Đại Học Quốc Gia Thành Phố Hồ Chí Minh

Trường Đại Học Khoa Học Tự Nhiên TP. Hồ Chí Minh

Khoa Công Nghệ Thông Tin

BÁO CÁO ĐỒ ÁN CUỐI KÌ

Tên đồ án: **Rắn săn mồi**

Họ và tên sinh viên: **Khưu Thùy Kỳ**

Giáo viên hướng dẫn: **Trương Toàn Thịnh**

MỤC LỤC

GIẢI QUYẾT CÁC YÊU CẦU ĐỒ ÁN:

1. *Xử lí đầu rắn chạm vào thân Rắn:*

Hàm *IsValid()* được tái sử dụng để kiểm tra phần tử đầu rắn có trùng với thân rắn hay không. Kết hợp giá trị trả về của *IsValid()* vào các hàm duy chuyển để gọi hàm *ProcessDead()*

Hàm *IsValid()* sẽ trả về giá trị *false* nếu vị trí truyền vào trùng với một vị trí nào đó của rắn hiện tại. Do đó *!IsValid()* sẽ trả về *true* và gọi hàm ProcessDead() khi đầu rắn chạm vào bất cứ đâu trên thân rắn.

```
473     void MoveLeft()
474     {
475         // Xử lí chết khi rắn chạm vào bản thân hoặc vào tường
476         if ((snake[SIZE_SNAKE - 1].x - 1 == 0) || (!IsValid(snake[SIZE_SNAKE - 1].x - 1, snake[SIZE_SNAKE - 1].y)))
477         {
478             ProcessDead();
479 
480         }
481         else
482         {
483             if (snake[SIZE_SNAKE - 1].x - 1 == food[FOOD_INDEX].x && snake[SIZE_SNAKE - 1].y == food[FOOD_INDEX].y)
484             {
485                 Eat();
486                 FOOD = 0;
487             }
488             if (snake[SIZE_SNAKE - 1].x - 1 != 0)
489             {
490                 for (int i = 0; i < SIZE_SNAKE - 1; i++)
491                 {
492                     snake[i].x = snake[i + 1].x;
493                     snake[i].y = snake[i + 1].y;
494                 }
495                 snake[SIZE_SNAKE - 1].x--;
496             }
497         }
498     }
```

2. ***Xử lí lưu trò chơi và tải trò chơi đã lưu:***

2.1 Lưu Game:

Hàm *SaveGame()* sẽ được gọi. Chương trình sẽ tạo một thư mục theo đường dẫn mặc định bằng hàm *CreateDirectoryA()* thuộc thư việc <*windows.h*> . Sau đó người dùng được yêu cầu nhập vào tên file. Tên file sẽ được ghép vào đường dẫn và được mở để ghi các giá trị cần thiết của game. Quá trình nhập sẽ được lặp đến khi nào người dùng muốn dừng hoặc đã tìm được file. Sau đó chương trình sẽ chờ người dùng nhập vào phím bất kì bằng hàm *_kbhit()*

```
263  void SaveGame(HANDLE handle_t1)
264  {
265      ChangeColor(15);
266      string str, save_file = file;
267      PauseGame(handle_t1);
268      CreateDirectoryA(file.c_str(), NULL);           //Tạo thư mục lưu game
269
270      //Nhập tên file
271      GotoXY(WIDTH_CONSOLE + ((SizeConsole.X - WIDTH_CONSOLE) / 2 - 14), HEIGH_CONSOLE / 2);
272      printf("Nhap ten file de luu du lieu");
273      GotoXY(WIDTH_CONSOLE + ((SizeConsole.X - WIDTH_CONSOLE) / 2 - 9), HEIGH_CONSOLE / 2 + 1);
274      printf("(Toi da 10  ky tu)");
275      GotoXY(WIDTH_CONSOLE + ((SizeConsole.X - WIDTH_CONSOLE) / 2 - 5), HEIGH_CONSOLE / 2 + 2);
276      printf(">>");
277      cin >> str;
278      save_file.append(str);
279      ofstream save(save_file);
280      if (save)
281      {
282          save << CHAR_LOCK << endl;
283          save << MOVING << endl;
284          save << SPEED << endl;
285          save << FOOD_INDEX << endl;
286          save << WIDTH_CONSOLE << endl;
287          save << HEIGH_CONSOLE << endl;
288          save << SIZE_SNAKE << endl;
289          save << GATE << endl;
290          for (int i = 0; i < SIZE_SNAKE; i++)
291          {
292              save << snake[i];
293          }
294          for (int i = FOOD_INDEX; i < MAX_SIZE_FOOD; i++)
295          {
296              save << food[i];
297          }
298          GotoXY(WIDTH_CONSOLE + ((SizeConsole.X - WIDTH_CONSOLE) / 2 - 24), HEIGH_CONSOLE / 2 + 3);
299          printf("Da luu thanh cong. Chon phim bat ky de tiep tuc");
300          _getch();
301      }
302      else
303      {
304          GotoXY(WIDTH_CONSOLE + ((SizeConsole.X - WIDTH_CONSOLE) / 2 - 22), HEIGH_CONSOLE / 2 + 3);
305          printf("Luu khong thanh cong. Hay thu ten file khac");
306          GotoXY(WIDTH_CONSOLE + ((SizeConsole.X - WIDTH_CONSOLE) / 2 - 14), HEIGH_CONSOLE / 2 + 4);
307          printf("Chon phim bat ky de tiep tuc");
308          _kbhit();
309      }
310      StartGame();
311      if (GATE != 0)
312          ShowTheGate();
313      ResumeThread(handle_t1);
314      FOOD = 0;
315      flag = false;
316  }
```

Để xóa những gì đã hiện ra trên màn hình và tiếp tục chơi game. Ta gọi hàm StartGame có chức năng xóa màn hình mà vẽ lại khung. Sau đó mở lại cổng nếu lúc save đã hết food. Gán biến flag là false để chương trình nhận phím di chuyển mới.

2.2 Load Game:

Hàm *LoadGame()* sẽ được gọi và thực hiện tương tự như hàm *SaveGame()* nhưng không tạo thư mục như *SaveGame()* mà chỉ yêu cầu người dùng nhập vào tên file, mở file và đọc các giá trị từ file.

```
637  void LoadGame(string save_file, int &exitcode)
638  {
639      ChangeColor(15);
640      string str;
641
642      //Nhập tên file để tải dữ liệu game đã lưu
643      char exit = 'Y';
644      do
645      {
646          printf("\nNhap ten file: ");
647          cin >> str;
648          save_file.append(str);
649          ifstream save(save_file);
650          str.erase();
651          if (!save)
652          {
653              printf("Khong tim thay file! Muon nhap lai ten file khong? (Co chon Y, Khong chon phim bat ky)\n");
654              exit = toupper(_getch());
655
656              if (exit == 'Y')
657                  continue;
658              else
659                  break;
660          }
661          else
662          {
663              save >> CHAR_LOCK;
664              save >> MOVING;
665              save >> SPEED;
666              save >> FOOD_INDEX;
667              save >> WIDTH_CONSOLE;
668              save >> HEIGH_CONSOLE;
668              save >> HEIGH_CONSOLE;
669              save >> SIZE_SNAKE;
670              save >> GATE;
671              for (int i = 0; i<SIZE_SNAKE; i++)
672                  save >> snake[i];
673              for (int i = FOOD_INDEX; i< MAX_SIZE_FOOD; i++)
674                  save >> food[i];
675              FOOD = 0;
676              exitcode = 0;
677              break;
678          }
679      } while (1);
680      if (exit != 'Y')
681      {
682          printf("Ban muon choi game moi khong? (Co chon Y, Khong chon phim bat ky)\n");
683          exit = toupper(_getch());
684          if (exit == 'Y')
685          {
686              ResetData();
687              exitcode = 0;    // Tiếp tục chương  trình
688          }
689          else
690              exitcode = 1;    // exitcode 1 đánh dấu thoát khỏi chương trình
691          return;
692      }
693  }
```

Nếu người dùng nhập sai tên file. Chương trình sẽ hỏi người dùng có muốn chơi game mới không. Và lựa chọn chơi game mới hay thoát game sẽ trả về các exitcode khác nhau và gọi các hàm tương ứng.

3. ***Xử lí giữ nguyên độ dài rắn:***

Hàm *LevelUp()* sẽ được gọi để xử lí lên cấp sau khi rắn ăn hết food và qua cổng. Khi đó rắn sẽ được tạo lại và chờ người dụng bấm phím bất kì để tiếp tục di chuyển.

```
416  void Levelup()
417  {
418
419      if (SPEED + SPEED_INCREASE >= MAX_SPEED)    // Khi lên hết cấp, trở về độ dài và tốc độ mặc định
420      {
421          SIZE_SNAKE = 8;
422          SPEED = DEFAULT_SPEED;
423      }
424      else
425          SPEED += SPEED_INCREASE;                //Chưa hết cấp thì tăng tốc mỗi lần lên cấp
426
427      //Mỗi lần lên cấp, rắn xuất hiện trên màn hình và chờ nhấn phím để duy chuyển
428      for (int i = 0; i < SIZE_SNAKE; i++)
429      {
430          snake[i] = { i + 3,15 };
431      }
432      snake[SIZE_SNAKE] = { 0,0 };
433      GATE = 0;
434      GotoXY(snake[0].x, snake[0].y);
435      DrawSnakeAndFood("O");
436      FOOD = 1;
437      _kbhit();
438  }
```

Độ dài rắn được giữ nguyên khi lên cấp, chỉ tăng tốc độ khi lên cấp. Khi đã đạt cấp tối đa rắn sẽ trở về chiều dài ban đầu và tốc độ ban đầu.

4. ***Xử lí ăn xong food ở một cấp:***

Sau khi ăn xong hết food ở một cấp. Hàm *ShowTheGate()* sẽ được gọi để mở cổng. Và người dùng sẽ điều khiển rắn đi đến cổng. Khi rắn chạm cổng. Thay vì chết vì chạm cổng, một điều kiện sẽ được thêm vào hàm *MoveRight()* để đảm bảo rắn không chết khi đến tọa độ biên. Hàm *MoveThroughGate()* sẽ được gọi để rắn chuyển động qua cổng và biến mất từ từ. Hàm *CloseTheGate()* được gọi để đóng cổng. Sau đó hàm *LevelUp()* được gọi để xử lí lên cấp và tiếp tục trò chơi.

```
183     //Mở cổng
184     void ShowTheGate()
185     {
186         ChangeColor(11);
187         CONSOLE_SCREEN_BUFFER_INFO csbInfo;
188         GetConsoleScreenBufferInfo(GetStdHandle(STD_OUTPUT_HANDLE), &csbInfo);  //Lấy vị trí rắn hiện tại
189         GotoXY(WIDTH_CONSOLE, HEIGH_CONSOLE / 2);                               //Mở cổng
190         printf("%c", ' ');
191         GotoXY(WIDTH_CONSOLE, HEIGH_CONSOLE / 2 - 1);
192         printf("%c%c", 223, 223);
193         GotoXY(WIDTH_CONSOLE, HEIGH_CONSOLE / 2 + 1);
194         printf("%c%c", 220, 220);
195         GotoXY(csbInfo.dwCursorPosition.X, csbInfo.dwCursorPosition.Y);         //Phục hồi con trỏ tại vị trí rắn
196         GATE = 1;
197     }
198
199
200     //Đóng cổng
201     void CloseTheGate()
202     {
203         ChangeColor(15);
204         GotoXY(WIDTH_CONSOLE, HEIGH_CONSOLE / 2);
205         printf("%c ", 221);
206         GotoXY(WIDTH_CONSOLE, HEIGH_CONSOLE / 2 - 1);
207         printf("%c ", 221);
208         GotoXY(WIDTH_CONSOLE, HEIGH_CONSOLE / 2 + 1);
209         printf("%c ", 221);
210         GATE = 2;
211     }

215     void MoveThroughGate()
216     {
217                                                 //Khi rắn chạy qua cổng sẽ biến mất dần
218         int tempSN = SIZE_SNAKE;
219         while (tempSN > 0)
220         {
221             for (int i = 0; i < SIZE_SNAKE - 1; i++)
222             {
223                 snake[i].x = snake[i + 1].x;
224                 snake[i].y = snake[i + 1].y;
225             }
226             snake[SIZE_SNAKE - 1].x++;
227             for (int i = 0; i < tempSN; i++)
228             {
229                 ChangeColor(15);
230                 GotoXY(snake[i].x, snake[i].y);
231                 printf("%c", BODY[i % 8]);
232             }
233             Sleep(1000 / SPEED);
234             for (int i = 0; i < tempSN; i++)
235             {
236                 ChangeColor(0);
237                 GotoXY(snake[i].x, snake[i].y);
238                 printf(" ");
239             }
240             tempSN--;
241         }
242     }
243
```

Cổng được tạo ra mặc định ở giữa cạnh bên phải của khung game. Mở và đóng cổng chỉ đơn thuần là xóa và in các ký tự ở các tọa độ cụ thể.

5. ***Xử lí hiệu ứng va chạm:***

Khi rắn va chạm vào tường hoặc thân, hàm *ProcessDead()* sẽ được gọi. Rắn sẽ nhấp nháy 5 lần trước khi hiện ra thông báo chết và chờ người dùng chọn tiếp tục chơi hay dừng chơi.

```
387  void ProcessDead()
388  {
389      // Khi chết rắn nhấp nháy 5 lần
390      for (int i = 0; i < 5; i++)
391      {
392          //ChangeColor(112);
393          DrawSnakeAndFood(" ");
394          Sleep(250);
395          DrawSnakeAndFood("O");
396          Sleep(250);
397      }
398
399      // Và xuất hiện thông báo chết, hỏi tiếp tục hay thoát game
400      ChangeColor(12);
401      STATE = 0;
402      FOOD = 0;
403      GAME_EXIT = true;
404      int x = WIDTH_CONSOLE + ((SizeConsole.X - WIDTH_CONSOLE) / 2);
405      int y = HEIGH_CONSOLE / 2;
406      GotoXY(x - 2, y - 2);
407      printf("DEAD");
408      GotoXY(x - 12, y);
409      printf("Tro lai menu nhan phim Y");
410      GotoXY(x - 11, y + 2);
411      printf("De thoat nhan phim ESC");
412  }
```

6. ***Xử lí màn hình chính:***

Màn hình menu được thể hiện theo cách. Người dùng có thể chọn bằng cách di chuyển lên xuống giữa các mục và enter để chọn. Hoặc người dùng có thể chọn bằng cách phím chữ cái.

Hàm *ShowMenu()* sẽ đảm nhiệm chức năng hiển thị menu xử lí các lựa chọn bằng hàng loạt các cấu trúc *switch*.

```
845  void ShowMenu()
846  {
847      GAME_EXIT = false;
848      FixConsoleWindow();         //Cố định kích thước màn hình
849      Nocursortype();             //Ẩn con trỏ
850      CONSOLE_SCREEN_BUFFER_INFO csbInfo;
851      GetConsoleScreenBufferInfo(GetStdHandle(STD_OUTPUT_HANDLE), &csbInfo);
852      SizeConsole.X = csbInfo.srWindow.Right - csbInfo.srWindow.Left + 1;
853      SizeConsole.Y = csbInfo.srWindow.Bottom - csbInfo.srWindow.Top + 1;
854      type = 1;
855      MainMenu();
856      int sub = 0;            //biến đại diện cho menu con.
857      int exitcode = 0;
858      while (exitcode == 0)   //chạy đến khi nào có lỗi hoặc muốn dừng game
859      {
860          COORD Pos;
861          char temp = ' ';
862          Pos = { SizeConsole.X / 2 - 14, SizeConsole.Y / 2 - 1 };
863
864          while (1)           //Lựa chọn trong menu
865          {
866              if (kbhit())
867                  temp = toupper(getch());
868              switch (sub)
869              {
870                      case 0:     //Menu chính
871                      {
872                          switch (temp)   //Di chuyển trong menu
873                          {
874                              case 'W':   //Di chuyển lên
875                              { ... }
892                              case 'S':   //Di chuyển xuống
893                              { ... }
910                              case 'N':   //Chơi game mới
911                              { ... }
921                              case 'T':   //Tải game đã lưu
922                              { ... }
933                              case 'C':   //Cài đặt cửa sổ
934                              { ... }
939                              case 27:    //Thoát game
940                              { ... }
943                              case 13:    //xử lí các lựa chọn khi nhấn enter
944                              { ... }
987                          }
988                          break;
989                      }
990                      case 1:     //Menu Phụ
991                      {
992                          switch (temp)
993                          {
994                              case 'W':   //Di chuyển lên
995                              { ... }
1012                             case 'S':   //Di chuyển xuống
1013                             { ... }
1030                             case 13:    //Xử lí các lựa chọn khi nhấn Enter
1031                             { ... }
1070                         }
1071                         break;
1072                     }
1073             }
1074             temp = ' ';
1075         }
1076
1077     }
1078 }
```

Khi lựa chọn Chơi game, hàm *PlayGame()* sẽ được gọi và tạo thread bắt đầu chơi game.

Khi lựa chọn tải game, hàm *LoadGameMenu()* sẽ được gọi. Hàm *LoadGameMenu()* sẽ gọi hàm *LoadGame()* để thực hiện tải game đã lưu.

```
798  void LoadGameMenu(int &exitcode)
799  {
800      string str, save_file = file;
801      ChangeColor(15);
802      GotoXY(SizeConsole.X / 2 + 14, SizeConsole.Y / 2 );
803      LoadGame(save_file, exitcode);
804  }
```

Khi lựa chọn Cài đặt, hàm sẽ gọi *SettingMenu()* để thiết lập màn hình. Hàm *SettingMenu()* sẽ gọi hàm *NormalScreenMode()* hoặc *FullScreenMode()* tùy vào lựa chọn của người dùng.

```
808  void SettingMenu()
809  {
810      system("cls");
811      if (type == 1)
812          ChangeColor(6);
813      else
814          ChangeColor(14);
815      GotoXY(SizeConsole.X / 2 - 10, SizeConsole.Y / 2 - 3);
816      printf("Che do cua so\n");
817      if (type == 1)
818          ChangeColor(14);
819      else
820          ChangeColor(6);
821      GotoXY(SizeConsole.X / 2 - 10, SizeConsole.Y / 2 - 2);
822      printf("Che do day man hinh\n");
823      GotoXY(SizeConsole.X / 2 - 10, SizeConsole.Y / 2 - 1);
824      ChangeColor(14);
825      printf("Tro ve\n");
826      ChangeColor(14);
827      if (type == 1)
828      {
829          GotoXY(SizeConsole.X / 2 - 14, SizeConsole.Y / 2 - 2);
830          printf(">");
831          GotoXY(SizeConsole.X / 2 + 14, SizeConsole.Y / 2 - 2);
832          printf("<");
833      }
834      else
835      {
836          GotoXY(SizeConsole.X / 2 - 14, SizeConsole.Y / 2 - 3);
837          printf(">");
838          GotoXY(SizeConsole.X / 2 + 14, SizeConsole.Y / 2 - 3);
839          printf("<");
840      }
841  }

613  void FullScreenMode()
614  {
615      HANDLE handle = GetStdHandle(STD_OUTPUT_HANDLE);
616      SetConsoleDisplayMode(handle, CONSOLE_FULLSCREEN_MODE, 0);
617      CONSOLE_SCREEN_BUFFER_INFO csbInfo;
618      GetConsoleScreenBufferInfo(handle, &csbInfo);
619      SizeConsole.X = csbInfo.srWindow.Right - csbInfo.srWindow.Left + 1;
620      SizeConsole.Y = csbInfo.srWindow.Bottom - csbInfo.srWindow.Top + 1;
621  }
622
623
624  //Chế độ cửa sổ
625  void NormalScreenMode()
626  {
627      HANDLE handle = GetStdHandle(STD_OUTPUT_HANDLE);
628      SetConsoleDisplayMode(handle, CONSOLE_WINDOWED_MODE, 0);
629      CONSOLE_SCREEN_BUFFER_INFO csbInfo;
630      GetConsoleScreenBufferInfo(handle, &csbInfo);
631      SizeConsole.X = csbInfo.srWindow.Right - csbInfo.srWindow.Left + 1;
632      SizeConsole.Y = csbInfo.srWindow.Bottom - csbInfo.srWindow.Top + 1;
633  }
```

7. ***Các hàm khác:***

Hàm đổi màu *ChangeColor(),* Sử dụng hàm *SetConsoleTextAttribute()* có trong thư viện <*windows.h*> để đổi màu chữ.

```
79  void ChangeColor(int i)
80  {
81      SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), i);    //Đổi màu ký tự theo mã i
82  }
```

Hàm *FixConsoleWindow()* được dùng để cố định không cho người dùng thay đổi kích thước màn hình bằng cách hàm của thư viện <*windows.h*>

```
59  void FixConsoleWindow()
60  {
61      HWND consoleWindow = GetConsoleWindow();                    //Lấy handle console
62      LONG style = GetWindowLong(consoleWindow, GWL_STYLE);       //Lấy các thuộc tính hiện tại của console
63      style = style & ~(WS_MAXIMIZEBOX) & ~(WS_THICKFRAME);       //Bỏ ô phóng to và bỏ viền cửa sổ console
64      SetWindowLong(consoleWindow, GWL_STYLE, style);             //Thiết lập thuộc tính đã điều chỉnh
65  }
```

Hàm *NoCursorType()* được dùng để ẩn con trỏ.

```
69  void Nocursortype()
70  {
71      CONSOLE_CURSOR_INFO Info;
72      Info.bVisible = FALSE;  //Ẩn con trỏ
73      Info.dwSize = 20;
74      SetConsoleCursorInfo(GetStdHandle(STD_OUTPUT_HANDLE), &Info);   //Thiết lập thuộc tính cho con trỏ theo biến info
75  }
```

Và các hàm quá tải toán tử cho việc nhập xuất dễ dàng hơn

```
42  //overload toán tử >> để nhập mảng snake và food
43  istream& operator >> (istream& inDEV, POINT &p)
44  {
45      inDEV >> p.x >> p.y;
46      return inDEV;
47  }
48
49
50  //overload toán tử << để xuất mảng snake và food
51  ostream& operator << (ostream& outDEV, POINT p)
52  {
53      outDEV << p.x << " " << p.y << endl;
54      return outDEV;
55  }
```

Nguồn tham khảo:

Website: https://docs.microsoft.com/en-us/windows/console/console-functions

Tài liệu: “hướng dẫn đồ án rắn săn mồi” Bộ môn Công Nghệ Phần Mềm, Môn Kỹ Thuật Lập Trình, Trường Đại Học Khoa Học Tự Nhiên TpHCM.