**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA KHOA HỌC MÁY TÍNH**

****

**BÁO CÁO ĐỒ ÁN MÔN HỌC**

**ĐỀ TÀI: GAME BATTLE CITY**

**(Đã chỉnh sửa theo góp ý của giảng viên)**

**Giảng viên hướng dẫn: Th.S Huỳnh Thị Thanh Thương**

**Lớp: CS106.I22**

**Nhóm sinh viên thực hiện:**

|  |  |  |
| --- | --- | --- |
| **STT** | **Họ và Tên** | **MSSV** |
| **1** | **Trần Quốc Phong** | **16520930** |
| **2** | **Nguyễn Huỳnh Phú** | **16520934** |
| **3** | **Diệp Minh Tâm** | **16521057** |
| **4** | **Trần Triều Tân** | **16521081** |

**Thành phố Hồ Chí Minh, 04/06/2018**

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

……………………………………………………………………………………….

.………………………………………………………………………………………

.………………………………………………………………………………………

.………………………………………………………………………………………

.………………………………………………………………………………………

.………………………………………………………………………………………

.………………………………………………………………………………………

.………………………………………………………………………………………

.………………………………………………………………………………………

.………………………………………………………………………………………

.………………………………………………………………………………………

.………………………………………………………………………………………

.………………………………………………………………………………………

.………………………………………………………………………………………

.………………………………………………………………………………………

.………………………………………………………………………………………

.………………………………………………………………………………………

.………………………………………………………………………………………

……………………..,ngày……….tháng………năm 2018

Người nhận xét

(Ký và ghi rõ họ tên)

Mục lục

Lời mở đầu...........................................................................................................................4

Chương 1: Giới thiệu bài toán.............................................................................................6

1.1 Nguồn gốc trò chơi.............................................................................................6

1.1.1 Giới thiệu sơ lược.........................................................................................6

1.2 Mô tả trò chơi......................................................................................................6

Chương 2: Cơ sở lý thuyết..................................................................................................7

2.1 Nhận dạng và cách giải quyết vấn đề.................................................................7

Chương 3. Phân tích và thiết kế...........................................................................................9

3.1 Phân tích bài toán................................................................................................9

3.2 Cấu trúc dữ liệu và cách biểu diễn trạng thái của bài toán...............................11

3.2.1 Cấu trúc dữ liệu...........................................................................................11

3.3 Các vấn đề và thuật giải....................................................................................13

3.3.1 Một số vấn đề quan trọng...........................................................................13

3.3.2 Vấn đề va chạm các Tank...........................................................................16

3.3.3 Vấn đề dứt điểm ván đấu............................................................................16

3.3.4 Vấn đề tìm đường đi...................................................................................17

3.3.5 Vấn đề tìm đường đi A\*.............................................................................19

3.3.6 Vấn đề gặp đá và tự động bắn....................................................................20

3.4 Ví dụ minh hoạ.................................................................................................21

Chương 4. Ứng dụng.........................................................................................................26

4.1 Giới thiệu chương trình ứng dụng....................................................................26

4.2 Cài đặt...............................................................................................................27

4.2.1 Ngôn ngữ lập trình JavaScript...................................................................27

4.3 Kết quả chạy chương trình................................................................................37

Chương 5. Kết luận............................................................................................................39

5.1 Kết quả đạt được...............................................................................................39

5.2 Hạn chế.............................................................................................................39

5.3 Hướng phát triển...............................................................................................40

Tài liệu tham khảo.............................................................................................................40

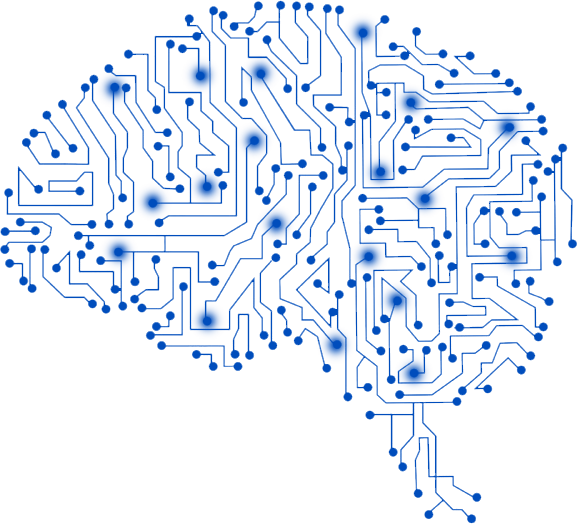
Chương 6: Phân công công việc và đánh giá chất lượng làm việc………………………41

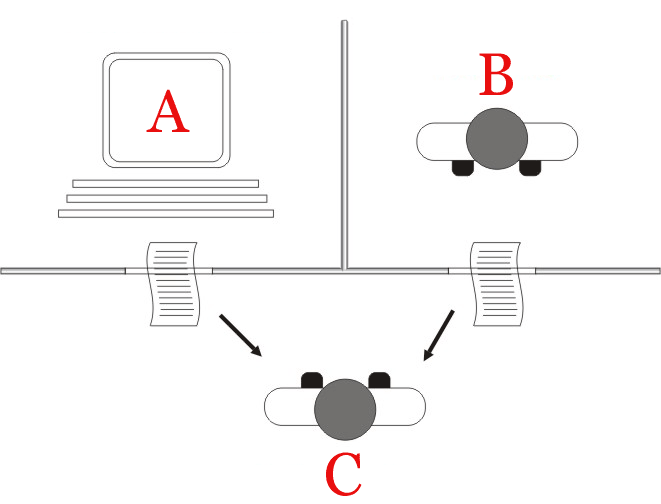
6.1 Phân công công việc…………………………………………………………..41

6.2 Đánh giá chất lượng làm việc…………………………………………………41

Tóm tắt nội dung chỉnh sửa……………………………………………………………42

Lời mở đầu

Ngày nay, ở Việt Nam cụm từ được nhắc đến nhiều nhất trên các phương tiện truyền thông đại chúng đó là “Cuộc cách mạng công nghiệp 4.0”. Vậy “Cuộc cách mạng công nghiệp 4.0” là gì? Nhắc đến cụm từ trên thì không thể không nhắc đến “Trí tuệ nhân tạo” – là một phần của “Cuộc cách mạng công nghiệp 4.0”, do đó nhóm chúng tôi sau khi bàn bạc với nhau đã chọn Game Battle City để làm đồ án môn học và báo cáo – vì game này chơi khá là vui và cách chơi đơn giản. Qua môn học này nhóm chúng tôi có tiếp xúc, va chạm với AI nên sẽ bớt bỡ ngỡ, đồng thời cũng hình dung được thế nào là “Cuộc cách mạng công nghiệp 4.0”. Dù đây không phải là đồ án cao siêu gì, nhưng sẽ để lại dấu ấn khó phai trong mỗi thành viên. Hơn nữa trí tuệ nhân tạo là xu hướng tất yếu trong sự phát triển của loài người ở thế kỉ 21.



Chương 1. Giới thiệu bài toán

1.1. Nguồn gốc trò chơi

1.1.1 Giới thiệu sơ lược

“Battle City” với tên gọi thông thường theo tiếng Việt “Bắn xe tăng”, là một game bắn súng đa hướng được sản xuất cho dòng máy “Family Computer” và phát hành năm 1985 bởi . Người chơi sẽ điều khiển xe tăng của mình để bắn vào đối thủ - xe tăng do máy tính điều khiển, đồng thời bảo vệ khung thành của mình. Tuy game có tất cả 35 màn chơi nhưng do thời gian và khả năng có hạn nên nhóm chúng tôi chỉ demo lại 1 màn game.

1.2. Mô tả trò chơi

Đến với game Battle City chúng ta có:

* **Bản đồ của game hiển thị gồm**: các ô gạch, bụi cây, khung thành, người chơi, AI được sắp xếp theo một trật tự đã định sẵn.
* **Di chuyển**: người chơi dùng các phím A – D – W – S, hoặc các phím mũi tên, để di chuyển theo thứ tự trái – phải – lên – xuống. AI do máy điều khiển cũng di chuyển tương tự.
* **Chiến đấu**: người chơi dùng phím Space để bắn ra đạn, AI do máy tính điều khiển bắn ra đạn.

Quyết định thắng – thua như sau:

* **Thắng**: người chơi sẽ thắng khi tiêu diệt hết AI, đồng thời bảo vệ được khung thành.
* **Thua**: người chơi bị AI tiêu diệt hoặc là khung thành bị AI tiêu diệt.

**Lưu ý: đối tượng “người chơi” được nhắc đến bên trên sẽ do giảng viên điều khiển, còn AI là được sinh viên lập trình sẵn.**

1. NAMCO là một tập đoàn Nhật Bản điều hành các trung tâm trò chơi và các công viên giải trí.

Chương 2. Cơ sở lý thuyết

2. Nền tảng lý thuyết

2.1. Nhận dạng và cách giải quyết vấn đề

Trò chơi Battle City thuộc dạng tìm kiếm có thông tin, nghĩa là tìm tiếm dựa trên tri thức có được của bài toán.

Ý tưởng ban đầu: Thuật toán heuristic A\* - do giảng viên gợi ý.

Thuật toán heuristic A\*: Thuật toán heuristic này đánh giá một nút dựa trên chi phí từ nút gốc đến nút đó – *g(n)*, cộng với chi phí từ nút đó đến nút đích – *h(n)*.

Do giá trị *g(n)* cho biết chi phí đường đi từ nút gốc đến nút *n* và *h(n)* là ước lượng chi phí đường đi ngắn nhất từ nút *n* đến nút đích do đó: *f(n) = g(n) + h(n)*, và *f(n)* = ước lượng chi phí của lời giải “tốt nhất” qua *n.*

Vì vậy, nếu ta tìm kiếm một lời giải “tốt nhất” thì cách dễ nhất là lấy nút có giá trị *g(n) + h(n)* là nhỏ nhất. Hơn nữa, nếu *hàm heuristic h(n)* thoả một số điều kiện nhất định thì tìm kiếm A\* sẽ đầy đủ và tối ưu – hiển nhiên là *h(n)* là *hàm heuristic* chấp nhận được.[1]

Mã giả thuật toán heuristic A\* như sau:

Mỗi đỉnh p tương ứng với một số độ tốt *f(p) = g(p) + h(p).*

Bước 1: Open := {s};

Close := {};

g(s) := 0;

f(s) = h(s);

Bước 2: While (Open != {})

2.1 Chọn p thuộc Open có f(p) nhỏ nhất.

2.2 Nếu p là trạng thái kết thúc thì thoát, thông báo kết quả.

2.3 Chuyển p qua Close, và mở các q sau p.

2.4 Xét các đỉnh kề q của p.

TH1: q không thuộc Close, không thuộc Open

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

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

Thêm q vào Open;

TH2: q thuộc Open

if(g(q) > g(p) + cost(p, q))

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

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

prev(q) = p;

TH3: q thuộc Close

if(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 hưởng từ sự thay đổi của q;

Bước 3: Không tìm được.

Chương 3. Phân tích và thiết kế

3.1 Phân tích bài toán

Nhắc lại: Trò chơi Battle City thuộc dạng tìm kiếm có thông tin, nghĩa là tìm tiếm dựa trên tri thức có được của bài toán.

Giả sử ta mô hình hoá bản đồ của trò chơi như sau:

START: Vị trí bắt đầu của AI

END: Vị trí mục tiêu mà AI cần tìm đến

Có 4 thao tác di chuyển: lên – trái – xuống – phải.

1: Là những ô có thể di chuyển trên bản đồ.

0: Là những ô có vật cản trên bản đồ. x

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | **START** | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | **0** | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | **0** | **0** | **0** | 1 | 1 |
| 1 | **0** | 1 | 1 | **0** | **0** | 1 | 1 |
| **0** | 1 | 1 | 1 | 1 | 1 | 1 | **0** |
| **0** | **0** | 1 | **0** | 1 | 1 | 1 | 1 |
| 1 | 1 | **0** | 1 | 1 | 1 | **END** | 1 |
| 1 | 1 | **0** | **0** | 1 | 1 | 1 | 1 |

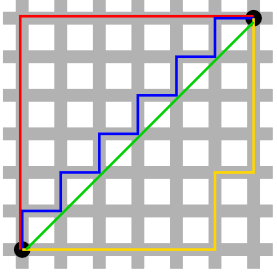
Hỉnh 3.1

y

Câu hỏi đặt ra là: Tìm đường đi có khoảng cách ngắn nhất từ START tới END ?

Dựa vào cách mô hình hoá này, ta thấy START có toạ độ xác định, END cũng có toạ độ xác định => Dùng khoảng cách Euclid, = , cũng có thể dùng khoảng cách Manhattan,

Ví dụ về khoảng cách Euclid và khoảng cách Manhattan

Các đường gấp khúc như màu đỏ, xanh dương, vàng biểu diễn cho khoảng cách Manhattan, đường thẳng màu xanh lá cây biểu diễn cho khoảng cách Euclid.

Về mặt lập trình, để đạt hiệu năng cao khi thực thi chương trình chúng ta sẽ dùng khoảng cách Manhattan, do trong quá trình tính toán không phải sử dụng phép nâng luỹ thừa bậc 2 và phép khai căn bậc 2.

Sử dụng thuật toán heuristic A\* với công thức: *f(n) = g(n) + h(n)* ở chương 2, *h(n)* sẽ là khoảng cách Manhattan. Về bản chất thì A\* gồm thuật toán tìm đường đi ngắn nhất (Dijkstra) trong đồ thị có kèm theo hàm đánh giá.

Biểu diễn bài toán tìm kiếm.

**Trạng thái:** Vị trí tìm kiếm sẽ bắt đầu tại vị trí bất kì trong bản đồ.

**Trạng thái ban đầu:** Tất cả trạng thái đều có thể là trạng thái ban đầu, tuy nhiên không phải trạng thái nào cũng tới được trạng thái đích xác định trước.

**Hàm trạng thái con:** Phát sinh một trạng thái mới bằng cách di chuyển theo quy tắc lên – phải – xuống – trái.

**Chi phí đường đi:** Mỗi bước đều có chi phí bằng 1 – *g(n)* và kèm theo hàm *lượng giá heuristic h(n).*

3.2 Cấu trúc dữ liệu và cách biểu diễn các trạng thái của bài toán.

3.2.1 Cấu trúc dữ liệu

- Sơ đồ hướng đối tượng tổng quát:

Shape  
x  
y  
lastmove

Action

type

exists, image

current\_direction

Ego,Block,Steel, River,Forest  
X  
Y

type

exists

Bullet

X

Y

Lastmove

Type

Exist

TankAI

X

Y

Lastmove

action

type

exists

current\_direction

TankPlayer

X

Y

Lastmove

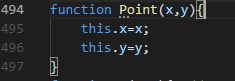
action

type

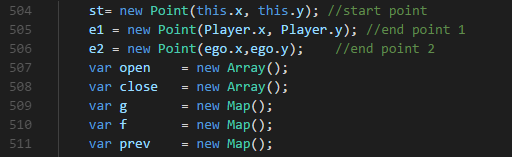
exists

current\_direction

Biểu diễn cấu trúc dữ liệu Point thông qua ngôn ngữ lập trình JavaScript:



* Dùng mảng để biểu diễn cho tập open, tập close.
* Dùng cấu trúc dữ liệu Map để biểu diễn cho g, f, prev, để lưu các giá trị tương ứng với các ô được xét tới.



3.3 Các vấn đề và thuật giải.

Tìm đường cho Tank AI đến vị trí mục tiêu theo đường gần nhất, sau khi đến được vị trí có thể thực hiện việc kết thúc trò chơi (tiêu diệt đại bàng hoặc tiêu diệt người chơi) thì thực hiện hàm bắn, công việc này sẽ được thực hiện trên mỗi vòng loop.

X: tọa độ x;

Y: tọa độ y;

Lastmove: (1, 2, 3, 4) dùng để tính this.x và this.y tiếp theo 1 up, 2 down, 3 right, 4 left

Action: Nếu action = 1 => tankPlayer bắn

Type: type = 1 => player, type = 2 => AI, , type = 6 => EGO, type = 7 => player bullet, type = 8 => AI bullet

Exists: false là đã trúng đạn, true là còn sống

Image: dùng để tạo đường dẫn trực tiếp đến hình ảnh để render lên màn hình

current\_direction: hướng quay mặt hiện tại ( 1 , 2 ,3 ,4 tương ứng với lastmove)

**Giải thuật:**

3.3.1 Một số vấn đề quan trọng:   
 1. Kiểm tra ô (x,y) có phải là địa hình (River, Steel, Block hay không)  
 Input(x,y)  
 Output: true nếu là các dạng địa hình kể trên  
 false nếu không phải

Các mảng lưu giá trị của River, Steel, Block là biến toàn cục.  
 Ý tưởng thực hiện:  
 Cho vòng lặp chạy qua từng đối tượng trong các mảnh River, Steel, Block  
 Nếu trùng x,y thì dừng hàm, trả về true  
 Nếu kết thúc vòng lặp mà không trùng thì trả về false  
 Mã giả  
 For(i=0; I < Object\_Array.length; i++)

If(Object\_Array[i].x ==x && Object\_Array[i].y ==y)

Return true;

Return false;

2. Cập nhật Array (Cập nhật các Object và hướng đạn để thực hiện việc né đạn cho AI A\*)  
 Input(none)

Output(Array Result)

Ý tưởng thực hiện:

Phần 1: cho 2 vòng lặp chạy từ (x=50,y=50) tới (x=width, y = heigh) đồng thời dùng hàm CheckSteel, CheckRiver ( đã nêu ở vấn đề 1) kiểm tra, nếu hàm trả về true thì set Result[y/50-1][x/50-1]= 1 else = 0;

Mã giả:

For(i = 0; i<height; i++)

For(j = 0; j < width; j++)

If(CheckRiver(j,i) || CheckSteel(j,i))

A[i][j] = 1;

Else

A[i][j] = 0;

Phần 2: đối với Bullet: cho biến i lần lượt chạy qua hết các phần tử trong mảng Bullets, xét vị trí (x,y, lastmove) từ đó set 1 cho tất cả các vị trí sẽ nằm trên đường bay của bullet[i]

Mã giả:

Bước 1: vòng lặp chạy qua từng phần tử của mảng Bullet

For(i=0; i<Bullet\_Array; i++)

Bước 2:

2.1

Kiểm tra hướng bay của bullet

If(Bullet\_Array[i].lastmove == 1) // 1 = up ⬄ y+=50;

2.2

Set 1 cho các ô nằm trên đường bay của bullet

Lastmove = 1;

For(y = Bullet\_Array[i].y; y <= height; y+=50) // lasmove = 2 ( y> 0; y -=50)

A[y/50-1] [Bullet\_Array[i].x] = 1;

Lastmove =3

For( x= Bullet Array[i].x; x <=width; x+=50) //lastmove = 4 ( x >0; x-=50)

A[Bullet\_Array[i].y/50-1][x/50-1] = 1

/50 – 1 là để map từ màn hình đồ họa về cấu trúc mảng

Bước 3:

Return A;

3.3.2. Vấn đề tránh va chạm giữa các tank

Ý tưởng: với mỗi tank, nếu lastmove tiếp theo mà = 1; và ở phía y + 50 của nó có 1 tank khác, thì lập tức set lastmove = 0; // nhường đường;

Input (none)

Output(none) thực hiện bằng procedure

Các bước thực hiện:

1: cho biến i chạy qua từng xe trong mảng tankAI

For(i=0; i<tanks[i].length; i++)

2: Kiểm tra nếu tank gọi hàm có phải tank[i] hay không, nếu không phải thì mới xét tiếp:

If(this != tank[i])

2.1

Kiểm tra lastmove của tank gọi hàm(this) với vị trí sắp đi tới của nó đã có tank hay chưa

If((this.lastmove ==1) && ((this.x == tank[i].x) && (this.y+50 == tank[i].y) )|| (this.x == Player.x && this.y+50 == Player.y)

Return 1;

Tương tự cho các lastmove khác, cách + - đã nêu trong vấn đề 2

3.3.3 Vấn đề dứt điểm ván đấu (bắn người chơi và bắn Eagle)

Ý tưởng, nếu xe đang quay mặt lên mà trên đường đi đó có xuất hiện người chơi hoặc đại bàng thì thực hiện hàm bắn

Thực hiện bằng procedure

Input: x,y,direction

Output: true nếu trên hướng quay mặt AI có mục tiêu

False nếu không phải true;

Bước 1: kiểm tra hướng quay của tank = lastmove

If(direction ==1)

Bước 2: kiểm tra các ô trên đường quay mặt của AI) bằng vòng lặp

For(i=y; i<=height; i+=50)

If(i==target.y && x == target.x)

Return false;

Tương tự với 3 hướng còn lại

Return false;

3.3.4 Tìm đường đi(setMove1 và setMove3) (tìm theo thuật toán Người đưa thư)

Input: dùng biến toàn cục

Output: this.lastmove (1,2,3,4)

Bước 1: đặt giá trị cho biến (giá trị vô cùng)

Left = right = up = down = 999; (999 là quy ước)

Bước 2: Kiểm tra nếu có đi qua trái phải trên dưới được không ( bước đi tiếp theo không ra ngoài màn hình đồ họa, và không trở về vị trí trước đó)

if(this.x == width || this.lastmove ==4)

right =1000; //can't go to the right

else

if(this.x == 50 || this.lastmove == 3)

left =1000; //can't go to the left

if(this.y ==height || this.lastmove == 2)

down = 1000; //cant go down

else

if(this.y == 50 || this.lastmove ==1)

up =1000; // can't go up;

//1000 là không đi được;

Bước 3: nếu đi được qua hướng đó thì kiểm tra ô đó có phải là Water||Steel hay không, nếu không phải thì tính tiếp khoảng cách Mahatan của ô đó, nếu không phải thì trả về giá trị 1000;

if(right!=1000){ // x+50

if(checkWater(this.x+50,this.y)=== false && checkSteel(this.x+50,this.y)===false)

right = (Math.abs(this.x + 50 - ego.x) + Math.abs(this.y - ego.y))/50;

else

right = 1000;

//tương tự tính tất cả 3 giá trị còn lại // với lastmove 1 thì chọn ego làm mục tiêu, với lastmove3 thì chọn player làm mục tiêu

Bước 4:

Tính min(Left,Right,Up,Down) và chọn bước đi tiếp theo

var move = Min(up, down, left, right);

if(move === up)

this.lastmove= 2;

if(move === down)

this.lastmove= 1;

if(move === left)

this.lastmove= 4;

if(move === right)

this.lastmove= 3;

3.3.5 Tìm đường đi A\*(setMove2)

Chọn H (chọn min của Manhattan đến Player và Manhattan đến Ego)

Thực hiện A\*

Chọn bước đi tiếp theo dựa trên việc truy ngược, chỉ chọn 1 bước đi đầu tiên

Hàm H:

Input: x,y

Output: giá trị H của ô (x,y)

Thực hiện

M1 = abs(x-Player.x)+abs(y-Player.y)

M2 = abs(x- ego.x) + abs(y-ego.y)

Return min(M1,M2);

Thực hiện A\*

3.3.6. Vấn đề về gặp đá và tự động bắn  
Input: none (dùng biến toàn cục)  
Output: cập nhật this.lastmove hoặc ra chỉ thị bắn

Ý tưởng thực hiện:  
Nếu bước tiếp theo của tank là đá thì gọi hàm bắn theo hướng đó, tank dừng lại 1 frame, nếu không phải thì thực hiện update x, y

Bước 1: kiểm tra bước tiếp theo của tank  
 if(this.lastmove ==1 ){ // (1,2,3,4)

Bước 2: kiểm tra vị trí tiếp theo có phải block hay không, nếu phải thì quay mặt về hướng đó, đồng thời bắn, thoát hàm và không update x,y.  
 if(checkBlock(this.x, this.y+50)==true){

This.current\_direction = 1;

This.fire();

Return;

}

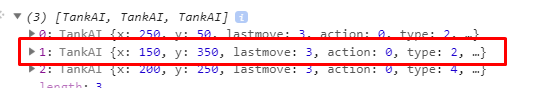
Bước 3:

Nếu bước 2 không đúng thì update hướng quay mặt và x,y tương ứng với lastmove  
 this.curent\_direction = 1;

This.y+=50; //y+-50; x+-50

3.4 Ví dụ minh hoạ

1. Thuật toán người đưa thư ( dành cho 2 wave đầu tiên )  
 1.1 Đối với wave 1:

  
  
Ego có các thuộc tính sau:

Ego.x = width/2=500  
Ego.y = height/2 = 300

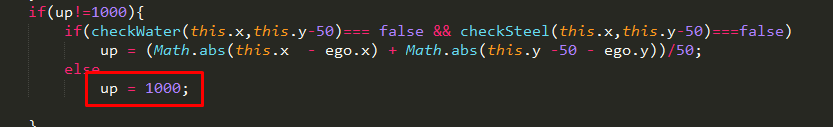


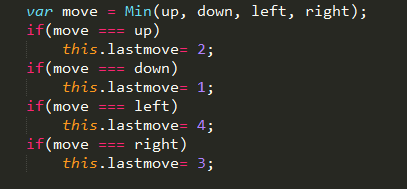



Áp dụng công thức trên để tính toán, ta có các chỉ số sau:  
   
 A picture containing text

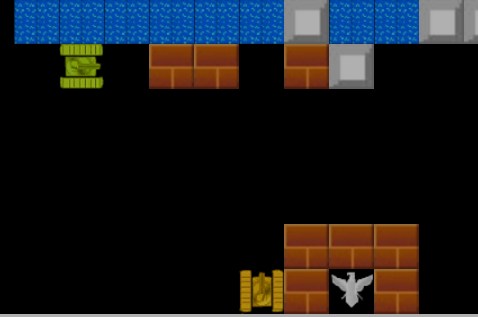
Description generated with high confidence

Giá trị 1000 do điều kiện if else sau:



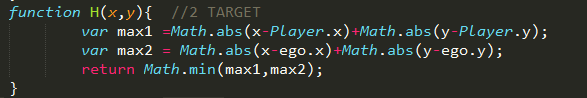
Sau đó hàm min(a,b,c,d) sẽ chọn giá trị nhỏ nhất trong các giá trị vừa tính để chọn bước đi tiếp theo:

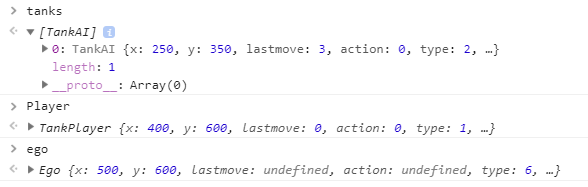
Trong trường hợp này sẽ là right = 7;

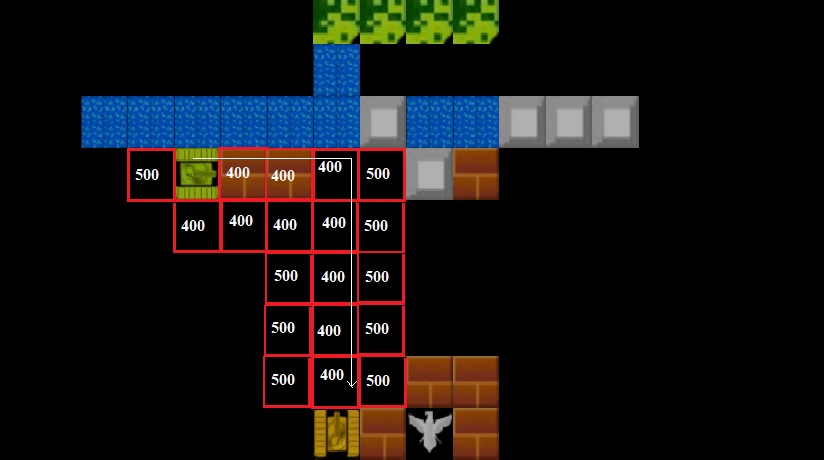


Next frame

1.2 Tương tự với tank AI turn 2 (đuổi theo Player)

2. Thuật toán A\* tìm đường đi:





Thứ tự push vào open lần lượt là trái, phải, trên, dưới. Nếu xảy ra giá trị trùng nhau thì chọn min đầu tiên, tức là theo thứ tự push. Tank Ai sẽ đi theo hướng mũi tên trên.



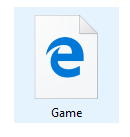
Với mỗi lần update vị trí mới của tank AI, hàm checkBlock sẽ được gọi để kiểm tra xem có phải bước đi tiếp theo trùng vào block không, nếu trùng thì tạm dừng bước đi, gọi hàm bắn, nên trong thuật toán A\* lẫn người đưa thư sẽ chỉ kiểm tra điều kiện về trùng River và Steel.

Chương 4. Ứng dụng

4.1. Giới thiệu chương trình ứng dụng

Tên chương trình: Game Battle City.

**Yêu cầu:** Cần cài đặt trình duyệt web trước khi chơi – bất kỳ trình duyệt web nào hỗ trợ JavaScript.

**Để vào chơi game:** vào thư mục BattleCity, tìm file tên Game, nhấp đúp chuột vào.

**Di chuyển:** gồm có 3 tank mau xanh lá sẽ tự di chuyển, do sinh viên lập trình, 1 tank màu vàng do người chơi điều khiển với các phím A – W – S – D, tương ứng với di chuyển sang trái – lên – xuống – phải. Không thể di chuyển xuyên qua steel, brick, water, nhưng có thể di chuyên xuyên qua forest.

**Chiến đấu:** Người chơi tấn công các AI để bảo vệ khung thành thông qua phím J (hoặc spacebar) – bắn ra đạn tiêu diệt AI. AI cũng có khả năng tương tự, nhưng được lập trình tự bắn.

**Bản đồ:** Được dựng sẵn gồm các vị trí của steel, brick, water, forest do sinh viên cài đặt. Vị trí đặc biệt trong map là eagle được bảo vệ bởi 1 lớp brick.

**Tank:** màu xanh lá là Tank AI, còn màu vàng là Tank do người chơi điều khiển.

**Mở rộng so với game thông thường:** cheat code làm cho Eagle bất tử - phím Enter, người chơi bất tử - phím U. Chọn lượt các phím 1, 2, 3 để chuyển ***lượt chơi*** tương ứng. (mỗi lượt chơi sẽ phát sinh số lượng tank (3,3,1) với độ khó tăng dần)

**Quyết định thắng/thua:**

* Player sẽ thắng khi: trên bản đồ không còn tank AI – dĩ nhiên là eagle chưa bị tiêu diệt.
* Player sẽ thua khi: bị tank AI tiêu diệt hoặc không bảo vệ được eagle – tức là eagle bị tank AI bắn hạ.

4.2 Cài đặt

4.2.1 Ngôn ngữ lập trình JavaScript.

Một vài nét về JavaScript:

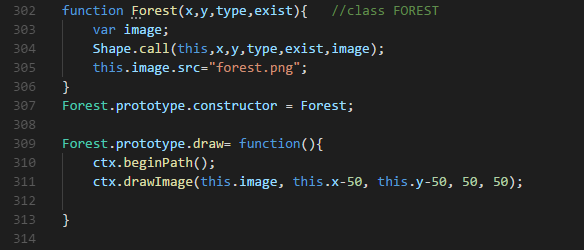
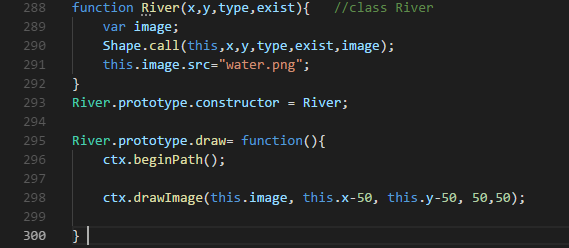
JavaScript, theo phiên bản hiện hành, là một [ngôn ngữ lập trình thông dịch](https://vi.wikipedia.org/w/index.php?title=Interpreted_language&action=edit&redlink=1) được phát triển từ các ý niệm [nguyên mẫu](https://vi.wikipedia.org/w/index.php?title=L%E1%BA%ADp_tr%C3%ACnh_d%E1%BB%B1a_tr%C3%AAn_nguy%C3%AAn_m%E1%BA%ABu&action=edit&redlink=1). Ngôn ngữ này được dùng rộng rãi cho các [trang web](https://vi.wikipedia.org/wiki/Website)(phía người dùng) cũng như phía máy chủ (với Nodejs). Nó vốn được phát triển bởi [Brendan Eich](https://vi.wikipedia.org/w/index.php?title=Brendan_Eich&action=edit&redlink=1) tại [Hãng truyền thông Netscape](https://vi.wikipedia.org/wiki/Netscape) với cái tên đầu tiên *Mocha*, rồi sau đó đổi tên thành *LiveScript*, và cuối cùng thành JavaScript. Giống [Java](https://vi.wikipedia.org/wiki/Java_(ng%C3%B4n_ng%E1%BB%AF_l%E1%BA%ADp_tr%C3%ACnh)), JavaScript có [cú pháp](https://vi.wikipedia.org/wiki/C%C3%BA_ph%C3%A1p_h%E1%BB%8Dc) tương tự [C](https://vi.wikipedia.org/wiki/C_(ng%C3%B4n_ng%E1%BB%AF_l%E1%BA%ADp_tr%C3%ACnh)), nhưng nó gần với [Self](https://vi.wikipedia.org/w/index.php?title=Self_(ng%C3%B4n_ng%E1%BB%AF_l%E1%BA%ADp_tr%C3%ACnh)&action=edit&redlink=1) hơn Java. .js là [phần mở rộng](https://vi.wikipedia.org/w/index.php?title=Ph%E1%BA%A7n_m%E1%BB%9F_r%E1%BB%99ng&action=edit&redlink=1) thường được dùng cho [tập tin](https://vi.wikipedia.org/wiki/T%E1%BA%ADp_tin) [mã nguồn](https://vi.wikipedia.org/wiki/M%C3%A3_ngu%E1%BB%93n) JavaScript.

Một vài điểm thú vị của JavaScript so với C/C++:

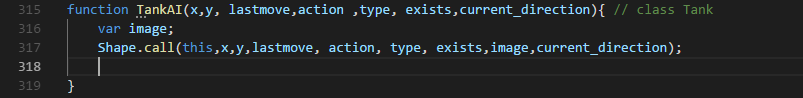
Khai báo biến có từ khoá var, let trước tên biến.

Toán tử quan hệ là các toán tử dùng để so sánh giữa hai toán hạng. Sau khi thực hiện một so sánh, chúng trả lại một giá trị true (đúng) hay false (sai). Các toán tử bao gồm: **==** (bằng nhau), **!=** (khác nhau), **===** (bằng nhau và cùng loại), **!==** (khác nhau và khác loại), **>** (lớn hơn), **<** (nhỏ hơn), **>=** (lớn hơn hoặc bằng), **<=** (nhỏ hơn hoặc bằng).

**Tiến hành cài đặt cụ thể các class, hàm:**

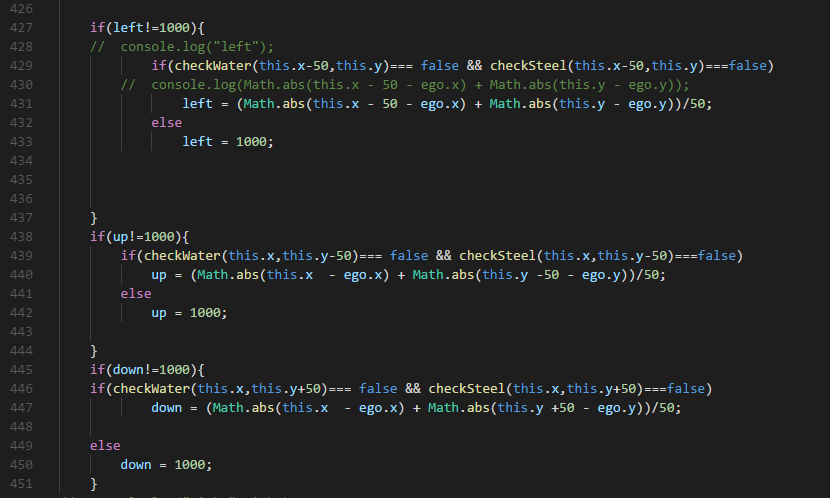
class **River**

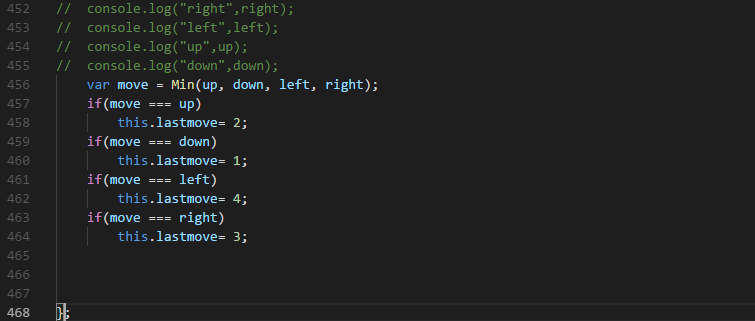
class **Forest**

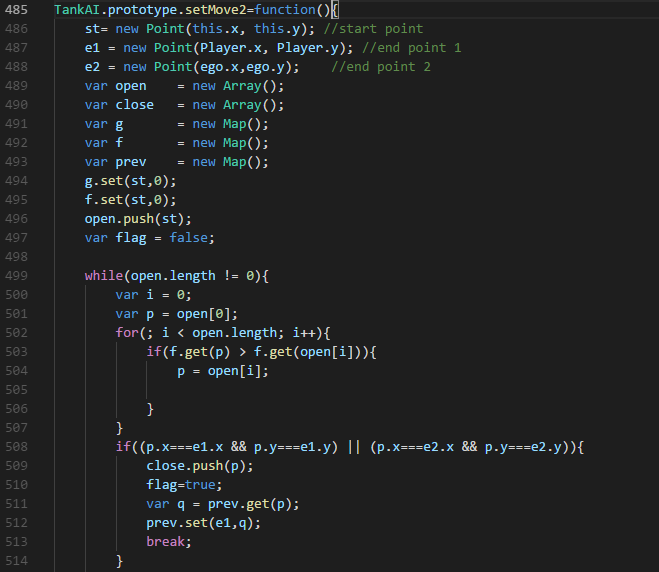
class **Tank**

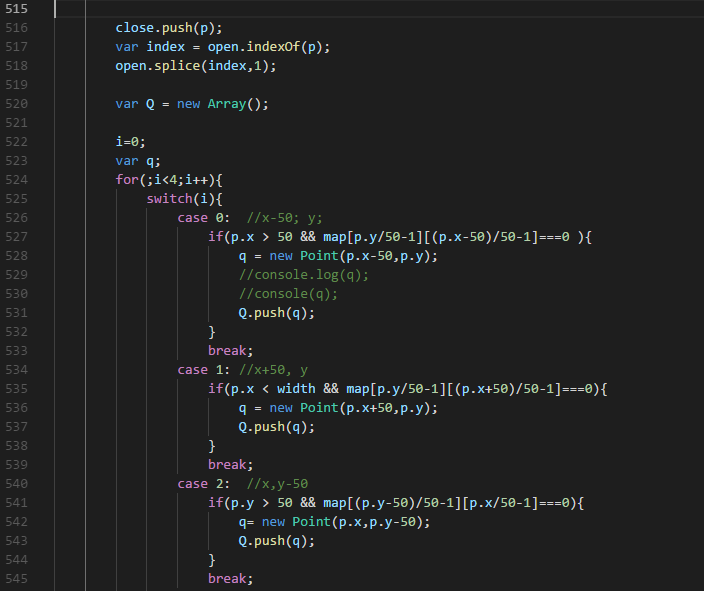


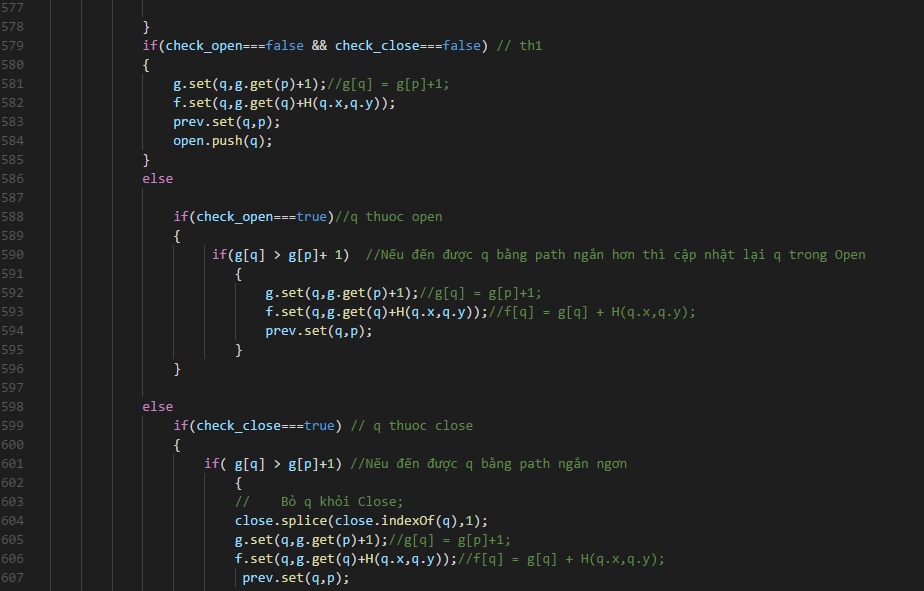
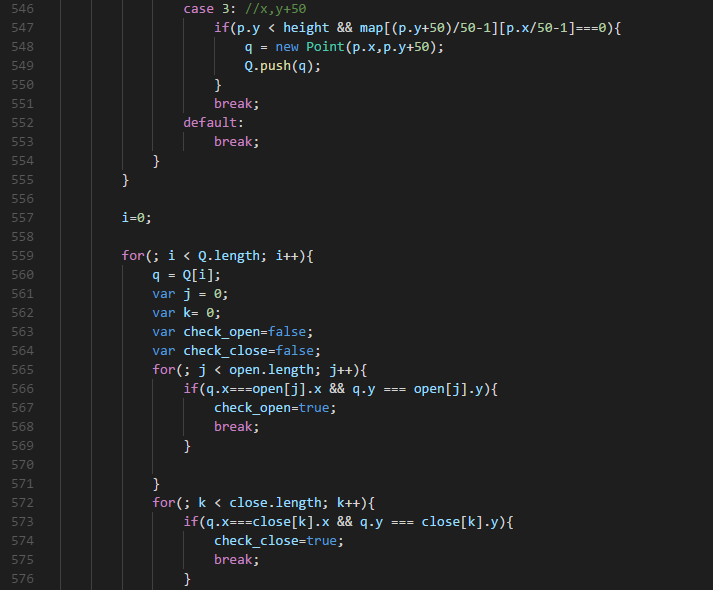
hàm setMove1

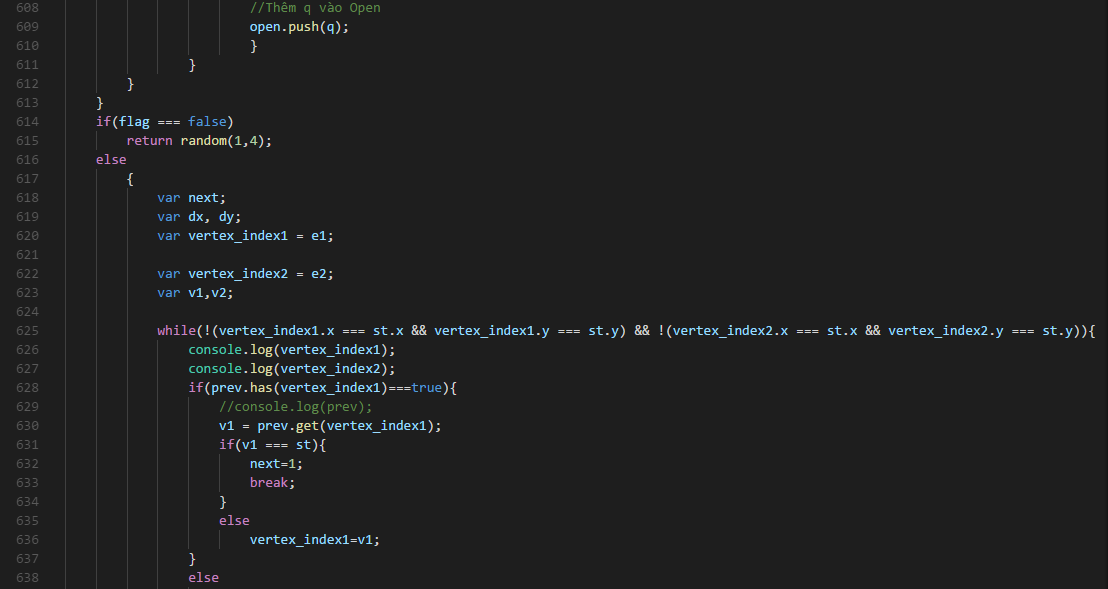




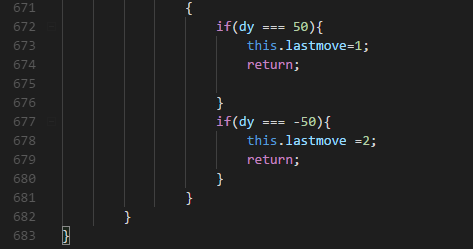
hàm **setMove2**



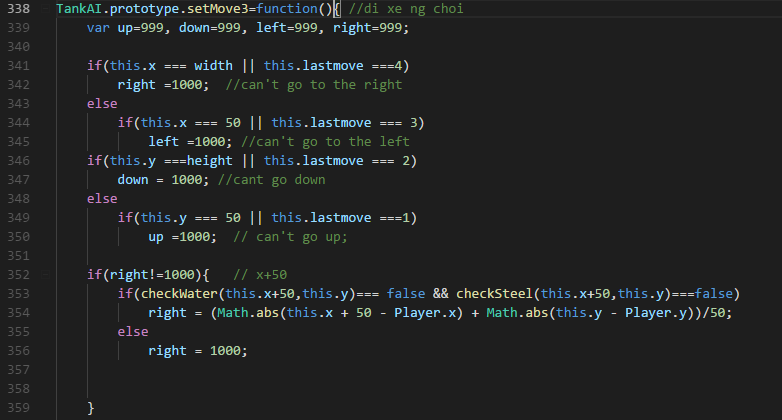


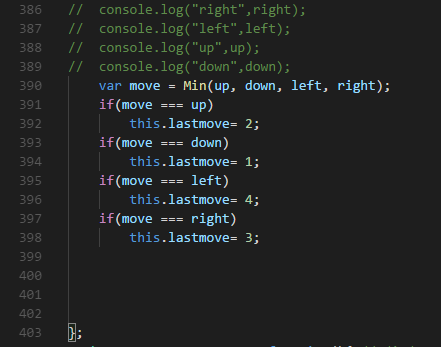


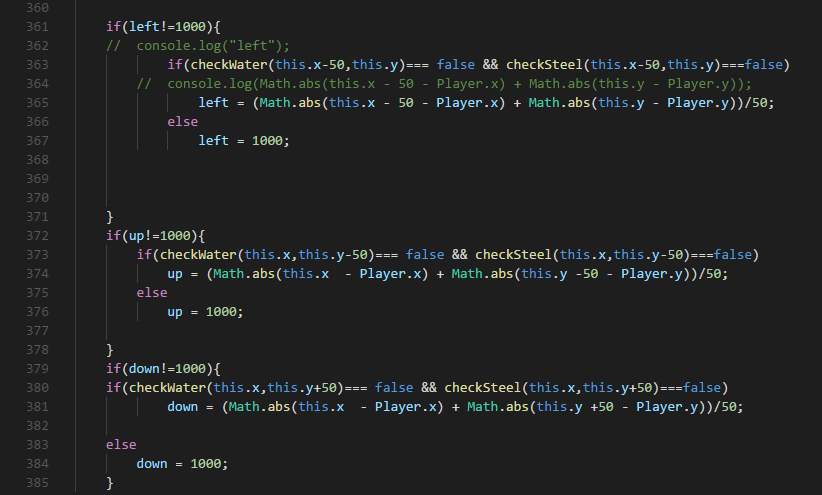




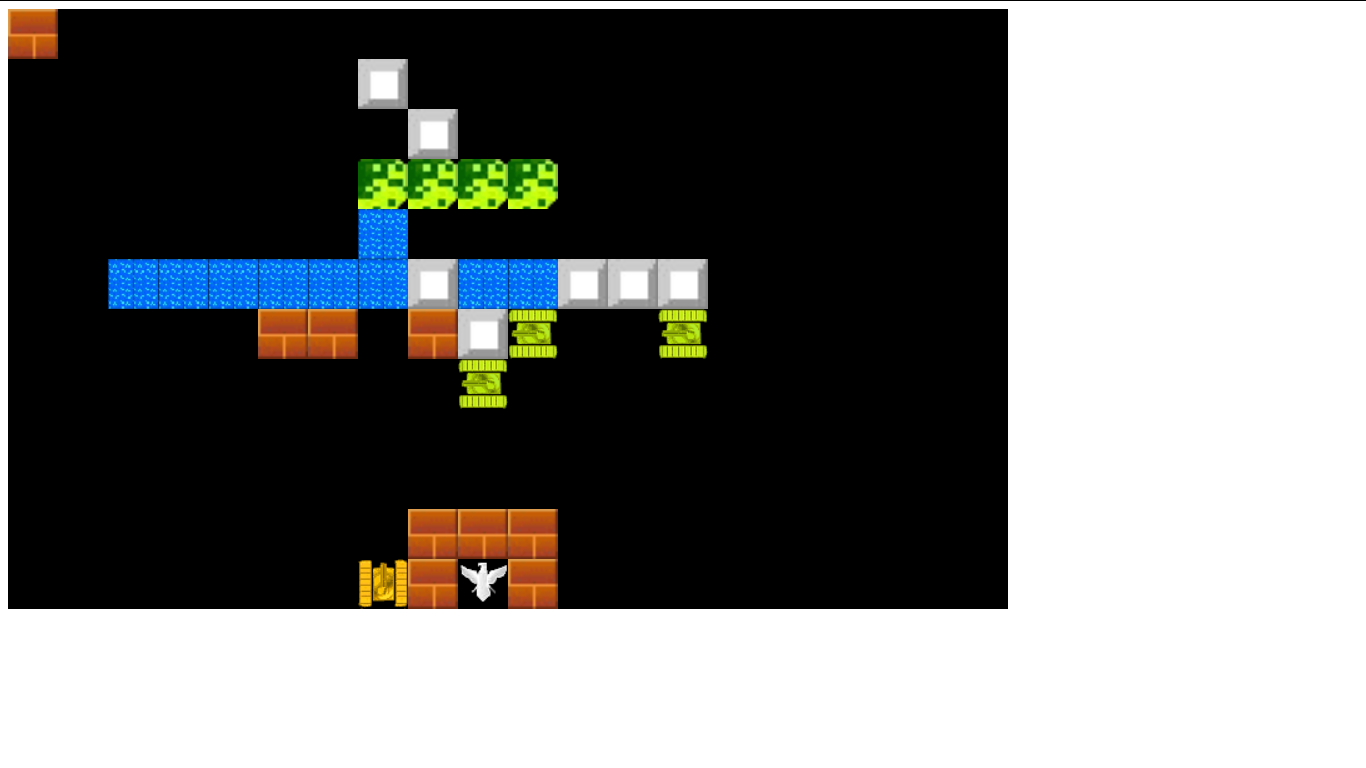
hàm **setMove3**

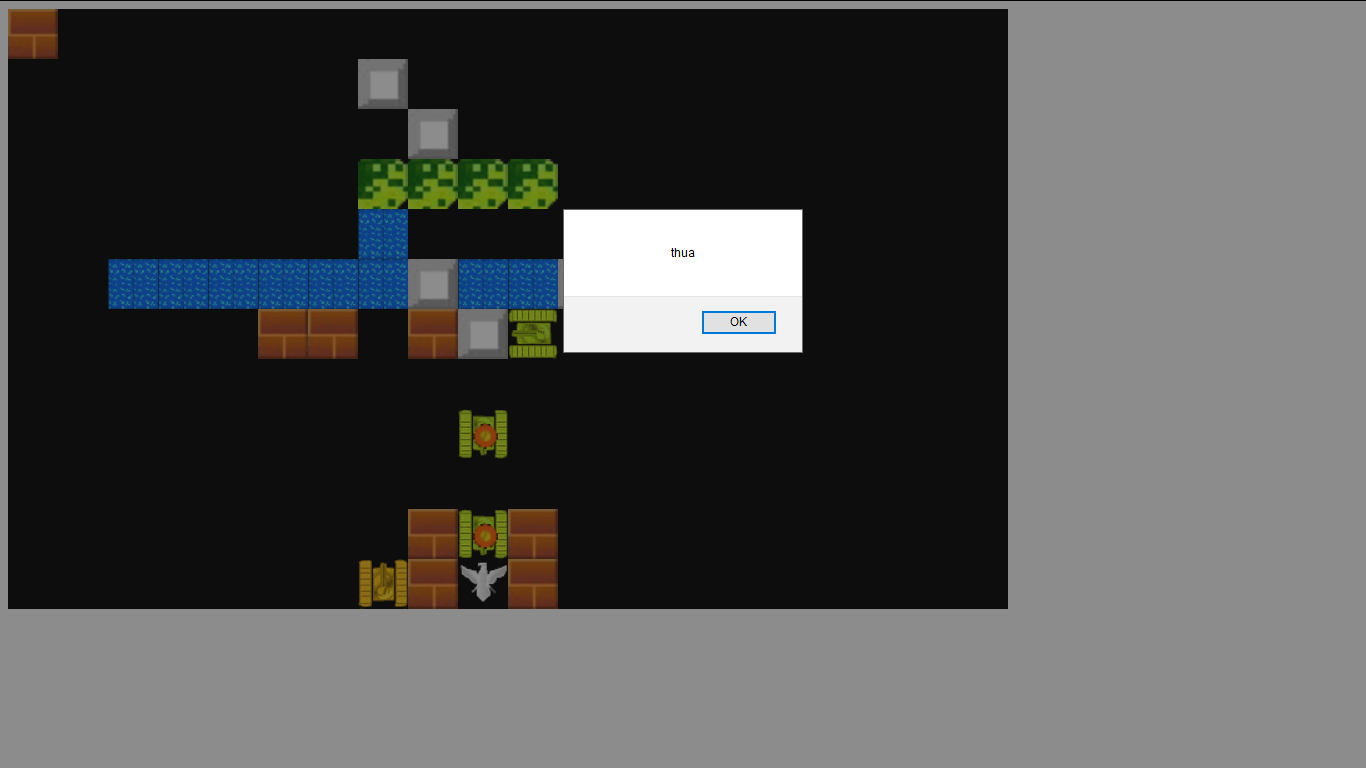


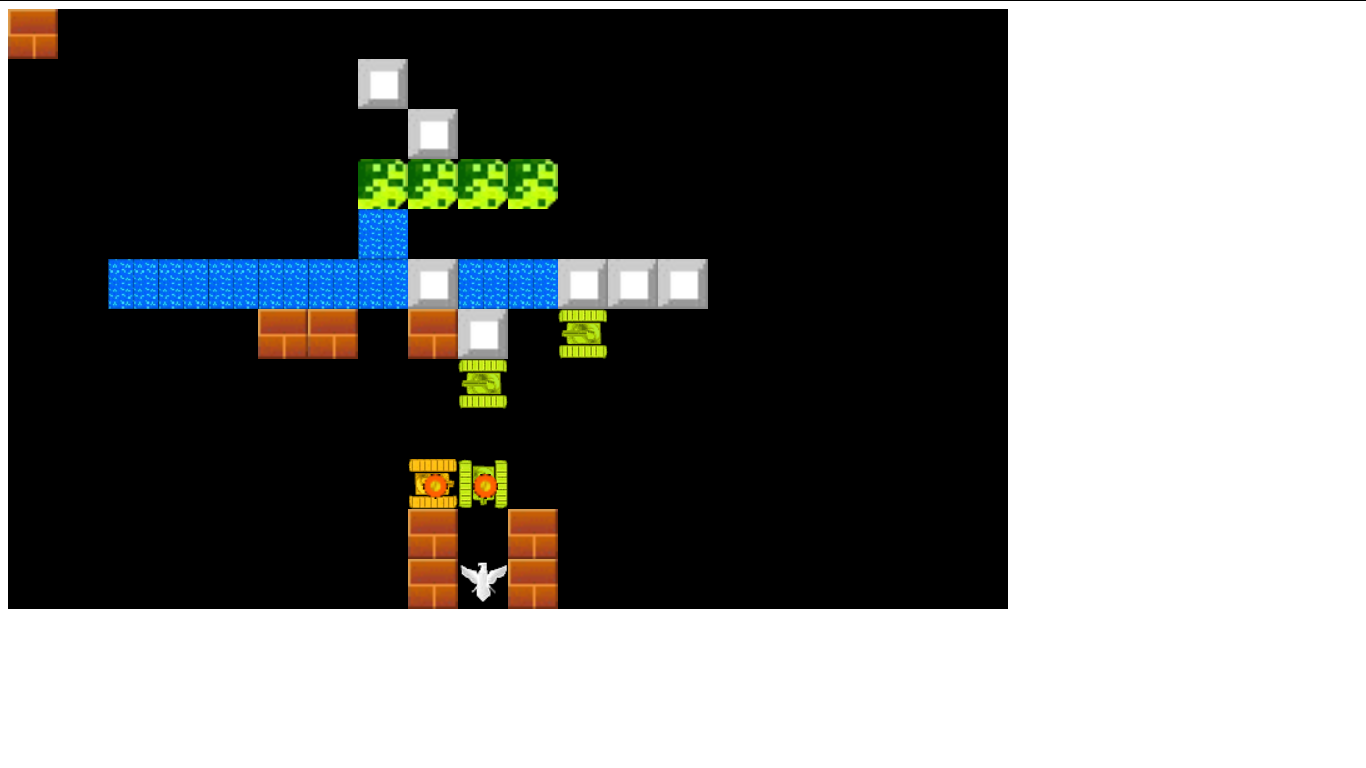




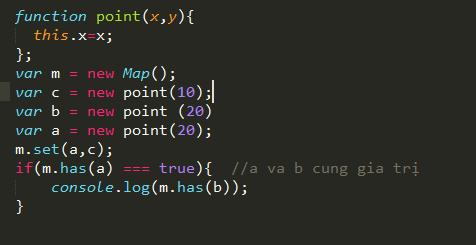
4.3 Kết quả chạy chương trình.

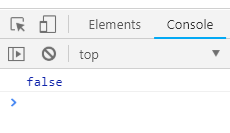
Chương trình chạy ổn định, bình thường.

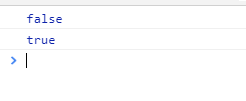


Khuyết điểm, map khá sơ sài, chưa được sinh động bắt mắt.

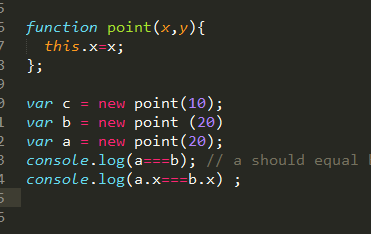
Ưu điểm: sinh viên vẫn có gắng hoàn thiện đồ án.

Khó khăn: JavaScript có một số phương thức kế thừa đặc biệt, dù cho các thuộc tính có bằng nhau nhưng chưa chắc đã bằng nhau. Gây khó khăn trong việc code A\*





Các hình ảnh dùng trong chương trình phải đi vay mượn, cắt ghép, chuyển thể từ gif sang jpg từ game thật tế



Chương 5. Kết luận

5.1. Kết quả đạt được

* Các sinh viên đều đóng góp để hoàn thành đồ án. Và quyển báo cáo.
* Hiểu được cách làm việc nhóm và hoàn thành một đồ án.

5.2 Hạn chế

* Chưa mạnh dạn trao đổi với giảng viên.
* Ngày bắt đầu làm đồ án khá trễ.
* Ý tưởng nghèo nàn.
* Khả năng lập trình mỗi sinh viên không tương đồng.

5.3 Hướng phát triển

* Cải thiện giao diện game, thêm âm thanh để thêm sinh động
* Tạo ra nhiều màn chơi
* Tích luỹ coin in game đổi xe tank cao cấp hơn.
* Nâng cấp đạn dược. Mỗi loại đạn có chức năng riêng.

Tài liệu tham khảo:

[1]Cơ sở trí tuệ nhân tạo – Lê Hoài Bắc, Tô Hoài Việt. NXB Khoa học Kỹ thuật – 2014.

[2] Slide bài giảng trí tuệ nhân tạo – UIT – Giảng viên: Huỳnh Thị Thanh Thương.

Hình ảnh - sơ đồ mạch điện hình bộ não, khoảng cách Euclid, khoảng cách Manhattan, logo UIT được lấy từ Internet.

Chương 6: Phân công công việc và đánh giá chất lượng làm việc

6.1 Phân công công việc:

1. Diệp Minh Tâm (16521057): code đồ họa, code xử lí vật lí, các hiệu ứng, thông báo thua, thắng, xử lí va chạm, đụng độ.

2. Trần Triều Tân (16521081): code thuật toán tank ngu (setmove 1,3), làm báo cáo đồ án.

3. Trần Quốc Phong (16520930): code thuật toán A\* lựa chọn 2 mục tiêu.

4. Nguyễn Huỳnh Phú (16520934): code thuật toán tank ngu (setmove 1,3).

6.2 Đánh giá chất lượng làm việc:

Các thành viên đều tham gia tích cực vào đồ án, tuy do khả năng code khác nhau nên sẽ có sự chênh lệnh trong tỉ lệ đóng góp.

|  |  |
| --- | --- |
| Diệp Minh Tâm | 100% |
| Trần Triều Tân | 100% |
| Trần Quốc Phong | 80% |
| Nguyễn Huỳnh Phú | 80% |

Chỉnh sửa:

Bổ sung:

-Báo cáo:

+ Cấu trúc dữ liệu và giải thuật, ví dụ, chạy tay thuật toán: mục 3.2, 3.3, 3.4 (trang 9-25)

+ Chương 4: bổ sung hướng dẫn chi tiết, hướng dẫn cheatCode (trang 26)

+ Bổ sung kết luận, hạn chế, ưu điểm, khó khăn (trang 38, 39, 40)

+ Hình các hàm quan trọng (trang 26 – 38)

-Chương trình demo:

+ Bổ sung cheatCode: nằm ở hàm TankPlayer.prototype.setControl.

+ Bổ sung hướng dẫn hiện trực tiếp ở phía dưới màn hình chơi trên file HTML.

+ Bổ sung chức năng random vị trí cho các tank khi xuất hiện.

+ Đã có thể chơi thắng máy (sử dụng cheatCode)

+ Nếu muốn thực hiện đi chéo thì chỉnh sửa hàm setControl bằng việc cho thêm 4 đoạn if else và chỉnh sửa hàm update với việc cho thêm 4 đoạn if else để update giá trị x,y

+ Update file readme.

+ Bổ sung, sắp xếp lại video.