CS 232: Final Project – Pseudo Code

Miffy Chen

Pseudo Code

**Tetrominoes.java**

**public enum Tetrominoes**

{

NoShape // default, coordinates set to 0, color set to white

IShape // Line shape, cyan

JShape // Mirrored L shape, blue

LShape // L shape, orange

OShape // Box shape, yellow

SShape // S shape, green

TShape // T shape, magenta (bright pink)

ZShape // Z shape, red

public int[][] coords; the coordinates of the shapes

public Color color; RGB color of the shapes

private Tetrominoes(int[][] coords, Color c) return coords and color of the shape

}

**Shape.java**

**public class Shape**

{

**private Tetrominoes pieceShape** : the shape of the piece that is picked

**private int[][] coords** : the coordinates of the shape picked

**private static ArrayList<Integer> list :** {1,2,3,4,5,6,7} stand for the indexes of

Tetrominoes and whichever one is picked randomly will be deducted from the list

**private static ArrayList<Integer> shapesCalled** : keeps track of the shapes called

**public Shape()** : constructor - set coordinates to [4][2] and set shape to no shape

**public void setShape()** : set the shape of the pieceShape – the current piece

**private void setX(int index, int x)** : set the X coordinate at index index

**private void setY(int index, int y)** : set the Y coordinate at index index

**public int x(int index)** : return coords[index][0]

**public int y(int index)** : return coords[index][1]

**public Tetrominoes getShape()** : return pieceShape

**private static int randomSevenShape()** :

Save the 7 shapes of Tetrominoes in their enum set order, to ArrayList list

Pick a shape randomly out of the available shapes left

Every time a shape is picked, remove it from the available list to be called next

When all 7 shapes have been called, reset the list to every shape, start a new set

**public void setRandomShape()** : make sure a set of Tetrominoes (set of 7) is called,

then call a new set. So that the pieces given to user won’t be repeated and the game

is more balanced and .

**public int minX()** : return the minimum X coordinate of the piece

**public int minY()** : return the minimum Y coordinate of the piece

**public Shape rotateLeft()** :rotate the current piece chosen in counter clockwise

**public Shape rotateRight()** :rotate the current piece chosen in clockwise

}

**Board.java**

**public class Board extends JPanel implements ActionListener**

**{**

**private static final int BOARD\_WIDTH = 10;** board width is 10

**private static final int BOARD\_HEIGHT = 22;** board height is 22

**private Timer timer;** (delay, action) delay set to 500, action set to “this”

500 is how slow the pieces fall

timer is activated when the game starts

timer stops when the game is paused by pressing “p”

timer starts again when the game is un-paused by pressing “p”

timer ends when the game ends

**private boolean isFallingFinished;** status of current piece is finished falling

**private boolean isStarted;** true when game is started, false when game is paused / ended

**private boolean isPaused;** true when user pressed “p” to pause game

**private int numLinesRemoved;** number of lines removed (score of game)

**private int curX; private int curY;**

**private Shape curPiece;**

**private JLabel statusBar** : display the score and instructions for user

**private Tetrominoes[] board** : board to display the tetrimino pieces

**public Board(Tetris parent)** : Constructor

**{**

Take in the Tetris class that opens up a new game

Make board visible, set board dimensions to BOARD\_WIDTH and BOARD\_HEIGHT

Set how fast the pieces fall

Declare, Instantiate, and set the statusBar

Call the KeyAdapter that listens for user key input

**}**

**public int squareWidth() {** return width of board / 10 **}** // how big a square’s width is

**public int squareHeight() {** return height of board / 22 **}** // how big a square’s height is

**public Tetrominoes shapeAt(int x, int y)** : for getting the shape by giving x&y coords

**private void clearBoard()** : set board to no shape, called at the start

**private void pieceDropped()** : called when a piece has completely touched the bottom

curPiece’s x&y coords are checked to see if it can go down another line

if there is a full line to be removed, remove it

when a piece is done falling, call newPiece()

**public void newPiece()** : called when a new piece does not exist on the board

get a random shape from the set of 7 choices, or the available ones

display the newPiece at the horizontal middle of the board

display the newPiece at the top of the board

check if this newPiece still have space to move

if not then it means the board is full, and the game is ended

**private void oneLineDown()** : move the piece one line down

**public void actionPerformed(ActionEvent ae)** : overriding the definition of actionPerformed to catch user input on keyboard, if the time set in timer has gone by and no action from user has been caught, move the piece down a line. If user action was caught, then the action can be performed up until the piece is done falling.

**private void drawSquare(Graphics g, int x, int y, Tetrominoes shape)** :

Acquire the Tetromino shape’s color in the enum

drawSquare with that color

draw shadows on squares by using same color but lighter shade on square’s top and

left side, and darker shade on square’s bottom and right side

**public void paint(Graphics g)** : override the paint()

Use getSize() and store the value in Dimension size

Draw the pieces using drawSquare() and match the dimensions of the board to

dimensions of the shape by calling squareWidth() and squareHeight(),

and count how many squares needed to be drawn for curPiece

by matching the values in enum Tetrominoes

**public void start()** : set the starting values of the Board class

isStarted (T/F for game has started), isFallingFinished (T/F if piece is done falling),

numLinesRemoved (start at 0, is the score), clearBoard(), newPiece(),

timer.start() for setting the speed that the pieces fall at

**public void pause()** : when the user chooses to pause game by pressing “p”

Stop the timer so curPiece stops falling

Display on the statusBar that game is being paused

If user presses “p” again to unpause

Start the timer again so curPiece continues to fall

Display on the statusBar that game is no longer being paused

**private boolean tryMove(Shape newPiece, int newX, int newY)** :

Try to move the current piece in , , , direction of the user’s choosing

Direction will be passed in through the form of newX, newY

If moving the piece surpasses the limitations of the board: return false

If piece can be moved: return true

**private void removeFullLines()** : called after a newPiece is done falling

Check every x horizon for lines that are full and should be cleared

(by going across the rows and checking for NoShape at column space)

If (lineIsFull): ++numFullLines and clear the line that is full

If (numFullLines > 0)

numLinesRemoved =+ numFullLines

Update score in statusBar

**private void dropDown()** : as long as the piece has not reached the bottom,

user can choose to move the piece all the way down as far as it could

**}**

**class TetrisKeyAdapter extends KeyAdapter**

**{**

**public void keyPressed(KeyEvent key)**

**{**

listen for keyboard input by user

“p” || “P” = pause();

“” = move piece 1 space to the left

“” = move piece 1 space to the right

“” = move piece 1 space down

“” = rotate piece in clockwise direction

“(space bar)” = drop piece all the way down to the bottom

**}**

**}**

**}**

**Tetris.java**

**public class Tetris extends JFrame**

{

**public Tetris()**

{

Constructor - Declare, Instantiate, and set a Board object

Set the status bar to the bottom of the display

Set the board on top of the status bar

Set the size (pixels) of the game

Set game title to (“New Tetris Game”)

**public static void main(String args[])**

{

Declare and Instantiate a new Tetris object

Open new window at the center of the screen

Make window visible

}

}