Petri (Game)

Alex Heinz, Paul Martin, and Alex Rozenshteyn

Project Overview:

Vision Statement: Petri is a Mac OS X application based on the pen-and-paper game "fungus," in which two or more fungus-like organisms vie for territory control in a grid-like petri dish. Players take turns placing randomly-generated one- to five-cell pieces in the dish, attempting to surround regions of the opponent's fungus on two sides, to capture territory and, eventually, eliminate their opponents from the dish.

Feature List:

- One to four players
  - Computer AI Players for single- and multiplayer games
  - Team games
  - Networked and single-computer (hotseat) multiplayer
  - Automatic local-network server discovery with Bonjour
- Animated graphics using CoreAnimation
- Rules variations (e.g., hexagonal game grid)
- Game recording and replays

Domain Model (Rules): Fungus is typically played on a variable-sized square grid, called the "dish." When the game begins, the dish is empty, save for one cell controlled by each of the players in the game, spread around the dish (usually near the corners.) The starting cell is known as the player's "head," and if it is captured or killed (see below) the corresponding player is eliminated from the game. Gameplay proceeds in turns; a typical turn is described below. Note that this description represents gameplay under normal fungus rules, and that variations may apply.

1. When the player's turn begins, he or she is given a "piece," initially off the board, awaiting placement. Pieces consist of arrangements of cells, and can occupy any of the following:
   - *i.* one cell
   - *ii.* two adjacent cells
   - *iii.* three cells in an L-shape or straight line
   - *iv.* any possible tetromino region (think Tetris pieces)
   - *v.* five cells in a straight line

2. Before placing the piece, if the player has any "bites," he or she may use them to attack an opponent. Bites are accumulated over time, or can be picked up from certain, marked locations in the dish. Bites work as follows:
   - *i.* the player selects a cell to bite, ensuring the cell meets the following requirements:
     - *a.* the cell is controlled by an opponent, and
     - *b.* the cell is horizontally or vertically adjacent to a cell controlled by the player
   - *ii.* if the player has "big bites" or "monster bites," the player may select additional cells controlled by the opponent: up to two for

big bites, or three for monster bites, provided the selected cells all lie in a straight line—the size of a player's bite is determined by the amount of territory the player currently controls

iii. all cells selected in this manner are killed (see "killed cells," below)

Players may use any number of bites in one turn, provided they have enough, but they may not bite after they have placed their piece.

3. The player attempts to place the piece he or she has been given at a location in the dish. The player may rotate (but not flip) the piece before placing it, but the piece may only be placed at a location fitting the following requirements:

i. the piece must lie entirely within the bounds of the dish, (i.e., all cells comprising the piece must correspond to a grid location)

ii. the piece cannot overlap any opponent's cells, and

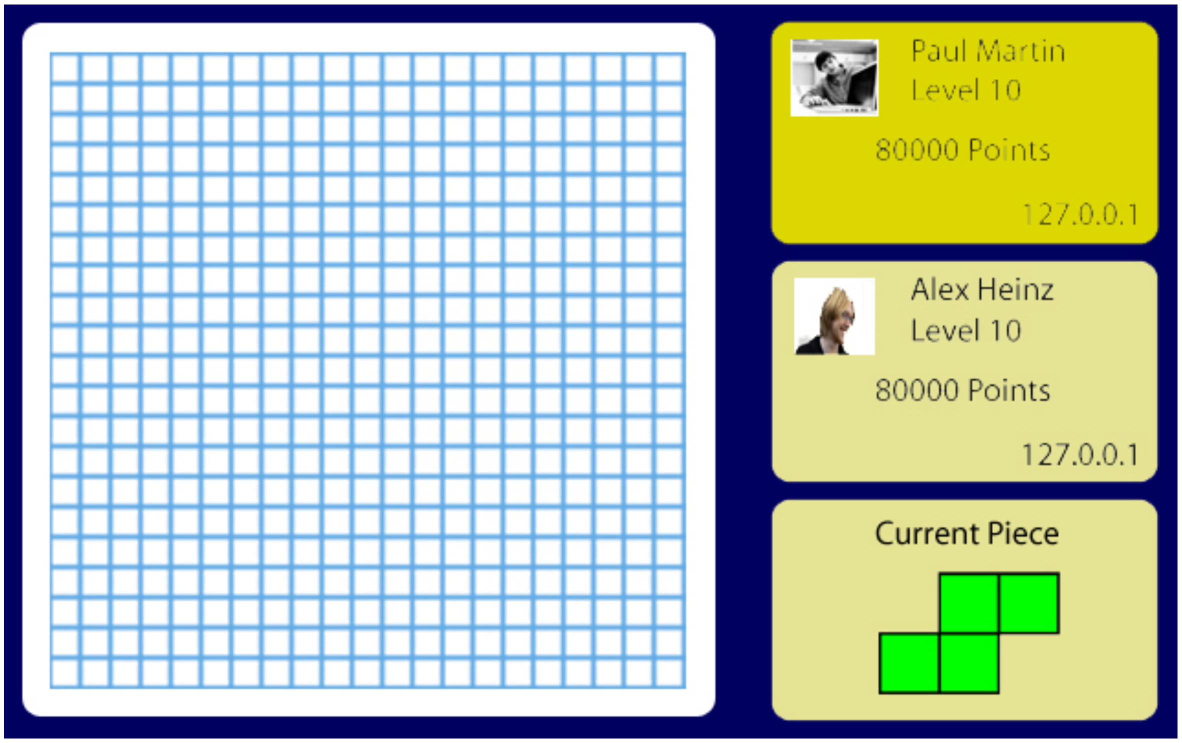
iii. at least one of the cells in the piece must be horizontally or vertically adjacent to a cell in the dish already controlled by the player

Placed pieces are added to the player's territory, and when the piece is placed, the player's turn ends. If the player cannot place the piece, he or she may choose to skip his or her turn.

The object of the game is to eliminate all opponents; elimination occurs, as stated before, when a player's head is captured or killed. Captures are the primary means of gaining territory in the game, while kills provide a more direct means of attacking opponent, without personal gain:

- Cells are captured by surrounding the cell, or a straight line of cells, on two sides, horizontally, vertically, *or* diagonally. Any cells surrounded this way are given to the capturing player. Captures *are* recursive: capturing a cell that would surround additional cells also captures the additional cells. Capturing a player's head captures *all* of the player's territory.
- Cells are killed when they are bitten, or when they are separated from a player's head, (either via a bite or a capture) such that no contiguous region of cells connects the cell to a player's head—diagonals are not considered connected. Killed cells are removed from the board, leaving the territory empty.

GUI Sketch:



Actors:

• *Human player:* human players constitute all users: the primary entity that will interact with the software. Players can play in games i.e. *make moves*, *leave games*, and in networked multiplayer games, *chat*. Players can be on the *host* or on the *client* machine.

- Host players can *start networked games*, *set the maximum number of players* in a game, *kick players*, and *add AI players.*
- Client players can *join networked games.*

• *AI player:* AI players serve as replacements for human players in games which require additional players to fill slots, such as single player games. They can *make moves* but cannot chat in (or choose to leave) games.

Use-cases:

• Player starting a single player game:
  - Open “Start game” dialog
  - Select “Single player game”
  - Select number of AI players
  - Configure game rules
  - Play game

- Player starting a offline multiplayer game:
  - Open “Start game” dialog
  - Select “Multiplayer game”
  - Select “Offline”
  - Select number of AI/human players
  - Configure game rules
  - Play game
- Player starting a network multiplayer game:
  - Open “Start game” dialog
  - Select “Multiplayer game”
  - Select “Networked”
  - Select maximum number of players
  - Configure game rules
  - Wait for players to join
  - Play game
- Player joining a hosted game:
  - Open “Join game” dialog
  - Select game to join
  - Wait for game to start
  - Play game
- Player on a client machine leaving a game:
  - Select “Leave game”
- Host ending a game:
  - Select “End game”
- Host kicking players:
  - Select player
  - Select “Kick”

Architecture:

Model: Each game is represented by an encapsulated model object, which is attached to the controller and view via a listener pattern that uses Objective-C’s key-value observation patterns. During network games, the application acting as the game’s host holds the model objects, and presents them to the client applications via Cocoa’s Distributed Objects API.

Controller/View: The view presents the model to the user via a board, pieces and player icons rendered as CoreAnimation layers, with Quartz Composer animations for the contents of the cells of the board, giving them a shifting organic appearance. Player input is handled with drag-and-drop actions.

Resources:

Platform: Mac OS X 10.6+
APIs: Cocoa, Distributed Objects, CoreAnimation, Bonjour
Development Tools: Apple Xcode, Interface Builder, Instruments
Animation: Quartz Composer