**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**

---o0o---



**ĐỒ ÁN CUỐI KỲ**

**TETRIS**

**Giảng viên hướng dẫn:** Bùi Tiến Lên

**…**

**…**

**Sinh viên thực hiện:**

Nguyễn Hà Anh – 22127013

Phan Lê Đức Anh –

Nguyễn Lê Yến Nhi –

Lê Nguyễn Yến Vy –

***TP. Hồ Chí Minh, tháng 11 năm 2023.***

MỤC LỤC

[Phân chia công việc 2](#_Toc153526162)

[Giới thiệu chung 3](#_Toc153526163)

[1. Luật chơi 4](#_Toc153526164)

[2. Ý tưởng – xây dựng game. 5](#_Toc153526165)

[a) Cấu trúc bảng 5](#_Toc153526166)

[b) Cấu trúc của 1 viên gạch - 1 khối gạch. 5](#_Toc153526167)

[c) Người chơi 8](#_Toc153526168)

[d) Game 9](#_Toc153526169)

[e) Hoạt động của thuật toán 10](#_Toc153526170)

# Phân chia công việc

|  |  |  |
| --- | --- | --- |
| Nguyễn Hà Anh  (22127013) |  | 100% |
| Phan Lê Đức Anh  (22127028) |  | 100% |
| Nguyễn Lê Yến Nhi  (22127…) |  | 100% |
| Lê Nguyễn Yến Vy  (22127…) |  | 100% |

# Giới thiệu chung

***Tetris***, một trò chơi video puzzle, ra đời vào năm 1985 do kỹ sư phần mềm Liên Xô Alexey Pajitnov sáng tạo. Trò chơi nhanh chóng trở thành một hiện tượng toàn cầu, ảnh hưởng đến nền văn hóa và công nghiệp game. Cái tên Tetris được anh ghép từ chữ "tetra" (một tiếp đầu ngữ chỉ số 4 trong tiếng Hi Lạp) và "tennis" (quần vợt). "Tetra" vì tất cả các mảnh ghép trong game đều gồm 4 đoạn, và "tennis" vì đây là môn thể thao yêu thích của tác giả.

Đặc trưng của Tetris là các khối hình học, được gọi là "Tetriminos," rơi vào một ma trận hình chữ nhật. Người chơi cần di chuyển, xoay, và thả chúng để tạo thành các hàng đầy đủ, khiến chúng biến mất và đem lại điểm số.

Các biến thể và luật chơi khác nhau của Tetris đã tạo ra nhiều phiên bản và sự sáng tạo. Ngay từ khi mới xuất hiện, Tetris đã thu hút cả những người lập trình, như được thấy trong các hướng dẫn viết game Tetris sáng tạo. Với sự phổ biến kéo dài qua hơn 30 năm, Tetris vẫn là một trong những trò chơi kinh điển được thế giới yêu thích.

# Luật chơi

**Mục Tiêu:**

Xếp các khối Tetriminos một cách cẩn thận để tạo ra các hàng đầy đủ ngang. Khi một hàng được tạo ra, nó sẽ biến mất, và người chơi sẽ đạt được điểm số.

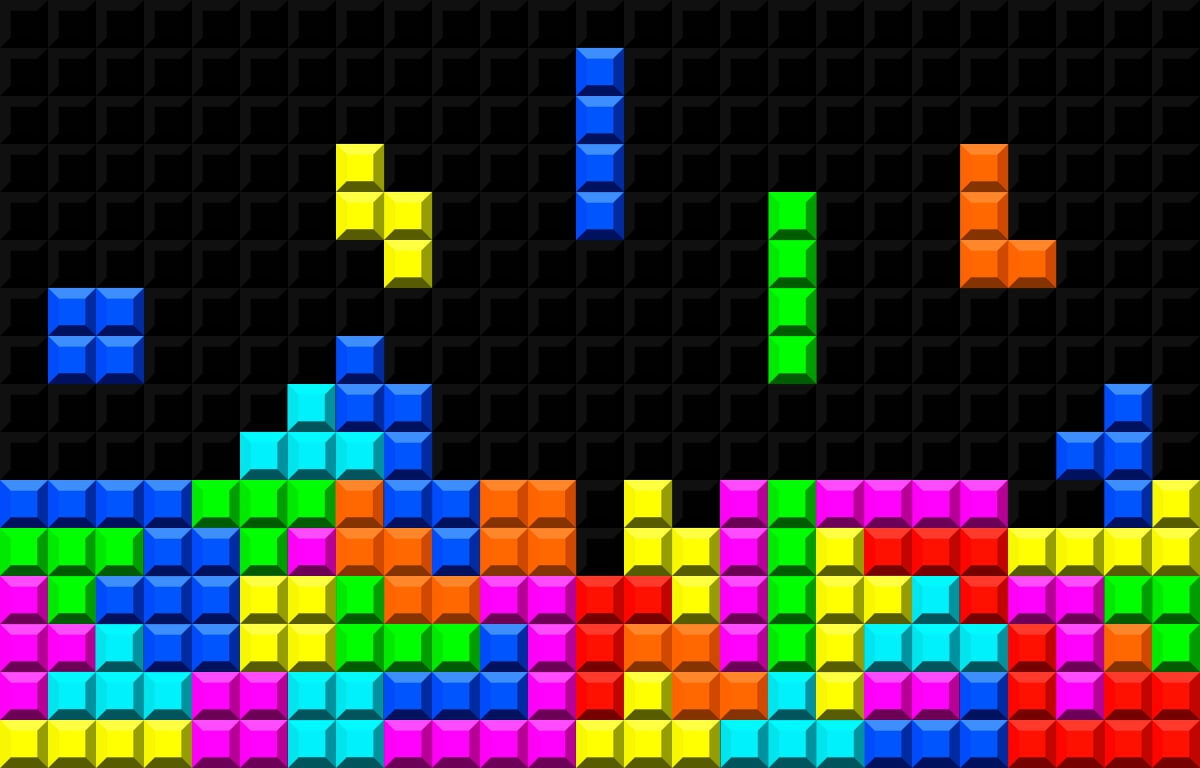
**Di Chuyển và Xoay Khối:**

Sử dụng các phím mũi tên trái, phải, và xuống để di chuyển khối sang trái, sang phải và làm chúng rơi nhanh hơn. Sử dụng phím xoay để xoay khối theo chiều kim đồng hồ.

**Điều kiện thua:**

Nếu khối chạm đến đỉnh màn hình, trò chơi kết thúc. Mục tiêu là chơi càng lâu càng tốt và đạt điểm số cao nhất.

Trong quá trình chơi, tốc độ rơi của khối Tetriminos sẽ tăng lên, làm tăng độ khó của trò chơi.

Hình 1: Game Tetris

# Ý tưởng – xây dựng game.

## Cấu trúc bảng

Nhóm sử dụng một bảng có kích thước 25x15 để thể hiện cho bảng game, song song với đó là các value được gán ở trên bảng:

* Value = 0: ô đó không chứa gạch.
* Value > 0: ô đã chứa gạch.

Một ô gạch chỉ có thể có value > 0 khi viên gạch không thể rớt xuống được nữa, còn lại thì trong quá trình viên gạch rớt, value của các ô sẽ luôn được gán là 0.

Để tạo ra được bảng như mô tả, nhóm đã thiết kế class Board:

class Board {

private:

Brick aBoard[BOARD\_ROW][BOARD\_COL];

public:

void drawBoard(); // vẽ bảng

void displayBoard(); // in bảng

void update(int row); // cập nhật bảng khi có 1+ hàng đầy gạch

void erase();

int getValueBrick(int row, int col); // lấy value của gạch tại vị trí [row][col]

void setValueBrick(int row, int col, int value); // đặt value…

};

Với class như trên, nhóm sử dụng một ma trận 2 chiều aBoard, mang thuộc tính của Brick (xem class Brick dưới đây) để tạo một ma trận 2 chiều lưu các value như đã nói ở trên, cùng với các hàm hỗ trợ vẽ bảng, in bảng, cập nhật bảng, lấy/đặt value của bảng.

## Cấu trúc của 1 viên gạch - 1 khối gạch.

Ở đây nhóm sử dụng khái niệm

* 1 viên gạch = 1 ô vuông gạch
* 4 viên gạch = 1 khối gạch

Với 1 viên gạch, nhóm sử dụng class Brick:

class Brick {

private:

int matrix[BRICK\_ROW][BRICK\_COL];

int value; // = 0 --> không tồn tại gạch

// > 0 --> tồn tại gạch

public:

Brick(); // constructor

int getValue();

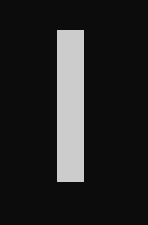
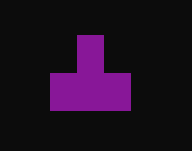
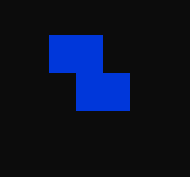
void setValue(int val);

void drawBrick(int x, int y, char c); // vẽ gạch

};

Bằng việc xây dựng class này, nhóm có thể dễ dàng trích xuất từng ô vuông gạch theo toạ độ x, y trên console, và có thể dễ dàng sửa đổi value của ô gạch để có thể sử dụng trên bảng.

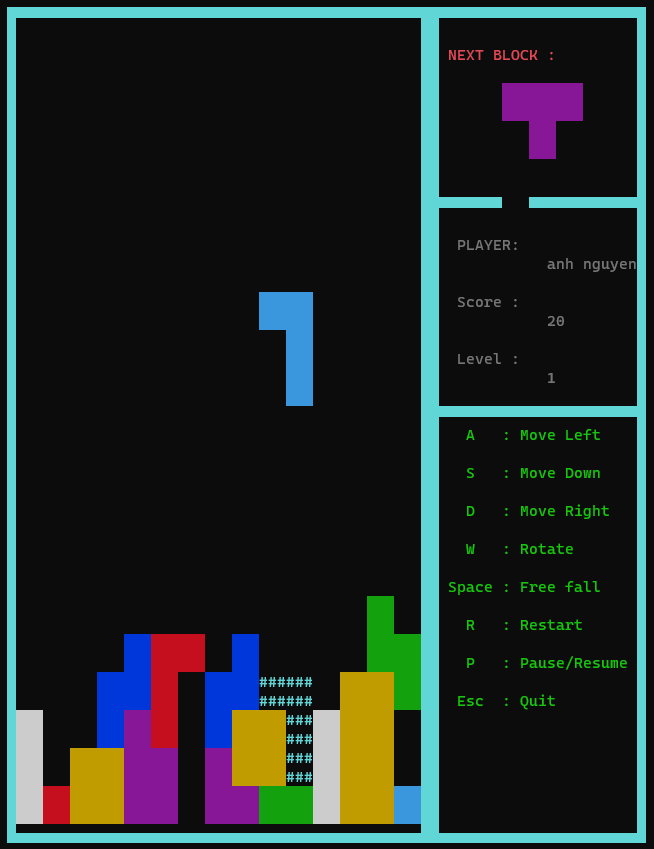
Với 1 khối gạch, nhóm đã thiết kế ra 7 loại gạch, tương ứng theo cách chữ cái I, J, L, O, S, T, Z.



Hình 2: 7 loại khối gạch trong game Tetris

Vì kích thước của 1 ký tự trên console có chiều dài và chiều rộng không bằng nhau, nên nhóm đã quy ước cứ 2 hàng, mỗi hàng có 3 kí tự sẽ bằng 1 ô vuông.

Mỗi khi đặt gạch vào bảng, thì bảng sẽ lưu lại vị trí đang có các viên gạch, đồng thời cũng sẽ đưa ra viên gạch mới để tiếp tục trò chơi. Đồng thời, trên bảng cũng sẽ xuất hiện bóng để hỗ trợ người chơi đặt gạch vào đúng vị trí.



class Gach {

private:

// ……

public:

Gach(); // constructor

~Gach(); // destructor

int getX\_FB();

void setX\_FB(int y);

int getY\_FB();

void setY\_FB(int x);

void random(); // tạo ngẫu nhiên gạch

void drawBlock(); // vẽ gạch

void eraseBlock(); // xoá gạch

void drawNext(Board& aboard); // vẽ gạch tiếp theo

void eraseNext(Board& aboard); // xoá hình vẽ gạch tiếp theo

bool checkInside(int row, int col); // kiểm tra gạch có đang nằm trong bảng không

// kiểm tra các khả năng di chuyển của gạch

bool left(int row, int col, Board& aboard);

bool right(int row, int col, Board& aboard);

bool down(int row, int col, Board& aboard);

bool leftMove(Board& aboard);

bool rightMove(Board& aboard);

bool downMove(Board& aboard);

void fall(Board& aboard);

bool rotate(Board& aboard);

// gán giá trị vào bảng, vẽ bóng, kiểm tra game và người chơi

void assignBoardValue(Board& aboard);

void drawShade(Board& aboard);

int checkGame(Board& aboard, Player& aplayer);

};

Đây là class có nhiều dữ liệu được lưu nhất, vì class này phải đồng thời xử lý những hành vi của những viên gạch, những khối gạch. Đồng thời, nó cũng xử lý như việc tạo khối gạch, vẽ bóng, kiểm tra xem các khối có hợp lệ hay không.

## Người chơi

Nhóm sử dụng class Player để quản lý các thông tin về người chơi, đồng thời đưa ra sự tương tác giữa game và người chơi:

class Player {

private:

string name;

int score;

int level;

public:

Player(); // constructor

int getLevel(); // lấy điểm hiện tại

int getScore(); // lấy level hiện tại

string getName(); // lấy tên người chơi

void setName(string s); // đặt tên người chơi

void updateScore(int s); // update điểm số

void saveInfo(); // bảng ghi thông tin người chơi

void restart(); // reset điểm và level người chơi

void Input(ifstream& FileIn); // mở file lưu highscore

void Output(ofstream& FileOut); // lưu dữ liệu vào file highscore

};

Các tính năng hỗ trợ người chơi:

* Nhập vào thông tin người chơi.
* Lưu thông tin, highscore của người chơi.
* Giúp người chơi restart lại game.

## Game

Để game có thể hoạt động, nhóm sử dụng class Game, là tổng hợp của toàn bộ các class khác, và các function để có thể build game hoàn thiện. Chính vì vậy, đây sẽ là class quan trọng nhất, vì class này sẽ quản lý các hàm hoạt động thông qua đây.

class Game {

protected:

Board aboard;

Player aPlayer;

Gach \*currBlock, \*nextBlock, \*shade;

public:

int menu(); // vẽ menu, đồng thời return các mode ở menu

void inputName(); // vẽ màn hình nhập tên người chơi

void inGameInstruction(); // vẽ phần hướng dẫn chơi trong game

int Play(int mode); // hàm play, sẽ return giá trị res ảnh hưởng đến ending

void End(int res); // in ra màn hình ending của game

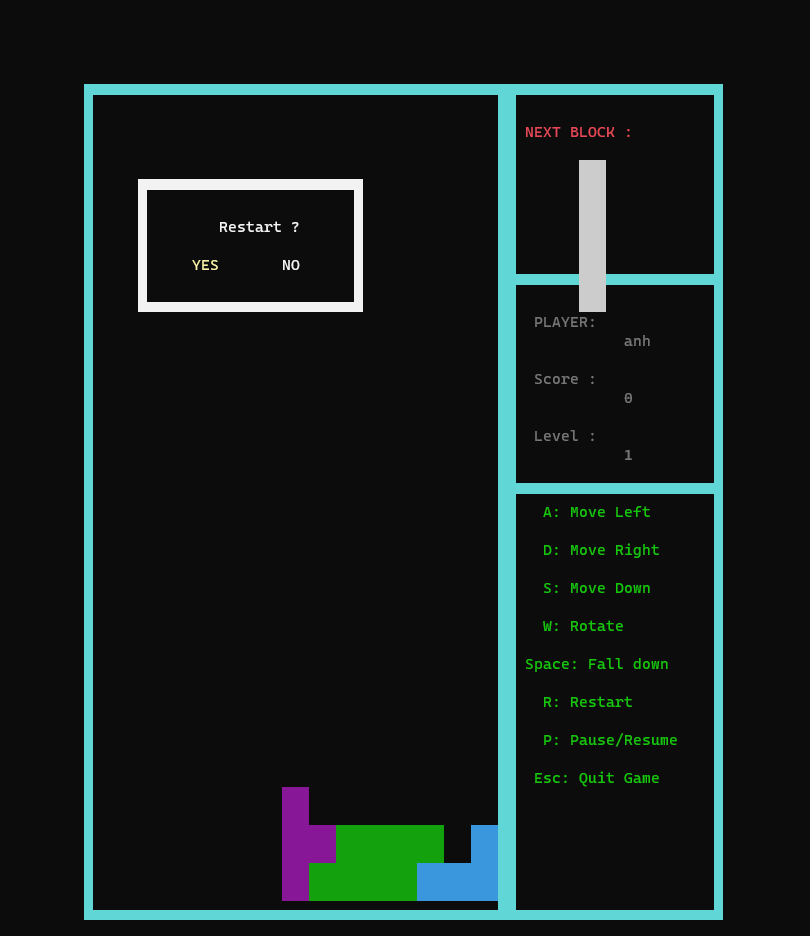
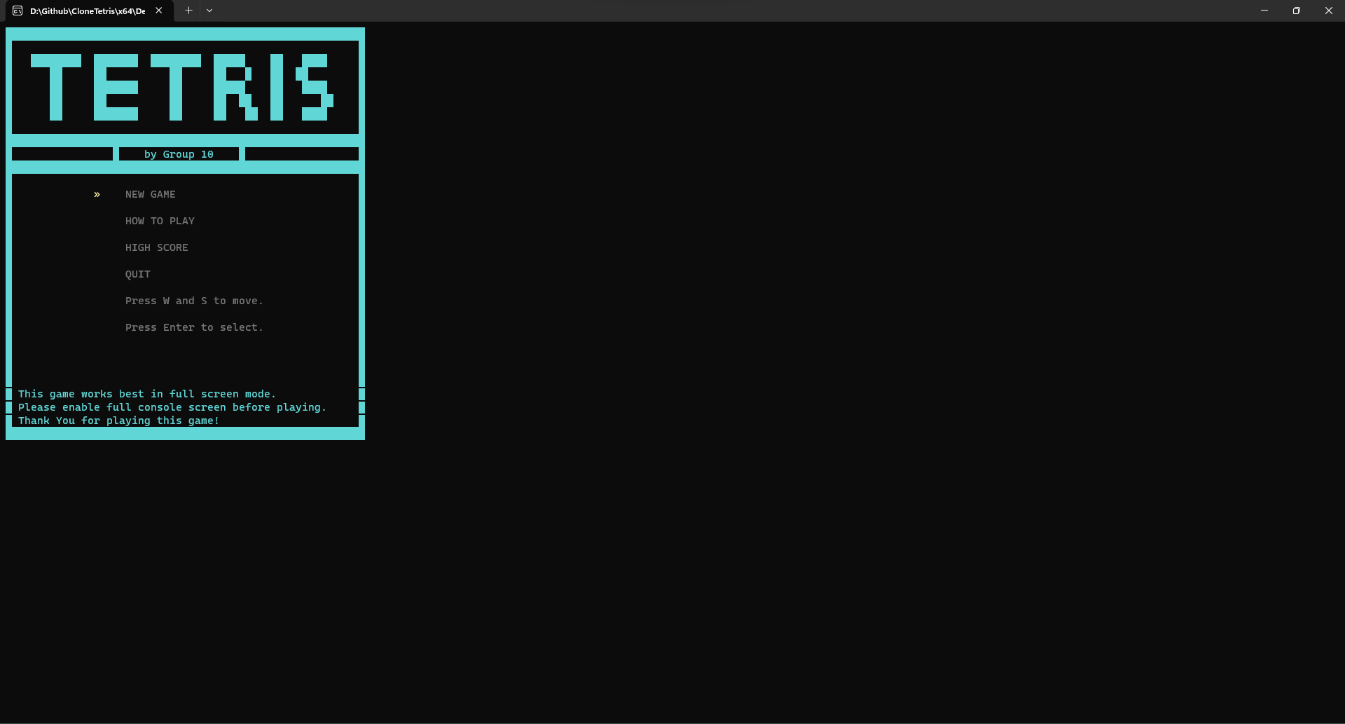
void restartGame(int res); // bắt đầu lại trò chơi

void viewInstruction(); // mode 2, mở hướng dẫn trò chơi

void viewScore(); // xem highscore

int draw\_Y\_N\_board(int x, int y, string c); // vẽ menu chọn Yes/No

void erase\_Y\_N\_board(int x, int y); // xoá menu

};

## Hoạt động của thuật toán

Khi bắt đầu chạy, game sẽ đưa chúng ta đến với lựa chọn mode = x.menu(); với 4 mode tương ứng:

* Mode 1: bắt đầu trò chơi.
* Mode 2: xem hướng dẫn trò chơi.
* Mode 3: xem highscore.
* Mode 4: thoát game.

Khi bắt đầu mode 1:

* Thiết lập Bảng board với toàn bộ giá trị các ô = 0 và điền/xuất thông tin người chơi.
* Tạo 1 khối gạch để bắt đầu chơi.
* Tạo 1 cái bóng với dữ liệu của viên gạch.
* Bắt đầu nhận input từ bàn phím
* Xử lý input.
* Khi gạch chạm đáy, xử lý gạch, bóng, bảng.
* Cập nhật lại các thông số trạng thái gạch, bóng, bảng, điểm, level nếu tăng
* Kiểm tra điều kiện thắng/thua/quit game.
* Nếu thắng 🡪 Hỏi người chơi restart.
* Nếu thua 🡪 Hỏi người chơi restart.
* Sắp xếp thông tin người chơi v.v, lưu lại vào highscore.

# Sơ đồ UML:

# Sơ đồ ERD: