# Potential-Field Based navigation in StarCraft

Johan Hagelbäck, *Member, IEEE*

***Abstract*—Real-Time Strategy (RTS) games are a sub-genre of strategy games typically taking place in a war setting. RTS games provide a rich challenge for both human- and computer players (bots). Each player has a number of workers for gathering resources to be able to construct new buildings, train additional workers, build combat units and do research to unlock more powerful units or abilities. The goal is to create a strong army and destroy the bases of the opponent(s). Armies usually consists of a large number of units which must be able to navigate around the game world. The highly dynamic and real-time aspects of RTS games make pathfinding a challenging task for bots. Typically it is handled using pathfinding algorithms such as A*, which without adaptions does not cope very well with dynamic worlds. In this paper we show how a bot for StarCraft uses a combination of A* and potential fields to better handle the dynamic aspects of the game.**

## I. Introduction

Real-Time Strategy (RTS) games provide many challenges both for human and computer-controlled (bot) players. The player usually starts with a command center and a number of workers. The workers must be used to gather one or more types of resources from resource spots and drop them off at the command center. The resources can in turn be used to construct buildings that expand your bases, or to build units that may attack the opponent or defend your own base(s). It is also common that RTS games have technology trees where the player can invest resources in upgrades for units and/or buildings. The game usually ends when a player has destroyed all the buildings of the opponents which requires a lot of skill and micro-management by the player.

Navigation of units in RTS games is typically handled with pathfinding algorithms of which A* is the most common. The A* algorithm does not handle dynamic worlds very well. A found path is static and it takes time for an agent to travel along it. If the path is blocked by a moveable object, which was at another place when the path was calculated, the path becomes obsolete and the agent has to re-calculate all or parts of it. Extensive work has been done to modify A* to work better in highly dynamic worlds. Silver proposes an addition of an extra time dimension to the pathfinding graph to allow units to reserve a node at a certain time [1]. The work of Olsson addresses the issue of changes in the pathfinding graph due to the construction or destruction of buildings [2]. Koenig and Likachev have made contributions to the field with their work on Real-Time A* [3], [4].

*Potential Fields* is a concept originating from robotics. It was first introduced by Khatib for real-time obstacle avoidance for manipulators and mobile robots [5]. It works by placing attracting or repelling charges at important locations in the virtual world. An attracting charge is placed at the position to be reached, and repelling charges are placed at the positions of obstacles. Each charge generates a field of a specific size. A repelling field around obstacles are typically small while the attracting field of positions to be reached has to cover most of the virtual world. The different fields are weighted and summed together to form an aggregated field. The total field can be used for navigation by letting the robot move to the most attracting position in its near surroundings. Many studies concerning potential fields are related to spatial navigation and obstacle avoidance, for example the work by Borenstein and Massari [6], [7]. Alexander describes the use of fields for obstacle avoidance in the games Blood Wake and NHL Rivals [8]. Johnson describes obstacle avoidance using fields in the game The Thing [9].

In a series of papers Hagelbäck and Johansson have described the use of Potential Fields for navigation in the open-source RTS game ORTS. They suggest that each interesting object in the game world, friendly, neutral or hostile, generates a Potential Field based on the current agent type and the type of affecting object.

If the agent has a goal to move to, an attracting field is also placed at the goal position. This field is typically linear or exponential and has the most attractive value at the goal (distance=0), and is decaying to zero [10].

Fields around obstacles such as other own units are small and repelling to make agents pass around obstacles at a suitable distance. Figure 1 shows an example of a field generated by an obstacle.

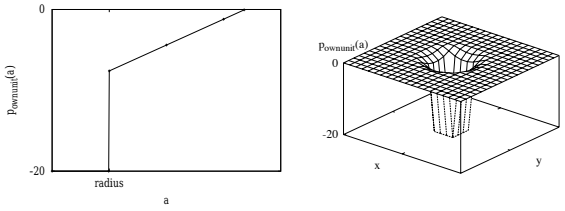


Fig. 1. The repelling field generated by obstacles at distance $a$.

Fields generated by opponent units are attracting using a piecewise linear or exponential function with the highest value at some other distance than zero. The reason is that we want to keep agents at a proper distance from enemy units, which typically is the maximum shooting distance (MSD) of the agent. This will make our forces surround the enemy around the maximum shooting distance since that is where the most attracting potentials are [10]. Figure 2 shows an example of

Johan Hagelbäck is with the Blekinge Institute of Technology, Karlskrona, Sweden (email: johan.hagelback@bth.se)

978-1-4673-1194-6/12/$31.00 ©2012 IEEE

a piecewise linear field generated by an opponent unit. MDR stands for Maximum Detection Range. Outside this distance an agent is not affected by an opponent unit.

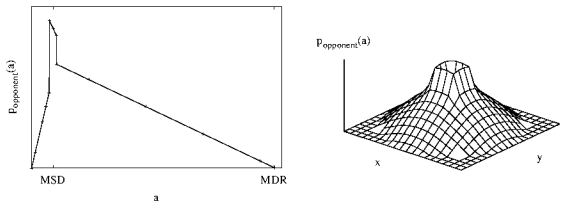


Fig. 2. The attracting field generated by opponent units at distance $d$.

Figure 3 shows a screenshot from StarCraft where a number of Terran Siege Tanks using piecewise linear potential fields surrounds a Zerg base at maximum shooting distance (shown as blue lines) of the tanks.

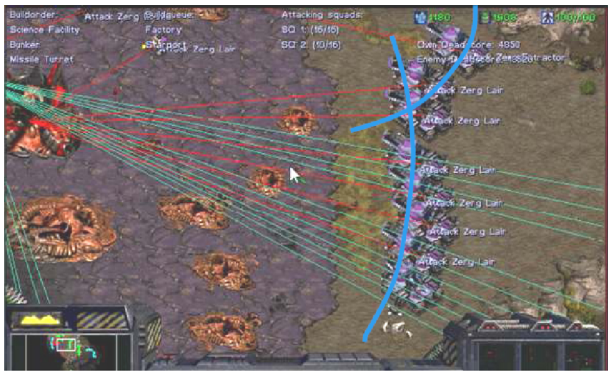

Fig. 3. Screenshot from a Terran force attacking a Zerg base.

When navigating using potential fields a total field is calculated using Eq. 1, where N is the number of objects affecting position $(x, y)$ and $w_i$ is the weight for subfield $i$. The agent checks the total potential in all positions within a certain range of itself. The range, or lookahead, is typically the max distance the unit can travel in a certain time frame. The agent then moves to the position with the highest (most attractive) potential. Note that in this paper higher potential field values are the most attractive, but in some literature lower values are instead the most attractive.

$$p_{total}(x, y) = \sum_{i=1}^{N} w_i p_i(x, y) \quad (1)$$

The problem with this approach is that the agent can get stuck in a local optima. It means that the highest potential is at the current position of the agent, and that position is not the end destination. A possible solution to this problem is to let the agent move some distance in a random direction when it is stuck. It can however not guarantee that the agent will move back to the original position and once again get stuck. A more robust solution is to assign small repelling fields to the last positions of the agent, similar to a pheromone trail used by ants. The effect is that the trail pushes the agent forward and prevents it from going back in the direction it came from. This approach has been used with success in ORTS where a trail of the 20 last positions were used [11].

The upper picture in Figure 4 shows another effect that can happen if the total potential field is always calculated as a weighted sum of all subfields. If several opponent units are in the same area, as often happens in RTS games, the highest potential in the total field tends to be in the center of the opponent force. This would make our agents approach the enemy at their strongest point [11].

The proposed solution was to calculate the total potential field using Eq. 2. The first part of the equation is the same as Eq. 1, except that N here means all affecting objects except enemy units and buildings. The second part means taking max of all enemy fields generated by enemy units and buildings, M, instead of a weighted sum of them. This is illustrated in the bottom picture in Figure 4. This change greatly improved the performance of the ORTS bot [11].

$$p_{total}(x, y) = \sum_{i=1}^{N} w_i p_i(x, y) + max_{j=1}^{M}[p_j(x, y)] \quad (2)$$

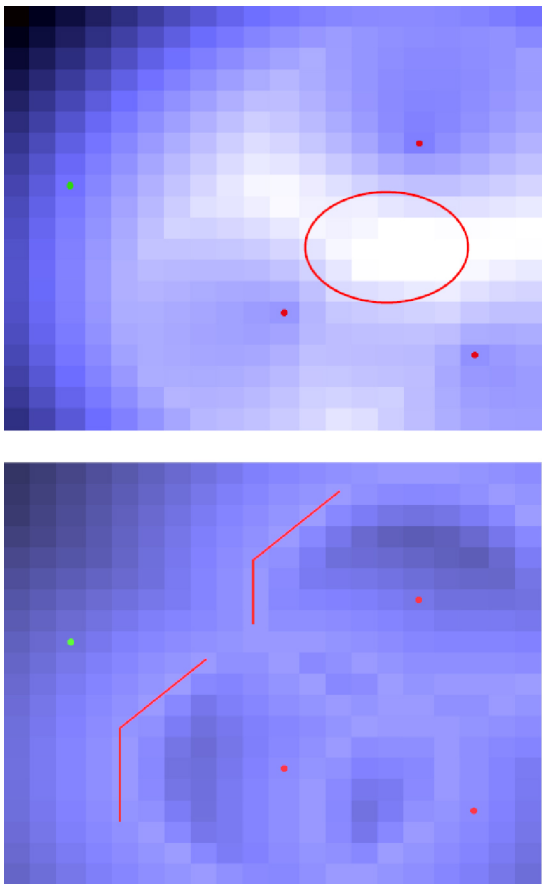

Fig. 4. Total potential field when using the sum of opponent fields (top) or the max of opponent fields (bottom). Light areas have higher potential than darker areas.

Another proposed improvement was to not only let generated fields depend on the agent and the affecting object, but also on the internal state of the agent [11]. In the upper picture in Figure 5 an agent is in attack state and the field generated by the opponent unit has the highest potential at the maximum shooting distance of the agent. In the bottom picture in Figure 5 the agent is in retreat state and the highest potential is outside the maximum shooting distance of the opponent unit.

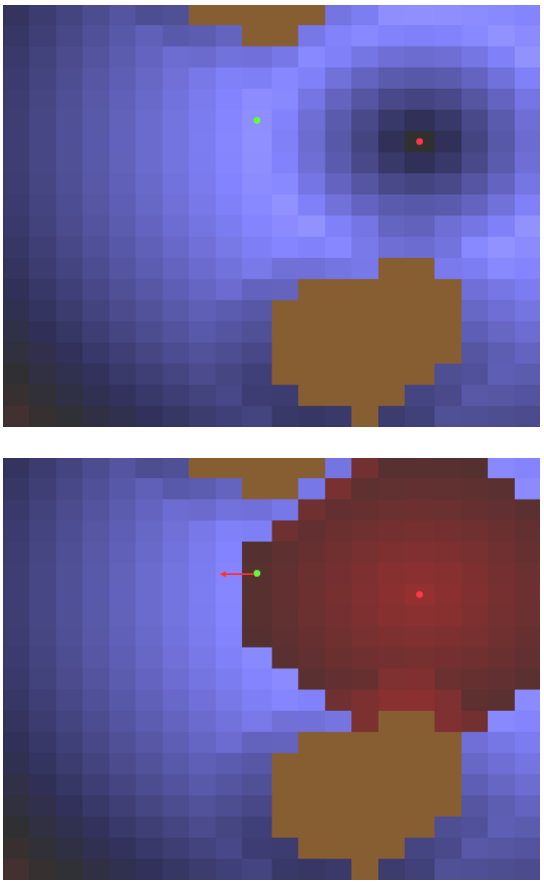

Fig. 5. Total potential field for an agent in attack state (left) and in retreat state (right). Red areas are repelling.

## II. Potential Fields in StarCraft

We have implemented a bot for playing the famous RTS games StarCraft:Broodwar using the BWAPI framework to communicate with the StarCraft engine [12]. It uses a navigation system that can switch between A* (the built-in pathfinding in StarCraft), potential fields, or a combination of both. The bot is released as open source and is available for download[1].

In previous work the potential field based navigation system was implemented in the ORTS engine and tested in a scenario called Tankbattle [11], [10]. In this scenario only one type of unit (tanks) and one type of building (command center) was used. StarCraft is a much more complicated game with a wide range of units and buildings with different properties. StarCraft features for example both ground and air units. Some units can only attack air targets while others only can attack ground targets. There are also units that can attack both air and ground targets, but might have a different maximum shooting distance for air and ground. StarCraft also features units that cannot directly attack opponents, for exemple Protoss High Templars or Terran Science Vessels. These units can have some sort of support functionality (Terran Science Vessels can detect cloaked (hidden) units) or can attack with special abilities which costs energy and can therefore only be used a limited number of times (High Templars' Psionic Storm).

Potential Fields generated by obstacles follow the principles described in the Introduction. Each obstacle generates a small repelling field with the size of the radius of the agent plus the radius of the obstacle plus a minimum distance when passing around each other, in our implementation half the size of a Terran Marine unit. The only exception is for the cloaked units Protoss Dark Templars and Zerg Lurkers. Unless the opponent has a detector unit nearby, Lurkers and Dark Templars cannot be seen by the enemy. They can however be damaged by for example Terran Siege Tanks which do damage in an area around the target instead of just damaging the target. To avoid getting unnecessary damage Dark Templars and Lurkers have a minimum distance of the splash damage radius of Terran Siege Tanks between themselves and other units.

The potential field generated by opponent units are affected by the following unit properties:

- If the agent (own unit) can attack the opponent.
- If the opponent can attack the agent.
- If the agent is in offensive or defensive state.
- The shooting distance of the weapon that can attack the opponent (ground or air).
- The shooting distance of the enemy weapon that can attack the agent (ground or air).
- The detection range of the agent, i.e. if the agent can detect the opponent unit or not.

Depending on the values of these properties different fields are generated around opponent units. The fields can be attracting or repelling, have different size, and have the highest potential field value at different distances from the opponent unit. Table I shows the different combinations of properties and the types of fields generated. Note that when the agent cannot attack an opponent unit and the opponent cannot attack the agent is a special case. If the agent can attack with special abilites a field is generated, otherwise no field is created. The highest potential of a single field is in my implementation set to a float value of 200. In practice this value can be anything, but it should be high enough to be able to generate a decaying field over the whole game world and have a noticeable increase in potential field values between two adjacent positions (i.e. it depends on the size of the game world and if the implementation uses integer, float or double values for representing values).

As in previous work the potential field navigation system

[1]BTHAI Project - http://code.google.com/p/bthai/

| Agent State | Can attack | Can be attacked | Field type | Highest potential | Field size |
|---|---|---|---|---|---|
| Offensive | Yes | Yes | Attracting | MSD of agent | Detection range of agent |
| Defensive | Yes | Yes | Repelling | > MSD of opponent | > MSD of opponent |
| Offensive | Yes | No | Attracting | MSD of agent | Detection range of agent |
| Defensive | Yes | No | Attracting | MSD of agent | Detection range of agent |
| Offensive | No | Yes | Repelling | > MSD of opponent | > MSD of opponent |
| Defensive | No | Yes | Repelling | > MSD of opponent | > MSD of opponent |
| Offensive | No | No | - | - | No field generated |
| Defensive | No | No | - | - | No field generated |
| Offensive | No | No | Attracting | MSD of special ability | Detection range of agent |
| Defensive | No | No | Attracting | MSD of special ability | Detection range of agent |

TABLE I
THE DIFFERENT TYPES OF FIELDS GENERATED BY OPPONENT UNITS.

uses a pheromone inspired trail to avoid getting stuck in local optima. The total potential field is calculated using Eq. 2 to avoid having the most attractive potentials in the middle of an enemy force.

The ORTS engine used in previous work randomly generates terrain based on a seed number. The resulting terrain often have large open areas and few narrow passages which usually can be avoided without taking too long detours. This is ideal for a potential field based navigation system since there are few areas which can cause local optima problems, and some of them can be avoided by filling narrow passages and concave terrain blocks [10].

The terrain in a typical StarCraft map is much more complicated. Bases can often only be accessed through one or more narrow passages and defending these are crucial for winning the game. Air units do not have any limitations and can move anywhere on the map. The heavy use of chokepoints and narrow passages can cause problems for a potential field based navigation system due to numerous local optimas.

We suggest to use a combination of A* and potential fields to 1) minimize the problem of numerous local optimas when using potential fields by using A* when applicable, and 2) benefit from potential fields' ability of handling highly dynamic game worlds. When using the combined approach for navigation A* is used for navigation when no enemy unit is detected by an agent. This allows for effective pathfinding over long distances. As soon as any enemy unit or building is detected the navigation is switched to use potential fields. By doing this most local optima are avoided while still benefit from several positive effects of using potential fields, for example the ability of surrounding enemy units.

Figure 6 shows a screenshot from StarCraft where the potential field values are for a Terran Siege Tank are displayed. Figure 7 shows a similar view as seen by a Terran Marine. Note that the most attracting potentials (light areas) are at different distances from the opponent units due to Siege Tanks and Marines having different maximum shooting distance.

## III. EXPERIMENTS

We have conducted a series of experiments with two versions of the bot playing Terran and facing the built-in AI in StarCraft playing Terran, Protoss and Zerg. The two bot versions are navigation using A* only, and navigation using the combined A* and potential field approach. Each bot version play two matches each on two maps versus the three opponent races.

The maps selected are:

- *Destination 1.1*. A two player map that was part of the map pool in the AIIDE 2011 StarCraft bot tournament.
- *Fading Realm*. A two player map with lots of plateaus at different levels.

The results from the experiment are presented in Table II. They show that the built-in AI did not cause much trouble for either of the bot versions. Both won 11 of 12 games, losing only one game each against Protoss doing an early Zealot rush. The difference in average score between the two bot versions is also very small. To do a better comparison we conducted a second experiment where the two versions played against each other. The results from this experiment are presented in Table III. They show that the combined A* and potential fields approach is clearly the best by winning 5 out of 6 games.

Note that both versions use the same tactics, buildorder, squad setup etc. The only difference is agents using potential fields or not when getting close to enemy units or buildings.

We have also run some games using potential fields only for navigation. These games did however all end in a draw. The bot is very strong in defense, and the numerous choke points of the maps proved to be too complex to handle even with a 20 position trail for solving local optima. The bot was simply not able to move enough forces to the enemy bases to win a game. For this approach to work on a general StarCraft map a more effective way of dealing with local optima is needed.

| Map | Winner | PF+A* Score | A* Score |
|---|---|---|---|
| Destination | PF+A* | 60079 | 44401 |
| Destination | PF+A* | 60870 | 34492 |
| Destination | PF+A* | 54630 | 30545 |
| Fading Realm | A* | 52233 | 79225 |
| Fading Realm | PF+A* | 72549 | 45814 |
| Fading Realm | PF+A* | 82441 | 44430 |
| Avg | | 63800 | 46485 |
| StDev | | 11527 | 17198 |

TABLE III
RESULTS FROM THE BOT PLAYING AGAINST ITSELF.

| Navigation | Map | Opponent | Winner | Own Score | Opp Score | Diff Score |
|---|---|---|---|---|---|---|
| A* | Destination | Terran | BTHAI | 70597 | 40866 | 29731 |
| A* | Destination | Terran | BTHAI | 110570 | 65735 | 44835 |
| A* | Fading Realm | Terran | BTHAI | 75196 | 41194 | 34002 |
| A* | Fading Realm | Terran | BTHAI | 74224 | 40735 | 33489 |
| A* | Destination | Protoss | Opp | 14497 | 32150 | -17653 |
| A* | Destination | Protoss | BTHAI | 104307 | 69208 | 35099 |
| A* | Fading Realm | Protoss | BTHAI | 110003 | 76361 | 33642 |
| A* | Fading Realm | Protoss | BTHAI | 105990 | 67238 | 38752 |
| A* | Destination | Zerg | BTHAI | 51486 | 30981 | 20505 |
| A* | Destination | Zerg | BTHAI | 89065 | 60446 | 28619 |
| A* | Fading Realm | Zerg | BTHAI | 75396 | 39197 | 36199 |
| A* | Fading Realm | Zerg | BTHAI | 72101 | 41370 | 30731 |
| Avg | | | | 79453 | 50457 | 28996 |
| StDev | | | | 27817 | 16050 | 15823 |
| PF+A* | Destination | Terran | BTHAI | 87178 | 51776 | 35402 |
| PF+A* | Destination | Terran | BTHAI | 84743 | 47196 | 37547 |
| PF+A* | Fading Realm | Terran | BTHAI | 86583 | 51571 | 35012 |
| PF+A* | Fading Realm | Terran | BTHAI | 73082 | 38332 | 33859 |
| PF+A* | Destination | Protoss | BTHAI | 109536 | 65580 | 43956 |
| PF+A* | Destination | Protoss | Opp | 12952 | 27001 | -14049 |
| PF+A* | Fading Realm | Protoss | BTHAI | 105147 | 68243 | 36904 |
| PF+A* | Fading Realm | Protoss | BTHAI | 122476 | 84960 | 37516 |
| PF+A* | Destination | Zerg | BTHAI | 83073 | 60569 | 22504 |
| PF+A* | Destination | Zerg | BTHAI | 73484 | 44395 | 29089 |
| PF+A* | Fading Realm | Zerg | BTHAI | 69614 | 36579 | 33035 |
| PF+A* | Fading Realm | Zerg | BTHAI | 66558 | 45427 | 21131 |
| Avg | | | | 81202 | 51877 | 29326 |
| StDev | | | | 27457 | 15835 | 15090 |

TABLE II
RESULTS FROM THE BOT PLAYING AGAINST THE BUILT-IN AI IN STARCRAFT.


Fig. 6. Total potential field for a Terran Siege Tank.


Fig. 7. Total potential field for a Terran Marine.

## IV. DISCUSSION AND FUTURE WORK

The results from the two experiments show that a combined A* and potential field based navigation system outperforms the same bot using only A* for navigation. As discussed earlier in this paper, potential field based approaches can effectively surround opponents by using piecewise linear or exponential subfields. When combining A* and potential fields it is important to not switch to use PF's too late since that will not give agents any time to spread out around the enemy. It is almost equally important to not switch too early and risk getting stuck in complex local optimas. The detection range for StarCraft units are in all cases (except Siege Tanks in siege mode) slightly longer than the maximum shooting distance and worked well to use as a limit for the switch.

A navigation system solely based on potential fields did however not work at all due to the complexity and numerous local optima in the selected StarCraft maps. Table 8 shows an example of a map that is very difficult for an approach based on potential fields only. The screenshot is from the top of the map showing the starting area of one of the players.

The starting area of the other player is below. The yellow circle shows the only entry/exit to the starting area. If agents are given the order to move to the opponent starting point, the most attractive potential would be to the south of the own starting area and agents would miss the northern exit. This can probably be solved by using better techniques for solving local optima but that is out of scope of this paper.

Fig. 8. Screenshot from the Baby Step map in the Broodwar expansion. The yellow circle shows the only entry/exit point of the starting area.

A possible direction for future work could be to detect chokepoints, for example with A*, and place attracting sub-goals at each of them. This could be a solution to the extensive problems with local optima using potential fields only as navigation system.

An earlier version of the bot used in the experiments was part of the 2011 StarCraft bot competition [13]. The bot was playing Zerg with a quite daring Lurker rush tactic and ended on a not so impressive 10:th place out of 13 participants. Since then a lot of bugs have been fixed (the bot had a crash rate of almost 13%) and some minor improvements made. It would be interesting to see how competitive the current version of the bot is, and we plan to participate in the 2012 StarCraft bot competition.

## REFERENCES


[1] D. Silver, "Cooperative Pathfinding," in *AI Game Programming Wisdom 3*. Charles River Media, 2006.

[2] P.-M. Olsson, "Practical Pathfinding in Dynamic Environments," in *AI Game Programming Wisdom 4*. Charles River Media, 2008.

[3] S. Koenig, "A comparison of fast search real-time situated agents," in *Proceedings of Autonomous Agents and Multi-agent Systems (AAMAS)*, 2004.

[4] S. Koenig and M. Likhachev, "Real-Time Adaptive A*," in *Proceedings of Autonomous Agents and Multi-agent Systems (AAMAS)*, 2006.

[5] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The International Journal of Robotics Research*, 1986.

[6] J. Borenstein and Y. Koren, "The vector field histogram: fast obstacle avoidance for mobile robots," *IEEE Journal of Robotics and Automation*, 1991.

[7] M. Massari, G. Giardini, and F. Bernelli-Zazzera, "Autonomous navigation system for planetary exploration rover based on artificial potential fields," in *Proceedings of Dynamics and Control of Systems and Structures in Space (DCSSS) 6th Conference*, 2004.

[8] B. Alexander, "Flow Fields for Movement and Obstacle Avoidance," in *AI Game Programming Wisdom 3*. Charles River Media, 2006.

[9] G. Johnson, "Avoiding Dynamic Obstacles and Hazards," in *AI Game Programming Wisdom 2*. Charles River Media, 2006.

[10] J. Hagelbäck and S. J. Johansson, "Using Multi-agent Potential Fields in Real-Time Strategy Games," in *L. Padgham and D. Parkes editors, Proceedings of the Seventh International Conference on Autonomous Agents and Multi-agent Systems (AAMAS)*, 2008.

[11] ——, "The Rise of Potential Fields in Real Time Strategy Bots," in *4th Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, 2008.

[12] A. Heinermann, "BWAPI — An API for interacting with with StarCraft:Broodwar," http://code.google.com/p/bwapi/ URL last visited on 2012-06-20, 2012.

[13] M. Buro, "The 2nd Annual AIIDE Starcraft AI Competition," https://skatgame.net/mburo/sc2011/ URL last visited on 2012-06-20, 2011.