Reversi Game

Introduction

This document provides an overview and description of the Reversi game implementation using C++ classes. The Reversi game is a two-player game played on an n × n grid, where players take turns placing their colored pieces on the board. The goal is to have the most cells holding pieces of your color at the end of the game. The game follows specific rules for piece placement and flipping of opponent's pieces.

Classes Overview

The Reversi game implementation involves several classes to represent the game components and implement the game logic. The following classes are used:

1. **Subject** (abstract base class): Represents the subjects in the Observer pattern. It is provided in the `subject.h` file.

2. **Observer** (abstract base class): Represents the observers in the Observer pattern. It is provided in the `observer.h` file.

3. **Cell**: Represents a single cell on the game grid. It implements the Observer class, making each cell an observer of its neighbors. The `cell.h` file contains the implementation of this class.

4. **Grid:** Implements a two-dimensional grid of cells. It manages the game board and handles interactions between cells. The `grid.h` file contains the implementation of this class.

5. **State**: Provides a representation of a cell's state. It defines the messages passed between cells during the game. The `state.h` file contains the implementation of this struct.

6. **TextDisplay**: Keeps track of the character grid to be displayed on the screen. It observes every cell on the grid and updates the text representation accordingly. The `textdisplay.h` file contains the implementation of this class.

7. **info.h**: Defines the structure for queries issued by the observer on the subject. It is used for communication between the cells and the grid.

Game Rules and Implementation Details

The Reversi game follows specific rules for piece placement and flipping of opponent's pieces. The rules and implementation details are as follows:

- At the beginning of the game, the middle 4 cells of the grid are populated following the pattern Black-White-Black-White.

- The game is played by two players, Black and White, who take turns placing their colored pieces on the board. Black plays first.

- Each player places their piece in an empty cell on the grid.

- If a new piece forms a line segment with an existing piece of the same color, such that all cells in between are occupied and of the opposite color, those in-between pieces are flipped to the same color as the new piece and the existing piece.

- A legal move in standard Reversi must cause at least one flip. However, in this implementation, the program is not required to enforce this rule. Players can play a piece on any cell within the grid, as long as it is not currently occupied by another piece.

- The game ends when there are no more empty cells on the grid.

- The player with the most cells holding pieces of their color at the end of the game wins.

Observer Pattern and Implementation

The Reversi game implementation employs the Observer pattern to handle interactions between cells and keep track of the game state. Each cell on the grid is an observer of its neighbors. The TextDisplay class is also an observer of every cell.

In the Observer pattern, a subject's collection of observers does not distinguish what kinds of observers are actually in the collection. Therefore, a cell can have arbitrary cells or displays subscribed to it. When notifying observers, the implementation iterates through the collection and notifies them.

When a new piece is placed on a cell, the cell notifies its observers (adjacent cells) about the new piece. The notified cells relay the message to their neighbors until a cell stops relaying and instead replies back. If a neighbor cell contains a piece of the same color as the new piece, it replies back to indicate the end of the relay. The direction of the line

formed by the new piece can be determined based on the information passed along during the relay.

The State struct provided in the `state.h` file is used to represent the messages passed between cells during the game. It contains information about the position, color, and direction of the new piece.

Operator Overloading

To enhance the functionality and user experience of the Reversi game implementation, operator overloading is employed.

1. **Operator<< for TextDisplay**: The operator<< is overloaded for the TextDisplay class. It allows the entire grid to be printed out when this operator is invoked. Each empty cell is represented as "-", and cells with white or black pieces are represented as "W" and "B" respectively.

2. **Operator<< for Grid**: The operator<< is overloaded for the Grid class. It enables printing a grid, which invokes the operator<< for the TextDisplay class. This makes the grid appear on the screen in the desired format.

Program Commands

The program accepts the following commands via stdin:

1. **new n**: Creates a new n × n grid, where $n \geq 4 \wedge n \equiv 0 \pmod 2$. If there is already an active grid, it is destroyed and replaced with the new one. The middle 4 squares are initialized following the Black-White-Black-White pattern.

2. **play r c**: Within a game, plays a piece at row r and column c of the current player's color. If the specified position is already occupied by a piece or is outside the grid, the input is ignored and nothing is done.

The program continues accepting commands until the input stream is exhausted or when the game is over. The game ends when there are no more empty cells on the grid. After the game is over, the program displays the winner (Black Wins!, White Wins!, or Tie!) to stdout before terminating.