CS 4500 Project

Phase 2: Rule Checkers & the Game Board

Due: Wednesday, October 23, 11:59pm

Submission: Place the artifacts in your repository as follows (all paths are relative to the top-level of the project directory):

For the *Design Task*: place `rules.md` in the `Planning/` directory.

For the *Programming Task*: place `board.PP` in the `Common/` directory.

For the *Testing Task*: place `xtiles` and `tile-test` in a new directory called `2/`.

Design Task

For Phase 1, you were asked to write an analysis and plan for a system for running Tsuro games. At the end of this assignment, you will find an example of such an analysis and plan (written by Prof. Felleisen). The description of this phase is based on that analysis.

We will separate the mechanics of running the game from enforcing rules of the game. For this, we will take away responsibilities from the referee component and introduce a separate *rule checker* component. While the referee component will serve as a supervisor of Tsuro games, implementing the mechanics of running a game, the rule checker component will decide the legality of player actions and provide an implementation of the rules of the game.

For the Design Task, design an interface for Tsuro rule checkers. This interface will provide operations for checking legality of tile placements in all phases of the game. Include what information a rule checker will need to perform these operations.

The referee *must* use such a rule checker. A player *may* use it to check if its move is allowed, or it may decide to take its chances and play any which way it wants.

The document must not exceed two pages.

Programming Task

Implement your board data representation, including the following operations:

1. Placing a player's first tile and avatar on the board
2. Adding a tile on behalf of a player whose avatar occupies a place on the board
3. Creating a board from a bunch of *initial* tile & player placements;

   This operation should reject placements that result in boards which contain

   (a) tiles neighboring other tiles,
   (b) tiles which are not placed on the periphery, and
   (c) tiles where the avatar does not occupy a port that faces the board's interior.

4. Creating a state from a bunch of *intermediate* tile & player placements;

You might need other operations to implement these. Consider making those public if you see a need for players or referees to use them.

The board itself cannot enforce rules (this is the role of a rule checker), however, including basic constraints relating to the board's "physical" functionality is possible.

Testing Task

Create a test harness for testing your tile implementation. This should be an executable called `xtiles`. The harness consumes a batch of JSON lines from `STDIN` and produces its results to `STDOUT`.

The input specifies a tile, a rotation and entry port. The output specifies, for the given tile, which exit port is reachable from the entry port.

Each input line has the shape

```
        [tile-index, degrees, port]
```

Where

- `tile-index` uniquely identifies a basic tile configuration. These configurations and indices are listed in the table in Tiles below.
- `degrees` is `0`, `90`, `180`, or `270`
- `port` is one of `"A"`, `"B"`, `"C"`, `"D"`, `"E"`, `"F"`, `"G"`, and `"H"` (see the diagram in Tiles below)

Each corresponding output line of the harness looks like this

```
        ["if ", port, " is the entrance, ", port, " is the exit."]
```

Create at least three tests and save them in `2/tile-tests`.

Tsuro, A Plan

(by Matthias Felleisen)

Goal

The goal is to develop a gaming framework for running Tsuro tournaments for "AI" players running on remote computers.

Let the best hacker win, and we cash in on it.

A tournament will pitch players against other players in several rounds, including knock-out rounds. Each game is supervised by a "referee" to ensure compliance with the rules. A rule-breaking or non-responsive player is immediately eliminated from the tournament.

Software Components

The description of Tsuro suggests the following software components:

- A *player interface* to which the creators of external players program. The player interface must spell out all phases of Tsuro:
  - how to place an initial tile and colored token on the board;
  - how to take turns, i.e., the information a player needs to compute a turn and the information it uses to request an action;
  - how/whether to receive information about the end of a game.

As we work out this complex interface specification, we may need to develop additional concepts.

- Our team should implement at least one *player* to validate the interface.
- A *referee* must supervise a game of players; it may assume nothing about players but the existing interface, that is, all interactions between referee and player components go through this interface.
- The software framework needs components that represent the physical game pieces:
  - *tiles* and their internal graph
  - *colored tokens*, which belong to a specific player
  - *boards* with the desired number of separate squares

  Together with the player interface these pieces make up the common ontology that players and referees use to communicate.

- For dealing with rounds of games, we will need to build a tournament manager that runs rounds of games. The creator of the administrator must know the interface of referees and players; its contact with players should be limited to signing them up and connecting them to a referee for a game.
- The human operators of the referees and players may wish to observe Tsuro games. To implement such viewings, we may need observer components for the players, the referees, or even the administrators.

Building It

Our build plan consists of two phases.

The goal of the first phase is to build a complete prototype in one language that we can demo for our angel investors.

The goal of the second phase is to break up this monolithic prototype so that we can connect the administrator to remote players. We will then be able to demonstrate this to our investors as the alpha release of our product.

For the first phase we propose to build the above components in this order:

- the basic game pieces, tiles and colored tokes, because the player and the referee must use the exact same representation;
- the game board;
- the player interface;
- the referee and the player;
- a *tournament manager*.

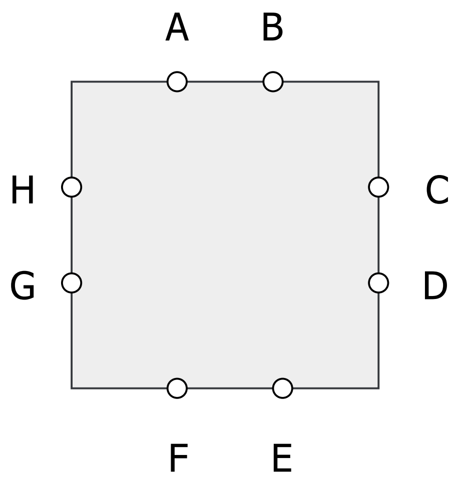
Once we have the interface, we could develop to it with work divided between two teams.

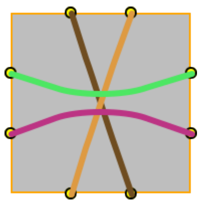
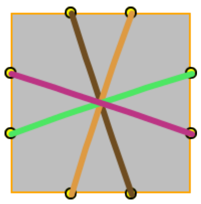
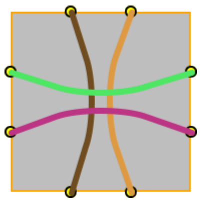
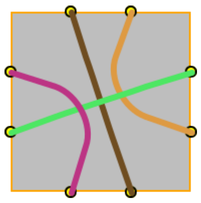
For the second phase, we will use the remote proxy pattern to go from the monolithic prototype system to the distributed version. This will require the construction of

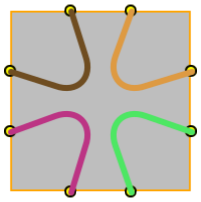
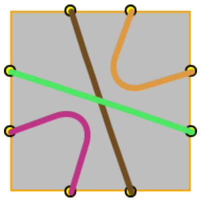
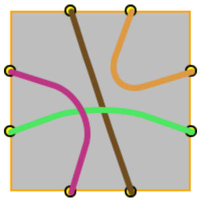
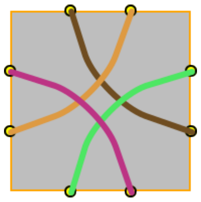
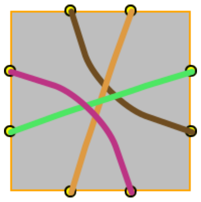
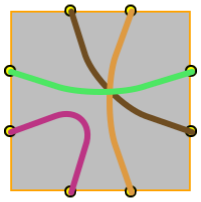
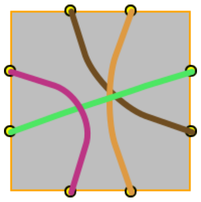
- a remote proxy player
- a remote proxy server
- a communication layer (TCP for now; perhaps UDP for speed, HTML for web access).

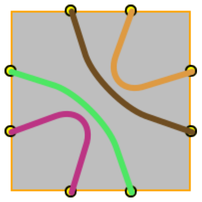
Tiles

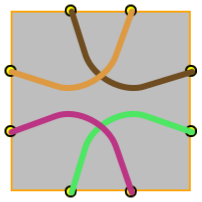
Ports are named as follows:

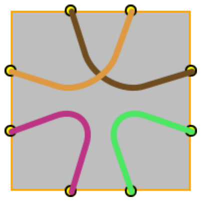


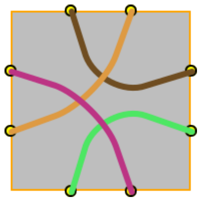
tile-index	Image	... as Text
0		((A E) (B F) (C H) (D G))
1		((A E) (B F) (C G) (D H))
2		((A F) (B E) (C H) (D G))
3		((A E) (B D) (C G) (F H))

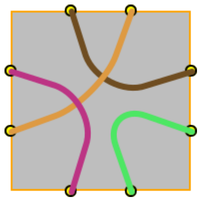
4		((A H) (B C) (D E) (F G))
5		((A E) (B C) (D H) (F G))
6		((A E) (B C) (D G) (F H))
7		((A D) (B G) (C F) (E H))
8		((A D) (B F) (C G) (E H))
9		((A D) (B E) (C H) (F G))
10		((A D) (B E) (C G) (F H))

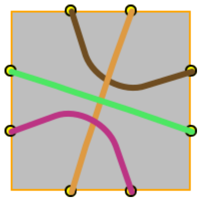
11  ((A D) (B C) (E H) (F G))

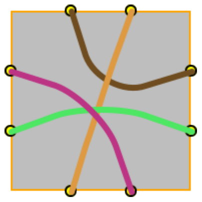
12  ((A C) (B H) (D F) (E G))

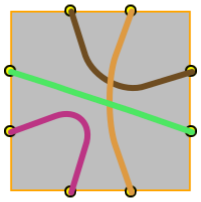
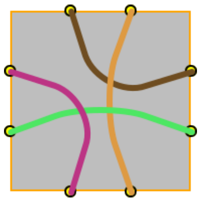
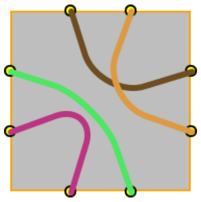
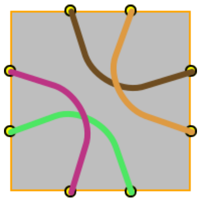
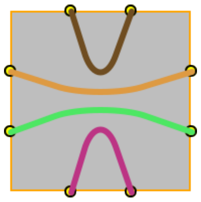
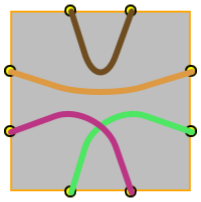
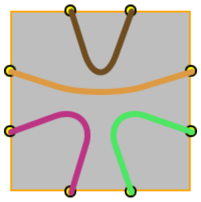
13  ((A C) (B H) (D E) (F G))

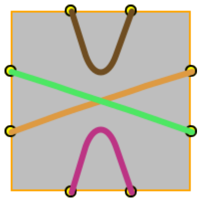
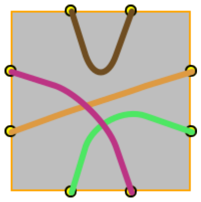
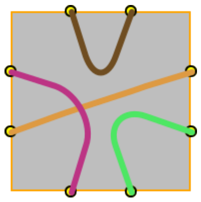
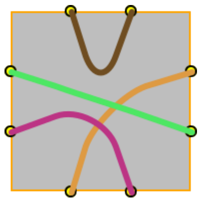
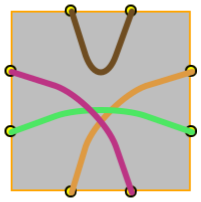
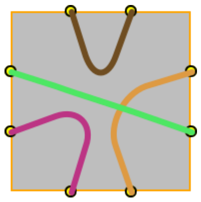
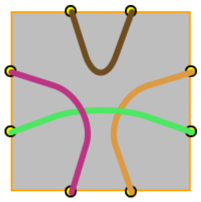
14  ((A C) (B G) (D F) (E H))

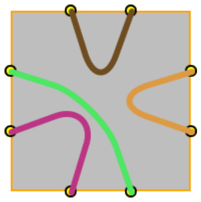
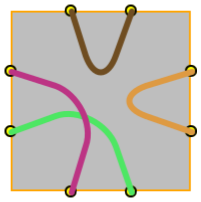
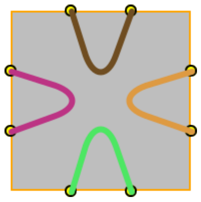
15  ((A C) (B G) (D E) (F H))

16  ((A C) (B F) (D H) (E G))

17  ((A C) (B F) (D G) (E H))

18		((A C) (B E) (D H) (F G))
19		((A C) (B E) (D G) (F H))
20		((A C) (B D) (E H) (F G))
21		((A C) (B D) (E G) (F H))
22		((A B) (C H) (D G) (E F))
23		((A B) (C H) (D F) (E G))
24		((A B) (C H) (D E) (F G))

25		((A B) (C G) (D H) (E F))
26		((A B) (C G) (D F) (E H))
27		((A B) (C G) (D E) (F H))
28		((A B) (C F) (D H) (E G))
29		((A B) (C F) (D G) (E H))
30		((A B) (C E) (D H) (F G))
31		((A B) (C E) (D G) (F H))

32		((A B) (C D) (E H) (F G))
33		((A B) (C D) (E G) (F H))
34		((A B) (C D) (E F) (G H))