TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KHOA HỌC MÁY TÍNH



BÁO CÁO ĐỒ ÁN MÔN TRÍ TUỆ NHÂN TẠO

BẮN XE TĂNG

*(đã chỉnh sửa theo yêu cầu giảng viên)*

Lớp: CS106.I22

Giảng viên hướng dẫn: HUỲNH THỊ THANH THƯƠNG

Nhóm thực hiện:

1. Võ Thành Cường 16521497
2. Nguyễn Phạm Long Duy 16520299
3. Nguyễn Trần Quốc Toàn 16521257
4. Phạm Ngọc Phúc Thuần 16521206

*Tp. Hồ Chí Minh, tháng 07 năm 2018*

**MỤC LỤC**

[LỜI NÓI ĐẦU 3](#_Toc518853059)

[HỢP ĐỒNG NHÓM 4](#_Toc518853060)

[BẢNG PHÂN CÔNG CÔNG VIỆC 8](#_Toc518853061)

[BẢNG ĐÁNH GIÁ NỘI BỘ NHÓM 9](#_Toc518853062)

[CHƯƠNG 1: GIỚI THIỆU ĐỒ ÁN 11](#_Toc518853063)

[1. LỊCH SỬ TRÒ CHƠI: 11](#_Toc518853064)

[2. MÔ TẢ TRÒ CHƠI: 12](#_Toc518853065)

[CHƯƠNG 2: CƠ SỞ LÝ THUYẾT 17](#_Toc518853066)

[1. NGÔN NGỮ LẬP TRÌNH C#: 17](#_Toc518853067)

[2. LÝ THUYẾT ÁP DỤNG: 17](#_Toc518853068)

[CHƯƠNG 3: PHÂN TÍCH VÀ THIẾT KẾ 24](#_Toc518853069)

[1. PHÂN TÍCH BÀI TOÁN: 24](#_Toc518853070)

[2. THIẾT KẾ CẤU TRÚC DỮ LIỆU VÀ TRẠNG THÁI: 24](#_Toc518853071)

[3. XÂY DỰNG BÀI TOÁN: 28](#_Toc518853072)

[CHƯƠNG 4: ỨNG DỤNG 45](#_Toc518853073)

[1. GIỚI THIỆU CHƯƠNG TRÌNH ỨNG DỤNG: 45](#_Toc518853074)

[2. HƯỚNG DẪN CÀI ĐẶT TRÒ CHƠI: 49](#_Toc518853075)

[3. HƯỚNG DẪN CÁCH CHƠI: 51](#_Toc518853076)

[CHƯƠNG 5: KẾT LUẬN 54](#_Toc518853077)

[1. KẾT QUẢ ĐẠT ĐƯỢC: 54](#_Toc518853078)

[2. HẠN CHẾ: 54](#_Toc518853079)

[3. HƯỚNG PHÁT TRIỂN: 54](#_Toc518853080)

[TÀI LIỆU THAM KHẢO 56](#_Toc518853081)

[NỘI DUNG ĐÃ CHỈNH SỬA THEO YÊU CẦU 57](#_Toc518853082)

[LỜI CẢM ƠN 58](#_Toc518853083)

# LỜI NÓI ĐẦU

Cách mạng công nghiệp lần thứ tư (hay còn gọi là *Cách mạng công nghiệp 4.0*) đánh dấu sự phát triển mạnh mẽ của kỷ nguyên khoa học – kỹ thuật – công nghệ ngày nay. Cuộc cách mạng phát triển ba trụ cột chính là kỹ thuật số, công nghệ sinh học, vật lý. Trong đó, Trí tuệ nhân tạo (AI) đang nổi lên và trở thành tâm điểm của cuộc cách mạng lần này.

Trí tuệ nhân tạo tạo từng được biết đến qua cuộc đua giữa con người và trí tuệ nhân tạo trong các trò chơi như cờ vua, cờ vây, game online, hay thậm chí cả các cuộc đố vui. Nếu như trước đây chúng ta chỉ thấy được những ứng dụng của trí tuệ nhân tạo qua các trò chơi phổ biến, năm 2018, trí tuệ nhân tạo đã có mặt ở tất cả các lĩnh vực và đời sống của con người, ví dụ như các trợ lý ảo được cài đặt trên các thiết bị điện thoại di động thông minh ngày nay. Theo một báo cáo nghiên cứu gần đây, các nhà phân tích thị trường dự báo thị trường trí tuệ nhân tạo sẽ đạt tới 47 tỷ USD vào năm 2020. Vì thế, trí tuệ nhân tạo đang nhanh chóng trở thành nền tảng cốt lõi của các ngành công nghiệp trên thị trường hiện nay.

Nhờ khoa Khoa học Máy tính và cô Huỳnh Thị Thanh Thương, nhóm đã có cơ hội thực hiện đồ án này và thêm một bước gần hơn với lý tưởng máy móc có trí thông minh. Từ những kiến thức nền tảng được truyền tải ở mỗi buổi học và việc tìm tòi, học hỏi, nghiên cứu các kiến thức mới, nhóm đã hoàn thành đồ án Bắn xe tăng. Trong quá trình thực hiện đồ án, những sai sót là không thể tránh khỏi. Chính vì vậy, nhóm mong nhận được những ý kiến đóng góp từ giảng viên để đồ án được hoàn thiện hơn.

# HỢP ĐỒNG NHÓM

1. THÀNH VIÊN NHÓM:

|  |  |  |  |
| --- | --- | --- | --- |
| STT | HỌ VÀ TÊN | MSSV | VAI TRÒ |
| 1 | Nguyễn Trần Quốc Toàn | 16521257 | Nhóm trưởng |
| 2 | Nguyễn Phạm Long Duy | 16520299 | Thành viên |
| 3 | Võ Thành Cường | 16521497 | Thành viên |
| 4 | Phạm Ngọc Phúc Thuần | 16521206 | Thành viên |

1. HỢP ĐỒNG NHÓM:
   1. Mục tiêu:

* Cùng hợp tác, cùng giúp đỡ lẫn nhâu trong tất cả các vấn đề mà nhóm đề ra.
* Có ý thức trách nhiệm với những gì mình làm, rèn luyện kỹ năng sống lẫn thái độ làm việc nhóm.
* Sau khi kết thúc môn học, sản phẩm có thể sử dụng được và các thành viên có thể vận dụng các kỹ năng lập trình và làm việc vào đời sống.
  1. Hiệp định:
* Ra nguyên quyết định theo nguyên tắc đồng thuận.
* Hoàn thành công việc đúng thời hẹn.
* Đề cao sự tôn trọng giữa các thành viên trong nhóm.
* Mục đích cuối cùng đạt được là hoàn thành tốt công việc nhóm.
  1. Các quy định và kỹ cương của nhóm:
* Có mặt đầy đủ trong các buổi họp nhóm.
* Tích cực trao đổi đóng góp ý kiến cá nhân để hoàn thiện đồ án
* Chịu trách nhiệm về những việc mình được phân công, hoàn thành hết khả năng của bản thân.
* Có vấn đề phát sinh lập tức thông báo cho nhóm để kịp thời chỉnh sửa.
* Tôn trọng sự hợp tác của mỗi thành viên trong nhóm.
* Hình thức họp nhóm: chủ yếu họp mặt online bằng Messenger, Teamview, … và cũng có những buổi họp mặt cụ thể (sẽ thông báo buổi họp trước 3 ngày để các bạn chuẩn bị).

1. BẢNG ĐÁNH GIÁ VỀ MỐI QUAN HỆ HỢP TÁC:
   1. Cách đánh giá thành viên:

* Cách thức tính phần trăm đóng góp của mỗi thành viên dựa vào công việc và mức độ làm việc của các thành viên tính theo tỷ trọng của bảng đánh giá chi tiết bên dưới.
  1. Bảng đánh giá chi tiết:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***STT*** | ***Tỷ trọng*** | ***Tiêu chí*** | ***Tuyệt vời*** | ***Tốt*** | ***Tạm được*** | ***Kém*** |
| 1 | 40% | Đóng góp về nội dung, chất lượng bài làm | Số lượng bài làm nhiều. Chất lượng bài làm tốt. Đánh giá bài làm của các bạn khác và góp ý để chỉnh sửa. Sẵn sàng hướng dẫn lại cho bạn/sửa lại/làm lại nếu thấy chưa đúng | Số lượng bài làm nhiều. Chất lượng bài làm tốt. Đánh giá bài làm của các bạn khác và góp ý chỉnh sửa | Có chịu khó làm bài.  Chất lượng bài làm không tốt, nhưng biết học hỏi, lắng nghe góp ý của các bạn và chỉnh sửa lại | Không làm bài, hoặc làm ít và chất lượng bài làm kém, sai nhiều nhưng không chỉnh sửa/làm lại theo góp ý của các bạn |
| 2 | 10% | Tính thần trách nhiệm | Hoàn thành tốt công việc được phân công trong thời gian đề ra, giúp đỡ các thành viên khác | Hoàn thành tốt công việc được phân công trong thời gian đề ra | Hoàn thành công việc | Không hoàn thành công việc |
| 3 | 15% | Giải quyết vấn đề (vấn đề có thể là khó khăn trong hoạt động nhóm, khó khăn khi làm bài tập)  Tư duy sáng tạo | Tham gia nhiệt tình đóng góp ý kiến trong các buổi họp nhóm  Luôn đưa ra được cách thức/ ý tưởng mới để giải quyết vấn đề  Giải được các bài tập khó | Đóng góp ý kiến tương đối nhiều | Thỉnh thoảng đóng góp | Không đóng góp ý kiến nào hết |
| 4 | 10% | Tinh thần hợp tác | Dựa trên sự tôn trọng và mục đích cuối cùng là đạt được kết quả tốt trong cộng việc và lợi ích của nhóm | Đạt kết quả tót trong công việc | Đạt được kết quả tạm được | Không đạt kết quả |
| 5 | 10% | Quản lý xung đột | Các bạn hợp tác với nhau tìm ra giải pháp tốt nhất cho cả đôi bên, chú trọng sự đồng thuận. | Cùng nhau tìm ra những giải pháp trung hòa để đôi bên cùng có một phần lợi ích | Quan tâm đến giữ các mối quan hệ chứ không cần quan tâm đến kết quả | Né tránh sự va chạm, sợ đối đầu, không quan tâm đến nhu cầu của các bên |
| 6 | 5% | Họp nhóm | 2 lần / tuần | 1 lần/ tuần | Nảy sinh vấn đề mới họp | Không hợp |
| 7 | 10% | Tìm kiếm thông tin | Tìm kiếm đầy đủ, phong phú các thông tin mà nhóm cần, tổng hợp một cách hiệu quả | Tìm kiếm đầy đủ, phong phú các thông tin | Tìm kiếm thông tin để đối phó | Không tìm báo cáo sơ sài |

# BẢNG PHÂN CÔNG CÔNG VIỆC

|  |  |  |
| --- | --- | --- |
| STT | THÀNH VIÊN | CÔNG VIỆC |
| 1 | Nguyễn Trần Quốc Toàn | * Hướng dẫn thiết kế cấu trúc dữ liệu * Kiểm tra và chạy thử chương trình * Xây dựng các cải tiến Bulldozer * Xây dựng cơ chế bắn * Xây dựng cơ chế né đạn * Quay video thực hiện chương trình |
| 2 | Nguyễn Phạm Long Duy | * Xây dựng chế độ một người chơi cơ bản * Xây dựng môi trường để kiểm tra độ chính xác của code * Thiết kế chướng ngại vật * Hỗ trợ chuẩn hoá code vào môi trường * Kiểm tra và chạy thử chương trình |
| 3 | Võ Thành Cường | * Xây dựng giải thuật A\* cho xe tăng * Kiểm tra và chạy thử chương trình * Viết báo cáo * Thiết kế màn chơi |
| 4 | Phạm Ngọc Phúc Thuần | * Hỗ trợ xây dựng môi trường ứng dụng * Hỗ trợ xây dựng giải thuật A\* cho xe tăng * Xây dựng giao diện tương tác với người chơi * Kiểm tra và chạy thử chương trình * Viết báo cáo |

# BẢNG ĐÁNH GIÁ NỘI BỘ NHÓM

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *STT* | *THÀNH VIÊN* | *CÔNG VIỆC ĐƯỢC PHÂN CÔNG* | *MỨC ĐỘ HOÀN THIỆN CÔNG VIỆC* | *ĐÁNH GIÁ* |
| 1 | Nguyễn Trần Quốc Toàn | Hướng dẫn thiết kế cấu trúc dữ liệu | Hướng dẫn chi tiết về xây dựng cấu trúc dữ liệu | 100% |
| Kiểm tra và sửa lỗi chương trình | Phát hiện lỗi và sửa chữa kịp thời |
| Xây dựng các cải tiến Bulldozer và Hunter | Thuật toán tìm đường đi ngắn nhất từ pháo đài |
| Xây dựng cơ chế bắn | Thêm khả năng tiết kiệm đạn |
| Xây dựng cơ chế né đạn | Hoàn thành |
| Thiết kế màn chơi | Thiết kế màn chơi đơn giản và thể hiện đủ tính năng của xe tăng |
| 2 | Nguyễn Phạm Long Duy | Xây dựng chế độ một người chơi cơ bản | Bổ sung thêm cách chơi bằng tay chơi. | 90% |
| Xây dựng môi trường | Xây dựng Node và các cơ chế cơ bản của xe tăng. |
| Thiết kế chướng ngại vật | Hai chướng ngại vật: tường gạch và tường thép |
| Hỗ trợ chuẩn hoá code vào môi trường | Sửa code để phù hợp với môi trường |
| Chạy thử chương trình | Tăng độ mượt của game |
| Quay video thực hiện chương trình | Quay video hiện thực game |
| 3 | Võ Thành Cường | Xây dựng giải thuật A\* cho xe tăng | Xây dựng giải thuật A\* cho 3 xe tăng | 95% |
| Kiểm tra và chạy thử chương trình | Test game, tìm và sửa lỗi trong quá trình chạy |
| Viết báo cáo | Hỗ trợ viết báo cáo: vẽ sơ đồ |
| 4 | Phạm Ngọc Phúc Thuần | Hỗ trợ xây dựng môi trường ứng dụng | Xây dựng Node, kiểm tra Node. | 95% |
| Hỗ trợ xây dựng giải thuật A\* cho xe tăng | Thiết kế sơ đồ lớp các xe tăng |
| Xây dựng giao diện tương tác với người chơi | Thiết kế các scene main menu, help, about |
| Kiểm tra và chạy thử chương trình | Kiểm tra các scene và chương trình trước khi thành phẩm |
| Viết báo cáo | Hoàn thành báo cáo |

# CHƯƠNG 1: GIỚI THIỆU ĐỒ ÁN

1. LỊCH SỬ TRÒ CHƠI:

*Battle City* (*Bắn xe tăng*) là một trò chơi điện tử bắn súng đa hướng được Namco giới thiệu vào ngày 9 tháng 9 năm 1985. Sự xuất hiện của *Battle City* được xem là một bước cải tiến lớn so với phiên bản tiền nhiệm – *Tank Battalion,* đồng thời, góp phần quan trọng cho sự thành công của phiên bản tiếp theo – *Tank Force.*

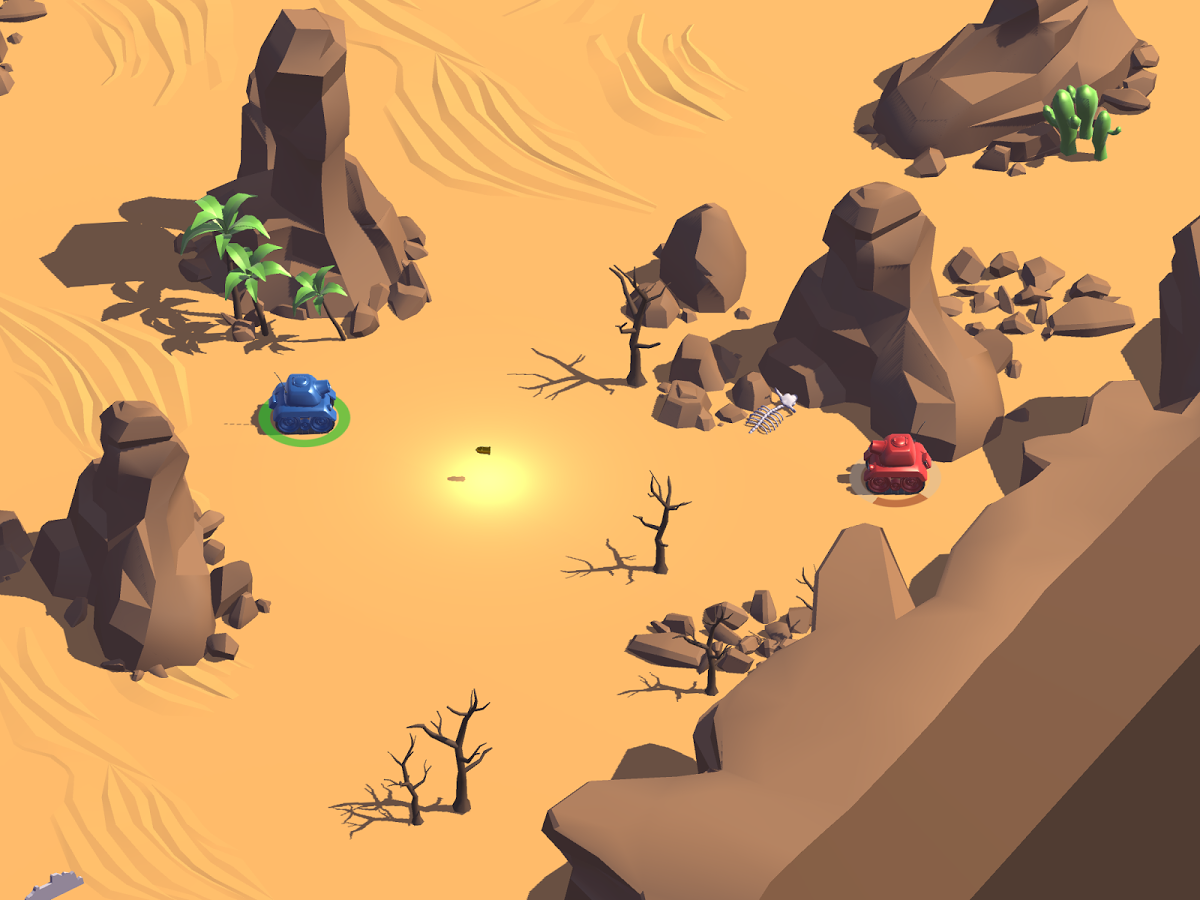


*Hình 1: Bộ sản phẩm trò chơi Battle City cho máy điện tử 4 nút*

**

*Hình 2: Máy điện tử 4 nút Famicom*

Từ nền tảng trò chơi trên máy điện tử 4 nút, Battle City đã được phát triển thêm trên nền tảng *arcade game* (dùng máy chơi game dùng xu). Năm 1991, Nova đã đưa trò chơi vào *Game Boy*, một hệ thống giả lập chơi game cầm tay 8-bit được phát triển và chế tạo bởi Nitendo. Hiện tại, *Battle City* đã được phát triển trên các nền tảng mới với nhiều sự thay đổi về lối chơi và cải tiến về đồ hoạ, điển hình là TANKS! được xây dựng bằng phần mềm Unity.



*Hình 3: Một cảnh từ trò chơi TANKS!*

1. MÔ TẢ TRÒ CHƠI:
   1. Cốt truyện:

Bạn là thành viên cuối cùng của lực lượng chỉ huy xe tăng tinh nhuệ. Bạn không có sự lựa chọn nào khác ngoài bảo vệ pháo đài và chống lại toàn bộ quân địch. Hãy lập ra chiến lược để tiêu diệt được quân địch nhanh nhất có thể. Đừng quên né đạn của quân địch và bảo vệ pháo đài nhé!

* 1. Mô tả sơ lược:
* Bạn xuất hiện bên trái pháo đài ở phần dưới của khung hình. Các xe tăng của quân địch sẽ xuất hiện từ một trong ba vị trí ở phần trên cùng của khung hình.
* Ở mỗi màn của trò chơi, bạn phải loại bỏ tổng cộng 20 xe tăng để đến được màn tiếp theo của trò chơi.
* Bạn có thể bắn đạn theo bốn hướng. Tiêu diệt xe tăng địch chỉ cần một lần bắn. Tuy nhiên, quân địch có một loại xe tăng yêu cầu bạn cần đến bốn lần bắn để tiêu diệt hoàn toàn.
* Nếu bạn bị trúng đạn của quân địch, bạn sẽ mất một mạng, Nếu pháo đài của bạn bị bắn, trò chơi sẽ kết thúc ngay lập tức, đồng nghĩa với việc bạn đã thua.
* Bạn có thể bắn đạn để phá tường. Một bức tường tiêu chuẩn cần 4 lần bắn để phá huỷ.
* Xe tăng địch nhấp nháy màu đỏ sẽ cung cấp *sức mạnh (power-up)* bất cứ khi nào bị bắn trúng. Những *sức mạnh* sẽ xuất hiện ngẫu nhiên ở mọi nơi trong khung hình.
  1. Số lượng màn chơi cơ bản: 50 màn chơi.
  2. Mô tả các thành phần trò chơi cơ bản trên nền tảng Famicom:
     1. *Các nút điều khiển:*
* File:Nintendo-Dpad-Neutral.png : Dùng các phím D-pad để điều khiển xe tăng theo bốn hướng (trái, phải, trên dưới). Xe tăng không di chuyển theo đường chéo.
* A button hoặc B button : Dùng một trong hai phím để bắn đạn theo hướng nòng súng của xe tăng.
* Start button : Bắt đầu hoặc tạm dừng trò chơi.
* Select button : Không có chức năng.
  + 1. *Xe tăng:*
  + Người chơi:

|  |  |  |
| --- | --- | --- |
| Battle City Tank Player1.png | Người chơi thứ nhất điều khiển xe tăng màu vàng và người chơi thứ hai điều khiển xe tăng màu xanh lá. Cả hai xe tăng đều có nòng súng riêng. Chỉ được phép bắn lượt đạn mới khi lượt đạn cũ đã hết. Nếu một người chơi bắn người chơi kia, người chơi kia sẽ tạm thời đừng di chuyển, mặc dù có thể né hoặc bị tiêu diệt. | Battle City Tank Player2.png |

* + Quân địch: có 4 loại xe tăng với các thông số khác nhau.
    1. *Sức mạnh (Power-up):*

Mỗi sức mạnh sẽ xuất hiện tại một trong 16 vị trí ngẫu nhiên trên màn hình bất kì khi nào một trong ba xe tăng nhấp nháy bị bắn. Sức mạnh này sẽ biến mất khi một xe tăng nhấp nháy khác xuất hiện. Chạm vào các sức mạnh để thu nhập chúng (chúng sẽ biến mất) và người chơi có thể nhận được 500 điệm khi sức mạnh được sử dụng.

|  |  |  |
| --- | --- | --- |
| *Sức mạnh* | *Tên gọi* | *Công dụng* |
|  | Lưu đạn  (Grenade) | * Tiêu diệt toàn bộ xe tăng của quân địch đang xuất hiện trên khung hình. * Không áp dụng cho các xe tăng đã bị tiêu diệt |
|  | Mũ sắt (Helmet) | * Tạo một lá chắn tạm thời xung quang xe tăng của người chơi để ngăn chặn đạn của quân địch. |
|  | Xẻng (Shovel) | * Biến các tường gạch xung quanh pháo đài thành tường thép, mang lại khả năng bất tử tạm thời cho các bức tường và ngăn chặn quân địch phá huỷ pháo đài. * Xây dựng lại những bức tường chưa bị phá huỷ hoàn toàn. |
|  | Ngôi sao (Star) | * Tăng sức tấn công của người chơi lên một cấp (có 4 cấp độ: mặc định, cấp độ 2, cấp độ 3, cấp độ 4). * Khi người chơi chết (mất một mạng), các thông số này sẽ trở về mặc định. * Ngôi sao thứ nhất (cấp độ 2): tốc đạn bắn ngang với tốc độ đạn bắn của Power Tanks. * Ngôi sao thứ hai (cấp độ 3): người chơi có thể bắn 2 viên đạn trong một khung hình. * Ngôi sao thứ ba (cấp độ 4): đạn bắn có thể phá huỷ được tường thép (yêu cầu 2 lần bắn) và gây thiệt hại gấp đôi đối với tường gạch! |
|  | Xe tăng (Tank) | * Người chơi được thêm một mạng * Cách duy nhất để có thêm một mạng là phải đạt được 20,000 điểm |
|  | Thời gian (Timer) | * Sức mạnh này làm đóng băng tạm thời thời gian và ngăn mọi sự di chuyển của xe tăng địch. * *Gợi ý:* Hãy sử dụng khả năng này để tiếp cận xe tăng địch và tiêu diệt chúng. |

* + 1. *Môi trường:*

Mỗi màn chơi sẽ có nhiều loại chướng ngại vật khác nhau nằm rải rác xung quanh. Chìa khoá để sống sót và chiến thắng là hãy sử dụng các đặc tính của chúng để tạo lợi thế cho bản thân.

|  |  |  |
| --- | --- | --- |
| *Môi trường* | *Loại môi trường* | *Đặc tính* |
|  | Tường gạch | * Xe tăng và đạn không thể vượt qua tường gạch. * Tường gạch có thể bị phá uỷ bởi 4 lần bắn. |
|  | Tường thép | * Xe tăng và đạn HOÀN TOÀN không thể vượt qua tường thép. |
|  | Rừng | * Xe tăng và đạn có thể di chuyển dưới tán cây. * *Cẩn thận:* quân địch di chuyển dưới tán cây sẽ rất khó phát hiện. Đặc biệt, đạn bay dưới tán cây sẽ càng khó phát hiện hơn! |
|  | Sông | * Xe tăng không thể vượt qua sông. * Nhưng đạn có thể bay qua sông. |
|  | Băng | * Xe tăng có thể di chuyển trên băng. Xe tăng sẽ phải trượt lên một chút để dừng lại hoàn toàn. * *Cẩn thận:* đạn có thể bay trên băng. |

# CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

1. NGÔN NGỮ LẬP TRÌNH C#:

* C# là một ngôn ngữ lập tình đơn giản, được phát triển bỏi đội ngũ kỹ sư của Micrrosoft vào năm 2000, trong đó người dẫn đầu là Anders Hejlsberg và Scott Wiltamunth.
* C# là ngôn ngữ lập trình hiện đại, hướng đối tượng và nó được xây dựng trên nền tảng của hai ngôn ngữ mạnh nhất là C++ và Java. Do đó, C# được miêu tả là ngôn ngữ có sự cân bằng giữa C++, Visual Basic, Delphi và Java.
* C# được thiết kế cho Common Language Infrastructure (CLI), mà gồm Executable Code và Runtime Environment, cho chép người dùng sử dụng các ngôn ngữ high-level đa dạng trên các nền tảng và cấu trúc máy tính khác nhau.
* C# với sự hỗ trợ mạnh mẽ của .NET Framework giúp cho việc tạo một ứng dụng Windows Forms hay WPF (Windows Prensentation Foundation),… trở nên rất dễ dàng.

1. LÝ THUYẾT ÁP DỤNG:

Bài toán trong trò chơi Bắn xe tăng (Battle City) được xây dựng và giải quyết trên cơ sở lý thuyết thuật giải A\*.

* 1. Định nghĩa thuật giải A\*:

A\* là giải thuật tìm kiếm trong đồ thị, tìm đường đi từ một từ một đỉnh hiện tại đến đỉnh đích có sử dụng hàm để ước lượng khoảng cách hay còn gọi là *hàm Heuristic.*

*Heuristic là gì?* Heuristic là phương pháp giải quyết vấn đề dựa trên phỏng đoán, ước chừng, kinh nghiệm, trực giác để tìm ra giải pháp gần như là tốt nhất, nhanh chóng, dễ dàng.

*Hàm Heuristic là gì?* Hàm Hueristic là hàm ứng với mỗi trạng thái hay mỗi sự lựa chọn một giá trị ý nghĩa đối với vấn đề dựa vào giá trị hàm này ta lựa chọn hành động.

Từ trạng thái hiện tại A\* xây dựng tất cả các đường đi có thể đi dùng hàm ước lược khoảng cách (hàm Heuristic) để đánh giá đường đi tốt nhất có thể đi. Theo mỗi dạng bài khác nhau mà hàm Heuristic sẽ được đánh giá khác nhau. A\* luôn tìm được đường đi ngắn nhất nếu tồn tại đường đi như thế. Trong Battle City, nhóm sẽ xây dựng các hàm Heuristic cho xe tăng của quân địch đã được liệt kê ở chương 1, phần 2.

A\* lưu giữ một tập các đường đi qua đồ thị, từ đỉnh bắt đầu đến đỉnh kết thúc. Tập các đỉnh có thể đi tiếp được lưu trong tập Open. Thứ tự ưu tiên cho một đường đi đươc quyết định bởi hàm Heuristic được đánh giá bằng công thức:

|  |  |  |
| --- | --- | --- |
|  | *f(x) = g(x) + h(x)* |  |

Trong đó:

* *g(x)* là chi phí của đường đi từ đỉnh xuất phát cho đến thời đỉnh hiện tại;
* *h(x)* là hàm ước lượng chi phí từ đỉnh hiện tại đến đỉnh đích;
* *f(x)* là độ tốt, dùng để so sánh và chọn đường đi, *f(x)* có giá trị càng thấp thì độ ưu tiên càng cao.
  1. Mô tả thuật giải A\*:
* Tạo tập Open để chứa các đỉnh lân cận nhưng chưa xét đến.
* Tạo tập Close để chứa các đỉnh đã xét qua.
* cost(p, q): khoảng cách giữa p, q.
* g(p): khoảng cách từ đỉnh bắt đầu đến đỉnh hiện tại.
* h(p): giá trị được ước lượng từ đỉnh hiện tại đến đỉnh kết thúc.
* f(p) = g(p) + h(p)
* Bước 1:
  + - Open: = {s} // s là đỉnh bắt đầu
    - Close: = { }
* Bước 2: while (Open != { })
  + - Chọn p thuộc tập Open có f(p) nhỏ nhất.
    - Nếu p là đỉnh kết thúc thì thoát và thông báo kết quả.
    - Chuyển p qua tập Close và mở các q sau p.
      * Nếu q đã có trong Open:
        + Nếu g(q) > g(p) + cost(p, q):

g(q) = g(p) + cost(p, q)

f(q) = g(q) + h(q)

prev(q) = p // prev = đỉnh cha // đỉnh cha của q là p

* + - * Nếu q chưa có trong Open và q chưa có trong Close:
        + g(q) = g(p) + cost(p, q)
        + f(q) = g(q) + h(q)
        + prev(q) = p
        + Thêm q vào Open
      * Nếu q có trong Close:
        + Nếu g(q) > g(p) + cost(p, q)

Bỏ q khỏi Close

g(q) = g(p) + cost(p, q)

f(q) = g(q) + h(q)

prev(q) = p

Thêm q vào Open

Cập nhật các đỉnh chịu ảnh hướng từ sự thay đổi của q

* Bước 3: Không tìm được.
  1. Bài toán mô phỏng tìm đường đi ngắn nhất A\* ĐƠN GIẢN:

h = 6

|  |
| --- |
| 8  3  2  3  8  11  7  4  7  10  5  h = 1  h = 5  h = 7  h = 3  h = 4  h = 3 |

h(A) = 6; h(B) = 5; h(C) = 4; h(D) = 3; h(E) = 3; h(F) =7; h(G) = 1.

Tìm đường đi ngắn nhất từ đỉnh A đến đỉnh F.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Bước* | *p* | *Các đỉnh nối với p* | *Open* | *Close* |
| 0 |  |  | A6 |  |
| 1 | A | B | B5 | A |
| 2 | B | C | C4 | A, B |
| 3 | C | D | D3 | A, B, C |
| 4 | D | B, E, G | E3, G1 | A, B, C, D |
| 5 | E | F | G1, F7 | A, B, C, D, E |
| 6 | G | A, B, F | F7 | A, B, C, D, E, G |
| 7 | F  (dừng) | E |  |  |

Cây tìm kiếm với đồ thị trên:

|  |
| --- |
| h(F) = 7; g(F) = 29  h(E) = 3; g(E) = 26;  f(E) = 26 + 3 = 29  h(G) = 1; g(G) = 31;  f(G) = 31 + 3 = 34  h(D) = 3; g(D) = 20;  f(D) = 20 + 3 = 23  h(C) = 4; g(C) = 13;  f(A) = 13 + 4 = 17  h(B) = 5; g(B) = 8;  f(B) = 8 + 5 = 13  h(A) = 6; g(A) = 0;  f(A) = 0 + 6 =6 |

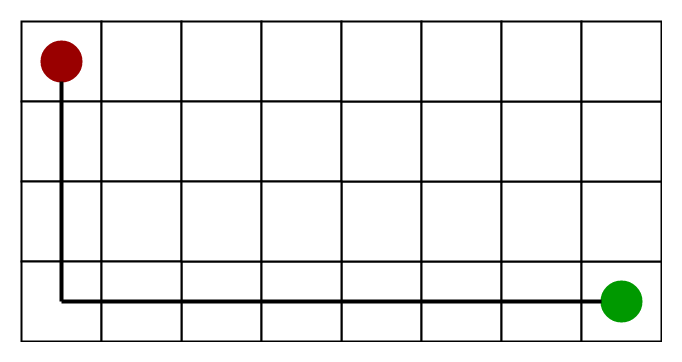
* 1. Các cách tính giá trị Heuristic:

Các bài toán trong thực tế không hề biết trước giá trị heuristic (*h(x)*).

Do đó, việc xây dựng các hàm Heuristic để tính giá trị của *h(x)* là vô cùng quan trọng. Sau đây là ba phương pháp tính *h(x)* thường được sử dụng:

* + 1. *Đường Manhattan:*

Sử dụng đường Manhattan khi chỉ được phép di chuyển theo bốn hướng (trái phải, trên, dưới). Ví dụ như hình bên dưới, đường đi ngắn nhất từ chấm đỏ đến chấm xanh lá theo áp dụng đường Manhattan.



*Hình 4: Đường đi theo phương pháp đường Manhattan*

Công thức tính *h(x)*:

|  |
| --- |
| h = abs(current\_cell.x – goal.x) + abs(current\_cell.y – goal.y) |

* + 1. *Đường Diagonal:*

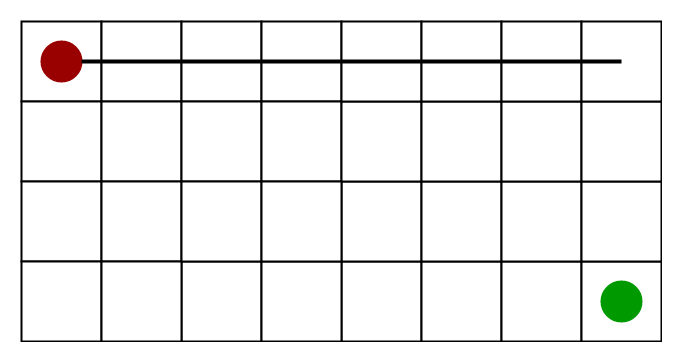
|  |  |  |
| --- | --- | --- |
| X | X | X |
| X | *Điểm hiện tại* | X |
| X | X | X |

Sử dụng đường Diagonal khi được phép di chuyển theo tám hướng (các ô được đánh dấu X) xung quanh điểm hiện tại.

Công thức tính *h(x)*:

|  |
| --- |
| h = max {abs(current\_cell.x – goal.x), abs(current\_cell.y – goal.y)} |

Đường đi ngắn nhất từ chấm đỏ đến chấm xanh lá theo áp dụng đường Diagonal:



*Hình 5: Đường đi theo phương pháp đường Diagonal*

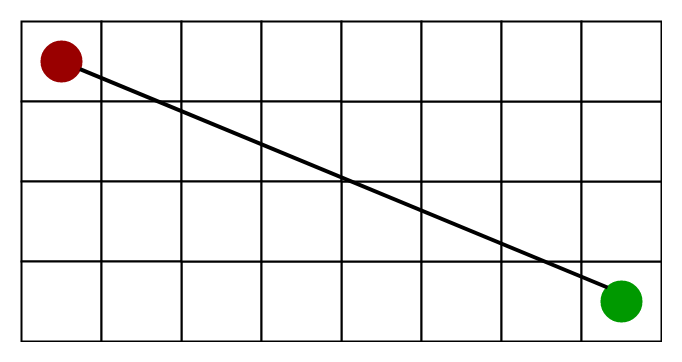
* + 1. *Đường Euclidean:*

Sử dụng đường Euclidean khi được phép di chuyển theo TẤT CẢ các hướng. Tức là đường đi không còn giới hạn là đường thẳng mà có thể là đường chéo!

Công thức tính *h(x)*:

|  |
| --- |
| h = sqrt((current\_cell.x – goal.x)\*2 + (current\_cell.y – goal.y)\*2) |

Đường đi ngắn nhất từ chấm đỏ đến chấm xanh lá theo áp dụng đường Euclidean:



*Hình 6: Đường đi theo phương pháp đường Euclidean*

*\* Chú giải các phép toán:*

* abs(x): trị tuyệt đối của x.
* sqrt(x): căn bậc hai của x.
* max(x, y): giá trị lớn nhất giữa x và y.
  1. Nhận xét về thuật giải A\*:
     1. *Ưu điểm:*
* Linh động, tổng quát.
* Nhanh chóng tìm được lời giải.
  + 1. *Hạn chế:*
* Tốn khá nhiều bộ nhớ để lưu lại những trạng thái (đỉnh) đã đi qua.

# CHƯƠNG 3: PHÂN TÍCH VÀ THIẾT KẾ

PHÂN TÍCH BÀI TOÁN:

* 1. Đặt vấn đề:

Trò chơi *Battle City* (Bắn xe tăng) là một trò chơi điện tử kinh điển của thế giới. Do đó, việc thiết kế và cải tiến trò chơi hoàn chỉnh yêu cầu nhiều thời gian xây dựng và khả năng xử lý bug (lỗi) trên công cụ hỗ trợ xây dựng Unity.

* 1. Các vấn đề cần giải quyết:
* Xây dựng môi trường trò chơi hoàn chỉnh.
* Xây dựng thuật giải A\* cho từng loại xe tăng.
* Cải tiến cơ chế bắn và cơ chế né đạn của xe tăng.

1. THIẾT KẾ CẤU TRÚC DỮ LIỆU VÀ TRẠNG THÁI:
   1. Không gian trạng thái của bài toán: Vị trí của các xe tăng trên bản đồ.
   2. Sơ đồ cấu trúc chương trình:

|  |
| --- |
| *Eagle*  *steel*  *Game over*  *Màn chơi*  *Stage 1*  *Scene*  *Bulldozer*  *Seeker*  *Player*  player  *HunterA*  *About*  *Main menu*  *Giao diện*  *BrickWall*  *Node*  *Bản đồ*  *Help*  *Tank*  *Battle City* |

* 1. Xử lý tính toán:
     1. *Bắt đầu chương trình:*

Khi người chơi chọn *Play* ở Main menuthì chương trình bắt đầu chuyển (loadLevel) sang màn chơi đầu tiên (stage 1).

* + 1. *Các xử lý khi chương trình chạy:*
* Khởi tạo bản đồ: dựa vào bản đồ, các node sẽ được khởi tạo giá trị.
* Pháo đài: cập nhật vị trí node ở pháo đài và máu cho pháo đài.
* Xe tăng: các xe tăng (người chơi và quân địch) sẽ xuất hiện ở các vị trí cố định trên bản đồ, đồng thời cập nhật vị trí node tại các vị trí của xe tăng xuất hiện.
* Xe tăng tìm đường đi và tiêu diệt mục tiêu theo giải thuật A\* (được trình bày ở chương 3, mục 3.2)
  + 1. *Điều kiện dừng:*
* Điều kiện để kết thúc một màn chơi:
  + Người chơi bị tiêu diệt hoàn toàn.
  + Nhà chính bị tiêu diệt.
  + Tất cả xe tăng địch bị tiêu diệt.
  1. Thông tin cấu trúc dữ liệu:
     1. *Bản đồ:*
* Bản đồ sẽ được xây dựng dựa trên class Node:
  + Các thuộc tính:
    - Node TopNode: node phía trên
    - Node BottomNode: node phía dưới
    - Node LeftNode: node phía bên trái
    - Node RightNode: node phía bên phải
    - Vector3 position: vị trí của node hiện tại.
    - int Dist\_Base: khoảng cách của node đến node pháo đài.
    - int obstacle: trọng số của node
  + Các phương thức:
    - Start(): khởi tạo position (vị trí của node hiện tại).
    - getGameobj(): tìm GameObject tại đó.
    - getName(): lấy tên node và chuyển tên node sang dạng số nguyên.
    - OnTriggerStay2D(Collider2D other): gán giá trọ trọng số cho node.
* Pháo đài được xây dựng trong class Eagle:
  + Các thuộc tính:
    - int hp: máu của pháo đài.
    - Node CurrentNode: vị trí node của pháo đài.
  + Các phương thức:
    - CheckV(): hàm tính giá trị Dist\_Base của một node (được phân tích ở mục 3.4)
    - Update(): tạo một vòng lặp *while* để tính hết giá trị Dist\_Base của tất cả các node.
* Chướng ngại vật *tường gạch* trên bản đồ được xây dựng thành class BrickWall bao gồm các phương thức:
  + OnCollisionEnter2D: phá huỷ tường gạch.
  + OnTriggerExit2D: gán trọng số cho node của tường gạch.
* Chướng ngại vật *tường thép* trên bản đồ được xây dựng thành class steel bao gồm phương thức OnTriggerExit2D – gán trọng số cho node của tường thép.
  + 1. *Xe tăng:* được xây dựng là lớp cơ sở cho các xe tăng khác – class Tanks.
* Các thuộc tính:
  + Node startNode: vị trí node đầu tiên xe tăng xuất hiện
  + GameObject Base
  + GameObject player
  + Node currentNode
  + Node nextNode
  + Node[] pre
  + float[] g
  + float[] fbase
  + float[] fplayer
  + Node currentPoint
  + Transform firePosition
  + GameObject bullet
  + GameObject Explosion
  + AudioSource shootaudio
* Các phương thức:
* Move(): hàm để giúp xe tăng di chuyển.
* facing(): hàm xác định vị trí tiếp theo của xe tăng.
* turnDirection(): hàm giúp cho xe quay mặt.
* RayShoot(): xác định vật cản và bắn.
* Shoot(): hàm bắn đạn.
* OnCollisionEnter2D(Collision2D collision): Kết quả của màn chơi.
* virtual Node findMinNode(): tìm Node có trị số bé nhất trong mảng f.
* virtual void calF(): hàm tính f.
* virtual float calH(): hàm Heuristic.
* virtual void calG(): hàm tính g.
* virtual bool CheckName(): kiểm tra vị trí kết thúc.
* virtual void Init(): hàm khởi tạo giá trị bắt đầu.
* virtual Node TraceBack(): hàm truy xuất đường đi.
  + 1. *Các lớp dẫn xuất xe tăng:*
* *Player (Người chơi):* 
  + input(): các thao tác được người chơi nhập từ bàn phím hoặc tay cầm.
* *Bulldozer:* mục tiêu tiêu diệt là pháo đài nên gán Node target = base và thay đổi ở các phương thức sau:
* override void calF(): tính giá trị f khi mục tiêu là pháo đài.
* override void calG(): tính giá trị g khi mục tiêu là pháo đài.
* override bool CheckName(): điều kiện dừng lại khi chạm đến Node pháo đài.
* override Node TraceBack(): truy xuất đường đi của Bullozer.
* *Seeker:* mục tiêu tiêu diệt là người chơi nên gán Node target = player và thay đổi ở các phương thức sau:
  + override void calF(): tính giá trị f khi mục tiêu là player.
  + override void calG(): tính giá trị g khi mục tiêu là player
  + override bool CheckName(): điều kiện dừng lại khi chạm đến Node player
  + override Node TraceBack(): truy xuất đường đi của Seeker
* *HunterA*: mục tiêu tiêu diệt bao gồm cả pháo đài và người chơi nên gán Node target1 = base và Node target2 = player. Các phương thức thay đổi:
  + override void calF(): tính giá trị f tốt nhất trong cả mục tiêu là player và base.
  + override void calG(): tính giá trị g khi mục tiêu là player và base.
  + override bool CheckName(): điều kiện dừng lại khi chạm đến Node player hoặc đến Node base.
  + override Node TraceBack(): truy xuất đường đi của Hunter.

Ngoài ra, class HunterA còn gồm các phương thức khác:

* Threat\_Detection (): hàm phát hiện nguy hiểm (được phân tích ở mục 3.4).
* Evade (): hàm né đạn đặc biệt dùng cho hunter (được phân tích ở mục 3.4).

1. XÂY DỰNG BÀI TOÁN:
   1. Lớp Node:



*Hình 7: Các node đã được xây dựng trong bản đồ*

* Số lượng node trên bản đồ: 180 node.
* Xây dựng từng node, mỗi node bao gồm những thuộc tính sau:

|  |  |  |
| --- | --- | --- |
|  | TopNode |  |
| LeftNode | CurrentNode | RightNode |
|  | BottomNode |  |

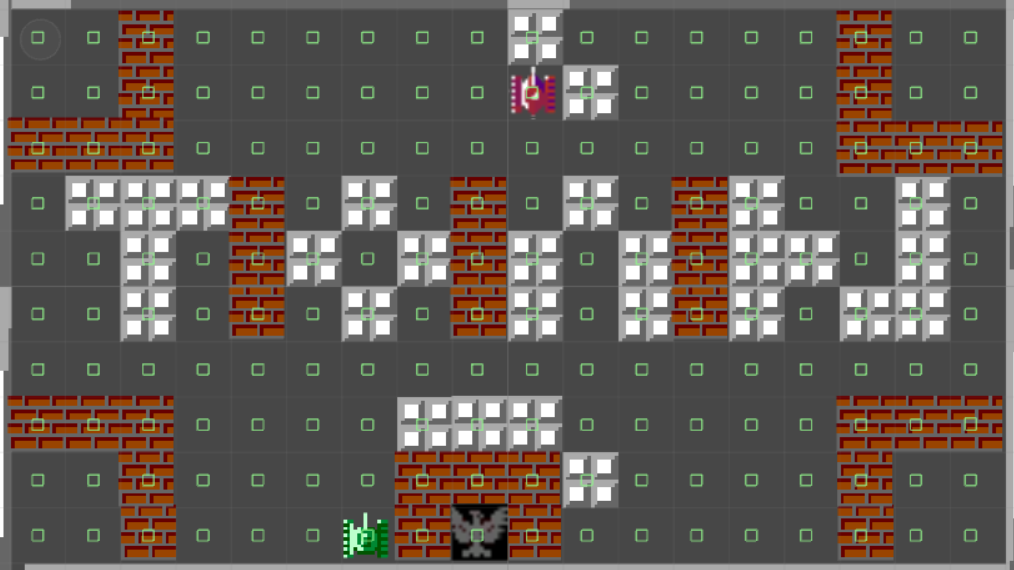
* + Node TopNode: node phía trên
  + Node BottomNode: node phía dưới
  + Node LeftNode: node phía bên trái
  + Node RightNode: node phía bên phải
  + Vector3 position: vị trí của node hiện tại (x, y, z); z = 0 do chương trình được xây dựng theo đồ hoạ 2D.
  + int Dist\_Base: dùng để cải tiến thuật giải tìm đường đi ngắn nhất đến pháo đài (xem chi tiết ở mục 3.3).
  + int obstacle: trọng số tại các node được tính ở hàm OnTriggerStay2D:
    - Trọng số mặc định là 0
    - Trọng số của tường gạch là 1
    - Trọng số của tường thép hoặc các xe tăng của quân địch là 9999

|  |
| --- |
| void OnTriggerStay2D (Collider2D other)  {  // steel = tường thép  // bulldozer, seeker, hunter = xe tăng quân địch  if ((other.tag=="steel") || (other.tag=="bulldozer") || (other.tag=="seeker") || (other.tag=="hunter"))  obstacle = 9999;  // brick = tường gạch  if (other.tag == "brick")  obstacle = 1;  } |

* Ưu điểm của việc xây dựng Node:
  + Bản đồ trực quan hơn.
  + Đơn giản hoá trong việc xây dựng các hàm xử lý đối với xe tăng.
* Hạn chế: chỉ áp dụng được cho bản đồ nhỏ.
  1. Xây dựng thuật giải A\* để tìm đường đi của các xe tăng:

*Bước 1:* Khởi tạo các biến cơ bản từ lớp tank: tạo các biến Node startNode, currentNode, nextNode, currentPoint; các mảng float g[], f\_base[], f\_player, pre[]. Khai báo các đối tượng để triển khai thuật giải A sao GameObject Base, target.

|  |
| --- |
| public Node startNode; //lưu trạng thái bắt đầu của xe tăng  public Node currentNode, nextNode; //lưu vị trí hiện tại và vị trí tiếp theo của xe tăng  protected Node[] pre; // mảng lưu đường tốt nhất đến mục tiêu  protected float[] g, fbase, fplayer; //khai báo mảng g và f đến 2 mục tiêu  protected Node currentPoint; // lưu vị trí hiện tại của xe tăng sau khi thực hiện thuật giải A sao. |



Ví dụ hình trên, ta sẽ có các biến khởi tạo cơ bản của xe Seeker như sau:

* Vị trí bắt đầu của xe tăng: startNode = 92;
* Mục tiêu người chơi đang ở node 69: target = 69;

*Bước 2:* Tìm đỉnh tốt nhất tiếp theo dựa trên nền tảng thuật giải A sao: Gán các giá trị cần thiết trong hàm Init() và khởi tạo hai tập danh sách Open và Close để lưu các giá trị f của từng Node.

|  |
| --- |
| Init(); // Gán các biến cần thiết trong thuật giải A sao  List<Node> open = new List<Node>(); //tạo list Open lưu các Node  List<Node> close = new List<Node>(); //tạo list Close. |

Theo ví dụ trên:

* Gán giá trị currentNode = startNode.
* Node 93, 82 được thêm vào tập Open.

*Bước 3:* Thêm vào tập Open vị trị hiện tại của xe tăng và kiểm tra tập Open có phải là danh sách rỗng hay không.

|  |
| --- |
| open.Add(currentNode); //thêm vào Open vị trí currentNode  while (open.Count != 0) //kiếm tra số phần tử trong tập Open |

*Bước 3.1:* Nếu Open là tập rỗng: Không tìm được đường đi đến mục tiêu và kết thúc chương trình. Trong trạng thái này vì sẽ không tồn tại người chơi hay pháo đài nên trò chơi kết thúc.

*Bước 3.2:* Nếu Open không phải là tập rỗng: Gán giá trị currentNode bằng giá trị nhỏ nhất trong tập open và kiểm tra tiếp điều kiện nếu đỉnh hiện hành trùng với mục tiêu hay không.

*Bước 3.2.1:* Nếu đỉnh hiện hành trùng với mục tiêu thì truy xuất đường đi từ trong danh sách list, xóa toàn bộ phần từ trong tập open và close và kết thúc thuật giải.

|  |
| --- |
| if (CheckName()) //hàm kiểm tra vị trí của xe tăng so với mục tiêu  {  break;  }  open.Clear();  close.Clear();  return TraceBack(currentPoint); |

*Bước 3.2.2:* Nếu không trùng với mục tiêu thì ta tiếp tục loại bỏ đỉnh hiện hành ra khỏi tập open, thêm nó vào tập close và duyệt tất cả đỉnh phù hợp so với đỉnh hiện hành vào danh sách list nextNode mới.

|  |
| --- |
| open.Remove(currentPoint); //xóa đỉnh hiện tại ra khỏi tập open  close.Add(currentPoint); // và thêm vào tập close |

* Duyệt tất cả đỉnh q kề với đỉnh p có trong danh sách nextNode:

|  |
| --- |
| foreach (Node q in nextPoint) |

* Nếu q đã có trong Open thì kiểm tra điều kiện nếu g(q) > g(p) + cost(p, q) như sau:

|  |
| --- |
| if (close.IndexOf(q) == -1 && open.IndexOf(q) == -1)  {  calG(ref g, q); //lời gọi hàm tính mảng g  calF(ref fbase, ref fplayer, q); //lời gọi hàm tính f pre[int.Parse(q.getGameobj().name)] = currentPoint;//tạo quan hệ cha con  open.Add(q); //Thêm đỉnh q vào tập open  } |

* Nếu q chưa có trong Open và q chưa có trong Close:

|  |
| --- |
| if (g[q.getName()] > g[currentPoint.getName()] + calH(currentPoint.getGameobj(), q.getGameobj()))  {  calG(ref g, q); //lời gọi hàm tính mảng g  calF(ref fbase, ref fplayer, q); //lời gọi hàm tính f  pre[q.getName()] = currentPoint; //tạo quan hệ cha con  } |

* Nếu q có trong Close: kiểm tra điều kiện nếu g(q) > g(p) + cost(p, q) như sau:

|  |
| --- |
| if (g[q.getName()] > g[currentPoint.getName()] + calH(currentPoint.getGameobj(), q.getGameobj()))  {  close.Remove(q); //loại đỉnh q ra khỏi tập close  open.Add(q); // và thêm vào tập open  calG(ref g, q); // lời gọi hàm tính g  calF(ref fbase, ref fplayer, q); // lời gọi hàm tính f  pre[q.getName()] = currentPoint; //tạo quan hệ cha con  } |

Sau khi kết thúc hàm findNextNode(), currentNode mới được trả về là Node 93. Như vậy ta được đường đi cho Seeker như sau:



Vì xe luôn di chuyển nên thuật giải A sao được cập nhật liên tục nên những trạng thái mới sẽ liên tục được sinh ra.



Node Target sẽ cập nhật một giá trị mới tương ứng vị trí hiện tại mới nhất của người chơi. Và trong trường hợp hình trên nó sẽ mang giá trị 49. Như vậy đường đi của Seeker sẽ luôn được cập nhập khi người chơi di chuyển sang một Node mới.



* 1. Một số hàm suy diễn phụ thuật giải A\*:
* *Hàm Init():* Sẽ tạo mảng g, f\_base, f\_player, pre với 180 phần tử tương ứng 180 Node có trong bản đồ. Khởi tạo phần tử g của Node hiện tại đầu tiên bằng 0 và f\_base sẽ bằng giá trị heuristic giữa currentNode đến pháo đài và f\_player sẽ bằng giá trị heuristic giữa currentNode đến người chơi.

|  |
| --- |
| g = new float[180];  fbase = new float[180];  fplayer = new float[180];  pre = new Node[180];  g[int.Parse(currentNode.getGameobj().name)] = 0;  fbase[int.Parse(currentNode.getGameobj().name)] = calH(currentNode.getGameobj(), Base);  fplayer[int.Parse(currentNode.getGameobj().name)] = calH(currentNode.getGameobj(), player); |

* *Hàm checkname():* sẽ kiểm tra vị trí hiện tại của xe tăng có trùng với mục tiêu của mình hay không. Vì mỗi xe có mỗi mục tiêu khác nhau nên phải override hàm Checkname trong lớp tank chính.

Xét ví dụ với Bullozer: xe Bullozer sẽ tìm đến nhà chính nên vị trí kết thúc sẽ là Base.name. Và tương tự cho các loại xe còn lại.

|  |
| --- |
| public override bool CheckName()  {  return ((Base.name == currentPoint.name));  } |

* *Hàm TraceBack():* sẽ truy xuất đường đi đến mục tiêu sao khi thực hiện xong quá trình tìm Node tiếp theo dựa vào tập pre.

Ví dụ với xe tăng Bullozer:

|  |
| --- |
| public override Node TraceBack (Node currentPoint)  {  try  {  Node nextMove = new Node();  int u = int.Parse(Base.name);  while (true)  {  nextMove = pre[u];//truy xuất đường đi đến mục tiêu  if (nextMove.getGameobj().name == currentNode.name)  {  break;//nếu tìm đến mục tiêu thì break  }  u = int.Parse(nextMove.getGameobj().name);  return GameObject.Find(u + "").GetComponent<Node>();  }  }  return currentNode;  //gán giá trị tiếp theo của Node làm giá trị hiện tại.  } |

* *Hàm CalG():* lời gọi hàm tính g sẽ thực hiện tính số bước đã di chuyển cộng thêm trong số của đỉnh tiếp theo (q.obstacle)

|  |
| --- |
| public override void calG(ref float[] g, Node q)  {  g[int.Parse(q.getGameobj().name)] = g[int.Parse(currentPoint.getGameobj().name)] + q.obstacle;  } |

* *Hàm CalF():* lời gọi hàm tính g sẽ thực hiện phép toán f = g + h

|  |
| --- |
| public override void calF(ref float[] fBase, ref float[] fPlayer, Node q)  {  fbase[int.Parse(q.getGameobj().name)] = g[int.Parse(q.getGameobj().name)] + calH(Base, q.getGameobj());  } |

* 1. Thuật toán tìm đường đi ngắn nhất đến pháo đài:

Pháo đài là mục tiêu của xe tăng Bulldozer và Hunter. Đặc biệt, pháo đài nằm cố định tại một vị trí trên bản đồ nên hàm calH có thể cải tiến lại như sau:

|  |
| --- |
| float calH(GameObject a, GameObject b)  {  return a.GetComponent<Node>().Dist\_Base;  } |

Theo như mô tả ở trên, Dist\_Base là khoảng cách của từng node được tính từ pháo đài. Giá trị của Dist\_Base được xác định từ hàm CheckV thuộc lớp Eagle:

|  |
| --- |
| // u: node hiện tại  // v: node lần lượt là các trái, phải, trên, dưới của node u  // Q: hàng đợi chứa các node  // Free: tập chứa các node đã tính  // hàm sẽ được cập nhật (trong hàm update) liên tục cho đến khi Q rỗng = tất cả các node đều được tính  void CheckV (ref Node u, ref Node v, ref Queue Q, ref HashSet<int> Free)  {  // kiểm tra node có tồn tại hoặc node đã được tính chưa  if ((v != null) && (!Free.Contains(v.getName())))  {  if (v.obstacle == 9999) // nếu node là chướng ngại vật (CNV)  v.Dist\_Base = -1;  else  {  Q.Enqueue(v); // thêm node đó vào hàng đợi  v.Dist\_Base = u.Dist\_Base + 1;  Free.Add(v.getName()); // node đã tính  }  }  } |

Mô phỏng bản đồ khi dùng Dist\_Base (hàm calH được cải tiến):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| h = 5 | h = 4 | h = 3 | h = 4 | *CNV*  h = -1 |
| h = 4 | *CNV*  h = -1 | h = 2 | h = 3 | h = 4 |
| h = 3 | h = 2 | h = 1 | *CNV*  h = -1 | h = 3 |
| h = 2 | h = 1 | *Pháo đài*  h = 0 | h = 1 | h = 2 |

Với cách cải tiến này, Dist\_Base chính là giá trị Heuristic của từng node trên bản đồ. Việc này giúp tiết kiệm thời gian tính toán và giảm độ phức tạp. Đồng thời, thuật toán đảm bảo xe tăng luôn tìm được đường đi tối ưu nhất đến pháo đài.

So sánh đường đi với hàm Heuristic được xây dựng ở lớp Tank (mục 3.2):

|  |  |
| --- | --- |
| *Dùng calH của lớp Tank* | *Dùng calH dùng Dist\_Base* |
|  |  |

* 1. Cơ chế bắn của các xe tăng quân địch:
* Đầu tiên, chỉnh hướng nòng súng (hướng bay của đạn) theo hướng đi (quay mặt) của xe tăng hiện tại.

|  |
| --- |
| float x = (firePosition.position.x – transform.position.x);  float y = (firePosition.position.x – transform.position.x);  if(!(x==0 && y==0))  {  dx = (Mathf.Abs(x) >= Mathf(y)) ? 1 : 0;  dy = 1 – (dx\*1);  if(x>0)  dx\*=1;  else dx\*=-1;  if(y>0)  dy\*=1;  else dy\*=-1;  } |

* Sử dụng tính năng của *Raycast* để phát hiện xe tăng của người chơi:
  + Chỉnh hướng quét theo hướng của nòng súng.
  + Nếu phát hiện có người chơi trên đường quét của tia (*ray*), xe tăng sẽ đưa ra quyết định có bắn hay không.
  + DX và DY là khoảng cách mà tia đã quét được, đây cũng là khoảng cách của đạn bắn ra để tiêu diệt người chơi.
  + Người chơi phải nằm trong tầm bắn R (tầm nhất định sao cho người chơi có ít thời gian né đạn nhất) thì xe tăng mới bắn đạn.

|  |
| --- |
| RaycastHit2D hit = Physics2D.Raycast (firePosition.position, new Vector2(dx, dy)); // chỉnh hướng quét của tia (ray)  float R = 10; // tầm bắn cho phép của xe tăng  if (hit.collider != null) //phát hiện chướng ngại vật trên đường quét  {  // kiểm tra chắc chắn rằng có chướng ngại vật  // kiểm tra chướng ngại vật có phải người chơi hay không  if ((hit.rigidbody != null)&&(hit.rigidbody.tag == “Player”))  {  // tính khoảng cách từ xe tăng đến người chơi hiện tại  float DX = Mathf.Abs(hit.rigidbody.position.x – transform.position.x)  float DY = Mathf.Abs(hit.rigidbody.position.y – transform.position.y)  // Kiểm tra người chơi có nằm trong tầm bắn R  if (((DX<=R) && (dx!=0)) || ((DY<=R) && (dy!=0)))  {  Shoot(); //bắn đạn  }  }  } |

* Minh hoạ cho cơ chế bắn:
  + Phát hiện người chơi và bắn (xe tăng người chơi nằm trong tầm bắn R):



* + Phát hiện người chơi nhưng không bắn (xe tăng người chơi không nằm trong tầm bắn R):

Ảnh có chứa ảnh chụp màn hình, màn hình, thiết bị điện tử, máy tính

Mô tả được tạo với mức tin cậy rất cao

* Nhờ vào cơ chế này, xe tăng có thể tiết kiệm số lượng đạn bắn ra.
  1. Cơ chế né đạn của xe tăng Hunter:
* Tương tự như cơ chế bắn, cơ chế né đạn sử dụng tính năng của *RayCast* để phát hiện đạn đang hướng tới xe tăng.
* Xây dựng hàm phát hiện nguy hiểm (đạn) tại hướng *hit* đang xét:
  + Impact\_Time: thời gian sẽ xảy ra va chạm.

Impact\_Time = khoảng cách từ vị trí đạn đến xe tăng / vận tốc của viên đạn

* + Evade\_Time: thời gian xe tăng né đạn.

Evade\_Time = (1 + 0.2) / tốc độ di chuyển của xe tăng

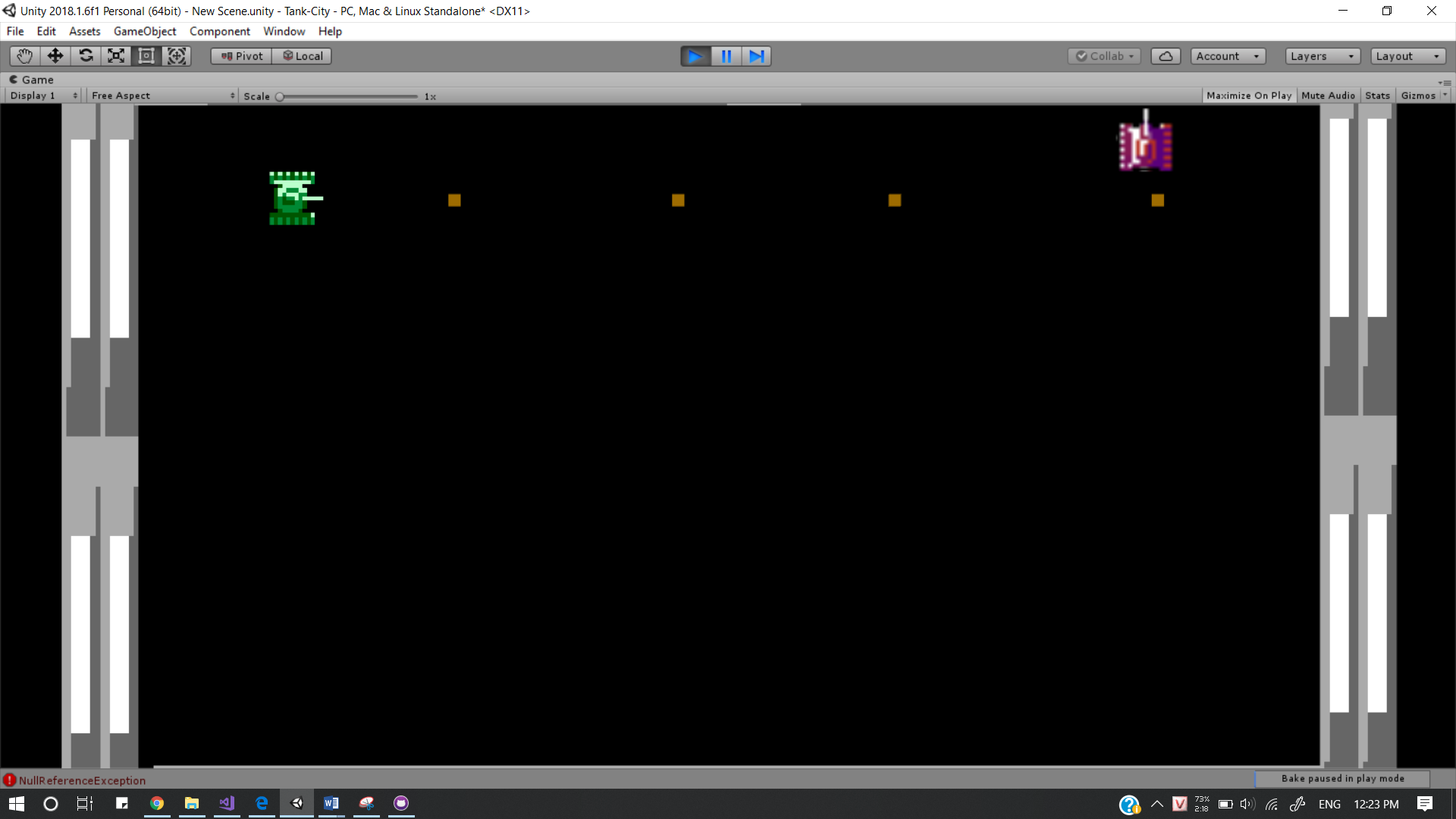
*Trong đó:*

*1 là khoảng cách để di chuyển qua vùng có đạn.*

*0.2 là khoảng cách trừ hao để đưa ra quyết định vị trí di chuyển đến.*

* + Clear\_Time: thời gian đạn bay qua vị trí hiện tại của xe tăng. Clear\_Time dùng để đảm bảo an toàn cho xe tăng khi né, tránh được trường hợp đạn vẫn chưa kết thúc mà xe tăng đã tiếp tục di chuyển và bị đạn tiêu diệt.

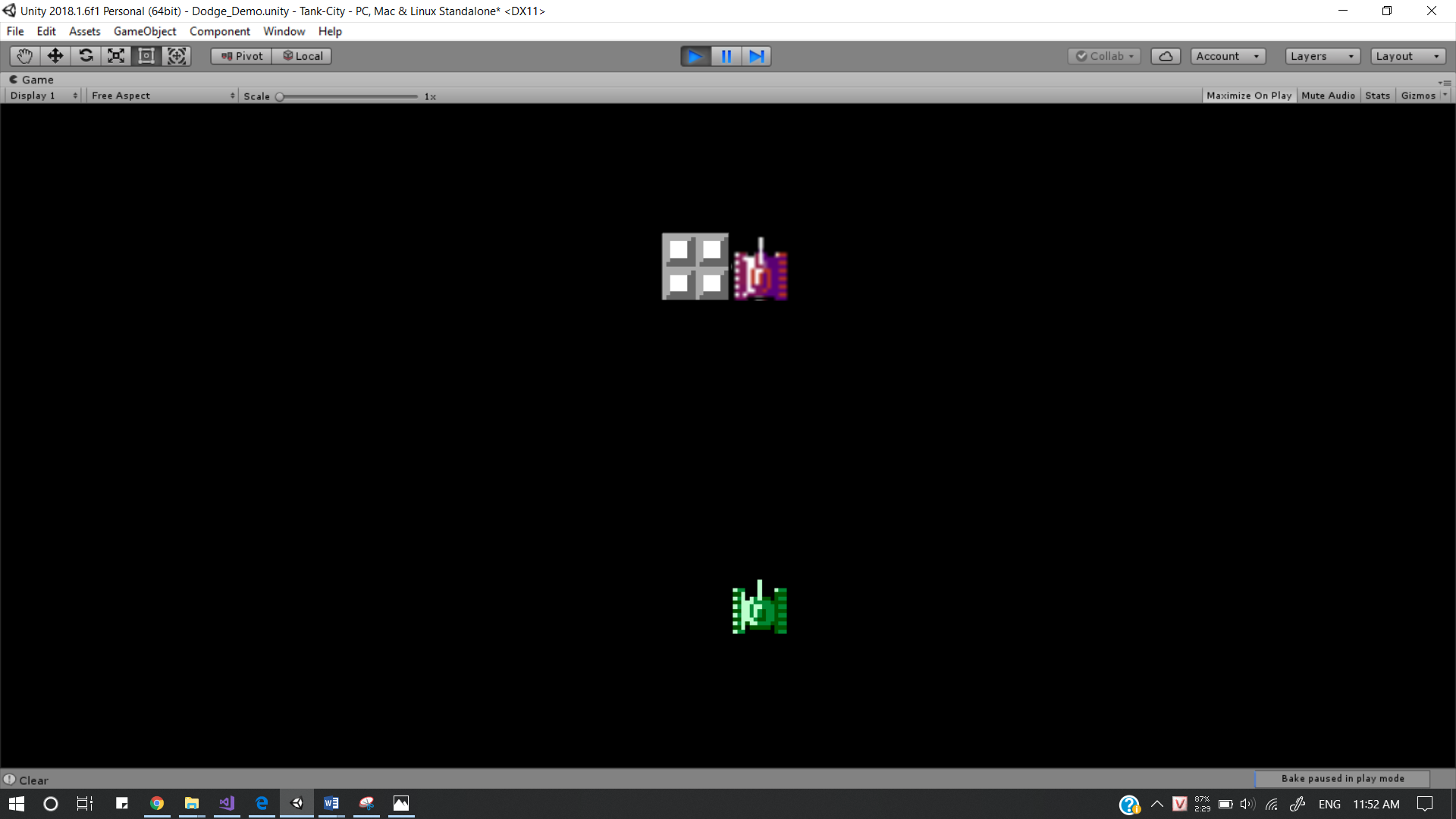
Clear\_Time = (khoảng cách từ vị trí đạn đến xe tăng +1) / vận tốc của viên đạn



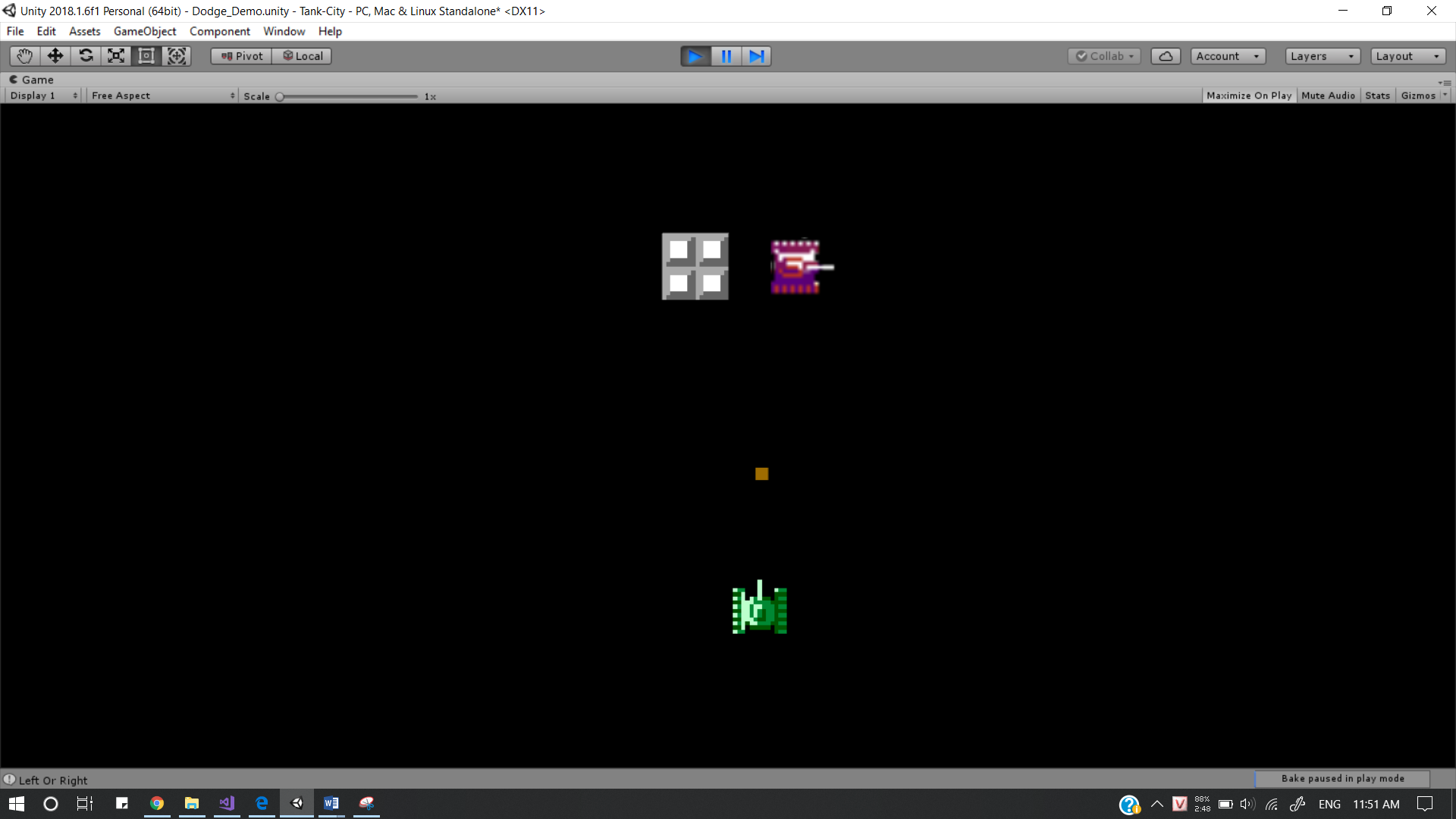
* + Vận tốc vủa viên đạn mặc định là 6.

|  |
| --- |
| // hit là hướng đang xét  // hit: left, right, up, down  public int Threat\_Detection (RaycastHit2D hit, ref float Clear\_Time)  {  if (hit.collider != null) // phát hiện có chướng ngại vật  {  // kiểm tra chắn chắn rằng có chướng ngại vật  // kiểm tra chướng ngại vật có phải là bullet (đạn) không  if((hit.rigidbody != null) && (hit.rigidbody.tag == "bullet"))  {  float Impact\_Time = hit.distance / 6;  float Clear\_Time = (hit.distance + 1) / 6;  float Evade\_Time =(float)(1.2) / this.speed;  // 6 là vận tốc viên đạn  if (Impact\_Time > Evade\_Time)  return 1;  else return 2;  }  }  return 0;  } |

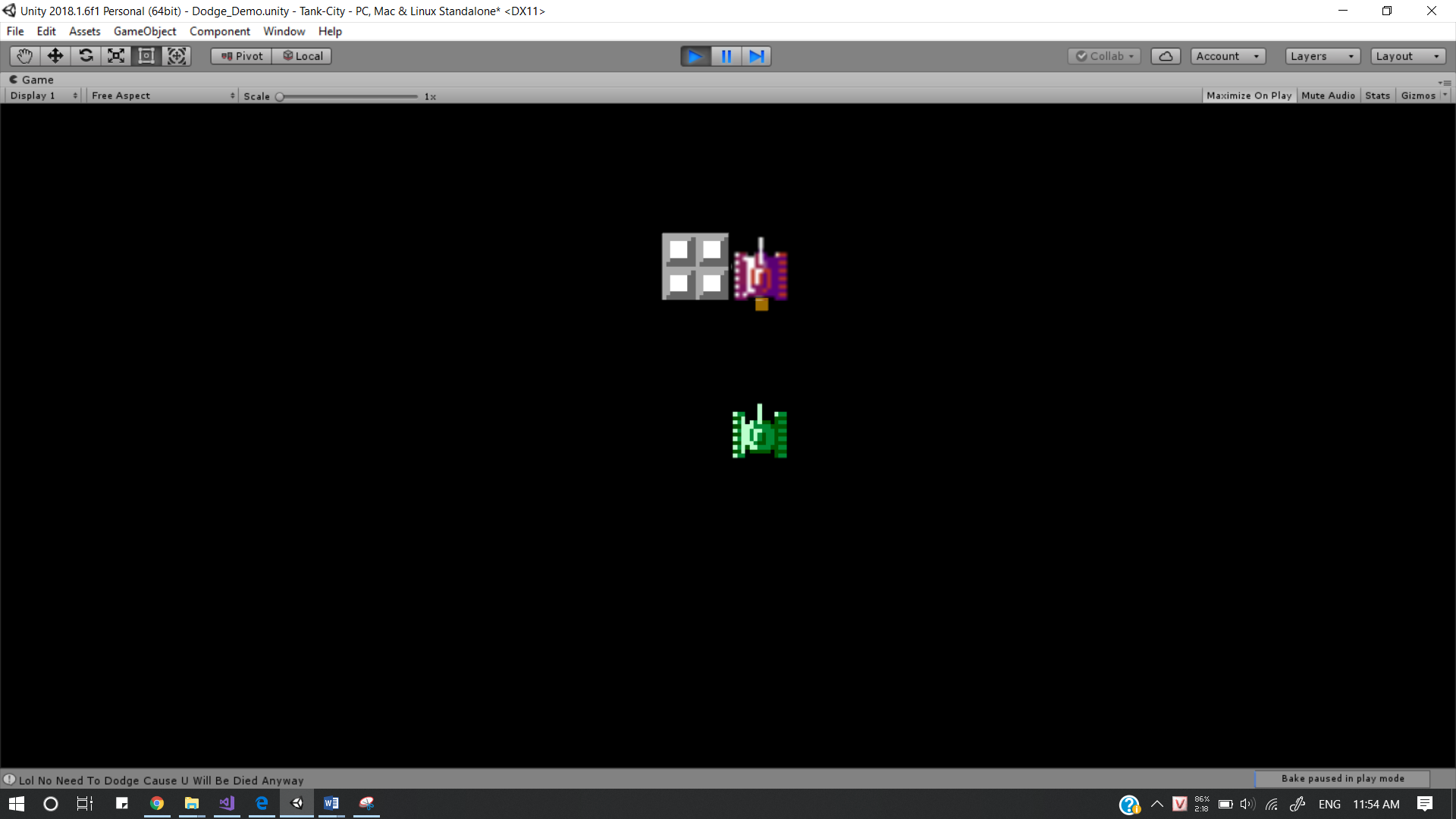
* Từ hàm phát hiện nguy hiểm, xe tăng sẽ rơi vào 3 trường hợp:
  + Trường hợp Threat\_Detection(hit) = 0: không phát hiện ra nguy hiểm (đạn).



* + Trường hợp Threat\_Detection(hit) = 1: phát hiện nguy hiểm (đạn) và có thể né được.



* + Trường hợp Threat\_Detection(hit) = 2: phát hiện nguy hiểm (đạn) nhưng không thể né được.



* Phân tích phương thức né đạn của xe tăng:
  + Xác định hướng bay của đạn bằng cách sử dụng bitmap:

|  |  |  |
| --- | --- | --- |
|  | Trên  0001 (1) |  |
| Trái  0100 (4) | Vị trí xe tăng  0000 | Phải  1000 (8) |
|  | Dưới  0010 (2) |  |

Nhờ vào việc sử dụng bitmap, xe tăng có thể nhanh chóng phát hiện hướng đi của viên đạn. Ví dụ, nếu viên đạn bay đến từ phía bên phải:

|  |
| --- |
| int code = 0; // hướng đạn đến  bool Died = false; // xe tăng có né được hay không  float tmp = -1; // dùng để tính giá trị Clear\_Time (nếu xe tăng né được)  float Clear\_Time = 0;  switch (Threat\_Detection(R,ref tmp))  {  case 1: // trường hợp xe tăng né được đạn  {  code |= 8; // hướng có đạn và là hướng bên phải  if (Clear\_Time < tmp)  Clear\_Time = tmp;  break;  }  case 2: // trường hợp xe tăng không né được đạn  {  Died = true; // xe tăng chắc chắn bị tiêu diệt  break;  }  default: // trường hợp không phát hiện nguy hiểm (đạn)  break;  } |

* + Sau khi phát hiện hướng đạn, tìm hướng né:
    - Nếu đạn bay tới theo phương ngang (từ phía bên trái hoặc bên phải) thì xe tăng sẽ né bằng cách di chuyển lên trên hoặc xuống dưới:

|  |
| --- |
| if (C % 4 == 0) // C là hướng bay của đạn: Trái (4) và Phải (8)  {  // xét node ở phía trên của node (vị trí) xe tăng hiện tại  Node u = this.currentNode.TopNode;  if (Reachable(u)) //kiểm tra độ an toàn của node  nextNode = u; // nếu node an toàn, xe tăng sẽ đi tới node đó  else {  // nếu node ở phía trên không an toàn,  // kiểm tra node ở phía dưới  u = this.currentNode.BottomNode;  // nếu node an toàn, xe tăng sẽ đi tới node đó  if (Reachable(u))  nextNode = u;  }  // đợi tới khi an toàn, xe tăng mới tiếp tục di chuyển tiếp  Wait(Clear\_Time);  } |

* + - Nếu đạn bay tới theo phương dọc (từ phía bên trên hoặc bên dưới) thì xe tăng sẽ né bằng cách di chuyển sang bên trái hoặc bên phải:

|  |
| --- |
| else // Trên (1) và Dưới (2)  {  // xét node ở bên trái của node (vị trí) xe tăng hiện tại  Node u = this.currentNode.LeftNode;  if (Reachable(u)) //kiểm tra độ an toàn của node  nextNode = u; // nếu node an toàn, xe tăng sẽ đi tới node đó  else  {  // nếu node ở bên trái không an toàn,  // kiểm tra node ở bên phải  u = this.currentNode.RightNode;  // nếu node an toàn, xe tăng sẽ đi tới node đó  if (Reachable(u))  nextNode = u;  }  // đợi tới khi an toàn, xe tăng mới tiếp tục di chuyển tiếp  Wait(Clear\_Time);  } |

* + Trong đó:
    - C là hướng đạn bay của đạn đã được xác định ở phần trên.
    - Reachable(u) là hàm kiểm tra độ an toàn của node.

|  |
| --- |
| bool Reachable(Node u)  {  if (u == null || u.obstacle == 9999) // 9999 là tường thép  {  return false;  }  return true;  } |

* + - Wait(Clear\_Time) là hàm đợi cho đến khi hết thời gian Clear\_Time, nghĩa là xe tăng sẽ đợi cho đến khi chắc chắn an toàn thì mới có thể tiếp tục di chuyển.

|  |
| --- |
| IEnumerator Wait (float Time)  {  yield return new WaitForSeconds(Time);  } |

# CHƯƠNG 4: ỨNG DỤNG

1. GIỚI THIỆU CHƯƠNG TRÌNH ỨNG DỤNG:
   1. Các thông tin cơ bản của chương trình:

* Chương trình thiết kế cho một người chơi.
* Hai trường hợp kết thúc trò chơi:
  + Người chơi chiến thắng: là khi người chơi tiêu diệt được tất cả xe tăng của quân địch.
  + Người chơi thua cuộc: là khi người chơi hoặc pháo đài bị tiêu diệt.
  1. Ngôn ngữ lập trình: C#
  2. Các công cụ, phần mềm hỗ trợ:
* Visual Studio
* GitHub
* Adobe Photoshop CC
* Unity
  1. Mô tả trò chơi:
     1. *Các nút điều khiển:*

*Cách 1: Sử dụng bàn phím và chuột:*

* Các phím mũi tên (Left, Right, Up Down) dùng để di chuyển xe tăng.
* Dùng Space để bắn đạn.

*Cách 2: Sử dụng tay cầm:*

* Dùng Joystick trái để di chuyển xe tăng.
* Dùng phím B để bắn đạn.
  + 1. *Pháo đài:*

|  |  |
| --- | --- |
|  | Pháo đài có một mạng và được bảo vệ bởi một lớp tường gạch. Pháo đài có thể bị tiêu diệt bởi xe tăng hoặc đạn của xe tăng quân địch. Pháo đài bị tiêu diệt đồng nghĩa là trò chơi kết thúc. |

* + 1. *Xe tăng người chơi:*

|  |  |
| --- | --- |
|  | Người chơi có thể tiêu diệt xe tăng của quân địch hoặc phá tường bằng cách bắn đạn. Tuy nhiên, người chơi chỉ được bắn lượt đạn tiếp theo sau khi một khoảng thời gian nhất định. |

* + 1. *Xe tăng quân địch:*

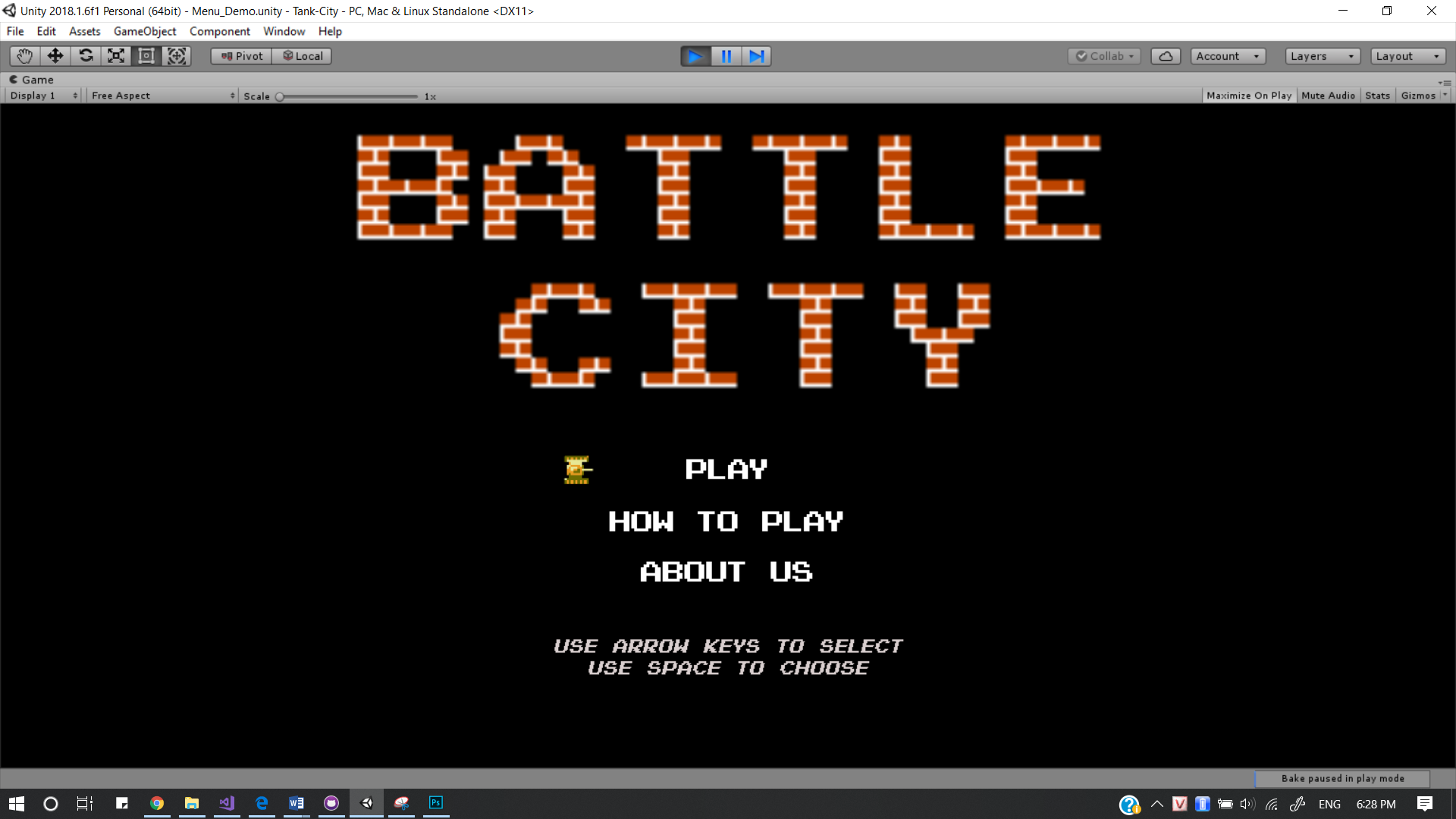
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Quân địch* | *Loại xe* | *Máu* | *Tốc độ*  *di chuyển* | *Thông tin thêm* |
|  | Seeker | 1 | 3 | Tìm đường đi tốt nhất đến người chơi và bắn mục tiêu |
| Ảnh có chứa đối tượng, mẫu họa, văn bản  Mô tả được tạo với mức tin cậy cao | Bulldozer | 1 | 1 | Tìm đường đi tối nhất đến nhà chính và phá huỷ nó (chỉ có thể càng về phía trước) |
| Ảnh có chứa mẫu họa  Mô tả được tạo với mức tin cậy cao | Hunter | 1 | 2,5 | Xem xét vị trí tối ưu nhất giữa người chơi và pháo đài và đi đến nó và thủ tiêu (có thể bắn và né đạn một cách linh hoạt) |

* + 1. *Chướng ngại vật:*

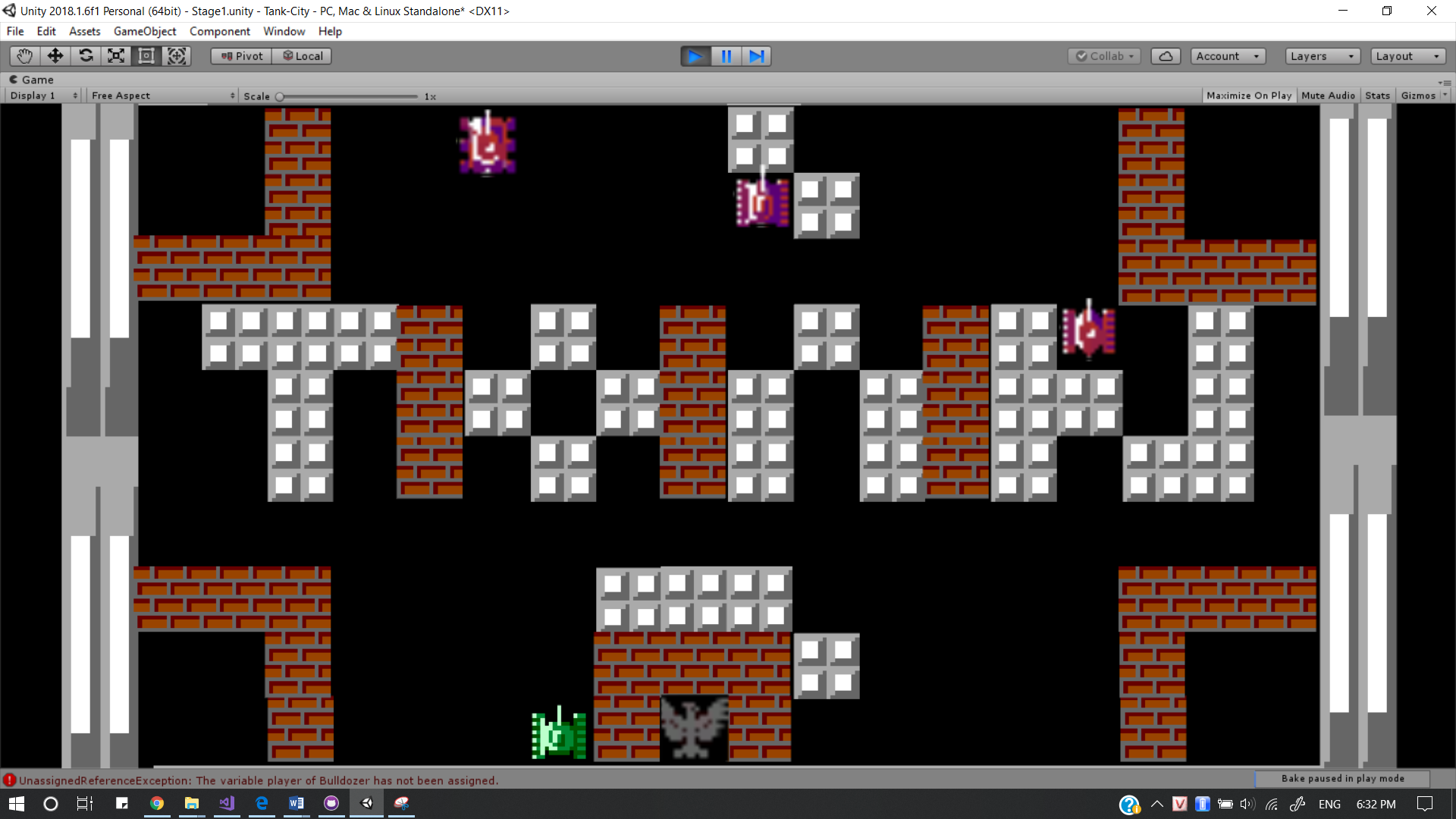
Mỗi màn chơi sẽ có nhiều loại chướng ngại vật khác nhau nằm rải rác xung quanh. Chìa khoá để sống sót và chiến thắng là hãy sử dụng các đặc tính của chúng để tạo lợi thế cho bản thân.

|  |  |  |
| --- | --- | --- |
| *Môi trường* | *Loại môi trường* | *Đặc tính* |
|  | Tường gạch | * Xe tăng và đạn không thể vượt qua tường gạch. * Tường gạch có thể bị phá uỷ bởi 4 lần bắn. |
|  | Tường thép | * Xe tăng và đạn HOÀN TOÀN không thể vượt qua tường thép. |

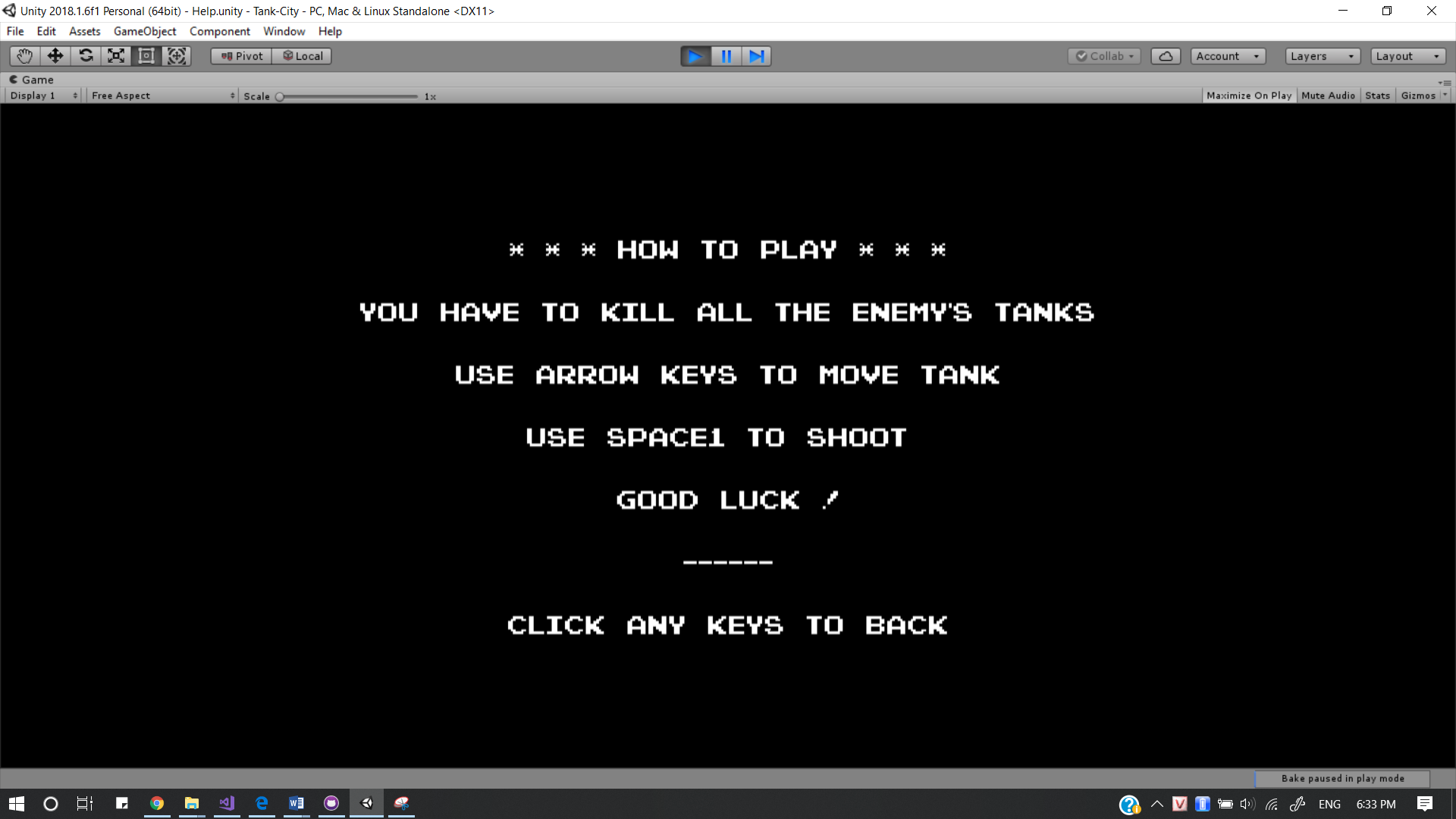
* 1. Giao diện tương tác:
* Giao diện trò chơi được thiết kế theo giao diện Battle City truyền thống:



* Giao diện Main menu có ba sự lựa chọn: *Play*, *Help*, *About*.
* *Play:* dẫn đến màn 1 (stage 1) để người chơi bắt đầu chơi.



* + *Help*: hướng dẫn cách chơi cho người chơi.



* + *About*: giới thiệu về các thành viên của nhóm.

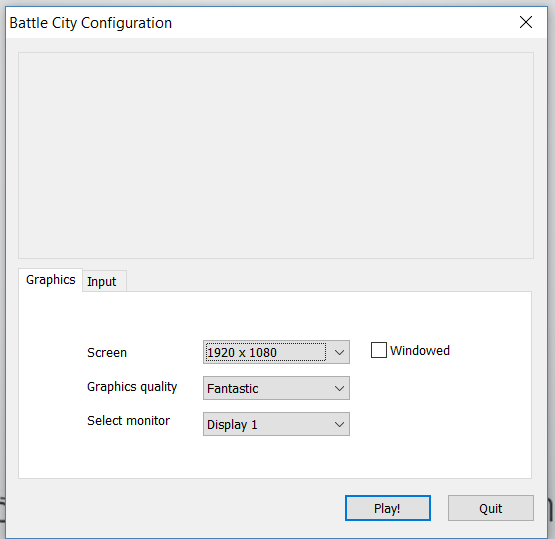


* 1. Các màn chơi:
* Màn chơi 1:
* Màn chơi 2:
* Màn chơi 3:
* Màn chơi 4:
* Màn chơi 5:

1. HƯỚNG DẪN CÀI ĐẶT TRÒ CHƠI:
   1. Hướng dẫn cài đặt từ file exe:

*Bước 1:* Mở thư mục cài đặt chương trình bao gồm các file sau và nháy đúp chuột vào biểu tượng *Demo\_Battle\_City.exe* để khởi động chương trình.

*Bước 2:* Đợi ứng dụng khởi động và nhấn *Play*.

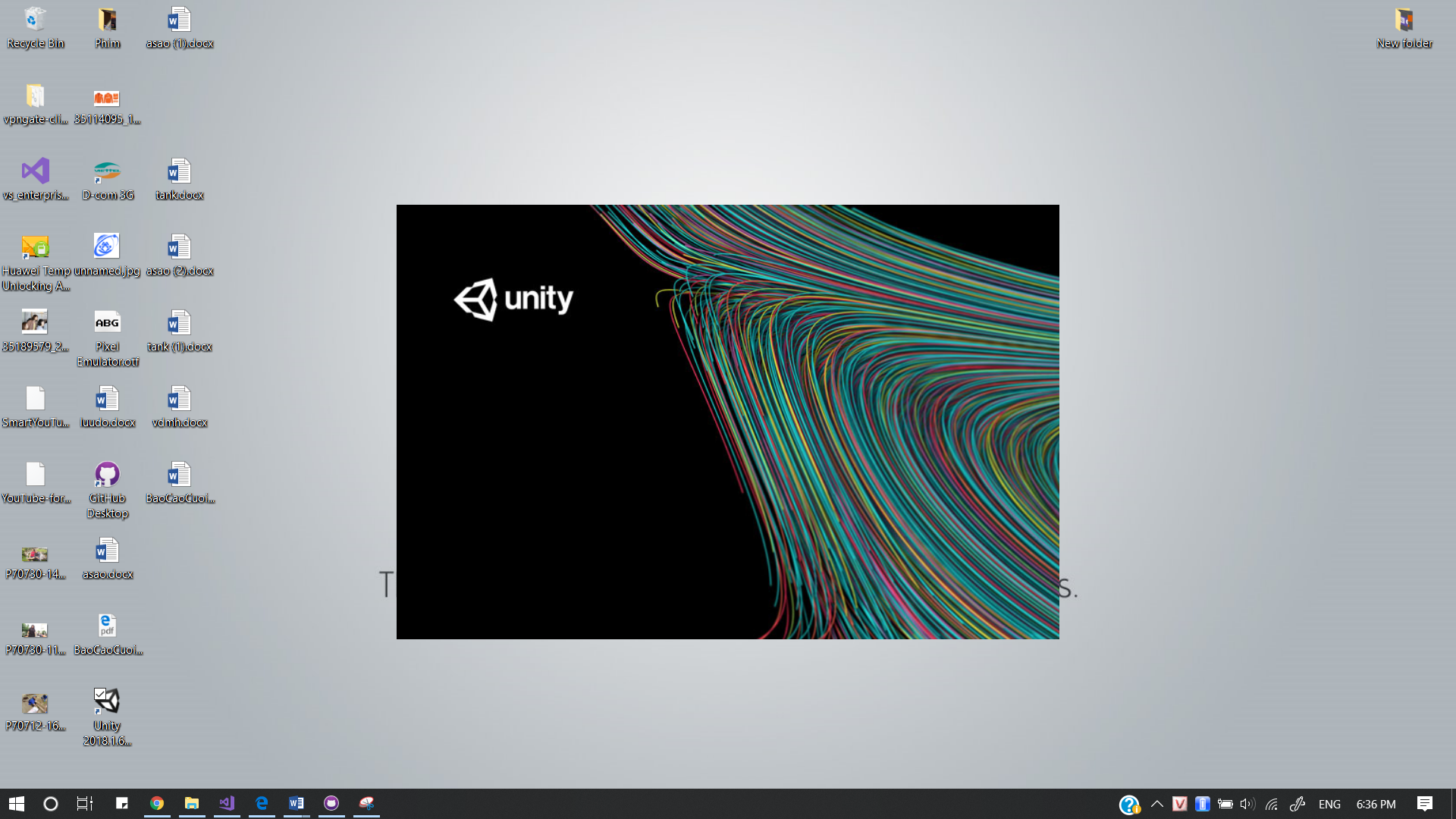


*Bước 3:* Đợi chương trình khởi động Main Menu và sẵn sàng chơi.

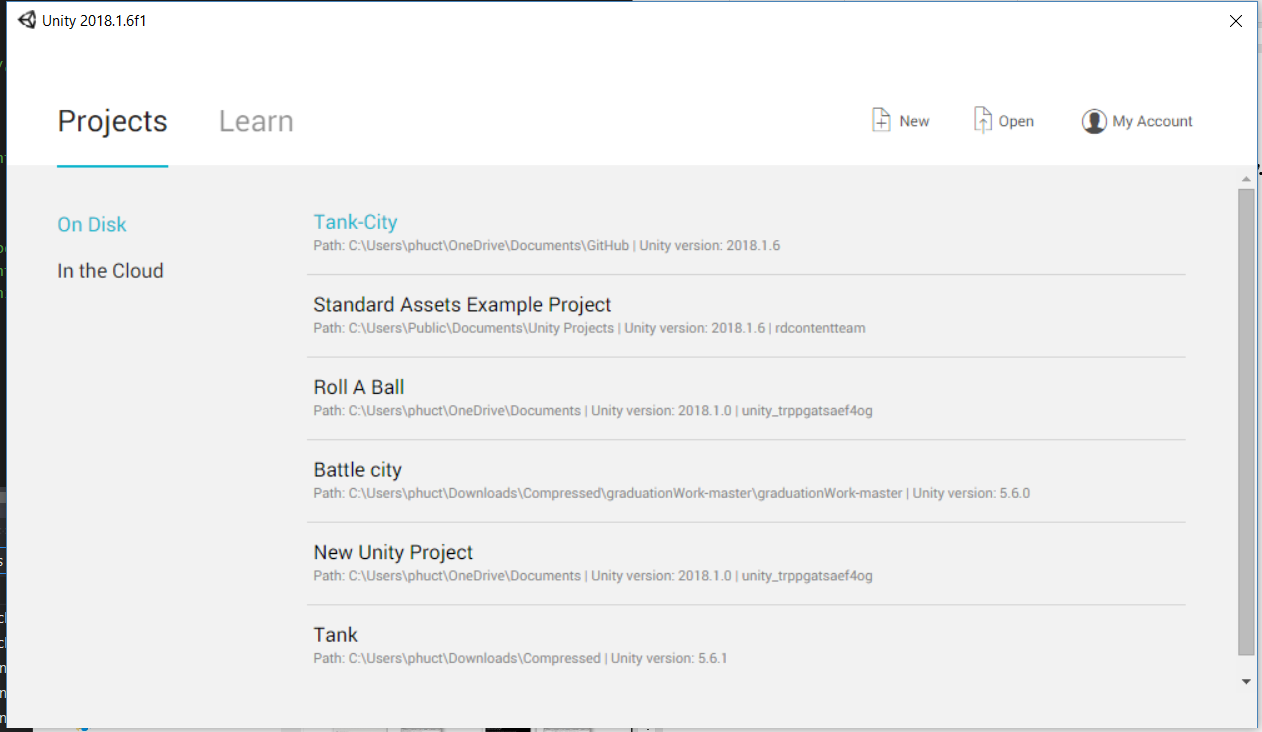
* 1. Hướng dẫn cài đặt bằng Unity:

*Bước 1:* Mở chương trình Unity và đợi chương trình khởi động.



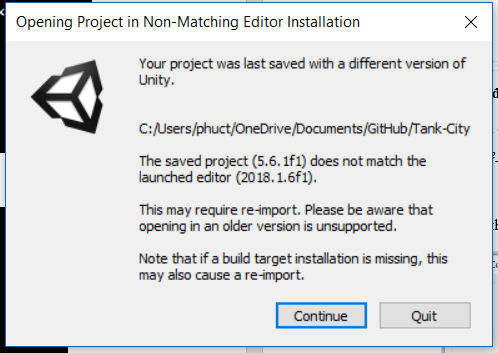


*Bước 2:* Chọn *Open*.



*Bước 3:* Tìm vị trí thư mục Tank-City và chọn *Select Folder*.

*Bước 4:* Nếu gặp phải trường hợp sau, chọn *Continue*.



*Bước 5:* Chọn Scene *Menu\_Demo* (nếu không được hiện thị sẵn) và nhấn vào nút  để khởi chạy chương trình.



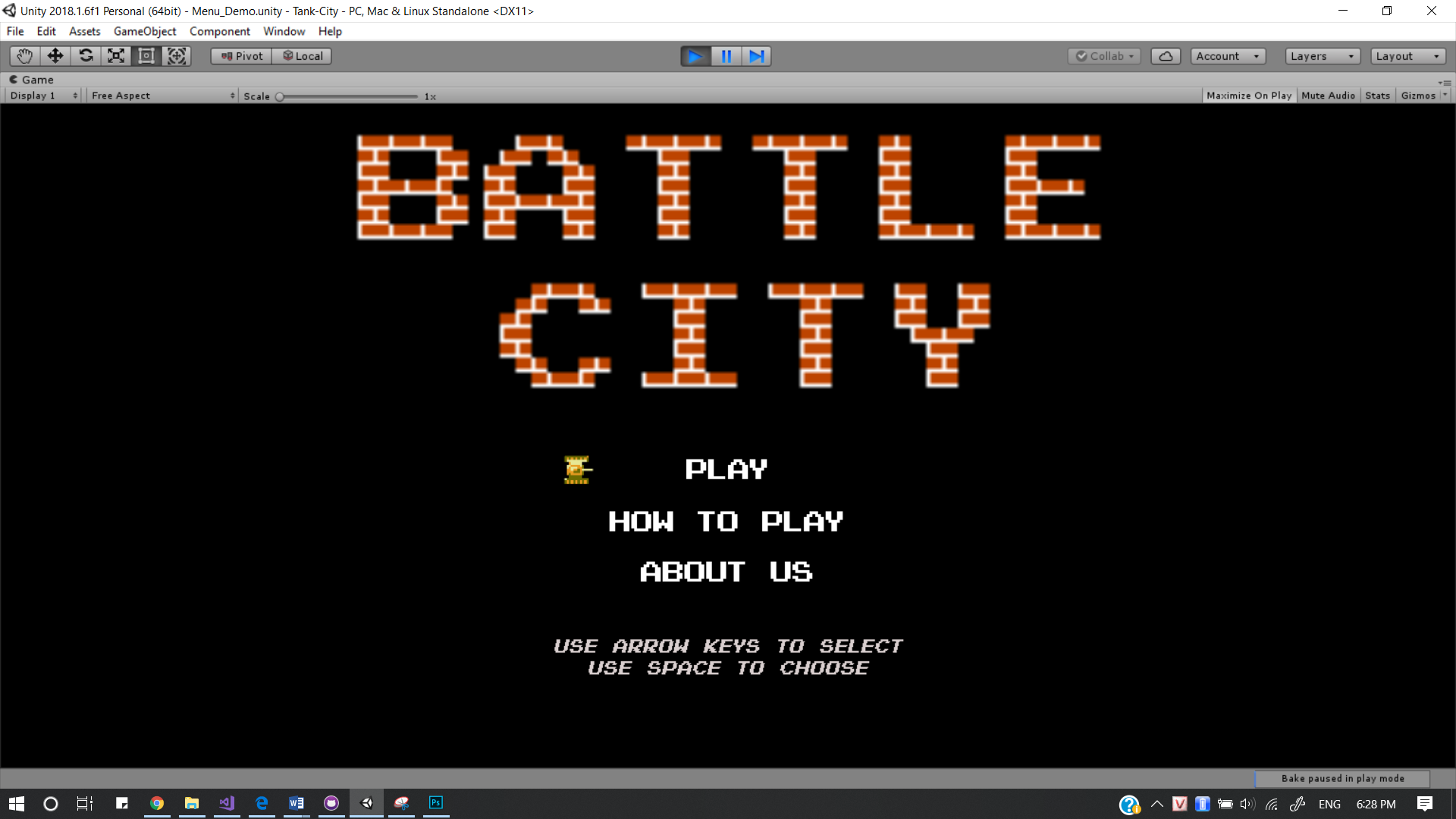
*Bước 6:* Đợi chương trình khởi động Main Menu và sẵn sàng chơi.

1. HƯỚNG DẪN CÁCH CHƠI:
   1. Cách điều khiển xe tăng:
      1. *Bàn phím:*

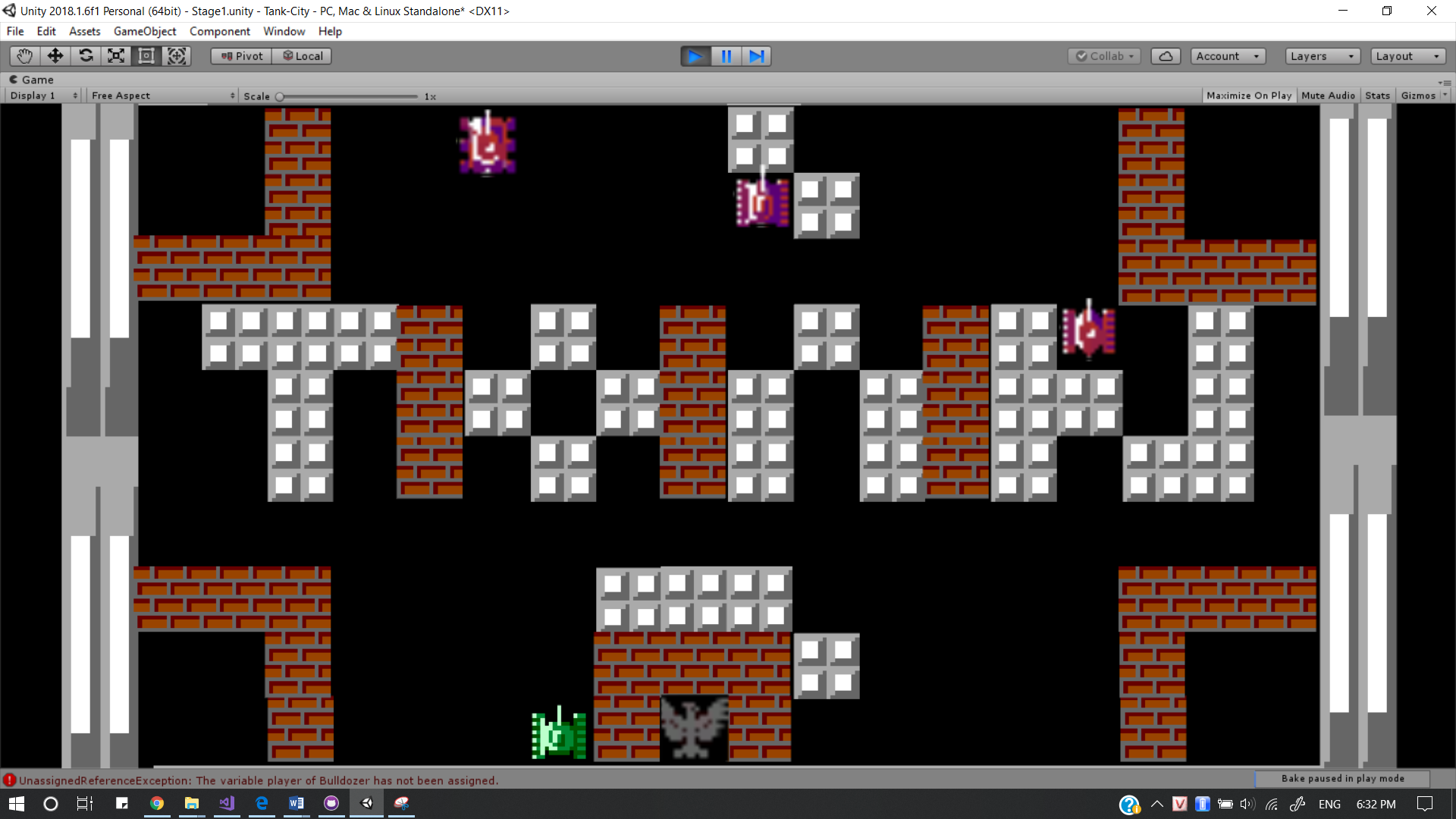
* : dùng các phím mũi tên để di chuyển xe tăng theo bốn hướng (trái, phải, trên, dưới).
* : dùng phím Space để bắn đạn.
  + 1. *Tay cầm:*

|  |
| --- |
| Di chuyển xe tăng  Bắn đạn |

* 1. Sử dụng Main menu:



Khi chương trình đang ở Main menu như hình trên, người chơi có thể thay đổi các lựa chọn bằng hai phím mũi tên (lên và xuống) và xác nhận lựa chọn bằng phím Space. Để bắt đầu chơi, người chơi để lựa chọn ở *Play* như hình trên và nhấn Space để xác nhận lựa chọn. Chương trình sẽ chuyển tới màn chơi 1 (stage 1):



* 1. Mục tiêu của người chơi ở tất cả các màn chơi:
* Tiêu diệt toàn bộ xe tăng của quân địch.
* Bảo vệ pháo đài.

# CHƯƠNG 5: KẾT LUẬN

1. KẾT QUẢ ĐẠT ĐƯỢC:

* Xây dựng được trò chơi với giao diện, hiệu ứng theo kiểu game 8-bit, tạo được được sự gần gũi và quen thuộc đối với người chơi.
* Hai sự lựa chọn về cách chơi (dùng chuột và bàn phím hoặc dùng tay cầm).
* Áp dụng được kiến thức đã học được để xây dựng trò chơi.
* Học thêm được ngôn ngữ lập trình C#.
* Biết cách sử dụng phần mềm Unity cơ bản.
* Trò chơi có thêm nhiều cải tiến mới, tăng độ khó và hứng thư cho người chơi.

1. HẠN CHẾ:

* Nhóm lần đầu tiếp xúc với phần mềm Unity và ngôn ngữ lập trình C# nên gặp khó khăn trong quá trình chuẩn hoá code vào Unity.
* Không có sự đồng nhất ý kiến dẫn đến sự chậm trễ đồ án.
* Thành viên có vấn đề riêng tư dẫn đến sự chậm trễ đồ án.
* Trò chơi chỉ có một màn chơi (stage) duy nhất.

1. HƯỚNG PHÁT TRIỂN:
   1. Về mặt xây dựng môi trường:

* Đồ hoạ: có thể thay đổi để thu hút người chơi hơn:
  + Xe tăng có nhiều sự lựa chọn về màu sắc.
  + Đa sắc màu và cỡ chữ.
  + Thay đổi hiệu ứng nổ bắt mắt.
  + Thêm nhiều hình ảnh minh hoạ cách chơi.
* Âm thanh:
  + Lựa chọn nhạc nền phù hợp.
  + Ghép nhạc nền vào các scene.
  + Thêm âm thanh hiệu ứng.
* Chức năng: bổ sung thêm các chức năng tiện ích cho người chơi:
  + Load game: lưu trạng thái trò chơi hiện hành.
  + High scores: lưu điểm của người chơi vào bảng điểm.
  + Construction: người chơi có thể xây dựng màn chơi.
* Bổ sung thêm các màn chơi.
  1. Về mặt lập trình:
* Hoàn thiện chuẩn hoá theo hướng đối tượng.
* Cải thiện thuật giải A\*: sử dụng Heap thay vì List cho tập Open, và sử dụng cây nhị phân cho tập Close.

TÀI LIỆU THAM KHẢO

* Chương 1: Giới thiệu đồ án

Link: <https://strategywiki.org/wiki/Battle_City>

* Chương 2: Cơ sở lý thuyết:
  + Ngôn ngữ lập trình C#:
* <https://vietjack.com/csharp/csharp_la_gi.jsp>
* <http://it.die.vn/n/ngon-ngu-lap-trinh-c-thang/>
* <https://freetuts.net/c-sharp-la-gi-tong-quan-ve-c-sharp-1045.html>
  + Lý thuyết áp dụng: Thuật toán A\*: Slide bài giảng và các trang web tham khảo:
* <https://www.geeksforgeeks.org/a-search-algorithm/>
* <https://www.stdio.vn/articles/thuat-giai-a-262>

# NỘI DUNG ĐÃ CHỈNH SỬA THEO YÊU CẦU

Chương 1: Giới thiệu đồ án:

* Thay đổi “Mô tả trò chơi chi tiết” thành “Mô tả các thành phần trò chơi cơ bản trên nền tảng Famicom”: từ trang 12 đến trang 15

Chương 3: Phân tích và thiết kế:

* Thay đổi “Sơ đồ lớp Tank” thành hai mục “Xử lý tính toán” và “Thông tin cấu trúc dữ liệu”: từ trang 23 đến trang 27
* Sửa thứ tự nội dung và thêm ví dụ minh hoạ mục 3.2 “Xây dựng thuật giải A\* để tìm đường đi của của các xe tăng”: từ trang 28 đến trang 33.
* Thêm mục 3.3 “Các hàm suy diễn phụ thuật giải A\*”: từ trang 33 đến 35.
* Thêm ví dụ minh hoạ ở trnag 38, 39, 40, 41.

Chương 4:

* Thêm mô tả thành phần trò chơi: từ trang 44 đến trang 46.
* Hướng dẫn cài đặt: thêm cách cài bằng Unity: từ trang 48 đến trang 50.
* Thêm hướng dẫn cách chơi: từ trang 50 đến trang 52.

# LỜI CẢM ƠN

Đầu tiên, Nhóm xin gửi lời cảm ơn chân thành đến tập thể quý Thầy, Cô Trường Đại học Công nghệ Thông tin – Đại học Quốc gia Thành phố Hồ Chí Minh.

Đặc biệt, nhóm xin gửi lời cảm ơn và lòng biết ơn sâu sắc đến cô Huỳnh Thị Thanh Thương – giảng viên môn Trí tuệ nhân tạo. Cô đã trực tiếp hướng dẫn và giải đáp các thắc mắc cho nhóm trong quá trình làm đồ án.

Trong suốt thời gian thực hiện đề tài, nhóm đã vận dụng được những kiến thức nền tảng đã được tích luỹ từ các buổi học và phối hợp với việc tự học hỏi và nghiên cứu kiến thức mới. Đồng thời, những ý kiến đóng góp của cô đã giúp đồ án của nhóm ngày càng hoàn thiện hơn. Nhờ đó, đồ án đã thuận lợi hoàn thành.

Nhóm tin tưởng rằng những kiến thức này sẽ là thành trang cho mỗi thành viên trong tương lai.

*Thành phố Hồ Chí Minh, ngày 04 tháng 07 năm 2018*