Symmetries Tutorial

GamesCrafters 2007

1 January 2007
Yanpei Chen

Symmetries Tutorial

Agenda

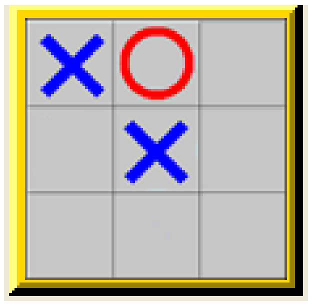
- What are symmetries
- Why are they needed
- Common symmetries
- How to implement symmetries

What are symmetries

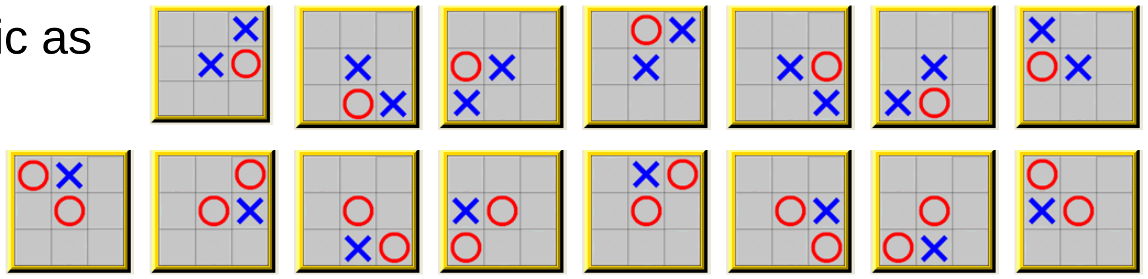
Symmetric positions are

Any set of positions that are the “equivalent” for humans

But “different” for computers



Symmetric as



The left position is the representative “canonical position” of all positions shown

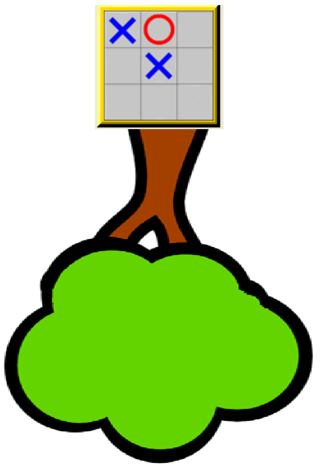
Symmetries depend on game rules

e.g. In Gamesman Tic-Tac-Toe, X always moves first

So piece flipping (second row) does not need to be considered

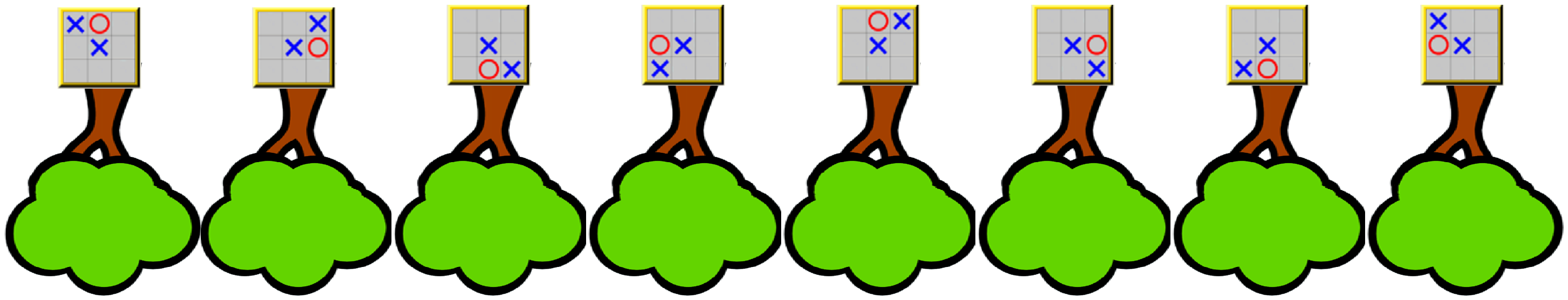
Why do we need symmetries

Computers are not very smart



Human solves

Computer solves



Symmetries lead to huge space and time savings

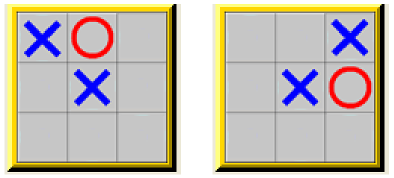
Space savings – we could store only the canonical in the database

Time savings – we could solve only the canonical positions

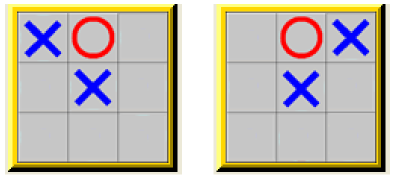
Common symmetries

Geometric symmetries

Rotation

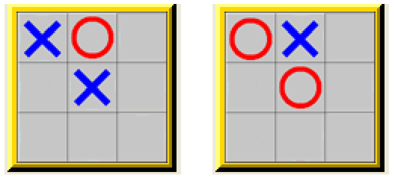


Reflection



Symmetries in pieces

Flipping



Other equivalences in pieces

Other game specific stuff

Spinning – Finding the canonical position

Often the canonical position is the position with the smallest hash value

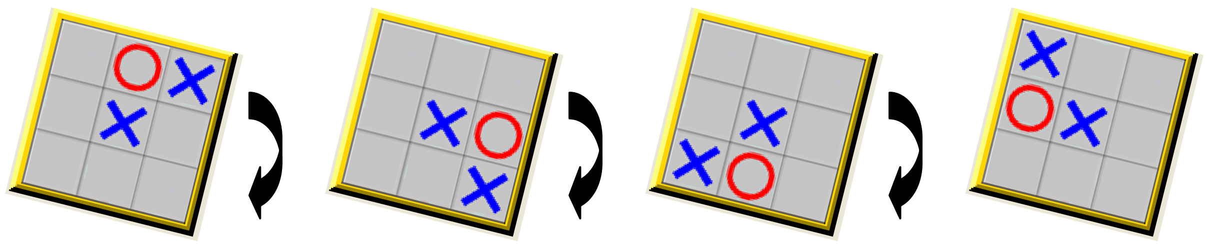
For Gamesman, POSITION is the hash value

So canonical position = smallest POSITION in the equivalent class

Given a POSITION, we find all its equivalent, symmetric “brothers”

This is done by simulating spinning the board around and rehashing

Then we designate board with the smallest hash value as the canonical



Thus, finding the canonical position is often referred to as “spinning”

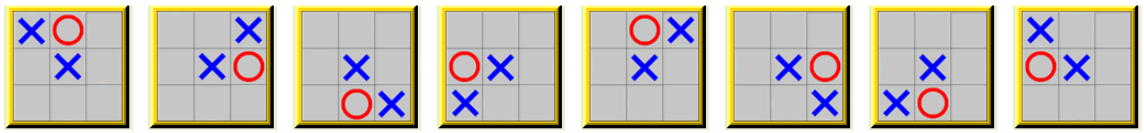
How to implement symmetries

Game modules need to implement only one function

```
POSITION gCanonicalPosition (POSITION p);
```

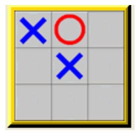
Must adhere to interface

Takes in any position



Performs spinning

Returns the canonical position



Freedom in implementing game specific symmetries

e.g. Quarto!

Multiple implementations

```
POSITION yanpeiGetCanonical(POSITION p);
POSITION marioGetCanonical(POSITION position);
POSITION (*getCanonical)(POSITION p) = &marioGetCanonical;

void InitializeGame() {
…
    gCanonicalPosition = getCanonical;
…
}
```

How to turn on Symmetries

```
----- Main (Pre-Solved) Menu for Quarto -----

      s)      (S)TART THE GAME
      w)      START THE GAME (W)ITHOUT SOLVING

      Evaluation Options:

      o)      (O)bjective toggle from STANDARD to REVERSE
      d)      (D)ebug Module BEFORE Evaluation
      g)      (G)ame-specific options for Quarto
      2)      Toggle (2)-bit solving (currently OFF)
      p)      Toggle Global (P)osition solving (currently OFF)
      l)      Toggle (L)ow Mem solving (currently OFF)
      m)      Toggle Sy(M)metries (currently OFF)

      h)      (H)elp

      q)      (Q)uit
```

Examples

Symmetries implemented in

- Tic-Tac-Toe – Simple; mttt.c
- Bagh Chal – Simple; mbaghchal.c
- Quarto! – Not so simple; mquarto.c
- Others?

Add to this list!!!

Summary

Symmetries = positions that are “same” for humans

Needed because wasteful to solve equivalent positions

Common symmetries = geometric, pieces flipping

Implement using `POSITION gCanonicalPosition(POSITION p)`