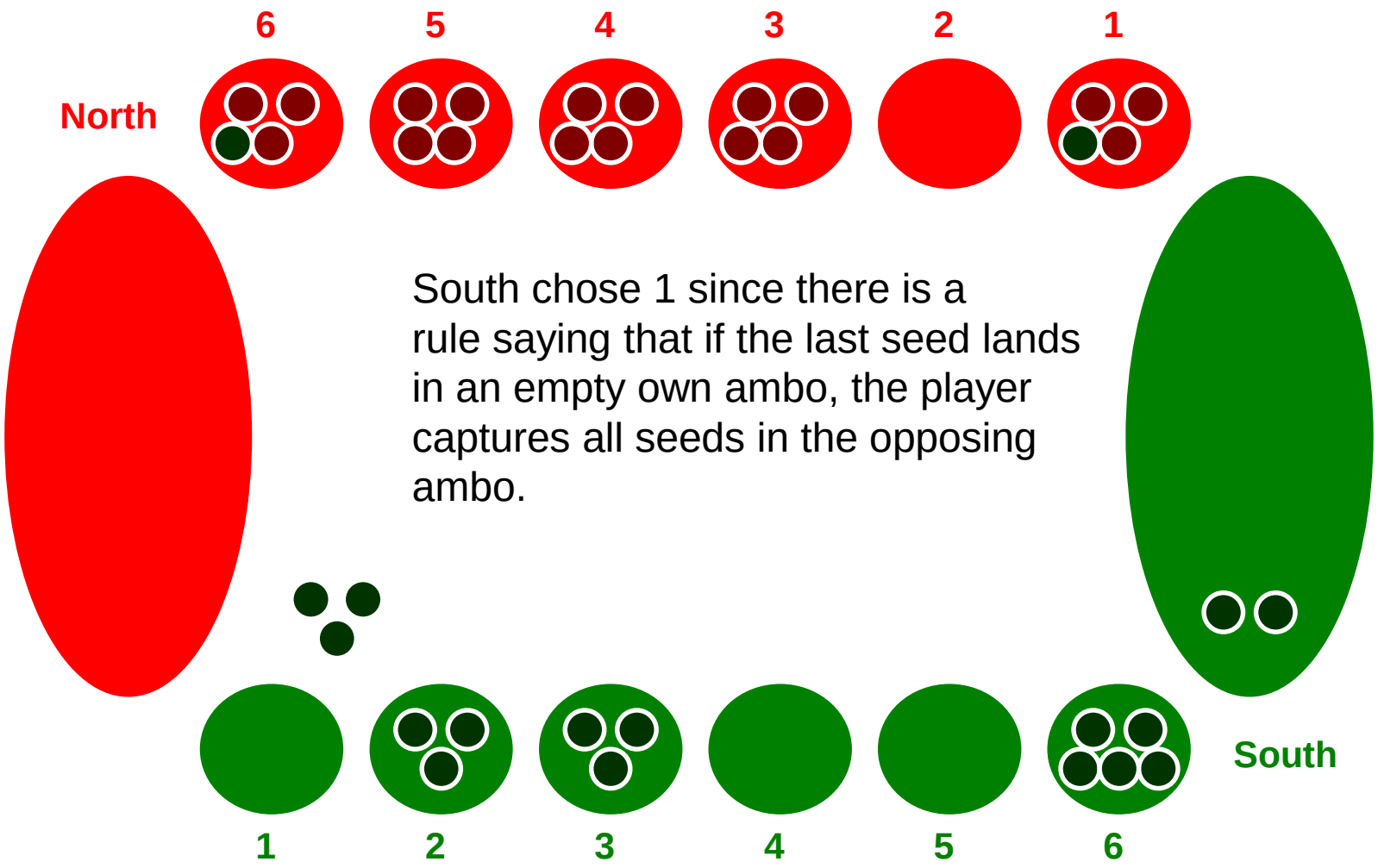
North’s move


6
5
4
3
2
1
North
North chose 2. Pick up and spread seeds.
South
1
2
3
4
5
6

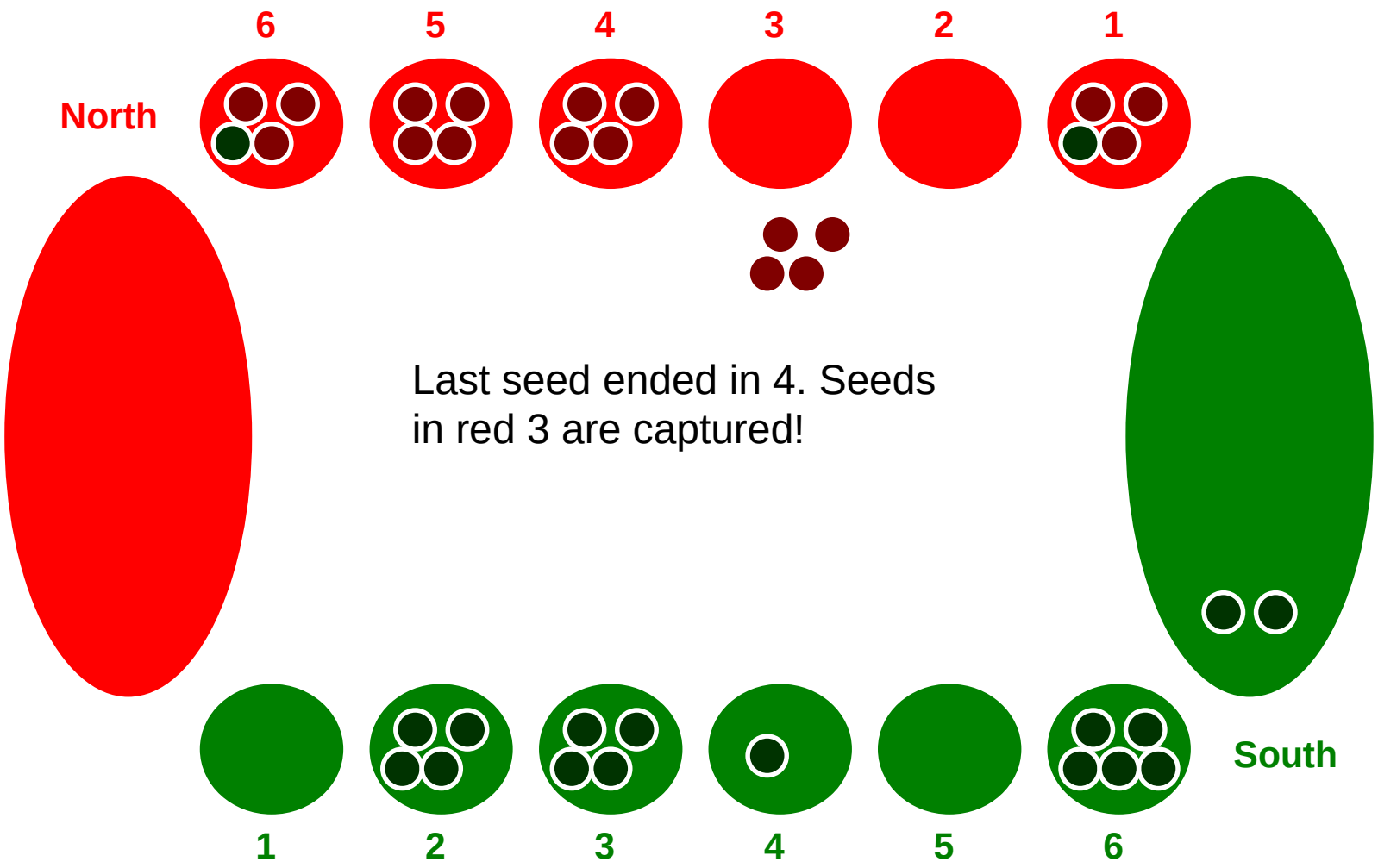
North's move



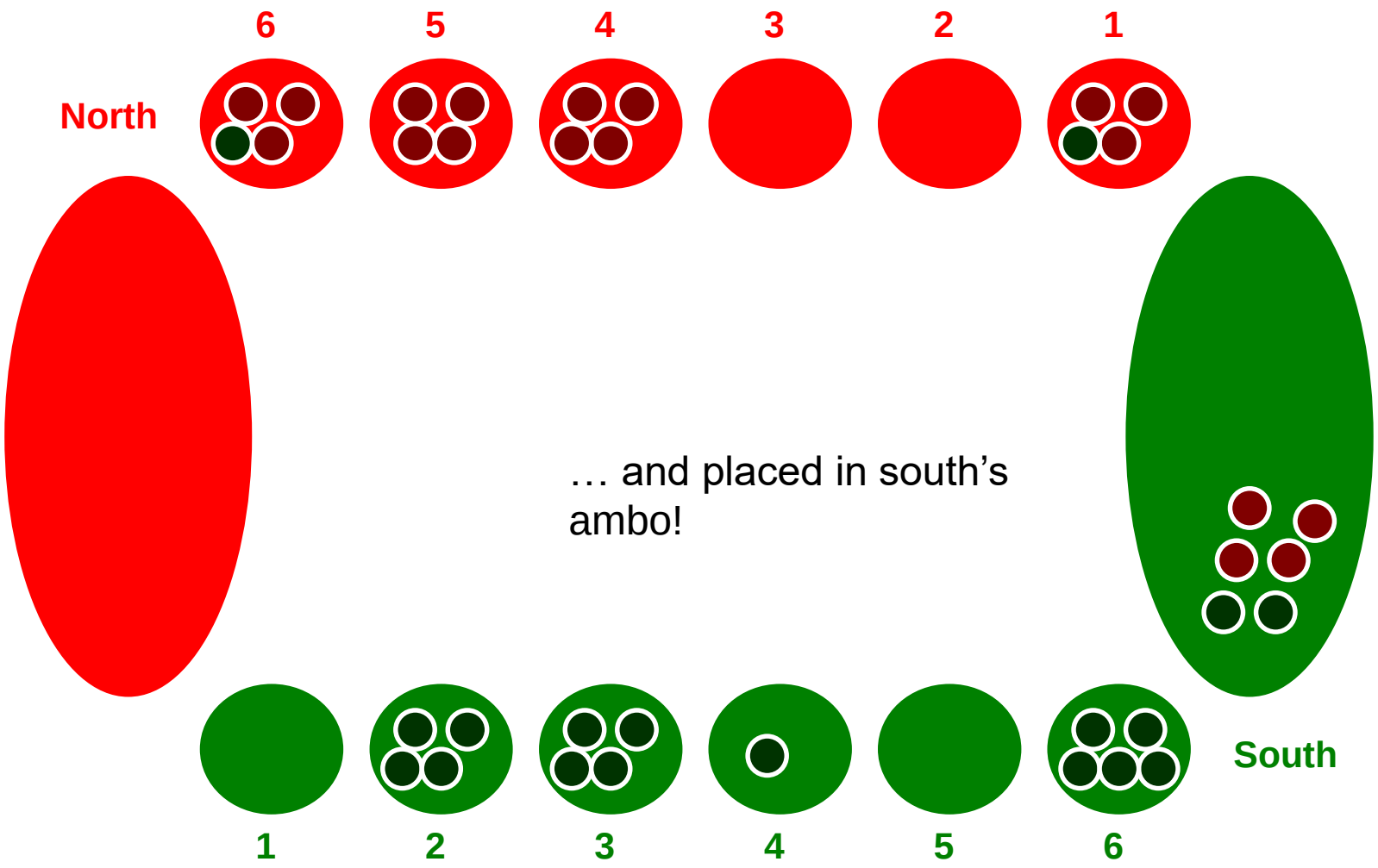
South’s move


6
5
4
3
2
1
North
South chose 1 since there is a rule saying that if the last seed lands in an empty own ambo, the player captures all seeds in the opposing ambo.
South
1
2
3
4
5
6

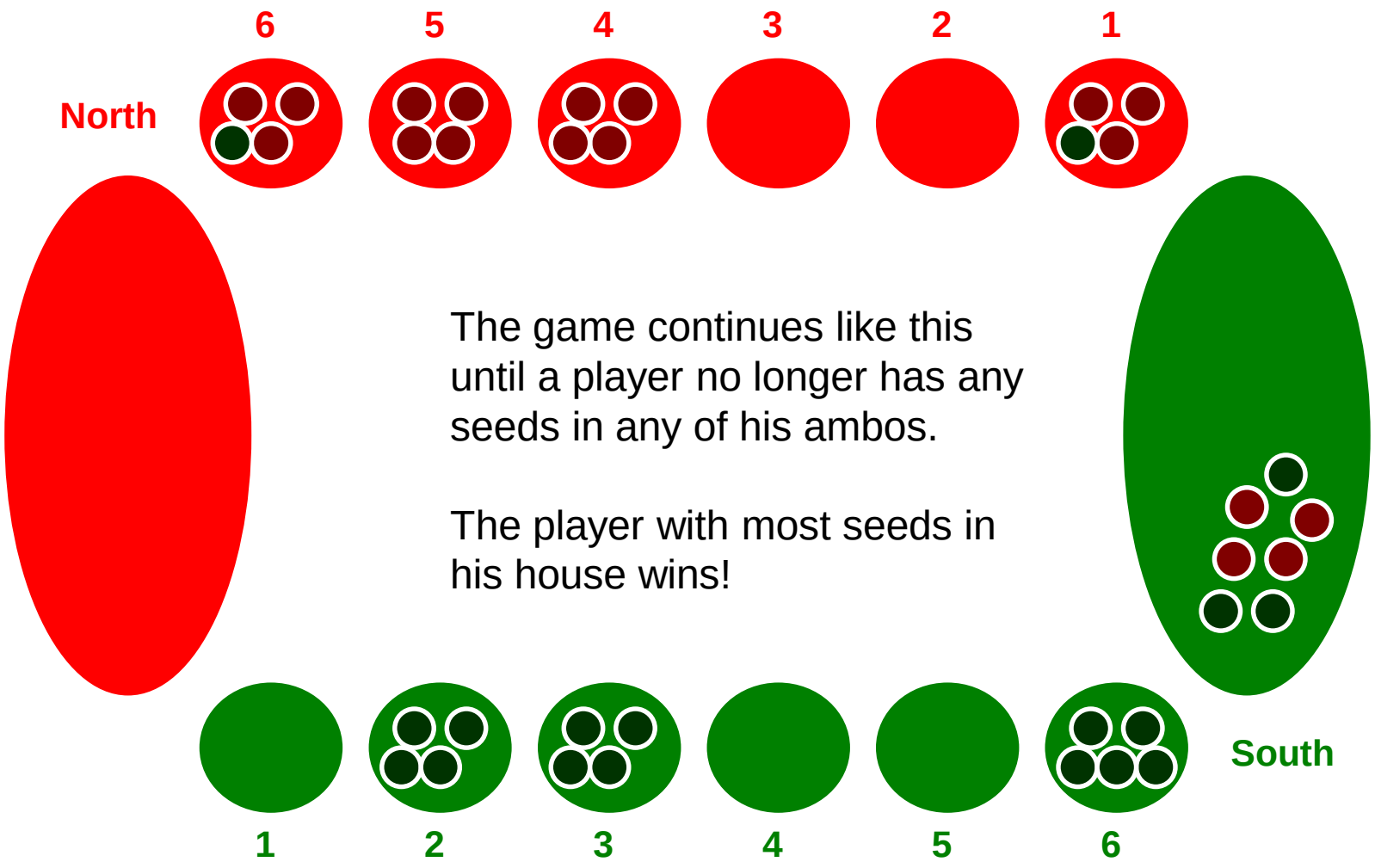
South’s move


6
5
4
3
2
1
North
Last seed ended in 4. Seeds in red 3 are captured!
South
1
2
3
4
5
6

South’s move



Game ends



The game continues like this until a player no longer has any seeds in any of his ambos.

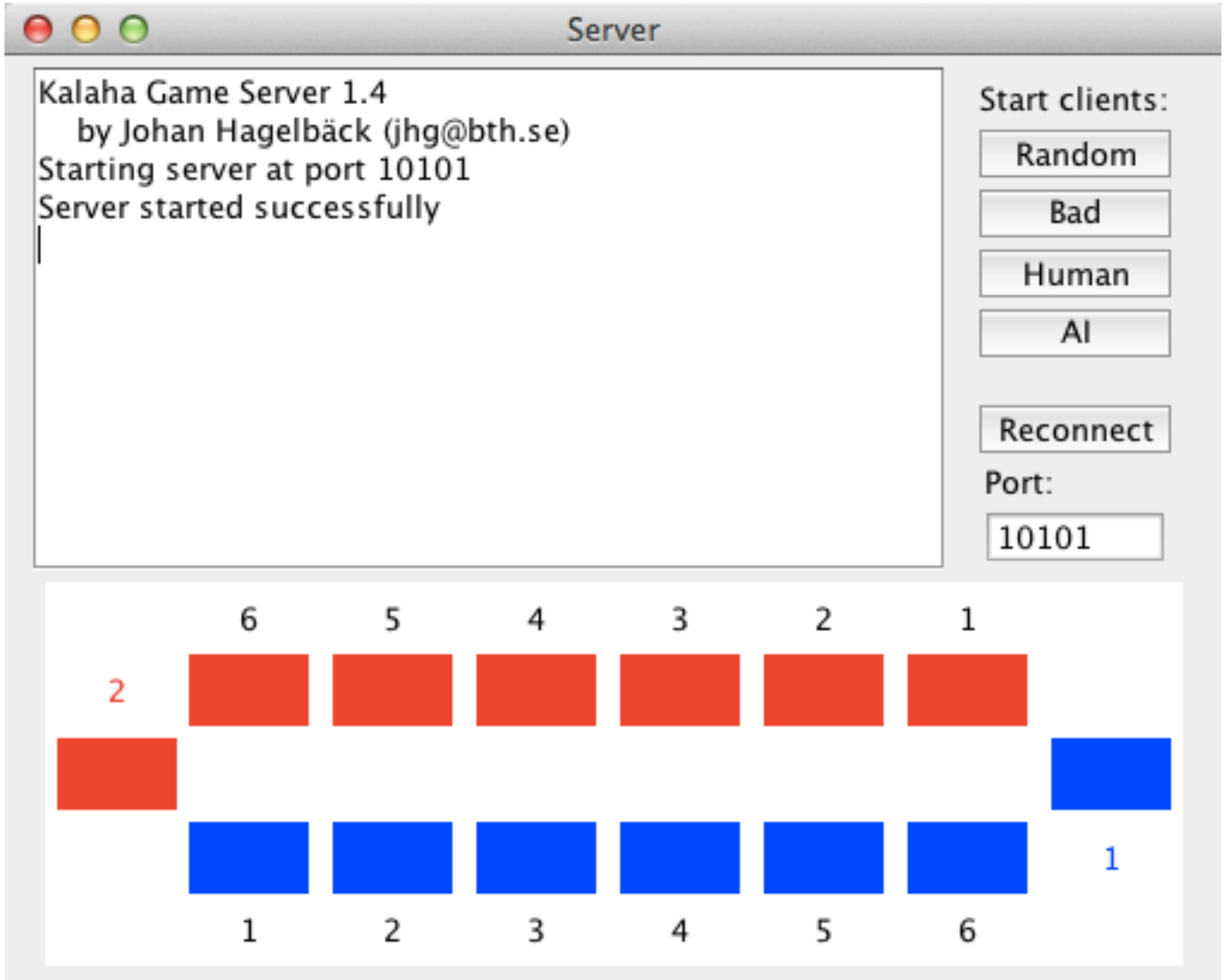
The player with most seeds in his house wins!

Complete rules

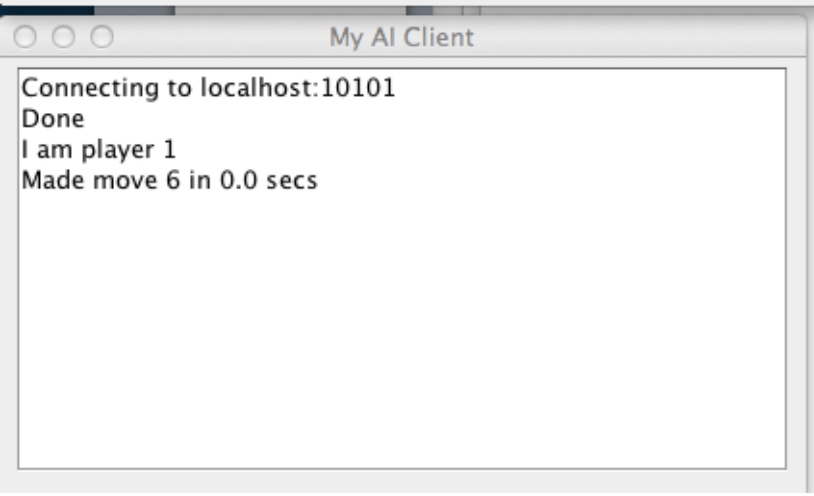
1. At the beginning of the game, three pebbles are placed in each ambo. Typically, the winner of the previous game starts the next game.
2. Each player controls the six ambos and their pebbles on his side of the board. His score is the number of pebbles in the kalah to his right.
3. Players take turns sowing their pebbles. On a turn, the player removes all pebbles from one of the ambos under his control. Moving counter-clockwise, the player drops one pebble in each ambo in turn, including the player's own kalah but not his opponent's.
4. If the last sown pebble lands in the player's kalah, the player gets an additional move. There is no limit on the number of moves a player can make in his turn.
5. If the last sown kalah lands in an empty ambo owned by the player, and the opposite ambo contains pebbles, both the last pebble and the opposite pebbles are captured and placed into the player's kalah.
6. When one player no longer has any pebbles in any of his ambos, the game ends. The other player moves all remaining pebbles to his kalah, and the player with the most pebbles in his kalah wins.

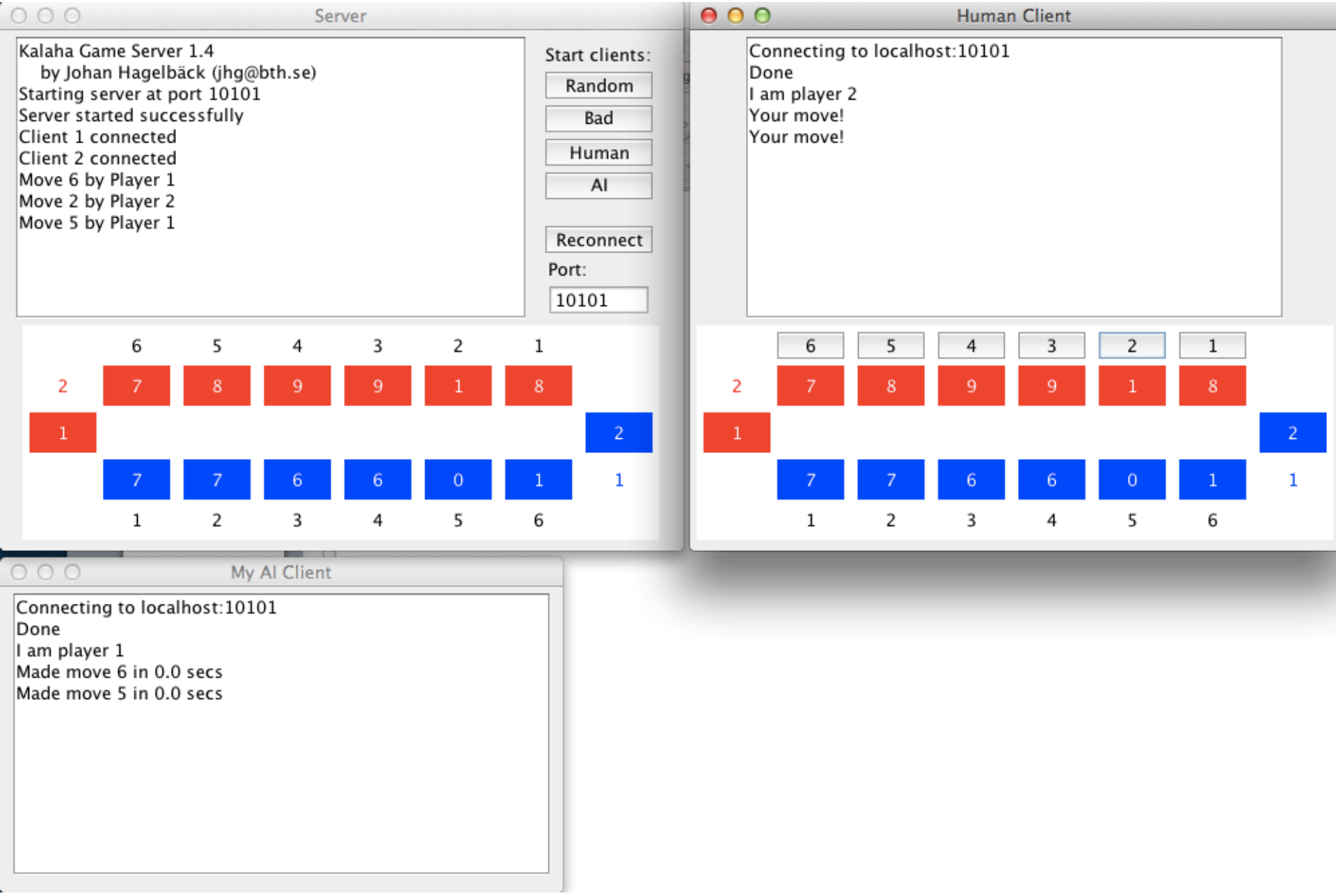
http://en.wikipedia.org/wiki/Kalaha

The Application




Server
Kalaha Game Server 1.4
by Johan Hagelbäck (jhg@bth.se)
Starting server at port 10101
Server started successfully
Client 1 connected
Client 2 connected
Move 6 by Player 1
Start clients:
Random
Bad
Human
AI
Reconnect
Port:
10101
6 5 4 3 2 1
2 6 7 7 7 7 7
0 1
6 6 6 6 6 0 1
1 2 3 4 5 6
Human Client
Connecting to localhost:10101
Done
I am player 2
Your move!
6 5 4 3 2 1
2 6 7 7 7 7 7
0 1
6 6 6 6 6 0 1
1 2 3 4 5 6


My AI Client
Connecting to localhost:10101
Done
I am player 1
Made move 6 in 0.0 secs


Server
Kalaha Game Server 1.4
by Johan Hagelbäck (jhg@bth.se)
Starting server at port 10101
Server started successfully
Client 1 connected
Client 2 connected
Move 6 by Player 1
Move 2 by Player 2
Move 5 by Player 1
Start clients:
Random
Bad
Human
AI
Reconnect
Port:
10101
6 5 4 3 2 1
2 7 8 9 9 1 8
1 2
7 7 6 6 0 1 1
1 2 3 4 5 6
Human Client
Connecting to localhost:10101
Done
I am player 2
Your move!
Your move!
6 5 4 3 2 1
2 7 8 9 9 1 8
1 2
7 7 6 6 0 1 1
1 2 3 4 5 6
My AI Client
Connecting to localhost:10101
Done
I am player 1
Made move 6 in 0.0 secs
Made move 5 in 0.0 secs

Example

- A version of the program modified for the web can be found here: http://aiguy.org/Kalaha.html

Application code

- The only class you need to make changes in is AIClient.java
- Changes are to be made in the method:
  - int getMove(GameState currentBoard)
- Here you shall make a Minimax search, and return the best move (1-6).
- The GameState class contains methods for making a move and update the board.
- Make sure to clone() the GameState each time you need a new board state.

Requirements

- Grade E:
  - Minimax with Depth-First search stopping at a pre-defined depth level (>4).
- Grade D:
  - Minimax with Iterative Deepening search stopping before a max time of 5 seconds.
- Grade C:
  - Minimax with Depth-First search and Alpha-Beta pruning stopping at a pre-defined depth level (>4).
- Grade B:
  - Minimax with Iterative Deepening search and Alpha-Beta pruning stopping time of 5 seconds.
- Grade A:
  - As Grade B, but with an opening book that has stored >100 game states that are relevant for deciding the first move. This shall be used instead of Minimax for deciding the first move from the AI.

Requirements

- All solutions have to be compatible with the given code.
  - No change in programming language or IDE is permitted.
- Comment your code to make the grading task easier.

What to submit

- The complete source code for the Kalaha program containing your AI code, including the project files.
- A note about which grade you are aiming for.
- Submit to It's Learning no later than: **28th September 2017, 23:59**