**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT THÀNH PHỐ HỒ CHÍ MINH   
KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**

**🙟🕮🙝**

Logo, icon

Description automatically generated

**ĐỒ ÁN MÔN HỌC**

**TRÍ TUỆ NHÂN TẠO**

**ĐỀ TÀI : TRÒ CHƠI CƯỚP CỜ**

**SVTH: ĐOÀN THỊ THANH VÂN 19110313**

**LÊ TRẦN MINH NHỰT 19110257**

**NGUYỄN THANH MINH ĐỨC 19110017**

**LÊ VŨ MINH HOÀNG 19119181**

**NGUYỄN THỊ XUÂN THANH 19110285**

Tp. Hồ Chí Minh, tháng 09 năm 2021

# **LỜI CẢM ƠN**

Để hoàn thành tốt đề tài và bài báo cáo này, chúng em xin gửi lời cảm ơn chân thành đến giảng viên, người đã trực tiếp hỗ trợ chúng em trong suốt quá trình làm đề tài. Chúng em cảm ơn thầy đã đưa ra những lời khuyên từ kinh nghiệm thực tiễn của mình để định hướng cho chúng em đi đúng với yêu cầu của đề tài đã chọn, luôn giải đáp thắc mắc và đưa ra những góp ý, chỉnh sửa kịp thời giúp chúng em khắc phục nhược điểm và hoàn thành tốt cũng như đúng thời hạn đã đề ra.

Chúng em cũng xin gửi lời cảm ơn chân thành các quý thầy cô trong khoa Đào tạo Chất Lượng Cao nói chung và ngành Công Nghệ Thông Tin nói riêng đã tận tình truyền đạt những kiến thức cần thiết giúp chúng em có nền tảng để làm nên đề tài này, đã tạo điều kiện để chúng em có thể tìm hiểu và thực hiện tốt đề tài. Cùng với đó, chúng em xin được gửi cảm ơn đến các bạn cùng khóa đã cung cấp nhiều thông tin và kiến thức hữu ích giúp chúng em có thể hoàn thiện hơn đề tài của mình.

Đề tài và bài báo cáo được chúng em thực hiện trong khoảng thời gian ngắn, với những kiến thức còn hạn chế cùng nhiều hạn chế khác về mặt kĩ thuật và kinh nghiệm trong việc thực hiện một dự án phần mềm. Do đó, trong quá trình làm nên đề tài có những thiếu sót là điều không thể tránh khỏi nên chúng em rất mong nhận được những ý kiến đóng góp quý báu của các quý cô cô để kiến thức của chúng em được hoàn thiện hơn và chúng em có thể làm tốt hơn nữa trong những lần sau. Chúng em xin chân thành cảm ơn.

Cuối lời, chúng em kính chúc quý cô, quý cô luôn dồi dào sức khỏe và thành công hơn nữa trong sự nghiệp trồng người. Một lần nữa nhóm chúng em xin chân thành cảm ơn cô.

**TP. Hồ Chí Minh, tháng 6 năm 2021**

**Nhóm sinh viên thực hiện**

*Nguyễn Thanh Minh Đức - 19110017*

*Lê Trần Minh Nhựt - 19110257*

*Lê Vũ Minh Hoàng - 19119181*

*Đoàn Thị Thanh Vân - 19110313*

*Nguyễn Thị Xuân Thanh - 19110285*

# **NHẬN XÉT**

**ĐIỂM SỐ**

|  |  |  |  |
| --- | --- | --- | --- |
| **TIÊU CHÍ** | **NỘI DUNG** | **TRÌNH BÀY** | **TỔNG** |
| **ĐIỂM** |  |  |  |

**Nhận xét của giáo viên**

......................................................................................................................................................

......................................................................................................................................................

......................................................................................................................................................

......................................................................................................................................................

......................................................................................................................................................

......................................................................................................................................................

......................................................................................................................................................

......................................................................................................................................................

......................................................................................................................................................

......................................................................................................................................................

......................................................................................................................................................

......................................................................................................................................................

*TP. Hồ Chí Minh, tháng 9, năm 2021*

*Giáo viên chấm điểm*

**MỤC LỤC**

[**LỜI CẢM ƠN** 1](#_Toc91093574)

[**NHẬN XÉT** 2](#_Toc91093575)

[**MỤC LỤC** 3](#_Toc91093576)

[**MỞ ĐẦU** 4](#_Toc91093577)

[**1.** **Giới thiệu đề tài** 4](#_Toc91093578)

[**2.** **Kế hoạch** 4](#_Toc91093579)

[**3.** **Danh sách phân công** 5](#_Toc91093580)

[**NỘI DUNG** 6](#_Toc91093581)

[**Phần I: TỔNG QUAN** 6](#_Toc91093582)

[**1.** **Lý do chọn đề tài** 6](#_Toc91093583)

[**2.** **Mục tiêu của đồ án** 6](#_Toc91093584)

[**3.** **Những thử thách của đề tài** 6](#_Toc91093585)

[**4.** **Phương pháp đã đề xuất** 9](#_Toc91093586)

[**5.** **Lựa chọn giải pháp** 10](#_Toc91093587)

[**6.** **Phương pháp đánh giá** 10](#_Toc91093588)

[**PHẦN 2 : SẢN PHẨM** 12](#_Toc91093589)

[**1.** **Giới thiệu trò chơi đối kháng** 12](#_Toc91093590)

[**2.** **Cách thức thực hiện** 14](#_Toc91093591)

[**3.** **Khó khăn và kinh nghiệm** 14](#_Toc91093592)

[**PHẦN 3 : TỔNG KẾT** 15](#_Toc91093593)

[**1.** **Kết luận** 15](#_Toc91093594)

[**2.** **Tham khảo** 15](#_Toc91093595)

# **MỞ ĐẦU**

### **Giới thiệu đề tài**

Với ý tưởng bắt nguồn từ trò chơi truyền thống là cướp cờ. Đây là một trò chơi truyền thống khá phổ biến tại Việt Nam. Trong đó, người chơi được chia thành hai đội với nhiệm vụ là đoạt được lá cờ trong sàn đấu để mang về điểm cho đội mình. Đến hết giờ, đội mang về được nhiều cờ nhất sẽ là đội chiến thắng.

Trong đồ án này, trò chơi cướp cờ sẽ được tái hiện với một phiên bản hoàn toàn mới. Người chơi sẽ liên tục di chuyển tìm những lá cờ xuất hiện ngẫu nhiên theo thông báo của ban tổ chức, hai người chơi khác đội không thể vượt qua nhau, bạn có thể chiếm cờ lấy điểm nhưng đồng thời cũng có thể ngăn cản đối phương cướp cờ và cuối cùng nếu tổng điểm đem về lớn hơn sẽ chiến thắng.

### **Kế hoạch**

*Bảng 1.5.1. Kế hoạch thực hiện theo tuần*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tuần** | **Công Việc Thực Hiện** | **Ngày Bđ** | **Ngày Kt** | **Kết Quả** |
| 1 |  |  |  |  |
| 2 |  |  |  |  |
| 3 |  |  |  |  |
| 4 |  |  |  |  |
| 5 |  |  |  |  |
| 6 |  |  |  |  |
| 7 |  |  |  |  |
| 8 |  |  |  |  |
| 9 |  |  |  |  |
| 10 |  |  |  |  |
| 11 |  |  |  |  |
| 12 |  |  |  |  |

### **Danh sách phân công**

*Bảng 1.5.2. Phân công công việc*

|  |  |  |  |
| --- | --- | --- | --- |
| **TT** | **Tên sinh viên** | **Mô tả công việc** | **Đóng góp** |
| 1 | Nguyễn Thanh Minh Đức | Phân tích các thuật toán cho agent | 25% |
| 2 | Đoàn Thị  Thanh Vân | Phân tích các trạng thái, môi trường, agent type | 25% |
| 3 | Nguyễn Thị Xuân Thanh | Phân tích môi trường, thiết kế sàn đấu | 25% |
| 4 | Lê Vũ  Minh Hoàng | Phân tích các trường hợp gặp phải khi dùng nhiều agent | 25% |
| 5 | Lê Trần  Minh Nhựt | Phân tích cách vận hành các agent, đánh giá các phần, phân công nhóm | 25% |

# **NỘI DUNG**

## **Phần I:** **TỔNG QUAN**

### **Lý do chọn đề tài**

Chúng em chọn trò chơi cướp cờ làm đề tài cho đồ án này vì trò chơi là cách đơn giản để thể hiện ý tưởng và sự tư duy đối với việc ứng dụng thuật toán vào thực tế, thay vì một đồ án phức tạp liên quan đến môn học, đồng thời vận dụng những kiến thức đã tìm hiểu và học được trong môn Trí tuệ nhân tạo.

### **Mục tiêu của đồ án**

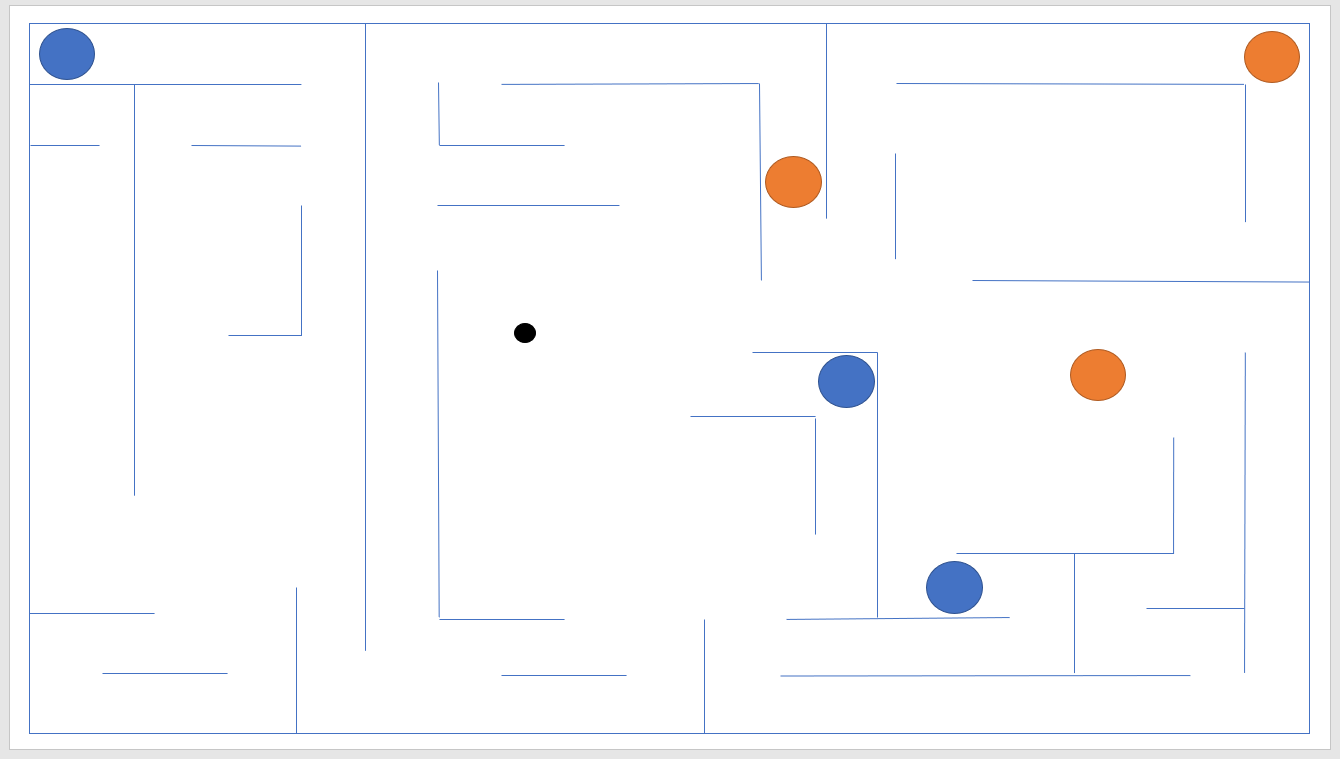
Đề tài này dựa trên bài toán tìm đường, nhưng thay vì một agent đi đến đích thì sẽ có một đội 3 agent tìm đường cướp cờ để ghi điểm. Từ đó, trò chơi thể hiện sự hoạt động của mô hình multi-agent.

### **Những thử thách của đề tài**

Khi phân tích, ta thấy tùy vào độ phức tạp mong muốn, đề tài có thể áp dụng nhiều kiểu môi trường khác nhau với nhiều thuật toán khác nhau.Do vậy, việc chọn một thuật toán thích hợp để minh họa đề tài cũng cần sự xem xét kĩ càng trong phạm vi kiến thức được truyền đạt và hiểu biết của nhóm và cân nhắc về độ phức tạp cũng như thời gian thực hiện hợp lý phù hợp với hạn báo cáo đồ án.

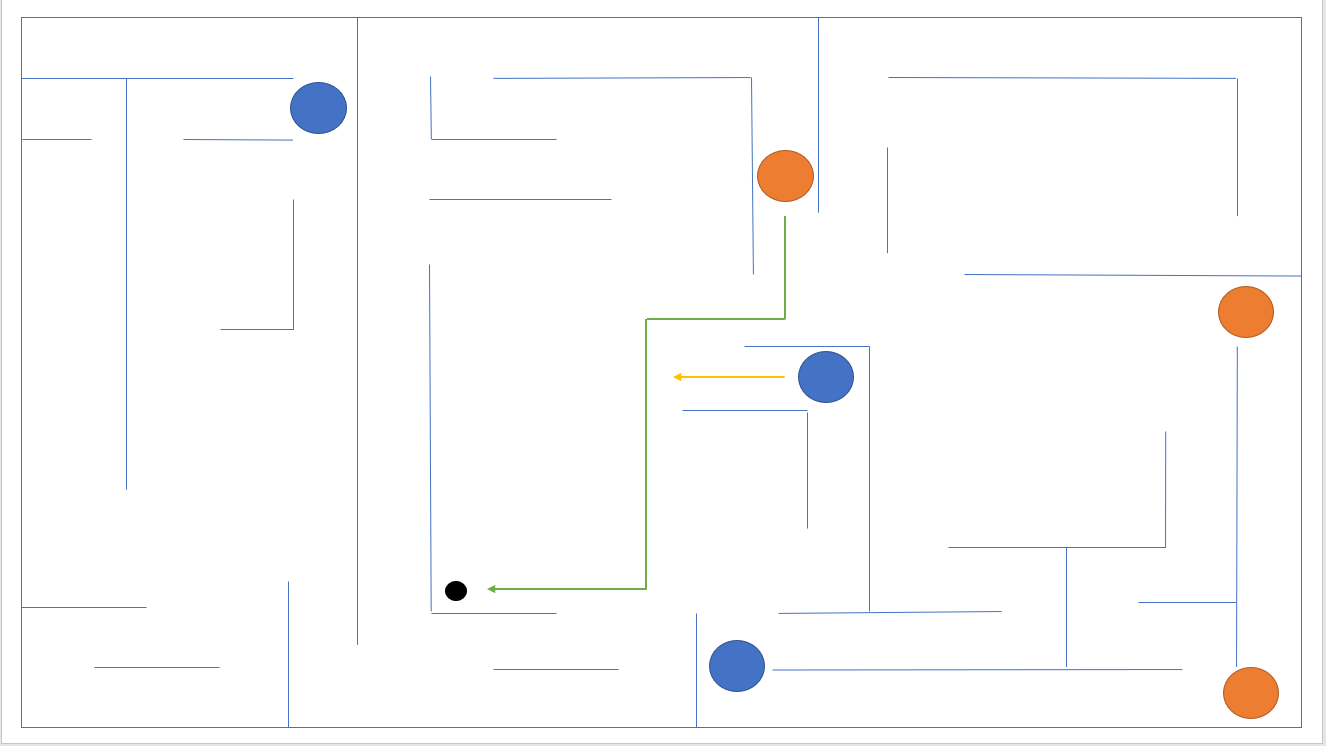
Agent type: Goal-based agent

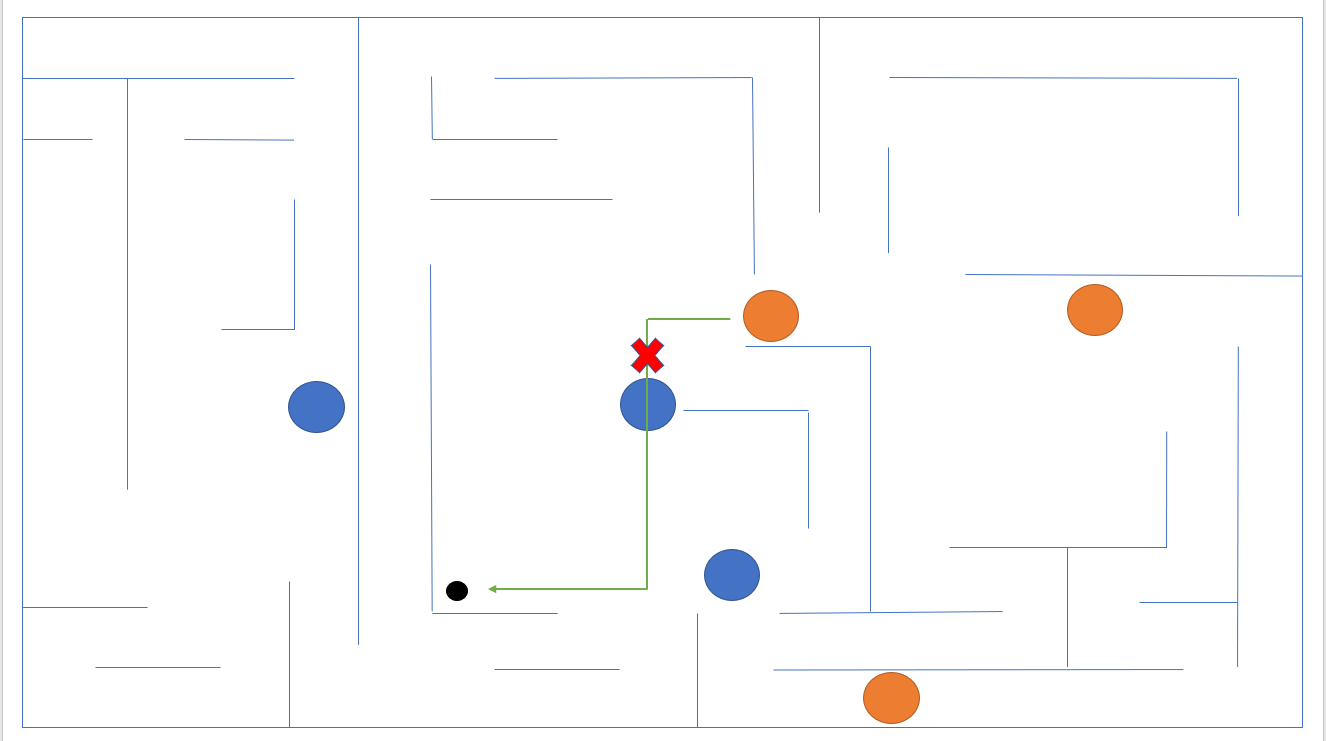
Về mặt môi trường được sử dụng trong đề tài :



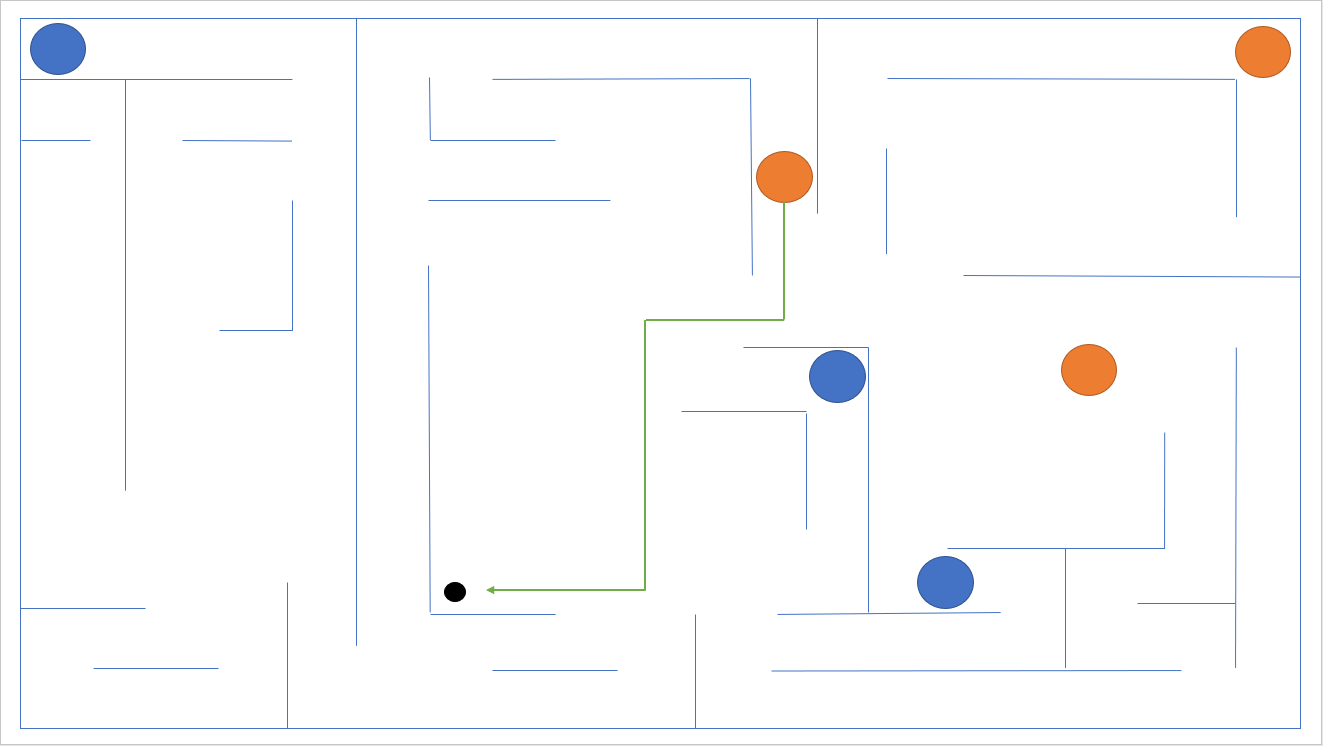
Với ý tưởng ban đầu của nhóm là xây dựng một sàn đấu với nhiều chướng ngại vật, ta sẽ tạo ra một bản đồ lớn và 2 đội chơi bao gồm 3 người chơi (màu xanh) và 3 agent đã được lập trình trước.

Môi trường có thể thay đổi liên tục nên agent phải luôn được cập nhật thông tin môi trường xung quanh.

*Agent tìm thấy đường đi đến cờ và người chơi cũng đang trên đường chiếm cờ*

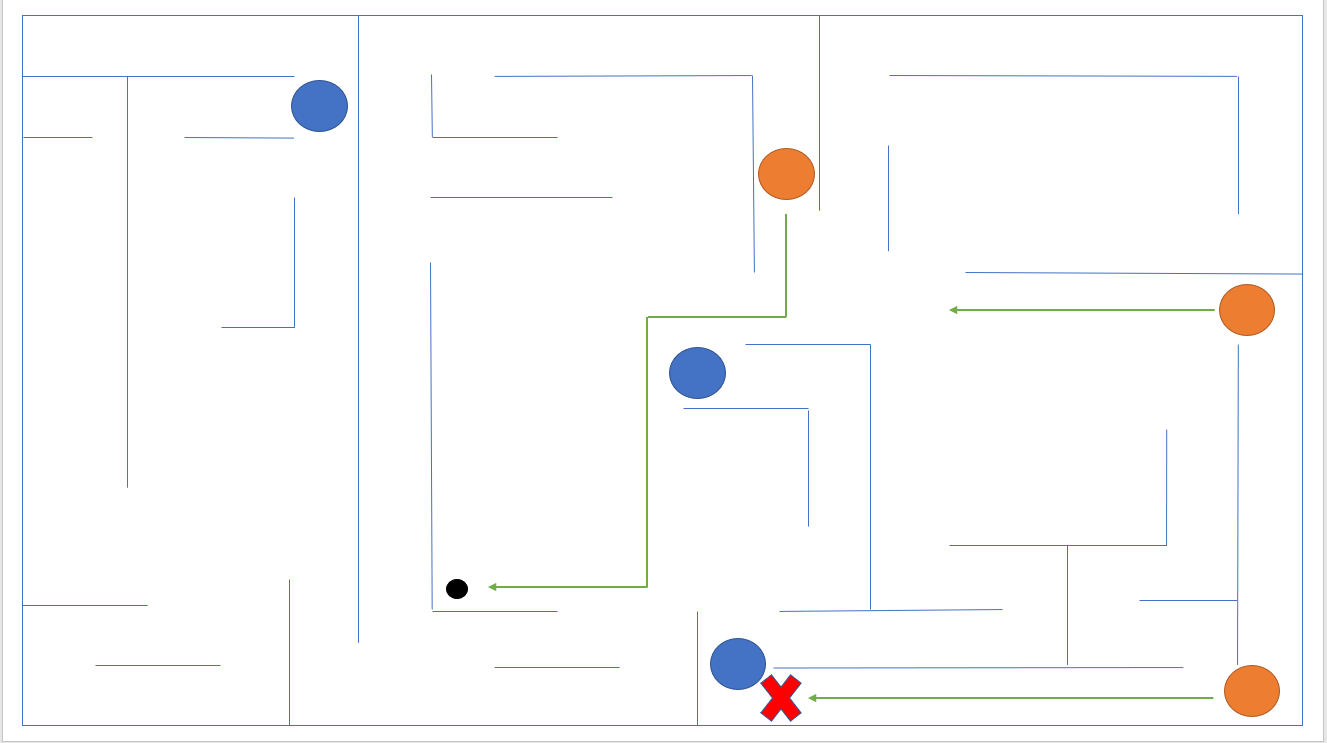
*Người chơi chặn đường đã tìm của agent*

Về mặt agent :



Cần phải cài đặt một thuật toán tìm đường với mức tối ưu khá cao để agent có thể bắt kịp 1 phần tốc độ của người chơi. Ta phải lựa chọn thuật toán phù hợp và đánh giá mức hiệu quả của nó đối với đề tài này. Ngoài ra, thuật toán được chọn nên thuộc phạm vi kiến thức mà mọi thành viên trong nhóm đều nắm được và có khả năng biến hóa, vận dụng vào một bài toán cụ thể. Có rất nhiều thuật toán phục vụ việc tìm đường nhưng tìm được thuật chung phù hợp với cả nhóm cũng là một trong các trở ngại cần giải quyết hàng đầu.

Về mặt chạy song song các agent :



Khi ta dùng nhiều agent trong một chương trình, những vấn đề cần quan tâm đến nhất là kiểm soát bộ nhớ, tối ưu việc dùng các agent hợp lý tùy trường hợp cờ để tránh lãng phí tài nguyên.

Dùng một thuật toán lên 3 agent sẽ rất chiếm bộ nhớ nếu ta không kiểm soát hợp lý. Ví dụ với giải thuật cần n byte để lưu trữ m node thì số byte cần để lưu trữ toàn bộ node của 3 agent sẽ là 3\*n\*m gấp 3 lần 1 bài toán thông thường

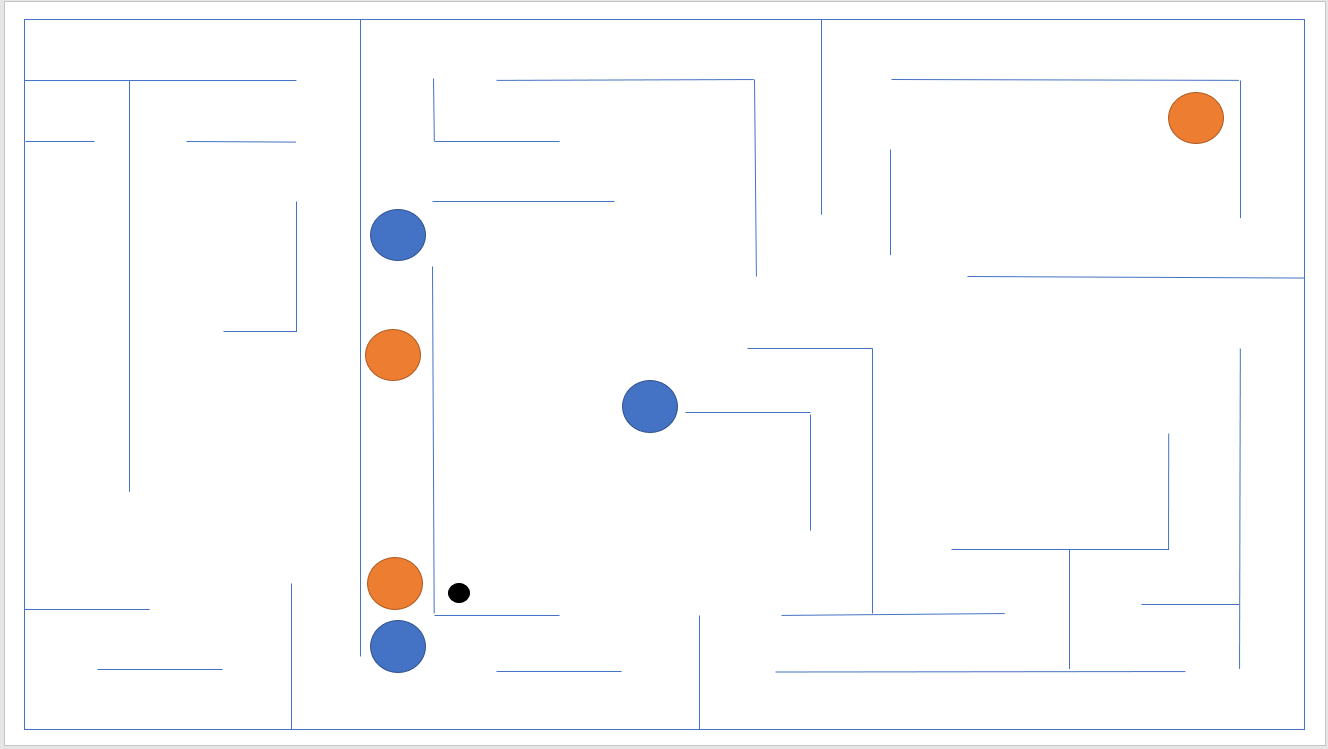
Cần tối ưu thời gian hợp lý với tốc độ chơi của người chơi.

### **Phương pháp đã đề xuất**

Với những phân tích về đề tài như ở mục 3, nhóm đã đề xuất môi trường sử dụng cho bài toán là Fully Observable and Multi-agent. Vì các thông tin trên sàn đấu luôn được cập nhật cho agent và sử dụng cùng lúc nhiều agent nên nhóm đã lựa chọn kết hợp 2 môi trường trên. Luôn cung cấp thông tin sàn đấu của đồng đội và địch cho agent để tìm lại đường đi mới sau mỗi trạng thái.

Và một số thuật toán được đưa ra nhằm giúp agent chiếm cờ là: DFS, BFS, Hill climbing, A\*,…

Để giải quyết vấn đề về bộ nhớ và thời gian, nhóm đề xuất giải pháp là sau mỗi trạng thái, các agent sẽ cùng đáng giá xem agent nào ở gần đích nhất để lựa chọn ra 1 hoặc 2 agent cùng đi chuyển. Phương pháp này sẽ giảm thiểu bộ nhớ phải dùng đến đồng thời giảm thời gian tìm đến cờ của các agent. Ngoài ra, nếu chỉ xét về khoảng cách ta sẽ gặp trường hợp cả 2 agent gần nhất đều bị chặn như hình :



Trong trường hợp này, agent sẽ không tìm được đường nếu cờ xuất hiện gần 2 agent đã bị chặn. Như vậy thì tỉ lệ máy thua sẽ rất cao nên nhóm đã đề xuất thêm một tiêu chí đánh giá là agent đó là kiểm tra agent ứng viên có bị chặn hết đường đi không, nếu agent ứng viên đã bị chặn thì máy sẽ đề xuất 1 agent khác thay thế trong trạng thái đó tìm đường và thực hiện kiểm tra lại ở mỗi trạng thái mới.

### **Lựa chọn giải pháp**

Sau khi đã liệt kê toàn bộ những ý tưởng có thể có trong đề tài này, nhóm quyết định chọn môi trường Fully Observable kết hợp với Multi-agent, để người chơi có trải nghiệm tốt nhất (không quá dễ để thắng) phương án Fully Observable được đề ra nhằm cung cấp thông tin cho các agent giúp agent tìm đường chiếm cờ nhanh hơn mà không cần dò tìm, khai phá thông tin sàn đấu.

Đối với thuật toán áp dụng cho agent, đây là một trong những yếu tố quyết định vì chọn thuật toán phù hợp với tình huống đưa ra sẽ mạng lại hiệu quả cao hơn, thế nên cần phân tích kĩ từng thuật toán.

Nếu dùng DFS, các agent có thể sẽ tìm đường rất nhanh nhưng cũng rất chậm, vì việc tìm đường của DFS phụ thuộc vào tỉ lệ đi đúng đường đến chiếm cờ và tỉ lệ đó khá thấp. Tuy có ưu điểm về việc tiết kiệm bộ nhớ vì không cần quét qua tất cả đường đi. Nhưng với nhược điểm phụ thuộc vào tỉ lệ đi đúng đường đến cờ không cao giải thuật khó tìm thấy đích nhanh chóng.

Còn về BFS, các agent có thể chiếm được cờ nhưng khi dùng 1 agent thì bộ nhớ để lưu số node đã bằng số ô trên sàn đấu ở trường hợp xấu nhất vậy với 3 hoặc nhiều hơn 3 agent cùng hoạt động sẽ cần tiêu tốn 1 bộ nhớ khá lớn. Điều này dẫn đến việc tốn bộ nhớ.

Hill-climbing là một biến thể của A\*, tuy kế thừa ưu điểm về thời gian và bộ nhớ của A\*, tuy nhiên, dùng Hill-climbing trong bài toán này có vẻ chưa hợp lý vì khi đi ngẫu nhiên sẽ ít cơ hội chiếm cờ trước người chơi hơn. Giải thuật này sẽ thích hợp hơn trong một bài sắp xếp hoặc lên lịch trình với một số yêu cầu có sẵn.

A\* với thời gian thực thi trong mức cho phép và bộ nhớ tương đối không quá lớn cùng hàm đánh giá H (Heuristic). A\* là một thuật toán tìm đường đi với mức tối ưu khá tốt. Mặt khác, các thành viên trong nhóm tương đối nắm rõ thuật toán này hơn những giải thuật còn lại. Đối với A\* trong đề tài này, điều quyết định hiệu quả của thuật toán là hàm H. Nếu xác định tốt hàm H thì việc tìm đường sẽ dễ dàng hơn.

### **Phương pháp đánh giá**

Đối với việc đánh giá, ta đánh giá hiệu quả các agent qua số điểm đạt được của chúng sau mỗi game đấu.

Đánh giá thông qua nhiều loại sàn đấu với các chướng ngại vật ở vị trí khác nhau.

Đánh giá thông qua cơ sở lý thuyết, tính toán độ phức tạp và bộ nhớ dựa trên các công thức tính của thuật toán A\* trong môi trường nhiều agent cùng hoạt động và đánh giá mức chính xác, sai lệch với thực tế.

Nâng cao tiêu chí đánh giá agent hoặc thêm vào một số tiêu chí phụ khác đánh giá hiệu quả khi chạy đề tài thực nghiệm.

## **PHẦN 2 : XÂY DỰNG**

1. **Xây dựng thuộc tính**

### **1.1. Các nhân tố xây dựng nên một bảng đồ trong AI :**

-Các biến ( nhân tố quan trọng nhất) : agents ( tính cả người lẫn máy AI), vật cản, điểm số, toạ độ nhân vật, toạ độ đích ( ở đồ án này chính là bánh pizza).

-Layout : Với một map ta có nhiều layout khác nhau, cụ thể như layout khung cửa sổ game, layout background, layout các biến có trên map, layout điểm số. Ở đây chúng ta sử dụng layout dạng tiled map ( là dạng layout thể hiện bằng những viên gạch hình vuông có màu sắc mà chùng ta hay thấy ở những game như flappy bird, mario, boom it,…). Các layout có các kích cỡ khác nhau được gộp chung lại với nhau như :

+ Trong layout khung cửa sổ game gồm : layout background, layout điểm số.

+ Trong layout background gồm : layout các biến có trên map.

-Hàm random : đây là hàm rất quan trọng đối với biến toạ độ đích để coin tự xuất hiện ngẫu nhiên bất kì vị trí trên bản đồ.

-Đồ hoạ và âm thanh : game trở nên quá đơn điệu nếu chỉ có số và chữ. Đồ hoạ sẽ giúp game trở nên sống động, màu sắc.

### **1.2. Cấu trúc xây dựng agent trong AI :**

Trong đề tài này, chúng ta có nhiều agent vì game có 1 người chơi đấu với nhiều máy nhưng cụ thể có hai loại agent chính : agent con người, agent robot.

-Agent con người : người chơi tự điều khiển một nhân vật trong game bằng bàn phím.

-Agent robot : các agent này được lập trình để thực hiện mục tiêu được giao. Hành động mà agent này thực hiện phụ thuộc vào khoảng cách với mục tiêu của nó. Các hành động nhằm mục đích giảm khoảng cách giữa trạng thái hiện tại và trạng thái mong muốn. Để đạt được mục tiêu, nó sử dụng thuật toán tìm kiếm (cụ thể ở đây được lập trình theo thuật toán A\*) tìm đường đến mục tiêu. Một hạn chế của Agent dựa trên mục tiêu là không phải lúc nào cũng chọn con đường tối ưu hóa nhất để đạt được mục tiêu cuối cùng.

### **2. Xây dựng bản đồ**

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Bảng | Code | Mô tả |
| 1 | Board.java | Random ra = new Random(); | Gán giá trị bản đồ ngẫu nhiên trong trò chơi |
| Color color;  private BufferedImage anh, wall , nen , wing;  @Override  public void Draw(Graphics g) {    color = new Color(ra.nextInt(256), ra.nextInt(256),ra.nextInt(256));  for (int i = 0; i < DoRongDai.countH; i++) {  for (int j = 0; j < DoRongDai.countW; j++) {  int x = j \* DoRongDai.cellW;  int y = i \* DoRongDai.cellH;    if( DoRongDai.b[i][j] == 1){  try {  wall = ImageIO.read(new File("image\\wall"+Mang.tuong+".png"));  } catch (IOException ex) {  Logger.getLogger(Board.class.getName()).log(Level.SEVERE, null, ex);  }  g.drawImage(wall, x, y,32,32, null);  }  else if( DoRongDai.b[i][j] != 1){  g.setColor(Color.BLACK);  g.fillRect(x, y, DoRongDai.cellW, DoRongDai.cellH);  }  }  }    try {  anh = ImageIO.read(new File("image\\bachkhoa.png"));  } catch (IOException ex) {  Logger.getLogger(Board.class.getName()).log(Level.SEVERE, null, ex);  }  g.drawImage(anh, 480, 0, 200, 200, null);    try {  wing = ImageIO.read(new File("image\\Angle.png"));  } catch (IOException ex) {  Logger.getLogger(Board.class.getName()).log(Level.SEVERE, null, ex);  }  g.drawImage(wing, 480,360 , null);    g.setColor(Color.RED);  g.setFont(new Font("Arial", Font.BOLD, 20)); //font, type, size  if(ToaDoBTL.ngang > 480 && ToaDoBTL.chay == 1)  ToaDoBTL.ngang --;  if(ToaDoBTL.ngang < 550 && ToaDoBTL.chay == 2)  ToaDoBTL.ngang ++;  if(ToaDoBTL.ngang == 480)  ToaDoBTL.chay = 2;  if(ToaDoBTL.ngang == 550)  ToaDoBTL.chay = 1;    g.drawString("BÀI TẬP LỚN", ToaDoBTL.ngang , ToaDoBTL.doc);  g.setColor(color);  g.drawString("TRÍ TUỆ NHÂN TẠO !", 480, 260);    g.setColor(Color.BLACK);    g.drawString("Điểm địch : " + Mang.DiemDich,500, 300);  g.drawString("Điểm ta : " + Mang.DiemTa,500, 330);  g.drawString("Thời gian: " + Mang.gio + ":" + Mang.phut +":" + Mang.giay , 500, 360);  g.drawString("Level :" + (7- Mang.level), 570 , 400);  // g.drawString(" MINH BK ", 570, 450);  }  class ToaDoBTL  {  static int ngang = 600;  static int doc = 230;  static int chay = 1;  } | Tạo layout, thiết lập toạ độ agent, bảng điểm số, hiện thị level, font chữ trong trò chơi. |
| 2 | GameBanCo.java | private void initWindow() {  //this.setSize(300, 300);  this.setLocation(200,200);  this.setTitle("ÁP DỤNG THUẬT GIẢI A\* VÀO GAME ");  this.pack();  this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  this.setVisible(true);    } | Tạo cửa sổ game |
| class Mang  {  static int [][] a;  static int n;  static int x,y;  static int DiemTa = 0 , DiemDich = 0  static int ThoiGianGame = 0;    static int Sleep = 100; trong vòng lặp While  static int level = 6;  static int DoNhay = 5;  static int gio = 0, phut = 0, giay = 0;  static int ThoiGianLap = 640;  static int tuong;  static int car;  static int carNC;  }  class DoRongDai{  static int cellW = 32;  static int cellH = 32;  static int countW = 15;  static int countH = 15  static int Them = 200;  static int [][] b;  } | Thiết lập trạng thái ban đầu khi vào game, thiết lập độ rộng dài của từng ô di chuyển, thiết lập thời gian chuyển bản đồ |
| public static void main(String[] args) throws FileNotFoundException, IOException {    InputStream is = new FileInputStream(new File("./image/nhachay2.wav"));  AudioStream as = new AudioStream(is);  AudioPlayer.player.start(as);  new GameBanCo();    } | Chèn thêm nhạc cho trò chơi |
| 3 | DiChuyenNgauNhien.java | public int RanDom(int khoang){  int map;  map = 1 + (int) (Math.random()\*khoang);  return map;  } | Hàm random để khi vào game sẽ chọn ngẫu nhiên một bản đồ |

**3. Xây dựng agent**

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Bảng | Code | Mô tả |
| 1 | GameManager.java | public void addComponent() { // hàm này chứa các xe của phe mình và phe địch  map.put("Car", new Car(2, Mang.a[0][0], Mang.a[0][1], false, true));  map1.put("Car", new ThuatToan(Mang.a[0][0], Mang.a[0][1]));  map.put("Car1", new Car(2, Mang.a[1][0], Mang.a[1][1], false, true));  map1.put("Car1", new ThuatToan(Mang.a[1][0], Mang.a[1][1]));  map.put("Car2", new Car(2, Mang.a[2][0], Mang.a[2][1], false, false));  map1.put("Car2", new ThuatToan(Mang.a[2][0], Mang.a[2][1]));  map.put("Car3", new Car(2, Mang.a[3][0], Mang.a[3][1], false, false));  map1.put("Car3", new ThuatToan(Mang.a[3][0], Mang.a[3][1]));  map.put("Car4", new Car(2, Mang.a[4][0], Mang.a[4][1], false, false));  map1.put("Car4", new ThuatToan(Mang.a[4][0], Mang.a[4][1]));  } | Tạo các agent trong game |
| 2 | CarNguoiChoi.java | public CarNguoiChoi( int huong, int posW, int posH, boolean KoOrCo, boolean LaBenNao) {  super( huong, posW, posH, KoOrCo, LaBenNao);  } | Thiết lập cách thức di chuyển cho người chơi |
| 3 | Car.java | public int posW;  public int posH;  public int ViTriW;  public int ViTriH;  public boolean LaNguoiChoi;  public boolean LaBenNao;  public int Huong;  public int KhoangDi = 0;  DiChuyenNgauNhien dc = new DiChuyenNgauNhien();  private BufferedImage carDich, carTa, carNC, diemKetThuc;  public Car(int huong, int posW, int posH, boolean KoOrCo, boolean LaBenNao) {  this.Huong = huong;  this.posW = posW;  this.posH = posH;  this.LaNguoiChoi = KoOrCo;  this.LaBenNao = LaBenNao;  TinhDoDaiThuc();  }  public void TinhDoDaiThuc() {  this.ViTriW = this.posW \* DoRongDai.cellW;  this.ViTriH = this.posH \* DoRongDai.cellH;  }  public void DiChuyenNgauNhien() {  if (this.KhoangDi <= 0) {  this.Huong = dc.TimHuongPhuHop(this.posW, this.posH);  this.KhoangDi = dc.KhoangCoTheDiDuoc(this.posW, this.posH, this.Huong);  }  if (this.KhoangDi > 0) {  this.DiChuyenDoanNgan();  }  }  public void DiChuyenDoanNgan() { // Di chuyển nhảy với độ ngắn, nhất định  if (Mang.ThoiGianGame % Mang.level == 0) {  if (this.Huong == 1)  this.posW++; // đi sang phải  else if (this.Huong == 2)  this.posH++; // đi xuông  else if (this.Huong == 3)  this.posW--; // di sang trai  else if (this.Huong == 4)  this.posH--; // di len tren  this.KhoangDi--;  this.TinhDoDaiThuc();  } else {  if (this.Huong == 1)  this.ViTriW += Mang.DoNhay;  else if (this.Huong == 2)  this.ViTriH += Mang.DoNhay;  else if (this.Huong == 3)  this.ViTriW -= Mang.DoNhay;  else if (this.Huong == 4)  this.ViTriH -= Mang.DoNhay;  }  } | Thiết lập toạ độ, hướng di chuyển trong bản đồ của agent |
|  |  | @Override  public void Draw(Graphics g) {  if (LaNguoiChoi) {  try {  carNC = ImageIO.read(new File("image\\Player\\spidey" + Mang.carNC + ".png"));  } catch (IOException ex) {  Logger.getLogger(Car.class.getName()).log(Level.SEVERE, null, ex);  }  g.drawImage(carNC, ViTriW, ViTriH, null);  } else if (LaBenNao == false) {  try {  carDich = ImageIO.read(new File("image\\Agent\\enemy.png"));  } catch (IOException ex) {  Logger.getLogger(Car.class.getName()).log(Level.SEVERE, null, ex);  }  g.drawImage(carDich, ViTriW, ViTriH, null);  } else if (LaBenNao == true) {  try {  carTa = ImageIO.read(new File("image\\Agent\\marvel" + Mang.car + ".png"));  } catch (IOException ex) {  Logger.getLogger(Car.class.getName()).log(Level.SEVERE, null, ex);  }  g.drawImage(carTa, ViTriW, ViTriH, null);  }  if (DiemCuoi.DichConOrKhong) { // Nếu đích còn thì mới vẽ ra chứ  try {  diemKetThuc = ImageIO.read(new File("image\\pizza.png"));  } catch (IOException ex) {  Logger.getLogger(Car.class.getName()).log(Level.SEVERE, null, ex);  }  g.drawImage(diemKetThuc, DiemCuoi.TargetFx \* DoRongDai.cellW, DiemCuoi.TargetFy \* DoRongDai.cellH, null);  }  } | Thiết kế hình ảnh cho người chơi, agent, đích trong trò chơi. |
|  | DiChuyenNgauNhien.java | public void NhapMapTuFile(String tenFile){  try ( Scanner sc = new Scanner(new File(tenFile))){  Mang.x = sc.nextInt();  Mang.y = sc.nextInt();  DiemCuoi.TargetFx = sc.nextInt  DiemCuoi.TargetFy = sc.nextInt();  Mang.n = sc.nextInt();  Mang.a = new int[Mang.n][2];  for(int i = 0 ; i < Mang.n ; i++)  for(int j = 0 ; j < 2 ; j++)  Mang.a[i][j] = sc.nextInt();  DoRongDai.b = new int[DoRongDai.countW][DoRongDai.countH];  for(int i = 0; i < DoRongDai.countW ; i++)  for(int j = 0; j< DoRongDai.countH ; j++)  DoRongDai.b[i][j] = sc.nextInt();  }catch(Exception ex){  System.out.printf("Loi gap phai: %s\n",ex.toString());  }  } | Lấy toạ độ xe người chơi, agent, đích và bản đồ ngẫu nhiên từ file map1.txt đến map10.txt |
|  | ThuatToan.java | class DiemCuoi {  public static int TargetFx, TargetFy; // Tọa độ của đích để chiếc xe hướng đến  public static boolean DichConOrKhong = true; // Đích ban đầu là có,  } | Tạo tọa độ đích cần đến |
|  |  | public class ThuatToan {  private Node start; // Nút Bắt đầu = tọa độ xe  private Node target; // Nút Kết thúc  private Node trunggian; // Các nút trung gian này có nhiệm vụ tìm con từ start - > target  private Node trunggian1;  private Node[][] nodeList; // Này là mảng 2 chiều Node tương ứng với 1 ô trong game.  public ThuatToan(int StartFx, int StartFy) {  initThuatToan(StartFx, StartFy);  }  public void ChayThuatToan() {  calculatePath();  TimCon();  }  public void initThuatToan(int StartFx, int StartFy) { // Khởi tạo thuật toán  nodeList = new Node[DoRongDai.countH][DoRongDai.countW];  for (int i = 0; i < DoRongDai.countH; i++)  for (int j = 0; j < DoRongDai.countW; j++) {  nodeList[i][j] = new Node(i, j).setX(i \* DoRongDai.countH).setY(j \* DoRongDai.countW);  }  start = nodeList[StartFx][StartFy];  target = nodeList[DiemCuoi.TargetFx][DiemCuoi.TargetFy];  for (int i = 0; i < DoRongDai.countH; i++)  for (int j = 0; j < DoRongDai.countW; j++) {  nodeList[i][j].calcH(target, 1);  if (DoRongDai.b[i][j] == 1) {  nodeList[i][j].setWall(true);  } else  nodeList[i][j].setWall(false);  }  trunggian = start;  }  public void TimCon() { // ngược lại với tìm cha, thì bây giờ tìm con  trunggian = target;  trunggian1 = target;  while (trunggian1 != start) {  trunggian1 = trunggian.getParent();  trunggian1.setChilden(trunggian);  trunggian = trunggian1;  }  trunggian = start; // trả lại trung gian = start ban đầu  }  public void clear() { // hàm này sẽ xóa toàn bộ những dữ liệu đã có trong nodeList, để trở lại ban đầu  start = null;  target = null;  trunggian = null;  for (int i = 0; i < DoRongDai.countH; i++)  for (int j = 0; j < DoRongDai.countW; j++)  nodeList[i][j].clear();  }  public void resetNode() {  target = null;  start = null;  for (int i = 0; i < DoRongDai.countH; i++)  for (int j = 0; j < DoRongDai.countW; j++) {  nodeList[i][j] = new Node(i, j).setX(i \* DoRongDai.cellW).setY(j \* DoRongDai.cellH);  }  }  // Trong game thì posW tính theo chiêu ngang, posH tính theo chiều dọc  // Còn với mảng, nodeList[i][j] , b[i][j] thì i tính theo chiều dọc, j tính theo chiều ngang  public void calculatePath() {  ArrayList<Node> closed = new ArrayList<Node>();  ArrayList<Node> open = new ArrayList<Node>();  Node currentNode = start;  open.add(currentNode);  while (open.isEmpty() != true) {  open.remove(currentNode);  closed.add(currentNode);  if (currentNode == target)  break;  for (int i = currentNode.getFX() - 1; i <= currentNode.getFX() + 1; i++) { // nghĩa là các giá trị cận kề,theo chiều dọc  if (i < 0 || i > DoRongDai.countH - 1) { // nếu <0 và > 20 thì bỏ qua  continue;  }  for (int j = currentNode.getFY() - 1; j <= currentNode.getFY() + 1; j++) { // nghĩa là các giá trị cận  // kề, theo chiều ngang  if (j < 0 || j > DoRongDai.countW - 1) {  continue;  }  if (i != currentNode.getFX() && j != currentNode.getFY()) {  continue; // Loại bỏ các nút chéo với currentNode  }  if (i == currentNode.getFX() && j == currentNode.getFY()) { // loại bỏ các nút chính nó  continue;  }  if (nodeList[j][i].isWall()) //i,j ở trong tọa độ game và nodeList là nghịch đảo của nhau.  continue;  Node node = nodeList[i][j];  if (!open.contains(node) && !closed.contains(node)) {  node.setG(node.calcG(currentNode)); // cấp nhập G cho node  node.setF(node.getF()); // cập nhập F cho node  node.setParent(currentNode);  open.add(node);  } else if (open.contains(node)) {  if (node.getG() > node.calcG(currentNode)) {  node.setG(node.calcG(currentNode));  node.setF(node.getF());  node.setParent(currentNode);  }  } else if (closed.contains(node)) {  if (node.getG() > node.calcG(currentNode)) {  node.setG(node.calcG(currentNode));  node.setF(node.getF());  node.setParent(currentNode);  closed.remove(node);  open.add(node);  }  }  }  }  currentNode = minOfList(open);  }  } | Khai triển thuật toán A\* |
|  |  | public Node minOfList(ArrayList<Node> open) {  Node n = open.get(0);  for (Node node : open) {  if (node.getF() < n.getF()) {  n = node;  }  }  return n;  } | Trả về node có F nhỏ nhất |

## **PHẦN 3 : SẢN PHẨM**

1. **Giới thiệu trò chơi đối kháng**

Game gồm 2 đội chơi : đội địch và đội ta . Đội ta bao gồm những agent được chọn ngẫu nhiên và người chơi điều khiển là spiderman trong hình vẽ . Đội địch cũng là những agent được chọn ngẫu nhiên . 2 đội di chuyển để ăn những chiếc bánh pizza xung quanh bản đồ. Ăn được 1 pizza sẽ giành được 1 điểm. Khi pizza xuất hiện thì những agent ( địch và ta) đều di chuyển tìm đường đi ngắn nhất dể đến một cách nhanh nhất. Khi pizza bị mất ( được ăn ) thì những agent sẽ di chuyển ngẫu nhiên cho tới khi pizza lại xuất hiện , thì chúng lại tiếp tục di chuyển theo thuật toán A\*.

Sau 1 khoảng thời gian ( trong game của em là 70 giây) thì map sẽ thay đổi ( nhạc nền thay đổi) , và giao diện của những agent cũng thay đổi ; level tăng lên 1 đồng nghĩa với những agent sẽ di chuyển nhanh hơn.

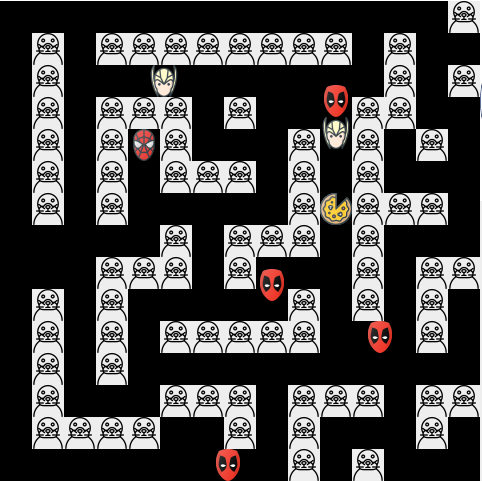
Game chỉ kết thúc khi đội nào ăn được 50 chiếc bánh pizza nhanh nhất!

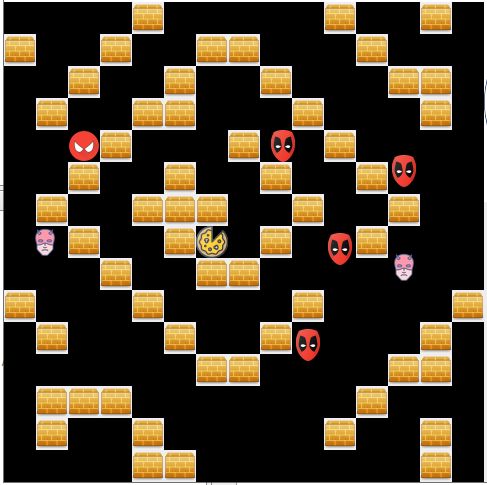
Người chơi dùng các phím mũi tên để di chuyển.

Trò chơi có tổng cộng 10 map và 6 nhân vật. Sau đây là một số map và những nhân vật sẽ xuất hiện ngẫu nhiên trong trò chơi :









1. **Cách thức thực hiện**

Sau khi tham khảo những thuật toán liên quan đến trò chơi, chúng em nhận thấy áp dụng thuật toán A\* là giải pháp tối ưu nhất cho trò chơi .Các agent di chuyển theo thuật toán A\* nên chúng sẽ di chuyển một cách nhanh nhất để đến khi có pizza và khi level càng tăng cao thì chúng sẽ di chuyển càng nhanh hơn . Game nhóm em làm mang tính chất giải trí cao và vui nhộn với âm thanh và map thay đổi liên tục trong 1 khoảng thời gian nhất định tạo hứng thú cho người chơi .

1. **Khó khăn và kinh nghiệm**

Khi nhận đề tài này, khó khăn lớn nhất của chúng em là việc tìm ra những thuật toán có thể áp dụng vào trò chơi. Các thuật toán đều có ưu điểm và nhược điểm riêng khiến cho việc lựa chọn gặp khá nhiều khó khăn.

Đây cũng là lần đầu chúng em thử tạo trò chơi nên việc chọn ngôn ngữ lập trình của chúng em gặp bế tắc. Việc tham khảo những source trò chơi khác trên mạng và hỏi ý kiến của những người có kinh nghiệm đã giúp chúng em tìm ra được ngôn ngữ lập trình thích hợp cho trò chơi này – Java.

## **PHẦN 3 : TỔNG KẾT**

### **Kết luận**

Thông qua việc tìm hiểu và nghiên cứu đề tài này giúp mọi người trong nhóm có cái nhìn toàn diện hơn trong việc ứng dụng trí tuệ nhân tạo vào giải quyết vấn đề thực tế. Đây là bài toán cổ điển trong trí tuệ nhân tạo cho các thuật toán mô hình hóa liên quan đến tím kiếm có tri thức bổ sung. Đề tào đã được nhiều người nghiên cứu giải quyết, nhưng cho đến nay vẫn chưa có cách giải quyết tối ưu cho tất cả không gian trạng thái trò chơi vì kícha thước tăng không gian trạng thái sẽ tăng lên rất nhanh.

Do thời gian có hạn nên đề tài không tránh khỏi những sai sót, mong thầy góp ý , đánh giá giúp em hoàn thiện tốt đề tài.

1. **Tham khảo**

Tài liệu tham khảo :

1. Stuart Russell and Peter Norvig : Sách Russell 2020 Artificial intelligence a modern approach (tái bản lần thứ tư)
2. Maxim Lapan : Sách Deep Reinforcement Learning Hands-On

Tài liệu trên mạng:

1. Thầy Trần Vũ Hoàng : Bài giảng Trí Tuệ Nhân Tạo (https://fhqx.hcmute.edu.vn/course/view.php?id=1129)
2. Thầy Thân Quang Khoát : Bài giảng Nhập môn Trí Tuệ Nhân Tạo (https://www.tailieubkhn.com/2021/02/tri-tue-nhan-tao.html)
3. Trang tìm kiếm wikipedia ([https://vi.wikipedia.org/wiki/Gi%E1%BA%A3i\_thu%E1%BA%ADt\_t%C3%ACm\_ki%E1%BA%BFm\_A\*](https://vi.wikipedia.org/wiki/Gi%E1%BA%A3i_thu%E1%BA%ADt_t%C3%ACm_ki%E1%BA%BFm_A*))