**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI**

**KHOA CÔNG NGHỆ THÔNG TIN**

------------\*\*\*-----------



**BÁO CÁO BÀI TẬP LỚN MÔN LẬP TRÌNH JAVA**

**ĐỀ TÀI:**

**THIẾT KẾ GAME TETRIS**

|  |  |  |  |
| --- | --- | --- | --- |
| **Giảng viên hướng dẫn** | **:** | **TS. BÙI HUẤN** |  |
| **Sinh viên thực hiện** | **:** | **Huỳnh Thành Vinh** | **211213982** |

**Hà Nội, tháng 05 năm 2023**

**Lời nói đầu**

Công nghệ thông tin ngày càng có vai trò quan trọng trong cuộc sống hàng ngày của chúng ta. Việc ứng dụng CNTT vào các lĩnh vực trong đời sống giúp công việc được tiến hành nhanh chóng và hiệu quả hơn. Có rất nhiều công việc mới phát triển song song với sự phát triển của CNTT , một trong số đó là hiệu ứng game , hướng đi dịch vụ mang lại hiệu quả kinh tế rất lớn.

Chúng em chọn đề tài “ lập trình game Tetris bằng ngôn ngữ java” nhằm tìm hiểu sâu hơn về ngôn ngữ java trong lập trình hướng đối tượng , từ đó viết một ứng dụng cụ thể thực nghiệm làm cơ sở kiến thức và định hướng, kế hoạch xây dựng những ứng dụng game cụ thể , phát triển theo hướng dịch vụ trong tương lai. Phần bài làm của chúng em còn nhiều thiếu sót rất mong nhận được sự góp ý từ thầy cô cùng tất cả mọi người. Chúng em xin chân thành cảm ơn.

**Mục lục**

**[I.](#_Toc134034933)****[Giới thiệu đề tài](#_Toc134034933)** [4](#_Toc134034933)

**[1.](#_Toc134034934)****[Mô tả game](#_Toc134034934)** [4](#_Toc134034934)

**[2.](#_Toc134034935)****[Phương hướng](#_Toc134034935)** [4](#_Toc134034935)

**[II.](#_Toc134034936)****[Xây dựng và viết code chương trình](#_Toc134034936)** [4](#_Toc134034936)

**[1.](#_Toc134034937)****[Lớp Tetris](#_Toc134034937)** [4](#_Toc134034937)

**[2.](#_Toc134034938)****[Lớp shape](#_Toc134034938)** [5](#_Toc134034938)

**[3.](#_Toc134034939)****[Lớp Board](#_Toc134034939)** [9](#_Toc134034939)

**[III.](#_Toc134034940)****[Tổng kết](#_Toc134034940)** [13](#_Toc134034940)

**[1.](#_Toc134034941)****[Kết luận](#_Toc134034941)** [13](#_Toc134034941)

**[2.](#_Toc134034942)****[Tài liệu tham khảo](#_Toc134034942)** [13](#_Toc134034942)

1. **Giới thiệu đề tài**

Lập trình trò chơi Tetris yêu cầu:

* Chơi được.
* Đảm bảo xây dựng theo đúng nguyên tắc của mô hình 3 lớp.
* Một số yêu cầu phát sinh trong quá trình xây dựng ngang đầy.
  1. **Mô tả game**
* Ở game Tetris, các hình khối (Tetromino) sẽ bị thay đổi vị trí cứ sau mỗi game tick hoặc khi người chơi nhấn phím di chuyển hay xoay khối. Như vậy, chúng ta đơn giản chỉ cần kiểm tra xem, với mỗi lần khối Tetromino bị thay đổi vị trí đó, nó có bị đè lên bất cứ phần tử nào khác hay không. Cụ thể, ta phải thực hiện từng bước kiểm tra giữa khối Tetromino với các phần tử:
* Đường biên dưới,
* Đường biên trái,
* Đường biên phải,
* Và với các khối Tetromino đã “hạ cánh”
* Trong game có 7 khối gạch có dạng hình chữ S, hình chữ Z, hình chữ T, hình chữ L, hình đường thẳng, hình chữ L ngược và hình vuông. Mỗi khối gạch được tạo nên bởi 4 hình vuông. Các khối gạch sẽ rơi từ trên xuống. Người chơi phải di chuyển và xoay các khối gạch sao cho chúng vừa khít với nhau, nếu các khối gạch lấp đầy một đường ngang thì người chơi ghi điểm. Trò chơi kết thúc khi các khối gạch chồng lên nhau quá cao.
  1. **Phương hướng**
* Trên cơ sở đã phân tích bài toán, dựa trên kiểu lập trình hướng đối tượng thì ta sẽ bắt tay vào xây dựng những lớp, những thuộc tính, những phương thức tương ứng mỗi khi phân tích vào bài toán.

1. **Xây dựng và viết code chương trình**

Để xây dựng được game vượt chướng ngại vật thì cần xây dựng 3 lớp.

1. **Lớp Tetris**

package Tetris;

import javax.swing.JFrame;

public class Tetris {

public static final int WIDTH = 445, HEIGHT = 629;

private Board board;

private JFrame Window;

public Tetris() {

Window = new JFrame("Tetris");

Window.setSize(WIDTH, HEIGHT);

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

Window.setResizable(false);

Window.setLocationRelativeTo(null);

board = new Board();

Window.add(board);

Window.addKeyListener(board);

Window.setVisible(true);

}

public static void main(String[] args) {

new Tetris();

}

}

Đầu tiên là file Tetris.java, file này là file main, làm các công việc khởi tạo cửa sổ, khởi tạo kích thước, chạy luồng main…

1. **Lớp shape**

package Tetris;

import java.awt.Color;

import java.awt.Graphics;

import static Tetris.Board.BOARD\_WIDTH;

import static Tetris.Board.BOARD\_HEIGHT;

import static Tetris.Board.BLOCK\_SIZE;

public class Shape {

private int x = 4, y = 0;

private int normal = 600;

private int fast = 50;

private int delayTimeForMovement = normal;

private long beginTime;

private int deltaX = 0;

private boolean collision = false;

private int[][] coords;

private Board board;

private Color color;

public Shape(int[][] coords, Board board, Color color) {

this.coords = coords;

this.board = board;

this.color = color;

}

public void setX(int x) {

this.x = x;

}

public void setY(int y) {

this.y = y;

}

public void reset() {

this.x = 4;

this.y = 0;

collision = false;

}

public void update() {

if (collision) {

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

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

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

board.getBoard()[y + row][x + col] = color;

}

}

}

checkLine();

board.setCurrentShape();

return;

}

boolean moveX = true;

if (!(x + deltaX + coords[0].length > 10) && !(x + deltaX < 0)) {

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

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

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

if (board.getBoard()[y + row][x + deltaX + col] != null) {

moveX = false;

}

}

}

}

if (moveX) {

x += deltaX;

}

}

deltaX = 0;

if (System.currentTimeMillis() - beginTime > delayTimeForMovement) {

if (!(y + 1 + coords.length > BOARD\_HEIGHT)) {

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

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

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

if (board.getBoard()[y + 1 + row][x + deltaX + col] != null) {

collision = true;

}

}

}

}

if (!collision) {

y++;

}

} else {

collision = true;

}

beginTime = System.currentTimeMillis();

}

}

private void checkLine() {

int bottomLine = board.getBoard().length - 1;

for (int topLine = board.getBoard().length - 1; topLine > 0; topLine--) {

int count = 0;

for (int col = 0; col < board.getBoard()[0].length; col++) {

if (board.getBoard()[topLine][col] != null) {

count++;

}

board.getBoard()[bottomLine][col] = board.getBoard()[topLine][col];

}

if (count < board.getBoard()[0].length) {

bottomLine--;

}else {

board.addScore();

}

}

}

public void rotateShape() {

int[][] rotatedShape = transposeMatrix(coords);

reverseRows(rotatedShape);

if ((x + rotatedShape[0].length > board.BOARD\_WIDTH) || (y + rotatedShape.length > 20)) {

return;

}

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

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

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

if (board.getBoard()[y + row][x + col] != null) {

return;

}

}

}

}

coords = rotatedShape;

}

private int[][] transposeMatrix(int[][] matrix) {

int[][] temp = new int[matrix[0].length][matrix.length];

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

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

temp[col][row] = matrix[row][col];

}

}

return temp;

}

private void reverseRows(int[][] matrix) {

int middle = matrix.length / 2;

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

int[] temp = matrix[row];

matrix[row] = matrix[matrix.length - row - 1];

matrix[matrix.length - row - 1] = temp;

}

}

public void render(Graphics g) {

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

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

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

g.setColor(color);

g.fillRect(col \* BLOCK\_SIZE + x \* BLOCK\_SIZE, row \* BLOCK\_SIZE + y \* BLOCK\_SIZE, BLOCK\_SIZE,

BLOCK\_SIZE);

}

}

}

}

public int[][] getcoords() {

return coords;

}

public void speedUp() {

delayTimeForMovement = fast;

}

public void speedDown() {

delayTimeForMovement = normal;

}

public void moveRight() {

deltaX = 1;

}

public void moveLeft() {

deltaX = -1;

}

public int getY() {

return y;

}

public int getX() {

return x;

}

}

Tiếp theo là file Shape.java, file này lưu thông tin về các khối gạch.

1. **Lớp Board**

import static Tetris.Board.BLOCK\_SIZE;

import java.awt.Color;

import java.awt.Font;

import java.awt.Graphics;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.KeyEvent;

import java.awt.event.KeyListener;

import java.awt.event.MouseEvent;

import java.util.Random;

import javax.swing.Timer;

import javax.swing.JPanel;

public class Board extends JPanel implements KeyListener {

public static int STATE\_GAME\_PLAY = 0;

public static int STATE\_GAME\_PAUSE = 1;

public static int STATE\_GAME\_OVER = 2;

private int state = STATE\_GAME\_PLAY;

private static int FPS = 60;

private static int delay = FPS / 1000;

private static final long serialVesionUID = 1l;

public static final int BOARD\_WIDTH = 10;

public static final int BOARD\_HEIGHT = 20;

public static final int BLOCK\_SIZE = 30;

private Timer looper;

private Color[][] board = new Color[BOARD\_HEIGHT][BOARD\_WIDTH];

private Random random;

private Color[] colors = { Color.decode("#ed1c24"), Color.decode("#ff7f27"), Color.decode("#fff200"),

Color.decode("#22b14c"), Color.decode("#00a2e8"), Color.decode("#a349a4"), Color.decode("#3f48cc") };

private Shape[] shapes = new Shape[7];

private Shape currentShape;

public int score = 0;

public Board() {

random = new Random();

shapes[0] = new Shape(new int[][] { { 1, 1, 1, 1 } }, this, colors[0]);

shapes[1] = new Shape(new int[][] { { 1, 1, 1 }, { 0, 1, 0 }, }, this, colors[1]);

shapes[2] = new Shape(new int[][] { { 1, 1, 1 }, { 1, 0, 0 }, }, this, colors[2]);

shapes[3] = new Shape(new int[][] { { 1, 1, 1 }, { 0, 0, 1 }, }, this, colors[3]);

shapes[4] = new Shape(new int[][] { { 0, 1, 1 }, { 1, 1, 0 }, }, this, colors[4]);

shapes[5] = new Shape(new int[][] { { 1, 1, 0 }, { 0, 1, 1 }, }, this, colors[5]);

shapes[6] = new Shape(new int[][] { { 1, 1 }, { 1, 1 }, }, this, colors[6]);

currentShape = shapes[0];

looper = new Timer(delay, new ActionListener() {

int n = 0;

@Override

public void actionPerformed(ActionEvent e) {

update();

repaint();

}

});

looper.start();

}

private void update() {

if (state == STATE\_GAME\_PLAY) {

currentShape.update();

}

}

public void setCurrentShape() {

currentShape = shapes[random.nextInt(shapes.length)];

currentShape.reset();

checkOverGame();

}

private void checkOverGame() {

int[][] coords = currentShape.getcoords();

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

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

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

if (board[row + currentShape.getY()][col + currentShape.getX()] != null) {

state = STATE\_GAME\_OVER;

}

}

}

}

}

@Override

protected void paintComponent(Graphics g) {

super.paintComponent(g);

g.setColor(Color.BLACK);

g.fillRect(0, 0, getWidth(), getHeight());

currentShape.render(g);

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

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

if (board[row][col] != null) {

g.setColor(board[row][col]);

g.fillRect(col \* BLOCK\_SIZE, row \* BLOCK\_SIZE, BLOCK\_SIZE, BLOCK\_SIZE);

}

}

}

// draw the shape

g.setColor(Color.WHITE);

for (int row = 0; row < BOARD\_HEIGHT; row++) {

g.drawLine(0, BLOCK\_SIZE \* row, BLOCK\_SIZE \* BOARD\_WIDTH, BLOCK\_SIZE \* row);

}

for (int col = 0; col < BOARD\_WIDTH + 1; col++) {

g.drawLine(col \* BLOCK\_SIZE, 0, col \* BLOCK\_SIZE, BLOCK\_SIZE \* BOARD\_HEIGHT);

}

if (state == STATE\_GAME\_OVER) {

g.setColor(Color.white);

g.drawString("GAME OVER", 50, 200);

}

if (state == STATE\_GAME\_PAUSE) {

g.setColor(Color.white);

g.drawString("GAME PAUSE", 50, 200);

}

g.setColor(Color.WHITE);

g.setFont(new Font("Georgia", Font.BOLD, 15));

g.drawString("SCORE", Tetris.WIDTH - 125, Tetris.HEIGHT / 2);

g.drawString(score + "", Tetris.WIDTH - 125, Tetris.HEIGHT / 2 + 20);

}

public Color[][] getBoard() {

return board;

}

@Override

public void keyTyped(KeyEvent e) {

}

@Override

public void keyPressed(KeyEvent e) {

if (e.getKeyCode() == KeyEvent.VK\_DOWN) {

currentShape.speedUp();

} else if (e.getKeyCode() == KeyEvent.VK\_RIGHT) {

currentShape.moveRight();

} else if (e.getKeyCode() == KeyEvent.VK\_LEFT) {

currentShape.moveLeft();

} else if (e.getKeyCode() == KeyEvent.VK\_UP) {

currentShape.rotateShape();

}

if (state == STATE\_GAME\_OVER) {

if (e.getKeyCode() == KeyEvent.VK\_SPACE) {

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

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

board[row][col] = null;

}

}

setCurrentShape();

state = STATE\_GAME\_PLAY;

}

}

if (e.getKeyCode() == KeyEvent.VK\_SPACE) {

if (state == STATE\_GAME\_PLAY) {

state = STATE\_GAME\_PAUSE;

} else if(state == STATE\_GAME\_PAUSE) {

state = STATE\_GAME\_PLAY;

}

}

}

@Override

public void keyReleased(KeyEvent e) {

if (e.getKeyCode() == KeyEvent.VK\_DOWN) {

currentShape.speedDown();

}

}

public void addScore() {

score ++;

}

}

Cuối cùng chúng ta có file Board.java. File này chịu trách nhiệm điều khiển các khối gạch cũng như xử lý các hoạt động diễn ra trong game.

1. **Tổng kết**
2. **Kết luận**

* Trong thời gian nghiên cứu và vận dụng những gì đã học ở trường và tìm hiểu thêm trên các lĩnh vực khác . Em đã viết được game Tetris (xếp gạch) nhằm mục đích giải trí lành mạnh . Chúng em hi vọng rằng từ những tiền đề này giúp chúng em phát triển hơn trong lĩnh vực này .
* Trong quá trình thực hiện đề tài , chúng em nhận được sự giúp đỡ của Thầy Vũ Huấn giúp chúng em hoàn thành tốt bài báo cáo.
* Vì thời gian làm báo cáo hạn hẹp , kinh nghiệm còn ít nên không tránh khỏi các sai sót , rất mong nhận được sự góp ý của thầy và các bạn.

1. **Tài liệu tham khảo**

* Slide bài giảng và tài liệu học tập môn “ Công Nghệ Java ” do giảng viên bộ môn Thầy Vũ Huấn cung cấp
* https://phocode.com/java/java-2d/java-2d-tro-choi-xep-hinh-tetris/