

**BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC CẦN THƠ
KHOA CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG**

**TIỂU LUẬN TỐT NGHIỆP
NGÀNH KHOA HỌC MÁY TÍNH**

**Đề tài:
MÔ PHỎNG TRÒ CHƠI
ADVANCE WARS VỚI UNITY VÀ ỨNG DỤNG TRÍ TUỆ NHÂN TẠO**

Cán bộ hướng dẫn

Ths Nguyễn Bá Diệp

Sinh viên thực hiện

Lê Duy Thanh

Mã số: B1310453

Khóa 39

Cần Thơ, 11/2017

MỤC LỤC

MỤC LỤC	1
Lời cảm ơn.....	3
Danh mục hình	4
Danh mục bảng	5
Tóm tắt.....	6
PHẦN GIỚI THIỆU.....	7
1. Đặt vấn đề.....	7
2. Mục tiêu đề tài.....	7
3. Đối tượng nghiên cứu.....	7
4. Công cụ sử dụng	7
5. Phương pháp nghiên cứu.....	7
6. Những đóng góp chính của đề tài.....	7
7. Sơ lược lịch sử trò chơi điện tử và trí tuệ nhân tạo trong trò chơi điện tử.....	7
8. Giới thiệu trò chơi Advance Wars.....	8
9. Giới thiệu công cụ Unity.....	8
PHẦN NỘI DUNG.....	9
CHƯƠNG 1 MÔ TẢ BÀI TOÁN.....	9
Mô tả chi tiết bài toán.....	9
1. Lập trình trò chơi Advance War với công cụ Unity	9
2. Phân tích và cài đặt các chiến thuật chơi trong trò chơi Advance War:	9
3. Viết tài liệu hướng dẫn:	9
CHƯƠNG 2: THIẾT KẾ, CÀI ĐẶT VÀ GIẢI PHÁP.....	10
1. Phân tích trò chơi Advance War.....	10
Mô tả tổng quát luật chơi:	10
2. Phân tích các đối tượng trong trò chơi.....	10
2.1. So sánh với phiên bản gốc.....	10
2.2. Phân tích chi tiết các đối tượng.....	11
2.3. Những quy tắc quan trọng trong thiết kế trò chơi Advance War.....	13
3. Công thức tính:.....	13

❖ Lượng máu sau giao tranh:.....	13
❖ Ma trận sát thương gây ra trong chiến đấu của mỗi loại đơn vị	14
❖ Điều kiện di chuyển.....	15
❖ Điều kiện giao tranh.....	16
❖ Bản đồ mức độ ảnh hưởng (Influence map)	17
4. Phân tích lượt đi – tương tác của người dùng với hệ thống.....	18
5. Các sơ đồ thiết kế ứng dụng.....	19
6. Thiết kế dữ liệu hướng đối tượng và cài đặt.....	20
6.1. Cấu trúc dữ liệu.....	20
6.2. Giải thuật	32
6.3. Cài đặt.....	34
7. Trí tuệ nhân tạo	34
7.1. Chiến lược chơi.....	34
7.2. Ứng dụng học máy vào trò chơi	35
a) Mỗi dòng dữ liệu.....	35
b) Tạo dữ liệu cho huấn luyện.....	36
c) Giải thuật học	36
d) Kết hợp Unity và python.....	36
e) Sử dụng mô hình.....	36
f) Kiểm thử.....	36
KẾT LUẬN.....	37
1. Kết quả đạt được	37
2. Hạn chế và hướng phát triển.....	37
Tài liệu tham khảo.....	38
PHỤ LỤC.....	39
1. Hướng dẫn chơi.	39
2. Hướng dẫn đọc và chỉnh sửa mã nguồn.	39

Lời cảm ơn

Đầu tiên, con xin được gửi lời cảm ơn đến gia đình, đến cha, mẹ đã cho con được một điều kiện tốt nhất để được học tập nghiên cứu suốt 4 năm đại học vừa qua. Tiếp theo, cũng xin được gửi lời cảm ơn đến khoa Công nghệ Thông tin và Truyền thông Đại học Cần Thơ - một môi trường học tập thật sự thoải mái, thân thiện và đầy tri thức. Cuối cùng cũng không quên gửi lời cảm ơn đến thầy Nguyễn Bá Diệp đã giúp đỡ, cung cấp cho em những tài liệu hữu ích trong suốt quá quá trình làm tiểu luận vừa qua cũng như những học phần Chuyên đề ngôn ngữ lập trình khoa học máy tính, kỹ thuật đồ họa, nguyên lý máy học. Xin chúc thầy thật nhiều sức khỏe!

Chân thành cảm ơn!

Lê Duy Thanh

Cần Thơ, 28/11/2017

Danh mục hình

Hình 1. Màn hình chính trò chơi Advanced War.....	11
Hình 2. Mô tả đường đi có thể một đơn vị	17
Hình 3. Mô tả bốn đối tượng mà phe đỏ có thể tiếp cận chiến đấu	18
Hình 4. Bản đồ ảnh hưởng (Influence map)	19
Hình 5. Sơ đồ tổng quát của trò chơi.	20
Hình 6. Trạng máy của một đơn vị chiến đấu	20
Hình 7. Trạng thái máy một thành phố	21
Hình 8. Sơ đồ tổng quát thể hiện mối liên hệ giữa các lớp	22
Hình 9. Ghi chú về dòng dữ liệu	36
Hình 10. Các nhãn của dòng dữ liệu được tạo ra	40
Hình 11. Hướng dẫn chơi	40

Danh mục bảng

Bảng 1. Các đơn vị chiến đấu	11
Bảng 2. Các địa hình	13
Bảng 3. Ma trận sát thương chuẩn (normal dame) gây ra trong chiến đấu của mỗi loại đơn vị.	16
Bảng 4. Thuộc tính lớp CellArmy	23
Bảng 5. Thuộc tính lớp CellTerrain	23
Bảng 6. Thuộc tính lớp CellPhysic	24
Bảng 7. Thuộc tính lớp LayerPhysic	24
Bảng 8. Thuộc tính lớp LayerArmy	25
Bảng 9. Thuộc tính lớp LayerPath	27
Bảng 10. Thuộc tính lớp Manager	29
Bảng 11. Thuộc tính lớp UserInterface	29
Bảng 12. Thuộc tính lớp LayerIM	30
Bảng 13. Thuộc tính lớp DisplayIM	31
Bảng 14. Thuộc tính lớp AI_DecisionMaking	32
Bảng 15. Thuộc tính lớp AI_Row	32
Bảng 16. Thuộc tính lớp AI_DataRow	33
Bảng 17. Thuộc tính lớp ConnectPython	33

Tóm tắt

Bài tiểu luận này, sẽ trình bày:

Lập trình trò chơi Advance War với công cụ Unity, từ đó tạo ra một khung làm việc có sẵn (framework) cho việc phát triển và ứng dụng các giải thuật trí tuệ nhân tạo khác nhau vào trò chơi. Ngoài ra, bài tiểu luận cũng được xem như một tài liệu hướng dẫn chi tiết cho việc sử dụng và nâng cấp sau này một cách dễ dàng.

PHẦN GIỚI THIỆU

1. Đặt vấn đề

Hiện nay, trí tuệ nhân tạo là một trong những lĩnh vực hấp dẫn và đang được nhiều đầu tư để nghiên cứu vì vô số tính ứng dụng của nó trong hầu hết mọi lĩnh vực. Song song đó, trò chơi điện tử cũng ngày càng được phát triển để đáp ứng nhu cầu giải trí của con người. Nhắc đến trò chơi điện tử, một thành phần không thể thiếu là trí tuệ nhân tạo, nó giúp trò chơi có độ khó, mô phỏng được thực tế và trở nên hấp dẫn hơn. Vì vậy, vấn đề đặt ra là làm thế nào để áp dụng trí tuệ nhân tạo vào trò chơi cũng như ngành công nghiệp giải trí nói chung một cách hiệu quả, đồng thời thể hiện tính ứng dụng của trí tuệ nhân tạo là không giới hạn.

2. Mục tiêu đề tài.

Phân tích, thiết kế và lập trình lại trò chơi Advance War của công bằng công cụ Unity. Đồng thời tạo ra một khung làm việc dựng sẵn (framework) nhằm phục vụ cho nghiên cứu và nâng cấp cũng như áp dụng những giải thuật trí tuệ nhân tạo vào hệ thống sau này.

3. Đối tượng nghiên cứu.

- Trò chơi Advanced Ward
- Công cụ lập trình game Unity
- Lý thuyết về trí tuệ nhân tạo – học máy

4. Công cụ sử dụng

- Phần mềm: Unity 2017, Visual Studio 2017, photoshop CS5
- Ngôn ngữ lập trình: C#, python
- Hệ điều hành: windows 7 64 bit

5. Phương pháp nghiên cứu

- Phân tích và tổng hợp lý thuyết: Phân tích, tìm hiểu các tài liệu về lập trình trò chơi điện tử và trí tuệ nhân tạo.
- Phương pháp thực nghiệm: Lập trình, kiểm thử trò chơi Advance War và ứng dụng trí tuệ nhân tạo.

6. Những đóng góp chính của đề tài

Xây dựng được một khung nền làm việc có sẵn phục vụ cho nghiên cứu và áp dụng các giải thuật trí tuệ nhân tạo về sau.

7. Sơ lược lịch sử trò chơi điện tử và trí tuệ nhân tạo trong trò chơi điện tử.

Khoảng năm 1950, những trò chơi điện tử đầu tiên ra đời nhưng vẫn còn rất đơn giản và sơ khai. Nhiều trò chơi điện tử ra đời nhưng cũng gặp nhiều giới hạn như hạn chế bộ nhớ hay tốc độ xử lý của máy tính. Vào những năm 1970, quá trình phát

triển máy tính bước qua thế hệ thứ tư, trò chơi điện tử có nhiều thay đổi đáng kể so với những bản trước đây về độ phức tạp, giao diện, tương tác được cải thiện tốt hơn và đã xuất hiện nhiều bản thương mại. Đến nay, trò chơi điện tử có nhiều bước phát triển, đặc biệt là đồ họa (từ 2d sang mô phỏng 3d với độ chân thật cao) và trí tuệ nhân tạo vượt qua trí tuệ con người (tiêu biểu là máy tính chơi cờ vua DeepBlue của IBM 1997, hay gần đây là máy chơi cờ vây AlphaGo của Google 2016).

8. Giới thiệu trò chơi Advance Wars

Advanced Wars là một trò chơi điện tử thuộc thể loại chiến thuật theo lượt được phát hành lần đầu năm 2001 bởi công ty Nintendo (Nhật Bản). Trò chơi mô phỏng các đội quân trên một địa hình đa dạng (đồi núi, sông, thành phố...), các thành phần tham gia chiến đấu được mô phỏng giống như thật tế: bộ binh, pháo binh, xe tăng, máy bay... Nhiệm vụ của người chơi là dùng một chiến thuật hợp lý để tiêu diệt toàn bộ quân địch hoặc chiếm giữ được bộ chỉ huy để giành chiến thắng. Trò chơi có hai chế độ: người đấu với người hoặc người với máy, mỗi trận đánh có thể lên đến năm đội đối đầu nhau trên cùng một bản đồ.

9. Giới thiệu công cụ Unity.

Unity là một công cụ hỗ trợ cho các nhà phát triển phần mềm ứng dụng 2d cũng như 3d nhằm rút ngắn thời gian trong quá trình xây dựng sản phẩm của mình. Một số tính năng quan trọng mà Unity hỗ trợ:

- Đa nền tảng, hỗ trợ kết nối thư viện bổ trợ ngoài (các plugin), hỗ trợ đồ họa hiệu năng cao: Unity sử dụng thư viện OpenGL và Direct X trên windows, linux, hay OpenGL ES trên di động. Đặc biệt ở chế độ đồ họa 3d, Unity hỗ trợ kết xuất (render) hình ảnh chất lượng cao, hỗ trợ bóng đổ (shader), sương mù (fog), khử răng cưa (anti aliasing), viết mã shader bằng ngôn ngữ Cg...
- Hỗ trợ ngôn ngữ lập trình C# (trước đây có cả JavaScript và Boo).
- Hỗ trợ tính toán vật lý: va chạm, hiệu ứng hạt (dòng nước, khói lửa), mô phỏng các lực vật lý (trọng lực, phản lực, dò tia..).
- Trí tuệ nhân tạo: cho phép nhà phát triển kết hợp huấn luyện mô hình với các giải thuật học máy thông qua một API ngôn ngữ python (hiện tại còn là một tính năng beta, ra mắt tháng 10/2017).

PHẦN NỘI DUNG

CHƯƠNG 1 MÔ TẢ BÀI TOÁN

Mô tả chi tiết bài toán

1. Lập trình trò chơi Advance War với công cụ Unity

Công việc đầu tiên cần thực hiện là lập trình trò chơi Advance War bằng công cụ Unity dựa trên bản gốc của công ty Nintendo. Việc lập trình này giúp ta có thể dễ dàng lấy các thông tin trong bộ nhớ nhằm phục vụ cho phần áp dụng trí tuệ nhân tạo. Đồng thời, đây cũng được xem là xây dựng một nền tảng sẵn có cho việc áp dụng trí tuệ nhân tạo học máy sau này. Sản phẩm của bước này là một trò chơi Advance War hoàn chỉnh chạy trên nền tảng Windows 7 64 bit.

Nội dung công việc:

- Phân tích: tìm hiểu luật chơi và các đối tượng trong trò chơi.
- Thiết kế các lớp, các giải thuật trong trò chơi.
- Kiểm thử chương trình.

2. Phân tích và cài đặt các chiến thuật chơi trong trò chơi Advance War:

Để đạt hiệu quả trong áp dụng học máy vào trò chơi nhằm đưa ra một quyết định thông minh nhất cho mỗi nước đi. Cần phải phân tích một số yếu tố đặc trưng có ảnh hưởng đến những quyết định bước đi đó. Sản phẩm của bước này là cài đặt một giải thuật trí tuệ nhân tạo vào trò chơi.

Nội dung công việc:

- Tìm hiểu về một số chiến lược để chơi Advanced War
- Tìm đặc trưng của một nước đi trong mỗi trận đấu
- Nghiên cứu lý thuyết và định hướng chọn phương pháp học máy để áp dụng làm trí tuệ nhân tạo cho chương trình

3. Viết tài liệu hướng dẫn:

Trình bày lại một cách hệ thống giúp cho người sử dụng sau này dễ hiểu để nâng cấp hoặc bổ sung tính năng cho trò chơi. Sản phẩm bước này là tài liệu hướng dẫn.

Nội dung công việc:

- Viết báo cáo tổng kết tiểu luận
- Viết tài liệu hướng dẫn về việc sử dụng mã nguồn trò chơi
- Viết tài liệu hướng dẫn chơi trò chơi

CHƯƠNG 2: THIẾT KẾ, CÀI ĐẶT VÀ GIẢI PHÁP

1. Phân tích trò chơi Advance War



Hình 1. Màn hình chính trò chơi Advanced War

Mô tả tổng quát luật chơi:

- Advance War là một trò chơi chơi theo lượt, nghĩa là mỗi bên sẽ có 1 lượt chơi (không giới hạn thời gian cho mỗi lượt), trong đó mỗi lượt gồm nhiều bước đi (mỗi bước đi là một hành động tác động vào một đơn vị). Sau khi thông báo kết thúc lượt của mình, hệ thống sẽ chuyển sang lượt đi cho đội bên kia. Cứ như thế lặp đi lặp lại cho đến khi kết thúc trò chơi.
- Bản đồ gồm 150 vị trí (ô) tương ứng 15 cột x 10 hàng. Ở mỗi ô sẽ chỉ chứa một đơn vị tại một thời điểm.
- Nhiệm vụ: mỗi bên sẽ di chuyển số quân lính của mình theo một chiến lược hợp lý để giành chiến thắng.
- Điều kiện thắng: Có thể giành chiến thắng bằng một trong ba trường hợp:
 - Tiêu diệt toàn bộ đơn vị của đội đối phương.
 - Chiếm được bộ chỉ huy của đội đối phương.
 - Đội đối phương chủ động đầu hàng.
- Kết thúc trò chơi: trò chơi kết thúc khi có một đội giành chiến thắng.
- Chế độ chơi: người đấu với máy và người đấu với người.

2. Phân tích các đối tượng trong trò chơi.

2.1. So sánh với phiên bản gốc

Để đơn giản hóa, phiên bản này sẽ lược bỏ đi một số đối tượng so với bản gốc:

- ✓ Những đối tượng sẽ có trong bản đơn giản: bộ binh, pháo binh, xe tăng, thành phố, đường, sông, núi, rừng cây; chỉ có hai đội trên bản đồ chiến đấu với nhau.
- ✓ Những đối tượng/chức năng bị lược bỏ: thuộc tính gas (năng lượng di chuyển – tức là ở bản đơn giản này, các đối tượng có thể di chuyển mãi mãi trong suốt trận đấu); xe chuyên chở; máy bay, tàu đã bị lược bỏ; tính năng có tối đa 5 đội cùng lúc trên một bản đồ bị lược bỏ.



2.2. Phân tích chi tiết các đối tượng

Sau đây là những đối tượng và thuộc tính tương ứng trong trò chơi Advanced Wars:

➤ Bản đồ:









Tất cả các đơn vị trò chơi được đặt trên bản đồ 150 ô (15 cột, 10 hàng). Mỗi ô sẽ chứa thông tin của địa hình và một đơn vị chiến đấu (nếu có) trên ô đó.

➤ Các đơn vị chiến đấu:

Stt	Tên	Hình ảnh (xanh/đỏ)	Định nghĩa	Số ô tối đa có thể di chuyển mỗi lượt	Địa hình có thể đi đến
1	Bộ binh (Infantry)		Có tính cơ động cao, sử dụng súng trường để chiến đấu.	3	Tất cả
2	Pháo binh (Mech)		Chuyên đánh tăng, dùng súng bắn tăng (bazooka) để chiến đấu.	3	Tất cả
3	Xe tăng (Tank)		Xe tăng, có sát thương cao, khả năng chống chịu tốt.	5	Đồng bằng, thành phố và cầu đường.

Bảng 1. Các đơn vị chiến đấu

➤ Địa hình:

Stt	Tên	Hình ảnh	Định nghĩa	Chỉ số bảo vệ đơn vị đứng trên địa hình này
1	Đồi núi (Mountain)		Địa hình đồi núi cao, có nhiều cây cối, thích hợp đánh úp từ trên cao.	2
2	Đồng bằng (Plain)		Đồng bằng bằng phẳng, thường nguy hiểm cho các đơn vị do không có chỗ ẩn nấp	0
3	Đường (Road)		Đường xá, dễ dàng di chuyển cho tất cả cá đơn vị nhưng không có chỗ ẩn nấp, khả năng bị sát thương cao	0
4	Cầu (Bridge)		Cầu nối 2 bờ sông, quan trọng đối với xe tăng vì đây là lối duy nhất để qua sông.	0
5	Rừng cây (Wood)		Khu vực rừng, dễ ẩn nấp	1
6	Bộ chỉ huy (Headquarter)		Bộ chỉ huy của các đơn vị mỗi đội. Thua cuộc nếu như bị đội địch đánh chiếm.	
7	Thành phố (City)		Khu vực thành phố, đặc biệt có khả năng giúp hồi phục cho các đơn vị khi đang chiếm giữ được thành phố đó.	2
8	Sông (River)		Khu vực sông ngòi, xe tăng không thể đi qua khu vực này	1

Bảng 2. Các địa hình

2.3. Những quy tắc quan trọng trong thiết kế trò chơi Advance War

- Tại mỗi thời điểm, tại một ô trên bản đồ chỉ có thể chứa tối đa một đơn vị chiến đấu.
- Thành phố:
 - Tại mỗi thời điểm, một thành phố có thể có một trong ba trạng thái: trung lập, sở hữu bởi đội đỏ, sở hữu bởi đội xanh (trường hợp chỉ có hai đội).
 - Một đơn vị ít nhất hai ngày để chiếm lấy một thành phố tức phải qua hai lần chiếm (không bắt buộc phải liên tục) thì mới thật sự chiếm được thành phố đó.
 - Nếu thành phố đang trong quá trình bị chiếm bởi đội A, thì có đội B đến chiếm thì xem như đội A đã hủy bỏ, đội B sẽ tiếp tục quá trình của A (xem như A chưa chiếm lượt nào, B đã được một lượt).
 - Thành phố chỉ có thể bị chiếm bởi Pháo binh hoặc Bộ binh, không thể đánh chiếm bằng xe tăng.
 - Sau mỗi ngày (mỗi lần chuyển lượt giữa hai đội xem như một ngày trôi qua) thì đơn vị nào đứng trên thành phố thuộc sở hữu của mình sẽ hồi phục 1 điểm máu (hp).
- Tính lượng máu còn lại sau giao tranh: đây là thể loại đánh theo trò chơi theo lượt vì vậy việc tính lượng máu còn lại cũng phải theo trình tự. Cụ thể đội bị tấn công sẽ tính trước đội tấn công. Đội tính máu trước (sẽ giảm hoặc giữ nguyên sau khi tính) sẽ bất lợi hơn do sát thương gây ra phụ thuộc vào lượng máu hiện tại. Nói cách khác, máu càng nhiều thì sát thương gây ra càng lớn, điều đó được thể hiện qua công thức tính lượng máu còn lại tại phần 3 tiếp theo.
- Trường hợp cả hai đơn vị (cũng là đơn vị cuối cùng của mỗi bên) giao tranh và cả hai cùng chết thì đơn vị chết trước xem như thua cuộc.

3. Công thức tính:

❖ Lượng máu sau giao tranh:

Ở đây phân biệt rõ đối tượng tấn công và đối tượng bị tấn công. Chủ động tấn công sẽ có lợi thế hơn vì kẻ bị tấn công cập nhật lượng máu trước mà cụ thể là lượng máu thường sẽ giảm sau cập nhật (sát thương gây ra lại phụ thuộc lượng máu hiện tại). Điều đó thể hiện rõ ở công thức tính lượng máu khi giao tranh:

$$\text{hp} = \text{hp} + \text{random}(0, 1) - (\text{dame} - \text{defence})$$

Trong đó:

- hp: lượng máu.
- random(0, 1): ngẫu nhiên 0 hoặc 1 (mô phỏng sự ngẫu nhiên trong chiến đấu)
- dame: lượng sát thương gây ra (xem chi tiết về thông số này ở bảng x)
- defence: chỉ số phòng thủ/giáp của đội cập nhật lượng máu. Chỉ số này chính là chỉ số bảo vệ của định hình tại vị trí đơn vị đang đứng (xem thêm bảng 2 trang 12)

*Xem thêm phần ví dụ cuối mục này để hiểu rõ công thức.

*Chú ý: Lượng máu là số nguyên dương, tối đa là 10.

❖ **Ma trận sát thương gây ra trong chiến đấu của mỗi loại đơn vị.**

Sau đây là ma trận sát thương chuẩn, lượng sát thương thật sự còn phụ thuộc vào lượng máu hiện tại theo công thức:

$$\text{dame} = \text{normalDame} * \text{hp} / 10$$

Trong đó:

- dame: lượng sát thương thực sự gây ra
- normalDame: lượng sát thương chuẩn
- hp: lượng máu hiện tại
- 10: số máu tối đa có thể có

		Bị tấn công		
		Pháo binh (Mech)	Bộ binh (Infantry)	Xe tăng (Tank)
Chủ động tấn công	Pháo binh (Mech)	2	3	3
	Bộ binh (Infantry)	2	2	1
	Xe tăng (Tank)	3	3	3

Bảng 3. Ma trận sát thương chuẩn (normal dame) gây ra trong chiến đấu của mỗi loại đơn vị.

Để dễ hình dung, ta có một số ví dụ sau.

Ví dụ 1: bộ binh máu 10 sẽ có sức mạnh hơn bộ binh máu 5 (xét cùng một địa điểm).

Ví dụ 2: Giả sử xe tăng (máu 10, đang đứng trên đồng bằng) chủ động tấn công bộ binh (máu 8, đang đứng trên sông). Khi đó, ta phải cập nhật lại lượng máu của cả hai phe (điều này là đúng với thực tế vì thường thì khi giao tranh xảy ra thì ít nhiều hai phe đều phải chịu một tổn thất nhất định):

- ✓ Cập nhật lượng máu cho đơn vị bị tấn công trước

$$hp_{\text{infantry}} = 8 + 1 - (3 \cdot 10 / 10 - 0) = 6$$

- ✓ Tiếp theo, cập nhật máu cho đơn vị bị tấn công (lúc này $hp_{\text{infantry}} = 6$).

$$hp_{\text{tank}} = 10 + 1 - (1 \cdot 6 / 10 - 0) = 10.4 \rightarrow 10$$

*Giả sử $\text{random}(0,1) = 1$

Như vậy trong trường hợp này, nhờ vào yếu tố may mắn ($\text{random}(0,1)$) mà xe tăng không bị tổn thất.

Ví dụ 3: Giả sử xe tăng (máu 10) tấn công bộ binh (máu 1) thì khả năng cao là khi cập nhật máu của bộ binh, bộ binh sẽ có lượng máu < 0 (hi sinh), lúc này không cần cập nhật tiếp máu của xe tăng nữa. Đây là cũng là trường hợp ngoài thực tế khi một đơn vị quá mạnh chiến đấu với một đơn vị yếu không có khả năng đánh trả.

❖ Điều kiện di chuyển

Để di chuyển một đơn vị đến một ô trên bản đồ, phải thỏa mãn ba điều kiện:

- ✓ Địa hình phù hợp
- ✓ Trong khoảng cách cho phép
- ✓ Ô đi đến còn trống

Mỗi đơn vị đều có một khoảng cách tối đa có thể di chuyển và mỗi đơn vị sẽ có thể di chuyển đến một số địa hình nhất định. Hai thông số điều kiện này có mô tả ở bảng x. Các đơn vị chiến đấu trang 11.

Sau đây là hình mô tả những ô có thể di chuyển (nền đỏ) của đơn vị được khoanh tròn xanh.



Hình 2. Mô tả đường đi có thể một đơn vị

❖ Điều kiện giao tranh

Để hai đơn vị có thể chiến đấu với nhau khi thỏa hai điều kiện:

- ✓ Khác phe
- ✓ Thuộc một trong bốn ô: đông, tây, nam, bắc. Như hình bên dưới, chỉ có bốn ô được khoanh đỏ là phe đỏ có thể tiếp cận để chiến đấu



Hình 3. Mô tả bốn đối tượng mà phe đỏ có thể tiếp cận chiến đấu

❖ Bản đồ mức độ ảnh hưởng (Influence map)

- Ảnh hưởng ở đây có thể hiểu là độ an toàn của cả một đội quân ta tương đương sự nguy hiểm cho đội địch.
- Mục đích của việc xây dựng bản đồ ảnh hưởng là nhằm cung cấp thông tin về tình hình thế trận trên bản đồ trò chơi, thông tin này sẽ được dùng cho phần trí tuệ nhân tạo ở mục sau.
- Như đã trình bày về đối tượng bản đồ ở phần 3.2. Ta có một bảng đồ chung là 150 (15 cột, 10 hàng) ô và bản đồ ảnh hưởng phải nói lên được sự ảnh hưởng của tất cả các đối tượng trong trò chơi gồm: các đơn vị chiến đấu và địa hình. Như vậy sẽ phải tính toán 3 bản đồ tương ứng với phe đỏ, phe xanh và địa hình. Cuối cùng là kết hợp chúng lại thành một bản đồ.
 - Công thức tính cho phe xanh/đỏ, ở mỗi đơn vị sẽ duyệt qua tất cả những ô mà nó có thể đi đến, ở mỗi ô đó được cập nhật theo công thức:

$$\text{value}_{\text{unit Red/Blue}} = 2 + \text{hp}$$

Trong đó:

- $\text{value}_{\text{unit Red/Blue}}$: giá trị ô của bản đồ ảnh hưởng phe xanh hoặc đỏ
- 2: hằng số
- hp: lượng máu của đơn vị này

Công thức tính cho địa hình

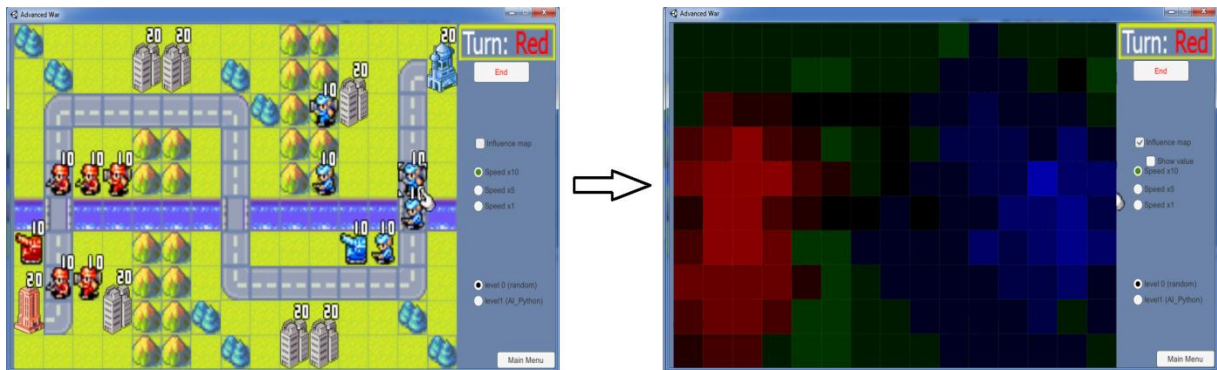
$$\text{value}_{\text{terrain}} = \text{defence}$$

Trong đó:

- $\text{value}_{\text{terrain}}$: giá trị ô của bản đồ ảnh hưởng
- defence: chỉ số bảo vệ (bảng x. trang 13)
- Hợp nhất thành một bản đồ ảnh hưởng:

$$\text{value}_{\text{InfluenceMap}} = \text{value}_{\text{terrain}} + \text{value}_{\text{unitRed}} + \text{value}_{\text{unitBlue}}$$

Để dễ hình dung ta có thể biểu diễn dạng đồ họa như sau:



Hình 4. Bản đồ ảnh hưởng (Influence map)

Trong đó:

- Màu đỏ: giá trị phe đỏ
- Màu xanh dương: giá trị phe xanh
- Màu xanh lá: giá trị địa hình
- Độ đậm nhạt: sự giao thoa các màu

4. Phân tích lượt đi – tương tác của người dùng với hệ thống.

Về phía người dùng, trình tự để tương tác lên một đơn vị mỗi lượt chơi:

1. Chờ đến lượt đi của mình
2. Chọn một đơn vị
3. Dự kiến điểm đến cho đơn vị đã chọn
4. Xác nhận điểm đến với hệ thống
5. Chờ hành động di chuyển (animation) của đơn vị đó kết thúc
6. Chọn một tùy chọn do hệ thống đưa ra:
 - “Chiếm giữ thành phố” (Capture)

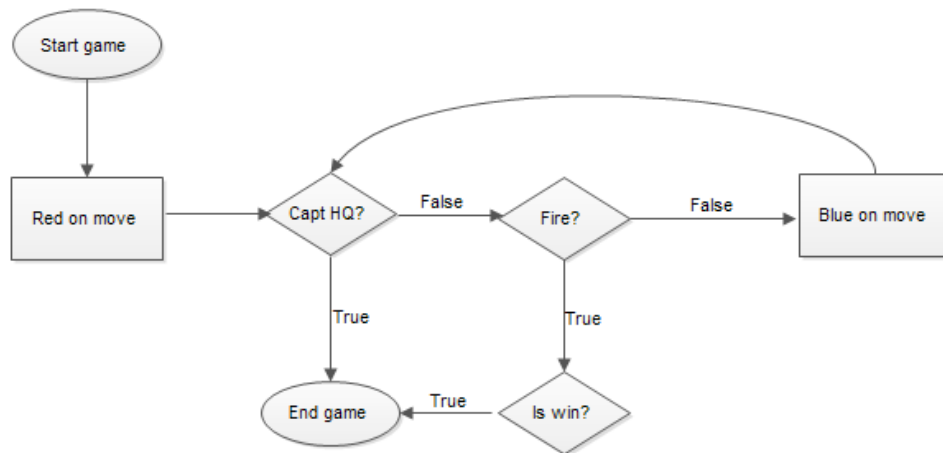
- “Chờ lệnh” (Wait)
- “Chiến đấu” (Fire)
- “Hủy bỏ hành động này (Cancel) ”

7. Chờ hệ thống xử lý tùy chọn ở. Tương tự cho những đơn vị tiếp theo cho đến hết và chuyển sang lượt đi đội đối phương.

Như phân tích ở trên là một hành động tương tác lên một đơn vị trong một lượt đi, trong một lượt đi bao gồm nhiều hành động như vậy tùy thuộc vào chiến lược và số lượng đơn vị còn lại trên bản đồ. Do tính phức tạp như trên, việc thiết kế hệ thống sao cho dễ cài đặt và quản lý đối tượng là quan trọng nhằm tránh tình trạng xung đột trong hệ thống trò chơi.

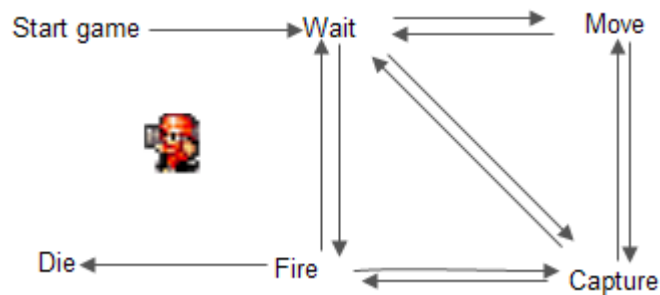
5. Các sơ đồ thiết kế ứng dụng.

- Sơ đồ tổng quát

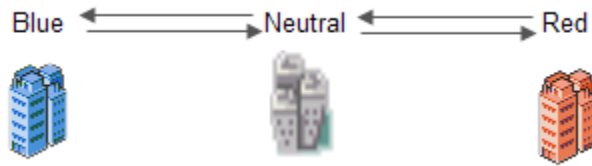


Hình 5. Sơ đồ tổng quát của trò chơi.

- Sơ đồ trạng trạng thái các đơn vị chiến đấu: trong quá trình sống của trò chơi, các đơn vị sẽ liên tục thay đổi trạng thái như sơ đồ sau:



Hình 6. Trạng máy máy của một đơn vị chiến đấu



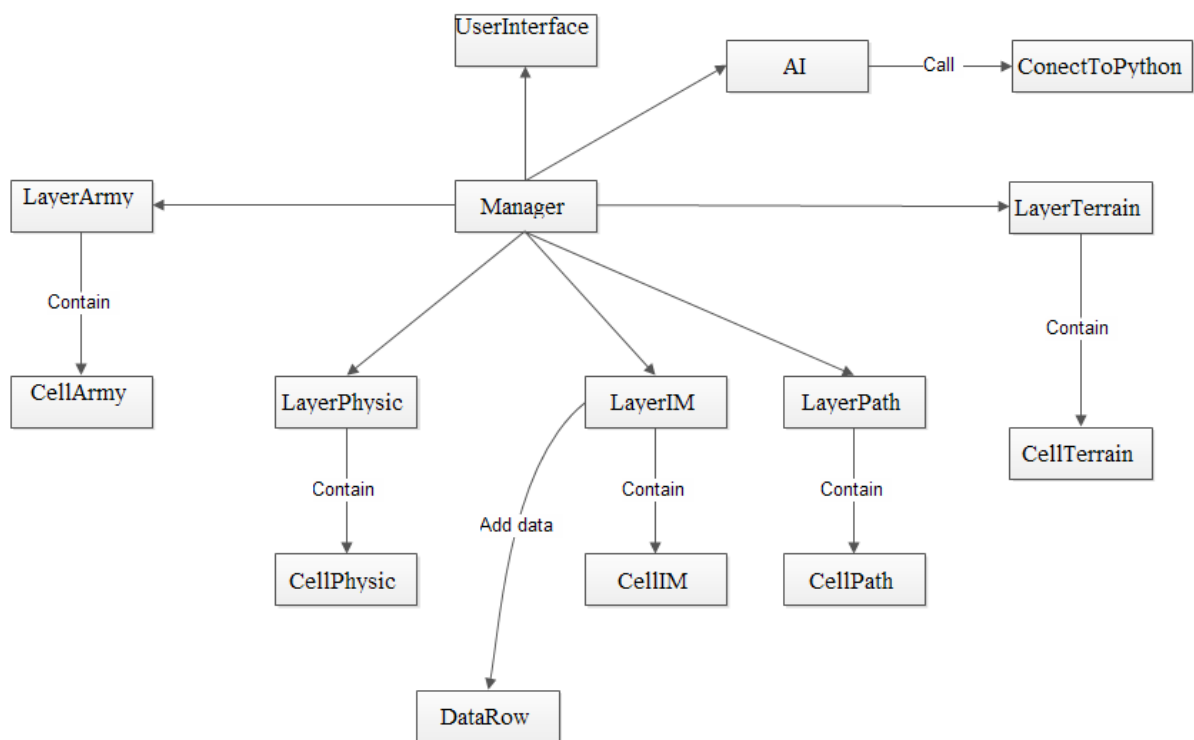
Hình 7. Trạng thái máy một thành phố

6. Thiết kế dữ liệu hướng đối tượng và cài đặt

6.1. Cấu trúc dữ liệu

Từ phân tích các đối tượng ở phần 3, ta có thể phân tích thành lớp đối tượng **Army** đại diện cho các đơn vị lính và lớp **Terrain** đại diện cho địa hình. Tuy nhiên, trò chơi này tương đối phức tạp, các bước đi tương đối dài dòng, nhiều trạng thái khác nhau. Vì vậy, cần phải xây dựng thêm một số lớp khác bên trong hệ thống. Tóm lại, ta có thể thiết kế các lớp như sau:

1. Lớp CellArmy
2. Lớp CellTerrain
3. Lớp CellPhysic
4. Lớp LayerPhysic
5. Lớp LayerArmy
6. Lớp LayerPath
7. Lớp Manager
8. Lớp UserInterface
9. Lớp LayerIM
10. Lớp DisplayIM
11. Lớp Option
12. Lớp AI_DecisionMaking
13. ArmyForAI
14. AI_Row
15. AI_DataRow
16. ConnectPython



Hình 8. Sơ đồ tổng quát thể hiện mối liên hệ giữa các lớp

Sau đây là mô tả chi tiết các lớp:

❖ **CellArmy**

Lớp này tạo ra các thể hiện chính là các đơn vị: bộ binh, pháo binh và xe tăng.

➤ **Thuộc tính:**

Stt	Tên	Kiểu	Mô tả
1	hp	Int	Lượng máu
2	move	Int	Tầm di chuyển tối đa
3	type	Enum Type (lập trình viên định nghĩa)	Loại của đơn vị này: pháo binh, bộ binh...
4	checkFire	Bool	Có khả năng bị tiếp cận? dùng cho trong giao tranh
5	isMove	Bool	Đã di chuyển

			trong lượt chơi này?
6	Anim	Animation (Unity type)	Đại diện cho hoạt hình (animation)

Bảng 4. Thuộc tính lớp CellArmy

➤ **Phương thức:**

- **void Awake():** Tạo các tham chiếu cho các biến
- **bool UpdateHp():** Cập nhật lại lượng máu cho đơn vị đồng thời trả về kiểu bool cho biết đối tượng này có bị tiêu diệt ($hp < 0$) sau khi cập nhật xong.
- **int GetDame():** Khi giao tranh với một đối thủ, sẽ cần tính lượng sát thương gây ra dựa vào các công thức tính ở mục 3.4.
- **void Capt():** thực hiện một hoạt hình (animation) chiếm giữ thành phố.
- **void Moved():** thiết lập một số thuộc tính liên quan nhằm nói lên đơn vị này đã di chuyển trong lượt đi này (tức không thể di chuyển thêm nữa, phải chờ đến lượt tiếp theo).
- **void Reset():** Ngược lại với hàm Moved().
- **void IntToSprite():** chuyển đổi số int hp sang dạng hình ảnh (sprite) để hiện thị cho người dùng trên màn hình.

❖ **CellTerrain**

Lớp này tạo ra các thể hiện chính là các địa hình: đồi núi, sông, rừng...

➤ **Thuộc tính:**

Stt	Tên	Kiểu	Mô tả
1	def	Int	Chỉ số bảo vệ đơn vị đang đứng trên địa hình này
2	type	Enum Type (lập trình viên định nghĩa)	Loại của địa hình này: đồi núi, sông, rừng...

Bảng 5. Thuộc tính lớp CellTerrain

➤ **Phương thức:** Không có

❖ **CellPhysic**

Lớp này là thể hiện cho mỗi ô trên bản đồ, có tác dụng nhận tương tác của người dùng lên bản đồ.

➤ **Thuộc tính:**

Stt	Tên	Kiểu	Mô tả
1	x	Int	Tọa độ x của ô trên bản đồ
2	y	Int	Tọa độ y của ô trên bản đồ
3	Parent	LayerPhycis	Tham chiếu đến lớp LayerPhycis

Bảng 6. Thuộc tính lớp CellPhysic

➤ **Phương thức:**

- **void OnEnable():** thiết lập tham chiếu đến lớp LayerPhycis
- **void OnMouseEnter():** được gọi khi người dùng *rê* chuột trái lên ô.
- **void OnMouseDown():** được gọi khi người dùng *nhấn* chuột trái lên ô.

❖ **LayerPhysic**

Lớp này chứa 150 phần tử, mỗi phần tử là một thể hiện của lớp **CellPhysic**

➤ **Thuộc tính:**

Stt	Tên	Kiểu	Mô tả
1	manager	Manager	Tham chiếu đến Manager
2	refUserInterface	UserInterface	Tham chiếu đến UserInterface
3	layerPhysic	CellPhysic[,]	Tham chiếu đến lớp CellPhysic

Bảng 7. Thuộc tính lớp LayerPhysic

➤ **Phương thức:**

- **void OnMouseEnter(int x, int y):** nhận thông tin và xử lý từ phương thức OnMouseEnter() của CellPhysic tương ứng.
- **void ClickOnMap(int x, int y):** nhận thông tin và xử lý từ OnMouseDown() của lớp CellPhysic tương ứng.

❖ **LayerArmy**

Quản lý 150 phần tử là các thể hiện của lớp CellArmy

➤ Thuộc tính:

Stt	Tên	Kiểu	Mô tả
1	refLayerPath	LayerPath_	Tham chiếu đến LayerPath
2	layerArmy	CellArmy_ [,]	Tham chiếu đến CellArmy
3	listArmy	List<ArmyForAI>	Danh sách ArmyForAI
4	refLayerIM	LayerIM	Tham chiếu đến LayerIM
5	refDataRows	AI_DataRow	Tham chiếu đến DataRows
6	listEnemyCanFire	List<ArmyForAI>	Danh sách ArmyForAI
7	refUserInterface	UserInterface	Tham chiếu đến UserInterface
8	manager	Manager_	Tham chiếu đến Manager
9	sprFire	SpriteRenderer	Hiệu ứng “vụ nổ” khi tiêu diệt một đơn vị
10	fire	Sprite[]	Mảng sprite của “vụ nổ”

Bảng 8. Thuộc tính lớp LayerArmy

➤ Phương thức:

- **List<ArmyForAI> GetListArmy(TeamType type):** trả về danh sách các đơn vị có kiểu type (phương thức này cho AI).
- **List<ArmyForAI> GetListCanFire():** Trả về danh sách các đơn vị có thể bị tấn công bởi đơn vị đang chọn hiện tại (phương thức này cho AI).
- **int[] AI_GetUnitBlueMaxHp():** trả về vị trí (x, y) của đơn vị đội xanh có chỉ số máu cao nhất hiện tại (phương thức này cho AI).
- **void AddValueToInfluenceMap2(Manager_.TypeArmy currentType):** cập nhật giá trị cho influence map.
- **void AddValueIM_Blue(Manager_.TypeArmy typeEnemy = Manager_.TypeArmy.none):** cập nhật giá trị cho influence map của đội xanh

- **void AddValueIM_Red(Manager_.TypeArmy type = Manager_.TypeArmy.none):** cập nhật giá trị cho influence map của đội xanh.
- **IEnumerator AnimFire(int x, int y, bool waitA2 = false):** hiệu ứng vụ nổ khi tiêu diệt một đơn vị
- **int IsWin (ref int numRed, ref int numBlue):** trả về giá trị 1 (đỏ thắng) hay -1 (xanh thắng)
- **bool IsFire(int x, int y):** đơn vị tại (x, y) có thể bị tấn công bởi đơn vị đang chọn hay không?
- **bool Select(Manager_.OnMove onMove):** người dùng có thể nhấn chọn một đơn vị nào không?
- **void Reset():** các đơn vị trở về trạng thái ban đầu.
- **bool IsNull(int x, int y):** vị trí (x, y) có trống hay không?
- **void AddHp(int x, int y, int value):** thêm máu cho một đơn vị
- **Manager_.TypeArmy GetType(int x, int y):** trả về kiểu của đơn vị tại vị trí (x, y)
- **int GetHp(int x, int y):** trả về lượng máu tại vị trí (x, y)
- **void ConfirmPosition():** Xác nhận sau di chuyển của người dùng.
- **void Capt():** chiếm giữ một thành phố
- **IEnumerator WaitCapt():** đợi hành động chiếm giữ thành phố
- **void Wait():** trả về phản hồi cho manager
- **bool Fire(int xEnemy, int yEnemy, int defEnemy, int defCurrent):** thực hiện chiến đấu
- **void Cancel():** người dùng hủy bước đi
- **void AnimMove(bool isAnim = true):** di chuyển một đơn vị với tùy chọn có animation.
- **string GetDataRowCurrent(int xCenter, int yCenter, int xStat, int yStart):** trả về một dòng dữ liệu trong xây dựng tập học của trí tuệ nhân tạo
- **void MoveWithoutAnim():** di chuyển một đơn vị ngay lập tức (không có animation)

- **IEnumerator MoveWithAnim():** di chuyển một đơn vị với animation
- **bool CheckFire(Manager_.OnMove oM):** kiểm tra có thể tấn công 4 ô xung quanh được hay không?

❖ **LayerPath:**

Vẽ đường đi của các đơn vị, quản lí 150 ô CellPath

➤ **Thuộc tính:**

Stt	Tên	Kiểu	Mô tả
1	iconSelect, iconSelectFire	Sprite	Hình ảnh của biểu tượng chọn đơn vị trên màn hình
2	select	Transform	Vị trí biểu tượng chọn
3	layerPath	CellPath_[,]	Tham chiếu đến CellPath
4	refLayerTerrain	LayerTerrain_	Tham chiếu đến LayerTerrtain
5	refLayerArmy	LayerArmy_	Tham chiếu đến LayerArmy
6	iconFire	bool	Trạng thái biểu tượng chọn đơn vị (select/fire)
7	listListPaths	List<List<Vector2>>	Danh sách những đường đi của một đơn vị
8	listArmyMove	List<Vector2>	Đường đi ngắn nhất của một đơn vị

Bảng 9. Thuộc tính lớp LayerPath

➤ **Phương thức:**

- **void Wait(),void Capt(),void Fire():** xóa các danh sách đường đi của cả LayerPath.
- **bool IsCheck(int row, int col):** vị trí (row, col) có được đánh dấu?
- **void SelectFire(bool isFire):** Thay đổi biểu tượng chọn theo trạng thái isFire
- **void ClearArea():** bỏ đánh dấu tất cả
- **void ClearPathAndArea():** bỏ đánh dấu tất cả, xóa các nút đường đi trên bản đồ
- **void DrawArea():** Vẽ khu vực có thể di chuyển đến

- **void SetIconSelectArmy(int x, int y):** đặt lại vị trí biểu tượng chọn đơn vị
- **List<Position2D> GetListPositionCanMoveTo():** trả về danh sách các điểm phân tử của đường đi
- **void DrawPath():** vẽ đường đi
- **void FindPath(int _xStart, int _yStart, int move, List<Vector2> _path):** tìm đường đi ngắn nhất
- **List<Position2D> GetListArea():** trả về danh sách điểm mà đơn vị có thể đi
- **void FindArea(Manager_.TypeArmy _type, int _move, int _x, int _y):** tìm tất cả các vị trí có thể đi của đơn vị.

❖ **Manager:**

Quản lí chung toàn bộ hệ thống

➤ **Thuộc tính**

Stt	Tên	Kiểu	Mô tả
1	numDay	int	Số ngày
2	one, two, three, four, five, six, seven, eight, nine, zero, sprRedCity, sprBlueCity, sprNoneCity	Sprite	Các sprite để cho phân giao diện
3	selectedX, selectedY, cursorX, cursorY, newX, newY	static int	Vị trí con trỏ chuột trên bản đồ
4	savedArmy	static CellArmy	Đơn vị đã chọn
5	saveState	StateGame	Lưu trạng thái
6	refLayerPath	LayerPath_	Tham chiếu đến LayerPath
7	refLayerArmy	LayerArmy_	Tham chiếu đến LayerArmy
8	refLayerTerrain	LayerTerrain	Tham chiếu đến LayerTerrain
9	refIM	LayerIM	Tham chiếu đến LayerIM
10	refOption	Option	Tham chiếu đến Option
11	refUserInterface	UserInterface	Tham chiếu đến

			ManagerInterface
12	refAI_DecisionBlue, refAI_DecisionRed	AI_DecisionMaking	Tham chiếu đến AI
13	refDataRows	AI_DataRow	Tham chiếu đến AI_DataRow

Bảng 10. Thuộc tính lớp Manager

➤ **Phương thức:**

- **void LoadGame():** khởi tạo các tham số và đối tượng AI
- **void Physic_OnMouseEnter(int x, int y):** Xử lý khi người dùng rê chuột trên bản đồ
- **void Physic_OnClick(int x, int y):** Xử lý khi người dùng nhấn lên bản đồ
- **void FeedBack():** Xử lý các phản hồi từ các đối tượng khác
- **void BlueWin():** Xử lý khi đội xanh thắng
- **void RedWin():** Xử lý khi đội đỏ thắng
- **void CheckWin():** Kiểm tra có đội thắng thua hay không?
- **void Cancel():** Xử lý khi người dùng hủy bước đi
- **void Option_Fire():** Xử lý khi yêu cầu chiến đấu
- **void Option_Wait():** Xử lý khi yêu cầu chờ lệnh
- **void Option_Capt():** Xử lý khi yêu cầu chiếm thành phố
- **void Option_End():** Xử lý khi yêu kết thúc ngày
- **void IMRed():** vẽ influence map

❖ **ManagerInterface**

Xử lý về giao diện người dùng

➤ **Thuộc tính**

Stt	Tên	Kiểu	Mô tả
1	recordeStep, pVsCom, comVsCom, speed_1x, speed_10x, speed_5x, level0, level1	Toggle	Các nút toggle trên giao diện
2	speedUp	int	Tốc độ trò chơi
3	aiDecisionBlue, aiDecisionRed	AI_DecisionMaking	Trí tuệ nhân tạo hỗ trợ quyết định nước đi cho máy.

Bảng 11. Thuộc tính lớp UserInterface

➤ **Phương thức**

- **void PsCom():** người đánh với máy
- **void ComVcCom():** máy đánh với máy
- **void Speed1x():** tốc độ 1x
- **void Speed5x():** tốc độ 5x
- **void Speed10x():** tốc độ 10x
- **void Level0():** quyết định của máy là ngẫu nhiên
- **void Level1():** quyết định của máy được kết nối với python

❖ **LayerIM**

Tính toán giá trị đồ ảnh hưởng (influence map)

➤ **Thuộc tính:**

Stt	Tên	Kiểu	Mô tả
1	display	DisplayIM[,]	Tham chiếu đến 150 phần tử DisplayIM
2	red, blue, value	Float[,]	Các mảng hai chiều gồm 150 phần tử chứa giá trị của influence map

Bảng 12. Thuộc tính lớp LayerIM

➤ **Phương thức**

- **void SetTerraint(int r, int c, int value):** đặt giá trị cho địa hình
- **void SetRed(Manager_.OnMove onMove, int r, int c, int value):** đặt giá trị cho đội đỏ
- **void SetBlue(Manager_.OnMove onMove, int r, int c, int value):** đặt giá trị cho đội xanh
- **void ValueToRender():** đặt giá trị cho hiển thị lên màn hình
- **void ResetAllZero():** đặt tất cả giá trị về 0
- **void CombineRedBlueTerr():** kết hợp các giá trị của đội xanh, đỏ và địa hình
- **float[] GetIM5x5(int xCenter, int yCenter):** trả về một phần (5x5) của influence map có tâm tại (xCenter, int yCenter)
- **void Nageitive(Manager_.OnMove oM):** nghịch đảo giá trị (âm thành dương) của đội khác với oM

❖ DisplayIM

Hiển thị influence map lên màn hình

➤ Thuộc tính:

Stt	Tên	Kiểu	Mô tả
1	manager	Manager	Tham chiếu đến Manager
2	value1, value2, mainColor	SpriteRenderer	Giá trị (dạng sprite) hiển thị lên màn hình
	valueR, valueB, terrain	int	Giá trị ô

Bảng 13. Thuộc tính lớp DisplayIM

➤ Phương thức:

- **void Reset():** đặt lại các giá trị bằng 0
- **void SetDisplay(int valueCell, float colorR, float colorTerrain, float colorB):** chuyển giá trị kiểu int sang giá trị màu
- **void ShowLabel():** hiển thị/ẩn giá trị trên màn hình
- **void IntToSprite(int value):** chuyển giá trị kiểu int sang kiểu sprite

❖ AI_DecisionMaking

Hỗ trợ ra quyết định cho máy các bước đi

➤ Thuộc tính

Stt	Tên	Kiểu	Mô tả
1	teamType	LayerArmy_.TeamType	Đội xanh hay đội đỏ
2	typeOp	Option.TypeOption	Kiểu tùy chọn: wait, fire, capt
	refManager	Manager_	Tham chiếu đến Manager
	refLayerArmy	LayerArmy_	Tham chiếu đến LayerArmy
	refLayerPath	LayerPath_	Tham chiếu đến LayerPath
	listArmy	List<ArmyForAI>	Danh sách ArmyForAI
	refOption	Option	Tham chiếu đến Option
	refUserInterface	UserInterface	Tham chiếu đến UserInterface
	refPython	ConnectPython	Tham chiếu đến ConnectPython

	refLayerIM	LayerIM	Tham chiếu đến LayerIM
	refDataRows	AI_DataRow	Tham chiếu đến AI_DataRow
	posFrom, posTo	Position2D	Vị trí
	armySelected	CellArmy_	Một đơn vị đã được chọn
	waitForAmim, waitForPython	bool	Chờ các animation hay không?
	waitFeedbackPython	static bool	Chờ kết nối đến python
	listPosArea_forMove	List<Position2D>	Danh sách các điểm trên bước đi của một đường đi

Bảng 14. Thuộc tính lớp AI_DecisionMaking

➤ **Phương thức**

- **IEnumerator Play():** Bắt đầu chế độ tự chơi của máy
- **bool _SelectArmy():** chọn một đơn vị
- **IEnumerator Move_ConnecPython():** di chuyển dựa trên quyết định của python đưa ra
- **void _Move_AI_random():** di chuyển một cách ngẫu nhiên
- **void SetChoiceOption(Option.TypeOption typeOp):** lựa chọn sau khi di chuyển

❖ **AI_Row**

Mỗi dòng dữ liệu trong tập dữ liệu huấn luyện cho trí tuệ nhân tạo

➤ **Thuộc tính**

Stt	Tên	Kiểu	Mô tả
1	influenceMap	float[]	5x5 phân tử giá trị influence map
2	hp	float	Giá trị hp
	posStart, posEnd	moveToHere	Vị trí bắt đầu, kết thúc một bước đi
	type	Manager_.TypeArmy	Kiểu của đơn vị này

Bảng 15. Thuộc tính lớp AI_Row

➤ **Phương thức: không có**

❖ **AI_DataRow**

Xử lý và ghi thành tệp lên đĩa cứng

➤ **Thuộc tính:**

Stt	Tên	Kiểu	Mô tả
1	rowsRed, rowsBlue	List<AI_Row>	Dòng dữ liệu đội xanh, đỏ

Bảng 16. Thuộc tính lớp AI_DataRow

➤ **Phương thức:**

- **void AddDataRow(bool isRed, Manager_TypeArmy _type, float[] _influenceMap, float _hp, int _posEnd, int _posStart, bool moveHere):** thêm một dòng dữ liệu
- **void SaveToDisk(bool isRed):** lưu tệp tin

❖ **ConnectPython**

Thực hiện kết nối đến python

➤ **Thuộc tính:**

Stt	Tên	Kiểu	Mô tả
1	notificatonError	GameObject	Thông báo không thể kết nối
	pathPythonFile	Text	Đường dẫn đến tệp python
	receiveFromPython, sentToPython	string	Chuỗi Unity gửi/nhận từ Python
	result	bool	Đã nhận được từ python
	client	UdpClient	
	ip	IPEndPoint	
	r, s	byte[]	
	data	byte[]	
	timer	float	
	waitPython	Bool	
	timeOut	Float	
	firsrLoad	bool	

Bảng 17. Thuộc tính lớp ConnectPython

➤ **Phương thức**

- **void Send(bool feedback):** gửi thông tin đến python
- **void Recive():** nhận thông tin
- **IEnumerator FailedConnect():** kiểm tra kết nối.

6.2. Giải thuật

Sau đây là một số giải thuật trong quá trình thiết kế trò chơi này.

❖ **Giải thuật tìm tất cả những ô có thể đi của một đơn vị đã chọn**

Giải thuật này được cài đặt bằng một hàm đệ quy, theo như luật chơi thì vị trí có thể đi đến phải đảm bảo ba điều kiện:

- + Nằm trong khoảng cách tối đa
- + Vị trí đó còn trống
- + Vị trí đó hợp lệ (Xem thêm bảng 1 trang 11)

Tại một điểm cho trước,

```
TimKhuVuc(vị trí hiện tại, move){
    if move<0
        return;
    else{
        if(vị trí hợp lệ)
            Đánh dấu vị trí này;
            Tiếp tục gọi lại hàm TimKhuVuc() tại 4 vị
            trí trên, dưới, trái, phải với move -= 1;
        }
    }
    move: giá trị bước đi tối đa.
```

Sau khi gọi hàm trên, những vị trí được đánh dấu là những ô có thể đi.

❖ **Giải thuật đường đi ngắn nhất từ vị trí một đơn vị đến vị trí con trỏ chuột trên màn hình.**

Sau khi đã tìm được khu vực có thể đi, ta tìm đường đi ngắn nhất. Hàm cũng được cài đặt một cách đệ quy

Từ một vị trí bắt đầu (đơn vị đã chọn) cho trước và một vị trí kết thúc (con trỏ chuột), duyệt qua bốn hướng trên, dưới, trái, phải :

```
TimViTri(vị trí hiện tại){
    Đánh dấu vị trí hiện tại;
    if(vị trí hiện tại == vị trí trỏ chuột)
        return;
    else{
        gọi lại hàm với 4 vị trí mới: trên, dưới,
        trái, phải;
    }
}
```

Sau khi gọi hàm trên, những vị trí được đánh dấu sẽ tạo thành đường đi ngắn nhất.

❖ **Giải thuật tính giá trị của bản đồ ảnh hưởng của các đơn vị (influence map)**

```

Duyệt qua tất cả các đơn vị{
    Với mỗi đơn vị duyệt qua tất cả các ô có thể đi đến{
        Thêm giá trị vào influence map tại vị trí tương ứng;
    }
}

```

6.3. Cài đặt

Như đã trình bày ở phần giới thiệu, tất cả sẽ được cài đặt bằng Unity với ngôn ngữ lập trình C#, sản phẩm cuối cùng là một tệp exe và thư mục data tương ứng chạy trên nền windows 64 bit. Riêng phần trí tuệ nhân tạo sẽ cài đặt bằng ngôn ngữ python.

7. Trí tuệ nhân tạo

7.1. Chiến lược chơi

Về chiến lược chơi, để có một quyết định nước đi thông minh phụ thuộc vào nhiều yếu tố như:

- Chiến lược: tấn công, phòng thủ?
- Quản lí các nguồn tài nguyên hiện có như thế nào?
- Thứ tự: trong một lượt đi, đơn vị nào đi trước, đơn vị nào đi sau?
- Những quyết định: Fire, Wait, Capt sẽ thực hiện như thế nào?

Vì vậy, số lượng cách để ra một bước đi là rất nhiều.

Về mỗi bước đi, nhận thấy có một số đặc trưng trả lời câu hỏi sau:

- Đơn vị nào được chọn?
- Di chuyển đơn vị từ đâu đến đâu?
- Hành động sau đó: Wait/Capt/Fire?
- Lượng máu hiện tại?
- Trường hợp Fire thì đơn vị nào để tấn công?

Đặc trưng: tên đơn vị, điểm bắt đầu, điểm kết thúc, lượng máu.

Tuy nhiên, những bước đi ở trên phụ thuộc vào một cái gọi là “tình hình” hiện tại của trận đấu hay sự ảnh hưởng của các đội trên bản đồ. Để nói lên được điều đó, chúng tôi sử dụng bản đồ ảnh hưởng (influence map) để nói lên đặc trưng “tình hình” của trận đấu. Đề xuất này dựa trên một số nghiên cứu và ứng dụng Modular Tactical Influence Maps [1], Survey of Real-Time Strategy Game AI [2], Adaptive Spatial Reasoning for Turn-based Strategy Games [3]. Cụ thể cách tính giá trị influence map đã được trình bày ở trang 17.

7.2. Ứng dụng học máy vào trò chơi

a) Mỗi dòng dữ liệu

Từ phần 7.1 đã trình bày, phần này sẽ ứng dụng học máy để hỗ trợ máy ra quyết định một bước đi. Nhắc lại đặc trưng một bước đi (có kết hợp influence map):

25 giá trị influence map , lượng máu , điểm bắt đầu, điểm kết thúc, Tên đơn vị

- ⇒ Có 29 giá kiểu nguyên (tên đơn vị được chuyển sang số nguyên)
- ⇒ Chú ý rằng influence map có 150 giá trị tương ứng 150 ô trên bản đồ, tuy nhiên ở đây chỉ lấy giá trị 25 ô xung quanh tâm tọa độ là điểm kết thúc (điểm đến).



Hình 9. Ghi chú về dòng dữ liệu

Một ví dụ:

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 12, 24, 24, 0, 0, 24, 48, 48, 0, 0, 0, 0, 0, 0, 10, 17, 33, 1]
```

Tuy nhiên, ở trên chỉ mới là dữ liệu vào, chúng ta chưa có nhãn mỗi dòng đó là gì?

Ta có:

Mỗi bước đi => một hàng dữ liệu (29 giá trị) tương ứng.



Hình 10. Các nhãn của dòng dữ liệu được tạo ra

Từ hình x, đơn vị có 13 ô có thể đi được, nếu chọn một trong 13 ô đó để đi thì ô đi đến đó có nhãn là 1, tất cả ô còn lại nhãn là 0.

Cũng trong hình trên, sau khi di chuyển đơn vị đó, sẽ có đến 13 dòng dữ liệu được tạo ra. Trong đó 1 dòng nhận 1 và 12 dòng nhận 0.

\Rightarrow Tập dữ liệu tạo ra không cân bằng.

b) Tạo dữ liệu cho huấn luyện

Để đạt hiệu quả trong học máy, chúng ta cần một lượng lớn dòng dữ liệu để huấn luyện. Cách đơn giản nhất là lập trình để tự máy đánh với máy, sau kết thúc mỗi trận, những bước đi của đội thắng sẽ được lưu lại làm tập huấn luyện.

c) Giải thuật học

Ở đây, sử dụng perceptron đa tầng: 2 tầng ẩn, mỗi tầng 10 nút.

d) Kết hợp Unity và python

Xây dựng mô hình perceptron sẽ được thực hiện bằng ngôn ngữ python.

e) Sử dụng mô hình

Sau khi có được mô hình đã huấn luyện, mỗi khi đến lượt đi của máy, máy sẽ duyệt qua tất cả các ô có thể đi và gửi đầu vào (29 giá trị input) cho python, sau đó python trả về kết quả 0 hoặc 1. 1 tương ứng với việc nên đi ở vị trí này, ngược lại nhận giá trị 0 nghĩa là duyệt tiếp ô tiếp theo. Trên ý tưởng là vậy, tuy nhiên khi triển khai lại có kết quả không hợp lý, điều này được trình bày rõ ở phần kết luận-hạn chế và hướng phát triển.

f) Kiểm thử

Tiến hành kiểm thử k-fold với $k = 10$, thực hiện 10 đợt kết quả trung bình 73%

KẾT LUẬN

1. Kết quả đạt được

- Lập trình trò chơi Advance War
- Triển khai được học máy vào trò chơi kết hợp Unity và ngôn ngữ Python

2. Hạn chế và hướng phát triển

Phần ứng dụng trí tuệ nhân tạo chỉ mới cài đặt thử nghiệm, ra quyết định không hiệu quả. Cần phải nghiên cứu thêm vì còn nhiều vấn đề chưa chính xác, đặc biệt là phần trí tuệ nhân tạo. Sau đây là một số vấn đề cần nghiên cứu thêm:

- 29 giá trị trong mỗi dòng tập dữ liệu học có thật sự đủ để đại diện cho đặc trưng một bước đi?
 - Chỉ lấy 25 giá trị của influence map là đủ, và vị trí của tâm 25 giá trị này đã hợp lí không?
 - Việc chuyển tên đơn vị thành kiểu số nguyên như vậy là có hợp lí.
 - Độ chính xác của mô hình là 73% là quá thấp, cần điều chỉnh để cao hơn.
- Về phần lập trình trò chơi: Về cơ bản đã hoàn thành, tuy nhiên các thiết kế, xây dựng lớp còn khá phức tạp, chưa rõ ràng, gây khó hiểu cho việc nâng cấp sau này.

Tài liệu tham khảo

[1] Modular Tactical Influence Maps - Dave Mark

http://www.gameaipro.com/GameAIPro2/GameAIPro2_Chapter30_Modular_Tactical_Influence_Maps.pdf

[2] *A Survey of Real-Time Strategy Game AI Research and Competition in StarCraft*

Santiago Ontañón, Gabriel Synnaeve, Alberto Uriarte, Florian Richoux, David Churchill, Mike Preuss

<https://hal.archives-ouvertes.fr/hal-00871001/document>

[3] *Adaptive Spatial Reasoning for Turn-based Strategy Games*

Maurice Bergsma and Pieter Spronck

<https://www.aaai.org/Papers/AIIDE/2008/AIIDE08-027.pdf>

Giáo trình Nguyên lý máy học 2012 Nhà xuất bản Đại học Cần Thơ

Ts.Đỗ Thanh Nghị-Ts.Phạm Nguyên Khang

Giáo trình Các hệ cơ sở tri thức và khai phá dữ liệu 2012 Nhà xuất bản Đại học Cần Thơ

Ts.Đỗ Thanh Nghị-Ts.Lê Thanh Vân

PHỤ LỤC

1. Hướng dẫn chơi.

Luật chơi đã trình bày ở mục 1

Chú ý trường hợp kết nối python thì phải khởi động python (vai trò server).

Một số ghi chú về giao diện:



Hình 11. Hướng dẫn chơi

2. Hướng dẫn đọc và chỉnh sửa mã nguồn.

- Thay đổi phần trí tuệ nhân tạo tại script: `AI_DecisionMaking.cs`
- Thay đổi xây dựng mô hình ở python (nằm trong thư mục Python): `serverPy.py`
- Tập map: trong thư mục `StreamingAssets` có tập “map”, tập này quy định bản đồ như thế nào. Có thể mở với notepad, chi tiết xem thêm `CreateMap.cs`