Shape, square

Description automatically generated

**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI**

**UNIVERSITY OF TRANSPOST AND COMMUNICATIONS**

Logo

Description automatically generated

**BÁO CÁO BÀI TẬP LỚN JAVA**

**Đề tài: Làm game 2D (game xếp gạch)**

|  |  |
| --- | --- |
| **Giáo viên hướng dẫn :** | **Vũ Huấn** |
| **Sinh viên thực hiện :** | **Phạm Hùng Dũng** |
|  | **Đỗ Thành Lưu** |
| **Lớp :** | **CNTT6 - k62** |

**Hà Nội, ngày 04 tháng 05 năm 2023**

**I. Giới thiệu đề tài: Game tetris (game xếp gạch)**

**1. Mô tả game:**

* Game có kích thước của cửa sổ là 400x600 pixel có chiều rộng chứa 14 ô vuông và chiều cao chứa 20 ô vuông với mỗi ô vuông có kích thước khoảng 28x30 pixel.
* Game có 7 khối gạch hình chữ S,Z,T,L,đường thẳng,hình vuông và các hình ngược lại của các chữ. Mỗi hình được tạo bởi 4 ô vuông. Các khối sẽ rơi từ trên xuống. Mỗi lần chạy chương trình là game sẽ tự động chạy ngay, dùng các ngút “🡨” để sang bên trái, nút “🡪” để sang bên phải, nút “🡩” để thay đổi hướng của các ô gạch, nút “🡫 ” để cho viên gạch rơi xuống ngay lập tức, nút “Space” để tạm dùng trò chơi. Nếu các ô gạch rơi xuống hết được một hàng sẽ được cộng một điểm đồng thời hàng đó sẽ mất và các hàng phía trên sẽ rơi xuống một ô, tương tự cho nhiều ô gạch. Game kết thúc khi khối gạch rơi tiếp theo cao hơn khoảng trống của các ô gạch đã chơi với mức giới hạn của game.

**2. Cấu trúc bài gồm 3 class:**

* Class Tetris: Là lớp làm công việc khởi tạo cửa sổ, khởi tạo kích thước, chạy luồng main.
* Class Shape: Là lớp lưu thông tin các khối gạch
* Class Board: Là lớp sự kiện (điều khiển các khối gạch và xử lý các hoạt động trong game).

**II. Phân tích.**

**1. Class tetris.**

-Chương trình được thực thi từ phương thức **“main()”** và bao gồm **lớp tetris** kế thừa từ **lớp JFrame.**

public Tetris() {

        initUI();

   }

   private void initUI() {

        statusbar = new JLabel(" 0");

        add(statusbar, BorderLayout.SOUTH);

        Board board = new Board(this);

        add(board);

        board.start();

        setSize(200, 400);

        setTitle("Tetris");

        setDefaultCloseOperation(EXIT\_ON\_CLOSE);

        setLocationRelativeTo(null);

   }

-Phương thức **“initUI()”** được dùng để thiết lập giao diện người dùng.

-Nhãn **JLabel** được tạo ra để hiển thị điểm của người chơi và được thêm bố vào bố cục **BorderLayout** ở dưới cùng của khung chứa **JFrame**. Sau đó một đối tượng **Board** được tạo và thêm vào khung chứa **JFrame**.

-Phương thức start() của đối tượng Board được gọi để bắt đầu trò chơi.

-Các phương thức **setSize(), setTitle(), setDefaultCloseOperation(), setLocationRelativeTo(null)** được sử dùng cài đặt kích thước, tiêu đề, để thoát khỏi trò chơi khi tắt và đặt vị trí của cửa sổ vào giữa màn hình.

 public JLabel getStatusBar() {

       return statusbar;

   }

-Phương thức getStatusBar() được sử dụng để truy cập đối tượng JLabel hiển thị điểm của người chơi

public static void main(String[] args) {

        SwingUtilities.invokeLater(new Runnable() {

            @Override

            public void run() {

                Tetris game = new Tetris();

                game.setVisible(true);

            }

        });

    }

Trong phương thức main() có:

**-SwingUtilities.invokeLater()** được sử dụng để khởi tạo một luồng trình diễn ra cùng với luồng hiện tại để hiển thị trò chơi.

**2. Class Shape.**

protected enum Tetrominoes { NoShape, ZShape, SShape, LineShape,

            TShape, SquareShape, LShape, MirroredLShape };

-Các khối hình được định nghĩa trong **enum Tetrominoes.** Ngoài 7 khối cơ bản thì sẽ có thêm 1 khối đặc biệt là “**NoShape**: khối này sẽ không có hình thù công dụng để xóa bảng và tạo khối mới.”

public Shape() {

    coords = new int[4][2];

    setShape(Tetrominoes.NoShape);

}

-Phương thức này sẽ tạo mảng “**coords**” để lưu giữ tọa đô của khối gạch.Phương thức **“setShape()”** dùng để đặt hình dạng của khối gạch được chỉ định.

coordsTable = new int[][][] {

   { { 0, 0 },   { 0, 0 },   { 0, 0 },   { 0, 0 } },

   { { 0, -1 },  { 0, 0 },   { -1, 0 },  { -1, 1 } },

   { { 0, -1 },  { 0, 0 },   { 1, 0 },   { 1, 1 } },

   { { 0, -1 },  { 0, 0 },   { 0, 1 },   { 0, 2 } },

   { { -1, 0 },  { 0, 0 },   { 1, 0 },   { 0, 1 } },

   { { 0, 0 },   { 1, 0 },   { 0, 1 },   { 1, 1 } },

   { { -1, -1 }, { 0, -1 },  { 0, 0 },   { 0, 1 } },

   { { 1, -1 },  { 0, -1 },  { 0, 0 },   { 0, 1 } }

};

-Mảng **“coordsTable”** sẽ lưu tọa độ của từng khối gạch

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

    for (int j = 0; j < 2; ++j) {

        coords[i][j] = coordsTable[shape.ordinal()][i][j];

    }

}

-Phương thức **“setShape()”** Sẽ quy định đối tượng khối gạch đó là loại khối gạch nào với tham số của phương thức này là 1 giá trị trong **“enumTetrominoes ”** và từu tham số đó sẽ lấy thứ tự của thuộc tính đó bằng phương thức **“ordinal()”** rồi dùng số thứ tự đó để lấy tọa độ trong mảng “coordsTable” và chépvafo mảng **“coords”.**

 public void setRandomShape() {

        Random r = new Random();

        int x = Math.abs(r.nextInt()) % 7 + 1;

        Tetrominoes[] values = Tetrominoes.values();

        setShape(values[x]);

    }

-Phương thức  **“setRandomShape()”** sẽ chọn ngẫu nhiên một loại gạch cho nó rơi xuống.

 public int minX() {

      int m = coords[0][0];

      for (int i=0; i < 4; i++) {

          m = Math.min(m, coords[i][0]);

      }

      return m;

    }

    public int minY() {

      int m = coords[0][1];

      for (int i=0; i < 4; i++) {

          m = Math.min(m, coords[i][1]);

      }

      return m;

    }

-Phương thức **“minX() và minY()”** dùng để tìm tọa độ x và y tối thiểu của các ô trong khối.

public Shape rotateLeft() {

if (pieceShape == Tetrominoes.SquareShape)

return this;

Shape result = new Shape();

result.pieceShape = pieceShape;

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

result.setX(i, y(i));

result.setY(i, -x(i));

}

return result;

}

-Phương thức này sẽ làm nhiệm vụ xoay khối gạch qua bên trái.

**3. Class Board.**

**-Board** kế thừa **JPanel** và triển khai **ActionListener** để theo dõi sự kiện phím.

-Trong lớp **Board** có 1 số biến sau:

 private final int BoardWidth = 14;

    private final int BoardHeight = 20;-

**-BoardWidth** và **BoardHeight**: chỉ số lượng ô vuông của chiều rộng và chiều cao

timer = new Timer(400, this);

timer.start();

**-timer**: Đối tượng Time dùng để cập nhập khối gạch khi nó rơi xuống

   private boolean isFallingFinished = false;

    private boolean isStarted = false;

    private boolean isPaused = false;

**- isFallingFinished, isStarted** và **isPaused**: các biến **boolean** để quản lý trạng thái trò chơi

private int numLinesRemoved = 0;

**- numLinesRemoved**: lưu số lượng các đường ngang đã bị xóa.

private int curX = 0;

    private int curY = 0;

**- curX và curY**: tọa độ của khối hình hiện tại.

-**statusbar**:Một **JLabel** để hiện thị trạng thái của trò chơi.

-**board**: Một mảng để lưu trữ tất cả các khối gạch của trò chơi.

Một số phương thức của Class Board:

 public Board(Tetris parent) {

        initBoard(parent);

    }

    private void initBoard(Tetris parent) {

       setFocusable(true);

       curPiece = new Shape();

       timer = new Timer(400, this);

       timer.start();

       statusbar =  parent.getStatusBar();

       board = new Tetrominoes[BoardWidth \* BoardHeight];

       addKeyListener(new TAdapter());

       clearBoard();

    }

**-initBoard**: Phương thức dùng để khởi tạo trò chơi.

public void actionPerformed(ActionEvent e) {

        if (isFallingFinished) {

            isFallingFinished = false;

            newPiece();

        } else {

            oneLineDown();

        }

    }

**-actionPerformed**: Phương thức được khai từ **ActionListener** để xử lsy sự kiện thời gian.

   public void start()  {

        if (isPaused)

            return;

        isStarted = true;

        isFallingFinished = false;

        numLinesRemoved = 0;

        clearBoard();

        newPiece();

        timer.start();

    }

    private void pause()  {

        if (!isStarted)

            return;

        isPaused = !isPaused;

        if (isPaused) {

            timer.stop();

            statusbar.setText("paused");

        } else {

            timer.start();

            statusbar.setText(String.valueOf(numLinesRemoved));

        }

        repaint();

    }

-**start** và **pause**: Phương thức để bắt đầu và tạm dừng trò chơi.

  private void doDrawing(Graphics g) {

        Dimension size = getSize();

        int boardTop = (int) size.getHeight() - BoardHeight \* squareHeight();

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

            for (int j = 0; j < BoardWidth; ++j) {

                Tetrominoes shape = shapeAt(j, BoardHeight - i - 1);

                if (shape != Tetrominoes.NoShape)

                    drawSquare(g, 0 + j \* squareWidth(),

                               boardTop + i \* squareHeight(), shape);

            }

        }

        if (curPiece.getShape() != Tetrominoes.NoShape) {

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

                int x = curX + curPiece.x(i);

                int y = curY - curPiece.y(i);

                drawSquare(g, 0 + x \* squareWidth(), boardTop + (BoardHeight - y - 1) \* squareHeight(), curPiece.getShape());

            }

        }

    }

    @Override

    public void paintComponent(Graphics g) {

        super.paintComponent(g);

        doDrawing(g);

    }

**-doDrawing** và **paintComponent**: Phương thức vẽ các khối gạch và bảng trò chơi.

 private void dropDown() {

        int newY = curY;

        while (newY > 0) {

            if (!tryMove(curPiece, curX, newY - 1))

                break;

            --newY;

        }

        pieceDropped();

    }

-**dropDown**: Phương thức để di chuyển khối gạch xuống.

 private void clearBoard() {

        for (int i = 0; i < BoardHeight \* BoardWidth; ++i)

            board[i] = Tetrominoes.NoShape;

    }

-**clearBoard**: Phương thức dùng để xóa tất cả các khối gạch trên bảng trò chơi.

    private void pieceDropped() {

        for (int i = 0; i < 4; ++i) {

            int x = curX + curPiece.x(i);

            int y = curY - curPiece.y(i);

            board[(y \* BoardWidth) + x] = curPiece.getShape();

        }

        removeFullLines();

        if (!isFallingFinished)

            newPiece();

    }

-**pieceDropped**: Phương thức được gọi khi một khối hình đã rơi xuống và được đặt xuống dưới bảng chơi.

 private void newPiece()  {

        curPiece.setRandomShape();

        curX = BoardWidth / 2 + 1;

        curY = BoardHeight - 1 + curPiece.minY();

        if (!tryMove(curPiece, curX, curY)) {

            curPiece.setShape(Tetrominoes.NoShape);

            timer.stop();

            isStarted = false;

            statusbar.setText("game over");

        }

    }

-**newPiece**: Phương thức dùng để tạo khối gạch mới.

 private boolean tryMove(Shape newPiece, int newX, int newY) {

        for (int i = 0; i < 4; ++i) {

            int x = newX + newPiece.x(i);

            int y = newY - newPiece.y(i);

            if (x < 0 || x >= BoardWidth || y < 0 || y >= BoardHeight)

                return false;

            if (shapeAt(x, y) != Tetrominoes.NoShape)

                return false;

        }

        curPiece = newPiece;

        curX = newX;

        curY = newY;

        repaint();

        return true;

    }

-**tryMove**: Phương thức dùng để kiểm tra xem khối gạch có thể di chuyển đến vị trí mới hay không.

 class TAdapter extends KeyAdapter {

         @Override

         public void keyPressed(KeyEvent e) {

             if (!isStarted || curPiece.getShape() == Tetrominoes.NoShape) {

                 return;

             }

             int keycode = e.getKeyCode();

             if (keycode == KeyEvent.VK\_SPACE) {

                 pause();

                 return;

             }

             if (isPaused)

                 return;

             switch (keycode) {

             case KeyEvent.VK\_LEFT:

                 tryMove(curPiece, curX - 1, curY);

                 break;

             case KeyEvent.VK\_RIGHT:

                 tryMove(curPiece, curX + 1, curY);

                 break;

             case KeyEvent.VK\_UP:

                 tryMove(curPiece.rotateLeft(), curX, curY);

                 break;

             case KeyEvent.VK\_DƠN:

                 dropDown();

                 break;

             }

         }

     }

-Lớp **TAdapter** là lớp sự kiện các phím chơi trò chơi.

**III. Câu hỏi thêm khi thuyết trinh:** Tạo nút restart khi gameover

public Board(Tetris parent) {

initBoard(parent);

}

// private void restartGame() {

// isStarted = false;

// isPaused = false;

// isFallingFinished = false;

// numLinesRemoved = 0;

// clearBoard();

// newPiece();

// timer.start();

// statusbar.setText("0");

//

// }

private void initBoard(Tetris parent) {

// JButton restartButton = new JButton("Restart");

// add(restartButton);

//

// restartButton.addActionListener(new ActionListener() {

// @Override

// public void actionPerformed(ActionEvent e) {

// restartGame();

//

// }

// });

**IV.Phần code:** https://github.com/phamdung128/gameTetris.