**BÁO CÁO BÀI TẬP LỚN**

1. **Giới thiệu đề tài**

**Rắn Săn Mồi** – Trò chơi đã quen thuộc với tuổi thơ của chúng ta. Game được bắn đầu với 3 đốt rắn, nhiệm vụ của chúng ta là ăn mồi, sau khi ăn mồi rắn sẽ dài ra, và chúng ta sẽ chinh phục sao cho con rắn có thể dài nhất có thể.

1. **Ý tưởng xây dựng đề tài**

* Game được viết bằng ngôn ngữ C++.
* Game được xây dựng chế độ chơi ofline.
* Sử dụng chức năng điều khiển bằng chuột và bàn phím.

Game được bắt đầu với số đốt nhất định, gồm nhiều chế độ chơi, từ dễ đến khó, con rắn sẽ cho phép di chuyển bằng bàn phím nhiệm vụ người chơi là di chuyển rắn để ăn mồi với điều kiện không được cắn vào thân và không chạm tường, sau khi ăn mồi thì con rắn sẽ dài ra và làm cho việc di chuyển ngày càng khó điều đó có thể làm cho người chơi thích thú hơn để chinh phục, mồi rắn sẽ tự động được random ở vị trí bất kì trong không gian với tọa độ x và y.

1. **Viết chương trình**

* Tạo **class** thứ nhất lưu các tọa độ trục tung và trục hoành**:**

class Point {

public:

int x,y,x0,y0;

};

Khi đó:

* Một con rắn sẽ là 1 mảng các đối tượng tương ứng với các đốt của rắn. Tại mỗi đốt, cặp tọa độ (x,y) sẽ lưu vị trí đốt hiện tại và cặp tọa độ (x0,y0) sẽ lưu ví trí trước đó của đốt hiện tại để các đốt sau đó của con rắn có thể sử dụng.
* Mỗi đối tượng thức ăn sẽ là 1 đối tượng . Ta chỉ cần sử dụng cặp biến (x, y) để lưu 1 đối tượng thức ăn. Tại một thời điểm trên màn hình chỉ có 1 thức ăn, và thức ăn đó xuất hiện ở 1 vị trí ngẫu nhiên bất kỳ.
* Tạo **class** thứ hai lưu điểm và tên người chơi:

class HighScore {

public:

int score;

char name[30];

};

Thông tin điểm cao sẽ được ghi ra một file txt và lưu lại để so sánh với các người chơi sau đó. Nếu người chơi mới đạt điểm cao hơn trong danh sách đã lưu thì sẽ thay thế thành tích của người chơi cũ tương ứng bởi thành tích của người chơi mới.

* Xây dựng các hàm thực hiện các chức năng:
* Hàm **Void KiemTraMoi ()**: Hàm này có chức năng kiểm tra trùng lặp khi tạo mới đối tượng Food. Kiểm tra chắc chắn rằng tọa độ thức ăn không trùng với thân Rắn.

bool checkPoint (){

for (int i = 0;i < snakeLength;i++){

if (food.x == snake[i].x && food.y == snake[i].y)

return false;

}

return true;

}

#### Hàm void VeDoiTuong (int x,int y,int r): Hàm này có chức năng vẽ một đối tượng hình tròn lên màn hình đồ họa có tọa độ tâm (x,y) và có bán kính

Void VeDoiTuong (int x, int y, int r){

circle (x,y,r);

floodfill (x,y,getcolor());

}

#### Hàm void drawFood (): Hàm này gọi lại hàm drawPoint () và hàm checkPoint () để thực hiện nhiệm vụ hiển thị thức ăn lên màn hình đồ họa.

void drawFood (){

int x = getcolor();

int c = rand() % 14 + 1;

setcolor(c);

setfillstyle (SOLID\_FILL,c);

drawPoint(food.x, food.y,5);

setcolor(x);

setfillstyle (1,BACKGROUND);

}

#### Hàm void initGame ():

Vẽ khung của trò chơi

Khởi tạo tọa độ ban đầu cho Snake (Gồm 3 nút tròn) và Food

Khởi tạo hướng đi ban đầu

Vẽ màu backgound và các đối tượng hiển thị không thay đổi vị trí.

void initGame(){

setbkcolor (15);

cleardevice ();

setwindowtitle ("SNAKE-Playing....");

endGame = false;

setfillstyle (1,9);

bar (MINX,MINY,MAXX,MINY+5);

bar (MAXX,MINY,MAXX-5,MAXY);

bar (MAXX,MAXY,MINX,MAXY-5);

bar (MINX,MINY,MINX+5,MAXY);

setcolor (0);

int bk = getbkcolor();

setfillstyle (SOLID\_FILL,0);

int a[] = {15,350,30,350,30,345,50,355,30,365,30,360,15,360,15,350};

fillpoly (8,a);

settextstyle (4,0,3);

outtextxy (70,340,"PHAI");

int b[] = {15,410,30,400,30,405,45,405,45,415,30,415,30,420,15,410};

fillpoly (8,b);

outtextxy (70,400,"TRAI");

int c[] = {235,370,235,355,230,355,240,340,250,355,245,355,245,370,235,370};

fillpoly (8,c);outtextxy (265,340,"LEN");

int d[] = {235,400,235,415,230,415,240,430,250,415,245,415,245,400,235,400};

fillpoly (8,d);

outtextxy (265,400,"XUONG");

bar (50,450,170,480);setcolor (15);

settextstyle(1,0,1);setbkcolor(0);

outtextxy(60,455,"SPACE");setbkcolor(bk);

setcolor(0);settextstyle (4,0,3);outtextxy(200,450,"DUNG/TIEP TUC");

bar (50,500,170,530);setcolor (15);

settextstyle(1,0,1);

setbkcolor(0);outtextxy(80,505,"ESC");setbkcolor(bk);

setcolor(0);settextstyle (4,0,3);outtextxy(200,500,"KET THUC");

snake[0].x = 60;snake[0].y = 50;

snake[1].x = 50;snake[1].y = 50;

snake[2].x = 40;snake[2].y = 50;

snakeLength = 3;

setfillstyle (1,BACKGROUND);

bar (25,25,415,215);

direction.x = 10;direction.y = 0; // Khoi tao di theo huong trai qua phai

setcolor (2); srand ( time(NULL));

//Init food

do{

food.x = (rand() % (39) + 3)\*10;

food.y = (rand() % (19) + 3)\*10;

}while (checkPoint() == false);

}

#### Hàm void drawGame (): Hàm này gọi lại hai hàm drawSnake() và hàm drawFood() và thực hiện thêm công việc hiển thị điểm của người chơi lên màn hình.

void drawGame (){

drawFood();

. int x = getcolor ();

. setcolor (2);

sprintf(score\_str,"%d",snakeLength\*level\*10-level\*30);

settextstyle(3,0,3);

outtextxy(450,20,"SCORE:");

setcolor(5);

outtextxy(600,20,score\_str);

setcolor (x);

}

#### Hàm void classic():

Hàm này thực hiện chức năng chơi cổ điển – Người chơi có thể đi xuyên tường.  
Chức năng của hàm:

* Di chuyển Rắn theo điều khiển hướng đi
* Nếu Rắn đi ra ngoài mép tường – cho rắn xuất hiện lại ở phía bên kia
* Kiểm tra rắn ăn mồi : tăng chiều dài, điểm số và random lại tọa độ thức ăn
* Kiểm tra game kết thúc

void classic(){

for (int i = 0; i < snakeLength; i++) {

if (i == 0) {

snake[0].x0 = snake[0].x;snake[0].y0 = snake[0].y;

snake[0].x += direction.x;

snake[0].y += direction.y;

}else{

snake[i].x0 = snake[i].x;snake[i].y0 = snake[i].y;

snake[i].x = snake[i-1].x0;snake[i].y = snake[i-1].y0

}

if (snake[i].x >= MAXX) snake[i].x = MINX + 10;

if (snake[i].x <= MINX) snake[i].x = MAXX - 10;

if (snake[i].y >= MAXY) snake[i].y = MINY + 10;

if (snake[i].y <= MINY) snake[i].y = MAXY - 10;

if (i != 0 && (snake[0].x == snake[i].x && snake[0].y == snake[i].y)) endGame = true;

}

if (snake[0].x == food.x && snake[0].y == food.y){

snake[snakeLength].x = snake[snakeLength-1].x0;snake[snakeLength].y = snake[snakeLength-1].y0;

snakeLength++;

PlaySound(TEXT("eatFood.wav"), NULL, SND\_ASYNC);

srand ( time(NULL));

do{

food.x = (rand() % (39) + 3)\*10;

food.y = (rand() % (19) + 3)\*10;

}while (checkPoint() == false);

}

}

#### Hàm void modern(): Hàm này tương tự hàm classic() nhưng thay vào đó rắn không thể đi xuyên tường và đó là một trong những điều kiện bổ sung vào kiểm tra game kết thúc của hàm này.

void modern(){

for (int i = 0;i < snakeLength;i++){

if (i == 0){

snake[0].x0 = snake[0].x;snake[0].y0 = snake[0].y;

snake[0].x += direction.x;

snake[0].y += direction.y;

}

else {

snake[i].x0 = snake[i].x;snake[i].y0 = snake[i].y;

snake[i].x = snake[i-1].x0;snake[i].y = snake[i-1].y0;

}

if (snake[0].x < MINX+5 || snake[0].y < MINY+5 || snake[0].x > MAXX-5 || snake[0].y > MAXY - 5 )

endGame = true;

if (i != 0 && snake[0].x == snake[i].x && snake[0].y == snake[i].y) endGame = true;

}

if (snake[0].x == food.x && snake[0].y == food.y){

snake[snakeLength].x = snake[snakeLength-1].x0;snake[snakeLength].y = snake[snakeLength-1].y0;

snakeLength++;

PlaySound(TEXT("eatFood.wav"), NULL, SND\_ASYNC);

//Random again food if sanke ate food

srand ( time(NULL));

do{

food.x = (rand() % (39) + 3)\*10;

food.y = (rand() % (19) + 3)\*10;

}while (checkPoint() == false);

}

#### Hàm void changeDirecton (int x):

x là biến kiểu nguyên lưu giữ giá trị ASCII tương ứng của của phím ở kiểu int. Đây là hàm thực thi việc nhận lệnh điều khiển hướng đi của rắn từ bàn phím, cụ thể là các phím mũi tên điều hướng và phím ESC. Hàm sẽ nhận vào mã phím nhập từ bàn phím và thực hiện thay đổi hướng đi theo mã phím tương ứng. Nếu người dùng nhấn phím ESC – Game sẽ kết thúc.

void changeDirecton (int x){

switch(x){

case 72:

if (direction.y != DIRECTION) {

PlaySound(TEXT("beep.wav"), NULL, SND\_ASYNC);

direction.y = -DIRECTION; direction.x = 0;

}

break;

case 80:

if (direction.y != -DIRECTION) {

PlaySound(TEXT("beep.wav"), NULL, SND\_ASYNC);

direction.y = DIRECTION; direction.x = 0;

}

break;

case 77:

if (direction.x != -DIRECTION) {

PlaySound(TEXT("beep.wav"), NULL, SND\_ASYNC);

direction.x = DIRECTION; direction.y = 0;

}

break;

case 75:

if (direction.x != DIRECTION) {

PlaySound(TEXT("beep.wav"), NULL, SND\_ASYNC);

direction.x = -DIRECTION; direction.y = 0;

}

break;

case 27:

endGame = true;

break;

}

}

#### Hàm void mainLoop (void (\*gloop)());

Hàm này chính là hàm lặp cho phép game chạy và thực thi toàn bộ các hành động trong khi game chạy  
Tại đây, mình sử dụng đệ quy để hàm gọi lại chính nó. Tham số truyền vào là một trong hai hàm classic() hoặc hàm modern() tùy theo lựa chọn của người chơi. Hàm này còn có chức năng Pause/Resum game khi đang chơi nếu người dùng ấn phím SPACE.

void mainLoop (void (\*xxx)()){

int x = getcolor();

if (!kbhit()){

xxx();

}

else{

char ch;

ch = getch();

if (ch == -32) ch = getch();

if ( ch == 32){

setcolor(0);outtextxy (100,100,"DUNG GAME");setcolor(x);

ch = getch();

while (ch != 32){

ch = getch();

delay(0);

}

setcolor (BACKGROUND);

outtextxy (100,100,"DUNG GAME");

setcolor (x);

delay(1000);

}

changeDirecton(ch);

mainLoop (xxx);

}

}

#### Hàm void initScore();

Hàm này kiểm tra file lưu điểm cao có rỗng hay không bằng cách gọi hàm isEmpty(). Nếu file rỗng thì hàm sẽ khởi tạo 5 giá trị mặc định có tên PLAYER và điểm cao là 0. Nếu hàm có chứa thông tin hàm sẽ đọc thông tin điểm cao lưu vào mảng kiểu HighScore.

#### Hàm void showHighScore();

Hàm này có chức năng đọc thông tin của file điểm cao và xuất ra thông tin lên màn hình đồ họa khi được gọi.

void showHighScore(){

FILE \*f;

f = fopen("highscore.txt", "r");

char ch[20];

settextstyle (1,0,5);

setcolor (15);

outtextxy (210,50,"DIEM CAO");

settextstyle (1,0,4);

int y = 150,count = 0;

while (!feof(f)){

if (count == 10) break;

count++;

fscanf(f, "%s", ch);

if (count%2 == 1){

setcolor (10);

outtextxy (180,y,ch);

y+=50;

}else{

setcolor (12);

outtextxy (500,y-50,ch);

}

}

fclose(f);

}

#### void getHighScore ();

Hàm này có chức năng ghi lại thông tin điểm cao của người chơi + lưu vào file

void getHighScore (){

FILE \*f;

f = fopen("highscore.txt", "w");

for (int i = 0;i < 5;i++){

fputs(highscore[i].name,f);

fputs(" ",f);

fprintf(f,"%d",highscore[i].score);

fputs("\n",f);

}

fclose(f);

}

* Hàm **checkHighScore (int \_score):**

Hàm này thực hiện nhiệm vụ nhận điểm cao của người chơi, lấy thông tin. tên người chơi nếu người chơi đó đạt thành tích cao. Hàm sẽ lưu giá trị vào mảng kiểu HighScore và gọi lại hàm getHighScore() để lưu lại thông tin.