TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI
PHÂN HIỆU TẠI TP. HỒ CHÍ MINH
BỘ MÔN CÔNG NGHỆ THÔNG TIN

BÁO CÁO BÀI TẬP LỚN

ĐỀ TÀI: TRÒ CHƠI TETRIS

Giảng viên hướng dẫn: TRẦN THỊ DUNG

Sinh viên thực hiện: LÊ MINH PHÁT

MAI QUỐC THỊNH

LÊ THIÊN HÒA

THÀNH NGỌC HUY

Nhóm: 16

Lớp: CÔNG NGHÊ THÔNG TIN

Khóa: 62

Tp. Hồ Chí Minh, năm 2022

TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI
PHÂN HIỆU TẠI THÀNH PHỐ HỒ CHÍ MINH

CÔNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập – Tự do – Hạnh phúc

NHIỆM VỤ THIẾT KẾ

BỘ MÔN: CÔNG NGHỆ THÔNG TIN

--------***--------

Họ tên Sinh Viên: Lê Minh Phát.
MSSV: 6251071071

Họ tên Sinh Viên: Lê Thiên Hòa.
MSSV: 6251071030

Họ tên Sinh Viên: Mai Quốc Thịnh.
MSSV: 6251071096

Họ tên Sinh Viên: Thành Ngọc Huy.
MSSV: 6251071037

1. Tên đề tài:
TRÒ CHƠI TETRIS
2. Mục đích, yêu cầu:
a, Mục đích:
- Tạo ra trò chơi tetris nhằm mục đích giải trí.
b, Yêu cầu:
- Dễ dàng sử dụng, giao diện thân thiện.
3. Nội dung và phạm vi đề tài:
- Sơ lược về trò chơi Tetris
- Xây dựng và thiết kế bài code
4. Công nghệ, công cụ và ngôn ngữ lập trình:
- DevC++, ngôn ngữ C/C++
5. Các kết quả chính dự kiến sẽ đạt được và ứng dụng:
- Quyển báo cáo gồm 3 phần
+ Chương 1: Tổng quan về trò chơi Tetris
+ Chương 2: Phân tích và trình bày ý tưởng
+ Chương 3: Thiết kế và sử dụng trò chơi
6. Giáo viên và cán bộ hướng dẫn:
Họ tên: Trần Thị Dung
Đơn vị công tác: Bộ môn Công Nghệ Thông Tin – Trường Đại học Giao thông Vận tải phân hiệu tại TP HCM.
Điện thoại: Email:

Ngày Tháng năm 2022 **Đã giao nhiệm vụ TKTN**
Trưởng BM Công nghệ thông tin **Giáo viên hướng dẫn**
ThS. Trần Phong Nhã

LỜI CẢM ƠN

Lời nói đầu tiên, em xin gửi tới Cô Trần Thị Dung Bộ môn Công nghệ Thông tin Trường Đại học Giao thông vận tải phân hiệu tại thành phố Hồ Chí Minh lời chúc sức khỏe và lòng biết ơn sâu sắc.

Em xin chân thành cảm ơn cô đã giúp đỡ tạo điều kiện để em hoàn thành đồ án với đề tài "Thiết kế trò chơi tetris", em xin cảm ơn cô đã nhiệt tình giúp đỡ, hướng dẫn cho em kiến thức, định hướng và kỹ năng để có thể hoàn thành đồ án tốt nghiệp này.

Tuy đã cố gắng trong quá trình nghiên cứu tìm hiểu tuy nhiên do kiến thức còn hạn chế nên vẫn còn tồn tại nhiều thiếu sót. Vì vậy em rất mong nhận được sự đóng góp ý kiến của cô và các bạn để đề tài của em có thể hoàn thiện hơn.

Lời sau cùng, em xin gửi lời chúc tới cô có thật nhiều sức khỏe, có nhiều thành công trong công việc.

Em xin chân thành cảm ơn.

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Tp. Hồ Chí Minh, Ngày…Tháng…. năm

Giáo viên hướng dẫn

MỤC LỤC

Contents

DANH MỤC CHỮ VIẾT TẮT

STT	Mô tả	Ý nghĩa
1	Tetriminos	Các khối gạch
2	Seed	là một giá trị nguyên, được sử dụng như là seed bởi giải thuật sinh số ngẫu nhiên.
3	Bool	là kiểu dữ liệu chỉ nhận một trong hai giá trị true (đúng) hoặc false (sai)
4	Console	Giao diện cơ bản
5	Clone	Bản sao
6	Play Field, Game Board	Khung trò chơi

Chương 1: TỔNG QUAN VỀ TRÒ CHƠI TETRIS

1, Giới thiệu về trò chơi Tetris:

Tetris có nguồn gốc xuất xứ từ Liên Xô, một nhà khoa học máy tính người Liên Xô đã tạo ra trò chơi này trong khi ông đang làm việc, thời gian vào ngày 6 tháng 6 năm 1984.

Cái tên Tetris được đặt theo tiền tố là "Tetra" của tiếng Hy Lạp có ý nghĩa là "bốn" tức là trò chơi có 4 phần. Với sức hấp dẫn của mình, Tetris đã được bán cho hầu hết các máy trò chơi điện tử cùng các hệ điều hành khác nhau của máy tính. Nó thực sự có sức hấp dẫn lớn tới toàn thị trường thế giới.

Nếu như bạn đã từng chơi trò xếp hình có ở các máy chơi điện tử cầm tay thì có lẽ các hình khối chữ L, O, J, S, T, Z hay là chữ l thẳng đứng chẳng còn xa lạ nữa. Trong số 7 khối hình này, chúng đều có một đặc điểm chung đó là được hình thành và cấu tạo từ những ô vuông nhỏ.

Khi các khối gạch chưa rơi xuống, người chơi có thể xoay đổi kiểu dáng từ đứng thành nằm hoặc ngược lại để phù hợp với những chỗ trống mình cần lấp đầy.

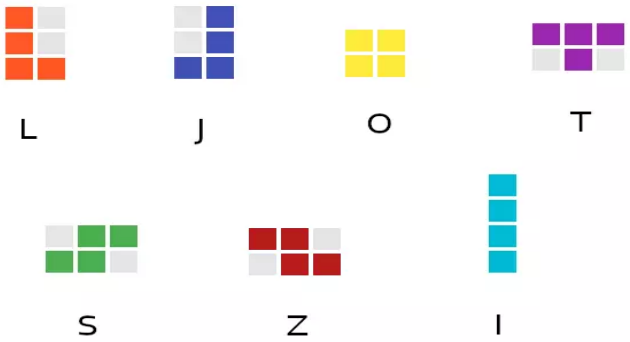
Các khối gạch này sẽ được rơi một cách ngẫu nhiên, đương nhiên bạn sẽ không thể chọn đâu là khối tiếp theo để thuận tiện cho việc xếp vào chỗ trống. Vì vậy chỉ còn cách là bạn tận dụng tối đa các hình khối đã có và lắp ghép chúng với nhau để tạo nên một thế trận ổn nhất, tiết kiệm diện tích và lấp đầy chỗ trống nhất.

2, Chi tiết về trò chơi Tetris:

Trò chơi có bảy loại khối hình: I (thẳng đứng), J, L, O (vuông), S, T, Z. Ta thấy mỗi khối gạch được cấu tạo từ 4 hình vuông nhỏ xếp lại với nhau. Ta có thể coi các khối gạch đó như là những hình chữ nhật có kích thước khác nhau.

Các hình khác được tạo ra khi xoay các khối cơ bản này các góc tương ứng 90 độ, 180 độ, 270 độ.

Một chuỗi ngẫu nhiên của Tetriminos rơi xuống sân chơi (một trục đứng hình chữ nhật, được gọi là "tốt" hay "ma trận").



2.1, Cách chơi:

Mục tiêu của trò chơi là di chuyển các khối gạch đang rơi từ từ xuống trong kích thước hình chữ nhật 20 hàng x 10 cột (trên màn hình). Chỗ nào có gạch rồi thì không di chuyển được tới vị trí đó. Người chơi xếp những khối hình sao cho khối hình lấp đầy 1 hàng ngang để ghi điểm và hàng ngang ấy sẽ biến mất.

Tại Tetris sẽ có 2 bảng khác nhau:

Thứ nhất, 1 bảng chính gồm 24 dòng và 10 cột, người chơi sẽ chỉ thể hiện ra màn hình 20 dòng còn 4 dòng còn lại thì không. 4 dòng này chính người chơi cũng sẽ không nhìn thấy bởi nó được dùng để tạo ra 1 khối gạch mới rơi xuống và bạn sẽ lấy chúng để lấp đầy vào chỗ trống.

Bảng thứ hai, được gọi là bảng "Next": Bảng này thể hiện những khối gạch tiếp theo sẽ được đưa vào trò chơi sau khi khối gạch hiện tại đã được xếp xong.

Lần lượt từng nhóm 4 khối sẽ rơi từ phía trên cùng của màn hình, chúng được xoay và di chuyển cho tới khi được rơi xuống màn hình và tiếp tục 4 khối tiếp theo sẽ rơi xuống, trình tự cứ vậy mà diễn ra cho tới khi người chơi thua cuộc.

Luật chơi Tetris quy định rằng nếu người chơi để cho khối hình rơi xuống cao hơn so với chiều cao của màn hình xếp gạch thì đồng nghĩa với việc trò chơi sẽ kết thúc, đương nhiên bạn sẽ là người thua cuộc.

Trò chơi kết thúc cũng là lúc các khối gạch không còn rơi xuống nữa. Khi chơi bạn cần lưu ý 1 điều đó là bạn chỉ có thể xóa được tối đa 4 hàng/lần thôi nhé.

Hãy nhớ tới các phím tắt sau đây để trận đấu của bạn diễn ra ổn định nhất:

- Phím mũi tên lên có nghĩa là bạn được phép xoay khối
- Phím mũi tên trái có nghĩa là bạn được di chuyển sang trái
- Phím mũi tên phải là được di chuyển sang phải
- Phím mũi tên chỉ xuống dưới có nghĩa là bạn có thể tăng tốc độ chơi của mình nhanh hơn.

2.2, Những điều thú vị về trò chơi Tetris:

Các hình khối xuất hiện trong Tetris nhìn có vẻ khá đơn giản thế nhưng theo thí nghiệm của các nhà khoa học thì cho rằng chơi Tetris trong thời gian dài có thể làm tăng độ dày của vỏ não lên tới 0.5mm. Vỏ não chính là nơi làm nhiệm vụ đảm bảo trí nhớ và phát triển nó cho bộ não của chúng ta, bằng những kỹ năng nghe, nhìn, suy nghĩ thường xuyên sẽ giúp bạn có được trí nhớ tuyệt vời. Vậy phải cảm ơn Tetris đã cho bạn 1 sân chơi tuyệt vời rồi.

Khi chơi game Tetris, bởi vì đây là trò game thuộc vào game trí tuệ, chính bởi vậy mà các bạn sẽ cần phải vận dụng tư duy trong suốt quá trình chơi, đồng thời tư duy kết hợp với thap tác của tay và khả năng quan sát của mắt. Bạn cần phải luyện thao tác tay thật nhanh nhẹn để di chuyển hình khối đang rơi xuống một cách nhanh nhất, bạn cần phải có tư duy cao để suy nghĩ trong thời gian rất ngắn về việc sắp xếp hình khối đó ở đâu cho phù hợp.

Bạn cần phải xếp những hình khối hoàn chỉnh lấp đầy hàng ngang, với mỗi lượt hàng ngang được lấp đầy thì bạn sẽ được ghi điểm. Số điểm sẽ được tính cụ thể trên hệ thống. Nếu như các bạn liên tục làm tan các hàng ngay trong màn hình thì sẽ nhanh chóng được lên các levels cao hơn.

Bạn không nên chỉ chú trọng vào một lối di chuyển xuống của hình khối, bạn cần phải kết hợp giữa các phím điều khiển sang trái, sang phải để đưa hình khối vào khu vực thích hợp. Hình khối cần được xếp một cách logic để không gặp khó khăn hay rối bời trong quá trình chơi game.

Nếu bạn xếp hình khối của mình quá cao trong khi đó không có hàng ngang nào được lấp đầy thì bạn sẽ bị thua, do đó bạn không nên xếp các khối hình cao chạm đỉnh khiến màn chơi của bạn bị thua nhé.

Chương 2: PHÂN TÍCH VÀ NÊU RA Ý TƯỞNG TRÒ CHƠI TETRIS

Game Board

Hay còn được gọi là Play Field, Game Board thường là một ma trận 20 hàng và 10 cột. Đây là nơi chứa các khối gạch rơi xuống.

Tetromino

Là những khối hình thù quái dị từ trên trời rơi xuống. Có 7 loại tất cả: khối chữ L, J, O, T, S, Z, và I. Mỗi loại khối có màu sắc tương ứng khác nhau.

Các khối này đều có thể bị xoay (theo chiều kim đồng hồ) cũng như di chuyển (sang trái hoặc phải), đương nhiên là nếu không có vật cản.

Game Tick

Là khoảng thời gian để khối tetromino rơi xuống thêm một ô.

Sau khi khối hiện tại rơi xuống tận cùng, chạm đáy hoặc các khối đã hạ cánh, nó sẽ bị gắn lại và một khối khác sẽ rơi xuống từ chính giữa của cạnh trên Game Board.

Ăn điểm

Khi một hàng bị lấp đầy, bạn sẽ được ăn điểm. Số hàng hoàn thành cùng lúc càng nhiều (tối đa 4 hàng), số điểm tăng thêm càng nhiều. Đồng thời, các hàng đã được lấp đầy sẽ biến mất, làm các khối ô bên trên rơi xuống.

Game Over

Trò chơi kết thúc khi lượng khối đã rơi xuống chồng chất lên đến mức chạm vào cạnh trên cùng của Game Board, và khối mới không thể rơi xuống được nữa.

Chương 3: THIẾT KẾ VÀ SỬ DỤNG TRÒ CHƠI TETRIS

1, Giới thiệu các thư viện, hàm và lệnh mới:

1.1, Thư viện window.h:

Windows.h là một tệp tiêu đề cụ thể của Window cho ngôn ngữ lập trình C và C++ chứa các khai báo cho tất cả các chức năng trong Window API, tất cả các macro phổ biến được các lập trình viên Windows sử dụng và tất cả các kiểu dữ liệu được sử dụng bởi các chức năng khác nhau và hệ thống con.

1.2, Thư viện stdlib.h:

Thư viện dùng khi cấp phát bộ nhớ động hoặc dùng mấy hàm xử lý chuyển đổi số sang xâu.

1.3, Lệnh define:

Là tiền sử lý trong ngôn ngữ C/C++ cho phép bạn đặt tên cho một hằng số nguyên hay hằng số thực. Trước khi biên dịch, trình biên dịch sẽ thay thế những tên hằng bạn đang sử dụng bằng chính giá trị của chúng. Quá trình thay thế này được gọi là quá trình tiền biên dịch.

Cú pháp: **#define name value**

1.4, Hàm rand và srand:

- Hàm rand trong C giúp chúng ta tạo ra một số ngẫu nhiên trong phạm vi từ 0 đến RAND_MAX.

Cú pháp: **rand();**

- Hàm srand cung cấp seed cho bộ sinh số ngẫu nhiên được sử dụng bởi hàm **rand**.

Cú pháp: **void srand (unsigned int seed).**

- Để mỗi seed khác nhau người ta thường dùng với unsigned int time(NULL) trong thư viện **time.h**, hàm **time(NULL)** trả về số giấy đã trôi qua kể từ ngày 1/1/1970.

- Hàm **srand()** thường được gọi trước khi gọi hàm **rand().**

2, Thực hiện Code:

2.1, Tạo struct Game:

```
typedef struct
{
char board[TETRIS_BOARD_HEIGHT][TETRIS_BOARD_WIDTH];
unsigned int score, timer;
} Tetris;
```

- Char board[TETRIS_BOARD_HEIGHT][TETRIS_BOARD_WIDTH] : ma trận 2 chiều Game Board.
- Biến score để lưu điểm của người chơi. Biến timer (đơn vị microseconds), có ý nghĩa là sau một khoảng thời gian timer thì khối Tetromino sẽ tự rơi xuống một bậc.

2.2, Tạo các thành phần chính của trò chơi:

2.2.a, Hàm di chuyển con trỏ đến tọa độ cho trước:

- Để tự do di chuyển trên màn hình Console ta cần di chuyển con trỏ chuột đến bất kì vị trí nào trên màn hình Console.

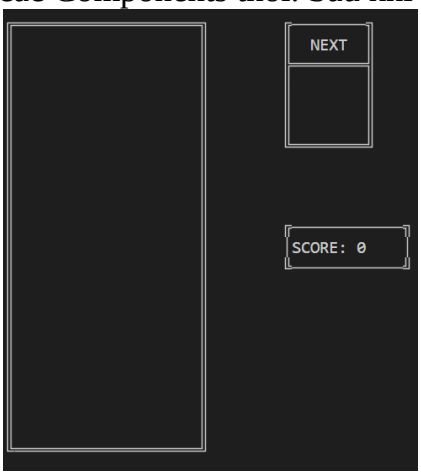
```
void MoveCursorToXY(unsigned short x, unsigned short y)
{
    COORD coord = (COORD){x, y};
    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), coord);
}
```

- Hàm này sẽ di chuyển con trỏ chuột tới vị trí `x`, `y`. Với góc trên bên trái là góc tọa độ. Lưu ý khi `MoveCursorToXY(x,y)` đến một tọa độ nào đó, nếu tại đó đang có chữ thì sẽ bị ghi đè lên. Vì không có tọa độ âm trong màn hình Console nên nếu truyền `x`, `y` âm thì có nghĩa là `MoveCursorToXY(0,0)`.
- Một lưu ý nữa là muốn sử dụng được hàm trên thì chúng ta phải `#include <windows.h>`.

2.2.b, Game Board, Next Shape, Socre Box:

- Đầu tiên ta cần có vài hằng để lưu lại tọa độ của các thành phần như Game Board, Next Shape, Score Box:
  - `BOARD_COORD_X`, `BOARD_COORD_Y`: tọa độ của Game Board.
  - `NEXT_COORD_X`, `NEXT_COORD_Y`: tọa độ của Next Shape.
  - `SCORE_COORD_X`, `SCORE_COORD_Y`: tọa độ của Score Box.
- Đối với Game Board ta cần chiều dài và chiều rộng (thường là 20x10). Ta dùng 2 hằng `TETRIS_BOARD_WIDTH` và `TETRIS_BOARD_HEIGHT` để lưu 2 giá trị giá trị này.

❖ Giờ chúng ta chỉ việc di chuyển tới vị trí cần vẽ, và sử dụng các ký tự trong bảng mã ASCII để vẽ các Components thôi. Sau khi vẽ ta được thế này:



2.3, Vấn đề về các khối Tetromino:

2.3.a, Biểu diễn các khối Tetromino:

❖ Các khối Tetromino được biểu diễn dưới dạng mảng Bool hai chiều với 0 là khoảng trống (empty), 1 là khối (block) như sau:

- Khối chữ I:
```
0 0 0 0
1 1 1 1
0 0 0 0
0 0 0 0
```
- Khối chữ O:
```
1 1
1 1
```
- Khối chữ L:
```
0 0 1
1 1 1
0 0 0
```
- Khối chữ J:
```
1 0 0
1 1 1
0 0 0
```
- Khối chữ Z:
```
1 1 0
0 1 1
0 0 0
```
- Khối chữ S:
```
0 1 1
1 1 0
0 0 0
```
- Khối chữ T:
```
0 1 0
1 1 1
0 0 0
```

❖ Tạo struct của Tetromino:

```
typedef struct
{
char **shape_matrix;
unsigned short width;
short x, y;
} Shape;
```

❖ Một khối Tetromino sẽ chứa những thông tin gì:

➢ Đầu tiên là hình dạng được biểu diễn dưới ma trận Bool 2 chiều. Ở đây ta sẽ dùng con trỏ đa cấp (cấp 2) để sử dụng như mảng hai chiều. Việc phải sử dụng con trỏ cấp hai mà không dùng hẳn mảng 2 chiều là vì các hình dạng của Tetris được biểu diễn như trên là các mảng có kích thước khác nhau, mà mảng 2 chiều khi khai báo bắt buộc phải có kích thước giống nhau mới khai báo được. Một bên là kích thước linh hoạt một bên là kích thước cứng, cho nên việc set một biến chứa các hình dạng của các Tetrimino là không thể với mảng hai chiều thông thường. Thế nên phải dùng con trỏ cấp 2 như một mảng hai chiều, khi khai báo ta chỉ việc khai báo như thế này:

```
char shape_matrix = (char *[]){(char[]){0, 0, 0, 0},
                               (char[]){1, 1, 1, 1},
                               (char[]){0, 0, 0, 0},
                               (char[]){0, 0, 0, 0}}
```

➢ Tiếp theo là độ rộng của mảng hay kích thước của mảng, do hình vuông nên ta dùng một biến `width` để lưu thông số này.

➢ Cuối cùng là tọa độ của khối Tetromino tức tọa độ của nó trong Game Board. Được lưu vào hai biến `x`, `y`.

2.3.b, Ánh xạ các khối Tetromino lên Game Board:

❖ Game Board cũng là ma trận Bool 2 chiều tương tự các Tetromino với kích thước 20x10 (được nêu ở trên).

❖ Khi ánh xạ các khối Tetromino lên Game Board sẽ như hình sau:

0	0	0	0	1	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	0	1	1	0	0	0	0

❖ Để mảng biểu diễn các khối Tetromino hiển thị trên mảng Game Board, ta cần một hàm để ghi Tetromino vào Game Board.

```
void WriteShapeToBoard()
{
    unsigned short i, j;
    for (i = 0; i < current->width; i++)
        for (j = 0; j < current->width; j++)
            if (current->shape_matrix[i][j])
                tetris.board[current->y + i][current->x + j] =1;
}
```

❖ Và cũng cần một hàm để xóa Tetromino ra khỏi Game Board ngược lại với hàm trên thôi.

```
void DeleteShapeFromBoard()
{
    unsigned short i, j;
    for (i = 0; i < current->width; i++)
        for (j = 0; j < current->width; j++)
            if (current->shape_matrix[i][j])
                tetris.board[current->y + i][current->x + j] = 0;
}
```

❖ Hàm in ra màn hình mảng Game Board:

```
void PrintShapeToConsole()
{
    unsigned short i, j;
    for (i = 0; i < TETRIS_BOARD_HEIGHT; i++)
        for (j = 0; j < TETRIS_BOARD_WIDTH; j++)
            if (tetris.board[i][j])
            {
                MoveCursorToXY(BOARD_COORD_X + 1 + 2 * j, BOARD_COORD_Y + 1 + i);
                printf("%c", 254);
            }
}
```

❖ Hàm xóa Shape khỏi màn hình Console:

```
void EraseShapeFromConsole()
{
    unsigned short i, j;
    for (i = 0; i < current->width; i++)
        for (j = 0; j < current->width; j++)
            if (current->shape_matrix[i][j])
            {
                MoveCursorToXY(BOARD_COORD_X + 1 + 2 * (j + current->x),
BOARD_COORD_Y + 1 + i + current->y);
                printf(" ");
            }
}
```

2.3.c, Lấy Random 1 trong 7 Shape:

❖ Hàm lấy ngẫu nhiên Shape:

```
void GetNewShape()
{
    if (current == NULL)
    {
        current = CopyShape(Tetrominos[rand() % 7]);
        next = CopyShape(Tetrominos[rand() % 7]);
    }
    else
    {
        current = next;
        next = CopyShape(Tetrominos[rand() % 7]);
    }
    current->x = (TETRIS_BOARD_WIDTH - current->width + 1) / 2;
    if (!CheckPosition(current))
game_on_flag = FALSE;
}
```

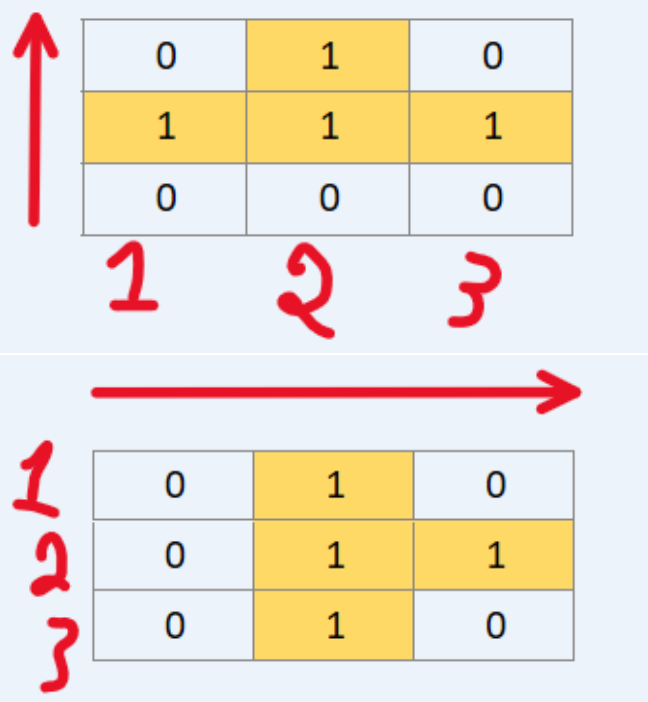
- Lúc đầu ta lấy hình dạng ngẫu nhiên cho biến `next` và `current` cùng lúc, sau đó ta chỉ lấy cho biến `next` còn `current` ta gán bằng `next` (cũ).

```
if (!CheckPosition(current))
        game_on_flag = FALSE;
```

- Đây là kiểm tra xem khối Tetromino đã chạm thành trên của Game Board chưa. Nếu có thì Game Over tức `game_on_flag = FALSE`.

2.3.d, Xoay các khối Tetromino (theo chiều kim đồng hồ):

- Để xoay một ma trận vuông 90° theo chiều kim đồng hồ, ta đổi hàng này thành cột kia như hình sau:



❖ Ta có hàm xoay như sau:

```
void RotateShape(Shape *shape)
{
    Shape *temp = CopyShape(*shape);
    unsigned short i, j;
    for (i = 0; i < shape->width; i++)
        for (j = 0; j < shape->width; j++)
            shape->shape_matrix[i][j] = temp->shape_matrix[shape->width - j -
1][i];
    DeleteShape(temp);
}
```

❖ Shape *temp = CopyShape(*shape); copy lại khối tetromino vào biến `temp`, DeleteShape(temp); xóa và giải phóng bộ nhớ của biến `temp` vừa sao chép.

2.4, Các hàm xử lý:

2.4.a, Hàm kiểm tra vị trí của khối Tetromino:

❖ Hàm này sẽ kiểm tra vị trí hiện tại của khối Tetromino thông qua các dữ liệu trong biến Struct Shape mà ta đã khai báo ở trên.

❖ Nó sẽ kiểm tra 2 trường hợp:

- Thứ nhất là vị trí hiện tại của khối Tetromino so với các thành của Game Board mà cụ thể là thành trái, phải và dưới. Nếu vị trí của khối Tetromino ra khỏi các thành này thì hàm sẽ trả về giá trị FALSE.
- Thứ hai là vị trí của khối Tetromino so với các khối đã rơi xuống dưới. Nếu một trong những phần của khối Tetromino trùng với một phần của các khối đã rơi xuống, thì hàm sẽ trả về giá trị FALSE. Tức là nếu một trong những nếu một phần tử (giá trị bằng 1) thuộc mảng 2 chiều bool của khối Tetromino trùng với một phần tử (giá trị bằng 1) của mảng 2 chiều bool của Game Board thì sẽ trả về FALSE.

❖ Sau đây là code:

```
unsigned short CheckPosition(Shape *shape)
{
    unsigned short i, j;
    for (i = 0; i < shape->width; i++)
        for (j = 0; j < shape->width; j++)
            if (shape->shape_matrix[i][j])
            {
                if (shape->x + j < 0 || shape->x + j >= TETRIS_BOARD_WIDTH ||
shape->y + i >= TETRIS_BOARD_HEIGHT) // nếu ra khỏi các thành của Game Board
                    return FALSE;
                else if (tetris.board[shape->y + i][shape->x + j]) // nếu trùng
với các khối khác
                    return FALSE;
```

```
        }
    return TRUE;
}
```

2.4.b, Hàm kiểm tra hàng có full chưa và tính điểm:

- Hàm này sẽ kiểm tra những hàng đã full (được lấp đầy) trong quá trình chơi, để tính điểm. Và sẽ xóa những hàng đó đi đi chuyển những hàng phía trên những hàng đó xuống dưới. Nó đồng thời cũng cập nhật điểm lên ô Score và giảm thời gian rơi của những khối Tetromino sau này đi (để tăng độ khó khi càng về sau).
- Đây là code:

```
void CheckRows()
{
    unsigned short i, j, counter = 0; // bien counter se dem so hang full (de
tinh diem sau khi check xong)
    for (i = 0; i < TETRIS_BOARD_HEIGHT; i++)
    {
        unsigned short sum = 0;
        // tinh tong cac phan tu cua hang thu i
        for (j = 0; j < TETRIS_BOARD_WIDTH; j++)
            sum += tetris.board[i][j];
        if (sum == TETRIS_BOARD_WIDTH) // neu tong bang chieu rong cua Game Board
=> hang full
        {
            counter++;
            // xoa hang full bang " "
            for (j = 0; j < TETRIS_BOARD_WIDTH; j++)
            {
                MoveCursorToXY(BOARD_COORD_X + 1 + 2 * j, BOARD_COORD_Y + 1 + i);
                printf(" ");
            }
            // di chuyen cac khoi phia tren hang full xuong duoi
            unsigned short k;
            for (k = i; k >= 1; k--)
                for (j = 0; j < TETRIS_BOARD_WIDTH; j++)
                {
                    // xoa cac chi tiet thua sau khi di chuyen xuong
                    if (!tetris.board[k - 1][j] && tetris.board[k][j])
                    {
                        MoveCursorToXY(BOARD_COORD_X + 1 + 2 * j, BOARD_COORD_Y +
1 + k);
                        printf(" ");
                    }
                    // di chuyen xuong (cac phan tu cua mang)
                    tetris.board[k][j] = tetris.board[k - 1][j];
                }
        }
    }
```

```
    tetris.timer -= 1000;                                   // giam thoi gian game
tick, lam khoi tetromino roi nhanh hon
    tetris.score += 100 * counter;                          // cong diem (voi 1
hang la 100 diem)
    MoveCursorToXY(SCORE_COORD_X + 8, SCORE_COORD_Y + 1); // update diem vao box
score
    printf("%u", tetris.score);
}
```

2.4.c, Hàm điều khiển và xử lý thông qua các phím bấm:

- ❖ Để chơi được game thì ta phải điều khiển nó (điều này hiển nhiên), nên ta cần 1 hàm để điều khiển game này. Mà cụ thể là di chuyển và xoay các khối Tetromino.
- ❖ Đây là code:

```
void ControlCurrentShape(char key)
{
    Shape *temp = CopyShape(*current); // sao chep ra 1 bien tam thoi de kiem tra
truoc vi tri
    switch (key)
    {
    case LEFT_KEY: // neu ban phim mui ten trai, khoi tetromino di chuyen sang
trai
        EraseShapeFromConsole();
        temp->x--;
        if (CheckPosition(temp)) // neu khoi tetromino (tam thoi) ko bi can tro
boi (cac thanh Game Board hay cac khoi khac)
            current->x--;        // di chuyen khoi tetromino hien tai sang trai 1
don vi
        break;
    case RIGHT_KEY: // neu ban phim mui ten phai, khoi tetromino di chuyen sang
phai
        EraseShapeFromConsole();
        temp->x++;
        if (CheckPosition(temp))
            current->x++; // di chuyen khoi tetromino hien tai sang phai 1 don vi
        break;
    case DOWN_KEY: // neu bam phim mui ten xuong, khoi tetromino roi xuong duoi 1
bac
        EraseShapeFromConsole();
        temp->y++;
        if (CheckPosition(temp)) // neu khoi tetromino chua cham day
            current->y++;        // di chuyen khoi tetromino xuong duoi 1 don vi
        else                     // neu khoi tetromino da cham day
        {
            WriteShapeToBoard();       // cap nhat toa do cua tetromino vao mang
Game Board
            CheckRows();               // kiem tra, tinh diem
            GetNewShape();             // tao khoi tetromino moi
            PrintNextShapeToConsole(); // hien thi khoi tetromino tiep theo
        }
```

```
        break;
    case ROTATE_KEY:              // neu bam phim mui ten len, xoay khoi
tetromino
        EraseShapeFromConsole();  // xoa hinh anh cua khoi tetromino hien tai
khoi man hinh Console
        RotateShape(temp);        // xoay khoi tetromino (tam thoi)
        if (CheckPosition(temp))  // kiem tra khoi tetromino sau khi xoay
            RotateShape(current); // xoay khoi tetromino hien tai
        break;
    case PAUSE_KEY: // tam dung game
        do
        {
            key = getch();
            if (key == ESCAPE_KEY) // neu nham Esc thi thoat chuong trinh
            {
                MoveCursorToXY(0, BOARD_COORD_Y + 1 + TETRIS_BOARD_HEIGHT);
                exit(0);
            }
        } while (key != PAUSE_KEY); // neu nham 'p' mot lan nua thi quay tro lai
game
        break;
    case ESCAPE_KEY: // thoat game
        MoveCursorToXY(0, BOARD_COORD_Y + 1 + TETRIS_BOARD_HEIGHT);
        exit(0);
    }
    DeleteShape(temp);      // xoa va giai phong bo nho cho bien tam thoi
    WriteShapeToBoard();    // cap nhat toa do cua khoi tetromino vao mang Game
Board
    PrintShapeToConsole();  // in mang Game Board ra man hinh Console
    DeleteShapeFromBoard(); // xoa toa do cua khoi teromino ra khoi mang Game
Board
}
```

2.4.d, Hàm update frame:

- ❖ Sau khi đã có tất cả các hàm hiện thị, điều khiển, xử lý, ... Chúng ta cần một hàm để cập nhật khung hình liên tục lên màn hình Console.
- ❖ Đây là code:

```
void UpdateFrame()
{
    struct timeval before, after;
    gettimeofday(&before, NULL);
    do
    {
        if (kbhit()) // neu nguoi dung bam bat ky phim nao
        {
            ControlCurrentShape(getch()); // FPS tang dan, sau khi 1 hang full
        }
        gettimeofday(&after, NULL);
```

```
        if ((unsigned int)(after.tv_sec * 1000000 + after.tv_usec - before.tv_sec
* 1000000 - before.tv_usec) > tetris.timer) // neu khoang cach giua before và
after lon hon tetris.timer ?
        {
            before = after;
            ControlCurrentShape(DOWN_KEY); // khoi tetromino roi xuong 1 don vi
        }
    } while (game_on_flag);
    system("cls");
    printf("Game over!" // game over
           "\nYour Score: %u"
           "\n(Press C to continue...)",
           tetris.score);
    char key;
    do
    {
        key = getch();
        if (key == ESCAPE_KEY) // neu bam Esc, thoat chuong trinh
            exit(0);
    } while (key != 'c' && key != 'C'); // lap cho den khi bam 'c' -> ket thuc
ham nay
    system("cls");                      // xoa man hinh Console
}
```

- Hàm này sẽ lặp liên tục và nhận phím bấm từ bàn phím rồi truyền vào hàm `ControlCurrentShape()` để xử lý (di chuyển, quay, kiểm tra, tính điểm,...). Tiếp theo sẽ kiểm tra thời gian sau mỗi Game Tick (timer trong struct Tetris) sẽ di chuyển khối Tetromino xuống 1 đơn vị. Bằng cách tạo ra hai biến `before` và `after` và lấy thời gian hiện tại (lúc gán giá trị cho biến) bằng hàm `gettimeofday()`. Ta tính khoảng cách thời gian giữa hai biến này, nếu khoảng cách đó lớn hơn giá trị của Game Tick (timer trong struct Tetris) thì sẽ di chuyển khối Tetromino xuống 1 đơn vị.

2.4.e, Hàm ghi và đọc điểm:

- Hàm ghi lại điểm:

```
void RecordScore()
{
    time_t mytime = time(NULL); // khai bao bien mytime de hien thi ngay gio hien
tai
    char player_name[200];
    printf("Enter your name:\n");
    gets(player_name);
    // mo file voi quyen doc va ghi
    FILE *info = fopen("./tetris_record.txt", "a+");
    // ghi file
    fprintf(info, "Player Name: %s\n", player_name);
```

```
    fprintf(info, "Played Date: %s", ctime(&mytime)); // dung ctime de chuyen doi
sang dang ky tu
    fprintf(info, "Score: %u\n", tetris.score);
    fprintf(info, "_____________________________________________\n");
    fclose(info); // dong file
    system("cls");
    // nhap lua chon
    printf("Press Y to see past records."
           "\nPress R to play again."
           "\nPress M to return to the menu."
           "\nPress any other key to exit.");
    char key = getch();
    switch (key)
    {
    case 'Y':
    case 'y':
        ViewRecordScore();
        game_on_flag = TRUE;
        OpeningScreen();
        break;
    case 'R':
    case 'r':
        game_on_flag = TRUE;
        StartGame();
        break;
    case 'M':
    case 'm':
        game_on_flag = TRUE;
        OpeningScreen();
        break;
    }
    exit(0);
}
```

- Hàm này ghi lại điểm của người chơi ở file tetris_record.txt (cùng cấp với file code của chúng ta), với mode là 'a+' tức là đọc và ghi, sẽ tạo file mới nếu chưa có file, sẽ ghi tiếp vào cuối file (không ghi đè lại mỗi lần ghi).
- Tiếp theo sẽ đưa ra các lựa chọn cho người chơi: xem danh sách điểm đã ghi, chơi lại game, trở về menu.

❖ Hàm xem danh sách điểm:

```
void ViewRecordScore()
{
    system("cls");
    // mo file voi quyen doc
    FILE *info = fopen("./tetris_record.txt", "r");
    if (info != NULL)
    {
        char ch;
        while ((ch = fgetc(info)) != EOF)
            printf("%c", ch);
```

```
    }
    fclose(info);
    system("pause"); // tam dung man hinh
}
```

2.4.f, Hàm start game:

- Hàm này sẽ xắp xếp trình tự thực hiện của các hàm trước đó:

```
void StartGame()
{
    system("cls");                                  // xoa man hinh
    LoadingScreen();                                // man hinh load
    tetris = (Tetris){{{0}}, 0, DEFAULT_TIMER}; // khoi tao game moi
    PrintGamePlayUI();                              // in man hinh game play
    GetNewShape();                                  // lay khoi tetromino moi
    PrintNextShapeToConsole();                  // hien thi khoi tetromino tiep
theo
    UpdateFrame();                                  //
    RecordScore();                                  // ghi lai diem cua nguoi choi
}
```

3, Một số bổ sung và hoàn thiện code:

- Hàm Loading:
  - Hàm này sẽ in ra màn hình hiệu ứng loading đẹp mắt.

```
void LoadingScreen()
{
    HANDLE handle = GetStdHandle(STD_OUTPUT_HANDLE);
    SetConsoleTextAttribute(handle, 11); // set mau xanh duong cho chu
    MoveCursorToXY(16, 14);
    printf("Loading...");
    MoveCursorToXY(10, 15);
    unsigned short i;
    for (i = 1; i <= 20; i++)
    {
        Sleep(100);
        printf("%c", 178);
    }
    system("cls");
    SetConsoleTextAttribute(handle, 7); // set mau chu lai binh thuong
}
```

- Hàm Instructions:
  - Hàm này in ra màn hình các hướng dẫn khi chơi game:

```
void InstructionsScreen()
{
    system("cls");
    printf("Welcome to the Tetris game!"
           "\n\n\nGame instructions:"
           "\n\n<-> Use left and right arrow keys to move blocks across
the screen, down arrow key to bring them down faster, and the up arrow key
to rotate them."
           "\n\n<-> Your objective is to get all the blocks to fill all
the empty space in a row at the bottom of the screen. Thus, the filled row
will vanish and you get awarded 100 points."
           "\n\n<-> Your game is over if a block reaches the top of the
screen."
           "\n\n<-> You can pause the game in its middle by pressing the P
key. To continue the paused game press P once again."
           "\n\n<-> If you want to exit the game at any point press Esc
(you will lose all progress)."
           "\n\n\nPress any key to continue...");
    if (getch() == ESCAPE_KEY)
        exit(0);
    system("cls");
}
```

❖ Hàm CopyShape:

➢ Hàm này để clone một biến riêng biệt từ biến thuộc kiểu dữ liệu Shape.

```
Shape *CopyShape(const Shape shape)
{
    Shape *copy = (Shape *)malloc(sizeof(Shape));
    copy->width = shape.width;
    copy->y = shape.y;
    copy->x = shape.x;
    copy->shape_matrix = (char **)malloc(copy->width * sizeof(char *));
    unsigned short i, j;
    for (i = 0; i < copy->width; i++)
    {
        copy->shape_matrix[i] = (char *)malloc(copy->width *
sizeof(char));
        for (j = 0; j < copy->width; j++)
            copy->shape_matrix[i][j] = shape.shape_matrix[i][j];
    }
```

❖ Hàm DeleteShape:

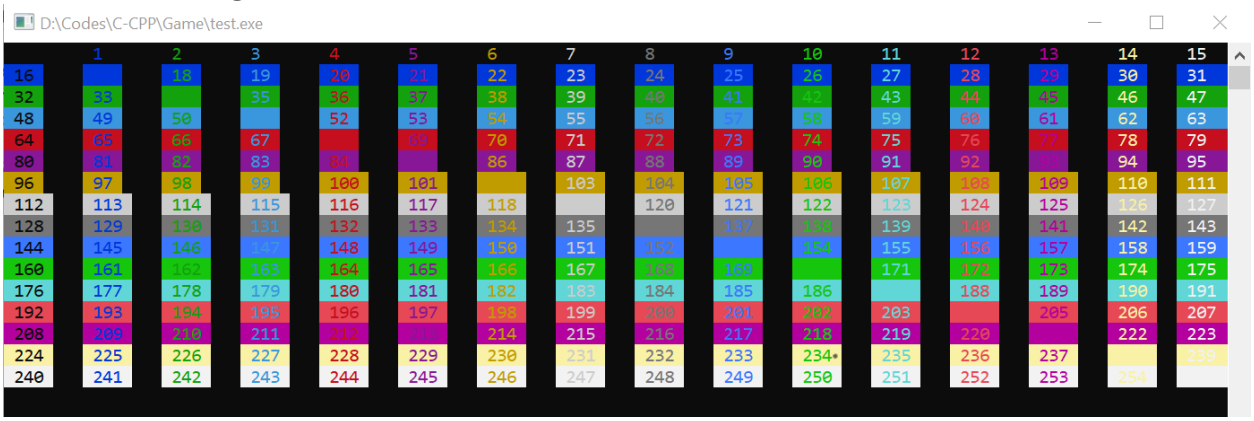
➢ Sau khi clone ra một biến tạm thời bằng Hàm CopyShape ở trên thì khi không cần dùng đến nữa thì chúng ta phải xóa nó đi để tiết kiệm tài nguyên của máy.

```
void DeleteShape(Shape *shape)
{
    unsigned short i;
    for (i = 0; i < shape->width; i++)
        free(shape->shape_matrix[i]);
    free(shape->shape_matrix);
    free(shape);
}
```

❖ Hàm Set màu cho chữ trong Console

```
void SetColor(int a)
{
    HANDLE handle = GetStdHandle(STD_OUTPUT_HANDLE);
    SetConsoleTextAttribute(handle, a);
}
```

➢ Hàm này sẽ đổi màu của chữ trong Console. Với tham số a truyen vào là các sô ứng với các màu như sau:



PHỤ LỤC

Toàn bộ code:

Truy cập vào Github của nhóm: https://github.com/maiquocthinh/Tetris_Game để có thể xem toàn bộ code.

TÀI LIỆU THAM KHẢO

[1] https://codelearn.io/sharing/lam-game-tetris-voi-cpp-don-gian-phan-1. [Accessed 2 April 2022].

[2] https://github.com/BlockDMask/Tetris_Game/blob/master/main.c. [Accessed 18 March 2022].

[3] https://stackoverflow.com/search?q=build+tetris+c. [Accessed 18 March 2022].

[4] https://vi.wikipedia.org/wiki/Tetris. [Accessed 10 April 2022].