俄羅斯方塊專題功能列項

2024/5/2

<https://www.youtube.com/watch?v=qBuYOJTk_Nk>

【基本題】

實現一個 Tetris（俄羅斯方塊）遊戲涉及許多核心功能，這些功能是遊戲能夠運行和玩家能夠互動的基礎。以下是十項必須實現的遊戲功能：

1. **遊戲界面初始化**：
   * 創建遊戲窗口和主遊戲畫面，包括遊戲區域和顯示下一個方塊的區域。
2. **方塊生成**：
   * 隨機生成各種形狀的方塊（如 I, O, T, S, Z, J, L）並在遊戲區域的頂部中央位置顯示。
3. **方塊控制**：
   * 允許玩家使用鍵盤操作（左移、右移、旋轉、快速下降）來控制方塊的移動和方向。
4. **方塊下落**：
   * 方塊自動以固定速度向下移動，速度隨遊戲進程逐漸加快。
5. **方塊堆疊**：
   * 當方塊移動到底部或堆疊在其他方塊上時，方塊停止移動並固定位置。
6. **行消除**：
   * 完整填滿一行方塊時，該行消除並加分，上方的方塊下移填補空位。
7. **得分系統**：
   * 設計得分機制，基於消除行數計分，並可能包括連續消除加分。
8. **遊戲結束條件**：
   * 當方塊堆積到遊戲區域頂部時遊戲結束。
9. **下一方塊預覽**：
   * 在遊戲界面旁顯示下一個將要出現的方塊形狀，以便玩家提前規劃移動。
10. **遊戲等級與速度調整**：
    * 隨著玩家的得分增加，遊戲等級提升，方塊下落的速度加快。

這些基本功能是任何 Tetris 遊戲的必要組件，確保了遊戲的可玩性和挑戰性。每一個功能的實現都需要仔細的計劃和程式設計，這樣才能創建出一個流暢和有趣的遊戲體驗。

【進階題】

若要進一步擴展 Tetris 遊戲的功能，可以考慮增加以下五項額外功能，這些將會提高遊戲的互動性和戰略性，使遊戲更加豐富和有趣：

1. **自定義控制和主題**：
   * 允許玩家自定義控制設置，選擇不同的按鍵來移動和旋轉方塊。同時，提供多種主題和背景音樂選擇，增加視覺和聽覺的多樣性。
2. **挑戰模式**：
   * 增加特殊的挑戰模式，例如時間限制模式，在一定時間內清除盡可能多的行；或者難度逐步增加的生存模式，挑戰玩家的極限。
3. **成就和排行榜**：
   * 設計一個成就系統，獎勵玩家達成特定目標，如連續消除多行、達到一定得分等。並且提供在線排行榜，讓玩家能夠看到自己在全球玩家中的排名。
4. **特殊方塊**：
   * 引入特殊方塊，如可以消除整個行或列的「炸彈」方塊，或者有其他獨特效果的方塊，增加遊戲的多樣性和策略層面。
5. **多人模式**：
   * 增加一個多人遊戲模式，允許兩位或更多玩家在分割畫面或通過網絡連接競爭。玩家可以互相攻擊，比如發送額外的行到對方遊戲區域。

這些額外功能將進一步增加 Tetris 遊戲的吸引力和重玩價值，提供更多自定義選擇和社交互動，使遊戲變得更加有趣和具挑戰性。這些改進也可以作為教學課題，幫助學生理解更複雜的程式設計和遊戲開發概念。

【實現建議】

實現一個基本的 Tetris 遊戲涉及多個步驟，包括遊戲邏輯的設計、用戶界面的開發以及事件處理。以下是一個建議的實作流程，使用 Java 和其 Swing 库來開發：

**1. 遊戲界面與基本設定**

* **創建遊戲窗口**：使用 **JFrame** 來創建遊戲的主窗口。
* **遊戲面板**：使用 **JPanel**，在其中繪製和更新遊戲的方塊。

**2. 遊戲元素設計**

* **方塊類型**：定義一個 **Tetromino** 類，它包括方塊的形狀、顏色和旋轉狀態。
* **遊戲板**：創建一個二維數組來代表遊戲板，存儲方塊的位置。

**3. 遊戲邏輯**

* **生成方塊**：隨機生成新的 **Tetromino**，並處理方塊的初始位置。
* **方塊控制**：添加鍵盤監聽器來處理用戶的移動（左、右、下、旋轉）指令。
* **方塊下落**：使用計時器（例如 **javax.swing.Timer**）來控制方塊的自動下落。

**4. 碰撞檢測與遊戲規則**

* **碰撞檢測**：在方塊移動或旋轉前檢測是否會和遊戲板上已存在的方塊或邊界發生碰撞。
* **行消除**：檢查並消除填滿的行，並根據消除的行數計算得分。

**5. 遊戲過程控制**

* **遊戲結束條件**：如果新生成的方塊在放置時已經和現有的方塊重疊，遊戲結束。
* **遊戲重置**：提供重新開始遊戲的選項。

**6. 圖形與音效**

* **繪製方塊**：在 **JPanel** 的 **paintComponent** 方法中實現方塊的繪製。
* **添加音效**：對關鍵動作（如方塊旋轉、行消除）添加音效反饋。

**程式碼示例**

以下是一個簡單的框架，用來啟動 Tetris 遊戲窗口：

import javax.swing.JFrame;

import javax.swing.JPanel;

import java.awt.Graphics;

public class Tetris extends JFrame {

public Tetris() {

this.setSize(300, 600);

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

this.add(new GamePanel());

this.setVisible(true);

}

public static void main(String[] args) {

new Tetris();

}

private class GamePanel extends JPanel {

@Override

protected void paintComponent(Graphics g) {

super.paintComponent(g);

// 畫遊戲元素

}

}

}

深入到 Tetris 遊戲的實作細節，我們將涵蓋遊戲的核心邏輯，包括方塊的生成、控制和碰撞檢測等，以及如何用 Java Swing 建構一個響應用戶操作的流暢用戶界面。

**7. 方塊的表示和生成**

* **方塊數據結構**：使用二維數組表示每種方塊（Tetromino）。每個方塊的旋轉可以表示為不同的二維數組配置。
* **方塊生成**：實現一個方塊生成器，用隨機數選擇下一個出現的方塊類型並處理其初始位置和方向。

**8. 方塊的控制和移動**

* **鍵盤控制**：使用 **KeyListener** 接口監聽鍵盤事件。具體來說，響應左、右、下箭頭鍵和空格鍵（用於旋轉）的操作。
* **移動和旋轉**：實現方法來處理方塊的左右移動、快速下移和旋轉。每次移動前，都要檢查目的地是否空閒（無其他方塊和邊界限制）。

**9. 碰撞檢測**

* **邊界檢測**：確保方塊在移動或旋轉時不會越過遊戲面板的邊界。
* **互相碰撞檢測**：當方塊移動或旋轉時，檢查是否會與已放置的方塊重疊。

**10. 遊戲進程管理**

* **自動下落**：使用 **javax.swing.Timer** 定時讓當前活動的方塊向下移動一格。這個動作的速度隨著遊戲進行逐漸增加。
* **行消除與得分計算**：檢查並消除填滿的行，根據消除的行數更新得分。實現累加得分和可能的多行消除加分。

**11. 遊戲結束和重新開始**

* **遊戲結束條件檢查**：如果新生成的方塊在頂部就無法放置（與其他方塊重疊），則遊戲結束。
* **重新開始功能**：提供用戶重新開始遊戲的選項，重置遊戲板和得分。

**12. 使用者界面和音效**

* **界面美化**：使用不同的顏色和形狀來區分不同的 Tetromino。設計一個簡潔明了的界面來顯示即將出現的方塊和當前得分。
* **音效添加**：對方塊移動、旋轉、行消除等事件添加聲音反饋，增加遊戲的沉浸感。

**13. 程式碼實例：方塊移動和旋轉**

public void move(int dx, int rotation) {

int newX = x + dx;

int newY = y;

int newRotation = (currentRotation + rotation) % 4;

if (isValidPosition(newX, newY, newRotation)) {

x = newX;

y = newY;

currentRotation = newRotation;

repaint();

}

}

在 Tetris 遊戲中，**Tetromino** 是一個核心的類別，代表了所有可以在遊戲中操作的方塊。每種 **Tetromino** 的形狀可以通過一組固定的格子（cells）來表示，這些格子在二維平面上定義了方塊的結構。以下是如何實現這個類別的一個基本概念：

**1. 定義 Tetromino 屬性**

每個 **Tetromino** 需要有幾個基本的屬性：

* **形狀（Shape）**：不同的形狀代表不同類型的 Tetromino（如 I, O, T, S, Z, J, L）。
* **顏色**：每種形狀可以有一個特定的顏色。
* **當前旋轉**：表示當前方塊的旋轉狀態。
* **位置**：方塊在遊戲板上的位置（通常以方塊左上角在遊戲板的格子位置來定義）。

**2. Tetromino 构造方法**

方塊的構造方法需要初始化這些屬性。例如，當一個新的 **Tetromino** 被創建時，它的形状、顏色和初始位置應該被設定。

**3. 方塊的旋轉**

方塊的旋轉可以通過一個二維數組來實現。每種形狀可以有一組不同的配置，每次旋轉改變到下一個配置。通常，旋轉可以通過固定旋轉中心，然後改變方塊的格子相對於這個中心的位置來實現。

**4. 移動操作**

**Tetromino** 類別應該提供方法來處理左移、右移和下移。這些方法應該檢查移動後是否會有碰撞或超出邊界的情況發生。

**5. 碰撞檢測**

**Tetromino** 類別可以有一個方法來檢查給定位置是否是合法的，這個方法會在旋轉或移動操作之前被調用，以確保這些操作不會導致方塊超出遊戲區域或進入其他方塊佔用的區域。

**示例代碼**

下面是一個 **Tetromino** 類別的簡化示例，展示了如何初始化並實現基本操作：

public class Tetromino {

private int[][] shape;

private Color color;

private int x, y; // 方塊在遊戲板上的位置

private int currentRotation = 0; // 當前旋轉索引

public Tetromino(int[][] shape, Color color, int startX, int startY) {

this.shape = shape;

this.color = color;

this.x = startX;

this.y = startY;

}

public void rotate() {

// 這裡假設 shape 每個旋轉是 90 度，並且按照一個固定的旋轉矩陣來實現

// 轉換當前 shape 數組來達到旋轉的效果

}

public void moveLeft() {

x--;

// 需要添加檢查碰撞和邊界檢測

}

public void moveRight() {

x++;

// 需要添加檢查碰撞和邊界檢測

}

public void moveDown() {

y++;

// 需要添加檢查碰撞和邊界檢測

}

// 繪製方塊的方法

public void draw(Graphics g) {

g.setColor(color);

for (int row = 0; row < shape.length; row++) {

for (int col = 0; col < shape[row].length; col++) {

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

g.fillRect((x + col) \* blockSize, (y + row) \* blockSize, blockSize, blockSize);

}

}

}

}

}