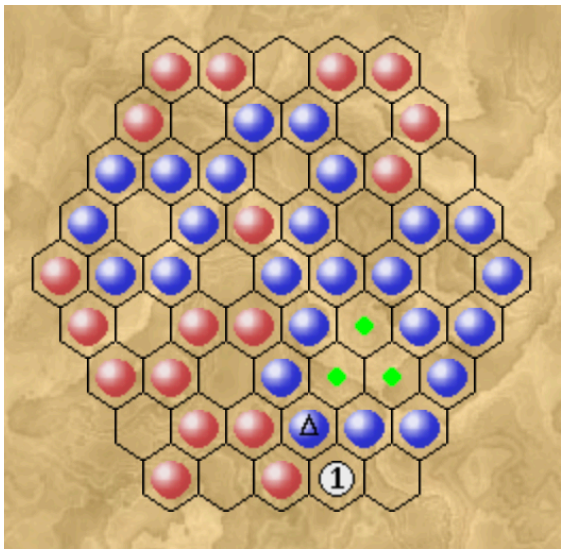
Java Programming Final Project

Project Overview

The objective of this final project is to implement a board game called Sabin Rains. Students can refer to https://www.di.fc.ul.pt/~jpn/gv/sabinrains.htm for guidance.



Game Rules

Sabin Rains is played on a 5x5 hexagonal board.

- Hex Ring: The set of six cells surrounding any given cell on the board. A hex ring may contain fewer than six cells, especially on the edges of the board.
- Drop:
  - Red and blue sides take turns, with red going first. Each round, each player drops a stone on an empty space.
  - Stones cannot be dropped on a cell without adjacent empty cells.
  - After placing a stone, the newly dropped stone may appear on up to 6 hex rings, centered on the 6 adjacent cells.
    - On each hex ring, it will flip all enemy stones that are sandwiched between it and another friendly stone (a flip is triggered when all cells between two friendly stones are occupied by enemy stones, not empty cells).
    - If there are five enemy stones on a hex ring, the newly placed stone will flip all five.
    - The flipping rule applies independently to each of the 6 hex rings affected by the new stone, meaning that flipping in one ring does not trigger a chain reaction in other rings.
- Objective: The game ends when there are no more valid moves left on the board. The player with the most pieces on the board wins.

Examples

1. In the image at the top, assume it is red's turn and red places a stone in cell [1]. Cell [1] is part of 4 hex rings, corresponding to 4 adjacent cells. Among them, the flipping rule is triggered only in the hex ring centered on the red stone to the left of [1], and the blue stone Δ is flipped to red.
2. It's blue's turn next, and there are only three cells left on the board where stones can be placed, marked with green dots. Blue can place a stones on any of these without triggering a flipping rule.
3. When it's red's turn again, red can place in the remaining two green-dotted cells. In this situation, red can always choose a placement that flips five blue stones in a hex ring and finally wins.

Requirements

This section describes the requirements and grading criteria. Any functionalities not being explicitly stated can be implemented in any way. The final project has a total score of 100, with an additional 10 points for optional content.

Basic Requirements (95 points)

Basic requirements are mandatory parts of the assignment.

GUI (35 points)

1. Draw the board (12 points): 5x5 hexagonal board (4 points), each cell a regular hexagon (4 points). The board background should be a color other than white and should not overlap with the red or blue of the stones (4 points).
2. Draw the stones (12 points): Colored circles.
    - Stones are circular (3 points).
    - Two colors of stones, red and blue (3 points).
    - Stones are placed in the exact center of the cell (3 points).
    - Stones should not overlap with the cell borders (3 points).
3. Highlighted areas (5 points): Different areas of the board have corresponding highlights.
    - Playable cell indicator: The center of playable cells is marked with a green dot (1 point).
    - Placement hint: The cell under the mouse, if it is a playable cell, is highlighted with a colored border (other than black) (1 point).
    - Flip indicator: The border of cells containing flippable stones is highlighted with a color different from black and different from placement hint (1 point).
    - Lock indicator: Locked areas on the board are colored gray (1 point), but the colors of stones on these cells are not affected (1 point). Locked areas refer to cells that are more than two cells away from any playable cell (not including two cells); these cells are neither playable nor can the colors of existing pieces be flipped.
4. Display game results (6 points): When the game ends, the result is displayed in a popup window. It only needs to display the win/loss result (3 points), e.g., "WINNER: RED" or "WINNER: BLUE".

Game Model (45 points)

The game model must be precisely implemented.

1. Initialization (2 points): Initialize an empty board according to the game rules.
2. Place a stone (16 points):
    - Move the mouse to a playable empty cell and click to place a stone according to the game rules (no animation required, place instantly) (12 points).
    - If the placement is not valid according to the game rules, it should not execute (4 points).
3. Board hints (6 points):
    - Playable and lock indicators are always present and update instantly when a stone is placed (2 points).
    - Move the mouse over a playable cell to immediately show the placement hint. Clicking the cell (thus placing a stone) instantly removes the placement hint (2 points).
    - Move the mouse over a playable cell to immediately show the flip hint. Clicking the cell (thus placing a stone) instantly removes the flip hint (2 points).
4. Flip stones (4 points): According to the rules, the colors of certain stones flip instantly upon placement (no animation required).
5. Alternate turns (2 points): Red moves first, then blue alternates. When it's red's turn, the stone placed is red, and during blue's turn, the stone is blue.
6. Determine winner (10 points): According to the game rules, once a move triggers the end of the game, neither side can make further moves (2 points). Then, determine the winner and display the result in a popup window (8 points).
7. Timing (5 points): A countdown timer must be displayed on the window, counting down in seconds (1 point). Each side must place a piece within 30 seconds; when one side places a stone, the other side's timer starts immediately (2 points). If time runs out, a stone is randomly placed in a playable area (2 points).

Here is the translation of the remaining sections of the Markdown file into English:

Performance Requirements (15 points)

The application has some basic performance requirements, and points will be deducted based on actual performance if these are not met.

1. Fluidity (7 points): Ensure the system runs smoothly without stuttering or noticeable delays. For example, highlights on the board should respond instantly when a stone is placed, and clicking a cell should place a stone instantly. Ensure that the user interface does not flicker.
2. Stability (8 points): The system must run stably without crashing or freezing. Each issue discovered during testing will result in a deduction of 2 points, up to a maximum of 8 points.

Optional Requirements (10 points)

The following are optional additional requirements for extra points. Interested students should **choose one** to complete. Points for these requirements are added directly to the final project score, with a maximum of 110 points.

1. Robot (10 points): Change the blue side to a robot aimed at maximizing wins with a strategy . Grading

will be based on the effectiveness and design of the strategy. The designed method and experiments on the strategy's effectiveness must be detailed in the project report.
2. Other self-designed extra functionalities should be confirmed with the teaching assistant beforehand.

Project Report (5 points)

The project report is mandatory and should include the following:

1. (Optional) Methods and experimental results corresponding to optional requirements.
2. Use class diagrams or natural language to describe the project architecture and design (5 points).
3. (Optional) Any other details you wish the teaching assistant to know, such as references used during the project or personal feelings of the course.

Submission Format

Submit all source code, the project report, and a directly runnable jar file (located in the root directory; missing the jar file results in a deduction of 10 points from the total score of the project) packed into a zip file.

```
└─2000123456_Zhang San.zip
      ├─ report.pdf // Project Report
      ├─ ***.jar    // Executable File
      └─ src        // Source Code Directory
```

Notes

- The programming assignment is an individual task; please complete it independently. If your submitted code references any materials or individuals, please cite these sources in your report; otherwise, it will be considered plagiarism and scored zero.
- Use Java language; there are no restrictions on the modules and libraries used. However, if non-built-in Java libraries are used, they need to be included in the jar package.
- If some functionalities, like board highlights, are complex to implement, prioritize ensuring the entire game can run normally to facilitate testing of other functionalities.