A Plan for Building Santorini

Version 7.0.0.20

Matthias Felleisen

September 22, 2018

Goal

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

A tournament will pitch every player against every other player in a "best of" game series, length to be determined. Each game is supervised by a "referee" to ensure compliance with the rules. A rule-breaking player is eliminated from the tournament, and all of its past games are awarded to the opponent.

Software Components

The description of Santorini 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 Santorini:
  - how to place workers 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 player interface.
- A *referee* must supervise a game of two players; it may assume nothing about players but the existing interface.
- The software framework needs components that represent the physical game pieces:
  - *board* with a number of separate squares

  - *workers*, which belong to a specific player
  - *buildings*, which may be as tall as four floors but only three-story buildings can be occupied for a winning move

- For dealing with an arbitrary number of players, we will need to build a tournament manager that runs all players against each other.

Building the Components

The first goal of our project has to be a framework for connecting AI players on a monolithic basis to each other. To this end, we propose to build the above components in the following order:

- the basic game pieces, because the player and the referee must use the exact same representation;
- using these we can specify the complete player interface and express the various pieces of information as data;
- both the referee and the player must be developed to the common interface, so we might be able to divide the work between two teams;

  Note At this point we can demo a prototype to possible sponsors.

- With the above components, implementing a *tournament manager* should be a straightforward task, both in a monolithic as well as a distributed form:
  - Building the former will probably be most of the work for ...
  - .. building the final, distributed product.

  A remote-proxy pattern should smoothen the transition from the first to the latter.