

# Warring States

made by:

Siying Qian(u6099927)

Lenna Nicholson(u4671448)

Lyu, Xiaoguang(u6464313)



## What we do

This game was made by Java and JavaFX, with the story background of warring states in ancient China.

In this game, we have:

- A launcher with instructions
- A card collection board
- A flag collecting system
- A notion board to tell player more information while playing
- Computer players with various difficulty levels
- Sound effects
- Hover over effects

# Design approach

We focused on making a game that was:

- Easy to understand and use
- Challenging and interesting to play
- Visually appealing

Our solution contains 3 main parts- the game class, the AI strategies class, and the warringstatesgame class

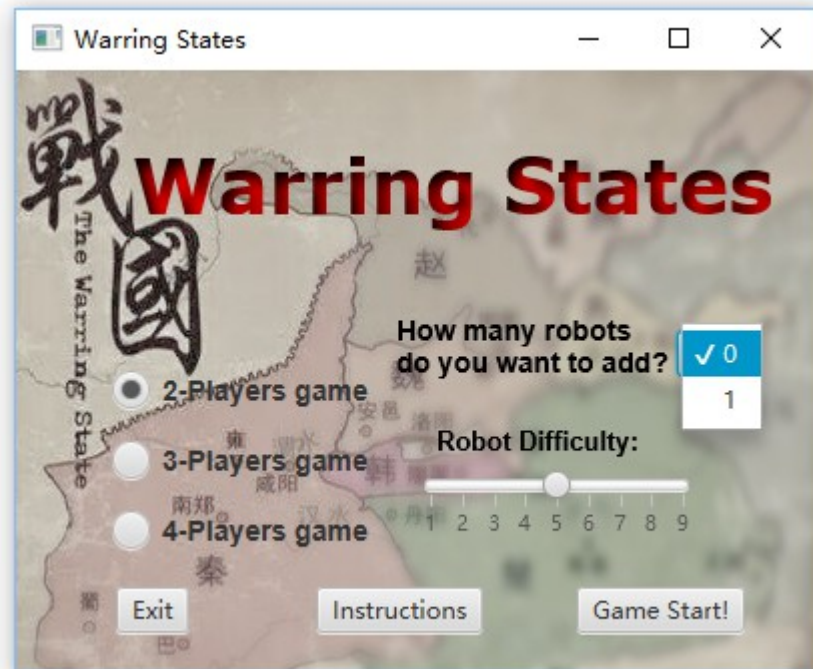
Strengths and weaknesses of our approach:

- Because much functionality within javafx, difficult to debug
- Focus on system that works for user

# Launcher

A easier way to start the game

Define how many players, how many robot, if the robot is more challenged, and also with a instruction of this game for new players



Players take turns to collect characters to their side, by clicking the board and moving Zhang Yi around the grid. On their turn, a player chooses a direction (North, East, South, or West) and a kingdom (Qin, Qi, Chu, Zhao, Han, Wei, or Yan). Zhang Yi then moves in the chosen direction to the location of the furthest away character from that kingdom, and collects that character card. If Zhang Yi passes other characters from the same kingdom while moving, he collects those characters as well. Each player may move Zhang Yi only once per turn.

At the end of their turn, if the player holds an equal or greater number of characters from a kingdom than any of their opponents, the player takes the flag of that kingdom. (If another player already holds the flag, it will change the belonging of this flag). The game ends when Zhang Yi cannot move, that is, when there are no cards in any direction (North, East, South, or West) from Zhang Yi. The player who holds the greatest number of flags at the end of the game wins. If two or more players hold the same number of flags, the player who holds the flag of the kingdom with the greatest number of characters wins.

Enjoy it!

[Back](#)



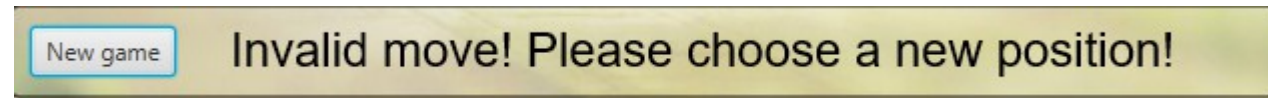
## Scoring

Lets players see which cards have been collected and who currently controls flags

# Invalid move

Hover over shows move is invalid

If clicked on, error message shown and sound effect played





## Valid move

Hover over shows move is valid

If clicked on, sound effect played

Shows which player's turn is next

Updates score board and flags

New game

Valid move. Next comes to Player 1's turn!

Lord Xinling	Wu Qi	King You	King Wuling	King Xuan	Lian Po
<div>楚</div> <div>哀王</div> <div>King Ai</div>	<div>齊</div> <div>鍾無艷</div> <div>Zhong Wuyan</div>	<div>秦</div> <div>惠文王</div> <div>King Hui</div>		<div>秦</div> <div>政王</div> <div>King Zheng</div>	<div>秦</div> <div>半八子</div> <div>Lady Mi</div>
<div>張儀</div> <div>Zhang Yi</div>	<div>齊</div> <div>君王后</div> <div>Queen Junwang</div>	<div>趙</div> <div>李牧</div> <div>Li Mu</div>		<div>齊</div> <div>孟嘗君</div> <div>Lord Menchang</div>	<div>齊</div> <div>襄王</div> <div>King Xiang</div>
<div>楚</div> <div>春申君</div> <div>Lord Chunshen</div>		<div>秦</div> <div>商鞅</div> <div>Shang Yang</div>		<div>楚</div> <div>考烈王</div> <div>King Kaolie</div>	<div>燕</div> <div>太子丹</div> <div>Prince Dan</div>

## End of game

Shows which player won the game

Sound effect played

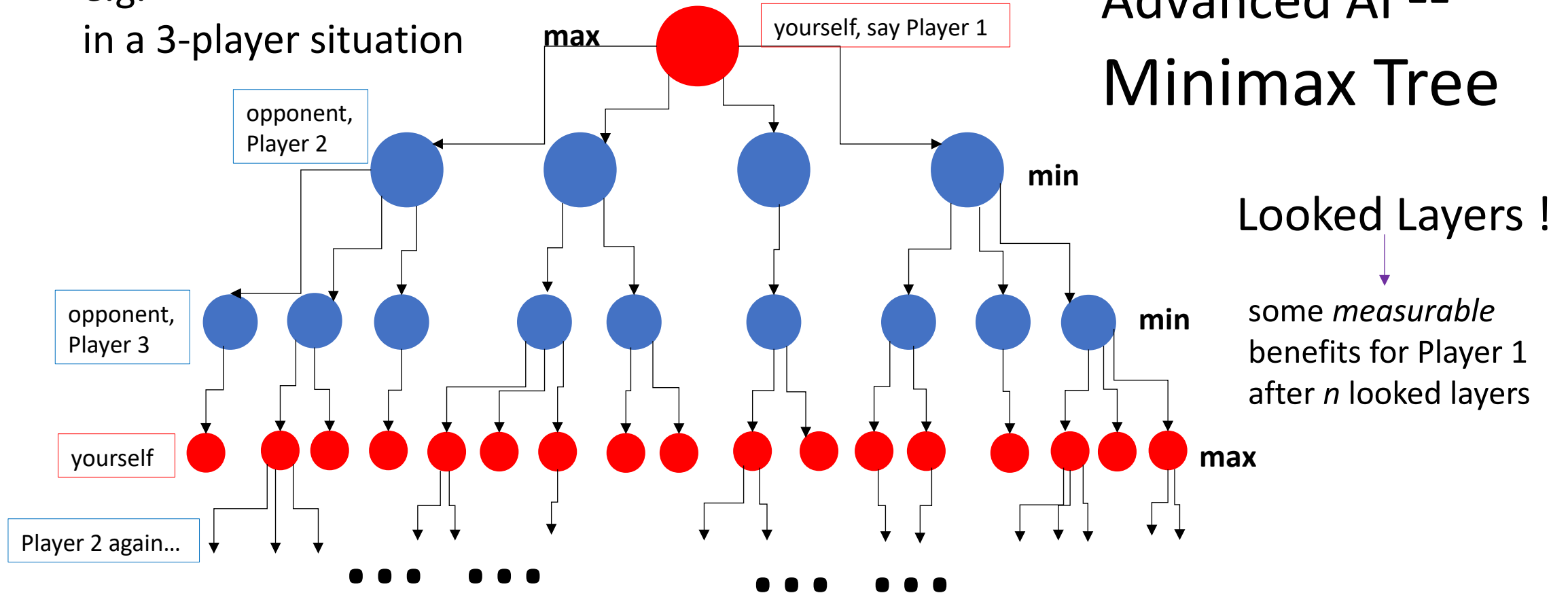
New game can be played by pressing restart (also available throughout game play)





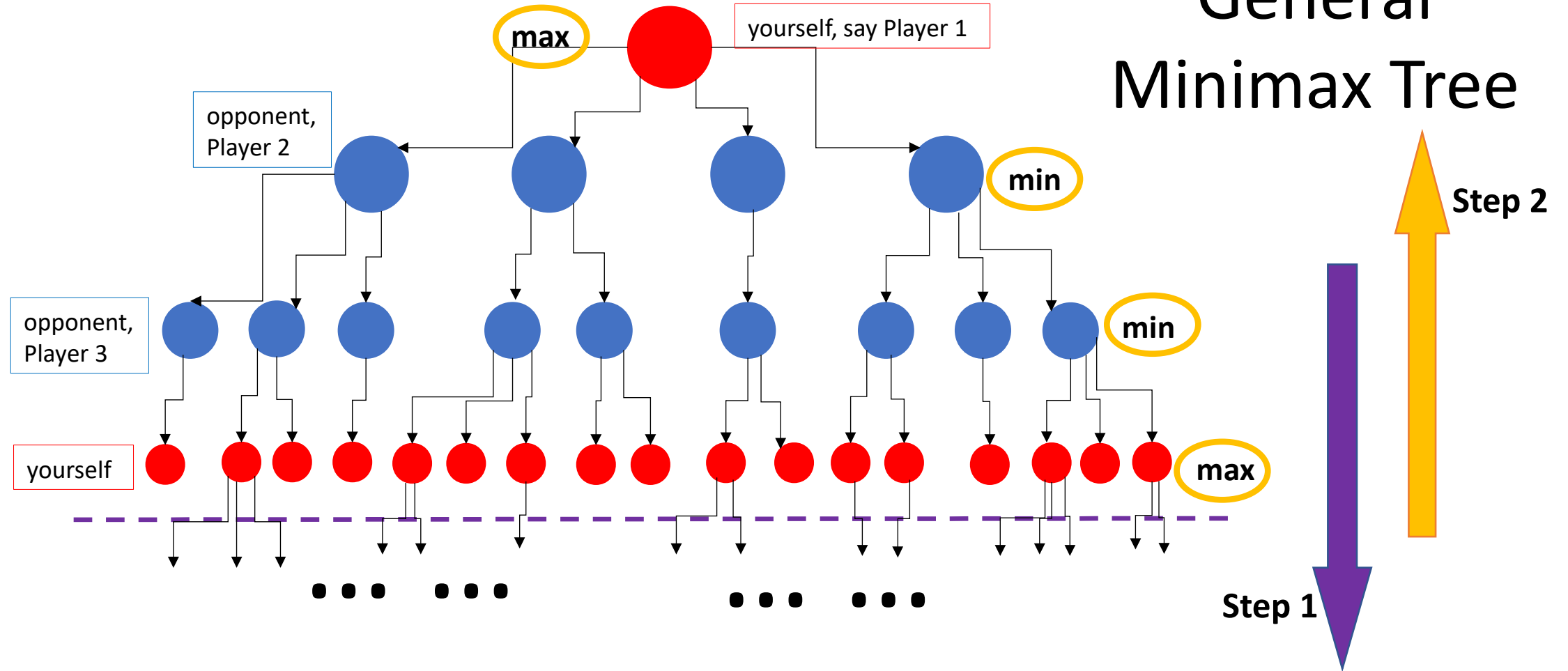
e.g.  
in a 3-player situation

# Advanced AI -- Minimax Tree



1. Both your **opponents** want to **minimize** your possible “benefits”.
2. You **self** want to **maximize** your possible “benefits”.

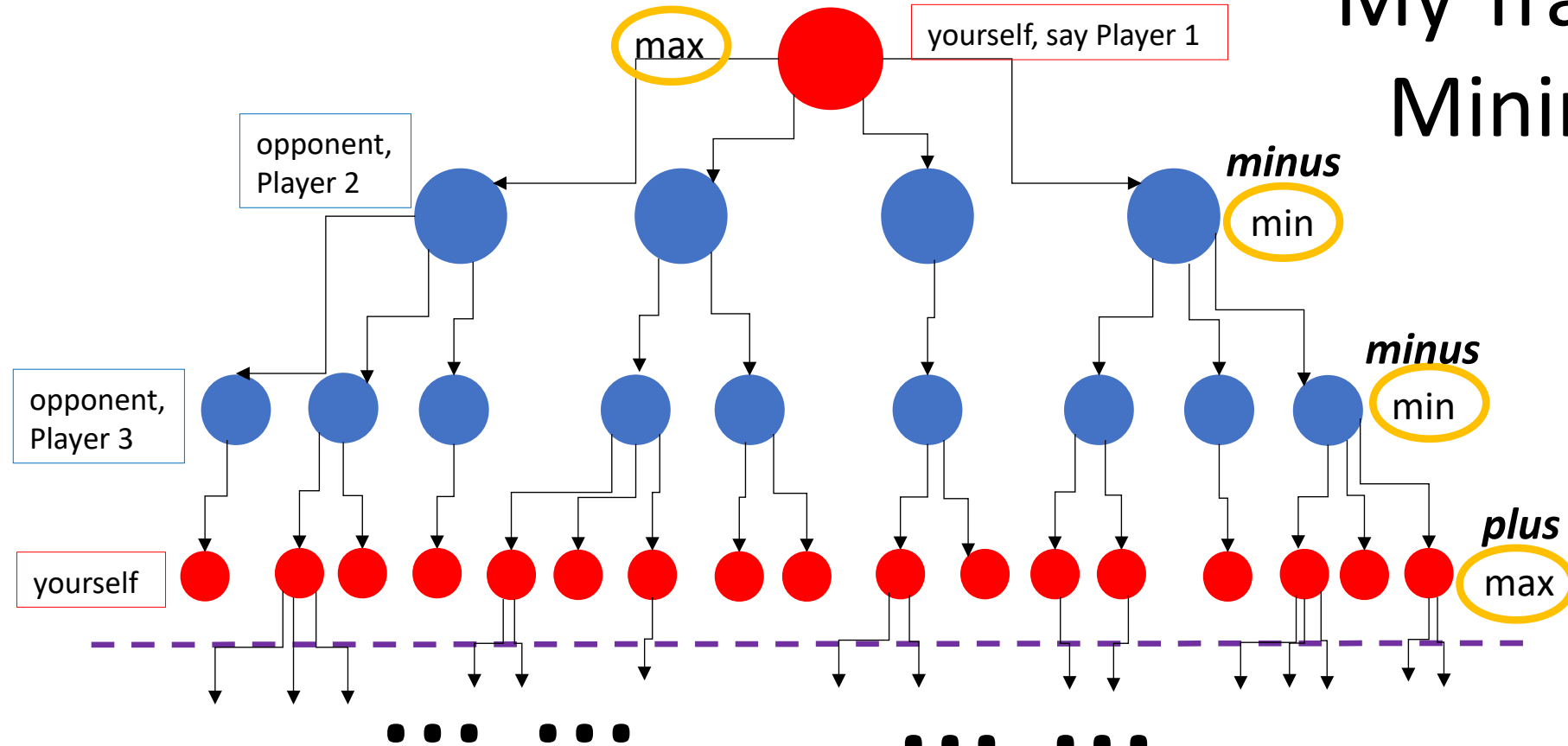
# General Minimax Tree



**Step 1:** Calculate from up to bottom & Get a group of *your* scores after a fixed looked layers

**Step 2:** Use "min" "max" idea to go back to top, and get your final first-step decision

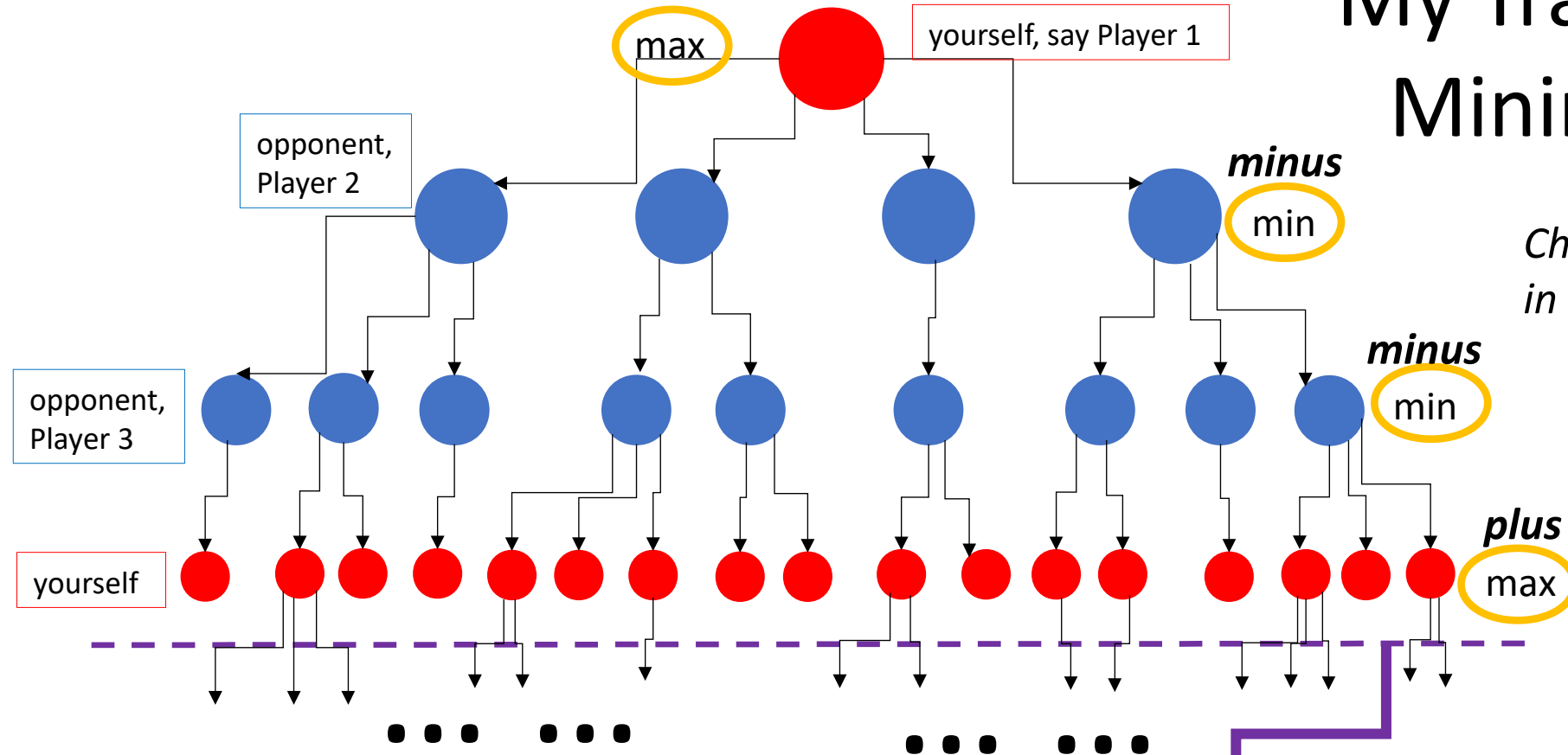
# My Transformed Minimax Tree



Always get the **maximum** score for the corresponding checked player.

high mark for **flag** collection,  
low mark for **more than one card**  
collection at a time

# My Transformed Minimax Tree



*Check all valid moves  
in each looked layer*

Record **move  
sequence, current  
board & updated  
score *through* the  
layer search  
process...**

Always get the **maximum**  
score for the corresponding  
checked player.

For a fixed looked layer, find the  
maximum current score for yourself,  
and its corresponding moveSequence.

***1st move !***