Bachelor thesis in the Computer Science and Media degree programme

Tactical Game AI with shared Knowledge based on Influence Maps

submitted by

Fred Newton, Akdogan

at the Stuttgart Media University on August 30, 2021

to obtain the academic degree of Bachelor of Science (B.Sc)

First examiner: Prof. Dr. Stefan Radicke

Second examiner: Prof. Dr. Joachim Charzinski

Zusammenfassung

Das Ziel dieser Forschung ist es zu bestimmen ob eine Taktische Game AI mit geteiltem wissen bassierend auf Einflusskarten agiert gegen eine AI deren Agenten alle eine eigene Einflusskarte generieren. Dazu wird die folgende Forschungsfrage gestellt: "Wie groß it der unterschied zwischen der AI deren agenten ihre Einflusskarte teilen oder wenn jeder Agent seine eigene Einflusskarte verwendet?".

Um die Forschungsfrage zu beantworten, wurde auf zwei verschiedenen Karten, in jeweils 100 Iterationen in jeder Konstellation die AI gegen einander anzutreten.

Der Versuch hat gezeigt, dass man mit nur einer Einflusskarte kein genaues Ergebnis hervorrufen kann. Zu 50% gewann die AI mit den Agenten die ihre Einflusskarte teilt und zu 50% die andere.

Es wird viel mehr benötigt wie verschiedene Strategien, Balancing sowie mehr Möglichkeiten zu geben um auf verschiedene Situationen zu reagieren.

Abstract

The aim of this research is to determine whether a Tactical Game AI with shared knowledge based on influence cards will act against an AI whose agents all generate their own influence card. The research question is: "How big is the difference between the AI whose agents share their influence map or when each agent uses its own influence map?

To answer the research question, the AIs were pitted against each other on two different maps, in 100 iterations in each constellation.

The experiment showed that you cannot produce an accurate result with just one influence map. The AI with the agents sharing its influence card won 50% of the time and the other 50% of the time.

Much more is needed, such as different strategies, balancing and more possibilities to react to different situations.

Honorary Declaration

Hiermit versichere ich, Fred Newton Akdogan, ehrenwörtlich, dass ich die vorliegende Bachelorarbeit (bzw. Masterarbeit) mit dem Titel: "Tactical Game AI with Shared Knowledge based on Influence Maps" selbstständig und ohne fremde Hilfe verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen der Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken entnommen wurden, sind in jedem Fall unter Angabe der Quelle kenntlich gemacht. Die Arbeit ist noch nicht veröffentlicht oder in anderer Form als Prüfungsleistung vorgelegt worden. Ich habe die Bedeutung der ehrenwörtlichen Versicherung und die prüfungsrechtlichen Folgen (§26 Abs. 2 Bachelor-SPO (6 Semester), § 24 Abs. 2 Bachelor-SPO (7 Semester), § 23 Abs. 2 Master-SPO (3 Semester) bzw. § 19 Abs. 2 Master-SPO (4 Semester und berufsbegleitend) der HdM) einer unrichtigen oder unvollständigen ehrenwörtlichen Versicherung zur Kenntnis genommen.

Signature	Date

Contents

Chapter 1

Introduction

influence map (IM) is popular in games artificial intelligence (AI) and is a very powerful tool and is no longer a rarity in the industry.

1.1 Motivation

The structure of an IM and the possible uses as explained in the pathfinding in Chapter 3.2.1.3 as well as the positioning of the agents have already been explored [Adaixo, 2014]. High-level AIs that use them for strategy finding or low-level AIs for micromanagement have also been researched, as can be read in Chapter 3.1 [Mellington, 2020] [Mark, 2013] [Champandard, 2021]. But how an AI behaves when its agents share an IM or each agent builds its own IM is still largely unexplored.

1.2 Scientific question

In this research, the question of how an AI with a shared IM performs against an AI which builds an IM for each agent is explored.

1.3 Aim of the research work

The aim of this research is to find out if it makes a big difference if the AI shares an IM or not, and thus to find out possible suggestions for improvement and to find possible problems.

1.4 The research method

Quantitative rounds are played in different constellations of AI. These are the same in all respects except how the IM is used. This is also performed on different maps in order to get a quantitative result based on the changed environment.

1.5 Structure of the Thesis

In the course of this scientific work, it will be examined how different approaches to IM influence behaviour. Theoretical terms such as "Tactical Game AI" and "Influence Map" are explained first. After that, the use of IM will be discussed. Furthermore, the part on "Dealing with unknowns" from chapter 3.2.4 is added, as this is an important point for the memory of the AI. Then it is about how an IM is constructed and how a blur is used. Then the structure of the expirement is explained.

Finally, the results are summarised and a final conclusion is drawn, including suggestions for improvement.

Chapter 2

Related work

In Rabin [2015] article discussed how IM works in general. As well as the important part of giving an agent or squad a memory of the current influence range on the map.

The article [Champandard, 2021] discusses how IM is implemented in the industry. Just like how an IM behaves on maps with narrow aisles and few open areas.

The possible modular structure was explained by Dave mark in the article [Mark, 2013] as well as a possible implementation. It was also explained where the possible power lies in IM. This was also discussed in Chapter 3.2.1.

Chapter 3

Theoretical background

3.1 Tactical Game AI

In Real Time Strategy (RTS) games, for example, there are many tasks for the AI on a tactical and strategic level. Tasks are completed for one or more agents to simulate tactical behaviour. With such an interface, AI planners can be generated and high-level battle plans can be created. These will then be executed by the tactical AI in the micromanagement system. [Michael van Lent, 2002]

3.2 Influence Map

All information is taken from the book AI for Games unless otherwise cited in this chapter [Mellington, 2020].

With an IM, the current distribution of a team's military strength can be mapped to any position on the map. In this way, different additive events can also be mapped. Many factors can affect military influence, such as the proximity of a unit, the proximity of a base, the length of time a unit has been last seen, the terrain, the current weather, the strength of a unit. Many factors have only a small influence. Based on how the abstract image is drawn across the map or level of influence, more marginal information can be communicated to the AI and tactical decisions can be made.

3.2.1 Usage

With IM, one can arrange a collection and storage of map data so in usable forms. This information can then be further processed into three general categories [Mark, 2013]:

1. Gathering information about our location and the area around us [Mark, 2013]
2. Targeting locations [Mark, 2013]
3. Movement destinations [Mark, 2013]

3.2.1.1 Information

The simplest form of information that the IM can represent is what the status of a cell is at a particular point. The status can be, for example, whether the cell is accessible or whether there is a collider that cannot be crossed [Mark, 2013]. This can then be extended with functions that return the cell with the highest or lowest influence value, as well as the status of the cells around you from the cell [Mark, 2013]. For example, to find out where the current danger from the enemies is from the position on which the agent is standing. This is then calculated with the influence value of the enemy influence and tells how high the threat is [Mark, 2013].

3.2.1.2 Targeting

One of the other uses of IM is for deciding which target to attack. For example, caching the cell in IM with the lowest influence value suggests the location with the most enemy agents. Therefore, depending on how aggressive or passive the agents are, this location can be avoided or moved into focus. [Mark, 2013]

3.2.1.3 Movement / Positioning

IM can have a great influence on pathfinding and pathfollowing. The same applies to tactical shooters and the positioning of agents, for which waypoints can be added as in Chapter 3.4. With this, a direction vector can be created with the high point of the highest concentration of influence values of the opponents in order to get the direction of how an agent can flee or retreat.

Pathfinding based on IM opens new ways to deal with dynamic environments [Adaixo, 2014]. The path generation can react to game-typical properties (for example: line of sight, sound, faction location, etc.) [Adaixo, 2014]. This allows agents to better navigate through the environment. For strategic games, IM displays general objectives for the pathfinder so that they can reach their goal as quickly as possible. For low-level tactical decision-making, in-game events are added to the IM (for example: a grenade exploding, shots flying by, sound propagation, etc.) [Adaixo, 2014]. Therefore, paths can always be rescheduled and recalculated based on these events [Adaixo, 2014].

3.2.2 Simple Influence

The influence of a unit in an area consists of how much its influence is weighted. Assuming it is a Real Time Strategy game and there is a foot soldier unit and a tank. Normally, a tank has more lives, more damage and a longer range than a simple foot soldier. This means that a larger Influence value is taken and injected into the IM at the unit's position. If you take the strength of a unit, it decreases with increasing distance. So the further away you are from the unit, the less influence it has. A linear drop-off model can be used for this. A doubling of the distance results in a halving impact:

$$I_d = \frac{I_0}{1+d} \tag{3.1}$$

I_d is the influence at a given distance. d is the distance from the unit to the point and I_0 is the influence at the distance value 0 to the unit. It would also be possible to use a more sloping initial drop off, with a greater range of influence:

$$I_d = \frac{I_0}{\sqrt{1+d}} \tag{3.2}$$

It is also possible to use an equation that first flattens out and then falls sharply:

$$I_d = \frac{I_0}{(1+d)^2} \tag{3.3}$$

3.2.3 Calculating the Influence

For the IM, a large calculation is needed for each unit on the map for each possible position. The execution time would be $O(nm)$ and the memory is $O(m)$. m represent the number of possible positions in the game and n the number of units. With a linear drop-off curve, the influence is covered with a threshold value. In this way, small values are not unnecessarily stacked on top of each other in a larger range:

$$r = \frac{I_0}{I_t - 1} \tag{3.4}$$

Where I_t is the threshold value for the influence. Thus, the influence of each unit is only applied to the places that are within the given radius. This limits the calculation time to $O(nr)$ for the time and to $O(m)$ for the memory. r is the number of locations that are within the given radius.

3.2.4 Dealing with unknowns

Here, only the influence of units that can be seen in their radius is calculated for the unit. Thus, an aspect called fog-of-war (FOW) is built in. This is important for investigating whether the shared knowledge (SK) of units makes a difference. In this way, units also have a maximum distance they can see and can only build a personal IM based on the friendly or enemy units they can see and incorporate this into their decisions. This can lead to problems for the AI decision making, because it does not have the same memory as humans have and cannot map the context. Therefore, it is important to give the AI some kind of memory. This can be mapped well with IM so that the AI can manage well in the FOW. It will also be very interesting to see if it makes a significant difference between the Shared and Unshared Knowledge Teams.

3.2.5 Influence Map Setup

All information in this section is quoted from the article [Champandard, 2021]. Otherwise they are cited as such.

3.2.5.1 2D Grid

For the IM, a 2-dimensional grid is stretched over the map and divided into a grid system. Then, all areas that cannot be walked on, such as walls or similar, are excluded from the calculation and ignored. This enables the unknown and the FOW. Because with this, the influence does not propar through obstacles but around them. After that, the cells that have the shortest distance to each agent are injected with their influence in the 2 dimensional grid. This means that these cells are always set to the influence value of the unit regardless of anything else.

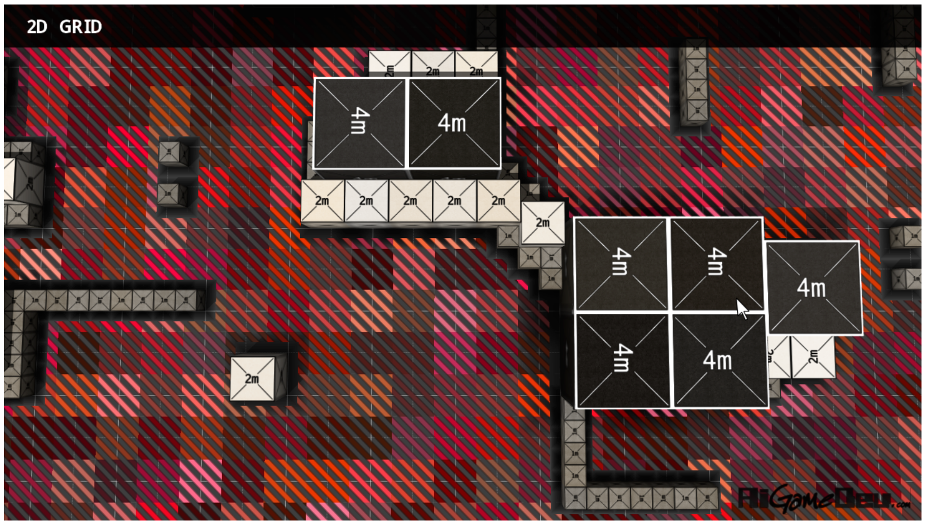

Figure 3.1: Fine Grid IM. The IM of this thesis was also built in the same style. Thus the cells are small enough not to protrude over walls and objects. This is good for maps with few open spaces and narrow passages. [Champandard, 2021] [Adaixo, 2014]

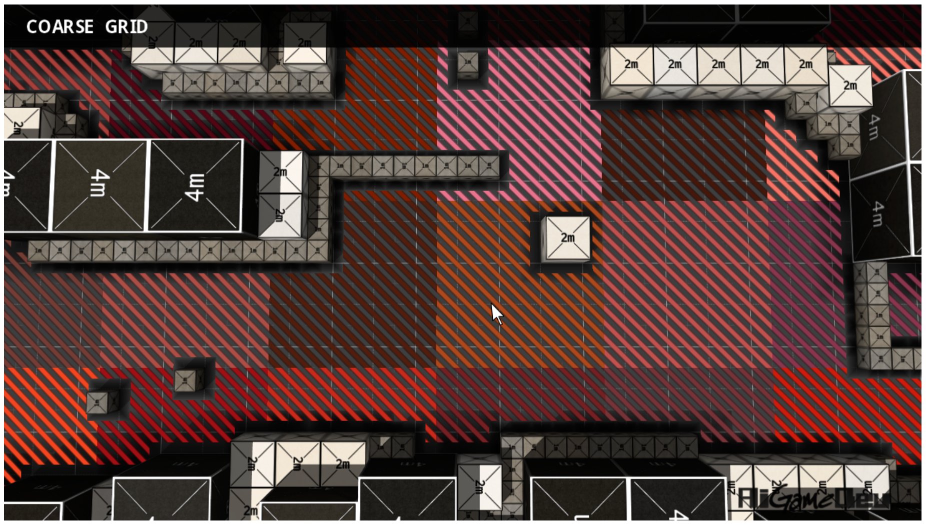

Figure 3.2: Coarse grid IM. With this map, large cells are placed on the map without taking into account whether the cell goes over the wall or not. This is good if you have open areas and few narrow passages. [Champandard, 2021] [Adaixo, 2014]

3.2.5.2 Area Graph

Area and waypoint graphs solve the limitations of 2D grids like in chapter 3.2.5.1. For example the problem of the big one that it consumes a lot of memory. Area graphs can also be used in 3D environments, for example spread over several floors. Although they bring benefits over grids, they may lack on precision where detail is needed for the decision making. [Champandard, 2021] [Adaixo, 2014]

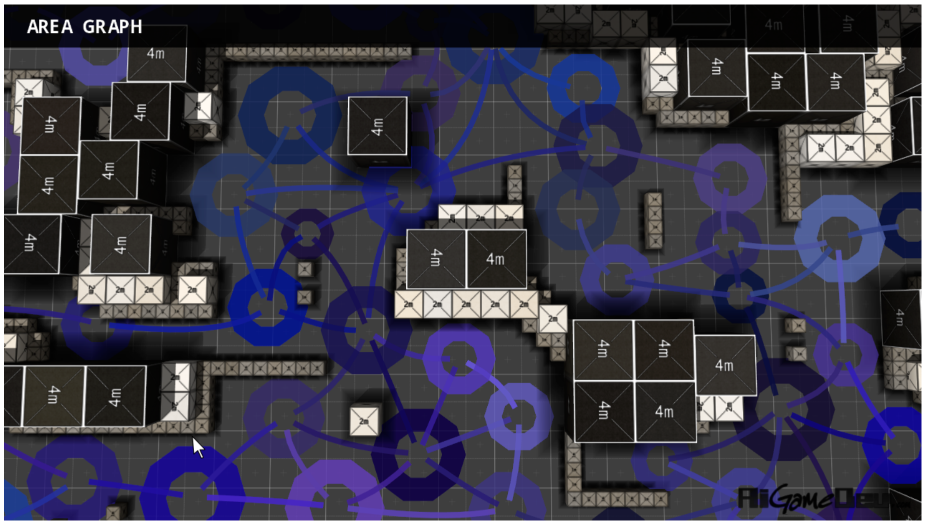

Figure 3.3: Area graph influence map. [Champandard, 2021] [Adaixo, 2014]

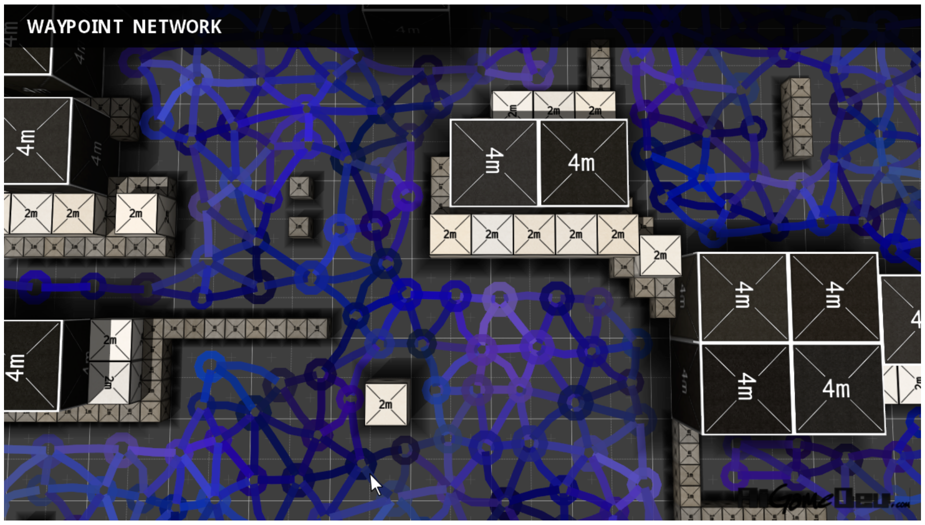

Figure 3.4: Waypoint network influence map. [Champandard, 2021] [Adaixo, 2014]

3.2.5.3 Properties

After setting the values of each agent in the IM, a blur algorithm is applied as explained in the chapter 3.3. The choice of which blur method is best suited for each game depends heavily on the game designer. For example, depending on whether there are many walls on the level or more open vials, it can be decided whether to use one of the distance-based blur algorithms described in Chapter 3.2.2 or a blur algorithm from Chapter 3.3. The important thing is to only apply this to cells that are accessible. Thus, the influence must propargize around walls and no distorted image is transmitted. The value from the blur algorithm is then multiplied by the decay to implement a decay of the influence on the range.

$$I_{xy} = b_{xy} * D \tag{3.5}$$

I_{xy} is the influence at the point x and y in the grid. This is equal to the blurred value b_{xy} at the point x and y from the algorithm multiplied by the decay value D.

- **Momentum** With momentum the influence is linearly interpolated from the cell. In this way the memory of the IM is suggested.

$$I_{xy} = I_{xy} + ((I_h - I_{xy}) * m) \tag{3.6}$$

I_h stands for the highest or lowest influence value that exists on the IM. If I_{xy} is greater than or equal to 0, the highest value is taken and if I_{xy} is less than 0, the lowest value is taken. m is the value for the momentum.

- **Decay** Momentum shows the strength of the scalar influence field. For larger cards, you tend to go for a lower decay value so that the influence can spread much further [Adaixo, 2014]. Decay is for the decay of the influence value within an IM so a kind of fading memory is built up and the influence continues to decrease depending on how far it is from its point of origin.
- **Update Frequency** The Update Frequency is based on how many resources should be used to update the map. For example, for high-level stategic AI, an update rate that takes more than one second (1Hz) is more likely to be chosen. For tactical AI that is more micromanagement oriented and needs to make decisions quickly, an update rate faster than one second is chosen (for example 5Hz). This allows the AI to react better to sudden changes. [Champandard, 2021] [Adaixo, 2014]

3.3 Blur

For the calculation of the influence on maps with narrow corridors and small areas a blur algorithm is used [Champandard, 2021].
This suggests that the algorithm is not affected by obstacles. Thus the influence flows around the corners. In this work, a boxblur algorithm was applied. But it would work just as well with a Gaussian blur. This is open to the individual preferences of how the influence should spread. For this work, the boxblur algorithm was chosen because it provides a more uniform distribution in all directions of influence.

0.000	0.000	0.000	0.000	0.000
0.000	**1.000**	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000
0.000	0.000	**-1.000**	0.000	0.000

Table 3.1: Boxblur influence example grid - Iteration 0. The inflow of two units was injected into a completely new IM. The cell with the value 1.000 is from the own team and the cell with the value -1.000 from the opposing team.

The Box Blur is a spatial domain linear filter. It takes a pixel (or in our case a cell) from the grid and takes itself and its surrounding pixels and calculates the average as the new value. A 3 by 3 box blur (radius 1) can also be described as a matrix:

$$K = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{3.7}$$

K is the average value of pixel and its surrounding values.[Handwiki, 2021]

0.250	0.200	0.200	0.030	0.008
0.240	**1.000**	0.160	0.006	0.007
0.210	0.180	0.150	0.040	0.002
0.060	-0.040	-0.070	-0.100	-0.010
0.005	-0.170	**-1.000**	-0.200	-0.078

Table 3.2: Boxblur influence example grid - Iteration 1. After the first iteration, from top left to bottom right, the cell was placed the 3 by 3 matrix. So that the center of the matrix lies on the cell and then calculated the mean value.

0.420	0.370	0.290	0.080	0.025
0.400	**1.000**	0.250	0.009	0.027
0.300	0.250	0.140	0.030	-0.007
0.068	-0.057	-0.130	-0.15	-0.069
-0.039	-0.221	**-1.000**	-0.271	-0.142

Table 3.3: Boxblur influence example grid - Iteration 2

Through this box blur, the influence is gradually expanded with 1.000 and -1.000 and then stagnates after a certain iteration. The Celle with 1.000 means that there is an allied unit that places its influence there and -1.000 means that there is an enemy unit.

3.4 Waypoints

Waypoints are positions distributed around the game world. With waypoints, the AI can use this for its pathfinder in order to progress in the game world. Tactical waypoints require more data describing these points in order to make a correct decision about which waypoint to use [Mellington, 2020].

Chapter 4

Experimental setup

The expirement consists of a map on which two teams compete against each other. This will be similar to the game mode ”Conquest” from Battlefield 1 [EA, 2016]. In Conquest, there are a certain number of places that a team tries to capture. At the beginning, each place is still uncaptured. When a team has taken a place, it always gets points added to their points account at certain time intervals. The team that has reached a certain number of points first after a certain time has won. One side will consist of a squad of five agents, the other side of five squads of one agent each. A squad always knows where its agents are and whether a unit sees a friendly or enemy agent and can thus build up an IM. When a team has won, information about the match is saved for later evaluation and the next match starts. This allows you to run this several thousand times so that the result is not falsified.

4.1 Game mode adjustments

Since in the game mode Conquest of Battlefield [EA, 2016] the current state of the capturable locations is always queried with a certain tick rate, this can lead to inaccurate values. Therefore the time a place is in possession of a team is counted to avoid the sampling problem.

4.2 Assumption

The side with the squad and its five agents has a slightly higher win rate than the side with the 5 squads with one agent each. This is because it has more information to decide, for example, which of the points to attack first or which point may have no enemy units.

4.3 Rules of the game

The goal of a match is to have captured capturable locations for the longest time. There are 3 capturable locations distributed for the agents. A location can be captured if there is no agent from the opposing team with an own agent at the location. In this case, the location's affiliation changes immediately. After the location is captured, the time is accumulated on the team's points account. The match ends after a total duration of 300 seconds. During the match, the agents shoot at the opposing team then. Each agent has 100 health points and each shot that hits takes away 50 health points. If an agent dies, it will be instantiated again after 5 seconds. On a random location on the map that does not count as a capturable point but more about that in chapter 4.3.2.

4.3.1 Playing field construction

Two different types of maps are used. An open map where an agent can more easily discover own team members or enemy units and a closed map with many corridors to not promote the unknown as explained in chapter 3.2.4. The unknown is the FOW that the agent cannot see on the map.

4.3.1.1 Open map structure

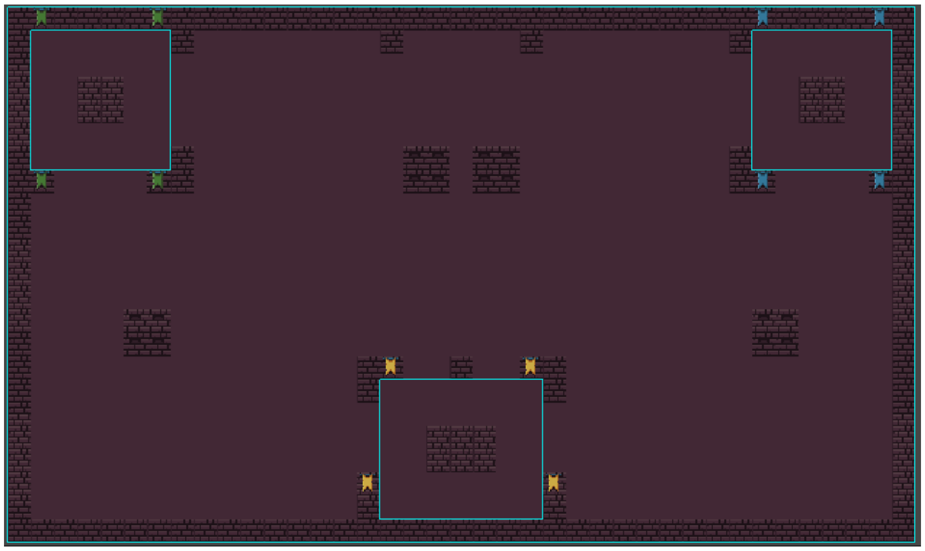

Figure 4.1: An open map with three capturable points marked with a cyan rectangle.

4.3.1.2 Closed map Structure

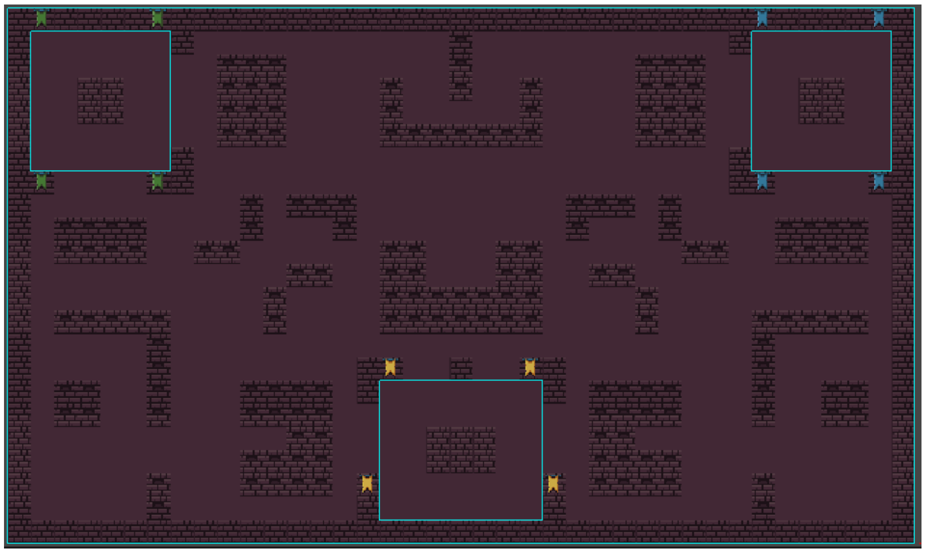

Figure 4.2: A closed map with three capturable points marked with a cyan rectangle. Here are many angles and small corridors distributed on the map and it was careful not to have too many open spaces so that the influence must propagate more around the corners and that the agents are not very line of sight to own team members or opponents.

4.3.2 Randomness factor

So that not every match runs the same, a random factor had to be added. Therefore the instantiation was chosen where an agent is placed when he died or when the match started. An agent then gets a random ball selected that is not in a capturable location. The randomization is done by the Unity3D engine random range function [Unity Random.Range, 2021]. If you always play a round with the same spawn locations you will always get a deterministic result and the agents will make the same decisions.

Chapter 5

Results

The shared team means that a squad of 5 agents share an IM. For the Unshared Team, it is 5 squads with one agent each. So each squad has its own IM and does not know what the IM of the other squad looks like from the team.

5.1 Open map

5.1.1 Shared vs Unshared

In the run of Shared vs Unshared on the open map as seen here in Figure 4.1, the Shared team won 53.0% of 100 runs. The team with the Unshared IM won 47.0%. The team with the shared influence card held the three points for an average of 414.4309201049805 seconds. The team with the unshared influence card held the three points for an average of 412.08373779296875 seconds.

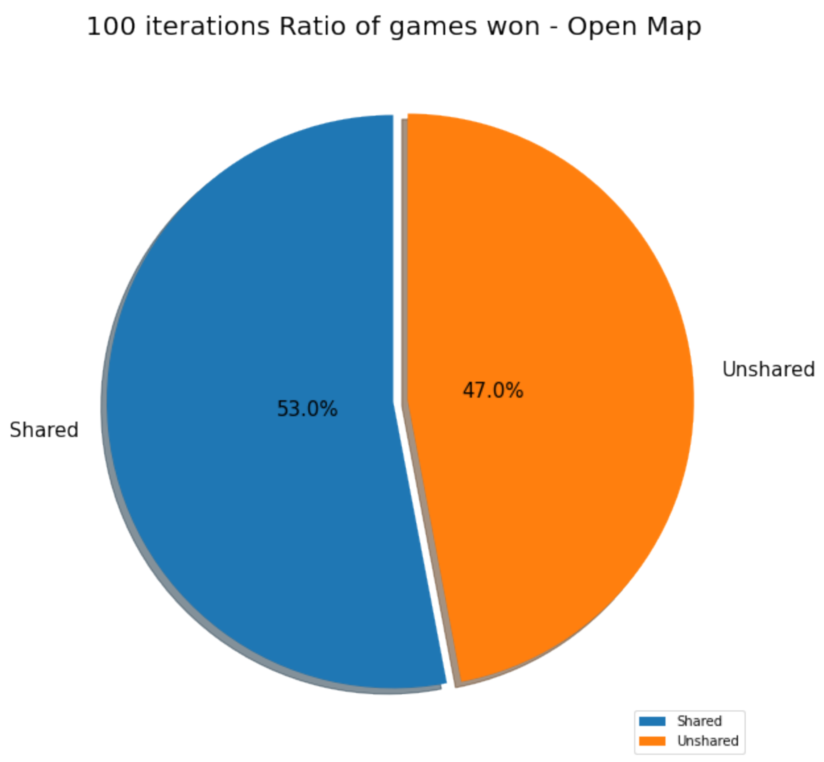

Figure 5.1: Shared vs Unshared - Open Map

5.1.2 Unshared vs Shared

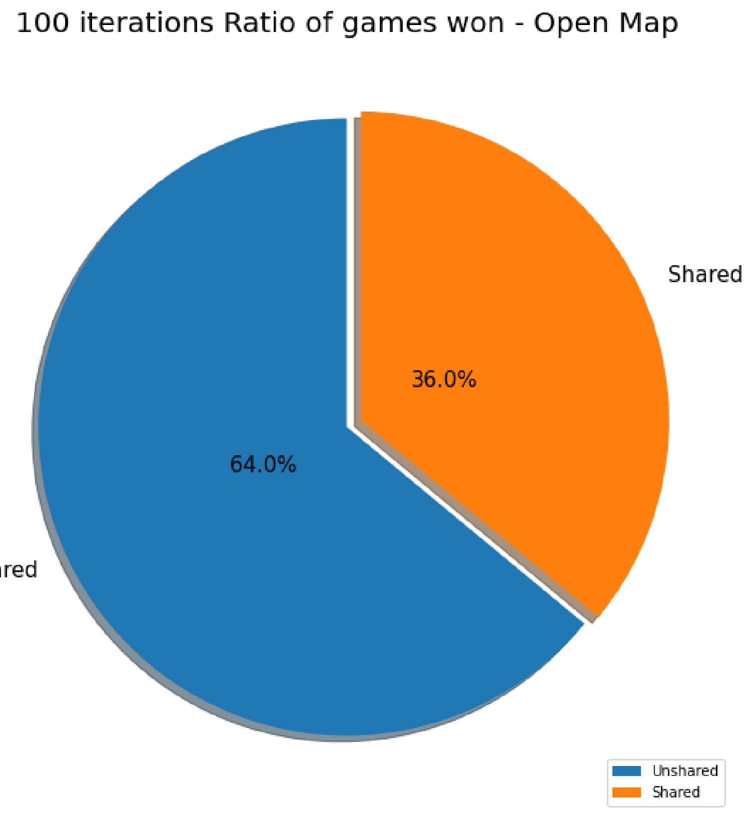

Figure 5.2: Unshared vs Shared - Open Map

In the run of Unshared vs Shared on the open map as seen here in Figure 4.1, the Shared team won 36.0% of 100 runs. The team with the Unshared IM won 64.0%. Shared and the Unshared team switched sides to see if it made a difference. The team with the shared influence card held the three points for an average of 450.8907879638672 seconds. The team with the unshared influence card held the three points for an average of 378.2070457458496 seconds.

5.1.3 Shared vs Shared

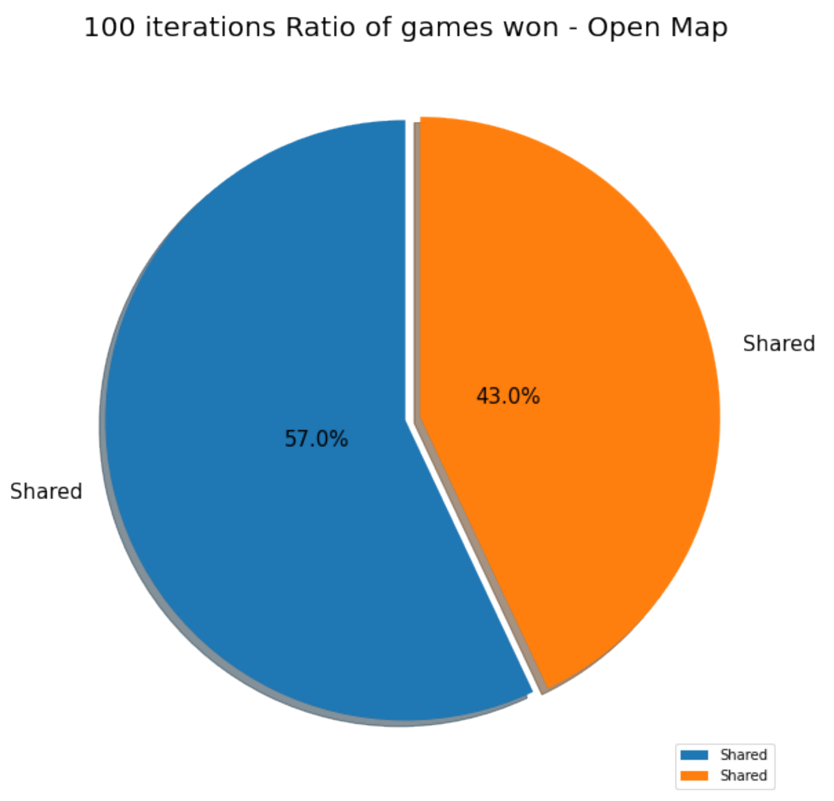

Figure 5.3: Shared vs Shared - Open Map

In the run of Shared vs Shared on the open map as seen here in Figure 4.1, the blue Shared team won 57.0% of 100 runs. The orange team with the Shared IM won 43.0%.

5.1.4 Unshared vs Unshared

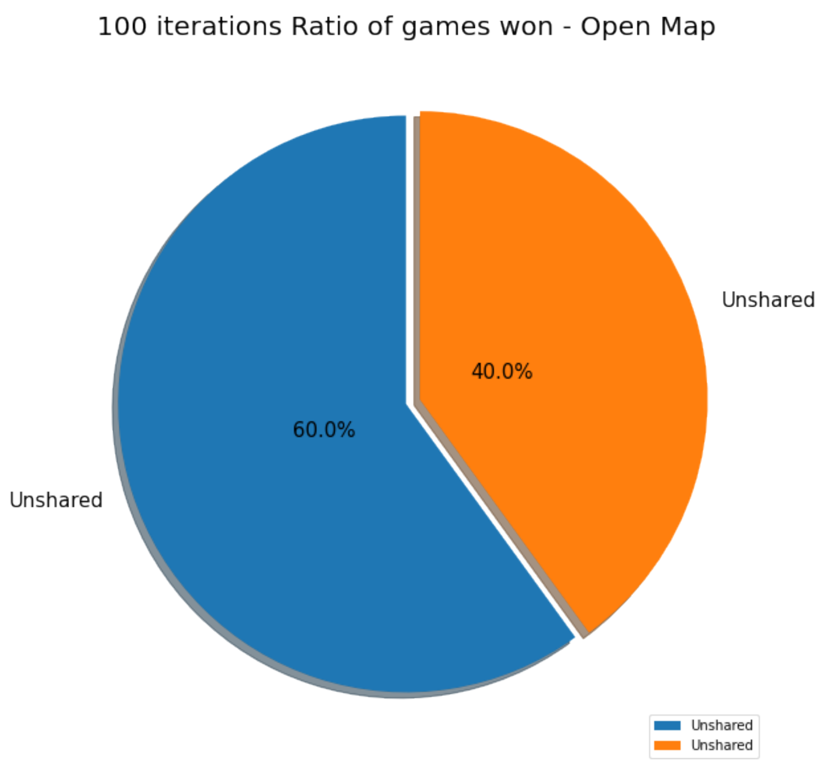

Figure 5.4: Unshared vs Unshared - Open Map

In the run of Unshared vs Unshared on the open map as seen here in Figure 4.1, the blue Unshared team won 60.0% of 100 runs. The orange team with the Unshared IM won 40.0%.

5.2 Closed map

5.2.1 Shared vs Unshared

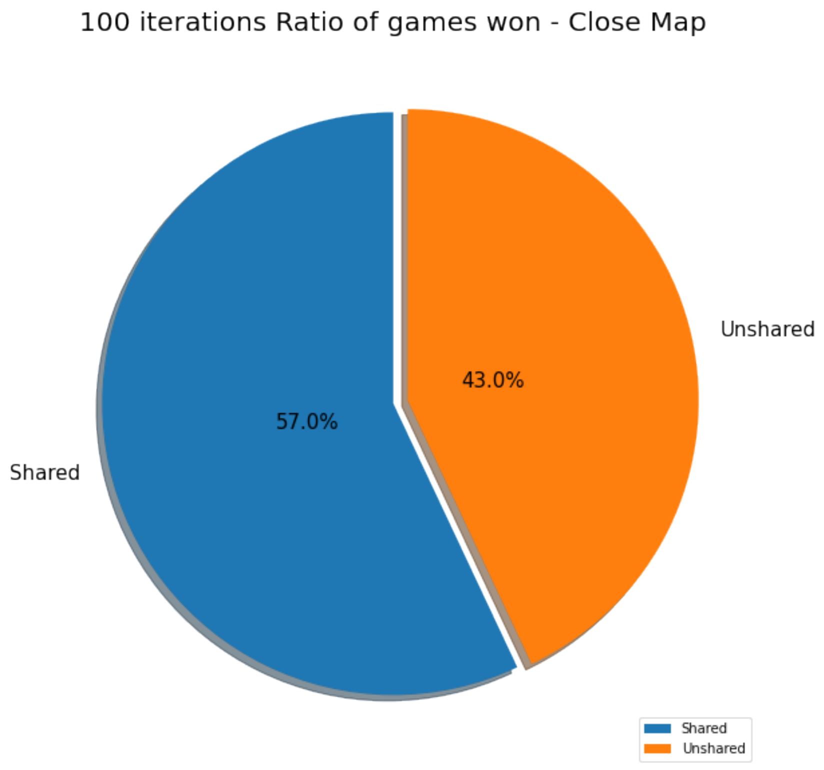

Figure 5.5: Shared vs Unshared - Closed Map

In the run of Shared vs Unshared on the closed map as seen here in Figure 4.2, the Shared team won 57.0% of 100 runs. The team with the Unshared IM won 43.0%. The team with the shared influence card held the three points for an average of 416.90047943115235 seconds. The team with the unshared influence card held the three points for an average of 401.9303968811035 seconds.

5.2.2 Unshared vs Shared

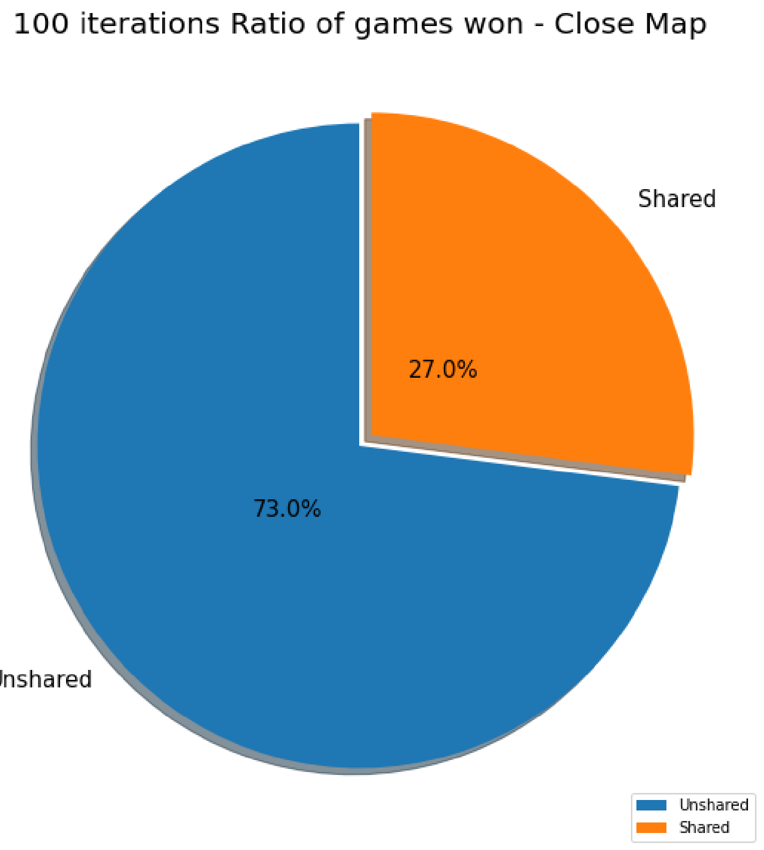

Figure 5.6: Unshared vs Shared - Closed Map

In the run of Unshared vs Shared on the closed map as seen here in Figure 4.2, the Shared team won 73.0% of 100 runs. The team with the Unshared IM won 27.0%. The team with the shared influence card held the three points for an average of 453.7474871826172 seconds. The team with the unshared influence card held the three points for an average of 377.55202514648437 seconds.

5.2.3 Shared vs Shared

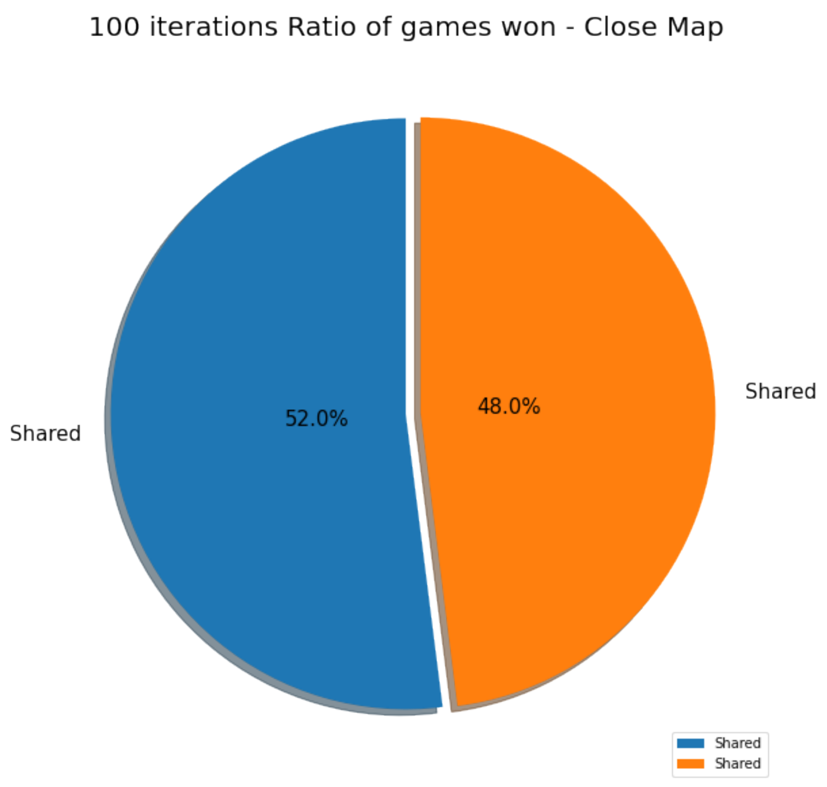

Figure 5.7: Shared vs Shared - Closed Map

In the run of Shared vs Shared on the closed map as seen here in Figure 4.2, the blue Shared team won 52.0% of 100 runs. The orange team with the Shared IM won 48.0%.

5.2.4 Unshared vs Unshared

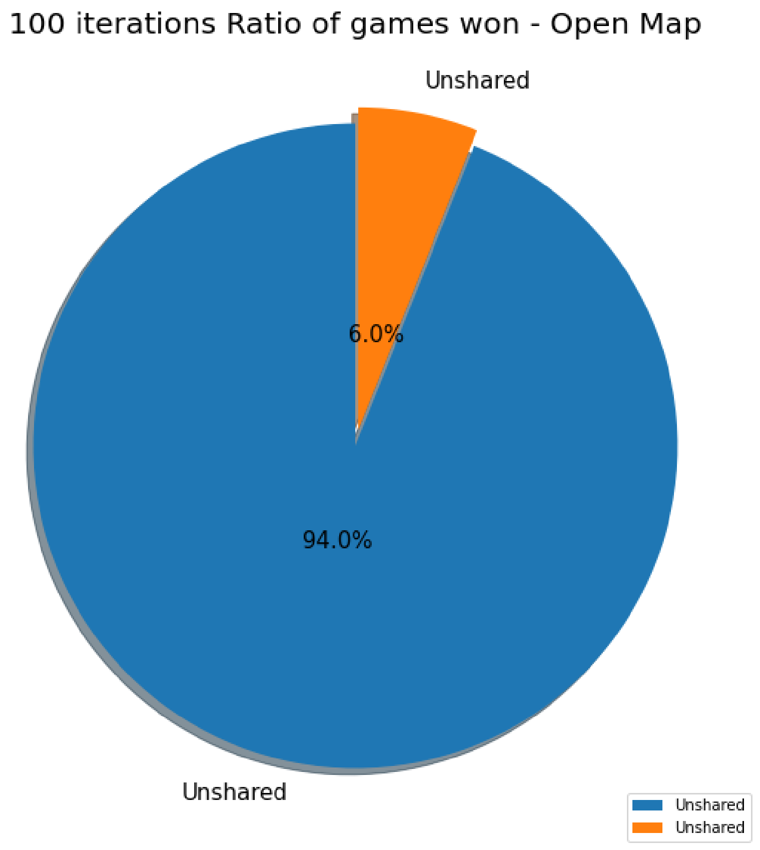

Figure 5.8: Unshared vs Unshared - Closed Map

In the run of Unshared vs Unshared on the closed map as seen here in Figure 4.2, the blue Unshared team won 94.0% of 100 runs. The orange team with the Unshared IM won 6.0%.

Chapter 6

Discussion

The expectations were, as written in chapter 4.2, that the team with the SK AI would be more likely to win than the team that generates a separate IM for each agent. This did not happen as expected. What is immediately noticeable is that the team listed as team 1 always wins. No matter if it is the team with SK or undivided knowledge of the IM. This may be because the IM of team 1 and all other actions in the code before team 2 are calculated seqeuntially and do not run in an asynchronous thread. This could be one of the possible probabilities that this happened. To confirm this, one would have to run more quantitative tests with 100 or more matches each to see if it was just a coincidence or not.

6.1 Upcoming questions

One of the questions that comes up is why team 1 always wins, even when the same AI competes against itself. Furthermore, there is the question of whether the pool of possible strategies was too small for the high-level AI to choose from and therefore there are too few variations to deal with different situations. The same is true for the lower-level AI, which may have too few additive influence events, such as including the flying of projectiles and additively changing the influence where a projectile was.

Chapter 7

Conclusion

Using IM for the high-level AI to choose appropriate strategies or for the low-level AI to help micromanage and react quickly to new circumstances is definitely very helpful. IM is also a very powerful tool for pathfollowing and pathfinding.

To address the scientific question from chapter 1.2 and the aim of this research from chapter 1.3, in this thesis there is no significant difference whether the AI shares an IM or whether each agent has its own.

Using IM is generally a very helpful tool for decision-making. But the strategy pool for the AI to decide on should be much bigger and it needs a lot of balancing. In an RTS or other real time games, it would be advantageous to have strategies like falling back, flanking, reformatting with the other agents from the squad, moving forward into cover and much more. This can be optimally linked with pathfinding from chapter 3.2.1.3 and the waypoints from chapter 3.4, as well as goal oriented action planning (GOAP), finite state machine (FSM) or a behaviour tree.

IM are only aids to decision making and cannot make decisions themselves. They only provide abstract data on the playing field and a kind of memory. In theory, it is better if the AI shares an IM, because that way the high-level AI can see a better abstract picture of the playing field and thus choose better strategies. Likewise, the low-level AI could then react better to possible situations and call up appropriate behaviour. This is also much more performant than generating a separate map for each agent on the field. This could then also be calculated in compute shaders to get more performance and to calculate those simple blur algorithms from chapter 3.3. Furthermore, more information like waypoints from chapter 3.4 and multiple layers could be added to an IM to get more accurate information as described in Dave Mark's work [Mark, 2013].

Chapter 8

Lists

List of Acronyms

IM	influence map
FOW	fog-of-war
AI	artificial intelligence
SK	shared knowledge
RTS	Real Time Strategy
GOAP	goal oriented action planning
FSM	finite state machine

List of Figures

List of Tables

Bibliography

Michaël Carlos Gonçalves Adaixo. Influence map-based pathfinding algorithms in video games, 2014.

Daniel Brewer and Rez" Graham. Knowledge is power: An overview of knowledge representation in game ai, July 2020. URL https://www.youtube.com/watch?v=Z6oZnDIgio4&t=994s.

Alex J. Champandard. The core mechanics of influence mapping, April 2021. URL https://www.gamedev.net/tutorials/programming/artificial-intelligence/the-core-mechanics-of-influence-mapping-r2799/.

EA. Battlefield 1, October 2016. URL https://www.ea.com/en-gb/games/battlefield/battlefield-1/modes?setLocale=en-gb.

Handwiki. Box blur, December 2021. URL https://handwiki.org/wiki/Box_blur.

Dave Mark. Modular tactical influence maps, September 2013.

Ian Mellington. *AI for Games*. CRC Press, third edition edition, December 2020.

Ryan McAlinden Poey Guan Tan Michael van Lent, Paul Carpenter. A tactical and strategic ai interface for real-time strategy games, 2002. URL https://www.aaai.org/Papers/Workshops/2004/WS-04-04/WS04-04-007.pdf.

Steven Rabin. *Game AI Pro 2: Collected Wisdom of Game AI Professionals*. A. K. Peters, Ltd., USA, 2015. ISBN 1482254794.

Technologies Unity Random.Range. Random.range, August 2021. URL https://docs.unity3d.com/ScriptReference/Random.Range.html.