TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN TP.HCM
KHOA CÔNG NGHỆ THÔNG TIN
MÔN: **KĨ THUẬT LẬP TRÌNH**

HƯỚNG DẪN ĐỒ ÁN

BĂNG QUA ĐƯỜNG

TP.HCM, ngày 10 tháng 05 năm 2020

MỤC LỤC

1 Giới thiệu

Trong phần đồ án này ta sẽ phối hợp các kĩ thuật và cấu trúc dữ liệu cơ bản để xây dựng một trò chơi đơn giản, băng qua đường.

Để thực hiện được đồ án này ta cần các kiến thức cơ bản như: xử lý tập tin, tiểu trình, handle, cấu trúc dữ liệu mảng một chiều, danh sách liên kết…

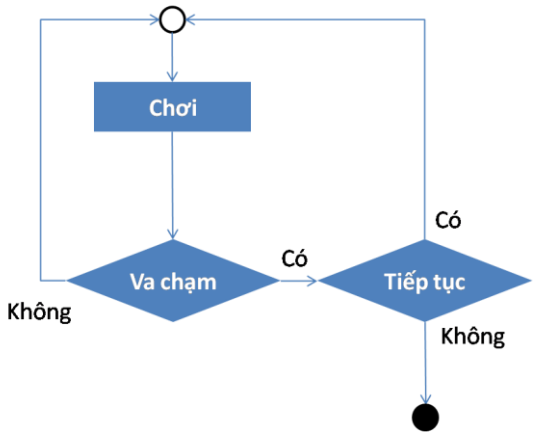
Phần hướng dẫn giúp sinh viên xây dựng trò chơi ở mức độ cơ bản, các em tự nghiên cứu để hoàn thiện một cách tốt nhất có thể.

2 Kịch bản trò chơi

Lúc đầu khi vào game sẽ xuất hiện các xe chạy qua lại và một kí tự "Y" đại diện cho người qua đường, người chơi sử dụng các phím 'W', 'A', 'S', 'D' để điều chỉnh hướng di chuyển của người qua đường và cố gắng tránh các xe.

Khi "Y" va chạm các xe thì chương trình thông báo yêu cầu người chơi chọn phím 'y' nếu muốn tiếp tục (chương trình sẽ reset trò chơi lại như lúc ban đầu) hoặc chọn 'bất kì phím nào' nếu muốn thoát trò chơi.

Khi "Y" đi qua được hết các xe thì thì tốc độ xe di chuyển nhanh hơn, nghĩa là lên 1 cấp (Vị trí của "Y" mới sẽ xuất hiện trở lại). Khi lên cấp 3 thì dữ liệu sẽ reset lại như lúc ban đầu.



Hình 1: Sơ đồ kịch bản trò chơi

3 Các bước xây dựng trò chơi

Trong phần này ta sẽ lần lượt đi qua các bước xây dựng trò chơi. Lưu ý đây chỉ là một gợi ý lập trình, sinh viên có thể tự thiết kế mẫu phù hợp trong quá trình làm đồ án.

Bước 1: Trong bước này ta sẽ cố định màn hình với kích thước thích hợp, làm điều này giúp tránh trường hợp người dùng tự co giãn màn hình sẽ gây khó khăn trong quá trình xử lý.

Dòng	
1	`void FixConsoleWindow() {`
2	`    HWND consoleWindow = GetConsoleWindow();`
3	`    LONG style = GetWindowLong(consoleWindow, GWL_STYLE);`
4	`    style = style & ~(WS_MAXIMIZEBOX) & ~(WS_THICKFRAME);`
5	`    SetWindowLong(consoleWindow, GWL_STYLE, style);`
6	`}`

Trong đoạn mã trên, kiểu HWND là một con trỏ trỏ tới chính cửa sổ Console. Để làm việc với các đối tượng đồ họa này, ta cần có những kiểu như thế. Cờ GWL_STYLE được xem là dấu hiệu để hàm GetWindowLong lấy các đặc tính mà cửa sổ Console đang có. Kết quả trả về của hàm GetWindowLong là một số kiểu long, ta sẽ hiệu chỉnh tại dòng số 4. Ý nghĩa là để làm mờ đi nút maximize và không cho người dùng thay đổi kích thước cửa sổ hiện hành. Sau khi đã hiệu chỉnh xong, ta dùng hàm SetWindowLong để gán kết quả hiệu chỉnh trở lại. Ta có thể thử nghiệm hàm trên và tự xem kết quả.

Bước 2: Trong trò chơi sẽ có rất nhiều vị trí mà ta muốn in tại đó, vì vậy ta cần có khả năng di chuyển tới tất cả các vị trí trong màn hình console.

Dòng	
1	`void GotoXY(int x, int y) {`
2	`    COORD coord;`
3	`    coord.X = x;`
4	`    coord.Y = y;`
5	`    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), coord);`
6	`}`

Trong đoạn mã này ta sử dụng struct _COORD (COORD), đây là một cấu trúc dành xử lý cho tọa độ trên màn hình console. Ta gán hoành độ và tung độ cho biến coord sau đó thiết lập vị trí lên màn hình bằng hàm SetConsoleCursorPosition. Lưu ý: hàm này cần một đối tượng chính là màn hình console (màn hình đen), vì vậy ta cũng cần có một con trỏ trỏ tới đối tượng này (HANDLE thực chất là void*). Ta có được bằng cách gọi hàm GetStdHandle với tham số là một cờ STD_OUTPUT_HANDLE.

Bước 3: Tiếp theo ta cần có dữ liệu để phục vụ trò chơi, để đơn giản ta sử dụng biến toàn cục. Ý nghĩa của từng biến và hằng số sinh viên tự xem trong bảng.

Dòng	
1	`//Hằng số`
2	`#define MAX_CAR 17`
3	`#define MAX_CAR_LENGTH 40`
4	`#define MAX_SPEED 3`
5	`//Biến toàn cục`
6	`POINT** X; //Mảng chứa MAX_CAR xe`
7	`POINT Y; // Đại diện người qua đường`
8	`int cnt = 0;//Biến hỗ trợ trong quá trình tăng tốc độ xe di chuyển`
9	`int MOVING;//Biến xác định hướng di chuyển của người`
10	`int SPEED;// Tốc độ xe chạy (xem như level)`
11	`int HEIGH_CONSOLE, WIDTH_CONSOLE;// Độ rộng và độ cao của màn hình console`
12	`bool STATE; // Trạng thái sống/chết của người qua đường`

Bước 4: Bước tiếp theo ta sẽ xây dựng hàm ResetData, mục tiêu của hàm này là để thiết lập dữ liệu về trạng thái ban đầu. Tổng cộng có MAX_CAR xe, mỗi xe dài MAX_CAR_LENGTH. Ngoài ra ta cũng khởi gán các giá trị ban đầu cho người qua đường, ví dụ MOVING = 'D' để chương trình vẽ kí tự "Y" ra màn hình, cho tốc độ SPEED = 1 và tọa độ ban đầu của "Y".

Dòng	
1	`//Hàm khởi tạo dữ liệu mặc định ban đầu`
2	`void ResetData() {`
3	`    MOVING = 'D'; // Ban đầu cho người di chuyển sang phải`
4	`    SPEED = 1; // Tốc độ lúc đầu`
5	`    Y = { 18,19 }; // Vị trí lúc đầu của người`
6	`    // Tạo mảng xe chạy`
7	`    if (X == NULL) {`
8	`        X = new POINT*[MAX_CAR];`
9	`        for (int i = 0; i < MAX_CAR; i++)`
10	`            X[i] = new POINT[MAX_CAR_LENGTH];`
11	`        for (int i = 0; i < MAX_CAR; i++)`
12	`        {`
13	`            int temp = (rand() % (WIDTH_CONSOLE - MAX_CAR_LENGTH)) + 1;`
14	`            for (int j = 0; j < MAX_CAR_LENGTH; j++)`
15	`            {`
16	`                X[i][j].x = temp + j;`
17	`                X[i][j].y = 2 + i;`
18	`            }`

19	} }
20	}
21	}

Bước 5: Công việc tiếp theo ta cần xây dựng là vẽ một hình chữ nhật bao quanh làm phạm vi. Hàm này vẽ hình chữ nhật dài height và rộng width.

Dòng	
1	`void DrawBoard(int x, int y, int width, int height, int curPosX = 0, int curPosY = 0)`
2	`{`
3	`GotoXY(x, y);cout << 'X';`
4	`for (int i = 1; i < width; i++)cout << 'X';`
5	`cout << 'X';`
6	`GotoXY(x, height + y);cout << 'X';`
7	`for (int i = 1; i < width; i++)cout << 'X';`
8	`cout << 'X';`
9	`for (int i = y + 1; i < height + y; i++)`
10	`{`
11	`GotoXY(x, i);cout << 'X';`
12	`GotoXY(x + width, i);cout << 'X';`
13	`}`
14	`GotoXY(curPosX, curPosY);`
15	`}`

Bước 6: Tiếp theo ta sẽ xây dựng hàm StartGame(), hàm này thực chất là tập các công việc cần làm trước khi vào trò chơi

Dòng	
1	`void StartGame() {`
2	`system("cls");`
3	`ResetData(); // Khởi tạo dữ liệu gốc`
4	`DrawBoard(0, 0, WIDTH_CONSOLE, HEIGH_CONSOLE); // Vẽ màn hình game`
5	`STATE = true;//Bắt đầu cho Thread chạy`
6	`}`

Dòng mã đầu tiên để xóa trắng màn hình, dòng mã thứ hai là lời gọi hàm ResetData() nhằm khôi phục dữ liệu mặc định ban đầu. Kế đó là gọi hàm DrawBoard() để vẽ hình chữ nhật xung quanh. Dòng lệnh cuối cùng là quan trọng, khi STATE = true thì các đối tượng mới chính thức được vẽ ra màn hình.

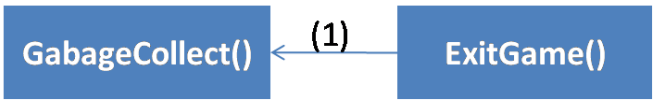
Hình 2: Sơ đồ gọi hàm từ StartGame()

Bước 7: Ngoài hàm StartGame(), ta cần xây dựng hai hàm ExitGame() và PauseGame() nhằm thực hiện chức năng thoát và dừng trò chơi khi cần.

Dòng	
1	`//Hàm dọn dẹp tài nguyên`
2	`void GabageCollect()`
3	`{`
4	`    for (int i = 0; i < MAX_CAR; i++)`
5	`    {`
6	`        delete[] X[i];`
7	`    }`
8	`    delete[] X;`
9	`}`
10	`//Hàm thoát game`
11	`void ExitGame(HANDLE t) {`
12	`   system("cls");`
13	`   TerminateThread(t, 0);`
14	`   GabageCollect();`
15	`}`
16	`//Hàm dừng game`
17	`void PauseGame(HANDLE t) {`
18	`   SuspendThread(t);`
19	`}`

Trong hàm ExitGame() ta thực hiện xóa trắng màn hình và ngắt tiểu trình đang chạy, ta cần có HANDLE của tiểu trình cần ngắt. Lưu ý: Do trong ứng dụng có sử dụng biến con trỏ nên khi thoát ta cần dọn dẹp các tài nguyên này trong hàm GabageCollect().

Trong hàm PauseGame() ta thực hiện tạm dừng chương trình bằng hàm SuspendThread()



Hình 3: Sơ đồ gọi hàm từ ExitGame()

Bước 8: Tiếp theo ta xây dựng hàm xử lý khi "Y" bị va chạm với xe và khi "Y" băng qua đường thành công.

Dòng	
1	`//Hàm xử lý khi người đụng xe`
2	`void ProcessDead() {`
3	`    STATE = 0;`
4	`    GotoXY(0, HEIGH_CONSOLE + 2);`

```
5       printf("Dead, type y to continue or anykey to exit");
6   }
7   //Hàm xử lý khi người băng qua đường thành công
8   void ProcessFinish(POINT& p) {
9       SPEED == MAX_SPEED ? SPEED = 1 : SPEED++;
10      p = { 18,19 }; // Vị trí lúc đầu của người
11      MOVING = 'D'; // Ban đầu cho người di chuyển sang phải
12  }
```

Khi người băng qua đường thất bại, có nghĩa là va chạm xe, thì ta thực hiện tạm ngưng không cho các xe chạy và in dòng chữ báo hiệu cho người chơi biết để quyết định có nên tiếp tục chơi hay không (cho STATE = 0 để duy trì Thread con chạy nhưng không cập nhật màn hình nữa). Trường hợp người băng qua đường thành công thì ta thực hiện tăng SPEED và khởi động lại vị trí cho người mới tiếp theo băng qua đường (cho MOVING = 'D' nhằm kích hoạt Thread vẽ biểu tượng "Y" mới).

Bước 9: Trong bước này, ta cài đặt các hàm để vẽ các xe di chuyển cũng như người băng qua đường (ký tự "Y"). Các hàm này đơn giản chỉ nhảy tới tọa độ của các đối tượng sau đó thực hiện in kí hiệu ra màn hình (Các toa xe là dấu "." Còn người qua đường là "Y")

```
Dòng
1   //Hàm vẽ các toa xe
2   void DrawCars(char* s)
3   {
4       for (int i = 0; i < MAX_CAR; i++) {
5           for (int j = 0; j < MAX_CAR_LENGTH; j++)
6           {
7               GotoXY(X[i][j].x, X[i][j].y);
8               printf(".");
9           }
10      }
11  }
12  //Hàm vẽ người qua đường
13  void DrawSticker(const POINT& p, char* s) {
14      GotoXY(p.x, p.y);
15      printf(s);
16  }
```

Bước 10: Để biết được người qua đường có va chạm xe hay không, ta chỉ cần xác định chỉ số dòng người qua đường đang đứng, có chỉ số dòng ta sẽ xem xét tương ứng toa xe đó có va chạm hay không. Để tiết kiệm chi phí xử lý, ta không cần xem xét tất cả các toa xe mà chỉ xét toa xe trùng với số dòng người qua đường đang đứng.

Dòng	
1	`//Hàm kiểm tra xem người qua đường có đụng xe không`
2	`bool IsImpact(const POINT& p, int d)`
3	`{`
4	`    if (d == 1 || d == 19)return false;`
5	`    for (int i = 0; i < MAX_CAR_LENGTH; i++)`
6	`    {`
7	`        if (p.x == X[d - 2][i].x && p.y == X[d - 2][i].y) return true;`
8	`    }`
9	`    return false;`
10	`}`

Bước 11: Trong bước này, ta sẽ cài đặt hàm di chuyển các toa xe. Thực ra ta không xem tốc độ Sleep của Thread làm độ khó cho trò chơi, vì nếu như vậy thì tốc độ của người băng qua đường "Y" cũng nhanh tương ứng, như vậy sẽ mất ý nghĩa tốc độ. Ở đây mỗi khi người chơi lên cấp thì toa xe di chuyển thêm một bước. Ví dụ ở cấp một thì toa xe dịch chuyển một đơn vị, nhưng khi lên cấp hai thì toa xe dịch chuyển hai đơn vị…

Dòng	
1	`void MoveCars() {`
2	`  for (int i = 1; i < MAX_CAR; i += 2)`
3	`  {`
4	`    cnt = 0;`
5	`    do {`
6	`      cnt++;`
7	`      for (int j = 0; j < MAX_CAR_LENGTH - 1; j++) {`
8	`        X[i][j] = X[i][j + 1];`
9	`      }`
10	`      X[i][MAX_CAR_LENGTH - 1].x + 1 == WIDTH_CONSOLE ? X[i][MAX_CAR_LENGTH - 1].x = 1 : X[i][MAX_CAR_LENGTH - 1].x++; // Kiểm tra xem xe có đụng màn hình không`
11	`    } while (cnt < SPEED);`
12	`  }`
13	`  for (int i = 0; i < MAX_CAR; i += 2)`
14	`  {`
15	`    cnt = 0;`
16	`    do {`
17	`      cnt++;`
18	`      for (int j = MAX_CAR_LENGTH - 1; j > 0; j--)`
19	`      {`
20	`        X[i][j] = X[i][j - 1];`
21	`      }`
22	`      X[i][0].x - 1 == 0 ? X[i][0].x = WIDTH_CONSOLE - 1 : X[i][0].x--;// Kiểm tra xem xe có đụng màn hình không`

```
23      } while (cnt < SPEED);
24    }
25  }
```

Trong hàm này ta xét hai trường hợp. Trường hợp một là các toa xe chẵn và trường hợp hai là các toa xe lẻ. Ở mỗi trường hợp ta có thêm vòng lặp do while để thực hiện tăng dần vị trí của toa xe. Lúc đầu do SPEED = 1 nên vòng lặp này thực chất chỉ làm một lần, nhưng khi SPEED++ thì vòng lặp này sẽ chạy SPEED lần để "từ từ" cập nhật vị trí toa xe. Do đây là hướng dẫn cơ bản nên ta chọn cách minh họa dễ hiểu nhất. Sinh viên có thể tìm cách khác để cập nhật tọa độ toa xe một cách trực tiếp nhanh nhất.

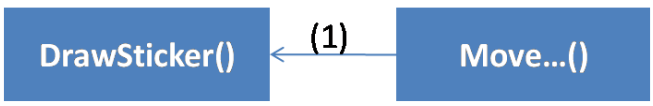
Bước 12: Trong bước này ta phải tìm một cách xóa vị trí xe cũ một cách khôn ngoan để vẽ lại vị trí xe mới. Một cách ta nghĩ ra ngay đó là dùng hàm System("cls") sau đó vẽ lại vị trí toa xe mới. Tuy nhiên cách làm này sẽ gây lãng phí vì thực tế tại một thời điểm chỉ có một vài toa xe cần vẽ lại. Lấy ví dụ toa xe di chuyển từ trái sang phải, như vậy tại một thời điểm ta chỉ cần xóa ô ở vị trí đầu tiên của toa xe là được. Tương tự với các toa xe chạy từ trái sang phải thì ta xóa ô ở vị trí cuối cùng.

```
Dòng
1   // Hàm xóa xe (xóa có nghĩa là không vẽ)
2   void EraseCars()
3   {
4     for (int i = 0; i < MAX_CAR; i += 2) {
5       cnt = 0;
6       do {
7         GotoXY(X[i][MAX_CAR_LENGTH - 1 - cnt].x, X[i][MAX_CAR_LENGTH - 1 - cnt].y);
8         printf(" ");
9         cnt++;
10      } while (cnt < SPEED);
11    }
12    for (int i = 1; i < MAX_CAR; i += 2) {
13      cnt = 0;
14      do {
15        GotoXY(X[i][0 + cnt].x, X[i][0 + cnt].y);
16        printf(" ");
17        cnt++;
18      } while (cnt < SPEED);
19    }
20  }
```

Bước 13: Tiếp theo ta sẽ xây dựng các hàm di chuyển tương ứng khi người chơi điều khiển "Y".

Dòng	
1	`void MoveRight() {`
2	`    if (Y.x < WIDTH_CONSOLE - 1)`
3	`    {`
4	`        DrawSticker(Y, " ");`
5	`        Y.x++;`
6	`        DrawSticker(Y, "Y");`
7	`    }`
8	`}`
9	`void MoveLeft() {`
10	`    if (Y.x > 1) {`
11	`        DrawSticker(Y, " ");`
12	`        Y.x--;`
13	`        DrawSticker(Y, "Y");`
14	`    }`
15	`}`
16	`void MoveDown() {`
17	`    if (Y.y < HEIGH_CONSOLE - 1)`
18	`    {`
19	`        DrawSticker(Y, " ");`
20	`        Y.y++;`
21	`        DrawSticker(Y, "Y");`
22	`    }`
23	`}`
24	`void MoveUp() {`
25	`    if (Y.y > 1) {`
26	`        DrawSticker(Y, " ");`
27	`        Y.y--;`
28	`        DrawSticker(Y, "Y");`
29	`    }`
30	`}`

Trong quá trình di chuyển "Y", nếu vượt quá hình chữ nhật đóng khung thì ta không xử lý, ngược lại ta thực hiện cập nhật tọa độ và vẽ lại vị trí mới cho "Y".



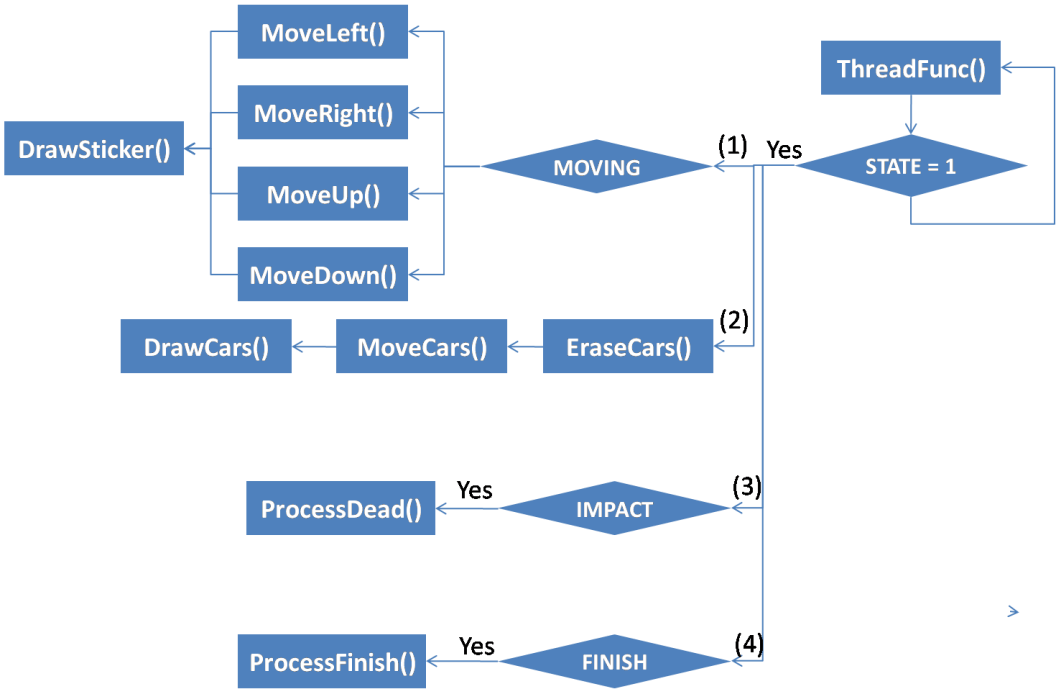
Hình 4: Sơ đồ gọi hàm từ các hàm Move…()

Bước 14: Tiếp theo ta sẽ cài đặt hàm chạy cho tiểu trình. Ta chỉ xử lý trường hợp khi người băng qua đường còn sống (STATE = true). Ta sẽ dựa vào biến MOVING để xác

định hướng di chuyển cho “Y”, đồng thời ta thực hiện cập nhật vị trí và vẽ lại tọa độ cho toa xe. Ta cũng kiểm tra xem có va chạm hay về đích hay không để có những xử lý thích hợp.

Dòng	
1	`void SubThread()`
2	`{`
3	`  while (1) {`
4	`    if (STATE) //Nếu người vẫn còn sống`
5	`    {`
6	`      switch (MOVING) //Kiểm tra biến moving`
7	`      {`
8	`        case 'A':`
9	`          MoveLeft();`
10	`          break;`
11	`        case 'D':`
12	`          MoveRight();`
13	`          break;`
14	`        case 'W':`
15	`          MoveUp();`
16	`          break;`
17	`        case 'S':`
18	`          MoveDown();`
19	`          break;`
20	`      }`
21	`      MOVING = ' ';// Tạm khóa không cho di chuyển, chờ nhận phím từ hàm main`
22	`      EraseCars();`
23	`      MoveCars();`
24	`      DrawCars(".");`
25	`      if (IsImpact(Y, Y.y))`
26	`      {`
27	`        ProcessDead(); // Kiểm tra xe có đụng không`
28	`      }`
29	`      if (Y.y == 1)`
30	`        ProcessFinish(Y); // Kiểm tra xem về đích chưa`
31	`      Sleep(50);//Hàm ngủ theo tốc độ SPEED`
32	`    }`
33	`  }`
34	`}`

Bên dưới là sơ đồ gọi hàm trong hàm SubThread



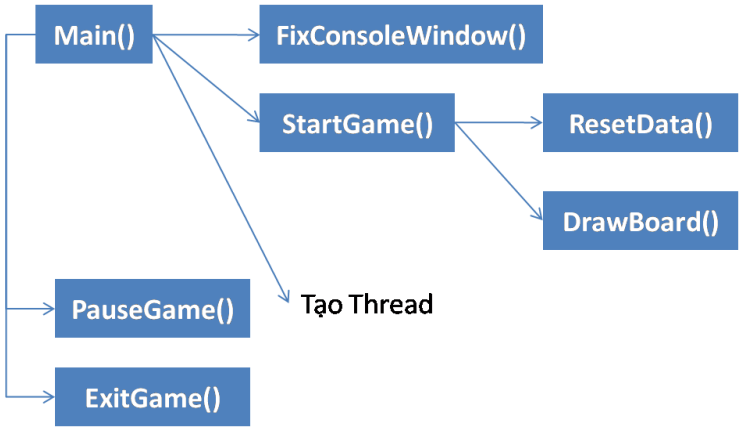
Hình 5: Sơ đồ gọi hàm từ hàm SubThread()

Bước 15: Cuối cùng ta xây dựng hàm main để thực hiện điều khiển tiểu trình đồng thời tiếp nhận phím từ người dùng.

Dòng	
1	`void main()`
2	`{`
3	`    int temp;`
4	`    FixConsoleWindow();`
5	`    srand(time(NULL));`
6	`    StartGame();`
7	`    thread t1(SubThread);`
8	`    while (1)`
9	`    {`
10	`        temp = toupper(getch());`
11	`        if (STATE == 1)`
12	`        {`
13	`            if (temp == 27) {`
14	`                ExitGame(t1.native_handle());`
15	`                return;`
16	`            }`
17	`            else if (temp == 'P') {`
18	`                PauseGame(t1.native_handle());`

```
19              }
20              else {
21                  ResumeThread((HANDLE)t1.native_handle());
22                  if (temp == 'D' || temp == 'A' || temp == 'W' || temp == 'S')
23                  {
24                      MOVING = temp;
25                  }
26              }
27          }
28          else
29          {
30              if (temp == 'Y') StartGame();
31              else {
32                  ExitGame(t1.native_handle());
33                  return;
34              }
35          }
36      }
37  }
```

Trong hàm main, bước đầu tiên là ta cố định màn hình ngăn ngừa người chơi thay đổi kích thước sẽ gây vỡ chương trình. Ta cũng tạo bộ số ngẫu nhiên cho các tọa độ chương trình. Sau đó ta gọi StartGame() để thực hiện chuẩn bị dữ liệu cho màn chơi. Sau câu lệnh thread t1(SubThread) thì tiểu trình sẽ xuất hiện và chạy song song cùng hàm main(). Hàm main() liên tục chờ phím từ người chơi, sau đó tùy vào phím người dùng chọn chương trình sẽ có phản ứng thích hợp.



Hình 6: Sơ đồ gọi hàm từ hàm main ()

4 YÊU CẦU ĐỒ ÁN

Trong phần hướng dẫn trên ta còn thiếu một vài chức năng cơ bản

4.1 Xử lý va chạm với người băng qua đường trước đó (3đ)

Trong hướng dẫn chưa xử lý việc "Y" va chạm với các "Y" đã về trước đó. Khi điều này xảy ra ta cũng xử lý tạm dừng trò chơi và hỏi ý kiến người dùng xem có muốn tiếp tục hay không?

4.2 Xử lý lưu trò chơi/tải trò chơi đã lưu (3đ)

Sinh viên bổ sung chức năng khi người dùng nhấn phím 'L' thì xuất hiện dòng lệnh yêu cầu người dùng nhập tên tập tin cần lưu lại. Tương tự khi nhấn phím 'T' thì xuất hiện dòng lệnh yêu cầu người dùng nhập tên tập tin cần tải lên.

Hướng dẫn: Sinh viên tự tổ chức cấu trúc tập tin để lưu dữ liệu biến toàn cục trong chương trình

4.3 Xử lý tạm dừng các toa xe (2đ)

Trong hướng dẫn, các xe di chuyển liên tục, sinh viên hãy hiệu chỉnh lại sao cho mỗi toa xe đều có thể dừng lại trong một khoảng thời gian tùy ý để giúp trò chơi dễ chơi hơn khi lên cấp.

4.4 Xử lý hiệu ứng khi va chạm (1đ)

Khi người qua đường va chạm xe thì tạo hiệu ứng đơn giản minh họa việc va chạm.

4.5 Xử lý màn hình chính (1đ)

Khi mở trò chơi, chương trình cho phép người dùng chọn 'Tải lại' (phím 'T') hoặc 'Bắt đầu chơi' (phím bất kì). Nếu người dùng chọn phím 'T' thì yêu cầu người dùng nhập tên tập tin và sau đó vào trò chơi. Ngược lại thì bắt đầu trò chơi với dữ liệu gốc mặc định.