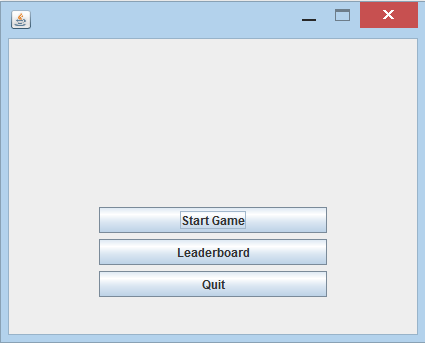
|  |
| --- |
|  |
| Tetris |
| Members:408850351徐雅玟/408850385鄭明佐 |
|  |
|  |
|  |

|  |
| --- |
|  |

**Specifications**

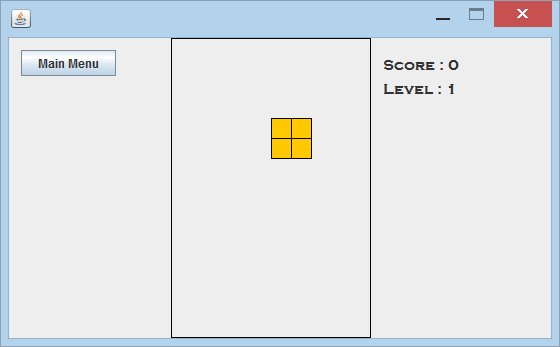
Step1

When the user first launches the program, they will be led to this screen. There are three JButtons on the screen. The user can click “start game” button to start the game. However, if the user clink “leaderboard” button, the user will be led to leaderboard screen. If the user clink “quit” button, the program will finish and exit the screen.



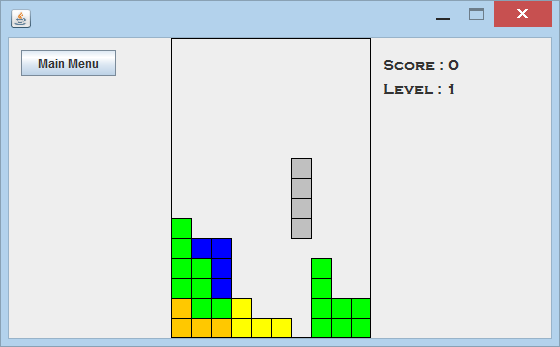
Step2

After users clink “start game” button, the game will start and the user will be led to this screen. The blocks will drop down automatically and the user can use arrow keys to move and rotate the blocks. There is also a JButton in the left hand side, if users clink it, they will be led to first screen.



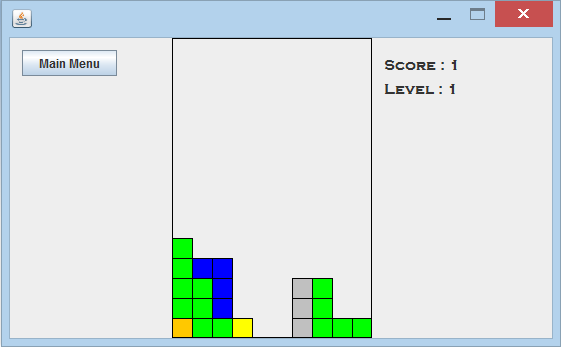
Step3

The users can use arrow keys to move and rotate the blocks. Users can arrange and place where they want.



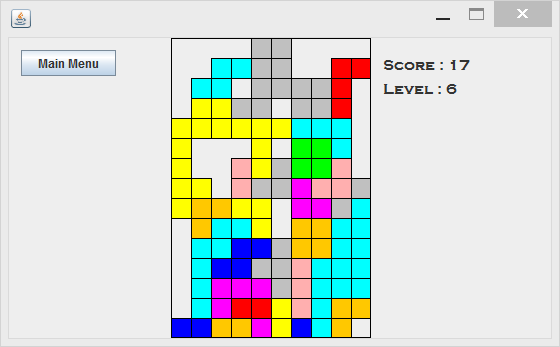
Step4

If the blocks assemble the line, the line will be cleaned, and score will be added.



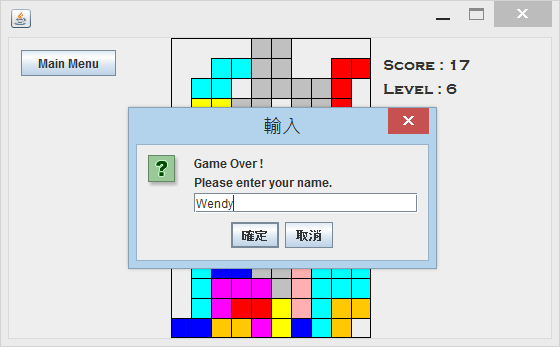
Step5

When the score increases, the level will upgrade and the speed will accelerate.



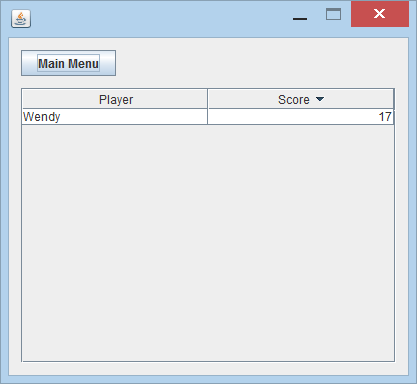
Step5

When the blocks accumulate to the top, the game will finish. When the game finish users can see a small screen, and users need to type their names.



Step6

After users enter their names, they will be led to leaderboard screen. Leaderboard will show the users’ names and their scores.



**Disclaimers**

**Purpose of the program**

The program is designed to a simple “Tetris” game, with the addition of the JOptionPane to let players enter their names. When the players finish the game they can see all the players’ name and all the players’ scores.

**Shape of Block (Block Structure)**

We use 1 and 0 to create the block. In block structure, 1 means there is a color in the block, 0 means empty, so that we can create different shapes of block.

**How to move the block**

There is something important that why is x++ in the move right method and why is x— in the move left method. Assuming the block is the origin, draw the x-axis and y-axis. If you want to move block right, then block will move from origin to x, and x is how much the steps user move, and so on.

**Check Bottom, Left and Right**

After we have the blocks, we need to make them move down. We use “drop block” method, in this method, we tell the computer that if there is nothing (null) under block, then block move down. And we use the check bottom method to let blocks stop if they are in the bottom.

**Database**

Adding the sound or music make game more interesting. We downloaded the sound/music what we want, and that program read the file. After program read the files of sound, players can hear when they are playing the game.

**Conclusion**

**408850385鄭明佐**

In the process of writing this app, I recognized a lot of methods and several packages. I think the most important thing is that I learn some good ways to learn it. This is very helpful to me. To build an app with the group members, I not only learn the skills of writing programs, but also improve the skills of communicating with the group members. I will often find opportunities to do some projects to practice my skills in the future. I am particularly interested in some visual methods. Interested, and I will continue to explore this kind of knowledge

**Conclusion**

**408850351 徐雅玟**

This is the first time I have made a project from scratch. For me, this is a big challenge. From small to large, we are all game players, but we don’t know how difficult it is to make games. Actually this time there are still a lot of unfinished places (bugs) in the production of the game, but doing a project is already a very fulfilling thing for me.

In this project, the biggest challenge I faced was debugging because I had not made a complete app. Even though I encountered a lot of difficulties in this project, I was very interested in making games. I hope to work in this direction in the future.

**Java Source Code**

**//AudioPlayer.java**

package tetris;

import java.io.File;

import java.io.IOException;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.sound.sampled.AudioSystem;

import javax.sound.sampled.Clip;

import javax.sound.sampled.LineUnavailableException;

import javax.sound.sampled.UnsupportedAudioFileException;

public class AudioPlayer {

private String soundsFolder = "tetrissounds" + File.separator;

private String clearLinePath = soundsFolder + "line.wav";

private String gameoverPath = soundsFolder + "success.wav";

private Clip clearLineSound, gameoverSound;

public AudioPlayer(){

try {

clearLineSound = AudioSystem.getClip();

gameoverSound = AudioSystem.getClip();

clearLineSound.open(AudioSystem.getAudioInputStream(new File(clearLinePath).getAbsoluteFile()));

gameoverSound.open(AudioSystem.getAudioInputStream(new File(gameoverPath).getAbsoluteFile()));

} catch (LineUnavailableException ex) {

Logger.getLogger(AudioPlayer.class.getName()).log(Level.SEVERE, null, ex);

} catch (UnsupportedAudioFileException ex) {

Logger.getLogger(AudioPlayer.class.getName()).log(Level.SEVERE, null, ex);

} catch (IOException ex) {

Logger.getLogger(AudioPlayer.class.getName()).log(Level.SEVERE, null, ex);

}

}

public void playClearLine(){

clearLineSound.setFramePosition(0);

clearLineSound.start();

}

public void playGameover(){

gameoverSound.setFramePosition(0);

gameoverSound.start();

}

}

**//GameArea.java**

package tetris;

import java.awt.Color;

import java.awt.Graphics;

import java.util.Random;

import javax.swing.JPanel;

import tetrisblocks.\*;

public class GameArea extends JPanel {

private int gridRows;

private int gridColumns;

private int gridCellSize;

private Color[][] background;

private TetrisBlock block;

private TetrisBlock[] blocks;

public GameArea(JPanel placeholder, int columns){

//placeholder.setVisible( false )

this.setBounds( placeholder.getBounds() );

this.setBackground( placeholder.getBackground() );

this.setBorder( placeholder.getBorder() );

gridColumns = columns;

gridCellSize = this.getBounds().width / gridColumns;

gridRows = this.getBounds().height / gridCellSize;

blocks = new TetrisBlock[]{ new IShape(),

new JShape(),

new LShape(),

new OShape(),

new SShape(),

new TShape(),

new ZShape(), };

}

public void initBackgroundArray(){

background = new Color[gridRows][gridColumns];

}

public boolean moveBlockDown(){

if(checkBottom() == false){

return false;

}

block.moveDown();

repaint();

return true;

}

public void moveBlockRight() {

if( block == null ) return;

if( !checkRight() ) return;

block.moveRight();

repaint();

}

public void moveBlockLeft(){

if( block == null ) return;

if( !checkLeft() ) return;

block.moveLeft();

repaint();

}

public void rotateBlock() {

if( block == null ) return;

block.rotate();

if(block.getLeftEdge() < 0) block.setX(0);

if(block.getRightEdge() >= gridColumns) block.setX(gridColumns - block.getWidth());

if(block.getBottomEdge() >= gridRows) block.setY(gridRows - block.getHeight());

repaint();

}

public void dropBlock() {

if( block == null ) return;

while( checkBottom() ){

block.moveDown();

}

repaint();

}

private boolean checkBottom(){

if( block.getBottomEdge() == gridRows){

return false;

}

int[][]shape = block.getShape();

int w = block.getWidth();

int h = block.getHeight();

for(int col = 0; col < w; col++){

for(int row = h - 1; row >= 0; row--){

if(shape[row][col] != 0){

int x = col + block.getX();

int y = row + block.getY() + 1;

if(y < 0) break;

if(background[y][x] != null) return false;

break;

}

}

}

return true;

}

private boolean checkLeft(){

if( block.getLeftEdge() == 0 ){

return false;

}

int[][]shape = block.getShape();

int w = block.getWidth();

int h = block.getHeight();

for(int row = 0; row < h; row++){

for(int col = 0; col < w; col++){

if(shape[row][col] != 0){

int x = col + block.getX() - 1;

int y = row + block.getY();

if(y < 0) break;

if(background[y][x] != null) return false;

break;

}

}

}

return true;

}

private boolean checkRight(){

if( block.getRightEdge() == gridColumns){

return false;

}

int[][]shape = block.getShape();

int w = block.getWidth();

int h = block.getHeight();

for(int row = 0; row < h; row++){

for(int col = w - 1; col >= 0; col--){

if(shape[row][col] != 0){

int x = col + block.getX() + 1;

int y = row + block.getY();

if(y < 0) break;

if(background[y][x] != null) return false;

break;

}

}

}

return true;

}

public int clearLines(){

boolean lineFilled;

int linesCleared = 0;

for(int r = gridRows - 1;r >= 0; r--){

lineFilled = true;

for(int c = 0; c < gridColumns; c++){

if(background[r][c] == null){

lineFilled = false;

break;

}

}

if(lineFilled){

linesCleared++;

clearLine(r);

shiftDown(r);

clearLine(0);

r++;

repaint();

}

}

if(linesCleared > 0){

Tetris.playClearLine();

}

return linesCleared;

}

private void clearLine(int r){

for(int i = 0;i < gridColumns; i++){

background[r][i] = null;

}

}

private void shiftDown(int r){

for(int row = r; row > 0; row--){

for(int col = 0; col < gridColumns; col++){

background[row][col] = background[row - 1][col];

}

}

}

private void drawBlock(Graphics g){

int h = block.getHeight();

int w = block.getWidth();

Color c = block.getColor();

int[][] shape = block.getShape();

for(int row = 0; row < h; row++){

for(int col = 0; col < w; col++){

if(shape[row][col] == 1){

int x = (block.getX() + col) \* gridCellSize;

int y = (block.getY() + row) \* gridCellSize;

drawGridSquare(g, c, x, y);

}

}

}

}

public void moveBlockToBackground(){

int h = block.getHeight();

int w = block.getWidth();

Color c = block.getColor();

int[][] shape = block.getShape();

int xPos = block.getX();

int yPos = block.getY();

for(int row = 0; row < h; row++){

for(int col = 0; col < w; col++){

if(shape[row][col] == 1){

background[row + yPos][col + xPos] = c;

}

}

}

}

private void drawBackground(Graphics g){

Color color;

for(int r = 0; r < gridRows; r++){

for(int c = 0; c < gridColumns; c++){

color = background[r][c];

if(color != null){

int x = c \* gridCellSize;

int y = r \* gridCellSize;

drawGridSquare(g, color, x, y);

}

}

}

}

private void drawGridSquare(Graphics g, Color color, int x, int y){

g.setColor(color);

g.fillRect(x, y, gridCellSize, gridCellSize);

g.setColor(Color.black);

g.drawRect(x, y, gridCellSize, gridCellSize);

}

@Override

protected void paintComponent(Graphics g){

super.paintComponent(g);

drawBackground(g);

drawBlock(g);

}

public boolean isBlockOutOfBounds() {

if(block.getY() < 0){

block = null;

return true;

}

return false;

}

public void spawnBlock() {

Random r = new Random();

block = blocks[r.nextInt( blocks.length )];

block.spawn(gridColumns);

}

}

**//GameForm.java**

package tetris;

import java.awt.Graphics;

import java.awt.event.ActionEvent;

import javax.swing.AbstractAction;

import javax.swing.ActionMap;

import javax.swing.InputMap;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.KeyStroke;

public class GameForm extends JFrame {

private GameArea ga;

private GameThread gt;

public GameForm() {

initComponents();

ga = new GameArea(gameAreaPlaceholder, 10);

this.add(ga);

initControls();

}

public void startGame() {

ga.initBackgroundArray();

gt = new GameThread(ga, this);

gt.start();

}

private void initControls(){

InputMap im = this.getRootPane().getInputMap();

ActionMap am = this.getRootPane().getActionMap();

im.put(KeyStroke.getKeyStroke("RIGHT"), "right");

im.put(KeyStroke.getKeyStroke("LEFT"), "left");

im.put(KeyStroke.getKeyStroke("UP"), "up");

im.put(KeyStroke.getKeyStroke("DOWN"), "down");

am.put("right", new AbstractAction() {

@Override

public void actionPerformed(ActionEvent ae) {

ga.moveBlockRight();

}

});

am.put("left", new AbstractAction() {

@Override

public void actionPerformed(ActionEvent ae) {

ga.moveBlockLeft();

}

});

am.put("up", new AbstractAction() {

@Override

public void actionPerformed(ActionEvent ae) {

ga.rotateBlock();

}

});

am.put("down", new AbstractAction() {

@Override

public void actionPerformed(ActionEvent ae) {

ga.dropBlock();

}

});

}

public void updateScore(int score) {

scoreDisplay.setText("Score : " + score);

}

public void updateLevel(int level) {

levelDisplay.setText("Level : " + level);

}

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

gameAreaPlaceholder = new javax.swing.JPanel();

scoreDisplay = new javax.swing.JLabel();

levelDisplay = new javax.swing.JLabel();

btnMainMenu = new javax.swing.JButton();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

setResizable(false);

gameAreaPlaceholder.setBackground(new java.awt.Color(238, 238, 238));

gameAreaPlaceholder.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(0, 0, 0)));

javax.swing.GroupLayout gameAreaPlaceholderLayout = new javax.swing.GroupLayout(gameAreaPlaceholder);

gameAreaPlaceholder.setLayout(gameAreaPlaceholderLayout);

gameAreaPlaceholderLayout.setHorizontalGroup(

gameAreaPlaceholderLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGap(0, 198, Short.MAX\_VALUE)

);

gameAreaPlaceholderLayout.setVerticalGroup(

gameAreaPlaceholderLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGap(0, 298, Short.MAX\_VALUE)

);

scoreDisplay.setFont(new java.awt.Font("Copperplate Gothic Bold", 0, 15)); // NOI18N

scoreDisplay.setText("score : 0");

levelDisplay.setFont(new java.awt.Font("Copperplate Gothic Bold", 0, 15)); // NOI18N

levelDisplay.setText("level : 1");

btnMainMenu.setText("Main Menu");

btnMainMenu.setFocusable(false);

btnMainMenu.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

btnMainMenuActionPerformed(evt);

}

});

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addContainerGap()

.addComponent(btnMainMenu)

.addGap(55, 55, 55)

.addComponent(gameAreaPlaceholder, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addComponent(levelDisplay, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addGap(96, 96, 96))

.addGroup(layout.createSequentialGroup()

.addComponent(scoreDisplay, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addContainerGap())))

);

layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(gameAreaPlaceholder, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGroup(layout.createSequentialGroup()

.addGap(18, 18, 18)

.addComponent(scoreDisplay)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(levelDisplay))

.addGroup(layout.createSequentialGroup()

.addContainerGap()

.addComponent(btnMainMenu))

);

pack();

setLocationRelativeTo(null);

}// </editor-fold>

private void btnMainMenuActionPerformed(java.awt.event.ActionEvent evt) {

gt.interrupt();

this.setVisible(false);

Tetris.showStartup();

}

public static void main(String args[]) {

/\* Set the Nimbus look and feel \*/

/\* Create and display the form \*/

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

new GameForm().setVisible(true);

}

});

}

// Variables declaration - do not modify

private javax.swing.JButton btnMainMenu;

private javax.swing.JPanel gameAreaPlaceholder;

private javax.swing.JLabel levelDisplay;

private javax.swing.JLabel scoreDisplay;

// End of variables declaration

}

**//GameThread.java**

package tetris;

import java.awt.Color;

import java.util.Random;

import java.util.logging.Level;

import java.util.logging.Logger;

public class GameThread extends Thread{

private GameArea ga;

private GameForm gf;

private int score;

private int level=1;

private int scorePerLevel=3;

private int pause=1000;

private int speedupPerLevel=100;

public GameThread(GameArea ga, GameForm gf) {

this.ga = ga;

this.gf = gf;

gf.updateScore(score);

gf.updateLevel(level);

}

@Override

public void run(){

while(true){

ga.spawnBlock();

while(ga.moveBlockDown()){

try{

Thread.sleep(pause);

} catch (InterruptedException ex) {

return;

}

}

if(ga.isBlockOutOfBounds()){

Tetris.gameOver(score);

break;

}

ga.moveBlockToBackground();

score += ga.clearLines();

gf.updateScore(score);

int lvl = score/ scorePerLevel + 1;

if(lvl > level){

level = lvl;

gf.updateLevel(level);

pause -= speedupPerLevel;

}

}

}

}

**//LeaderboardForm.java**

package tetris;

import java.io.FileOutputStream;

import java.io.ObjectOutputStream;

import java.util.ArrayList;

import java.util.Vector;

import javax.swing.RowSorter.SortKey;

import javax.swing.SortOrder;

import javax.swing.table.DefaultTableModel;

import javax.swing.table.TableModel;

import javax.swing.table.TableRowSorter;

public class LeaderboardForm extends javax.swing.JFrame {

private DefaultTableModel tm;

private String leaderboardFile = "leaderboard";

private TableRowSorter<TableModel> sorter;

public LeaderboardForm() {

initComponents();

initTableData();

initTableSorter();

}

private void initTableData(){

Vector ci = new Vector();

ci.add("Player");

ci.add("Score");

tm = ( DefaultTableModel ) leaderboard.getModel();

try{

FileOutputStream fs = new FileOutputStream(leaderboardFile);

ObjectOutputStream os = new ObjectOutputStream(fs);

tm.setDataVector( (Vector<Vector>)os.readObject(), ci );

os.close();

fs.close();

}catch(Exception e){}

}

private void initTableSorter(){

sorter = new TableRowSorter<>(tm);

leaderboard.setRowSorter(sorter);

ArrayList<SortKey> keys = new ArrayList<>();

keys.add( new SortKey(1, SortOrder.DESCENDING));

sorter.setSortKeys(keys);

}

private void saveLeadrboard(){

try{

FileOutputStream fs = new FileOutputStream(leaderboardFile);

ObjectOutputStream os = new ObjectOutputStream(fs);

os.writeObject(tm.getDataVector());

os.close();

fs.close();

}catch(Exception e){}

}

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

btnMainMenu = new javax.swing.JButton();

jScrollPane1 = new javax.swing.JScrollPane();

leaderboard = new javax.swing.JTable();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

setResizable(false);

btnMainMenu.setText("Main Menu");

btnMainMenu.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

btnMainMenuActionPerformed(evt);

}

});

leaderboard.setModel(new javax.swing.table.DefaultTableModel(

new Object [][] {

},

new String [] {

"Player", "Score"

}

) {

Class[] types = new Class [] {

java.lang.Object.class, java.lang.Integer.class

};

boolean[] canEdit = new boolean [] {

false, false

};

public Class getColumnClass(int columnIndex) {

return types [columnIndex];

}

public boolean isCellEditable(int rowIndex, int columnIndex) {

return canEdit [columnIndex];

}

});

jScrollPane1.setViewportView(leaderboard);

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(btnMainMenu)

.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED\_SIZE, 375, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

);

layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addContainerGap()

.addComponent(btnMainMenu)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED\_SIZE, 275, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

);

pack();

setLocationRelativeTo(null);

}// </editor-fold>

private void btnMainMenuActionPerformed(java.awt.event.ActionEvent evt) {

this.setVisible(false);

Tetris.showStartup();

}

public void addPlayer(String playerName, int score){

tm.addRow(new Object[] {playerName, score});

saveLeadrboard();

this.setVisible(true);

}

public static void main(String args[]) {

/\* Set the Nimbus look and feel \*/

//<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">

/\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.

\* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html

\*/

try {

for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {

if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());

break;

}

}

} catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(LeaderboardForm.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(LeaderboardForm.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(LeaderboardForm.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(LeaderboardForm.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

}

//</editor-fold>

/\* Create and display the form \*/

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

new LeaderboardForm().setVisible(true);

}

});

}

// Variables declaration - do not modify

private javax.swing.JButton btnMainMenu;

private javax.swing.JScrollPane jScrollPane1;

private javax.swing.JTable leaderboard;

// End of variables declaration

}

**//StartupForm.java**

package tetris;

public class StartupForm extends javax.swing.JFrame {

public StartupForm() {

initComponents();

}

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

btnStart = new javax.swing.JButton();

btnQuit = new javax.swing.JButton();

btnLeaderboard = new javax.swing.JButton();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

setResizable(false);

btnStart.setText("Start Game");

btnStart.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

btnStartActionPerformed(evt);

}

});

btnQuit.setText("Quit");

btnQuit.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

btnQuitActionPerformed(evt);

}

});

btnLeaderboard.setText("Leaderboard");

btnLeaderboard.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

btnLeaderboardActionPerformed(evt);

}

});

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGap(90, 90, 90)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(btnQuit, javax.swing.GroupLayout.PREFERRED\_SIZE, 228, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(btnLeaderboard, javax.swing.GroupLayout.PREFERRED\_SIZE, 228, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(btnStart, javax.swing.GroupLayout.PREFERRED\_SIZE, 228, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addContainerGap(90, Short.MAX\_VALUE))

);

layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()

.addContainerGap(168, Short.MAX\_VALUE)

.addComponent(btnStart)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(btnLeaderboard)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(btnQuit)

.addGap(37, 37, 37))

);

pack();

setLocationRelativeTo(null);

}// </editor-fold>

private void btnStartActionPerformed(java.awt.event.ActionEvent evt) {

this.setVisible(false);

Tetris.start();

}

private void btnQuitActionPerformed(java.awt.event.ActionEvent evt) {

System.exit(0);

}

private void btnLeaderboardActionPerformed(java.awt.event.ActionEvent evt) {

this.setVisible(false);

Tetris.showLeaderboard();

}

/\*\*

\* @param args the command line arguments

\*/

public static void main(String args[]) {

/\* Set the Nimbus look and feel \*/

//<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">

/\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.

\* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html

\*/

try {

for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {

if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());

break;

}

}

} catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(StartupForm.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(StartupForm.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(StartupForm.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(StartupForm.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

}

//</editor-fold>

/\* Create and display the form \*/

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

new StartupForm().setVisible(true);

}

});

}

// Variables declaration - do not modify

private javax.swing.JButton btnLeaderboard;

private javax.swing.JButton btnQuit;

private javax.swing.JButton btnStart;

// End of variables declaration

}

**//Tetris.java**

package tetris;

import javax.swing.JOptionPane;

public class Tetris {

private static GameForm gf;

private static StartupForm sf;

private static LeaderboardForm lf;

private static AudioPlayer audio = new AudioPlayer();

public static void start(){

gf.setVisible(true);

gf.startGame();

}

public static void showStartup() {

sf.setVisible(true);

}

public static void showLeaderboard(){

lf.setVisible(true);

}

public static void gameOver(int score){

playGameover();

String playerName = JOptionPane.showInputDialog("Game Over !\nPlease enter your name.");

gf.setVisible(false);

lf.addPlayer(playerName, score);

}

public static void playClearLine(){

audio.playClearLine();

}

public static void playGameover(){

audio.playGameover();

}

public static void main(String[] args){

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

gf = new GameForm();

sf = new StartupForm();

lf = new LeaderboardForm();

sf.setVisible(true);

}

});

}

}

**//TetrisBlock.java**

package tetris;

import java.awt.Color;

import java.util.Random;

public class TetrisBlock {

private int[][][][] allShapes = {

//I-Shape

{

//0 degree

{

{1},

{1},

{1},

{1}

, },

//90 degrees

{

{1, 1, 1, 1}

},

//180 degrees

{

{1},

{1},

{1},

{1}

},

//270 degrees

{

{1, 1, 1, 1}

}

},

//J-Shape

{

//0 degree

{

{0, 1},

{0, 1},

{1, 1}

},

//90 degrees

{

{1, 0, 0},

{1, 1, 1}

},

//180 degrees

{

{1, 1},

{1, 0},

{1, 0}

},

//270 degrees

{

{1, 1, 1},

{0, 0, 1}

}

},

//L-Shape

{

//0 degree

{

{1, 0},

{1, 0},

{1, 1}

},

//90 degrees

{

{1, 1, 1},

{1, 0, 0}

},

//180 degrees

{

{1, 1},

{0, 1},

{0, 1}

},

//270 degrees

{

{0, 0, 1},

{1, 1, 1}

}

},

//O-Shape

{

//0 degree

{

{1, 1},

{1, 1}

},

//90 degrees

{

{1, 1},

{1, 1}

},

//180 degrees

{

{1, 1},

{1, 1}

},

//270 degrees

{

{1, 1},

{1, 1}

}

},

//S-Shape

{

//0 degree

{

{1, 0},

{1, 1},

{0, 1}

},

//90 degrees

{

{0, 1, 1},

{1, 1, 0}

},

//180 degrees

{

{1, 0},

{1, 1},

{0, 1}

},

//270 degrees

{

{0, 1, 1},

{1, 1, 0}

}

},

//T-Shape

{

//0 degree

{

{1, 1, 1},

{0, 1 ,0}

},

//90 degrees

{

{0, 1},

{1, 1},

{0, 1}

},

//180 degrees

{

{0, 1, 0},

{1, 1, 1}

},

//270 degrees

{

{1, 0},

{1, 1},

{1, 0}

}

},

//Z-Shape

{

//0 degree

{

{1, 1, 0},

{0, 1, 1}

},

//90 degrees

{

{0, 1},

{1, 1},

{1, 0}

},

//180 degrees

{

{1, 1, 0},

{0, 1, 1}

},

//270 degrees

{

{0, 1},

{1, 1},

{1, 0}

}

}

};

private int[][] shape;

private Color color;

private int x, y;

private int[][][] shapes;

private int currentRotation;

private Color[] availableColors = {Color.GREEN, Color.RED, Color.BLUE,

Color.CYAN, Color.MAGENTA, Color.ORANGE,

Color.PINK, Color.LIGHT\_GRAY, Color.YELLOW};

public TetrisBlock(int[][] shape){

this.shape = shape;

initShapes();

}

private void initShapes(){

shapes = new int[4][][];

for(int i = 0; i < 4; i++){

int r = shape[0].length;

int c = shape.length;

shapes[i] = new int[r][c];

for(int y = 0; y < r; y++){

for(int x = 0; x < c; x++){

shapes[i][y][x] = shape[c - x - 1][y];

}

}

shape = shapes[i];

}

}

public void spawn(int gridWidth){

Random r = new Random();

currentRotation = r.nextInt( shapes.length );

shape = shapes[currentRotation];

y = -getHeight();

x = r.nextInt( gridWidth - getWidth() );

color = availableColors[r.nextInt( availableColors.length )];

}

public int[][] getShape(){ return shape; }

public Color getColor(){ return color; }

public int getHeight(){ return shape.length; }

public int getWidth(){ return shape[0].length; }

public int getX(){ return x; }

public void setX(int newX){ x = newX; }

public int getY(){ return y; }

public void setY(int newY){ y = newY; }

public void moveDown(){ y++; }

public void moveRight(){ x++; }

public void moveLeft(){ x--; }

public void rotate() {

currentRotation++;

if(currentRotation > 3) currentRotation = 0;

shape = shapes[currentRotation];

}

public int getBottomEdge(){ return y + getHeight(); }

public int getLeftEdge(){ return x; }

public int getRightEdge(){ return x + getWidth(); }

}

**//IShape.java**

package tetrisblocks;

import tetris.TetrisBlock;

public class IShape extends TetrisBlock{

public IShape(){

super( new int[][]{ {1, 1, 1, 1} } );

}

public void rotation(){

super.rotate();

if(this.getWidth() == 1){

this.setX(this.getX() + 1);

this.setY(this.getY() - 1);

}else{

this.setX(this.getX() - 1);

this.setY(this.getY() + 1);

}

}

}

**//JShape.java**

package tetrisblocks;

import tetris.TetrisBlock;

public class JShape extends TetrisBlock {

public JShape(){

super( new int[][]{ {0, 1},

{0, 1},

{1, 1} });

}

public void rotation(){

super.rotate();

if(this.getWidth() == 1){

this.setX(this.getX() + 1);

this.setY(this.getY() - 1);

}else{

this.setX(this.getX() - 1);

this.setY(this.getY() + 1);

}

}

}

**//LShape.java**

package tetrisblocks;

import tetris.TetrisBlock;

public class LShape extends TetrisBlock {

public LShape(){

super( new int[][]{ {1, 0},

{1, 0},

{1, 1} });

}

public void rotation(){

super.rotate();

if(this.getWidth() == 1){

this.setX(this.getX() + 1);

this.setY(this.getY() - 1);

}else{

this.setX(this.getX() - 1);

this.setY(this.getY() + 1);

}

}

}

**//OShape.java**

package tetrisblocks;

import tetris.TetrisBlock;

public class OShape extends TetrisBlock{

public OShape(){

super( new int[][]{ {1, 1},

{1, 1} });

}

public void rotation(){

super.rotate();

if(this.getWidth() == 1){

this.setX(this.getX() + 1);

this.setY(this.getY() - 1);

}else{

this.setX(this.getX() - 1);

this.setY(this.getY() + 1);

}

}

}

**//SShape.java**

package tetrisblocks;

import tetris.TetrisBlock;

public class SShape extends TetrisBlock{

public SShape(){

super( new int[][]{ {1, 0},

{1, 1},

{0, 1} });

}

public void rotation(){

super.rotate();

if(this.getWidth() == 1){

this.setX(this.getX() + 1);

this.setY(this.getY() - 1);

}else{

this.setX(this.getX() - 1);

this.setY(this.getY() + 1);

}

}

}

**//TShape.java**

package tetrisblocks;

import tetris.TetrisBlock;

public class TShape extends TetrisBlock{

public TShape(){

super( new int[][]{ {1, 1, 1},

{0, 1, 0} });

}

public void rotation(){

super.rotate();

if(this.getWidth() == 1){

this.setX(this.getX() + 1);

this.setY(this.getY() - 1);

}else{

this.setX(this.getX() - 1);

this.setY(this.getY() + 1);

}

}

}

**//ZShape.java**

package tetrisblocks;

import tetris.TetrisBlock;

public class ZShape extends TetrisBlock{

public ZShape(){

super( new int[][]{ {1, 1, 0},

{0, 1, 1} });

}

public void rotation(){

super.rotate();

if(this.getWidth() == 1){

this.setX(this.getX() + 1);

this.setY(this.getY() - 1);

}else{

this.setX(this.getX() - 1);

this.setY(this.getY() + 1);

}

}

}