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

Abstract

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

1.1 Motivation

During his studies, Mr Akdogan always wondered how the game AI shares its knowledge. Because if the artificial intelligence (AI) always knows everything, it would not be beneficial for the player and you want to give the player a good experience. While looking for a topic for his bachelor thesis, he came across a game developers conference (GDC) video from [Brewer and Graham, 2020] that talks about the representation of information and also about influence maps for the AI. This raised the question of how much influence shared knowledge (SK) has among AI as opposed to agents not sharing their knowledge with each other.

1.2 Scientific question

How big is the difference between the AI agents when they share their influence map or each agent uses its own influence map?

1.3 Structure of the Thesis

Chapter 2

Related work

In Rabin [2015] article discussed how influence map (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].

An IM is used to record the current balance of Military Influence at each position in a level. 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
2. Targeting locations

3. Movement destinations

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 high-level 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 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.

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. 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** 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** This parameter describes how often the influence is updated.

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

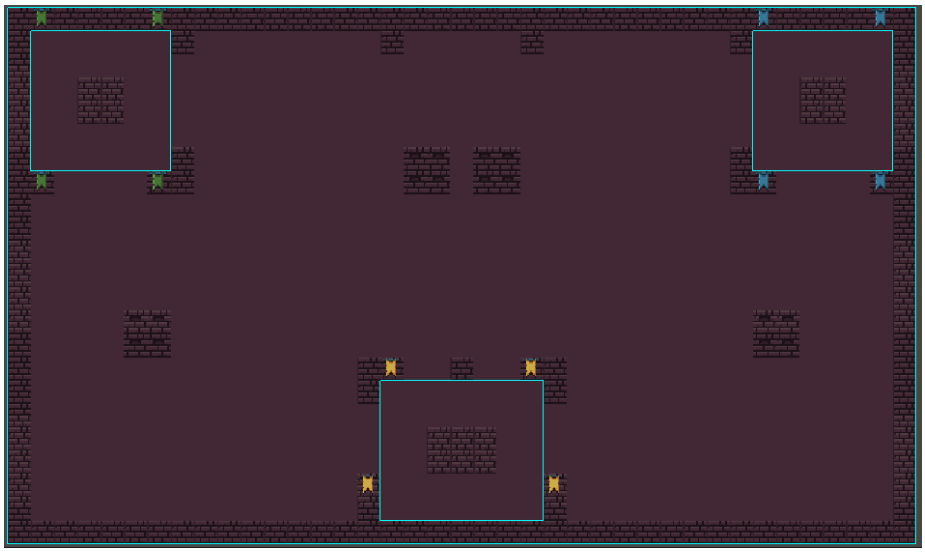

Figure 4.1: An open map with three capturable points marked with a cyan rectangle.

4.3.1.2 Closed map Structure

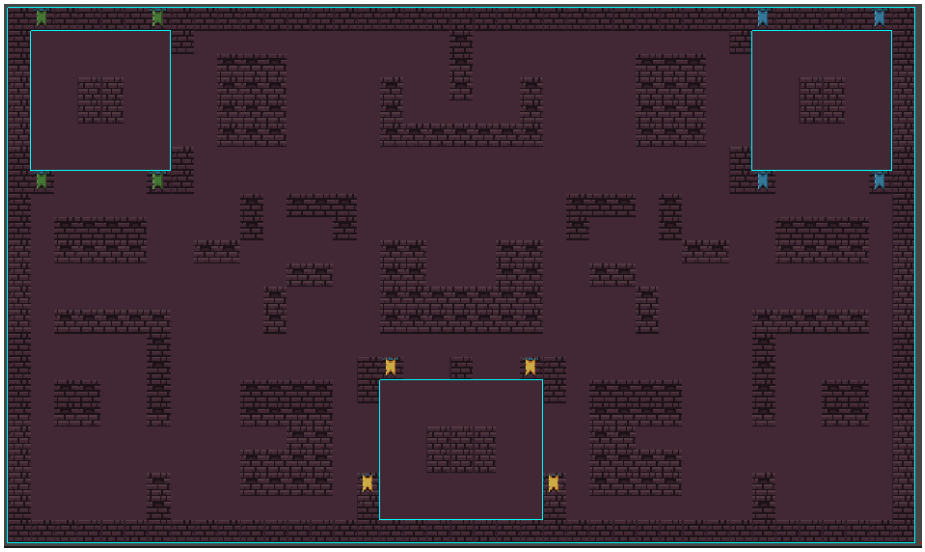

Figure 4.2: A closed map with three capturable points marked with a cyan rectangle. Here are many angles and small corridors distributed on the map and it was careful not to have too many open spaces so that the influence must propagate more around the corners and that the agents are not very line of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

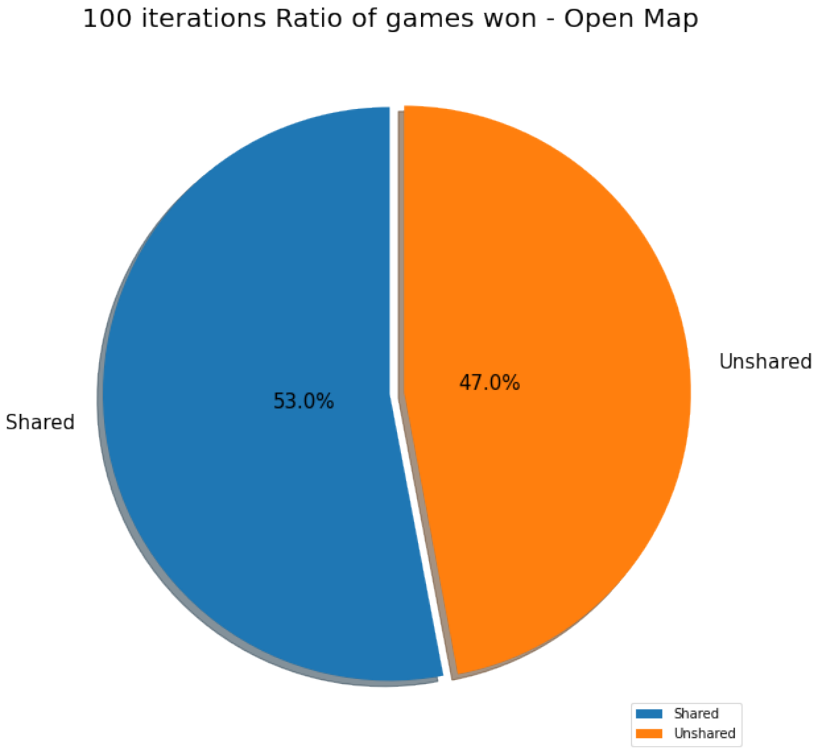

Figure 5.1: Shared vs Unshared - Open Map

5.1.2 Unshared vs Shared

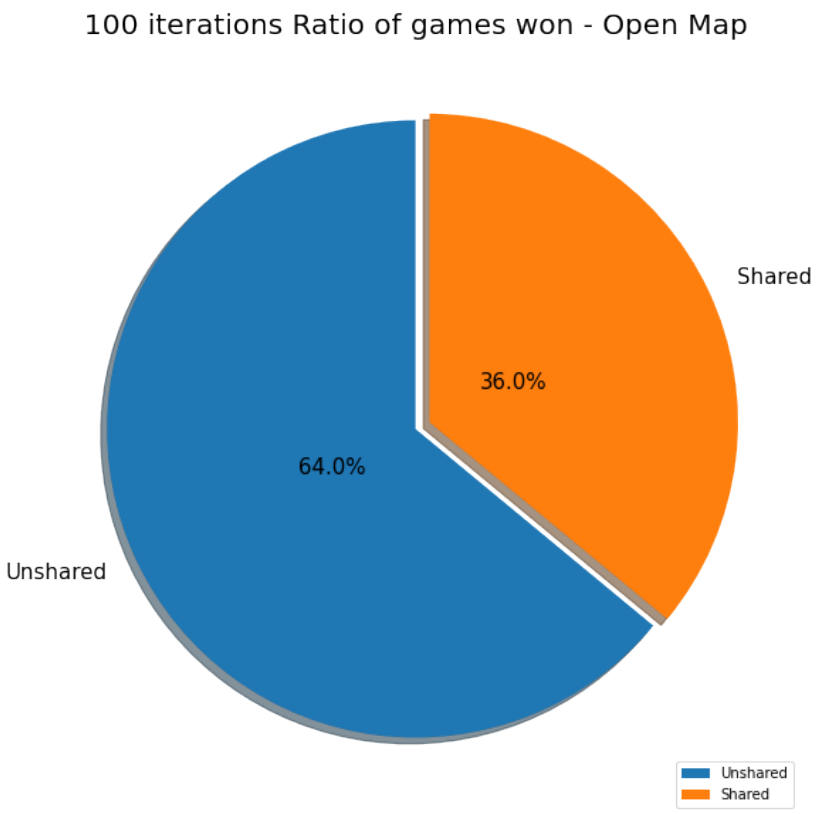

Figure 5.2: Unshared vs Shared - Open Map

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.

5.1.3 Shared vs Shared

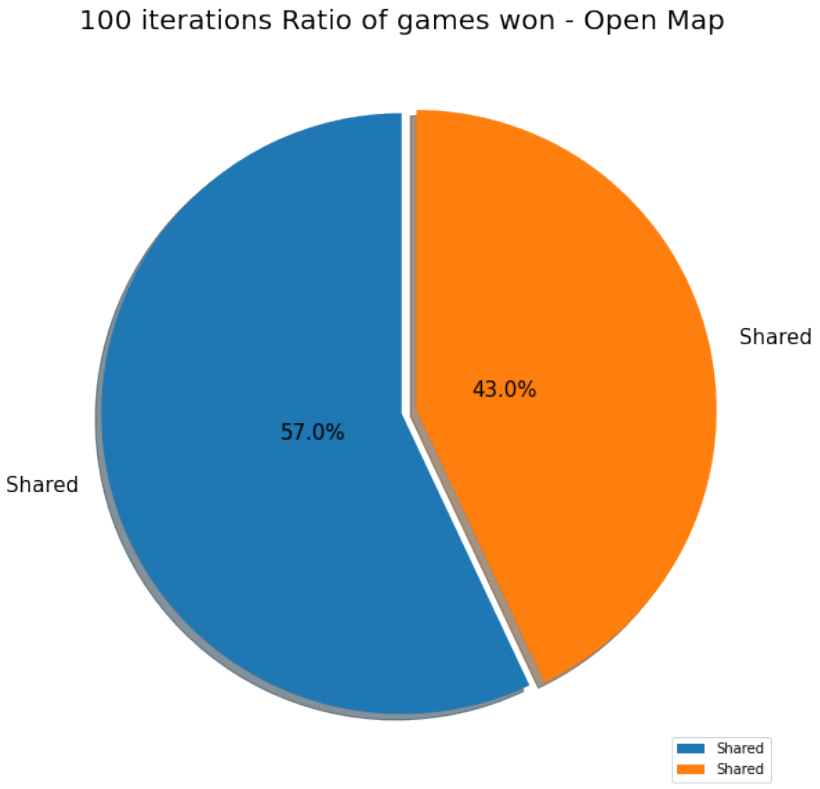

Figure 5.3: Shared vs Shared - Open Map

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

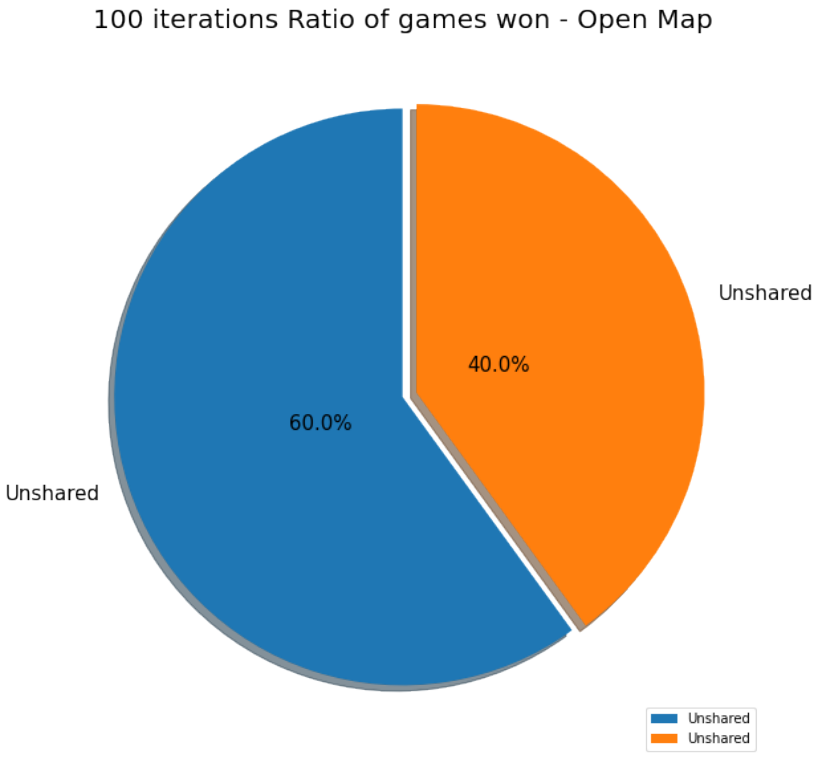

Figure 5.4: Unshared vs Unshared - Open Map

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

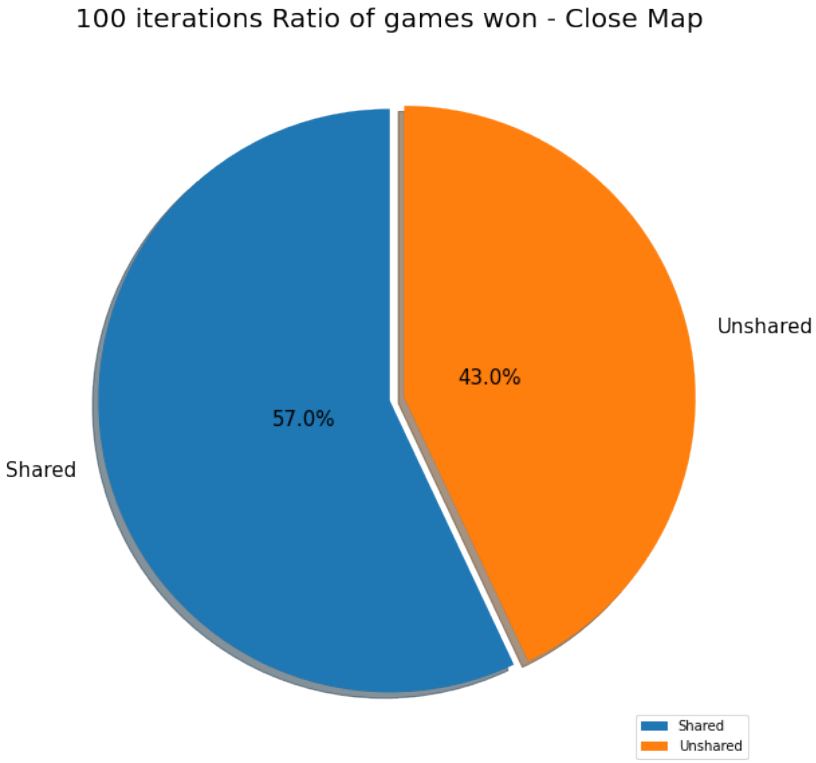

Figure 5.5: Shared vs Unshared - Closed Map

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%.

5.2.2 Unshared vs Shared

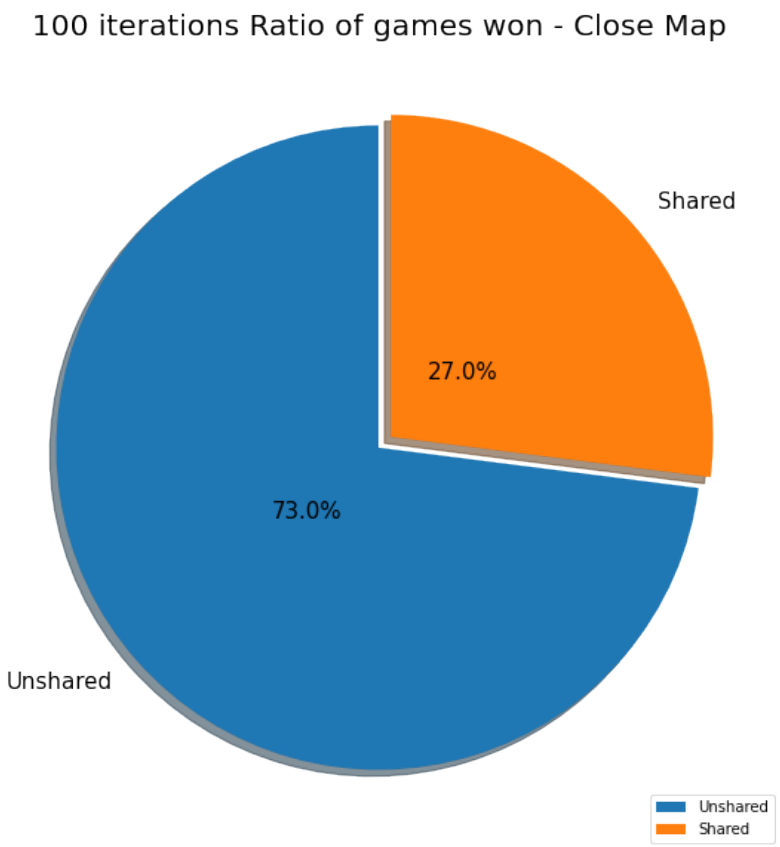

Figure 5.6: Unshared vs Shared - Closed Map

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%.

5.2.3 Shared vs Shared

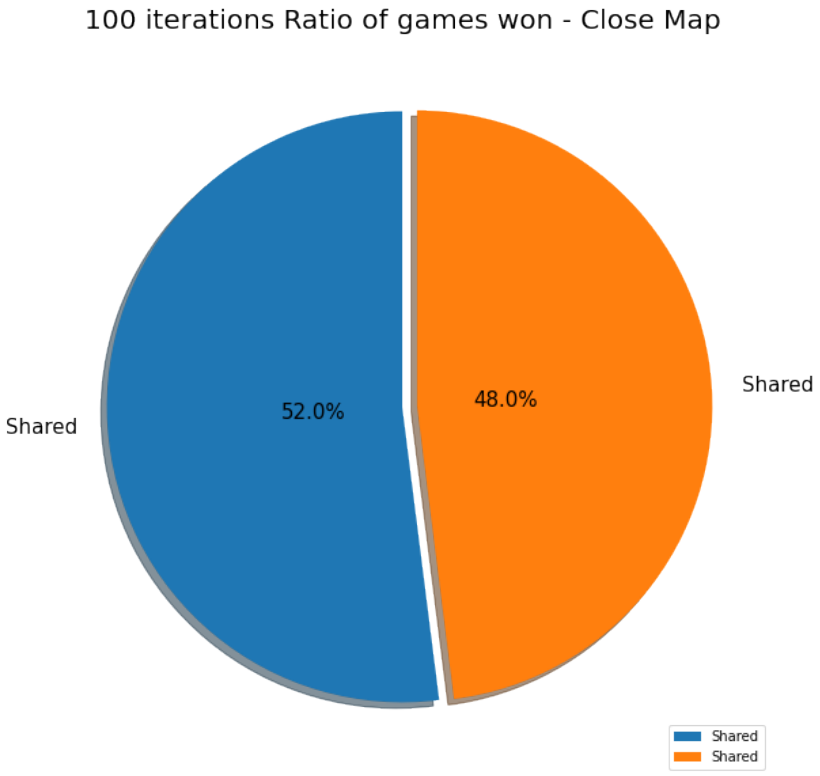

Figure 5.7: Shared vs Shared - Closed Map

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

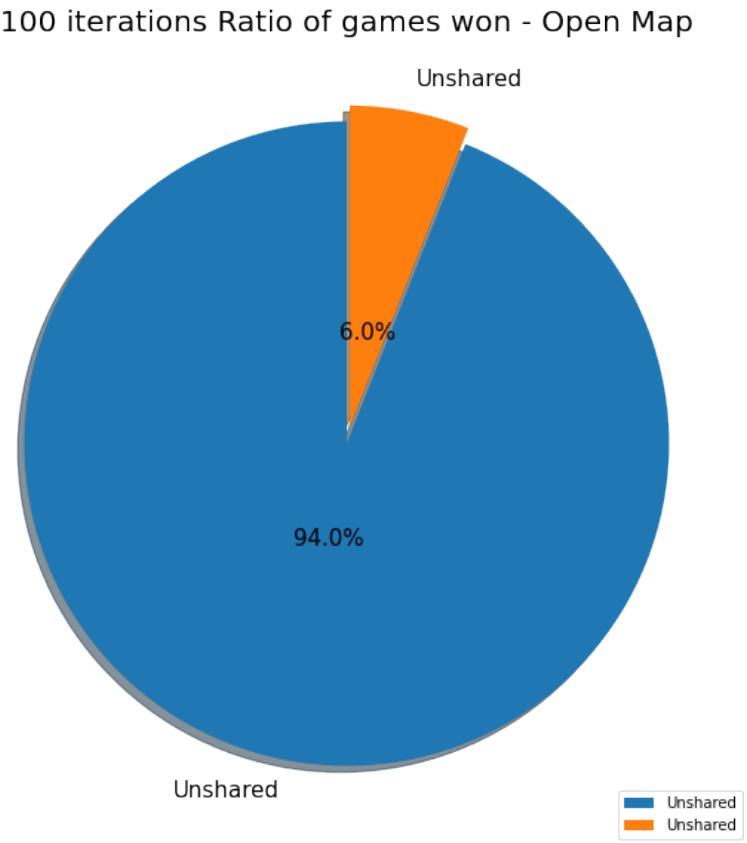

Figure 5.8: Unshared vs Unshared - Closed Map

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

As in Chapter 4.2, the assumption was that the team with the SK from the IM has a higher win rate than the team in which each agent has its own IM.
What is immediately noticeable is that team 1 always wins in the 8 times 100 iterations, regardless of whether it is equipped with SK or unshared knowledge. This can come from the fact that team 1 is the first in the calculation of the IM from the sequential flow in the code. This could be one of the possibilities, but to confirm this would require running several 100 iteration packages of the respective team setup.

Chapter 7

Conclusion

Chapter 8

Lists

List of Acronyms

IM influence map

FOW fog-of-war

AI artificial intelligence

GDC game developers conference

SK shared knowledge

RTS Real Time Strategy

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.