BỘ CÔNG THƯƠNG

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP THỰC PHẨM TP. HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

---------------------------



**ĐỒ ÁN KẾT THÚC MÔN HỌC**

**SINH VIÊN THỰC HIỆN**

2001216212 - Nguyễn Duy Tiến (NT)

2001210222 - Nguyễn Thanh Tiến

2001210933 - Trần Anh Khoa

**GVHD:** Trần Đình Toàn

**BÁO CÁO TIỂU LUẬN**

Môn học: Trí tuệ nhân tạo

*TP. HỒ CHÍ MINH, tháng 12 năm 2023*

**LỊCH LÀM VIỆC NHÓM TRONG HÀNG TUẦN**

|  |  |  |
| --- | --- | --- |
| Ngày nộp báo cáo | Công việc | Tên bài báo cáo |
| 24/09/2023 | Nhóm tiến hành nộp bài cáo đầu tiên | Thuật toán luyện kim |
| 02/10/2023 | Nhóm tiến hành nộp bài cáo thứ hai | Thuật toán vương hạo |
| 29/10/2023 | Nhóm tiến hành nộp bài cáo thứ ba | Thuật toán K-Nearest Neighbors |
| 27/11/2023 | Nhóm tiến hành nộp bài cáo thứ tư | Phương pháp Entropy |
| 03/12/2023 | Nhóm tiến hành nộp bài báo cáo kết thúc môn học | Tìm đường về nhà |

**CÔNG VIỆC MỖI THÀNH VIÊN TRONG NHÓM**

|  |  |  |  |
| --- | --- | --- | --- |
| MSSV | Họ tên | Phân công công việc | Mức độ hoàn thành |
| 2001216212 | Nguyễn Duy Tiến | Tìm hiểu thuật toán, lý thuyết thuật toán, tìm hiểu code, xây dựng powerpoint, demo | 100% |
| 2001210222 | Nguyễn Thanh Tiến | Tìm hiểu thuật toán, lý thuyết thuật toán, tìm hiểu code, xây dựng powerpoint, demo | 100% |
| 2001210933 | Trần Anh Khoa | Tìm hiểu thuật toán, ví dụ minh họa, tìm hiểu code, xây dựng word | 100% |

MỤC LỤC

[GIỚI THIỆU 3](#_Toc152546704)

[**a.** **Phạm vi của đồ án** 3](#_Toc152546705)

[**b.** **Mục tiêu** 3](#_Toc152546706)

[**c.** **Sự cần thiết và lí do chọn đề tài** 3](#_Toc152546707)

[PHÂN TÍCH ĐỀ TÀI 4](#_Toc152546708)

[**a.** **Phân tích yêu cầu** 4](#_Toc152546709)

[**b.** **Yêu cầu chức năng** 4](#_Toc152546710)

[THIẾT KẾ 5](#_Toc152546711)

[**a.** **Đề xuất sử dụng thuật toán** 5](#_Toc152546712)

[**b.** **Cách thức giải quyết bài toán** 5](#_Toc152546713)

[CÀI ĐẶT ỨNG DỤNG BÀI TOÁN 6](#_Toc152546714)

[KẾT LUẬN VÀ ĐỊNH HƯỚNG PHÁT TRIỂN 27](#_Toc152546715)

[TÀI LIỆU THAM KHẢO 28](#_Toc152546716)

[PHỤ LỤC 29](#_Toc152546717)

# GIỚI THIỆU

Sau khi kết thúc buổi báo cáo đề tài về các thuật toán, phương pháp đã bốc thăm của các nhóm ở buổi học đầu tiên. Tiếp theo các nhóm sẽ thực hiện báo cáo đồ án kết thúc môn học.

1. **Phạm vi của đồ án**

* Danh sách đồ án gồm có 15 đề tài, tương đương với 2 nhóm cùng 1 đề tài. Sử dụng các thuật toán, phương pháp trong học phần Trí tuệ nhân tạo đã học để thực hiện đề tài.

1. **Mục tiêu**

* Code đầy đủ thuật toán sử dụng phải rõ ràng tường minh, các thuật toán liên quan học máy phải có huấn luyện (training) mô hình (model).
* Thiết kế giao diện và vận hành ứng dụng.
* Viết và trình bày báo cáo.

1. **Sự cần thiết và lí do chọn đề tài**

* Sự cần thiết mà đề tài mang lại là kỹ năng tư duy, kỹ năng hướng dẫn đường, giải trí và thú vị,…
* Việc chọn đề tài "game tìm đường về nhà" vì đề tài khá gần gũi với nhóm, không gây bỡ ngỡ khi đọc đề tài, có thể hình dung công việc cần làm khi đọc đề tài. Điểm mạnh ở đề tài là không chỉ hỗ trợ trí tuệ và logic mà còn mang tính giúp người chơi hiểu rõ hơn về quy trình và quan niệm trong việc tìm đường đi trong thực tế.

# PHÂN TÍCH ĐỀ TÀI

1. **Phân tích yêu cầu**

* Sử dụng các thuật toán, phương pháp đã học trong học phần Trí tuệ nhân tạo
* Gồm có 11 thuật toán, phương pháp
* Số node đồ thị lớn hơn 20
* Tìm được đường đi từ A đến B

1. **Yêu cầu chức năng**

* Điểm bắt đầu và điểm kết thúc
* Chọn loại vật cản
* Hiển thị quá trình tìm đường
* Làm chậm quá trình vẽ đường đi
* Thời gian tìm đường
* Làm mới
* Xóa chướng ngại vật
* Xóa toàn bộ chướng ngại vật
* Thông báo chi phí tìm đường

# THIẾT KẾ

1. **Đề xuất sử dụng thuật toán**

* Tích hợp một thuật toán tìm đường, chẳng hạn như thuật toán A\* hoặc Dijkstra (đảm bảo tìm đường đi ngắn nhất) và Breadth First Search thuật toán tìm kiếm theo chiều rộng (thường sử dụng hàm heuristic để ước lượng khoảng cách đến mục tiêu) để cho phép người chơi và NPC tìm đường từ vị trí hiện tại đến nhà một cách hiệu quả.

1. **Cách thức giải quyết bài toán**

* Chế độ chơi
* Cung cấp chế độ chơi cho người chơi, nơi họ có thể điều khiển nhân vật để tìm đường về nhà.
* Cung cấp chế độ chơi tự động, trong đó NPC (non-player characters) sẽ tự động tìm đường về nhà (Nhóm đang thực hiện chức năng này).
* Hiển thị đường đi
* Hiển thị đường đi trên màn hình để người chơi có thể theo dõi.
* Đồng thời, cung cấp khả năng tắt/bật chức năng hiển thị đường đi để tăng thách thức.
* Thách thức tăng dần
* Tăng độ khó qua các cấp độ hoặc giai đoạn trong game, chẳng hạn như thêm vật cản, thay đổi địa hình, tăng hoặc giảm tốc độ di chuyển của NPC.
* Giao Diện Người Dùng
* Xây dựng giao diện người dùng thân thiện và dễ sử dụng.
* Hiển thị thông tin như số bước di chuyển và thời gian hoàn thành chặn đường.
* Thông báo
* Hiện thông báo khi NPC đã về đến điểm kết thúc bằng chí phí đường đi.
* Hoặc chưa hoàn thành nếu như bị vật cản, cản lại.

# CÀI ĐẶT ỨNG DỤNG BÀI TOÁN

**Form1.cs**

**A screen shot of a computer

Description automatically generated**

Khai báo các biến sẽ sử dụng sắp tới.

Constructor của lớp Form1 được gọi khi tạo một đối tượng Form1 mới. Trong constructor này, các biến thành viên được khởi tạo và gán giá trị mặc định.

**A computer screen with colorful text

Description automatically generated**

Trong phương thức này:

* Cập nhật nội dung của hai đối tượng ShowProcess và Slow là các chuỗi văn bản hiển thị trên giao diện.
* Gán giá trị màu nền của bàn cờ cho biến MauBanCo và thuộc tính BackColor của đối tượng BanCo.
* Gọi phương thức KhoiTaoToaDo của đối tượng main để khởi tạo các tọa độ trên bàn cờ.

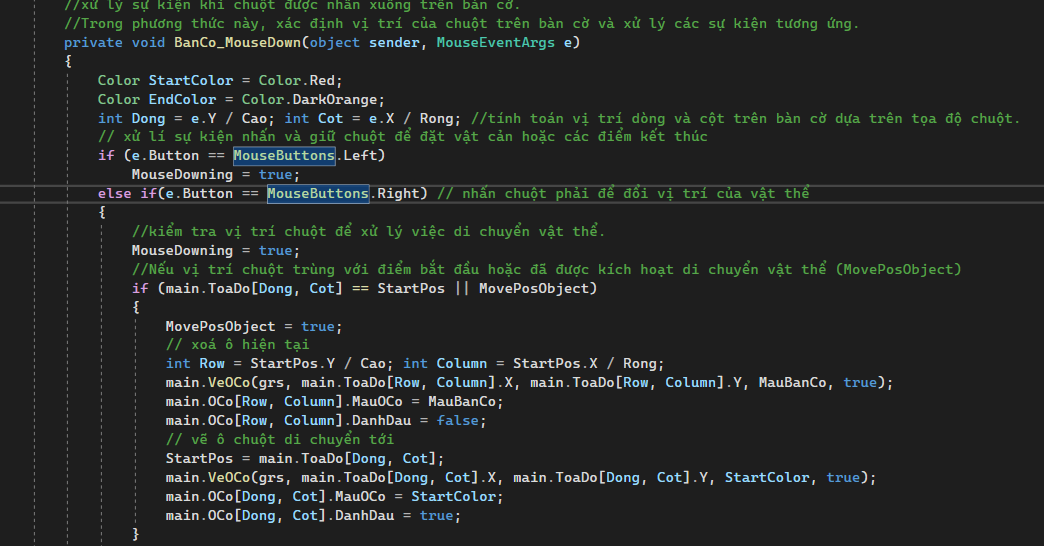
Biến hằng số Rong và Cao có giá trị 15. Chúng được sử dụng trong việc khởi tạo kích thước của các ô cờ trên bàn cờ.

Biến Dong và Cot là các biến nguyên dùng để lưu trữ số hàng và số cột trên bàn cờ. Ban đầu, chúng được gán giá trị là 45 và 77.

Phương thức BanCo\_Paint được gọi khi cần vẽ lại bàn cờ. Trong phương thức này:

Gọi phương thức VeLaiOCo của đối tượng main để vẽ lại các ô cờ trên bàn cờ.

Gọi phương thức VeBanCo của đối tượng main để vẽ bàn cờ.

****

Biến StartColor được gán màu đỏ và biến EndColor được gán màu cam đậm.

Dòng int Dong = e.Y / Cao; int Cot = e.X / Rong; tính toán vị trí dòng và cột trên bàn cờ dựa trên tọa độ chuột.

Nếu người dùng nhấn chuột trái (e.Button == MouseButtons.Left), biến MouseDowning được gán giá trị true.

Nếu người dùng nhấn chuột phải (e.Button == MouseButtons.Right), chúng ta xử lý việc di chuyển vật thể (ô cờ) trên bàn cờ.

Nếu vị trí chuột trùng với điểm bắt đầu (main.ToaDo[Dong, Cot] == StartPos) hoặc đã được kích hoạt di chuyển vật thể (MovePosObject), chúng ta thực hiện các bước sau:

* Gán MovePosObject = true để đánh dấu rằng đang di chuyển vật thể.
* Xoá ô hiện tại của vật thể bằng cách vẽ lại ô đó với màu nền của bàn cờ.
* Cập nhật vị trí điểm bắt đầu (StartPos) thành vị trí mới mà chuột di chuyển tới.
* Vẽ lại ô mới của vật thể bằng cách vẽ lại ô đó với màu StartColor và đánh dấu là đã được chọn.

**A computer screen with text on it

Description automatically generated**

Nếu vị trí chuột trùng với điểm bắt đầu (main.ToaDo[Dong, Cot] == StartPos) hoặc đã được kích hoạt di chuyển vật thể (MovePosObject), chúng ta thực hiện các bước sau:

* Gán MovePosObject = true để đánh dấu rằng đang di chuyển vật thể.
* Xoá ô hiện tại của vật thể bằng cách vẽ lại ô đó với màu nền của bàn cờ.
* Cập nhật vị trí điểm bắt đầu (StartPos) thành vị trí mới mà chuột di chuyển tới.
* Vẽ lại ô mới của vật thể bằng cách vẽ lại ô đó với màu StartColor và đánh dấu là đã được chọn.

**A computer screen shot of a program code

Description automatically generated**

Nếu biến Start đang được đặt thành true, tức là người chơi đang chọn điểm bắt đầu, chúng ta thực hiện các bước sau:

* Tăng giá trị của StartandEnd lên 1 để đếm số lượng điểm bắt đầu và điểm kết thúc đã được chọn.
* Gán StartPos thành vị trí mới mà chuột nhấn vào trên bàn cờ.
* Vẽ lại ô tại vị trí mới với màu StartColor và đánh dấu ô đó là đã được chọn.
* Gán Start thành false để kết thúc chế độ chọn điểm bắt đầu.
* Phương thức thoát ra và không xử lý thêm.

Nếu biến End đang được đặt thành true, tức là người chơi đang chọn điểm kết thúc, chúng ta thực hiện các bước sau:

* Tăng giá trị của StartandEnd lên 1 để đếm số lượng điểm bắt đầu và điểm kết thúc đã được chọn.
* Nếu chế độ chơi chỉ cho phép một vật thể hoặc chế độ tự động và đã có một điểm kết thúc được chọn (form2.OneObject || form2.Automatic) && ArrayEndPos.Count == 1), phương thức sẽ thoát ra mà không xử lý thêm.
* Thêm vị trí mới mà chuột nhấn vào vào danh sách ArrayEndPos.
* Vẽ lại ô tại vị trí mới với màu EndColor.
* Phương thức thoát ra và không xử lý thêm.

**A computer screen shot of a program code

Description automatically generated**

Nếu biến DelOne là true và SetObject\_Clicked là false, chúng ta thực hiện các bước sau:

* Kiểm tra nếu vị trí hiện tại trên bàn cờ (main.ToaDo[Dong, Cot]) không trùng với điểm bắt đầu (StartPos) và không tồn tại trong danh sách các điểm kết thúc (ArrayEndPos.IndexOf(main.ToaDo[Dong, Cot]) == -1).
* Nếu ô tại vị trí hiện tại (main.OCo[Dong, Cot]) không có màu giống với màu nền của bàn cờ (main.OCo[Dong, Cot].MauOCo != MauBanCo), thực hiện các bước sau:
* Vẽ lại ô tại vị trí hiện tại với màu nền của bàn cờ (MauBanCo).
* Gán VatCan của ô tại vị trí hiện tại thành false.
* Gán màu của ô tại vị trí hiện tại thành màu nền của bàn cờ - (MauBanCo).
* Phương thức thoát ra và không xử lý thêm.

**A screen shot of a computer program

Description automatically generated**

Nếu biến SetObject\_Clicked là true và không kích hoạt sự kiện xóa một vật cản (DelOne là false), chúng ta thực hiện các bước sau:

Kiểm tra nếu ô tại vị trí hiện tại trên bàn cờ (main.OCo[Dong, Cot]) không có màu giống với màu nền của bàn cờ (MauBanCo) và không có màu trắng (Color.White), thì phương thức sẽ thoát ra mà không xử lý thêm.

Nếu chế độ chọn là "Jungle" (đồng cỏ), thực hiện các bước sau:

* Vẽ lại ô tại vị trí hiện tại với màu xanh lá cây (Color.LawnGreen).
* Gán giá trị ChiPhi của ô tại vị trí hiện tại thành 10.
* Gán màu của ô tại vị trí hiện tại thành màu xanh lá cây.

Nếu chế độ chọn là "Mountain" (núi), thực hiện các bước sau:

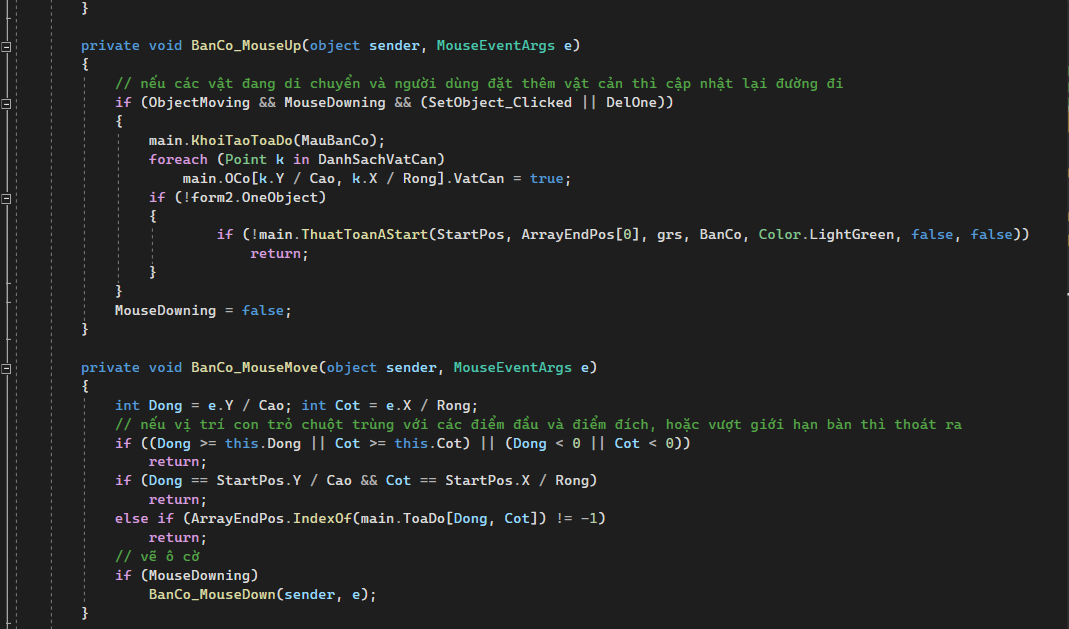
* Vẽ lại ô tại vị trí hiện tại với màu nâu (Color.Brown).
* Gán giá trị ChiPhi của ô tại vị trí hiện tại thành 20.
* Gán màu của ô tại vị trí hiện tại thành màu nâu.

Nếu chế độ chọn là "River" (sông), thực hiện các bước sau:

* Vẽ lại ô tại vị trí hiện tại với màu xanh dương (Color.Aqua).
* Gán giá trị ChiPhi của ô tại vị trí hiện tại thành 15.
* Gán màu của ô tại vị trí hiện tại thành màu xanh dương.

Nếu chế độ chọn là "Wall" (tường), thực hiện các bước sau:

* Gán VatCan của ô tại vị trí hiện tại thành true.
* Vẽ lại ô tại vị trí hiện tại với màu xám đen (Color.SlateGray).
* Gán giá trị ChiPhi của ô tại vị trí hiện tại thành 99999999999999 (giá trị lớn để biểu thị là vật cản).
* Gán màu của ô tại vị trí hiện tại thành màu xám đen.
* Thêm vị trí hiện tại vào danh sách DanhSachVatCan

****Phương thức BanCo\_MouseUp được gọi khi người dùng nhả chuột sau khi di chuyển các vật trên bàn cờ. Các bước thực hiện trong phương thức này như sau:

* Kiểm tra nếu đang có vật đang di chuyển (ObjectMoving) và người dùng đã nhấn chuột xuống (MouseDowning), và đang trong trạng thái đặt vật cản hoặc xóa một vật cản (SetObject\_Clicked hoặc DelOne đều là true).
* Khởi tạo lại các tọa độ trên bàn cờ (main.KhoiTaoToaDo(MauBanCo)).
* Đánh dấu các ô trên bàn cờ mà chứa vật cản (VatCan) dựa trên danh sách DanhSachVatCan.
* Nếu không chỉ định chỉ đường cho một vật cản duy nhất (form2.OneObject là false), thực hiện các bước sau:
* Nếu thuật toán tìm đường A\* (main.ThuatToanAStart) không thành công trong việc tìm đường từ điểm bắt đầu (StartPos) đến điểm kết thúc (ArrayEndPos[0]), phương thức sẽ thoát ra mà không xử lý thêm.
* Gán giá trị MouseDowning thành false.

Phương thức BanCo\_MouseMove được gọi khi di chuột trên bàn cờ. Các bước thực hiện trong phương thức này như sau:

* Tính toán chỉ số dòng (Dong) và chỉ số cột (Cot) dựa trên vị trí con trỏ chuột (e.Y / Cao và e.X / Rong).
* Kiểm tra nếu vị trí con trỏ chuột vượt quá giới hạn của bàn cờ hoặc nằm ngoài bàn cờ (Dòng < 0 hoặc Cột < 0), phương thức sẽ thoát ra mà không xử lý thêm.
* Kiểm tra nếu vị trí con trỏ chuột trùng với điểm bắt đầu (StartPos), hoặc nằm trong danh sách các điểm kết thúc (ArrayEndPos), phương thức sẽ thoát ra mà không xử lý thêm.
* Nếu người dùng đang nhấn giữ chuột (MouseDowning), gọi phương thức BanCo\_MouseDown để vẽ ô cờ tại vị trí con trỏ chuột.

A computer screen with text on it

Description automatically generated

Phương thức StartPoint\_Click được gọi khi người dùng nhấp vào nút Điểm bắt đầu (StartPoint). Các bước thực hiện trong phương thức này như sau:

* Gán giá trị Start thành true, đánh dấu rằng người dùng đã chọn điểm bắt đầu.
* Vô hiệu hóa nút Điểm bắt đầu (StartPoint) để ngăn người dùng chọn lại.

Phương thức EndPoint\_Click được gọi khi người dùng nhấp vào nút Điểm kết thúc (EndPoint). Các bước thực hiện trong phương thức này như sau:

* Hiển thị một hộp thoại (form2.ShowDialog()) để cho phép người dùng chọn các tùy chọn liên quan đến điểm kết thúc.
* Gán giá trị End thành true, đánh dấu rằng người dùng đã chọn điểm kết thúc.
* Vô hiệu hóa nút Điểm kết thúc (EndPoint) để ngăn người dùng chọn lại.

**A computer screen with colorful text

Description automatically generated**

Biến StartandEnd được sử dụng để đếm số lượng điểm bắt đầu và điểm kết thúc đã được chọn.

Phương thức Search\_Click được gọi khi người dùng nhấp vào nút Tìm kiếm (Search). Các bước thực hiện trong phương thức này như sau:

* Tạo một luồng mới (ThreadSearch) để thực hiện thuật toán tìm kiếm trong một luồng riêng biệt.
* Tạo một đối tượng Stopwatch (TinhThoiGian) để đo thời gian thực thi tìm kiếm.
* Đặt các biến Start và End thành false, đánh dấu rằng không có điểm bắt đầu và điểm kết thúc nào được chọn.
* Đặt biến MovePosObject thành false, đánh dấu rằng không có vật trên bàn cờ được di chuyển.
* Kiểm tra nếu đã chọn ít nhất hai điểm bắt đầu và điểm kết thúc (StartandEnd >= 2):
* Kích hoạt nút Xóa tất cả (DeleteAll).
* Thiết lập luồng TimKiem là một luồng nền (TimKiem.IsBackground = true).
* Bắt đầu chạy luồng TimKiem bằng cách gọi phương thức Start().
* Đặt StartandEnd về 0 để chuẩn bị cho lần tìm kiếm tiếp theo.
* Nếu chưa chọn đủ hai điểm bắt đầu và điểm kết thúc (StartandEnd < 2), hiển thị một hộp thoại thông báo yêu cầu người dùng chọn đủ hai điểm.
* Đặt các biến SetObject\_Clicked và DelOne thành false, đảm bảo rằng không có hoạt động đặt vật cản hoặc xóa vật cản nào đang được thực hiện.

**A computer screen shot of a program code

Description automatically generated**

Phương thức ThreadSearch chứa thuật toán tìm kiếm và được thực hiện trong một luồng riêng biệt.

* Một đối tượng TimerCallback được tạo để gọi phương thức ThoiGian theo khoảng thời gian nhất định.
* Một đối tượng System.Threading.Timer được tạo để chạy TimerCallback và gọi phương thức ThoiGian. Thời gian bắt đầu (0) và khoảng thời gian (1) được chỉ định.
* Nếu thuật toán tìm kiếm được chọn là A\* (AStart.Checked), thực hiện các bước sau:
* Gọi phương thức main.ThuatToanAStart để thực hiện thuật toán A\* từ điểm bắt đầu (StartPos) đến điểm kết thúc (ArrayEndPos[0]).
* Kiểm tra nếu thuật toán không thành công trong việc tìm đường, dừng bộ đếm thời gian (TinhThoiGian.Stop()) và hủy bỏ timer. Sau đó, phương thức kết thúc.

Nếu không chỉ định chỉ đường cho một vật cản duy nhất (form2.OneObject là true) và chế độ tìm kiếm bằng tay (form2.ThuCong là true), thực hiện các bước sau:

* Duyệt qua mỗi điểm kết thúc trong danh sách ArrayEndPos.
* Lấy gốc của ô hiện tại (point) để vẽ đường đi từ điểm bắt đầu (StartPos) đến điểm kết thúc. Đường đi được vẽ bằng phương thức main.VeDuongDi.
* Dừng bộ đếm thời gian (TinhThoiGian.Stop()) và hủy bỏ timer.

A computer screen shot of a program

Description automatically generated

Phương thức ThoiGian được gọi bởi TimerCallback để cập nhật thời gian đã trôi qua và hiển thị nó trên giao diện người dùng.

* Phương thức nhận một đối tượng sender nhưng không sử dụng nó trong thân phương thức.
* Nếu TinhThoiGian chưa được khởi chạy (TinhThoiGian.IsRunning là false), bắPhương thức ThoiGian được gọi bởi TimerCallback để cập nhật thời gian đã trôi qua và hiển thị nó trên giao diện người dùng.
* Phương thức nhận một đối tượng sender nhưng không sử dụng nó - trong thân phương thức.
* Nếu TinhThoiGian chưa được khởi chạy (TinhThoiGian.IsRunning là false), bắt đầu bộ đếm thời gian bằng cách gọi TinhThoiGian.Start().
* Dựa vào giá trị của TinhThoiGian.Elapsed.Milliseconds, thực hiện các bước sau để hiển thị thời gian trên giao diện người dùng:
* Nếu giá trị milliseconds nhỏ hơn 10, sử dụng định dạng "{seconds}.00{milliseconds}".
* Nếu giá trị milliseconds từ 10 đến 99, sử dụng định dạng "{seconds}.0{milliseconds}".
* Ngược lại, sử dụng định dạng "{seconds}.{milliseconds}".
* Sử dụng phương thức Invoke để đảm bảo rằng việc cập nhật giao diện được thực hiện trên luồng UI chính.

Phương thức LamMoi\_Click được gọi khi người dùng nhấp vào nút Làm mới (LamMoi). Các bước thực hiện trong phương thức này như sau:

* Tạo một đối tượng mới của Form2 để tạo lại form thứ hai.
* Đặt giá trị của StartandEnd về 0 để chuẩn bị cho một tìm kiếm mới.
* Kích hoạt các nút Điểm bắt đầu và Điểm kết thúc bằng cách đặt StartPoint.Enabled = EndPoint.Enabled = Moving.Enabled = true.
* Đặt điểm bắt đầu (StartPos) thành một điểm mới với tọa độ (0, 0).
* Xóa tất cả các điểm kết thúc trong danh sách ArrayEndPos bằng cách gọi ArrayEndPos.Clear().
* Đặt các biến ObjectMoving, DeleteAll.Enabled và MovePosObject thành false.

Thực hiện khởi tạo lại các tọa độ và trạng thái ban đầu của bàn cờ bằng cách gọi main.KhoiTaoToaDo(MauBanCo).

Refresh lại bàn cờ (BanCo.Refresh()) để xóa tất cả các vật cản và đường đi từ tìm kiếm trước.

**A computer screen shot of a program

Description automatically generated**

Trong phần khai báo biến, có một biến boolean là SetObject\_Clicked được khởi tạo giá trị ban đầu là false. Đây là một biến cờ để xác định xem có đang được click vào nút "Set Object" hay không.

Trong phương thức SetObject\_Click, khi nút "Set Object" được click, biến SetObject\_Clicked được gán giá trị true và các biến DelOne, Start, End được gán giá trị false. Điều này có nghĩa là người dùng đang đặt một vật cản và không đang xóa hoặc đặt điểm bắt đầu/điểm kết thúc.

Phần tiếp theo là một phần code được gắn thẻ #region với tên là "Chuyển động của vật thể". Đây là một phần xử lý chuyển động của vật thể.

Trong phương thức ChuyenDong, một vòng lặp for được sử dụng để lặp qua các vị trí đích của vật thể. Trong mỗi vòng lặp, một luồng Thread mới được tạo ra và khởi chạy phương thức VatDiChuyen trên luồng đó.

**Form2.cs**

**A computer screen shot of a black screen

Description automatically generated**

Khai báo và định nghĩa các biến và phương thức trong lớp Form2.

A computer screen shot of a program

Description automatically generated

Trong phương thức này, chúng ta thực hiện xử lý dựa trên các trạng thái của các nút radio và các giá trị của các điều khiển trên Form2.

* Nếu nút radio "One" được chọn, biến \_OneObject được đặt thành true.
* Nếu nút radio "Many" được chọn, chúng ta kiểm tra các trạng thái của nút radio "Manual" và "Auto".
* Nếu nút radio "Manual" được chọn, biến \_ThuCong được đặt thành true.
* Nếu nút radio "Auto" được chọn, biến \_Automatic được đặt thành true.
* Nếu giá trị của điều khiển Number là 0, hiển thị một hộp thoại cảnh báo và hủy sự kiện đóng Form2 bằng cách đặt e.Cancel = true.
* Nếu giá trị của điều khiển Number khác 0, biến \_Soluong được đặt thành giá trị của điều khiển Number.
* Biến \_OneObject được đặt thành false.

A computer screen shot of a program code

Description automatically generated

Trong phương thức này, chúng ta thực hiện các thay đổi liên quan đến giao diện của Form2 dựa trên trạng thái của nút radio "Many".

Nếu nút radio "Many" được chọn:

* Đặt nút radio "Auto" thành được chọn.
* Đặt kích thước tối thiểu và tối đa của Form2 tăng thêm 70 đơn vị chiều cao.
* Đặt kích thước của Form2 tăng thêm 70 đơn vị chiều cao.
* Hiển thị groupBox1 và các nút radio "Auto" và "Manual".
* Đặt vị trí của nút OK.

Nếu nút radio "Many" không được chọn:

* Đặt kích thước tối thiểu và tối đa của Form2 giảm đi 70 đơn vị chiều cao.
* Đặt kích thước của Form2 giảm đi 70 đơn vị chiều cao.
* Ẩn groupBox1 và các nút radio "Auto" và "Manual".
* Đặt vị trí của nút OK.
* Đặt fócus vào nút OK.

**A computer screen shot of a program code

Description automatically generated**

Trong phương thức này, chúng ta thực hiện các thay đổi liên quan đến giao diện của Form2.

* Đặt kích thước tối thiểu và tối đa của Form2 giảm đi 70 đơn vị chiều cao.
* Đặt kích thước của Form2 giảm đi 70 đơn vị chiều cao.
* Đặt vị trí của nút OK.

Phương thức Auto\_CheckedChanged là một phương thức xử lý sự kiện được gọi khi trạng thái của nút radio "Auto" thay đổi.

* Nếu nút radio "Auto" được chọn:
* Kích hoạt điều khiển Number.
* Nếu nút radio "Auto" không được chọn:
* Vô hiệu hóa điều khiển Number.
* Đặt fócus vào nút OK.

**Main.cs**

A computer screen with text

Description automatically generated

Trong đoạn mã trên, chúng ta khai báo các biến và khởi tạo mảng ToaDo và mảng OCo trong hàm khởi tạo của lớp Main.

Biến ToaDo là một mảng hai chiều kiểu Point có kích thước [Dong, Cot]. Mảng này sẽ được sử dụng để lưu trữ các tọa độ của các ô trên bàn cờ.

Biến OCo là một mảng hai chiều kiểu Oco có kích thước [Dong, Cot]. Mảng này sẽ được sử dụng để lưu trữ thông tin về các ô trên bàn cờ.

Biến Rong và Cao đại diện cho kích thước của một ô trên bàn cờ.

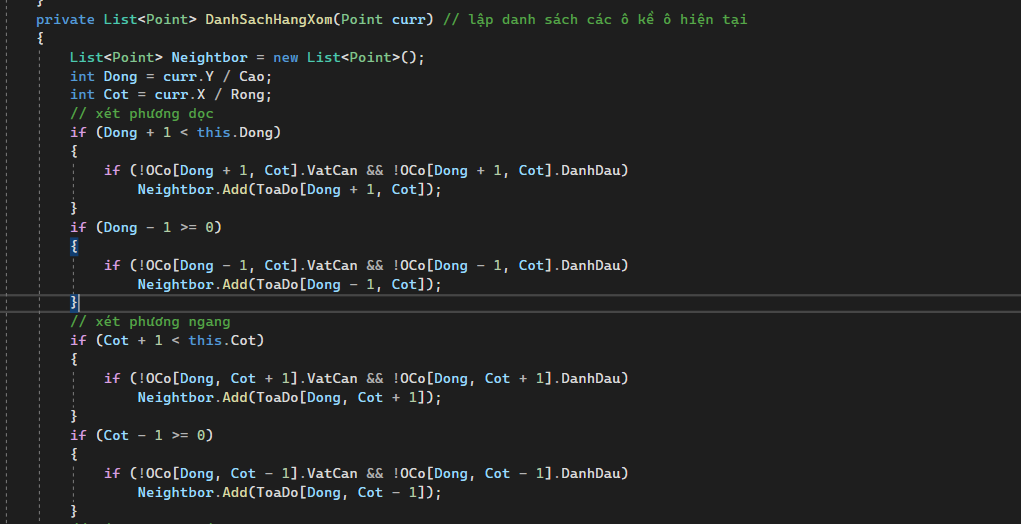
Biến Dong và Cot đại diện cho số lượng dòng và cột trên bàn cờ.

A computer screen with text and numbers

Description automatically generated

Trong đoạn mã trên, phương thức KhoiTaoToaDo được sử dụng để khởi tạo các tọa độ và thông tin về các ô trên bàn cờ.

* Với mỗi giá trị của biến i từ 0 đến (Dong - 1):
* Với mỗi giá trị của biến j từ 0 đến (Cot - 1):
* Gán tọa độ cho ô trên bàn cờ dựa trên giá trị của biến i và j. Tọa độ được tính bằng cách nhân j với Rong để xác định tọa độ X và nhân i với Cao để xác định tọa độ Y.
* Khởi tạo thông tin về ô trên bàn cờ. Đặt giá trị mặc định cho các thuộc tính của ô, bao gồm trạng thái (false), tọa độ (new Point(-1, -1)), loại (1), trạng thái được chọn (false) và màu nền (MauBanCo).



**A computer screen shot of a program code

Description automatically generated**

Trong đoạn mã trên, phương thức DanhSachHangXom được sử dụng để lập danh sách các ô kề với ô hiện tại (curr).

* Đầu tiên, chúng ta tính toán chỉ số của ô hiện tại trên mảng ToaDo dựa trên tọa độ của ô (curr).
* Sau đó, chúng ta kiểm tra các ô kề theo phương dọc và thêm vào danh sách Neightbor nếu ô đó không phải là vật cản (VatCan) và chưa được đánh dấu (DanhDau).
* Tiếp theo, chúng ta kiểm tra các ô kề theo phương ngang và thêm vào danh sách Neightbor nếu thỏa điều kiện tương tự.
* Cuối cùng, chúng ta kiểm tra các ô kề theo phương chéo và thêm vào danh sách Neightbor nếu thỏa điều kiện tương tự.
* Trả về danh sách Neightbor chứa các ô kề tìm được.

A computer screen shot of a program

Description automatically generated

Trong đoạn mã trên, phương thức VeOCo được sử dụng để vẽ một ô cờ tại vị trí (x, y) trên đối tượng Graphics g. Đầu tiên, chúng ta tạo một SolidBrush với màu color và sử dụng nó để vẽ một hình chữ nhật tại vị trí (x, y) với chiều rộng Rong và chiều cao Cao.

Nếu tham số VeOco là true, chúng ta sẽ vẽ lại các đường thẳng và cạnh của ô cờ. Cụ thể, chúng ta sử dụng g.DrawLine để vẽ các đường thẳng từ (x, y) đến (x, y + Cao) và từ (x, y) đến (x + Rong, y). Sau đó, chúng ta sử dụng g.DrawLine để vẽ các cạnh kề tiếp theo từ (x + Rong, y) đến (x + Rong, y + Cao) và từ (x, y + Cao) đến (x + Rong, y + Cao).

Phương thức VeLaiOCo được sử dụng để vẽ lại tất cả các ô cờ trên bàn cờ với màu MauBanCo. Chúng ta duyệt qua tất cả các ô trên mảng OCo và nếu màu của ô không phải là MauBanCo, chúng ta gọi phương thức VeOCo để vẽ lại ô đó.

**A computer screen shot of a program code

Description automatically generated**

Trong đoạn mã trên, phương thức XoaOCo được sử dụng để xóa ô cờ tại vị trí (x, y) bằng cách vẽ một hình chữ nhật có màu color. Đầu tiên, chúng ta tạo một SolidBrush với màu color và sử dụng nó để vẽ một hình chữ nhật tại vị trí (x, y) với chiều rộng Rong và chiều cao Cao.

**A computer screen shot of a code

Description automatically generated**

Trong đoạn mã trên, hàm HamDanhGia được sử dụng để đánh giá khoảng cách giữa hai điểm current và end. Đầu tiên, chúng ta tính khoảng cách theo trục x bằng cách lấy giá trị tuyệt đối của hiệu giữa end.X và current.X và lưu vào biến dx. Tương tự, chúng ta tính khoảng cách theo trục y bằng cách lấy giá trị tuyệt đối của hiệu giữa end.Y và current.Y và lưu vào biến dy.

Cuối cùng, chúng ta trả về giá trị nhỏ nhất giữa dx và dy bằng cách sử dụng hàm Math.Min(dx, dy). Điều này có nghĩa là hàm sẽ trả về khoảng cách ngắn nhất giữa current và end theo cả hai trục x và y.

**A screen shot of a computer program

Description automatically generated**

Trong quá trình tìm kiếm, chúng ta sử dụng một hàng đợi ưu tiên để lưu trữ các ô cờ. Đầu tiên, chúng ta đưa điểm start vào hàng đợi ưu tiên với ưu tiên dựa trên hàm đánh giá khoảng cách HamDanhGia.

Trong vòng lặp while, chúng ta lấy ô cờ tiếp theo từ hàng đợi ưu tiên và đánh dấu nó là đã được xét. Sau đó, chúng ta kiểm tra tất cả các ô cờ láng giềng của ô cờ hiện tại. Đối với mỗi ô cờ láng giềng, chúng ta cập nhật thông tin của nó, bao gồm gốc (để xác định đường đi), tổng chi phí tính từ điểm bắt đầu và đánh dấu nó là đã được xét. Nếu ô cờ láng giềng là điểm kết thúc, chúng ta vẽ đường đi từ điểm cuối cùng tìm được và trả về true.

Nếu HienThi được bật, chúng ta vẽ ô cờ láng giềng và hiển thị quá trình quét. Cuối cùng, chúng ta thêm ô cờ láng giềng vào hàng đợi ưu tiên với ưu tiên dựa trên tổng chi phí tính từ điểm bắt đầu và ước lượng khoảng cách đến điểm kết thúc.

Nếu hàng đợi ưu tiên trở thành rỗng, điều này có nghĩa là không tìm thấy đường đi và chúng ta hiển thị một thông báo lỗi và trả về false.

**A screen shot of a computer program

Description automatically generated**

Hàm sử dụng biến MauDuongDi để lưu trữ màu sắc cho việc vẽ đường đi. Ban đầu, biến này được đặt là Color.White.

Trong quá trình vẽ đường đi, hàm kiểm tra nếu điểm hiện tại không phải là điểm bắt đầu. Nếu điểm hiện tại có "gốc" (được truy vết từ ô trước đó), hàm tiếp tục xử lý.

Nếu ô hiện tại chưa được vẽ và là ô trống, nó sẽ được vẽ với màu MauDuongDi và ô đó được đánh dấu là đã vẽ. Nếu ô hiện tại chưa được vẽ và không phải là màu của vật cản, nó sẽ được vẽ với màu Color.Blue và ô đó được đánh dấu màu xanh.

Hàm sau đó di chuyển đến ô trước đó (theo "gốc") và nếu QuayCham là true, nó sẽ dừng trong 60 mili giây để làm chậm tốc độ vẽ.

Hàm sử dụng đệ quy để vẽ các ô trước đó trên đường đi cho đến khi đạt đến điểm bắt đầu. Khi đạt đến điểm bắt đầu, nó gọi hàm VeBanCo để vẽ toàn bộ bàn cờ (nếu QuayCham là false) và trả về true.

Nếu hàm hoàn thành mà không đạt đến điểm bắt đầu, nó trả về false.

**A screen shot of a computer program

Description automatically generated**

Lớp PriorityQueue<KieuGiaTri> là một hàng đợi ưu tiên.

Nó có thuộc tính Count để trả về số lượng phần tử trong hàng đợi.

Phương thức Enqueue được sử dụng để thêm một phần tử vào hàng đợi với một mức ưu tiên được chỉ định. Phần tử được thêm vào danh sách với cặp giá trị Tuple<KieuGiaTri, double> bao gồm phần tử và mức ưu tiên tương ứng.

Phương thức Dequeue được sử dụng để lấy ra và xóa phần tử có mức ưu tiên cao nhất khỏi hàng đợi. Để làm điều này, nó duyệt qua danh sách và tìm chỉ số của phần tử có mức ưu tiên thấp nhất. Sau đó, nó lấy phần tử đó ra khỏi danh sách và trả về giá trị của phần tử đó.

Đây là một lớp hàng đợi ưu tiên đơn giản được triển khai bằng cách sử dụng một danh sách và tìm kiếm tuần tự để tìm phần tử có mức ưu tiên cao nhất.

# KẾT LUẬN VÀ ĐỊNH HƯỚNG PHÁT TRIỂN

* Kết luận của đề tài viết chương trình game tìm đường về nhà trí tuệ nhân tạo là đã thành công trong việc xây dựng một trò chơi hấp dẫn và thú vị, trong đó trí tuệ nhân tạo được sử dụng để giải quyết vấn đề tìm đường từ điểm ban đầu đến điểm đích. Trò chơi này có thể tạo ra nhiều thách thức và trở ngại khác nhau trên đường đi, từ đó khám phá và xử lý.
* Định hướng phát triển của đề tài này có thể bao gồm:
* ***Tăng cường tính tương tác:*** Để làm cho trò chơi thêm sinh động và gần gũi với người chơi, có thể cung cấp các tùy chọn giao tiếp trực tiếp với trí tuệ nhân tạo, cho phép người chơi đặt câu hỏi, yêu cầu gợi ý hoặc xin giúp đỡ khi cần thiết.
* ***Mở rộng khả năng chơi:*** Bổ sung thêm các tính năng mới và thách thức trong trò chơi, như các loại địa hình khác nhau, các phương tiện di chuyển, hoặc các chức năng đặc biệt khác. Điều này giúp trò chơi trở nên đa dạng hơn và người chơi sẽ có nhiều cách tiếp cận khác nhau để giải quyết vấn đề.
* ***Cải thiện trí tuệ nhân tạo:*** Nâng cao khả năng tính toán và lựa chọn của trí tuệ nhân tạo để đảm bảo rằng nó luôn tìm ra đường đi tối ưu và thúc đẩy người chơi đến giải pháp tốt nhất. Cũng có thể cải thiện khả năng phân tích và sử dụng thông tin từ người chơi để điều chỉnh phương pháp giải quyết vấn đề.
* ***Mở rộng nền tảng và công nghệ:*** Phát triển hệ thống trò chơi đa nền tảng, cho phép người chơi trải nghiệm trò chơi trên các thiết bị khác nhau như điện thoại di động, máy tính bảng, hoặc máy tính cá nhân. Đồng thời, tận dụng các công nghệ mới như học sâu (deep learning) hay học tăng cường (reinforcement learning) để nâng cao hiệu suất và tính tương tác của trò chơi.
* Chương trình game tìm đường về nhà trí tuệ nhân tạo có thể trở thành một trò chơi thông minh và thú vị, mang lại những trải nghiệm giải trí và học hỏi mới cho người chơi.

# TÀI LIỆU THAM KHẢO

Slide giáo trình trí tuệ nhân tạo (Đại Học Công Thương TP.HCM)

<https://www.youtube.com/watch?v=G7XnNtF7UEE>

<https://github.com/HoangTran0410/p5js-playground/tree/master/2018/a-pathfinding?fbclid=IwAR3eBJxpUdc_e6lIE6WpBmUdHteowkXI9UpVFUSteNFGvR_yoVhPFuziqNg>

# PHỤ LỤC

1. Lịch làm việc nhóm trong hàng tuần
2. Công việc mỗi thành viên trong nhóm
3. Giới thiệu đồ án
4. Phân tích đề tài
5. Thiết kế
6. Cài đặt ứng dụng bài toán
7. Kết luận và định hướng phát triển