INTERNATIONAL UNIVERSITY - VNU HCMC

2024 - 2025 SEMESTER 01

IT069IU - OBJECT-ORIENTED PROGRAMING

DNAT GROUP FINAL PROJECT

TETRIS GAME

This report presents our final project for the OOP course: a Tetris game. The project demonstrates key OOP principles like encapsulation, inheritance, and polymorphism. It includes features such as dynamic block generation, collision detection, and a scoring system, showcasing how OOP is used to build engaging and maintainable software.

Contents

Chap 01: Introduction

1.1 About our group:

DNAT group is an Object-Oriented Programming project group. We have 4 members, including:

Full name - GitHub username	Student ID	Contribute	Note
Lê Võ Hồng Na HonggNa	ITITSB23007	100%	OptionPane.java + Application.java + UML model + Report + Presentation slide
Nguyễn Nhật Anh IcyKewtiie	ITITSB23006	100%	Board.java + Tetris.java + UML model + gathering all files and pushing to github
Phạm Quang Tường qtuong180402	ITDSIU20108	100%	MyShape.java + UML model + Report
Nguyễn Nhật Duy ericborder12	ITITWE22143	100%	StatusPane.java + UML model + Report

1.2 About the game project:

Tetris is one of the most iconic and enduring puzzle games in video game history. Created by Alexey Pajitnov in 1984, the game challenges players to manipulate falling geometric shapes, known as Tetrominoes, to form complete horizontal lines on a game board. When a line is successfully formed, it disappears, and the player earns points. As the game progresses, the falling blocks increase in speed, adding to the difficulty and excitement.

This report explores the key aspects of Tetris, including its gameplay mechanics, design, and impact on gaming culture. Additionally, the project demonstrates the

implementation of Tetris using modern programming techniques, highlighting challenges faced and solutions applied during the development process.

1.3 Our Tetris game:

As part of the ongoing development of our game project, we have introduced several new features aimed at enhancing the gameplay experience.

So first of all, we added some rules for our tetris game, that are

1. **Objective:**
- Arrange falling tetrominoes to create horizontal lines without gaps. When a line is completed, it disappears, and you earn points.
2. **Controls:**
- Move the blocks left or right, rotate the blocks to fit better and drop blocks faster by using the arrows keyboard.
3. **Game area:**
- The playfield is a grid, usually 10 columns wide and 20 rows tall.
4. **Tetromino shape:**
- Besides some basic shapes (such as: I, O, T, S, Z, J, L), we would like to add some special shapes.
5. **Clearing Lines**:
- Single line: 1 line cleared at once.
- Double, triple, or Tetris (4 lines) clears score higher points.
6. **Game over:**
- The game ends when blocks stack to the top of the playfield, and no more moves are possible.
7. **Increasing Difficulty**:
- As you play, the speed of falling blocks increases, making it harder to position them.

1.4 References:

Here are some key references that we use for our game project:

https://viblo.asia/p/cung-thu-viet-mot-game-xep-hinh-tetris-hoan-chinh-tu-con-so-0-phan-1-giao-dien-va-game-loop-gDVK2mdA5Lj

Chap 02:

Software Requirements

2.1 Functional Requirements:

1. **Game Mechanics**:
- The game must spawn random Tetrominoes (Tetris shapes) at the top of the game grid.
- Players can move Tetrominoes left, right, down, or rotate them.
- Completed rows should be cleared, and the score should increase.
2. **Level Progression**:
- The game speed should increase as the player completes more rows.
3. **Score System**:
- Award points for clearing rows. More rows cleared at once yield higher scores.
4. **Game Over**:
- The game ends when a Tetromino cannot fit at the top of the grid.
5. **Pause/Restart**:
- The game must include options to pause or restart.

2.2 Non-Functional Requirements

1. **Usability**:
- The controls should be intuitive and responsive.
2. **Scalability**:
- The code should be modular to allow adding features, such as additional Tetromino shapes, in the future.
3. **Cross-Platform**:
- The game should be playable on multiple platforms.

2.3 Technical Requirements

1. **Programming Language**: Java.
2. **Development Environment**: IDEs such as IntelliJ, Visual Studio, or Eclipse.
3. **Libraries**:

- Graphics handling library.
- Event-handling framework for controls.

4. **Design Pattern**:

- Use OOP principles like inheritance (Tetromino classes), encapsulation (game logic), and polymorphism (rotations and movements).

2.4 System Architecture

1. **Class Design**:

- **GameManager**: Controls game flow and state (start, restart, game over).
- **Grid**: Represents the playing field and manages row completion.
- **Tetromino**: Base class for shapes with derived classes (e.g: I, O, L, Z, etc.).
- **PlayerInput**: Handles keyboard events for movement and rotation.
- **ScoreTracker**: Calculates and displays the score.

2. **Graphics and UI**:

- Render the grid, Tetrominoes, and score using 2D graphics.

3. **New Features:**

- Using the slider to adjust the falling speed of the shapes.
- Using buttons to choose the difficulty level (including Normal, Hard and Extreme) which allow you to play with some special shape to engage game experience.
- Your highest score and the number of deleted lines will be recorded.
- Using a space key to make the shape fall immediately.
- Using the “s” key to increase the falling speed of the shapes.
- Using the “p” key to pause the game.

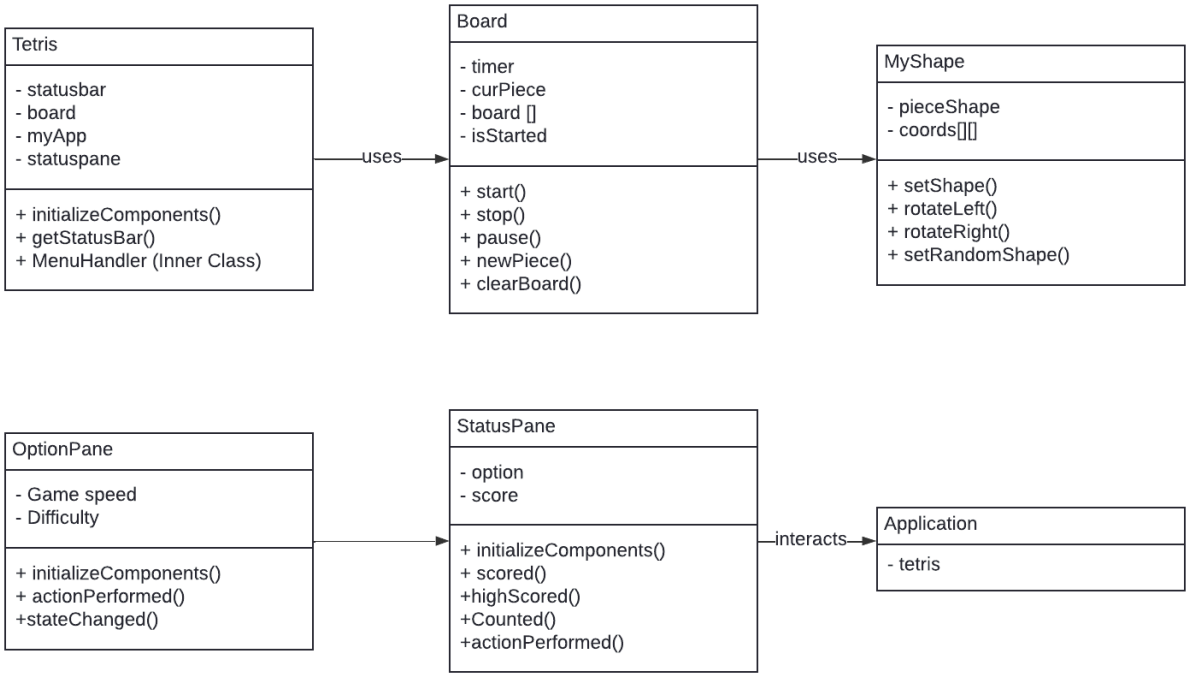
2.5 Use Case Scenario

We have created the use cases based on the Tetris game

Tetris game	Start game	Using arrow keys, “s” key, “p” key and space key to play game

	Setting	Game speed
		Difficulty

2.6 UML diagram



2.7 Solid Principle:

1. Single Responsibility Principle (SRP)

Application.java

- Purpose: Unclear due to lack of methods; likely acts as a container or utility class.
- Assessment: Requires further inspection to confirm adherence to SRP.

Board.java

- Purpose: Core game logic and board management.
- Assessment: Violates SRP if it handles too many unrelated concerns (e.g., game state, rendering, and user input).

MyShape.java

- Purpose: Represents and manipulates Tetris shapes.
- Assessment: Compliant with SRP as it focuses on shape-related functionality.

OptionPane.java

- Purpose: Manages user interactions through dialog boxes.

- Assessment: Compliant with SRP as it focuses solely on UI interactions.

  StatusPane.java
- Purpose: Tracks and updates game status (e.g., score, high score).
- Assessment: Compliant with SRP as it focuses on status updates.

  Tetris.java
- Purpose: Initializes and manages the game window and main menu.
- Assessment: Compliant with SRP if it limits itself to UI initialization and event handling.

2. Open-Closed Principle (OCP)

Board.java
- Assessment: Could violate OCP if changes to game logic require modification of the class rather than extending it.

  MyShape.java
- Assessment: Compliant with OCP if new shapes can be added without altering existing methods.

  OptionPane.java and **StatusPane.java**
- Assessment: Likely compliant with OCP if they allow customization through inheritance or configuration.

  Tetris.java
- Assessment: May violate OCP if modifications to UI require changing the class directly.

3. Liskov Substitution Principle (LSP)
- Subtypes must be substitutable for their base types without altering program behavior.
- No explicit inheritance hierarchies were observed, but further inspection is needed to confirm compliance.

4. Interface Segregation Principle (ISP)
- Classes should not be forced to implement interfaces they do not use.

  Board.java and **Tetris.java**
- Assessment: May violate ISP if they are required to implement unrelated methods due to broad interface contracts.
- Other classes seem focused, reducing the risk of ISP violations.

5. Dependency Inversion Principle (DIP)
- High-level modules should not depend on low-level modules; both should depend on abstractions.

  Board.java

- Assessment: Could violate DIP if it directly depends on concrete classes like MyShape or UI components.

Tetris.java

- Assessment: May violate DIP if it directly manages low-level UI components without abstractions.

Application.java

Class: Application

Methods: None listed.

Board.java

- Classes: Board, TAdapter
    + Methods: Extensive list, including game logic (e.g., addGameSpeed, start, pause, dropDown, newPiece, tryMove, etc.).

MyShape.java

- Class: MyShape
    + Methods: Related to handling shapes in the game (e.g., setShape, rotateLeft, rotateRight, setRandomShape, etc.).

OptionPane.java

- Class: OptionPane
    + Methods: User interaction handling (e.g., initializeComponents, actionPerformed, stateChanged).

StatusPane.java

- Class: StatusPane
    + Methods: Status tracking and updates (e.g., scored, highScored, Counted, etc.).

Tetris.java

- Classes: Tetris, MenuHandler
    + Methods: Game initialization and UI logic (e.g., initializeComponents, getStatusBar, actionPerformed).

Chap 03: Design & Implementation

Program Structure

1. Tetris.java

- Import library

```
3   import java.awt.BorderLayout;
4   import java.awt.event.ActionEvent;
5   import java.awt.event.ActionListener;
6   import java.awt.event.KeyEvent;
7   import javax.swing.JFrame;
8   import javax.swing.JLabel;
9   import javax.swing.JMenu;
10  import javax.swing.JMenuBar;
11  import javax.swing.JMenuItem;
12  import javax.swing.JPanel;
```

- Create JFrame containing main components (Board, StatusPane, OptionPane)

```java
public class Tetris extends JFrame {

    JLabel statusbar;
    Board board;
    Application myApp;
    StatusPane statuspane;
    public Tetris(Application app) {
        myApp = app;
        setSize(width:500, height:700);
        setTitle(title:"Tetris");
        setResizable(resizable:false);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        initializeComponents();
        setVisible(b:true);
    }
```

- Handle game menu with two options: New Game and Exit

```java
private void initializeComponents() {
    JMenuBar menubar = new JMenuBar();
    JMenu menu;
    JMenuItem menuitem;

    this.setJMenuBar(menubar);
    menu = new JMenu(s:"Game");
    menu.setMnemonic(KeyEvent.VK_G);
    menuitem = new JMenuItem(text:"New Game");
    menuitem.setMnemonic(KeyEvent.VK_N);
    menuitem.addActionListener(new MenuHandler(this));
    menu.add(menuitem);
    menu.addSeparator();
    menuitem = new JMenuItem(text:"Exit");
    menuitem.setMnemonic(KeyEvent.VK_X);
    menuitem.addActionListener(new MenuHandler(this));
    menu.add(menuitem);
    menubar.add(menu);
```

```java
    statuspane = new StatusPane(myApp);
    statusbar = new JLabel(text:"Choose New Game from the menu Game to start game.");
    board = new Board(myApp, this);

    JPanel sidePanel = new JPanel();
sidePanel.setLayout(new BorderLayout());

    sidePanel.add(statuspane, BorderLayout.NORTH);

// Thêm các thành phần vào JFrame
this.setLayout(new BorderLayout());
this.add(sidePanel, BorderLayout.EAST);   // Panel chứa StatusPane + HighScores bên phải
this.add(board, BorderLayout.CENTER);     // Bảng trò chơi ở giữa
    this.add(statusbar, BorderLayout.SOUTH);  // Thanh trạng thái bên dưới

}
```

- Implement the actions (Create new game and exit the game)

```
70    public class MenuHandler implements ActionListener
71    {
72        Tetris ter;
73        public MenuHandler(Tetris ter)
74        {
75            this.ter = ter;
76        }
77        public void actionPerformed(ActionEvent e) {
78            String s = e.getActionCommand();
79            if(s == "New Game")
80            {
81                board.start();
82            }
83            else if(s == "Exit")
84            {
85                ter.dispose();
86            }
87        }
88    }
```

2. StatusPane.java

- Import library

```
3   import java.awt.Color;
4   import java.awt.Dimension;
5   import java.awt.GridLayout;
6   import java.awt.event.ActionEvent;
7   import java.awt.event.ActionListener;
8   import javax.swing.BorderFactory;
9   import javax.swing.JButton;
10  import javax.swing.JLabel;
11  import javax.swing.JPanel;
```

- Create a controlling board at the right side of the window

```
14  public class StatusPane extends JPanel implements ActionListener{
15      Application myApp;
16      OptionPane option;
17      JLabel score, highestScores, count;
18      public StatusPane(Application app)
19      {
20          myApp = app;
21          setPreferredSize(new Dimension(width:120, height:625));
22          setBorder(BorderFactory.createEtchedBorder(Color.red, Color.blue));
23          initializeComponents();
24      }
```

- Show Start Game button and implement OptionPane to change the setting of the game. Also, create new labels called High Scores and Lines which indicates the number of lines completed.

```
25      private void initializeComponents() {
26          setLayout(new GridLayout( rows:10, cols:1));
27
28          option = new OptionPane(myApp);
29          JButton butStart = new JButton(text:"Start Game");
30          butStart.setPreferredSize(new Dimension(width:100, height:40));
31          butStart.addActionListener(this);
32          this.add(butStart);
33          JButton butOption = new JButton(text:"Setting");
34          butOption.setPreferredSize(new Dimension(width:100, height:40));
35          butOption.addActionListener(this);
36          this.add(butOption);
37          score = new JLabel(text:"Score: 0");
38          this.add(score);
39
40
41          highestScores = new JLabel(text:"High Scores: ");
42          this.add(highestScores);
43
44          count = new JLabel(text:"Lines: ");
45          this.add(count);
46      }
```

- Show current score

```
48      public void scored(int score)
49      {
50          this.score.setText("Score: "  +   Integer.toString(score));
51      }
```

- Calculate highest score

```
53      public void highScored(int[] highScores) {
54      StringBuilder scores = new StringBuilder(str:"<html>High Scores:<br>");
55      for (int i = 0; i < highScores.length; i++) {
56          scores.append( highScores[i]);
57      }
58      scores.append(str:"</html>");
59      this.highestScores.setText(scores.toString());
60      }
```

- Count the completed lines

```
62      public void Counted(int count){
63          this.count.setText("Line: "  +   Integer.toString(count));
64      }
```

- Implement actions when user clicks the button

```
67      public void actionPerformed(ActionEvent e) {
68          if(e.getActionCommand() == "Start Game"){
69              myApp.tetris.board.start();
70          }
71          else
72          {
73              myApp.tetris.board.stop();
74              option.setVisible(b:true);
75          }
76      }
```

3. OptionPane.java

- Import library

```
3   import java.awt.BorderLayout;
4   import java.awt.Dimension;
5   import java.awt.event.ActionEvent;
6   import java.awt.event.ActionListener;
7   import javax.swing.BorderFactory;
8   import javax.swing.ButtonGroup;
9   import javax.swing.JButton;
10  import javax.swing.JDialog;
11  import javax.swing.JLabel;
12  import javax.swing.JPanel;
13  import javax.swing.JRadioButton;
14  import javax.swing.JSlider;
15  import javax.swing.event.ChangeEvent;
16  import javax.swing.event.ChangeListener;
```

- Change speed

```
40      private void initializeComponents() {
41          // Label shows current speed and Slider to config the game speed
42          panel = new JPanel();
43          panel.setBorder(BorderFactory.createTitledBorder(title:"Game speed"));
44          label = new JLabel(text:"Current speed: 3");
45          panel.add(label, BorderLayout.NORTH);
46          speed = new JSlider(min:1, max:5, value:3);
47          speed.addChangeListener(this);
48          speed.setPreferredSize(new Dimension(width:250, height:50));
49          panel.add(speed, BorderLayout.CENTER);
50          this.add(panel, BorderLayout.NORTH);
51          panel.setVisible(aFlag:true);
```

- Select difficulty by clicking buttons (Normal, Hard, Extreme)

```
53          // Radio Button to set the difficulty in game
54          panel = new JPanel();
55          panel.setBorder(BorderFactory.createTitledBorder(title:"Difficulty"));
56          radio = new JRadioButton(text:"Normal");
57          radio.addActionListener(this);
58          radio.setSelected(b:true);
59          group.add(radio);
60          panel.add(radio);
61
62          radio = new JRadioButton(text:"Hard");
63          radio.addActionListener(this);
64          group.add(radio);
65          panel.add(radio);
66
67          radio = new JRadioButton(text:"Extreme");
68          radio.addActionListener(this);
69          group.add(radio);
70          panel.add(radio);
71          this.add(panel, BorderLayout.CENTER);
```

- Button OK and Cancel to confirm or cancel changes

```
73          // Button OK and Cancel
74          JPanel buttonPanel = new JPanel();
75
76          button = new JButton(text:"OK");
77          button.setPreferredSize(new Dimension(width:100, height:50));
78          button.setActionCommand(actionCommand:"OK");
79          button.addActionListener(this);
80          buttonPanel.add(button);
81
82          button = new JButton(text:"Cancel");
83          button.setPreferredSize(new Dimension(width:100, height:50));
84          button.setActionCommand(actionCommand:"Cancel");
85          button.addActionListener(this);
86          buttonPanel.add(button);
87
88          this.add(buttonPanel, BorderLayout.SOUTH);
```

- Implement actions when user clicks the button

```
94      @Override
95      public void actionPerformed(ActionEvent e) {
96          String com = e.getActionCommand();
97          if(com == "OK")
98          {
99              myApp.tetris.board.setSpeed(speed.getValue());
100             switch(difficulty)
101             {
102                 case 1:
103                     myApp.tetris.board.setDifficulty(numOfShape:7);
104                     break;
105                 case 2:
106                     myApp.tetris.board.setDifficulty(numOfShape:10);
107                     break;
108                 case 3:
109                     myApp.tetris.board.setDifficulty(numOfShape:13);
110                     break;
111             }
112             preSpeed = speed.getValue();
113             myApp.tetris.board.grabFocus();
114             this.setVisible(b:false);
115         }
```

```
116         else if(com == "Normal")
117         {
118             difficulty = 1;
119         }
120         else if(com == "Hard")
121         {
122             difficulty = 2;
123         }
124         else if(com == "Extreme")
125         {
126             difficulty = 3;
127         }
128         else if(com == "Cancel")
129         {
130             myApp.tetris.board.grabFocus();
131             this.setVisible(b:false);
132
133         }
134     }
```

- Show current speed using JSlider

```
137     public void stateChanged(ChangeEvent e) {
138         if (e.getSource().getClass().toString().endsWith(suffix:"JSlider")) {
139             label.setText("Current speed: " + speed.getValue());
140         }
141     }
```

4. MyShape.java

- Import library

```
3   import java.util.Random;
4   import java.lang.Math;
```

- Represent many shapes in Tetris using Enum

```
7   public class MyShape {
8
9       enum Tetrominoes { NoShape, ZShape, SShape, LineShape,
10                         TShape, SquareShape, LShape, MirroredLShape,
11                         CrossShape, TLongShape, SLongShape, SixSquareShape,
12                         AShape, CShape  };
```

- Define different blocks with specific coordinates

```
14      private Tetrominoes pieceShape;
15      private int coords[][];
16      private int[][][] coordsTable;
17
18
19      public MyShape() {
20
21          coords = new int[6][2];
22          setShape(Tetrominoes.NoShape);
23
24      }
```

- Create a three-dimensional array to store elements in 3D space

```
26      public void setShape(Tetrominoes shape) {
27
28          coordsTable = new int[][][] {
29                  // Normal
30              { { 0, 0 } ,  { 0, 0 },  { 0, 0 },  { 0, 0 },  { 0, 0 },  { 0, 0 } },
31              { { 0, -1 },  { 0, 0 },  { -1, 0},  { -1, 1},  { 0, 0 },  { 0, 0 } },
32              { { 0, -1 },  { 0, 0 },  { 1, 0 },  { 1, 1 },  { 0, 0 },  { 0, 0 } },
33              { { 0, -1 },  { 0, 0 },  { 0, 1 },  { 0, 2 },  { 0, 0 },  { 0, 0 } },
34              { { -1, 0 },  { 0, 0 },  { 1, 0 },  { 0, 1 },  { 0, 0 },  { 0, 0 } },
35              { { 0, 0 } ,  { 1, 0 },  { 0, 1 },  { 1, 1 },  { 0, 0 },  { 0, 0 } },
36              { { -1, -1},  { 0, -1},  { 0, 0 },  { 0, 1 },  { 0, 0 },  { 0, 0 } },
37              { { 1, -1 },  { 0, -1},  { 0, 0 },  { 0, 1 },  { 0, 0 },  { 0, 0 } },
38                  // Hard
39              { { -1, 0 },  { 0, 1 },  { 1, 0 },  { 0, -1},  { 0, 0 },  { 0, 0 } },
40              { { -1, -1 },  { 0, 1 },  { 1, 1 },  { 0, -1},  { 0, 0 },  { 0, 0 } },
41              { {-1, 1},  { 0, -1},  { 0, 0 },  { 0, 1 },  { 1, 1},  { 0, 0 }},
42                  // Extreme
43              { {-1, 0},  { -1, 1},  { 0, 0 },  { 0, 1 },  { 1, 1},  { 1, 0 }},
44              { {-1, 0},  { -1, 1},  { 0, 1 },  { 1, 1 },  { 1, 0},  { 0, 1 }},
45              { {-1, -1},  { -1, 1},  { 0, 0 },  { 0, 1 },  { 1, 1},  { 1, -1 }}
46          };
```

- Copy the coordinates of the shape from coords to coordsTable

```
48          for (int i = 0; i < 6 ; i++) {
49              for (int j = 0; j < 2; ++j) {
50                  coords[i][j] = coordsTable[shape.ordinal()][i][j];
51              }
52          }
53          pieceShape = shape;
```

- Setter to set the x and y coordinates for each point in the block

```
57      private void setX(int index, int x) { coords[index][0] = x; }
58      private void setY(int index, int y) { coords[index][1] = y; }
59      public int x(int index) { return coords[index][0]; }
60      public int y(int index) { return coords[index][1]; }
61      public Tetrominoes getShape()  { return pieceShape; }
```

- Pick a random shape from enum Tetrominoes

```
63      public void setRandomShape(int num)
64      {
65          Random r = new Random();
66          int x = Math.abs(r.nextInt()) % num + 1;
67          Tetrominoes[] values = Tetrominoes.values();
68          setShape(values[x]);
69      }
```

- Find the minimum value of x and y in the coords

```
71      public int minX()
72      {
73        int m = coords[0][0];
74        for (int i = 0; i < 6; i++) {
75            m = Math.min(m, coords[i][0]);
76        }
77        return m;
78      }
```

```
81      public int minY()
82      {
83        int m = coords[0][1];
84        for (int i = 0; i < 6; i++) {
85            m = Math.min(m, coords[i][1]);
86        }
87        return m;
88      }
```

- Create methods to rotate the block

```
90      public MyShape rotateLeft()
91      {
92          if (pieceShape == Tetrominoes.SquareShape || pieceShape == Tetrominoes.CrossShape)
93              return this;
94
95          MyShape result = new MyShape();
96          result.pieceShape = pieceShape;
97
98          for (int i = 0; i < 6; ++i) {
99              result.setX(i, y(i));
100             result.setY(i, -x(i));
101         }
102         return result;
103     }
```

```
105     public MyShape rotateRight()
106     {
107         if (pieceShape == Tetrominoes.SquareShape || pieceShape == Tetrominoes.CrossShape)
108             return this;
109
110         MyShape result = new MyShape();
111         result.pieceShape = pieceShape;
112
113         for (int i = 0; i < 6; ++i) {
114             result.setX(i, -y(i));
115             result.setY(i, x(i));
116         }
117         return result;
118     }
```

5. Board.java

- Import library

```
3   import java.awt.Color;
4   import java.awt.Dimension;
5   import java.awt.Graphics;
6   import java.awt.Graphics2D;
7   import java.awt.event.ActionEvent;
8   import java.awt.event.ActionListener;
9   import java.awt.event.KeyAdapter;
10  import java.awt.event.KeyEvent;
11  import javax.swing.BorderFactory;
12  import javax.swing.JLabel;
13  import javax.swing.JPanel;
14  import javax.swing.Timer;
```

- Constructor where we set the size and the frame of the board, create objects for current shape, timer, field for board status. There are also KeyListener to let user play from keyboard and clearBoard() in order to delete the entire board

```
39      public Board(Application app, Tetris parent) {
40          myApp = app;
41          setPreferredSize(new Dimension(width:300, height:600));
42          setFocusable(focusable:true);
43          setBorder(BorderFactory.createEtchedBorder(Color.black, Color.green));
44          curPiece = new MyShape();
45          timer = new Timer(900 / propSpeed, this);
46
47          statusbar = parent.getStatusBar();
48          highScoresLabel = new JLabel();
49          highScoresLabel.setText(text:"High Scores: 0");
50          parent.add(highScoresLabel);
51          board = new MyShape.Tetrominoes[BoardWidth * BoardHeight];
52          addKeyListener(new TAdapter());
53          clearBoard();
54          System.out.println(x:"Update High Score: ");
55          for (int score : highestScores) {
56              System.out.println(score);
57          }
58      }
```

- Update speed and restart timer

```
60      public void addGameSpeed(int newSpeed) {
61          this.propSpeed = newSpeed; // Cập nhật tốc độ
62          statusbar.setText("Game Speed: " + newSpeed + ". Restart to apply changes.");
63          timer.stop();
64          timer = new Timer(1000 / propSpeed, this); // Khởi động lại Timer
65      }
```

- Draw blocks falling down one-by-one by actionPerformed() function, which spawn a new block when the current one have finished falling and continues falling one line down otherwise

```
67      public void actionPerformed(ActionEvent e) {
68          if (isFallingFinished) {
69              isFallingFinished = false;
70              newPiece();
71          } else {
72              oneLineDown();
73          }
74      }
```

- Control the board by three functions start(), pause(), stop()

```
88      public void start() {
89          if (isPaused) {
90              return;
91          }
92          this.grabFocus();
93          isStarted = true;
94          isFallingFinished = false;
95          numLinesRemoved = 0;
96          score = 0; // Reset score at the start of a new game
97          count = 0;
98          clearBoard();
99          newPiece();
100         timer.start();
101         statusbar.setText(text:"Game Started!");
102     }
```

```
104     public void pause() {
105         if (!isStarted) {
106             return;
107         }
108
109         isPaused = !isPaused;
110         if (isPaused) {
111             timer.stop();
112             statusbar.setText(text:"Game paused - Press P to continue.");
113         } else {
114             timer.start();
115             statusbar.setText(String.valueOf(numLinesRemoved));
116         }
117         repaint();
118     }
```

```
120     public void stop() {
121         curPiece.setShape(MyShape.Tetrominoes.NoShape);
122         timer.stop();
123         isStarted = false;
124         statusbar.setText(text:"GAME OVER!");
125         count = 0;
126         myApp.tetris.statuspane.setEnabled(enabled:true);
127 
128         updateHighScores(); // Update high scores with the current score
129     }
130             // Save high scores to a file
```

- Use paintComponent() to draw the board and shapes

```
132     public void paintComponent(Graphics g) {
133         Graphics2D g2D = (Graphics2D) g;
134         super.paintComponent(g2D);
135 
136         Dimension size = getSize();
137         int boardTop = (int) size.getHeight() - BoardHeight * squareHeight();
138 
139         g2D.setColor(Color.BLACK);  // Chọn màu cho lưới
140         for (int x = 0; x < BoardWidth; x++) {
141             for (int y = 0; y < BoardHeight; y++) {
142                 // Vẽ các đường kẻ dọc
143                 g2D.drawLine(x * squareWidth(), y1:0, x * squareWidth(), BoardHeight * squareHeight());
144                 // Vẽ các đường kẻ ngang
145                 g2D.drawLine(x1:0, y * squareHeight(), BoardWidth * squareWidth(), y * squareHeight());
146             }
147         }
```

```
151         for (int i = 0; i < BoardHeight; ++i) {
152             for (int j = 0; j < BoardWidth; ++j) {
153                 MyShape.Tetrominoes shape = shapeAt(j, BoardHeight - i - 1);
154                 if (shape != MyShape.Tetrominoes.NoShape) {
155                     drawSquare(g2D, 0 + j * squareWidth(),
156                                boardTop + i * squareHeight(), shape);
157                 }
158             }
159         }
160 
161         if (curPiece.getShape() != MyShape.Tetrominoes.NoShape) {
162             for (int i = 0; i < 6; ++i) {
163                 int x = curX + curPiece.x(i);
164                 int y = curY - curPiece.y(i);
165                 drawSquare(g2D, 0 + x * squareWidth(),
166                            boardTop + (BoardHeight - y - 1) * squareHeight(),
167                            curPiece.getShape());
168             }
169         }
170     }
```

- Create functions to control how block falls down

```
172     private void dropDown() {
173         int newY = curY;
174         while (newY > 0) {
175             if (!tryMove(curPiece, curX, newY - 1)) {
176                 break;
177             }
178             --newY;
179         }
180         pieceDropped();
181     }
182 
183     private void oneLineDown() {
184         if (!tryMove(curPiece, curX, curY - 1)) {
185             pieceDropped();
186         }
187     }
```

- Create a function to clear the board when it is called (when users start a new game)

```
189     private void clearBoard() {
190         for (int i = 0; i < BoardHeight * BoardWidth; ++i) {
191             board[i] = MyShape.Tetrominoes.NoShape;
192         }
193         repaint();
194     }
```

- Implement logic of the game

```
196     private void pieceDropped() {
197         for (int i = 0; i < 6; ++i) {
198             int x = curX + curPiece.x(i);
199             int y = curY - curPiece.y(i);
200             board[(y * BoardWidth) + x] = curPiece.getShape();
201         }
202 
203         removeFullLines();
204 
205         if (!isFallingFinished) {
206             newPiece();
207         }
208     }
```

```
210     private void newPiece() {
211         curPiece.setRandomShape(numOfShape);
212         curX = BoardWidth / 2 + 1;
213         curY = BoardHeight - 1 + curPiece.minY();
214 
215         if (!tryMove(curPiece, curX, curY)) {
216             stop();
217         }
218     }
```

- Check and move the block if relevant when user move the block

```
220     private boolean tryMove(MyShape newPiece, int newX, int newY) {
221         for (int i = 0; i < 6; ++i) {
222             int x = newX + newPiece.x(i);
223             int y = newY - newPiece.y(i);
224             if (x < 0 || x >= BoardWidth || y < 0 || y >= BoardHeight) {
225                 return false;
226             }
227             if (shapeAt(x, y) != MyShape.Tetrominoes.NoShape) {
228                 return false;
229             }
230         }
231 
232         curPiece = newPiece;
233         curX = newX;
234         curY = newY;
235         repaint();
236         return true;
237     }
```

- Create a function to delete completed rows, update score and set block's state fixed

```
239     private void removeFullLines() {
240         statusbar.setText(text:"On the move...");
241         int numFullLines = 0;
242         for (int i = BoardHeight - 1; i >= 0; --i) {
243             boolean lineIsFull = true;
244 
245             for (int j = 0; j < BoardWidth; ++j) {
246                 if (shapeAt(j, i) == MyShape.Tetrominoes.NoShape) {
247                     lineIsFull = false;
248 
249                     break;
250                 }
251             }
```

```
253             if (lineIsFull) {
254                 ++numFullLines;
255                 ++count;
256                 myApp.tetris.statuspane.Counted(count);
257                 if (count >= 2) {
258                     statusbar.setText(count + " combo!");
259                     score += 500 * propSpeed * count;
260                 } else {
261                     statusbar.setText(text:"Good!");
262                     score += 500 * propSpeed;
263                 }
264                 for (int k = i; k < BoardHeight - 1; ++k) {
265                     for (int j = 0; j < BoardWidth; ++j) {
266                         board[(k * BoardWidth) + j] = shapeAt(j, k + 1);
267                     }
268                 }
269                 myApp.tetris.statuspane.scored(score);
270 
271             }
```

```
274         if (numFullLines > 0) {
275             numLinesRemoved += numFullLines;
276             isFallingFinished = true;
277             curPiece.setShape(MyShape.Tetrominoes.NoShape);
278             repaint();
279         }
280     }
```

- Draw squares with lines which have as same color as the block

```
282     private void drawSquare(Graphics2D g2D, int x, int y, MyShape.Tetrominoes shape) {
283         Color colors[] = {new Color(r:0, g:0, b:0), new Color(r:204, g:102, b:102),
284                 new Color(r:102, g:204, b:102), new Color(r:102, g:102, b:204),
285                 new Color(r:204, g:204, b:102), new Color(r:204, g:102, b:204),
286                 new Color(r:102, g:204, b:204), new Color(r:218, g:170, b:0),
287                 new Color(r:15, g:232, b:76), new Color(r:123, g:200, b:152),
288                 new Color(r:12, g:25, b:136), new Color(r:154, g:232, b:76),
289                 new Color(r:100, g:200, b:150), new Color(r:70, g:125, b:136)
290         };
291 
292         Color color = colors[shape.ordinal()];
293 
294         g2D.setColor(color);
295         g2D.fillRect(x + 1, y + 1, squareWidth() - 2, squareHeight() - 2);
296 
297         g2D.setColor(color.brighter());
298         g2D.drawLine(x, y + squareHeight() - 1, x, y);
299         g2D.drawLine(x, y, x + squareWidth() - 1, y);
300 
301         g2D.setColor(color.darker());
302         g2D.drawLine(x + 1, y + squareHeight() - 1,
303                 x + squareWidth() - 1, y + squareHeight() - 1);
304         g2D.drawLine(x + squareWidth() - 1, y + squareHeight() - 1,
305                 x + squareWidth() - 1, y + 1);
```

- Set speed with new timer

```
308     public void setSpeed(int speed) {
309         this.propSpeed = speed;
310         timer.stop();
311         timer = new Timer(1000 / speed, this);
312         timer.start();  // Đảm bảo Timer mới sẽ bắt đầu với tốc độ mới
313     }
```

- Change difficulty

```
315     public void setDifficulty(int numOfShape) {
316         this.numOfShape = numOfShape;
317         statusbar.setText(text:"You need to start game again...");
318         timer.stop();
319         timer = new Timer(1000 / propSpeed, this);
320     }
```

- Update high scores and insert it

```
321     // Check if the current score qualifies as a high score
322     private void updateHighScores() {
323     for (int i = 0; i < highestScores.length; i++) {
324         if (score > highestScores[i]) {
325             // Shift lower scores down
326             for (int j = highestScores.length - 1; j > i; j--) {
327                 highestScores[j] = highestScores[j - 1];
328             }
329             highestScores[i] = score; // Insert the new high score
330 
331             break;
332         }
333 
334     }
```

- Control logic of keyboard

```
346     class TAdapter extends KeyAdapter {
347
348         public void keyPressed(KeyEvent e) {
349
350             if (!isStarted || curPiece.getShape() == MyShape.Tetrominoes.NoShape) {
351                 return;
352             }
353
354             int keycode = e.getKeyCode();
355
356             if (keycode == 'p' || keycode == 'P') {
357                 pause();
358                 return;
359             }
360
361             if (isPaused) {
362                 return;
363             }
```

```
365             switch (keycode) {
366                 case KeyEvent.VK_LEFT:
367                     tryMove(curPiece, curX - 1, curY);
368                     break;
369                 case KeyEvent.VK_RIGHT:
370                     tryMove(curPiece, curX + 1, curY);
371                     break;
372                 case KeyEvent.VK_DOWN:
373                     tryMove(curPiece.rotateRight(), curX, curY);
374                     break;
375                 case KeyEvent.VK_UP:
376                     tryMove(curPiece.rotateLeft(), curX, curY);
377                     break;
378                 case KeyEvent.VK_SPACE:
379                     dropDown();
380                     break;
381                 case 's':
382                     oneLineDown();
383                     break;
384                 case 'S':
385                     oneLineDown();
386                     break;
```

6. Application.java

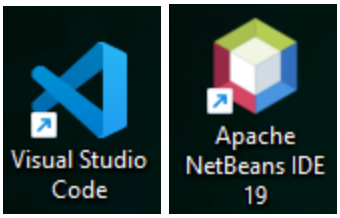
- Contain the main() function to run the application

```
4   public class Application {
5
6       Tetris tetris;
7
8       public Application() {
9           tetris = new Tetris(this);
10          tetris.setLocationRelativeTo(c:null);
11      }
12
        Run | Debug
13      public static void main(String[] args) {
14          new Application();
15      }
16
17  }
```

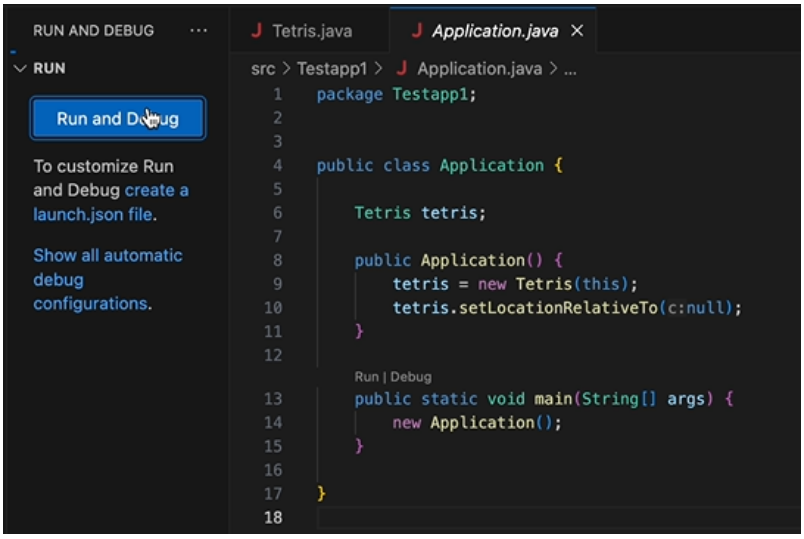
Chap 04: Demo

Instruction

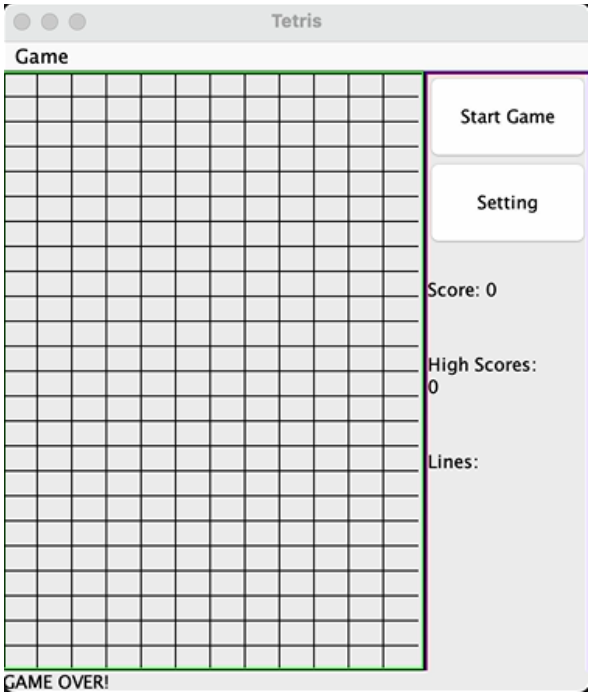
1. **Open source code files with IDE (VSCode, NetBeans, etc..)**



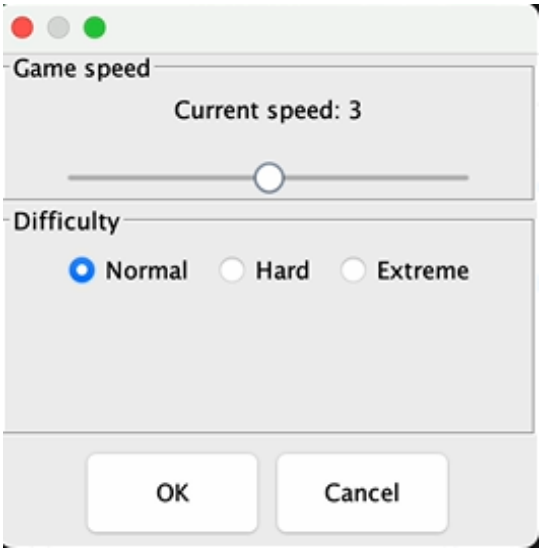
2. **Choose application.java and click Run**



3. A board game will appear on the screen

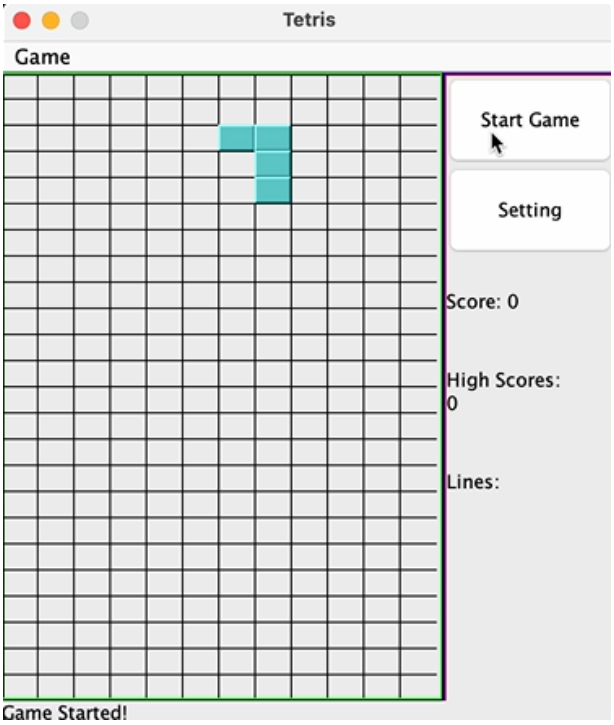


4. You can choose speed and difficulty in setting



5. For example, I want speed to be 1, then click OK

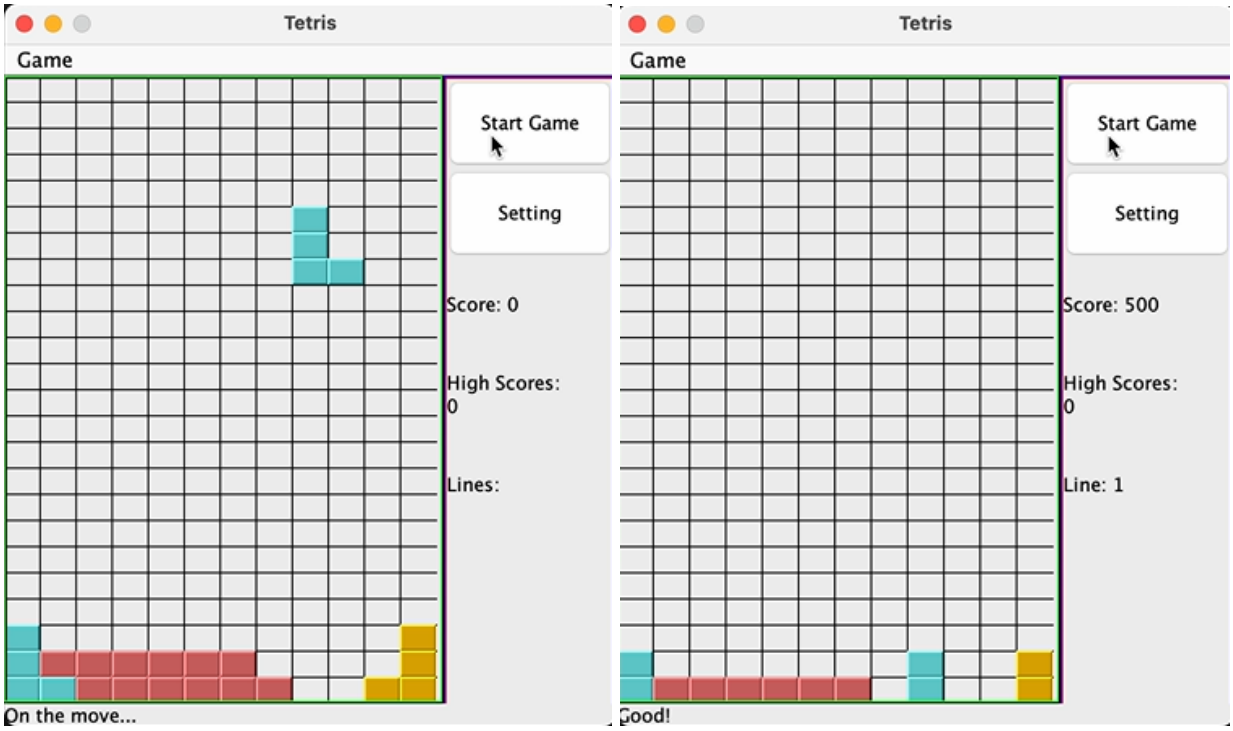
6. Click Start Game button to start a new game



A new block appears and falls down.

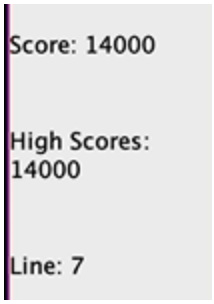
You can keep pressing the ‘s’ button to make the block fall down faster or press Space to let the block fall immediately.

7. A row is terminated when it is completed



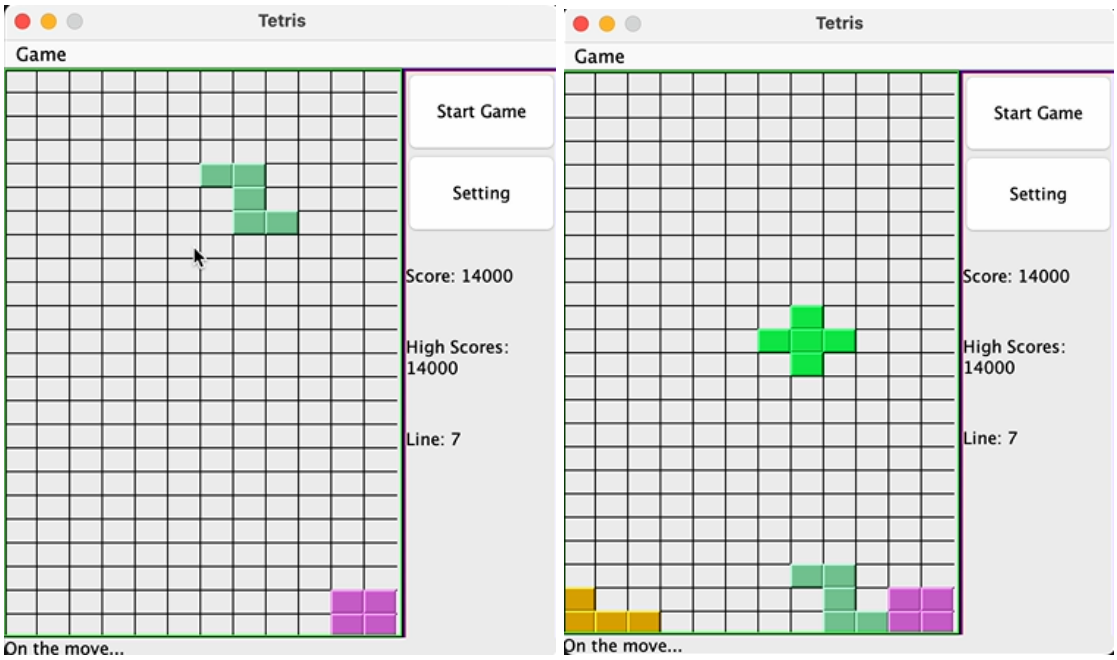
At the time, the system will update line number and score.

8. The game will save your high scores and the most number of lines completed



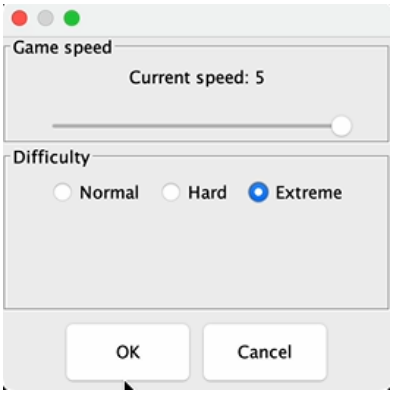
9. **If the game is easy, you can change difficulty to make it more challenging**

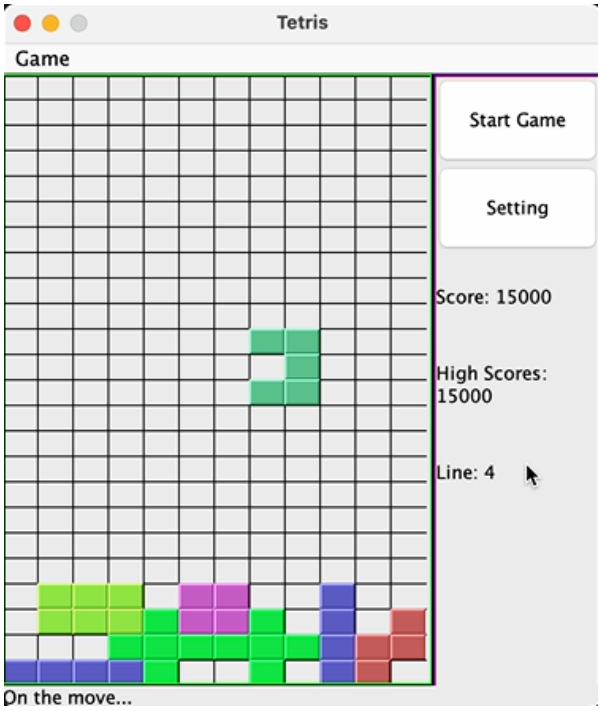
10. **Click Start game button to start a new game with new setting**

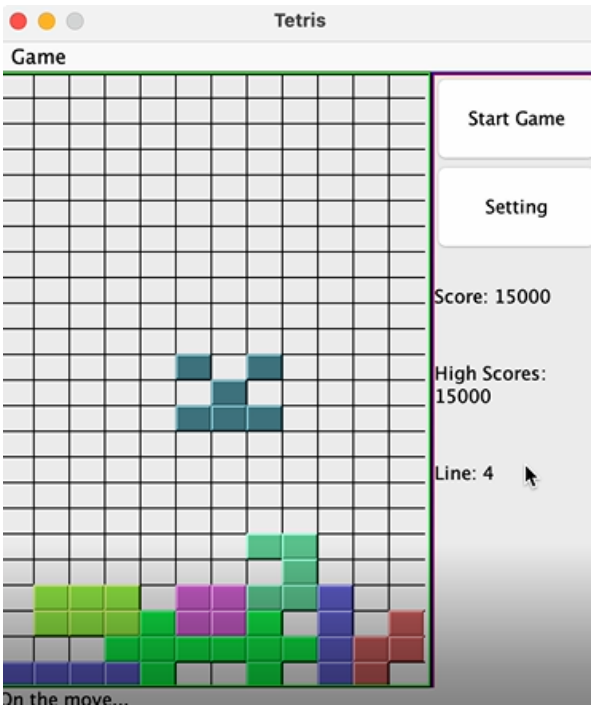


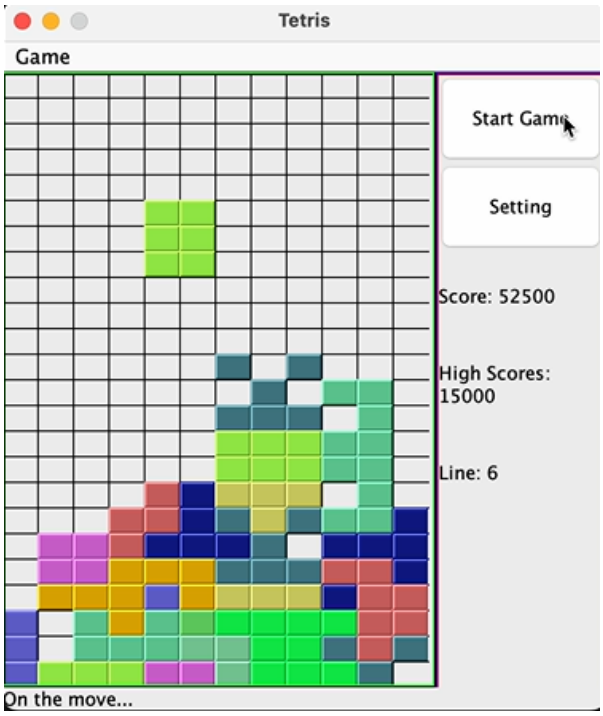
In hard mode, the game spawns some new types of shape, which make it harder to fit with other blocks.

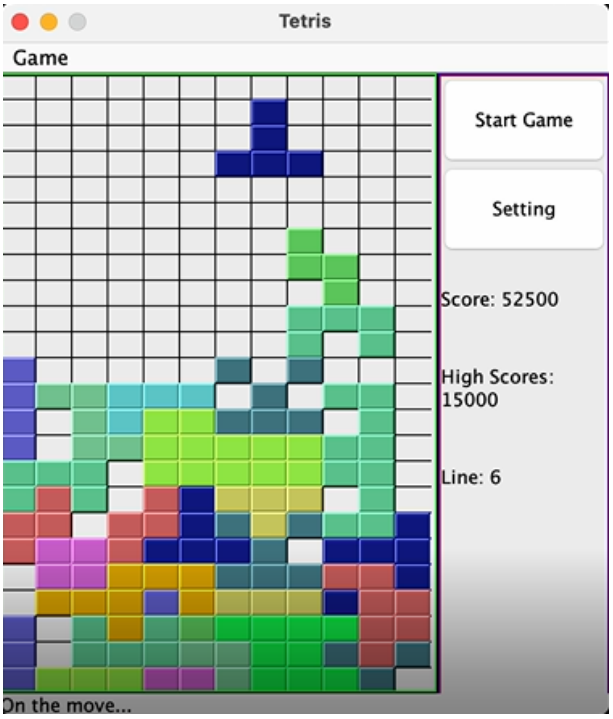
11. Finally, you can try Extreme mode, an upgrade version of Hard mode where special blocks spawn more frequent




Tetris
Game
Start Game
Setting
Score: 15000
High Scores:
15000
Line: 4
On the move...


Tetris
Game
Start Game
Setting
Score: 15000
High Scores:
15000
Line: 4


Tetris
Game
Start Game
Setting
Score: 52500
High Scores:
15000
Line: 6
On the move...


Tetris
Game
Start Game
Setting
Score: 52500
High Scores:
15000
Line: 6
On the move...