Warring States

made by:
Siying Qian(u6099927)
Lenna Nicholson(u4671448)
Xiaoguang Lyu(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

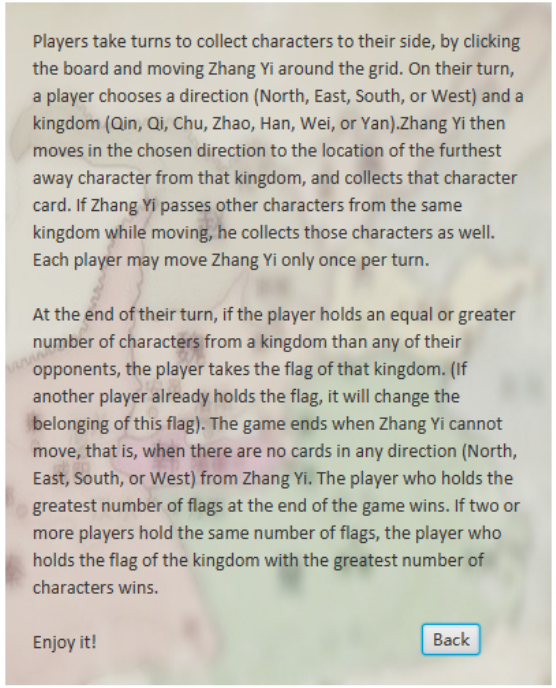
Strengths and weaknesses of our approach:

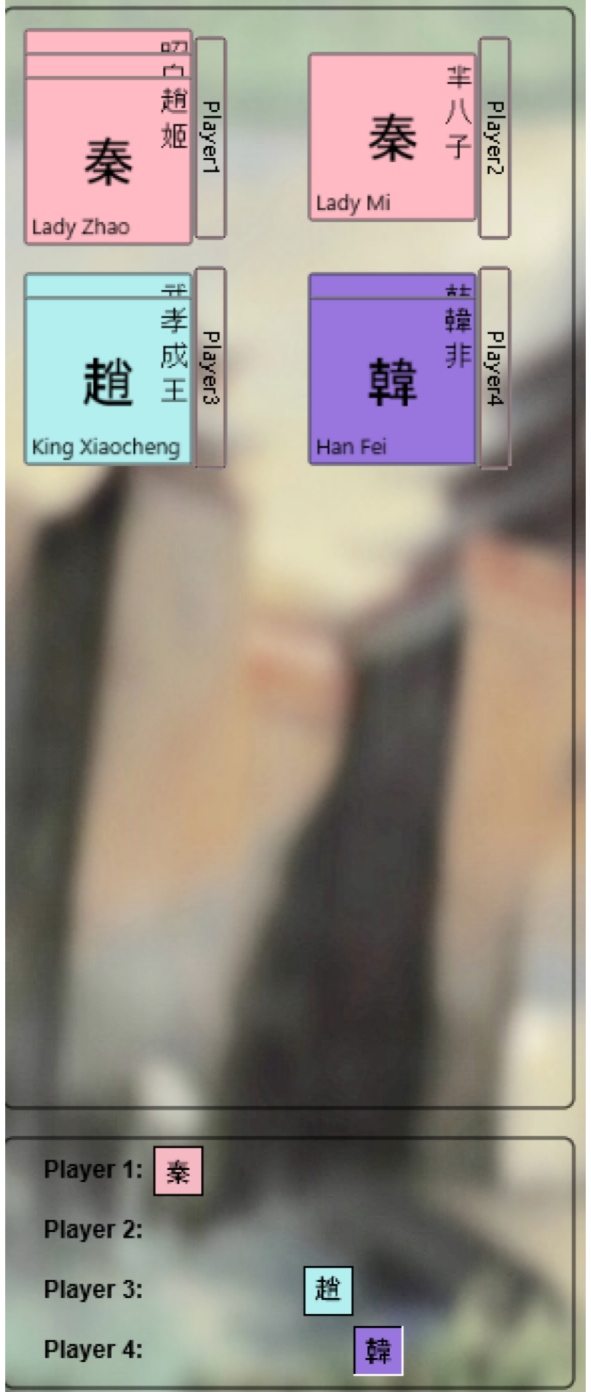
- Because much functionality within javafx, difficult to debug
- Focus on system that works for user

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





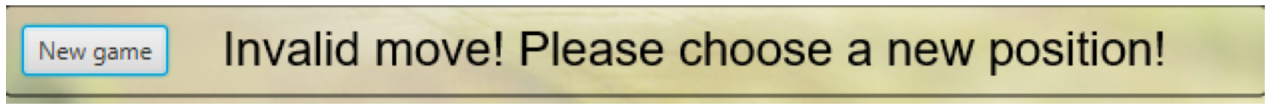
Scoring

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

Invalid move

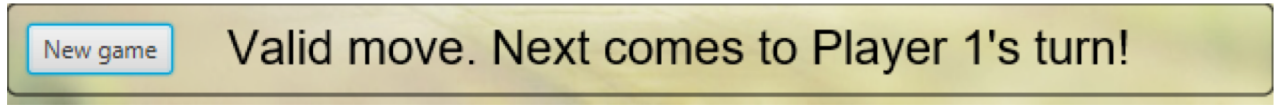
Hover over shows move is invalid

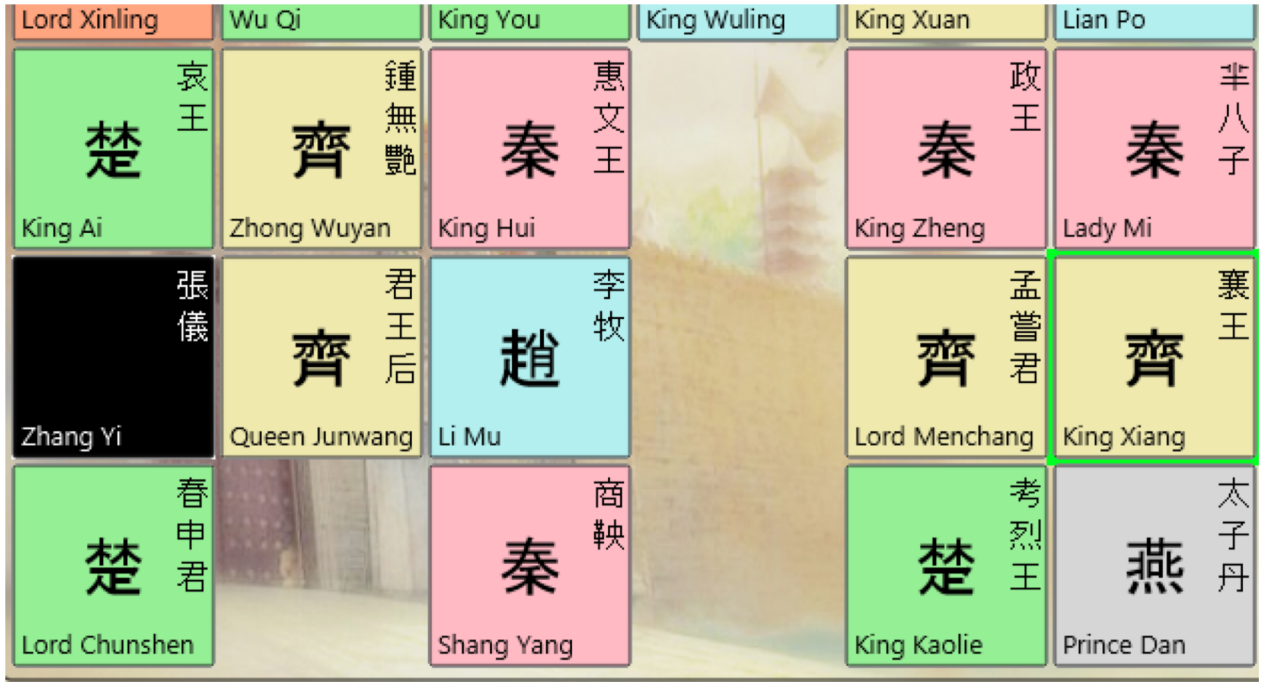
If clicked on, error message shown and sound effect played



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





End of game

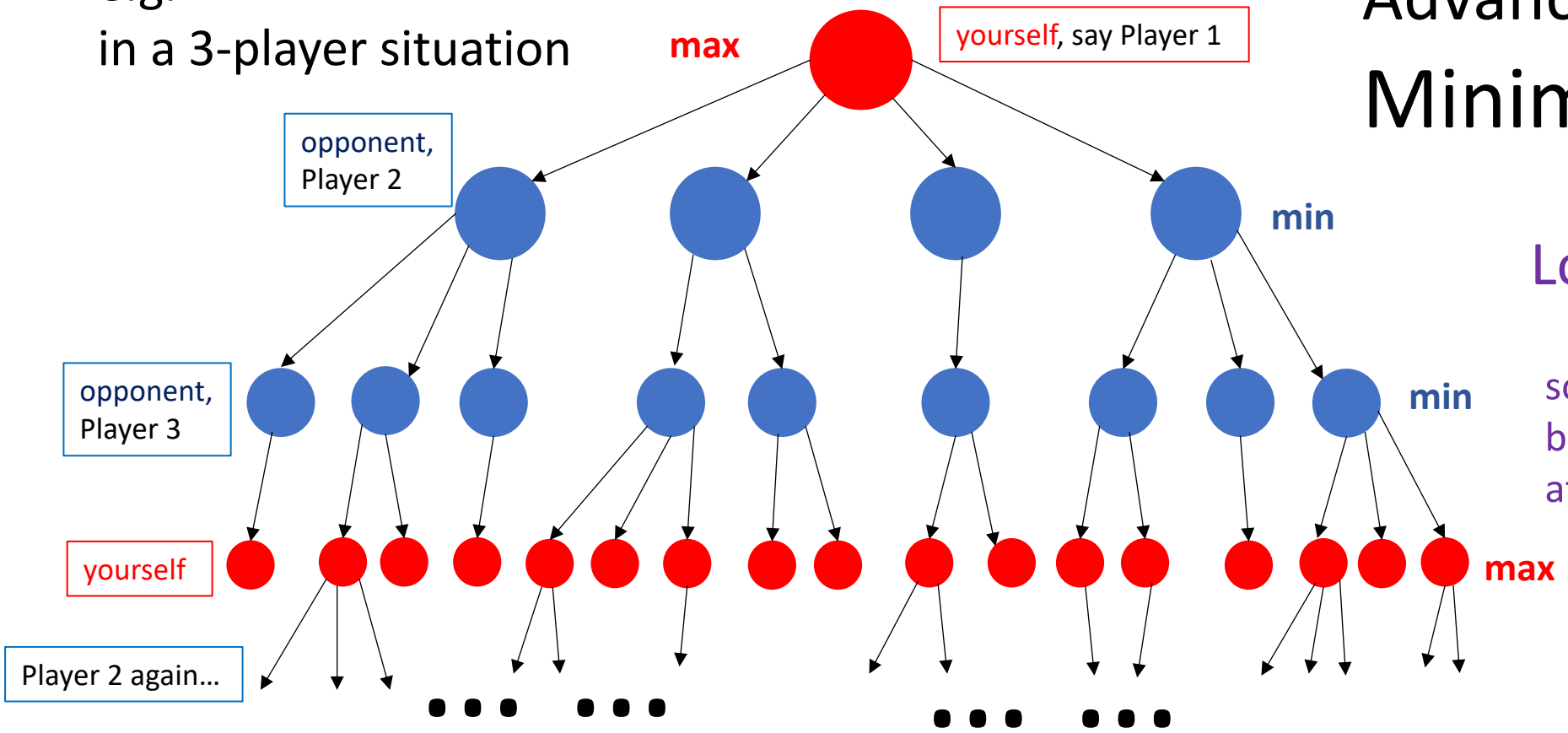
Shows which player won the game

Sound effect played

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

Advanced AI -- Minimax Tree

e.g.
in a 3-player situation

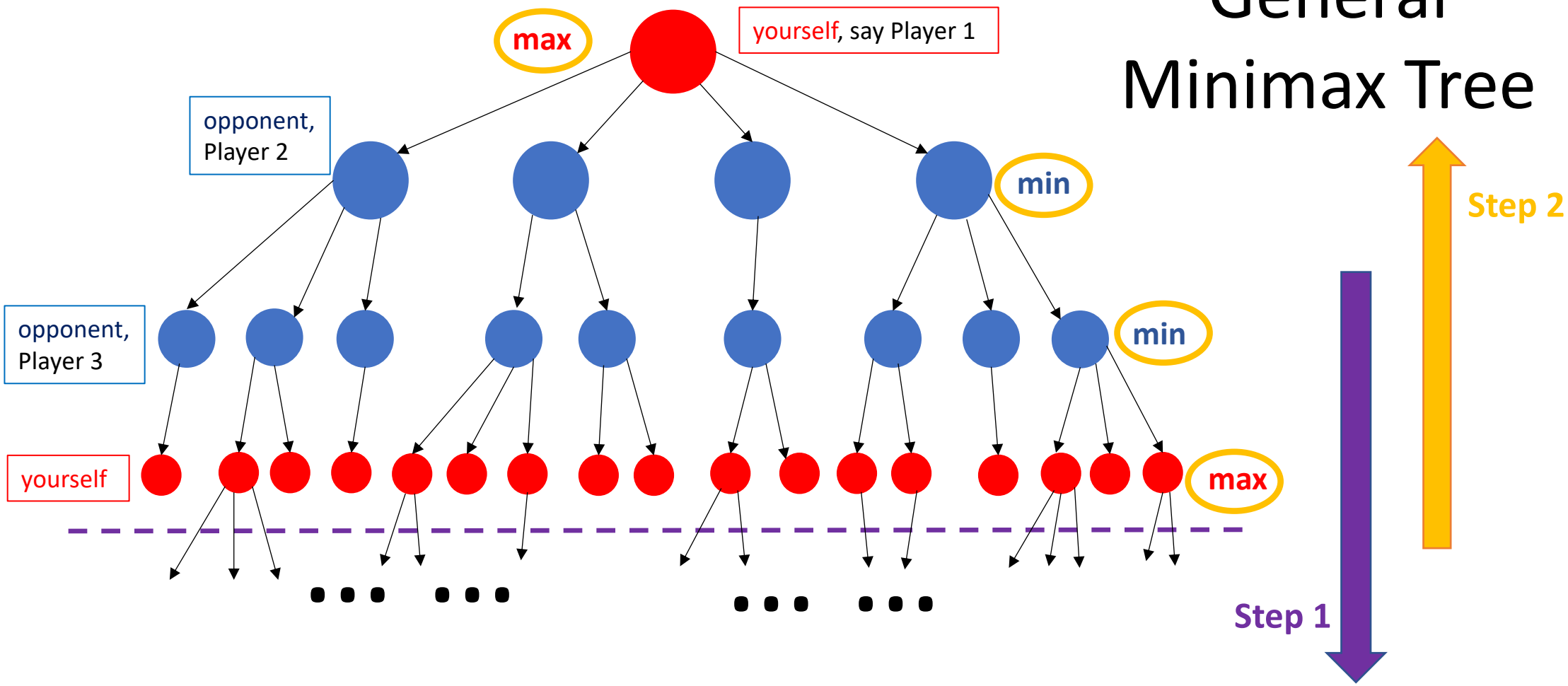


Looked Layers !

some *measurable* benefits for Player 1 after *n* looked layers

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

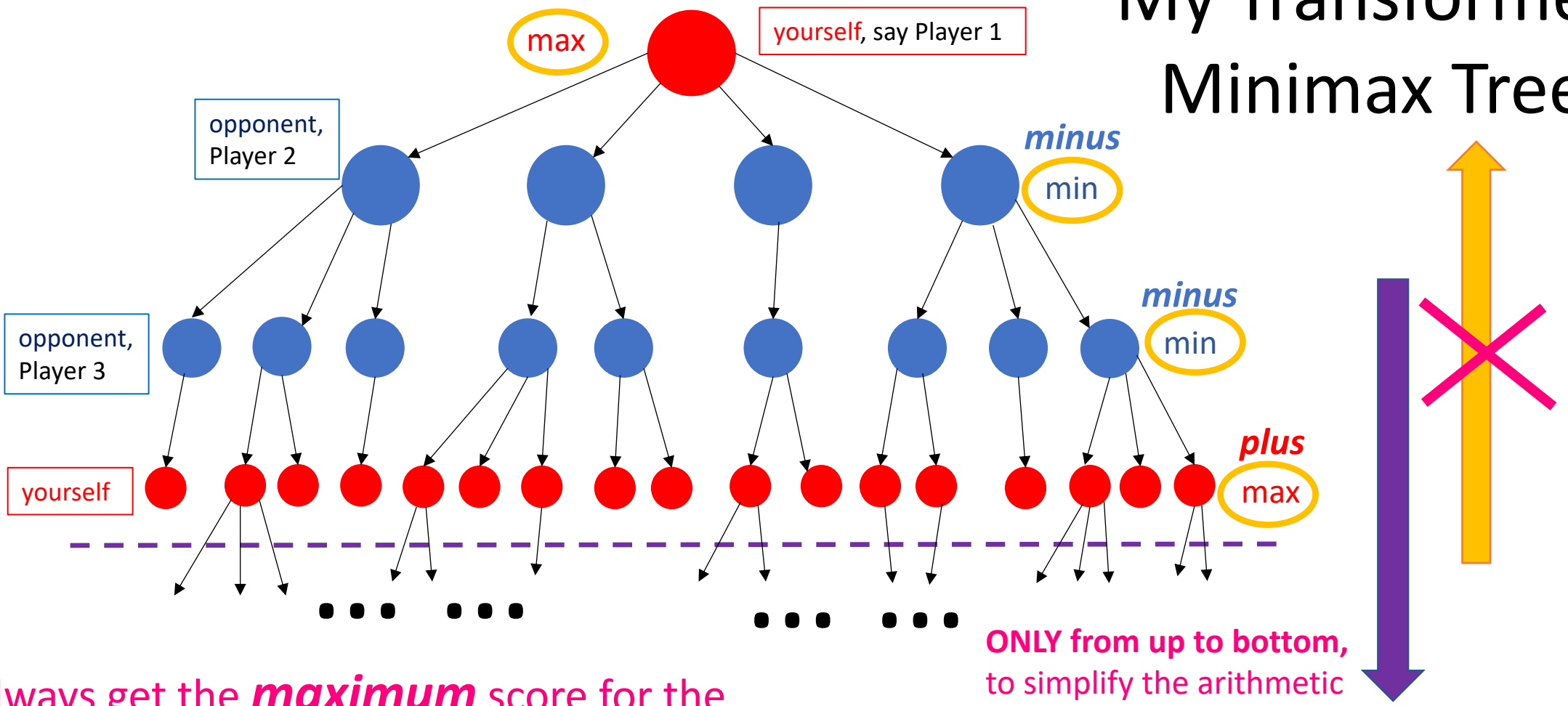
General Minimax Tree



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

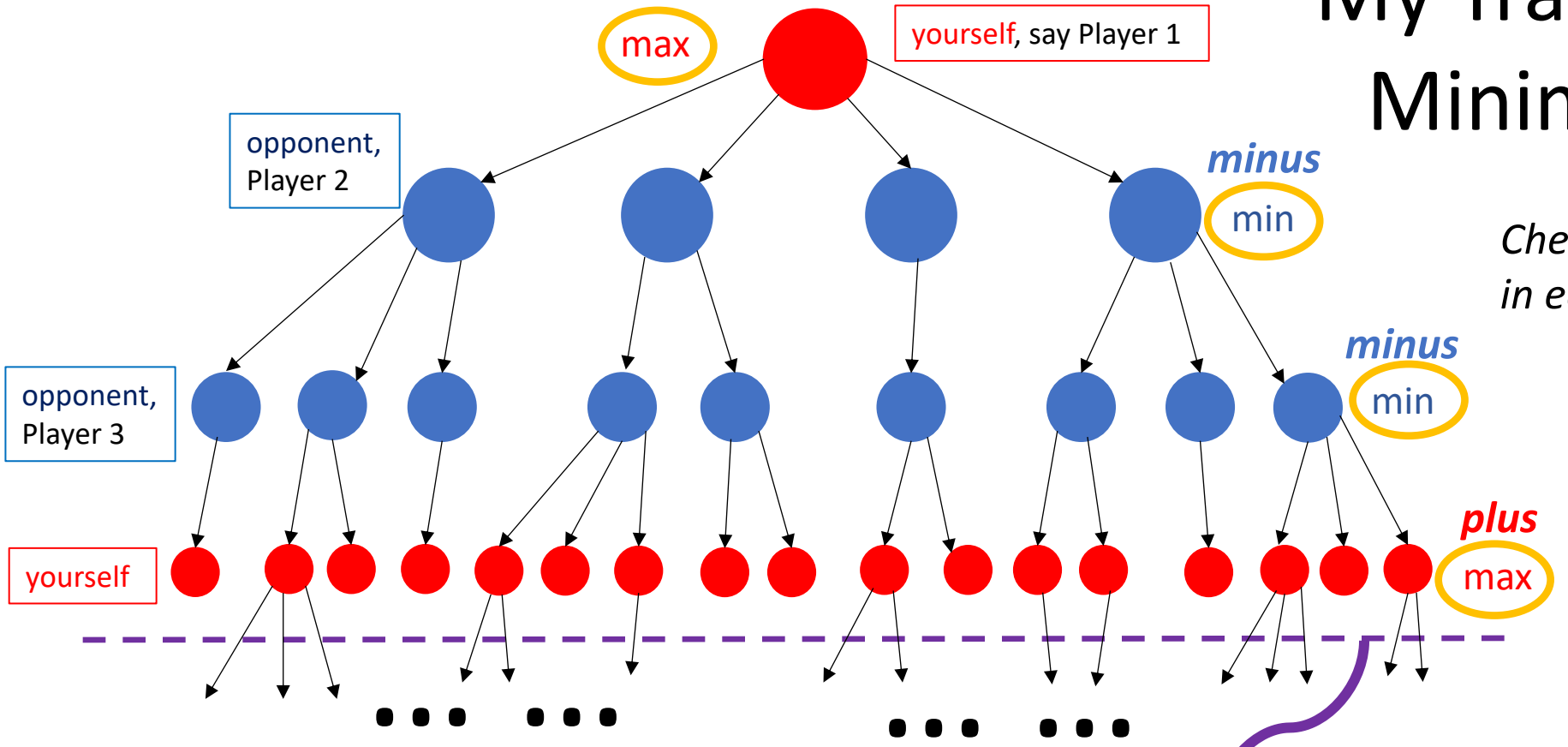
My Transformed Minimax Tree



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 !