CS 4500 Project

Phase 1: Analysis, Plan, Board & Tile Design

Part A – Analysis and Plan

Due: Friday, October 11, 11:59pm

Submission: Create a directory called `Tsuro` in your repository and within this directory create a folder named `Planning`. This is where you place the product of part A of this phase:

- `README.md` or `README.txt`
- `plan.md`, the project description resulting from the design task.

README

Write an initial README file, containing a brief description of the project and the current directory structure.

Design Task

Read the description of Tsuro (included below). These rules are not the original ones; instead they represent how Matthias would like to see the game done.

As the project, we will develop software to run Tsuro games and tournaments for automated players. Your friends may supply implementations for autonomous players, and you will provide the framework for running fair games for such players. The goal is to run tournaments where every signed up player gets a chance to participate in several rounds of Tsuro games, in a manner yet to be determined

Write up a project analysis. The analysis should consist of two parts:

Part 1 describes the identifiable components of your software system. Ask yourself:

- What are the pieces that make up an automated player?
- What are the pieces that make up the game software?

This prose uses “who” and “they” in reference to pieces of software. Think of them as possibly independent “actors” and humanize them temporarily. “who” knows what, “who” needs to know what, and how do “they” communicate

Read the Tsuro’ description carefully, look for nouns referring to actors or potential parts of the system.

Part 2 describes how you plan to proceed about implementing these pieces. Keep in mind that you wish to have “demo” software soon so that a potential client can admire fully working prototypes.

Upload your memo as `plan.md`. The memo must not exceed 2 pages. Use at most one page per part.

Part B – Board and Tile Design

Due: Wednseday, October 16, 11:59pm

Delivery: Use the `Tsuro` directory in your repository to deposit the following files.

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

For the *Programming Task*: place

- `tiles.PP` (or `Tiles.PP`, depending on the conventions in your programming language) in a new directory named `Common` within `Tsuro`
- `allTiles.PP` (or `AllTiles.PP`) and files `tile-1.EXT`, `tile-2.EXT` and `tile-3.EXT` in a new directory named `1` within `Tsuro`.

Here, `PP` is the file extension used by your chosen programming language and `EXT` is an appropriate file extension as specified in the Programming Task below.

Design Task

Artifact: `Planning/board.md`

An implementation of Tsuro clearly demands a data representation for boards. The players, as well as the game server need an interface to the board and a way of referring to things on the board (positions, directions). A player may also need a board representation for strategizing about moves.

Specify a data representation for Tsuro boards with interfaces for players and referees.

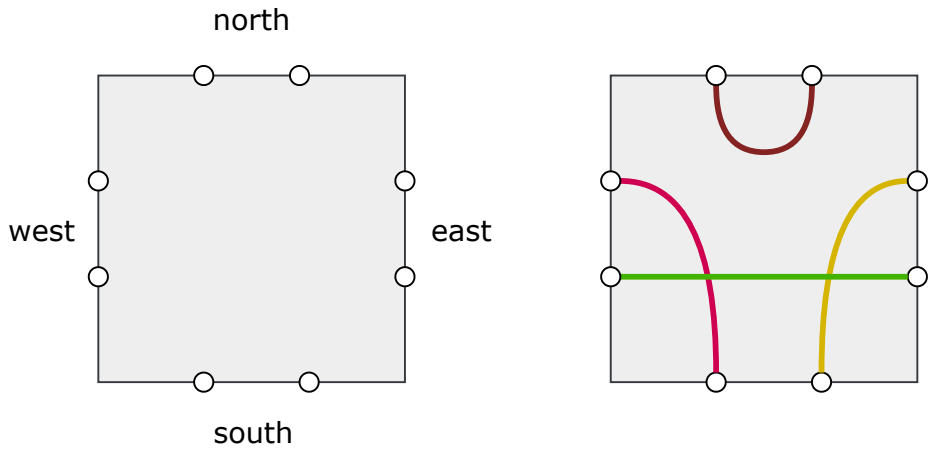
Describe *what* the desired methods/functions should compute without pre-determining *how* they should compute it. A data representation should include data definitions for all related forms of information (in particular, a representation for tiles).

The memo should be up to two pages long

Programming Task

Artifact: `Common/tiles.PP`, `1/allTiles.PP`, `1/tile-N.EXT` (for `N` $\in \{1, 2, 3\}$)

Develop a Tsuro tile representation. Keep your plan and your board specification in mind. A tile has 8 ports, two on each side and each port is connected to exactly one other port. We have determined that there are 35 distinct Tsuro tiles and all others can be obtained from these via rotations by 90, 180, 270 degrees.



a) Implement your data type for tiles.

b) Implement a function for rendering tiles graphically. This function may map a tile (representation) to ASCII art, graphical images (if your PL/IDE support this), decorations found on line, etc.

c) Implement a function that computes the data representations of the 35 unique configurations.

Submit:

i) The implementation in `Common/tiles.PP`
ii) The representations of all 35 tiles in `1/allTiles.PP`
iii) 3 example renderings of tiles as the files `1/tile-N.EXT` (for `N` $\in \{1, 2, 3\}$), using an appropriate value for `EXT`, e.g., `txt` if generating ASCII art; `jpg`, `png`, `svg`, etc. if generating graphical images. The files do not have to be directly generated by your code, but can be screenshots instead. You are welcome to submit more than 3 renderings, but include at least 3.

Note: Your submitted code will be evaluated based on inspection, so make sure it is readable and easy to follow.

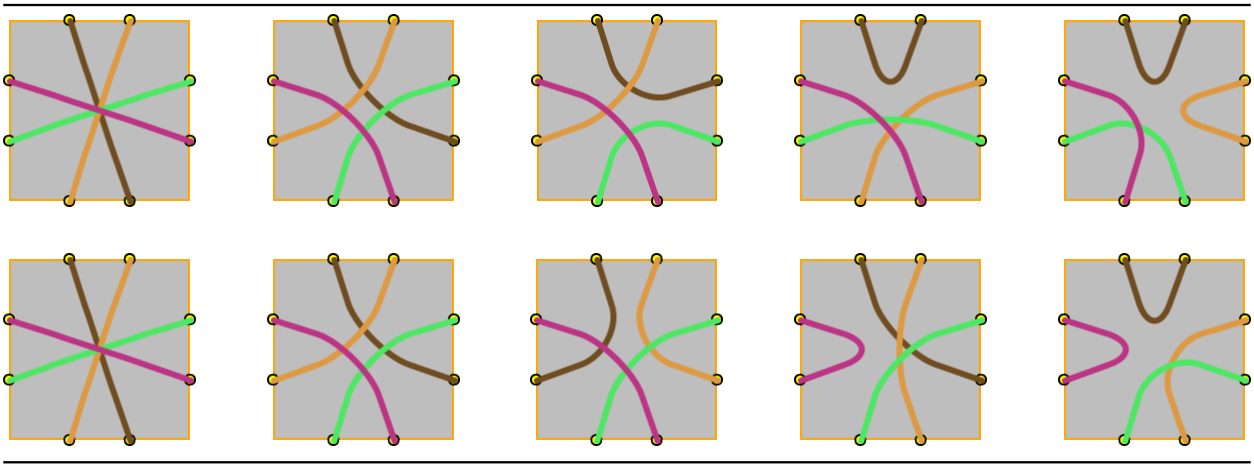
Tsuro'

Overview

Tsuro' is a board game for three to five players.

The *game board* is a 10×10 grid of square holes into which players can place tiles. We use *index* for natural numbers in $[0, 10)$. The origin of the board is the top-left corner, its top-most border is the x-axis, and its left-most border is the y-axis.

Each *tile* is a square with 8 *ports*, 2 per side. The four sides are called *north*, *east*, *south*, and *west*. Each tile specifies four distinct connections between two distinct ports; every port must have exactly one connection. Here are some examples:



During a game of Tsuro, a player may rotate a tile by 90, 180, or 270 degrees before placing it on the board. Two tiles are *equivalent* if they are equal or one can be transformed into the other via a rotation. The second row of the above image shows what a 270-degree rotation of the corresponding tile above looks like.

A *player avatar* is a colored token (white, black, red, green, blue) that represents a player on a port of one of the placed tiles. We call such a tile and port *occupied*.

The Goal of the Game

The goal of a round of Tsuro is to be the player with the last avatar on the game board.

Starting the Game

The board is initially empty.

Starting with the oldest player, each player receives three tiles, chooses one, rotates it as needed, places it on the board bordering the periphery on at least one side, and adds an avatar to a port that faces an empty board square.

The initial placement continues with the next-oldest player until the youngest one has placed an initial tile and avatar. Whenever it is a player's turn to place a tile and an avatar, it is informed of the current state of the game board.

This start-up round ends when all players have places their one tile and their avatar on the board.

Playing One Round

Starting with the oldest one and continuing in decreasing age, each player takes a turn to complete one round.

At the beginning of a turn, a player receives two tiles from the game referee. The player chooses one of the two given tiles and places it—possibly rotated in the above-mentioned manner— next to the one occupied by the player's avatar. This action connects the avatar port on the currently occupied tile to another port on the new tile and, perhaps transitively, to already placed tiles that border the new tile. All avatars that face the new tile are moved forward as far as possible until

1. they get to a port on a tile that borders an empty square; or
2. they reach the periphery of the game board, in which case they—and their owners—are eliminated from the game.

The other tile is discarded.

The round ends when every player has completed a turn.

Ending a Game

A game ends when

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.