# CS 4500
# Software Development

[Project: Tsuro']

October 8, 2019

# The Project

- Build a distributed implementation of a board game
- Software to run games and tournaments for automated players
- Most project assignments:
  - planning
  - specifying
  - coding

# Tsuro’

- Board game for three to five players
- Players place path tiles, move a token along path
- Goal is *not* to exit the game board

# Original Tsuro

# Tsuro' System Overview

- A service for running tournament games. E.g.:
  - Manages players
  - Manages game board
  - Enforces rules
- Players – client programs using the service for playing games against each other

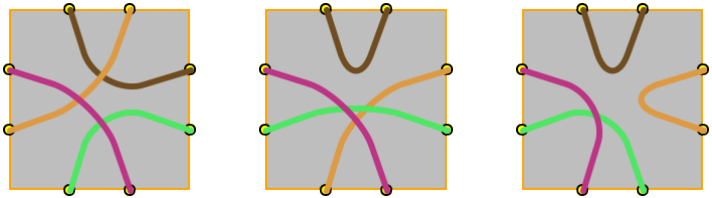
# Tsuro' Game Board

- $10 \times 10$ grid of squares
- Players place tiles
- Indexing: $0 \leq x, y < 10$
- Origin top-left corner, $x$-axis horizontal, $y$-axis vertical

# Tiles

- *Tile*: square with 8 *ports*, 2 per side
- Four sides: *north*, *east*, *south*, *west*
- Four distinct connections between two distinct ports
- Every port: must have exactly one connection

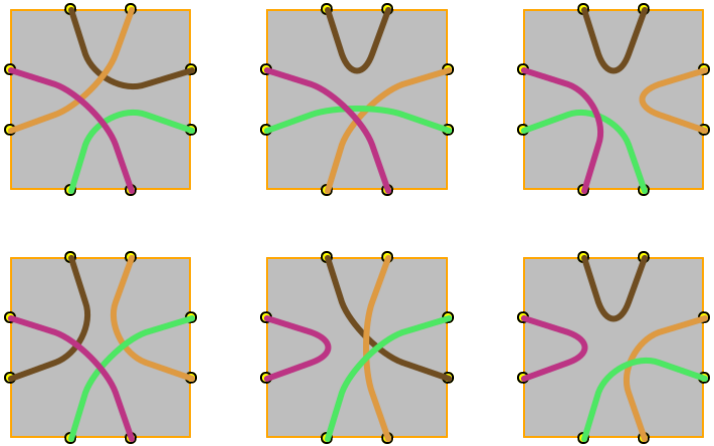
# Tiles



# Tiles

- Tiles may be rotated by 90, 180, or 270 degrees before placing on the board
- Two tiles are *equivalent*
  - if they are the same, or
  - transformed by rotation

# Equivalent Tiles



# Players

- *Player avatar:* colored token
- Colors: white, black, red, green, blue
- To be placed on one of the ports
- If avatar on port: port *occupied*

# Ending the Game

Game ends if one player is left on the board

- Then that player is the winner

# Starting

- Initially empty board
- Oldest player starts (connection age?)
- Each player gets 3 tiles
- Player rotates the tile as desired, places on board
- Places avatar on a port on edge of board
- All players place first tiles and avatars
- Each player – informed of state of the board

# Playing a Round

- Each player takes a turn in each round
- At the beginning: player receives two tiles
  - from the game *referee*
- Player chooses one of the two tiles and places it next to the one occupied by the player's avatar
  - Possibly rotated
  - The other tile is discarded
- This connects the avatar-occupied port to another port on the new tile
- Might connect to already placed tiles that border the new tile

# Playing a Round

- All avatars that face the new tile are moved forward as far as possible until:

(a) they get to a port bordering an empty square; or

(b) they reach the edge of the board
  - Player eliminated

- Round ends when every player has completed a turn

# Ending the Game

Game ends if

(a) there is only one avatar left on the board
- the avatar's owner is the winner;

(b) all remaining avatars reach the board's periphery during the same round;
- the owners of these avatars are joint winners

# Project: First Part

1. Project analysis & Project plan (to be released today, due Friday)
2. Tile analysis and representation implementation (released later this week, due next Wednesday)

# Other

## Repo

- Create a `Tsuro` directory in the top-level of your repo

## README

- Create a `README.md` (can also be `.txt`) in the `Tsuro` directory
- Overall description of the project – its purpose
- Description of the directory structure
  - What is the purpose of each sub-directory?
  - File or group of files? (if feasible)
- Rough roadmap, overview of design, milestones
- How tests harnesses for milestones are run
- How the complete internal unit test suites are run

# Other

## Testing

- Maintain a script/program that can run all unit tests
- Document how to run the script (in README)
- Run the script before you push
- Unit test cases might fail – this is OK during development
  - Can used during development as a reminder what needs to be done next
- Unit tests shouldn't break the top-level script (uncaught exception, segfault, etc.)
  - If a unit test fails, the output should include information to allow fixing the problem

# Group Work Tips

- Read description – meet, discuss and plan
- From the start: how to validate/verify the deliverables of the task
  - if applicable
- Delegate
- Form units – “task forces” (2 people, exceptionally 3) – remember pair programming?
- E.g.,
  - “Task Force A”: designs and implements (initial) tests
  - “Task Force B”: works on a rough draft of design document
- Keep each other updated, reconvene, touch base, merge work
- Reflect in your lab books (no need to write long prose)