CS 4500 Project

Phase 4: Implementing the Referee and Players

Due: Wednesday, November 6, 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 `observer.md` in the `Planning/` directory.

For the *Programming Task*: place

- `player.PP` (or `Player.PP`) in a new `Player/` directory within `Tsuro`
- `referee.PP` (or `Referee.PP`) in a new `Admin/` directory in `Tsuro`
- if applicable: `player-interface.PP` (or `PlayerInterface.PP`) in `Common/`

For the *Testing Task*: place `xrules`, `rules-tests`, and `README-tests.md` in a directory named `4/`.

Design Task

We would like to be able to watch the progress of a game. Design an interface for an "observing component" which should get updated by the game system and provide renderings of the current game state. You can also include mock-ups of the user interface/renderings for different stages of the game.

Programming Task

Implement a player and a referee. An example interaction sequence between them is provided at the end of this assignment.

Referee

A referee should run a complete Tsuro game for a list of players, arranged in decreasing order by “age” (representing a first-come-first-serve system).

The referee assigns each player a color from the sequence `"white"`, `"black"`, `"red"`, `"green"`, `"blue"`.

For now, the component should use tiles in the order given by their tile-index and cycle through them as necessary.

When a player requests the placement of a tile during its turn, the requested placement must not conflict with the given information and the tile placement must be checked as legal by the rule checker.

If a player’s tile placement causes the player’s exit from the board and if the player is forced to commit suicide because all new edges force it out, the player gets removed as if it had exited the board through the action of some other player and the chosen tile gets placed.

Player

A player should, when prompted by the referee, provide its initial placement for the initial move and intermediate placements when taking turns. The component should be parametrized by a *strategy*, which dictates how placements are chosen by the player. For testing purposes, provide a deterministic strategy, called `dumb` (or `Dumb`), which

1. For an initial placement, uses the third given tile without rotation and searches for the first legal spot available, clockwise, starting from $(0,0)$. To place the avatar, it searches, clockwise, for the first port that will face an empty square of the board and is connected to the edge of the board. The avatar should be placed on that port’s path (on the edge port or the port facing the empty square – at the end of the turn the avatar should be considered as being on the port facing the empty square).
2. For an intermediate placement, it uses the first given tile without rotating it and places it on the square adjacent to the player’s avatar. It does not check the legality of this action.

Feel free to implement a better strategy as well.

If applicable to your language, describe the complete player interface in `player-interface.PP` or `PlayerInterface.PP` in the `Common` directory.

Testing Task

Develop a test harness for the rule checker's capability of evaluating the legality of a turn.

The input is a JSON array of initial and intermediate placements as in Phase 3 (describing the current state of the board), followed by a turn specification:

```
  [ [color, tile-index, rotation, x, y], tile-index, tile-index ]
```

The turn specification contains: the player requesting the placement, the chosen tile index, its rotation and requested position, as well as tiles that were provided to the player by the referee.

The outputs are `"legal"` and `"illegal"`.

Create five tests and place them in the directory `rules-tests`. The test harness can accept input from STDIN or it can take a filename as an argument. You *do not* have to implement both options, but you need to specify which your test harness uses. Provide a `README-tests.md` with instructions and an example run of your test harness.

Player-Referee Interaction Sketch

Once sufficiently many players connect to the game, this is how we envision a referee-players interaction proceeding.

```
referee             player: P-1 ... ... player: P-N
       |                   |                    |
       |------------------>|                    |   playing as: color
       |                   |                    |
       .                   .                    .
       |                   |                    |
       |-------------------|------------------->|   playing as: color
       |                   |                    |
       |-----------------> |                    |   other players: colors
       |                   |                    |   % the colors of the other players
       .                   .                    .
       |                   |                    |
       |--------------------------------------> |   other players: colors
       |                   |                    |
       |-----------------> |                    |   initial: other placements
       | <================ |                    |   tile degrees port coords
       .                   .                    .
       |--------------------------------------> |   initial: other placements
       | <===================================== |   tile degrees port coords
       .                   .                    .
       |                   |                    |
       |-----------------> |                    |   take turn: other placements
       | <================ |                    |   tile degrees coords
       .                   .                    .
       | -------------------------------------> |   take turn: other placements
       | <===================================== |   tile degrees coords
       |                   |                    |
       .                   .                    .
       .                   .                    .
       .                   .                    .
```