ICS4U Final Tetris Project

Description of the Program

The goal of the project is to program a Tetris simulator. Tetris is a tile matching board game originally designed by Russian developer Alexey Pajitnov and released in 1984. The project requires a board with an r by c grid of empty tiles, where r and c can be specified by the user.

In the program there are 7 Tetris pieces composed of four tiles in different 2D geometric configurations. A piece is randomly selected by the computer and appears on the top of the board. The user must be able to manipulate the piece – move the piece horizontally and rotate it clockwise – with the keyboard. The active peace gradually moves down the board at a constant velocity; when it reaches the bottom of the board or another inactive piece it becomes inactive. Afterwards a new active piece is spawned from the centre top of the board.

The objective of Tetris is to survive and gain points. 100 points are awarded whenever an entire row in the grid is filled with inactive blocks. In that case the c blocks in the horizontal row are removed to provide additional space, and all blocks above the row are shifted 1 downwards. The game must end when a landed block reaches the top of the grid. In that case the player's score is finalized and vividly displayed.

Analysis of the Program

High level overview of the program

1. Class `Main` is the entry point of the program.
2. Class `TetrisPanel` inherits from `JPanel` and handles painting to the screen, scorekeeping, and keyboard events. It uses the standard off screen graphics optimization discussed in class.
3. Class `TetrisGrid` represents the game's grid. It does not deal with any game logic directly. It simply provides an interface for the grid that other classes can later use. It allows other classes to change the colour of a cell using `drawCell(int r, int c, int colour)` for example, or check if a cell is empty using `empty(int r, int c)`. An important point about `TetrisGrid` is that it does not draw the grid to the screen unless the method `finalize()` is called. This provides performance improvements for tasks requiring multiple updates.
4. Class `TetrisPiece` represents a single tetris block, and provides methods like `canMoveLeft`, `moveLeft`, and `rotateClockwise`. It stores an instance of `TetrisGrid` so it handles drawing itself to the grid. Internally, `TetrisPiece` represents each piece as a 4x4 array called `piece`. It stores integers `curRow` and `curCol` which represent its position in the grid (specifically, the row and column of the top-left cell in the grid).

5. Class `PieceManager` is a wrapper around class `TetrisPiece`. It deals with the basic logic of the piece, like spawning a piece, or moving it down at regular intervals. It depends on the interface of `TetrisPiece` but not its implementation. `PieceManager` is not strictly necessary. We could have put all the code for `PieceManager` in `TetrisPiece` for example. However, we prefer to think of `TetrisPiece` as a toolbox that `PieceManager` can use to perform game logic, similar to how `TetrisGrid` is a toolbox for `TetrisPiece` to draw to.

Description of classes

class `Main`

Methods

- `public static void main(String[] args)`

class TetrisPanel

Instance Fields

- `private int score`
- `private int nrows`
- `private int ncols`
- `private TetrisGrid grid`
- `private PieceManager pieceManager`
- `private BufferedImage osi`
- `private Graphics osg`
- `private Dimension dim`

Methods

- `public TetrisPanel(int r, int c)`
- `public void paint(Graphics g)`
- `public void die()`
- `public void addScore(int s)`
- `public void keyPressed(KeyEvent ke)`
- `public void keyReleased(KeyEvent ke)`
- `public void keyTyped(KeyEvent ke)`
- `private void init()`
- `private void updateImage()`

class `TetrisGrid`

Instance Fields

- `private TetrisPanel panel`
- `private int[][] grid`
- `private int nrows`

- private int ncols
- private static final Color[] cellColours

Methods

- public TetrisGrid(TetrisPanel p, int r, int c)
- public void display(Graphics g, int lft, int rt, int tp, int btm)
- public void clearFilledRows()
- public boolean isTopRow(int r)
- public boolean empty(int r, int c)
- public void drawCell(int r, int c, int colour)
- public void finalize()
- public int getRows()
- public int getCols()

class TetrisPiece

Instance Fields

- private int curRow
- private int curCol
- private TetrisGrid tgrid
- private int[][] piece

Methods

- public TetrisPiece(TetrisGrid grid)
- public boolean occupiesTop()
- public boolean canMoveDown()
- public void moveDown()
- public boolean canMoveLeft()
- public void moveLeft()
- public boolean canMoveRight()
- public void moveRight
- public boolean canRotateClockwise()
- public void rotateClockwise()
- private void undraw()
- private void draw()
- private void initialize()

class PieceManager

Instance Fields

- private static final long updateInterval

- `private TetrisGrid tgrid`
- `private TetrisPiece piece`
- `private TetrisPanel tpanel`
- `private Timer mainLoopTimer`

Methods

- `public PieceManager(TetrisPanel p, TetrisGrid grid)`
- `public void startSpawning()`
- `public void stopSpawning()`
- `public void moveLeft()`
- `public void moveRight()`
- `public void rotateClockwise()`
- `public void fastForward()`
- `private boolean timeStep`

Order of execution

1. In the main function of class `Main`, the user enters the number of rows and columns of the tetris grid. Then we create a new `JFrame`, set basic properties like title, size and location, and add a new `TetrisPanel` , which inherits from `JPanel`, instance to the frame. We pass the number of rows and columns into the TetrisPanel constructor. Finally, we set the frame as visible.

2. The execution then goes to `TetrisPanel`. In its constructor, we store the number of rows and columns in private variables `nrows` and `ncols`. We then add the current instance as its own `KeyListener` and get the focus in the window in order to recieve keyboard events. Finally, we call the private method `init()`.

3. `init` stops the currently running game, if it exists, and creates new `TetrisGrid` and `PieceManager` instances. It then tells `pieceManager` to start the game using the method `pieceManager.startSpawning()`.

4. In method `startSpawning`, the `PieceManager` starts running a new `Timer` at regular intervals. The `TimerTask` is the `timeStep` method, which goes forward in time by one step. This method moves the piece down if possible, and creates a new one otherwise. It also checks if the piece occupies the top row and cannot move farther down, in which case the game is over. If this happens, the method `die` in class `TetrisPanel` is called and the game is restarted.

5. In `TetrisPanel`, we mapped keys to functions in `PieceManager`. For example, if the left arrow is pressed, then the method `moveLeft()` in `PieceManager`is called.

6. Then, in `PieceManager`, the corresponding method in `TetrisPiece` is called. The `TetrisPiece` methods update the `TetrisGrid` instance accordingly, and then the method `finalize()` in `TetrisGrid` is called by `PieceManager`. This `finalize()` method draws the changes to the screen by calling the `repaint()` method of `TetrisPanel`. This in turn will call method `updateImage` in `TetrisPanel`, which will call method `display` in `TetrisGrid`. The `display` method draws the grid onto the screen.

Development Timeline

1. We made our initial version on May 29. It contained skeleton code for `Main`, `TetrisPiece`, `TetrisGrid`, `BlockManager`, and `TetrisPanel`. We later renamed `BlockManager` to `PieceManager`.
2. On June 4, we fleshed out the template we made on May 29. By the end of June 4, the program was runnable but user controls had not been added yet.
3. On June 5, we added keyboard controls and various other small updates. By the end of June 5, the game was playable.
4. On June 7, we added comments to the code and updated the Readme.md.
5. On June 8, we fixed several mistakes we made in method `initialize` of class `TetrisPiece` and added javadocs. This is the current version.

For more information see the commit history here:
https://github.com/cranberrysauce26/TetrisProject/commits/master
(https://github.com/cranberrysauce26/TetrisProject/commits/master)