

ĐỒ ÁN BĂNG QUA ĐƯỜNG CROSSY-ROAD – NHÓM 15

KỸ THUẬT LẬP TRÌNH

GVHD: Thầy Trương Toàn Thịnh

KỊCH BẢN GAME



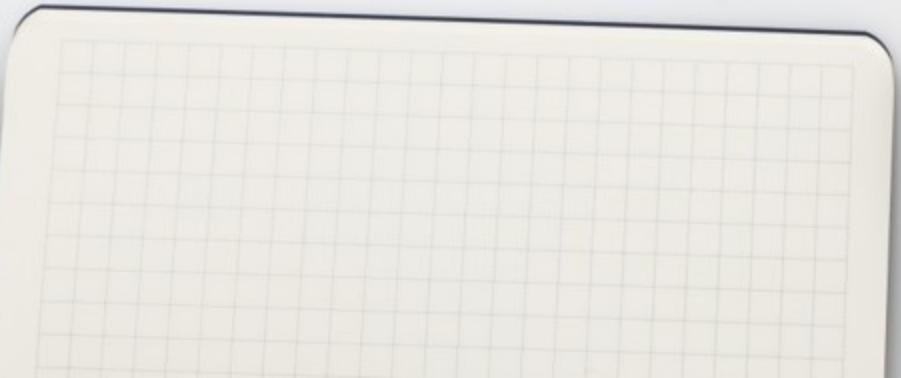
- ✓ Băng qua đường là tựa game giải trí với thao tác vô cùng đơn giản. Người chơi sẽ hóa thành nhân vật (Y) với nhiệm vụ băng qua đường lớn với dòng xe đông đúc mà không gặp tai nạn.
- ✓ Game có 3 level, độ khó sẽ tăng dần và không chạm mặt nhân vật trước nhưng có những khoảng thời gian các xe tạm đứng yên để người chơi có thể “thở” sau các pha tạt đầu xe kịch tính.
- ✓ Người chơi có thể nhấn phím các phím di chuyển (W, S, A, D), phím ‘P’ để dừng, phím ‘R’ để tiếp tục chơi, phím ‘K’ lưu game và ‘Esc’ để quay về màn hình chính.
- ✓ Khi vào game, màn hình Menu sẽ có 4 khung cho bạn lựa chọn: “New Game”, “Load Game”, “Exit” và “About”.

YÊU CẦU ĐỒ ÁN



- Xử lý va chạm với người băng qua đường trước đó
- Xử lý lưu trò chơi/tải trò chơi đã lưu
- Xử lý tạm dừng các toa xe
- Xử lý hiệu ứng khi va chạm
- Xử lý màn hình chính

CÁC BƯỚC XÂY DỰNG



Phần 1 (varsANDlibs.h)

Xây dựng thư viện chứa khai báo các thư viện chuẩn của C++, các hằng số, kiểu dữ liệu, biến toàn cục và hàm cơ bản

BƯỚC 1

Khai báo các thư viện chuẩn C++ cần thiết.

BƯỚC 2

Khai báo các biến cho game bao gồm MAX_CAR, MIN_CAR_LENGTH, MAX_CAR_LENGTH, WIDTH_CONSOLE, HEIGHT_CONSOLE, PLAYGROUND_SECTION_HEIGHT, INFO_SECTION_HEIGHT và thông số nhân vật.

BƯỚC 3

Dòng	
1	<code>struct CAR {</code>
2	<code> int direction;</code>
3	<code> int length;</code>
4	<code>};</code>
5	<code>CAR* carInfo;</code>
6	<code>int** carArray;</code>

- Khởi tạo cấu trúc CAR gồm 2 thông tin *direction* và *length*.
- Biến con trỏ *carInfo* có kiểu cấu trúc CAR dạng mảng động 1 chiều.
- Tiếp đến tạo mảng động 2 chiều *carArray*.

BƯỚC 4

Dòng	
1	<code>POINT player_pos;</code>
2	<code>int speedUP = 0;</code>
3	<code>int direction, speed, step;</code>
4	<code>bool state;</code>
5	<code>int prevPos[5];</code>
6	<code>string player_name;</code>
7	<code>POINT ghost_pos;</code>
8	<code>int ghost_height = 5, ghost_width = 7;</code>
9	<code>string ghost_shape[5];</code>

Tiếp tục khai báo các biến về vị trí người chơi (*player_pos*), các biến về xe (*direction, speed, step*), biến hỗ trợ tăng tốc xe khi bắt đầu di chuyển (*speedUP*), trạng thái sống hay chết (*state*); mảng lưu vị trí trước đó, tên người chơi; hiệu ứng về “hồn ma” (*ghost_pos*).

BƯỚC 5

Cố định màn hình tránh trường hợp người dùng tự co giãn màn hình gây khó khăn trong quá trình xử lý bằng cách gọi hàm FixConsoleWindow().

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

Kiểu HWND là con trỏ trỏ tới chính cửa sổ Console. Cờ GWL_STYLE 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.

BƯỚC 6

Xây dựng hàm deleteCarData() để giải phóng dữ liệu vùng nhớ trong ổ cứng sau khi ngừng chơi.

Dòng	
1	<code>void deleteCarData() {</code>
2	<code> if (carArray == NULL carInfo == NULL) return;</code>
3	<code> for (int i = 0; i < MAX_CAR; i++)</code>
4	<code> delete[] carArray;</code>
5	<code> delete[] carInfo;</code>
6	<code>}</code>

BƯỚC 7

Lập hàm bool isFileExist() để kiểm tra xem tệp có tồn tại hay không.

Dòng	
1	<code>bool isFileExist(string fileName) {</code>
2	<code> fstream fileInput(fileName + ".txt");</code>
3	<code> return fileInput.good();</code>
4	<code>}</code>

BƯỚC 8

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	<code>void GotoXY(int x, int y) {</code>
2	<code>COORD coord;</code>
3	<code>coord.X = x;</code>
4	<code>coord.Y = y;</code>
5	<code>SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE),</code> <code>coord);</code>
6	<code>}</code>

Struct COORD là cấu trúc xử lý tọa độ trên màn hình console. Gán hoành độ và tung độ cho biến *coord* sau đó thiết lập vị trí lên màn hình bằng hàm SetConsoleCursorPosition (HANDLE thực chất là *void**). Ta có được bằng cách gọi hàm GetStdHandle với tham số là một cờ STD_OUTPUT_HANDLE.

BƯỚC 9

Xây dựng hàm setTextColor() chỉnh màu cho ký tự in ra console.

BƯỚC 10

Dòng	
1	void resetData() {
2	deleteCarData(); direction = 'D'; speed = 1; step = 0;
3	player_name = DEFAULT_PLAYER_NAME;
4	player_pos = DEFAULT_CHARACTER_POS;
5	carArray = new int* [MAX_CAR];
6	carInfo = new CAR[MAX_CAR];
7	srand((unsigned)time(NULL));
8	for (int i = 0; i < MAX_CAR; i++) {
9	carInfo[i].direction = rand() % 2;
10	carInfo[i].length = MIN_CAR_LENGTH + rand() % (MAX_CAR_LENGTH - MIN_CAR_LENGTH + 1);
11	carArray[i] = new int[carInfo[i].length];
12	int temp = (rand() % (WIDTH_CONSOLE - carInfo[i].length)) + 1;
13	for (int j = 0; j < carInfo[i].length; j++)
14	carArray[i][j] = temp + j; }
15	memset(prevPos, 0, sizeof(prevPos)); }

- Cần xây dựng hàm resetData() nếu người chơi tạo game mới bằng việc đặt lại các giá trị biến đã gọi trước đó.
- Lệnh rand() tạo hướng, chiều dài xe ngẫu nhiên và vị trí ngẫu nhiên của xe thứ i nhờ biến temp.
- Sau cùng, reset mảng prevPos nhờ hàm memset.

Phần 2 (gameControl.h)

Xây dựng thư viện chứa các hàm sử dụng để quản lí và
điều khiển các chức năng khi
chơi game, các hàm trang trí, hiệu ứng

BƯỚC 1

Dòng	
1	bool isImpact(const POINT& p) {
2	if (player_pos.y == 2 player_pos.y == 24)
3	return false;
4	for (int i = 0; i < carInfo[player_pos.y - 3].length; i++)
5	if (p.x == carArray[player_pos.y - 3][i])
6	return true;
7	return false;
8	}

Khi người chơi đang đứng ở vị trí p.y tương ứng với xe thứ y – 3. Ta kiểm tra vị trí hoành độ p.x của nhân vật có trùng với bất kỳ đoạn nào của xe hay không. Nếu có trả về True (đã va chạm) và ngược lại.

BƯỚC 2

Xây dựng hàm processDead() xử lí khi người chơi thua do đụng xe.

Dòng	
1	void processDead() {
2	ghost_pos.y = max(player_pos.y, 8);
3	ghost_pos.x = min(player_pos.x, WIDTH_CONSOLE - 8);
4	setTextColor(8);
5	for (int i = PLAYGROUND_SECTION_HEIGHT; i > ghost_pos.y; --i) {
6	fillBox(1, i, 4, 0, "X_X");
7	fillBox(4, i, WIDTH_CONSOLE - 7, 0, " X_X"); }
8	while (ghost_pos.y > 5) {
9	fillBox(1, ghost_pos.y + 1, 4, 0, "X_X");
10	fillBox(4, ghost_pos.y + 1, WIDTH_CONSOLE - 7, 0, "X_X");
11	for (int i = ghost_height - 1; i >= 0; i--) {
12	fillBox(1, ghost_pos.y - i, WIDTH_CONSOLE - 2, 0, "");
13	GotoXY(ghost_pos.x, ghost_pos.y - i);
14	setTextColor(6);
15	cout << ghost_shape[ghost_height - i - 1]; }
16	ghost_pos.y--;
17	Sleep(75); }

Dòng	
18	while (ghost_pos.y > 0) {
19	fillBox(1, ghost_pos.y + 1, 4, 0, "X_X");
20	fillBox(4, ghost_pos.y + 1, WIDTH_CONSOLE - 7, 0, "X_X");
21	for (int i = ghost_pos.y - 2; i >= 0; i--) {
22	fillBox(1, ghost_pos.y - i, WIDTH_CONSOLE - 2, 0, "");
23	GotoXY(ghost_pos.x, ghost_pos.y - i);
24	setTextColor(6);
25	cout << ghost_shape[ghost_height - i - 1]; }
26	ghost_pos.y--;
27	Sleep(100); }
28	fillBox(1, 1, WIDTH_CONSOLE - 2, HEIGHT_CONSOLE - 3, "");
29	Sleep(250);
30	endGameMenu();
31	setTextColor(7); }

BƯỚC 3

Xây dựng hàm processPass() xử lí khi người chơi qua màn.

Dòng	
1	void processPass(POINT& p) {
2	prevPos[speed - 1] = p.x;
3	if (speed == MAX_SPEED) {
4	for (int i = 0; i < 3; ++i) {
5	GotoXY(prevPos[i], 2);
6	cout << " "; }
7	speed = 1; }
8	else {
9	speed++;
10	for (int i = 0; i < MAX_CAR; ++i)
11	carInfo[i].direction = rand() % 2; }
12	prevPos[speed - 1] = p.x;
13	p = DEFAULT_CHARACTER_POS;
14	direction = 'D';
15	}

- Lưu lại vị trí hoành độ của nhân vật hiện tại vào mảng *prev_pos* khi thắng.
- Có 2 trường hợp xảy ra:
 - ❑ Ở *level* < *MAX_LEVEL*, tăng *level* lên 1 đơn vị (*speed++*) và cập nhật lại hướng di chuyển mới cho từng xe.
 - ❑ Ở *MAX_LEVEL*, ta xoá các vị trí nhân vật đã về đích cũ trên console và đặt lại *level* *speed* = 1.
- Cuối cùng, ta chuyển nhân vật mới về vị trí mặc định, sẵn sàng cho màn game mới.

BƯỚC 4

Xây dựng hàm drawCharacter() in ra nhân vật trên màn hình console.

BƯỚC 5

Dòng	
1	void drawCars() {
2	setTextColor(8);
3	for (int i = 0; i < MAX_CAR; i++) {
4	string curCar(carInfo[i].length, char(223));
5	curCar[1] = char(15);
6	curCar[curCar.length() - 2] = char(15);
7	if (carInfo[i].direction == 1) {
8	curCar[0] = char(96);
9	curCar[curCar.length() - 1] = char(62); }
10	else {
11	curCar[0] = char(60);
12	curCar[curCar.length() - 1] = char(39); }
13	for (int j = 0; j < carInfo[i].length; j++) {
14	GotoXY(carArray[i][j], i + 3);
15	cout << curCar[j]; } }
16	setTextColor(7);
17	}

- Ta tạo ra chuỗi biểu thị xe bằng cách tạo mỗi string có độ dài bằng với độ dài xe và thay đổi các kí tự sao cho phù hợp.
- Dùng hàm GotoXY() đã xây dựng để di chuyển tới vị trí in ra từng đoạn xe.



BƯỚC 6

Dòng	
1	void moveCars() {
2	for (int i = 0; i < MAX_CAR; i++) {
3	if (carInfo[i].direction == 1) {
4	speedUP = 0;
5	do {
6	speedUP++;
7	for (int j = 0; j < carInfo[i].length - 1; j++) {
8	carArray[i][j] = carArray[i][j + 1]; }
9	carArray[i][carInfo[i].length - 1] + 1 == WIDTH_CONSOLE ? carArray[i][carInfo[i].length - 1] = 1 : carArray[i][carInfo[i].length - 1]++;
10	} while (speedUP < speed); }
11	if (carInfo[i].direction == 0) {
12	speedUP = 0;
13	do { speedUP++;
14	for (int j = carInfo[i].length - 1; j > 0; j--) {
15	carArray[i][j] = carArray[i][j - 1]; }
16	carArray[i][0] - 1 == 0 ? carArray[i][0] = WIDTH_CONSOLE - 1 : carArray[i][0]--;
17	} while (speedUP < speed); } } }

- Cần có hàm moveCar() cập nhật vị trí mới các toa xe vì hàm drawCars() chỉ in xe tĩnh. Sau mỗi lần cập nhật, mỗi phần của toa xe sẽ nhận vị trí của toa đi trước, khi hàm drawCars() được gọi thì toa xe này đã di chuyển lên trước.
- Biến speedUp giúp cập nhật vị trí liên tục.
- Ngoài ra, cần xử lý trường hợp đầu xe chạm biên trái - phải, khi đó vị trí đầu xe sẽ cập nhật thành vị trí đầu tiên đầu đối diện (1 hoặc WIDTH_CONSOLE – 1).

BƯỚC 7

Dòng	
1	void eraseCars() {
2	for (int i = 0; i < MAX_CAR; i++) {
3	if (carInfo[i].direction == 0) {
4	speedUP = 0;
5	do {
6	GotoXY(carArray[i][carInfo[i].length - 1 - speedUP], i + 3);
7	cout << " ";
8	speedUP++;
9	} while (speedUP < speed); }
10	if (carInfo[i].direction == 1) {
11	speedUP = 0;
12	do {
13	GotoXY(carArray[i][0 + speedUP], i + 3);
14	cout << " ";
15	speedUP++;
16	} while (speedUP < speed); } } }

- Xây dựng hàm eraseCar() để xoá phần đuôi xe thừa do sau khi in ra vị trí mới thì không có ký tự đè lên.
- Hoàn toàn có thể biết được độ dài và dễ dàng xoá nhờ vào biến *speed* và *speedUP*.

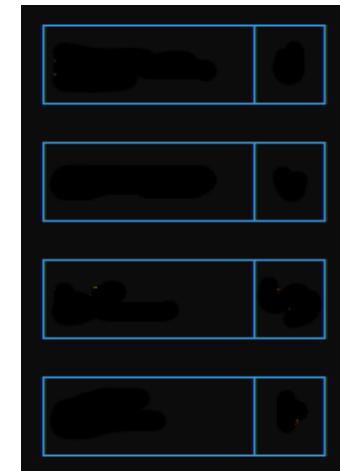
BƯỚC 8

Dòng	
1	void fillBox(int x, int y, int width, int height, string s) {
2	setTextColor(3);
3	for (int i = x; i <= x + width; i += int(s.length())) {
4	for (int j = y; j <= y + height; ++j) {
5	GotoXY(i, j);
6	cout << s; } } }

Hàm fillBox() dùng để vẽ khung diện tích hình chữ nhật màn hình console từ tọa độ (x, y) đến (x+width, y+height) với chuỗi s.

BƯỚC 9

Công dụng của hàm drawSelectBox() dùng để vẽ bảng biểu chèn nội dung chức năng các phím. Trong hàm sử dụng char(196) để vẽ thanh ngang, char(179) để vẽ thanh dọc. Các char còn lại dùng để vẽ các khớp nối.

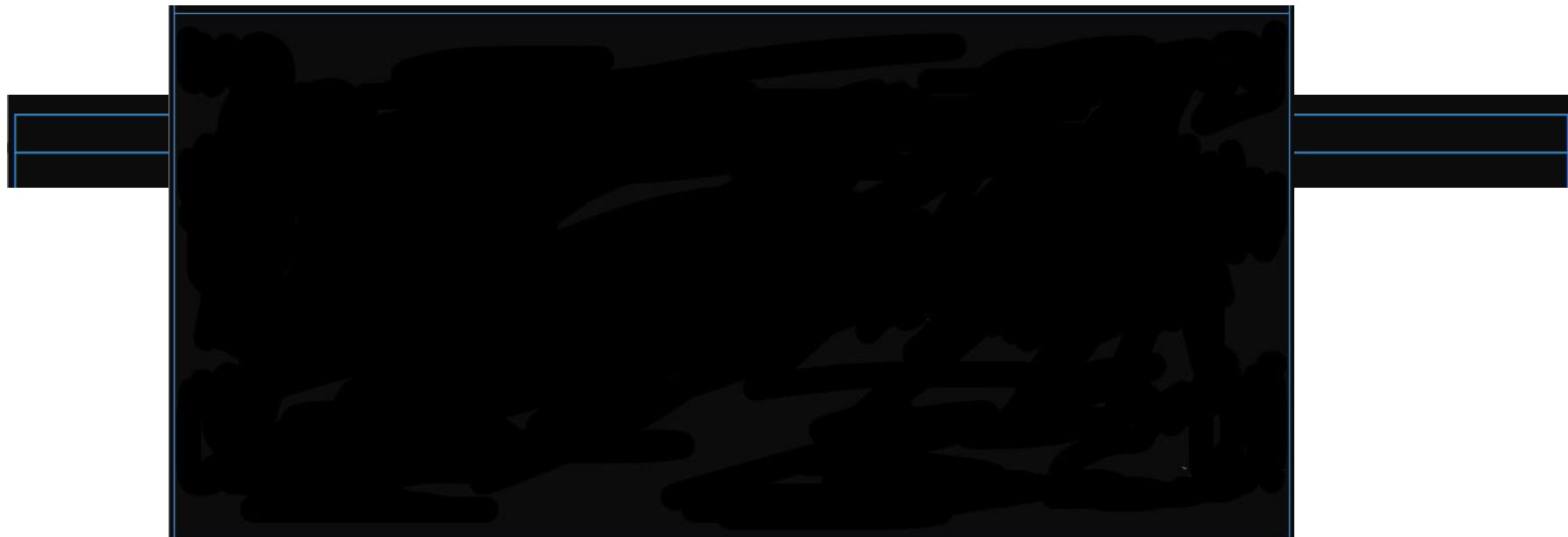


BƯỚC 10

Hàm drawAbout() in ra thông tin về nhóm, link source code và hướng dẫn chơi game.

BƯỚC 11

Xây dựng hàm drawBoard() để khung hình chữ nhật bao quanh làm phạm vi chơi.



Khung giao diện game

BƯỚC 16

Tạo hàm drawInfo() in chỉ số riêng của người chơi ra màn hình

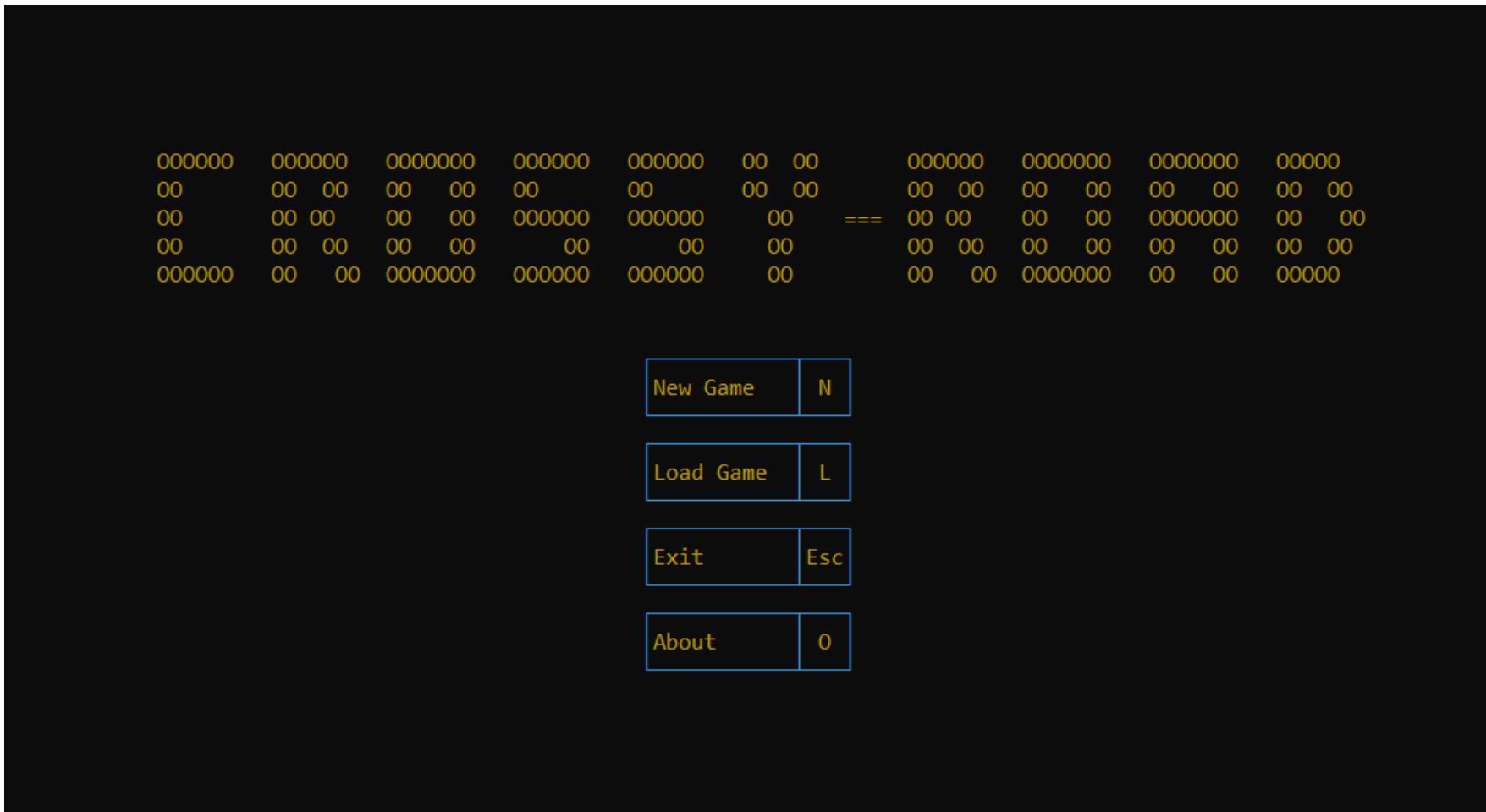
BƯỚC 17

Lập hàm drawGameMenu() hiển thị thông tin về tên người chơi, level, điểm số và các thao tác tùy chọn như di chuyển (W, S, A, D), Pause Game, Resume Game, Save Game, Return to Main Menu.



BƯỚC 18

Hàm startMenu() là giao diện chính khi vào game.



BƯỚC 19

endGameMenu() xử lí giao diện khi bạn thua và hỏi muốn chơi lại hay không.

Dòng	
... 25	ghost_pos.y = HEIGHT_CONSOLE - 3;
26	for (int i = 0; i < 5; ++i) {
27	fillBox(1, ghost_pos.y + 1, WIDTH_CONSOLE - 2, 0, " ");
28	for (int j = i; j >= 0; j--) {
29	GotoXY(WIDTH_CONSOLE - i - ghost_width - 10, ghost_pos.y - j);
30	setTextColor(6);
31	cout << ghost_shape[i - j];
32	}
33	Sleep(150);
34	}
...	
45	state = 0;
46	setTextColor(7);
47	}

BƯỚC 19

endGameMenu() xử lí giao diện khi bạn thua và hỏi muốn chơi lại hay không.



Phần 3 (gameCommand.h)

Xây dựng thư viện chứa hàm xử lý các thao tác được
yêu cầu từ người chơi

BƯỚC 1

Dòng	
1	void startGame() {
2	system("cls"); resetData(); drawGameMenu();
3	drawBox(0, 0, WIDTH_CONSOLE, PLAYGROUND_SECTION_HEIGHT);
4	drawBox(0, PLAYGROUND_SECTION_HEIGHT + 1, WIDTH_CONSOLE, INFO_SECTION_HEIGHT);
5	state = 1; }

- Hàm startGame() để khởi tạo lại dữ liệu ban đầu và tạo màn chơi mới cho người chơi.
- Dòng lệnh cuối cùng quan trọng khi state = 1 thì các đối tượng mới chính thức được vẽ ra màn hình.

BƯỚC 2

Xây dựng hàm thoát khỏi trò chơi, hàm tạm dừng và tiếp tục trò chơi. Cần lệnh HANDLE để ngắt tiểu trình đang chạy, tạm ngưng và tiếp tục trò chơi.

Dòng	
1	void exitGame(HANDLE t) {
2	system("cls");
3	TerminateThread(t, 0);
4	deleteCarData();
5	exit(1); }
6	void pauseGame(HANDLE t) {
7	SuspendThread(t); }
8	void resumeGame(HANDLE t) {
9	ResumeThread(t); }

BƯỚC 3

Dòng	
1	void saveGame(string player_name, POINT player_pos) {
2	string fileName = player_name + ".txt";
3	fstream fileOutput;
4	fileOutput.open(fileName, ios::out ios::trunc);
5	fileOutput << speed << "\n" << step << "\n";
6	for (int i = 0; i < MAX_CAR; i++) {
7	fileOutput << carInfo[i].length << "\n" << carInfo[i].direction << "\n";
8	for (int j = 0; j < carInfo[i].length; j++)
9	fileOutput << carArray[i][j] << "\n";
10	}
11	for (int i = 0; i < speed; ++i) {
12	fileOutput << prevPos[i] << "\n";
13	}
14	fileOutput << player_pos.x << "\n" << player_pos.y;
15	fileOutput.close();
16	}

- Hàm saveGame() có chức năng cơ bản là ghi file sử dụng thư viện fstream.h.
- Các thông số được ghi lên file gồm level và số bước đã di chuyển hiện tại, hướng di chuyển, độ dài và vị trí của từng xe, vị trí của các nhân vật đã về đích trước đó, vị trí người chơi hiện tại.
- Toàn bộ dữ liệu trên được lưu vào file có cú pháp filename.txt.

BƯỚC 4

Dòng	
1	void loadGame(string player_name, POINT& player_pos) {
2	system("cls");
3	drawBox(0, 0, WIDTH_CONSOLE, PLAYGROUND_SECTION_HEIGHT);
4	drawBox(0, PLAYGROUND_SECTION_HEIGHT + 1, WIDTH_CONSOLE, INFO_SECTION_HEIGHT);
5	string fileName = player_name + ".txt";
6	fstream fileInput(fileName);
7	fileInput >> speed >> step;
8	for (int i = 0; i < MAX_CAR; i++) {
9	fileInput >> carInfo[i].length >> carInfo[i].direction;
10	carArray[i] = new int[carInfo[i].length];

Dòng	
11	for (int j = 0; j < carInfo[i].length; j++)
12	fileInput >> carArray[i][j]; }
13	setTextColor(6);
14	for (int i = 0; i < speed; ++i) {
15	fileInput >> prevPos[i];
16	drawCharacter({ prevPos[i], 2 }, "Y"); }
17	fileInput >> player_pos.x >> player_pos.y;
18	fileInput.close();
19	drawCharacter(player_pos, "Y");
20	drawCars();
21	drawInfo();
22	drawGameMenu();

Hàm loadGame() nhờ vào thư viện fstream có khả năng đọc dữ liệu từ file .txt. Có một điểm cần lưu ý là phải cấp phát lại kích thước vùng nhớ mới cho mảng carArray tránh trường hợp thiếu hoặc thừa vùng nhớ. Sau đó ta in các thông tin đã đọc được ra màn hình console.

Phần 4 (gameMovement.h)

Xây dựng thư viện chứa các hàm hỗ trợ
di chuyển nhân vật trong game

Xây dựng các hàm hỗ trợ nhân vật di chuyển qua trái, phải, lên, xuống.

Dòng	
1	void moveRight() {
2	step++;
3	drawCharacter(player_pos, " ");
4	if (player_pos.x > WIDTH_CONSOLE - 2)
5	player_pos.x = 0;
6	player_pos.x++;
7	drawCharacter(player_pos, "Y");
8	}
...	
17	void moveDown() {
18	if (player_pos.y < PLAYGROUND_SECTION_HEIGHT) {
19	step++;
20	drawCharacter(player_pos, " ");
21	player_pos.y++;
22	drawCharacter(player_pos, "Y");
23	}
24	}
...	

- Đối với hai hàm chuyển động trái-phải vì nếu nhân vật chạm biên của vùng chơi và tiếp tục di chuyển, nhân vật sẽ cập nhật vị trí mới là tại biên của phía đối diện.
- Đối với hàm chuyển động lên xuống, nhân vật không được đi xuống nếu đang đứng ở vị trí xuất phát, cũng như không thể đi lên nếu đang ở vị trí đích.

Phần 5 (consoleControl.h)

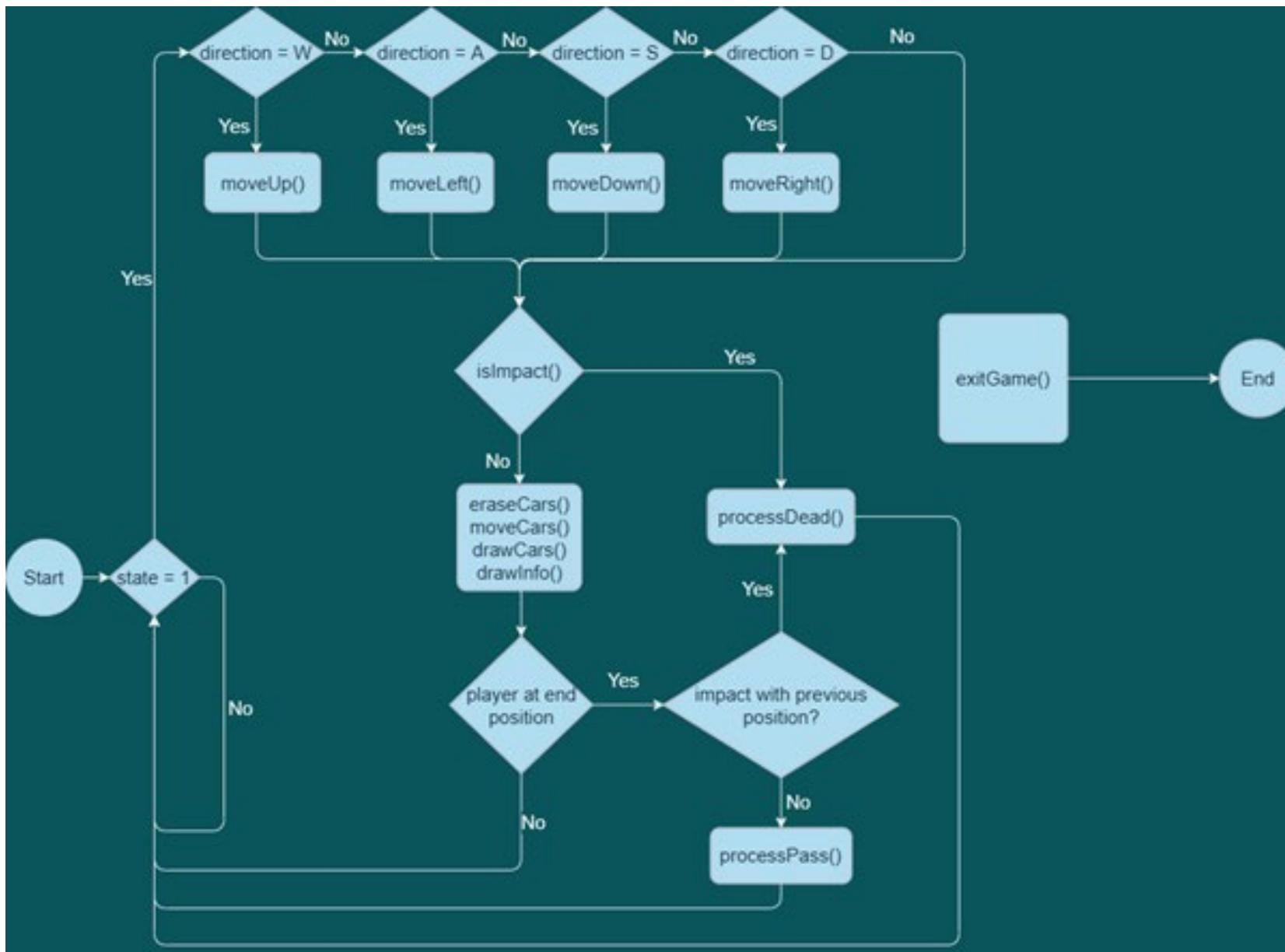
Xây dựng thư viện chứa hàm chạy cho
tiểu trình, quản lý những sự việc diễn ra trên màn hình
console

Xây dựng hàm subThread() quản lý chuyển động và mọi sự kiện trong game.

Dòng	
1	void subThread() {
2	int delay = 0;
3	int time2stop = speed * 10 + 1;
4	while (1) {
5	if (state) {
6	switch (direction) {
7	case 'A': moveLeft(); break;
8	case 'D': moveRight(); break;
9	case 'W': moveUp(); break;
10	case 'S': moveDown(); break; }
11	direction = ' ';
12	if (isImpact(player_pos)) {
13	processDead();
14	delay = 1;
15	time2stop = 0; }
16	delay == 1 ? time2stop++ : time2stop--;
17	if (time2stop == 0)
18	delay = 1;
19	else if (time2stop == speed * 10 + 1)
20	delay = 0;

Dòng	
21	if (delay == 0) {
22	eraseCars();
23	moveCars();
24	drawCars(); }
25	drawInfo();
26	if (player_pos.y == 2) {
27	if (player_pos.x == prevPos[0] player_pos.x == prevPos[1]) {
28	processDead();
29	delay = 1;
30	time2stop = 0; }
31	else {
32	processPass(player_pos);
33	delay = 0;
34	time2stop = speed * 10 + 1; } }
35	Sleep(time2stop); } } }

Xây dựng hàm subThread() quản lý chuyển động và mọi sự kiện trong game.

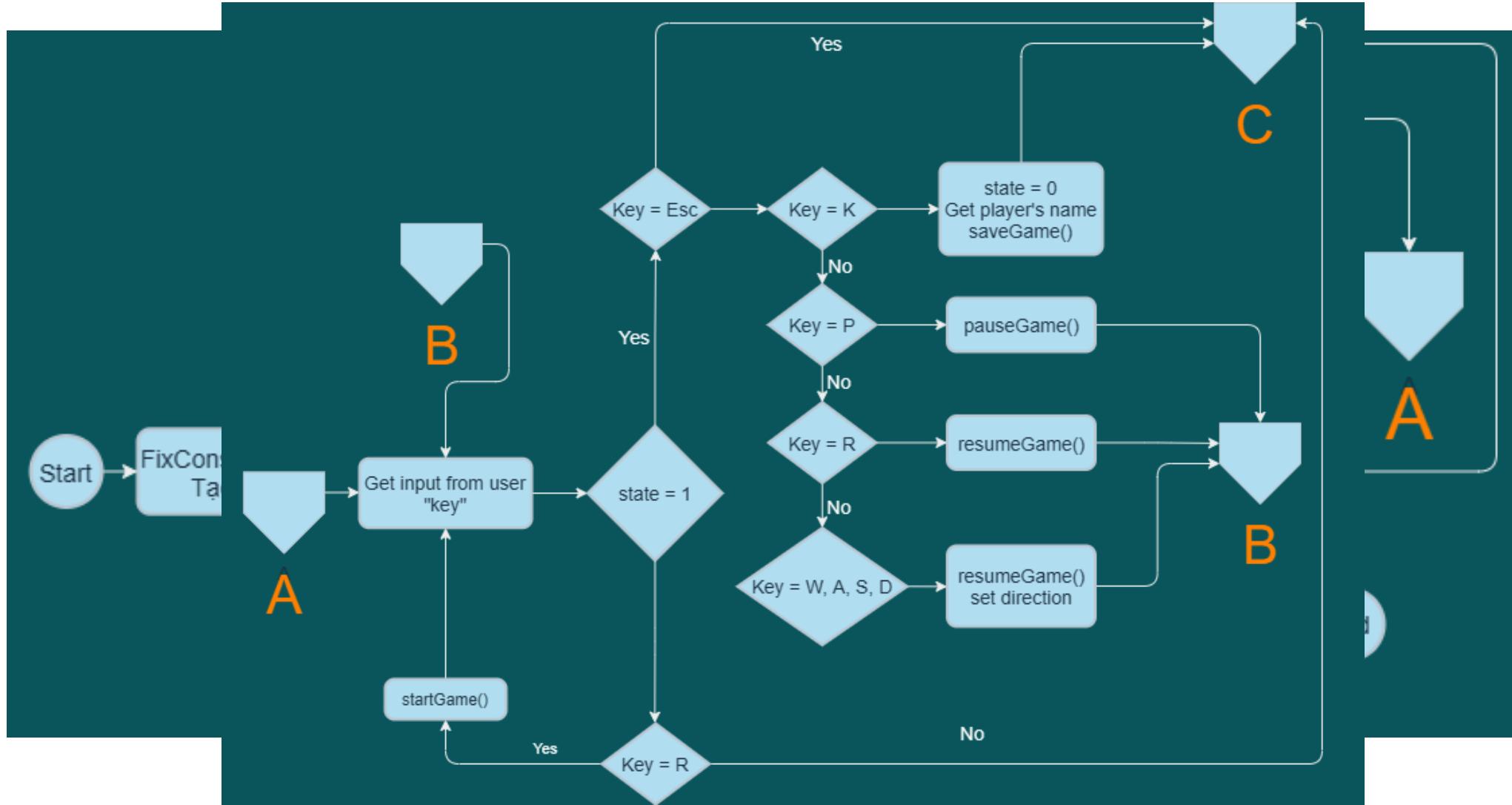


Phần 6 (CrossyRoadProject.cpp)

Xây dựng hàm main và file mã nguồn, sử dụng các hàm
và thư viện đã xây dựng trước đó

Xây dựng quản lý mọi nguồn tài nguyên, dữ liệu, sự kiện diễn ra trong suốt quá trình từ khi game được khởi động đến khi được đóng.

Phần 2



Demo video



Link source code

<http://tinyurl.com/Team15KTLT>

Tài liệu tham khảo

- Tài liệu đồ án từ giáo viên Trương Toàn Thịnh cung cấp.
- cplusplus: <https://wwwcplusplus.com/>
- cppreference: <https://en.cppreference.com/w/>
- Stackoverflow: <https://stackoverflow.com/>
- Geeksforgeeks: <http://geeksforgeeks.org>
- Dạy nhau học: <https://daynhauhoc.com/>
- <https://gist.github.com/toanmadrid/07aac3102973dd4d6c5cd2ea27c6dd15>
- Ascii Art: <https://www.asciiart.eu/religion/buddhism;>
<https://www.asciiart.eu/mythology/ghosts>

Thành viên nhóm 15

Nguyễn Minh An – 20120029

Võ Hoài An – 20120033

Vương Lê Đức Bình – 20120043

Trần Hoàng Phương Nam – 20120141

**CẢM ƠN
THẦY VÀ CÁC BẠN
ĐÃ QUAN TÂM THEO DÕI
PHẦN TRÌNH BÀY CỦA NHÓM 15**