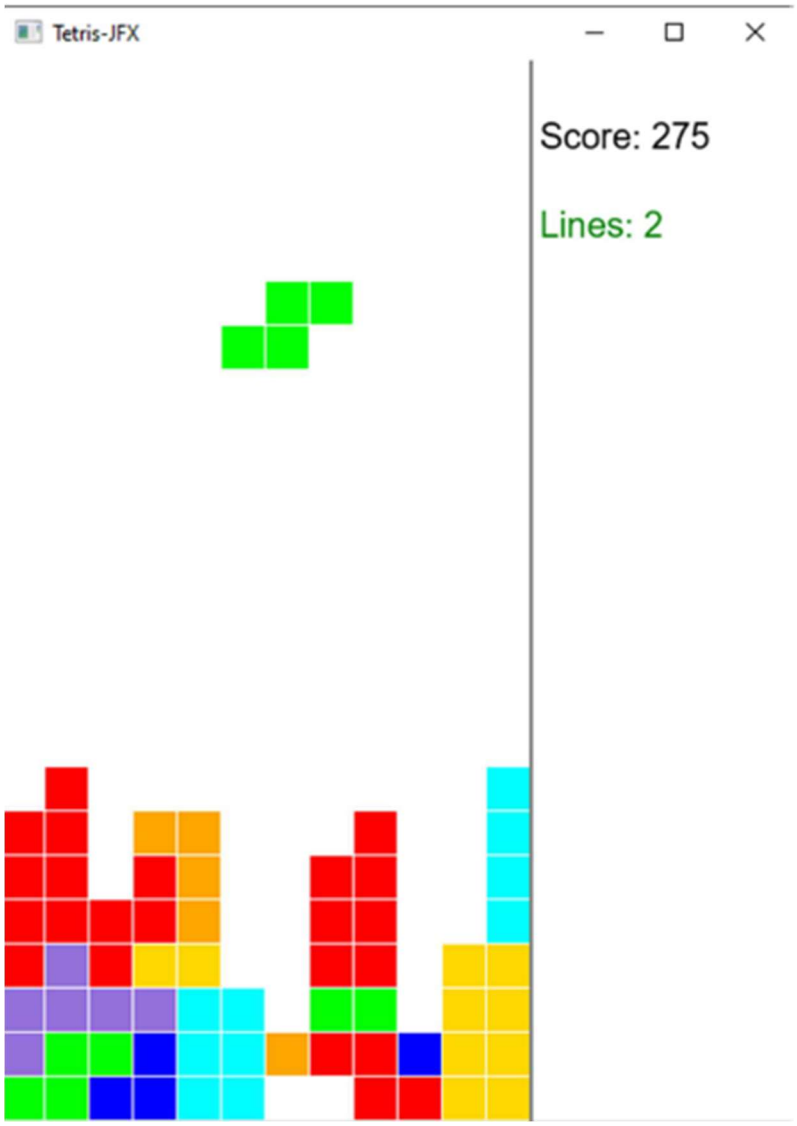
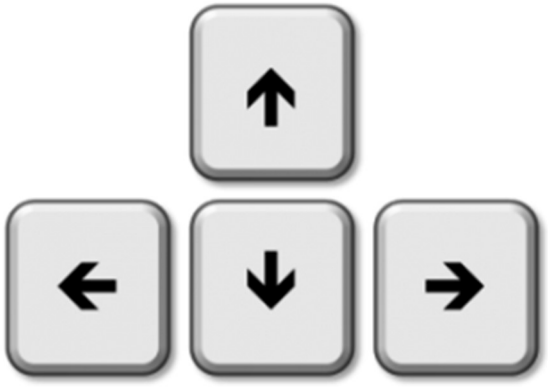
Tetris JFX Implementation manual

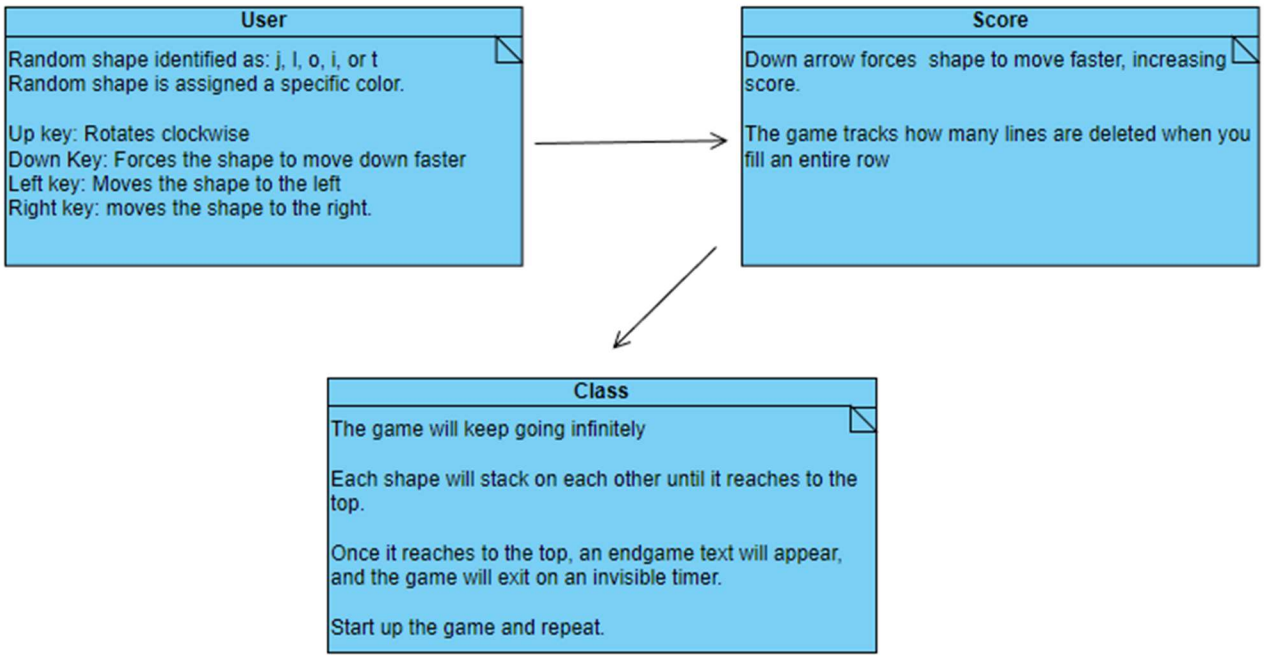
Scroll down for more details.

In case you have never played Tetris before, each shape appeared will be randomized with a distinct color.



Once you have your shape, you can use the arrow keys to control where you want to place it. The up arrow turns the shape clockwise, and the down arrow drags the shape down faster.





Once a shape hits the top of the screen, you lose. Repeat and have fun!


Tetris-JFX
Score: 301
Lines: 2
YOU LOST LOL

Details are marked with the “//” in the code.

```java
package application;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.Timer;
import java.util.TimerTask;
import javafx.application.Application;
import javafx.application.Platform;
import javafx.event.EventHandler;
import javafx.scene.Node;
import javafx.scene.Scene;
import javafx.scene.input.KeyEvent;
import javafx.scene.layout.Pane;
import javafx.scene.paint.Color;
import javafx.scene.shape.Line;
import javafx.scene.shape.Rectangle;
import javafx.scene.text.Text;
import javafx.stage.Stage;


public class App extends Application {

//v The variables
public static final int MOVE = 25;
//^ the game runs on a grid, making each moving shape move from one open tile to the next
public static final int SIZE = 25;
//^ the size of the tile of the GRID
public static int XWIDTH = SIZE * 12;
//^ game's width
public static int YHEIGHT = SIZE * 24;
//^ game's height
public static int[][] GRID = new int[XWIDTH / SIZE][YHEIGHT / SIZE];
//^ filling the game's area with the GRID
private static Pane group = new Pane();
private static Form object;
private static Scene scene = new Scene(group, XWIDTH + 150, YHEIGHT);
//^ side area containing the main score and line score
public static int score = 0;
private static int top = 0;
private static boolean game = true;
private static Form nextObj = controller.makeRect();
private static int linesNo = 0;









    Run | Debug
    public static void main(String[] args)  {
        launch(args);
    }//^ starts the game
```

```
    Line line = new Line(XWIDTH, 0, XWIDTH, YHEIGHT);
    Text scoretext = new Text("Score: ");
    scoretext.setStyle("-fx-font: 20 arial;");
    scoretext.setY(50);
    scoretext.setX(XWIDTH + 5);
    Text level = new Text("Lines: ");
    level.setStyle("-fx-font: 20 arial;");
    level.setY(100);
    level.setX(XWIDTH + 5);
    level.setFill(Color.GREEN);
    group.getChildren().addAll(scoretext, line, level);
    //^ this creates the black line seperating the game and score, and also creates the in game text

    Form a = nextObj;
    group.getChildren().addAll(a.a, a.b, a.c, a.d);
    moveOnKeyPress(a);
    object = a;
    nextObj = controller.makeRect();
    stage.setScene(scene);
    stage.setTitle("Tetris-JFX");
    stage.show();
    //^ tab title of the game


    //v time for the shapes to fall
    Timer fall = new Timer();
    TimerTask task = new TimerTask() {
        public void run() {
            Platform.runLater(new Runnable() {
                public void run() {
                    if (object.a.getY() == 0 || object.b.getY() == 0 || object.c.getY() == 0
                            || object.d.getY() == 0)
                        top++;
                    else
                        top = 0;

                    if (top == 2) {
                        // GAME OVER
                        Text over = new Text("YOU LOST LOL");
                        over.setFill(Color.RED);
                        over.setStyle("-fx-font: 60 arial;");
                        over.setY(250);
                        over.setX(10);
                        group.getChildren().add(over);
                        game = false;
                    }
                    // Exit once timer is hit
                    if (top == 15) {
                        System.exit(status: 0);
                    }

                    if (game) {
                        MoveDown(object);
                        scoretext.setText("Score: " + Integer.toString(score));
                        level.setText("Lines: " + Integer.toString(linesNo));
                    } // tracking and increasing score
                }
            });
        }
    };
    fall.schedule(task, delay: 0, period: 300);
    //^ falling speed
}
```

```
//moving the shape when you press the up key
private void MoveTurn(Form form) {
    int f = form.form;
    Rectangle a = form.a;
    Rectangle b = form.b;
    Rectangle c = form.c;
    Rectangle d = form.d;
    switch (form.getName()) {
    case "j":
        if (f == 1 && cB(a, x: 1, -1) && cB(c, -1, -1) && cB(d, -2, -2)) {
            MoveRight(form.a);
            MoveDown(form.a);
            MoveDown(form.c);
            MoveLeft(form.c);
            MoveDown(form.d);
            MoveDown(form.d);
            MoveLeft(form.d);
            MoveLeft(form.d);
            form.changeForm();
            break;
        }
        if (f == 2 && cB(a, -1, -1) && cB(c, -1, y: 1) && cB(d, -2, y: 2)) {
            MoveDown(form.a);
            MoveLeft(form.a);
            MoveLeft(form.c);
            MoveUp(form.c);
            MoveLeft(form.d);
            MoveLeft(form.d);
            MoveUp(form.d);
            MoveUp(form.d);
            form.changeForm();
            break;
        }
        if (f == 3 && cB(a, -1, y: 1) && cB(c, x: 1, y: 1) && cB(d, x: 2, y: 2)) {
            MoveLeft(form.a);
            MoveUp(form.a);
            MoveUp(form.c);
            MoveRight(form.c);
            MoveUp(form.d);
            MoveUp(form.d);
            MoveRight(form.d);
            MoveRight(form.d);
            form.changeForm();
            break;
        }
        if (f == 4 && cB(a, x: 1, y: 1) && cB(c, x: 1, -1) && cB(d, x: 2, -2)) {
            MoveUp(form.a);
            MoveRight(form.a);
            MoveRight(form.c);
            MoveDown(form.c);
            MoveRight(form.d);
            MoveRight(form.d);
            MoveDown(form.d);
            MoveDown(form.d);
            form.changeForm();
            break;
        }
        break;
```

This if statement code repeats for l, o, s, t, and z

```java
//v removes all the tiles that completes the row
private void RemoveRows(Pane pane) {
    ArrayList<Node> rects = new ArrayList<Node>();
    ArrayList<Integer> lines = new ArrayList<Integer>();
    ArrayList<Node> newrects = new ArrayList<Node>();
    int full = 0;
    //v checks which line is full
    for (int i = 0; i < GRID[0].length; i++) {
        for (int j = 0; j < GRID.length; j++) {
            if (GRID[j][i] == 1)
                full++;
        }
        if (full == GRID.length)
        lines.add(i);
        //lines.add(i + lines.size());
        full = 0;
    }

    //v deleting the row
    if (lines.size() > 0)
        do {
            for (Node node : pane.getChildren()) {
                if (node instanceof Rectangle)
                    rects.add(node);
            }
            score += 50;
            linesNo++;


            //v deleting block on row
            for (Node node : rects) {
                Rectangle a = (Rectangle) node;
                if (a.getY() == lines.get(index: 0) * SIZE) {
                    GRID[(int) a.getX() / SIZE][(int) a.getY() / SIZE] = 0;
                    pane.getChildren().remove(node);
                } else
                    newrects.add(node);
            }

            for (Node node : newrects) {
                Rectangle a = (Rectangle) node;
                if (a.getY() < lines.get(index: 0) * SIZE) {
                    GRID[(int) a.getX() / SIZE][(int) a.getY() / SIZE] = 0;
                    a.setY(a.getY() + SIZE);
                }
            }
            //v removes lines and clears arrays
            lines.remove(index: 0);
            rects.clear();
            newrects.clear();
            for (Node node : pane.getChildren()) {
                if (node instanceof Rectangle)
                    rects.add(node);
            }
            for (Node node : rects) {
                Rectangle a = (Rectangle) node;
                try {
                    GRID[(int) a.getX() / SIZE][(int) a.getY() / SIZE] = 1;
                } catch (ArrayIndexOutOfBoundsException e) {
                }
            }
            rects.clear();
        } while (lines.size() > 0);
}
```

```
//v control finally functions
private void MoveDown(Rectangle rect) {
    if (rect.getY() + MOVE < YHEIGHT)
        rect.setY(rect.getY() + MOVE);

}

private void MoveRight(Rectangle rect) {
    if (rect.getX() + MOVE <= XWIDTH - SIZE)
        rect.setX(rect.getX() + MOVE);
}

private void MoveLeft(Rectangle rect) {
    if (rect.getX() - MOVE >= 0)
        rect.setX(rect.getX() - MOVE);
}

private void MoveUp(Rectangle rect) {
    if (rect.getY() - MOVE > 0)
        rect.setY(rect.getY() - MOVE);
}


//v moving one block down if down is full
private void MoveDown(Form form) {
    if (form.a.getY() == YHEIGHT - SIZE || form.b.getY() == YHEIGHT - SIZE || form.c.getY() == YHEIGHT - SIZE
            || form.d.getY() == YHEIGHT - SIZE || moveA(form) || moveB(form) || moveC(form) || moveD(form)) {
        GRID[(int) form.a.getX() / SIZE][(int) form.a.getY() / SIZE] = 1;
        GRID[(int) form.b.getX() / SIZE][(int) form.b.getY() / SIZE] = 1;
        GRID[(int) form.c.getX() / SIZE][(int) form.c.getY() / SIZE] = 1;
        GRID[(int) form.d.getX() / SIZE][(int) form.d.getY() / SIZE] = 1;
        RemoveRows(group);

        Form a = nextObj;
        nextObj = controller.makeRect();
        object = a;
        group.getChildren().addAll(a.a, a.b, a.c, a.d);
        moveOnKeyPress(a);
    }

    //v moving one block down if down isnt full
    if (form.a.getY() + MOVE < YHEIGHT && form.b.getY() + MOVE < YHEIGHT && form.c.getY() + MOVE < YHEIGHT
            && form.d.getY() + MOVE < YHEIGHT) {
        int movea = GRID[(int) form.a.getX() / SIZE][((int) form.a.getY() / SIZE) + 1];
        int moveb = GRID[(int) form.b.getX() / SIZE][((int) form.b.getY() / SIZE) + 1];
        int movec = GRID[(int) form.c.getX() / SIZE][((int) form.c.getY() / SIZE) + 1];
        int moved = GRID[(int) form.d.getX() / SIZE][((int) form.d.getY() / SIZE) + 1];
        if (movea == 0 && movea == moveb && moveb == movec && movec == moved) {
            form.a.setY(form.a.getY() + MOVE);
            form.b.setY(form.b.getY() + MOVE);
            form.c.setY(form.c.getY() + MOVE);
            form.d.setY(form.d.getY() + MOVE);
        }
    }
}
```

```java
//v movement must be done for each tile
private boolean moveA(Form form) {
    return (GRID[(int) form.a.getX() / SIZE][((int) form.a.getY() / SIZE) + 1] == 1);
}

private boolean moveB(Form form) {
    return (GRID[(int) form.b.getX() / SIZE][((int) form.b.getY() / SIZE) + 1] == 1);
}

private boolean moveC(Form form) {
    return (GRID[(int) form.c.getX() / SIZE][((int) form.c.getY() / SIZE) + 1] == 1);
}

private boolean moveD(Form form) {
    return (GRID[(int) form.d.getX() / SIZE][((int) form.d.getY() / SIZE) + 1] == 1);
}

private boolean cB(Rectangle rect, int x, int y) {
    boolean xb = false;
    boolean yb = false;
    if (x >= 0)
        xb = rect.getX() + x * MOVE <= XWIDTH - SIZE;
    if (x < 0)
        xb = rect.getX() + x * MOVE >= 0;
    if (y >= 0)
        yb = rect.getY() - y * MOVE > 0;
    if (y < 0)
        yb = rect.getY() + y * MOVE < YHEIGHT;
    return xb && yb && GRID[((int) rect.getX() / SIZE) + x][((int) rect.getY() / SIZE) - y] == 0;
}
```