Nim Game Implementation

Global CIS Special Lecture 1 Final Project

Jin Haichen

Graduate School of Computer and Information Sciences

Hosei University

Tokyo, Japan

haichen.jin.24@stu.hosei.ac.jp

*Abstract*—This article explains how the Nim Game AI work, and show an implementation of this game with html and javascript.

Keywords—Nim Game, html, javascript, canvas

# Introduction

Nim Game is a [mathematical](https://en.wikipedia.org/wiki/Mathematical_game) [game of strategy](https://en.wikipedia.org/wiki/Game_of_strategy) in which two players take turns removing objects from distinct heaps. On each turn, a player must remove at least one object, and may remove any number of objects provided they all come from the same heap. The goal of the game is to avoid being the player who must remove the last object.

# Algorithm

As we know that for a complete information game, so for a given situation, there must be a player has winning strategy.

Nim Game is obviously a complete information game, and the winning strategy is really easy to understand.

For a given situation, there are several piles and each of them has some stones on it. If the exclusive or (XOR) of all the numbers of stones on the piles is not 0, it means the player at offensive position has the winning strategy. And if the XOR of all the numbers of stones on the piles is 0, it means the player at defensive position has the winning strategy.

1. When every pile has no stone on it, the XOR is 0 and obviously the player at defensive position has already win this game.
2. When there are some stones on the piles and the XOR is not zero, there must be a way to remove some of the stones on a pile to make the XOR 0.

First of all calculate the XOR of all piles.

S = A xor B xor C

And then do XOR with the result and some pile’s stone count that satisfies: after the XOR, the result is lower than the former count of stones on that pile. Then we just remove corresponding stones to let the stones remain has the count of the second XOR result.

If S xor A < A, then newA = S xor A

newS = newA xor B xor C = S xor A xor B xor C = 0

1. When there are some stones on the piles and the XOR is zero, no matter how the player remove the stones on one pile, the new XOR can’t be zero.

newA < A, so newA xor A != 0

0 = A xor B xor C != newA xor B xor C

So there must be a player has winning strategy and my AI is going to use the winning strategy if it can.

# Game Implementation

The game has 3 different status: configuration mode, game mode, game finish. I will show my implementation separately.

## Configuration

Before game starts, user can set the parameters of the game in configuration mode.



1. Configuration interface

Choosing to play with PVP, PVE, EVE (watch 2 AIs play game) and pile count and stone count in each pile to enjoy different game situations.

## Game mode

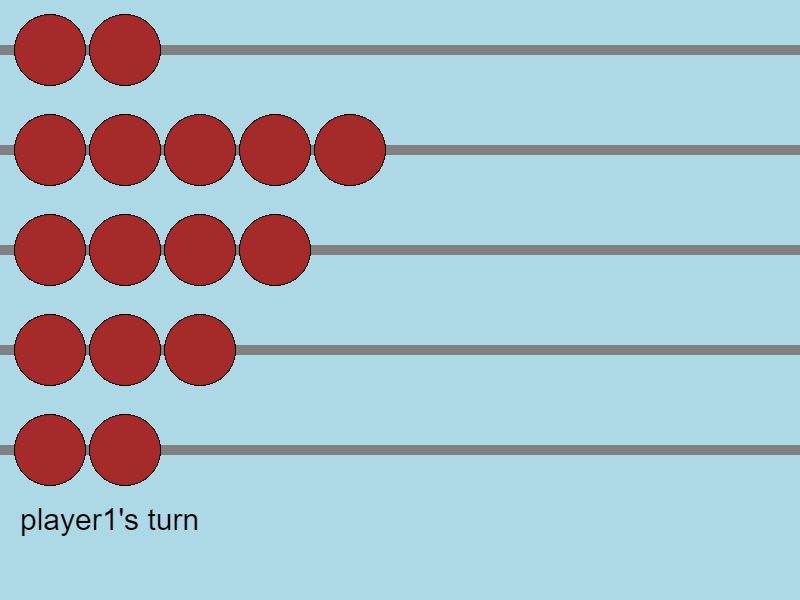
During the game, 2 players (player can be AI or human) remove stones alternatively, who removes the last stone from the last non-empty pile is the winner.

AI will “try its best” to win the game, he will definitely win if he can. He will also show whether he thinks he is already won.

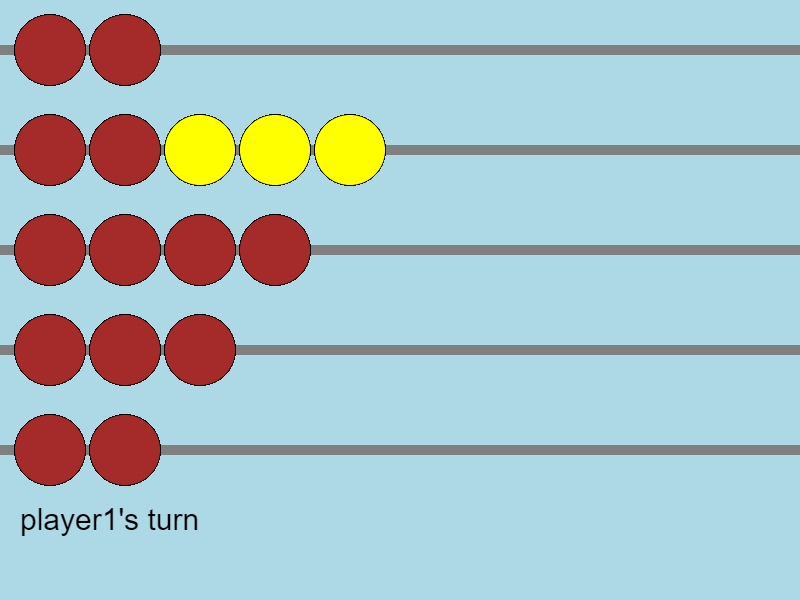
Human player can use mouse click to do the remove action.

## Game finish

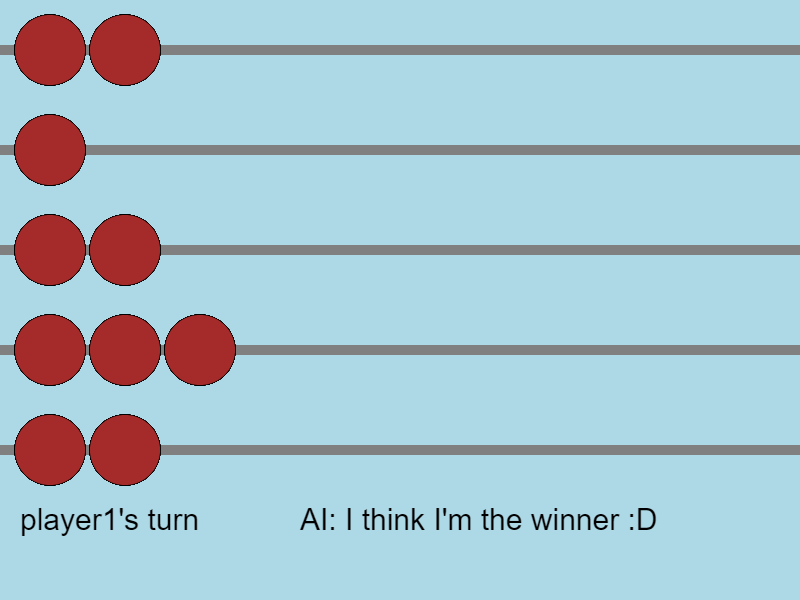
When all the stones are removed from the piles, game is over. The player who removes the last stone is the winner. Click “Play again” will let the program back to configuration mode.



1. Game Board



1. Click to remove stones



1. AI action



1. Game Finish

# Codes

There are 400 lines totally in this program so I won’t paste all the code but just explain each file.

## gameplay.js (102 lines)

Including all the logics in the Nim Game. I use a json to save the game status (players’ types, piles and stones, turn etc.). I also use promise for asynchronous operations such as waiting for human player to act.

## ui.js (215 lines)

I use canvas to draw the whole game board. It costs me many time to adjust the parameters avoiding ugly displays.

## index.html (43 lines)

Most of the lines are for the configuration DOMs.

## ai.js (42 lines)

Easiest part of the program. Calculating XOR to make decisions, if he can’t win, he will remove random stones from the pile with most stones. Wait 0.5 second for each turn for a better interface especially when 2 players are both AIs.