Intelligent Moving of Groups in Real-Time Strategy Games

Holger Danielsiek, Raphael Stüer, Andreas Thom, Nicola Beume, Boris Naujoks, Mike Preuss

Abstract— **This paper investigates the intelligent moving and path-finding of groups in real-time strategy (RTS) games exemplified by the open source game Glest. We utilize the technique of Flocking for achieving a smooth and natural movement of a group of units and expect grouping to decrease the amount of unit losses in RTS games. Furthermore, we present a setting in which Flocking will improve the game progress. But we also demonstrate a situation where Flocking fails. To prevent these annoying situations, we combined Flocking with Influence Maps (IM) to find safe paths for the flock in real time. This combination turns out to be an excellent alternative to normal movement in Glest and most likely in other RTS games.**

I. Introduction

Bio-inspired techniques (computational intelligence, CI) have mostly been integrated in simple games, e.g. arcade video games, for improving the game's artificial intelligence such as the intelligence of non-player-characters (NPC) or the handling of the game [1]. CI techniques can also be used in more complex games like RTS games. This genre deals with complex strategies and intelligent behavior so that the use of these methods is very beneficial.

Although path-finding of units can be calculated with the A*-algorithm [2], problems with the movement of groups still exist. Examples are the cohesion within a group of units, the inflexible behavior of a group, or a situation based on intelligent path modification of moving units.

Strategy games are usually placed in war or battle scenarios, where all players start with similar resources, have to build up a base as well as an army and destroy the other players' bases and armies. The RTS game we used is Glest, version 2.0, which is published under GNU Public License (GPL, [3]) by Figueroa et al. [4]. Glest is settled in a fantasy medieval world. The players can choose to control either a *magic* or a *tech* faction, whereas here we consider the magic faction only. Each unit and building of each faction has a property called *hit-points* (HP) representing the unit's health. When a unit is attacked, the HP are decreased by the damage value of the attacking weapon. A unit dies when its HP reaches zero.

Undoubtedly, the way units move is important for winning a game, because an intelligent moving behavior might reduce casualties of own units. One approach to solve a couple of problems related to the movement of groups is "Flocking", a technique introduced by Reynolds [5]. He describes an approach to simulate an animated flock of birds. Their motion shows significant differences from the conventional motion of a group because every unit shows a unique behavior while the group sticks together and reaches its global target.

It will be shown that, although performing already well, flocking alone is not able to cope with all problems related to group movement. The path of a group is still calculated using a normal A*-algorithm, which causes problems when passing through enemy territory. One technology to approach this problem are influence maps [6]. These are areal based representations of the knowledge that the game's artificial intelligence holds about the current game situation. So an IM indicates where units are situated or where the boundaries of the controlled area lie. This information is available for calculating a path for a group or unit, which could be more intelligent and might reduce the deficits mentioned.

The mentioned techniques flocking and influence map are introduced in the following sections. The sections also describe realization and combination of these techniques as well as their implementation in the RTS game Glest. Experiments investigating the behavior of the methods in detail are given in Section V. Section VI concludes our work and gives suggestions for future research.

II. Flocking

Flocking is a suitable method to simulate natural group movement in computer games of e.g. swarms of birds, schools of fishes, military units, or crowds. Shen and Zhou [7] describe the use of flocking to steer the behavior of military units in the futuristic ego-shooter unreal tournament. Davison [8] mentions the use of flocking to control the movement of military units or to coordinate the formation of any units in the ego-shooter Half-Life.

Within Reynold's fundamental paper on flocking [5], the individuals of the simulated flock are called "boids". These boids adjust their behavior and their motion by following three simple rules:

- **Separation:** Boids try to scatter from other boids. This rule can be seen as the opposite of the cohesion rule and represents the collision avoidance within the flock.
- **Cohesion:** The ambition of each boid is to stick together with the others. The boids try to advance to the centroid of the group.
- **Alignment:** Boids steer into the same direction as the other boids. Therefore, Reynolds calculates for each boid the average of the summed up direction vectors of the boids that are in the angle of view and in range of sight.

These rules are depicted in figure 1. The final direction of a boid's movement is calculated as the weighted sum of the results of these three rules.

All authors are with the Chair of Algorithm Engineering, Computational Intelligence Group, Faculty of Computer Science, Technische Universität Dortmund, Germany. E-mail: Holger.Danielsiek@tu-dortmund.de, Raphael.Stueer@tu-dortmund.de, Andreas.Thom@tu-dortmund.de, Nicola.Beume@tu-dortmund.de, Boris.Naujoks@tu-dortmund.de, Mike.Preuss@tu-dortmund.de

978-1-4244-2974-5/08/$25.00 ©2008 IEEE

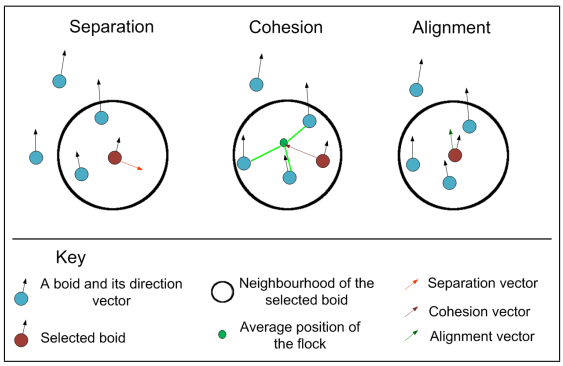

Fig. 1. Schematic diagram of the three flocking rules. The left part shows the resulting separation vector in this special situation, the middle one shows the cohesion influence for one boid and the right part shows the alignment of a boid considering all other boids in its neighborhood.

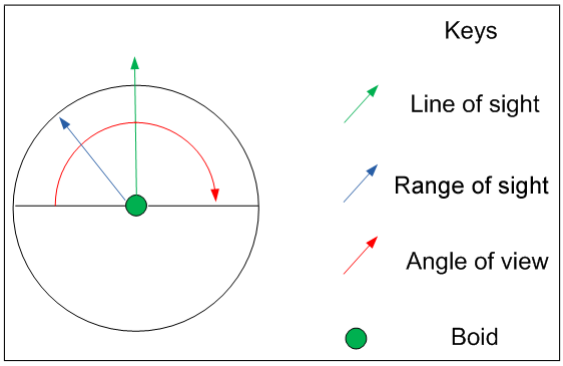

Fig. 2. Angle of view and range of sight of a boid

To implement these rules, the boids need additional information from their neighborhood. This is the location of a boid's neighbors, the directions, the neighbors are heading, and their distance to the original boid. Moreover, each boid features an angle of view (called `ViewAngle` later) and a range of sight, where the boid is able to recognize other boids and its environment (cf. Fig.2). This is invoked to avoid (fixed) obstacles.

The implementation at hand is different from Reynolds original idea to have boids moving around randomly and eventually building a flock [5]. Instead, the player selects some units he wants to group together. This group of units is called flock and a target is assigned to the flock by the player afterwards.

A path to the given target is obtained using the A*-algorithm [2] already implemented in Glest. To follow this path, a local path-finding incorporating the flocking method is started. Here, the direction v of one unit is calculated as follows:

$$v = w_{ref} \cdot v_{ref} + w_{coh} \cdot v_{coh} + w_{sep} \cdot v_{sep},$$

where v_{ref} is the reference vector reflecting the direction of the calculated flock path, $v_{coh} = pos_{avg} - pos_{unit}$ is the vector from the units position to the average position of the flock, and v_{sep} is the separation vector. The corresponding weights are w_{ref} , w_{coh}, and w_{sep}, the latter being referred to as `SeparationWeight`. The length of the separation vector $|v_{sep}|$ is referred to as `SeparationDistance`.

Flocking depends on a number of parameters that influence the behavior of every unit of a flock. The most significant parameters in our scenarios are `ViewAngle`, `SeparationWeight`, and `SeparationDistance`. The `ViewAngle` yields the possibility to shrink and widen the field of view of a boid. A reduced field of view makes the flock behave like a snake, each unit following its predecessor in-line. A broader field of view makes the flock move more scattered. `SeparationDistance` and `SeparationWeight` have an effect on the separation rule, i.e. the ambition of every boid to move away from other boids. For more information on the current implementation we refer to the technical report of the project group 511 [9]. To modify the parameter settings and predefined configurations in a convenient way, appropriate user interfaces are provided.

III. Influence Maps

Influence Maps are usually employed for tactical decisions. They exploit the topographic information of the actual map to represent how players influence different areas of it. Via IMs, players are able to find out where opponent units are concentrated or explore weak, exposed areas. They may be used in any topology. We limit ourselves to two-dimensional environments, since these are typical for most strategy games like Glest. A more broader approach can be found in [10].

A. Calculating Influence Maps

Usually, a map is divided into tiles. The IM is set to the same size as the game's map with typically less tiles than this. The tiles are initialized with a value of 0. To calculate the IM for a game situation, an influence value of every unit on the map has to be derived and propagated through the map according to a given formula. The value to be propagated through the map may stem from any numeric property of a game's unit, e.g. its hit-points or its attack strength. Usually, the influence of a unit decreases linearly or quadratically with increasing distance to the tile, where the unit is located.

The following example shows the calculation of a simple IM. On a rather small map with a height of seven and a width of six tiles (cf. Fig.3), two factions (blue and red) are instantiated. Each faction has three units on the map and each unit has an influence value of 4. The influence of these units decreases linearly with increasing distance d (measured in tiles) to the given unit. The result of the propagation is shown in the central image of figure 3. The influence map is finally retrieved by summing up the influences of all units and all factions. This is shown in the second image from the right in figure 3. Note that the resulting influence may exceed the influence of a single unit if more units of one faction are close to each other. Vice versa, the influences of two adversary units may also cancel each other out. The right image of this figure shows a graphical representation of the calculated influence with the boundaries of the controlled

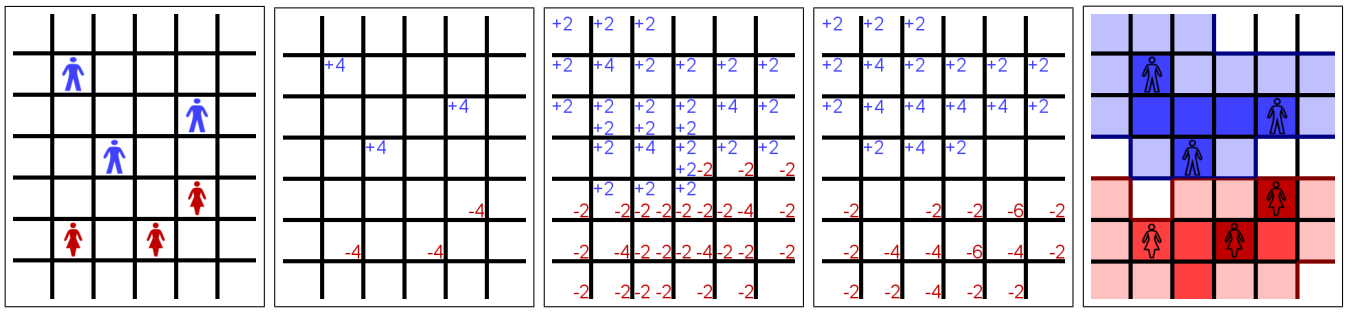

Fig. 3. Example of influence maps generation. The left figure shows the units of the different faction (male, blue and female, red) placed on the map. The second figure yields the local influence of the units ($+4$ for one and -4 for the other faction). The third figure shows the propagation of the influence values through the map, which are summed up in the fourth figure. Finally, the right figure depicts the resulting influence map with the influence of the own (blue) and the rival (red) faction plotted on the map with darker colors meaning higher influence values. In addition, the positions of the original units are shown as well as the borders of the player's influence (outlined in the player's color).

area outlined for each faction. The degree of influence is shown by shades of the faction's color.

B. Implementation details

In order to use influence maps for path-finding, they have to be kept up-to-date at all times. For this purpose, every unit caches its influence, which can be implemented in different ways. We combine the unit's hit-points (HP) and the damage (D) a unit is able to cause to its combat strength c_u:

$$c_u = \frac{D_u * HP_u}{100},$$

which is propagated over the range of sight of the unit decreasing linearly with increasing distance to it, like explained in the example above. Moving a unit now means to move the whole area of influence of a special unit to the new position. To this end, the influence is subtracted from the map at the old position, added at the new one, and propagated through the whole area of influence of the unit afterwards.

To calculate a path from a current position of a unit to a target position, the influence of the unit at hand has to be subtracted from the current IM. This pre-processing step is necessary to prevent units from being affected by their own influence. The resulting accumulated influence at each position (x, y) of the IM is called $\zeta_{x,y}$ in the following. This influence has to be integrated into the cost function used by the A*-algorithm of Glest to calculate an appropriate path for the unit. Note that this path will not reflect the shortest Euclidean distance directly.

To integrate the influence values received from the IM in the cost considered by the A*-algorithm, this cost τ is defined as the product of the Euclidean distance δ and a scaling factor from the IM:

$$\tau := \delta \cdot e^{\psi(x,y)}$$

The considered influence factor $\psi(x, y)$ is based on the influence ε received from the IM via

$$\varepsilon(x, y) = \frac{\zeta_{x,y} - c_u}{180}$$

A unit's strength is subtracted here to allow strong units to walk through tiles with a low overall influence of the opponent; 180 has proven to be a good scaling value. $\psi(x, y)$ then introduces boundaries for the ε to avoid inappropriate τ values.

$$\psi(x, y) = \begin{cases} -1 & \text{if } \varepsilon < -1, \\ 1 & \text{if } \varepsilon > 1, \\ \varepsilon & \text{otherwise} \end{cases}$$

With τ close to zero, a unit would never reach its final destination because A* would not be able to calculate an appropriate path with all cells sharing costs close to zero. Vice versa, expensive costs may result in dead ends even in open-range areas because every surrounding cell is much more expensive than the one the unit is located on. This would result in the unit stopping its move.

The actual implementation makes sure that tiles with a strong influence of the own faction become extremely cheap, whereas tiles with enemy influence become much more expensive. This way, the unit will prefer to move around enemy territory.

IV. Combination of techniques

To deal with flocks in our influence map implementation, a flock is considered as one single unit when calculating the path. Every unit has to update the influence map with its own cached influence as described above, but to determine the costs of a tile a new combat strength c_F for the complete flock F is calculated:

$$c_F = \frac{1}{SEW} \cdot \sum_{u \in F} HP_u \cdot \sum_{u \in F} D_u$$

The idea is that the combat strength of a group is higher than the simple sum of strengths of the single units. This holds because the flock itself causes more damage to single opponent units, while damage caused to the flock is scattered over the single units in the flock. To achieve some balancing of the synergy effects, we introduced the *synergy effect weight* (SEW). This parameter enables the user to scale the aggressiveness of the flocked units. If SEW is very high, the flock will not attack enemies on its way even if they would clearly win the combat. Vice versa, a very low value of SEW will make the flock attack even overwhelming enemy forces.

The correct choice of SEW leads to units taking the secure path around enemies if the enemy is stronger and the path around is possible. If the safe path is too long, it is more expensive to take this path compared to traveling through the enemy territory. In this case, our unit will take the direct path.

If our unit is stronger than the enemy, the unit/flock will take the direct path and attack the enemy in any case. This behavior is implemented having in mind that the territory is virtually already under control of the own faction.

A. Related Research

The general idea to couple the techniques of flocking and IM is not new. But to our knowledge, the combined techniques have not been used to support group movement and path-finding in RTS games.

Flensbak [11] uses three two-dimensional influence maps for each of the desired behaviors alignment, cohesion, and separation. The boids call the influence maps for information on their neighboring fields. For separation, so called spacers are placed around each boid to avoid collisions. The cohesion method tries to compress the flock as much as possible respecting the fields introduced by the spacers. For the alignment of the flock, the influence map holds an averaged vector of the direction and the velocity of the boids for each position. The influence maps are particularly used to control the shape of the flock here.

Miles and Louis use influence maps as nodes in trees to determine different game situations and as decision support system [12], [13]. These decisions are queued with priorities into a task queue. With every decision they follow one edge to the next child of the introduced tree. Finally, Co-Evolution is used to evolve *Influence Map Tree Based Strategy Game Players*. Although these players use information from the A*-algorithm working on an IM to determine the costs of their objectives, they do not use it for moving their units. This approach is close to the original idea of using IM to analyze game situations for strategic decision making.

Based on the different ideas introduced above and the assumption that the specialized inter-linkage of techniques always depends on the application at hand, we conclude that the combination presented here has never been reported before.

V. Experimental Evaluation

The experiments shall demonstrate advantages and drawbacks of the flocking and the influence map techniques in RTS games. We consider three different maps and observe the behavior of a group of units whereas the flocking and the influence map are turned on or off.

In the first experiment, the flocking is studied in comparison to the unsupported group behavior to get an impression of the performance of this technique. The second experiment surveys the influence map and its combination with flocking whereas the need for the combination of these two methods is exposed. At least three different values of a parameter affecting the IM function are considered. It shall give insights how the IM improve the flocking method. For all settings

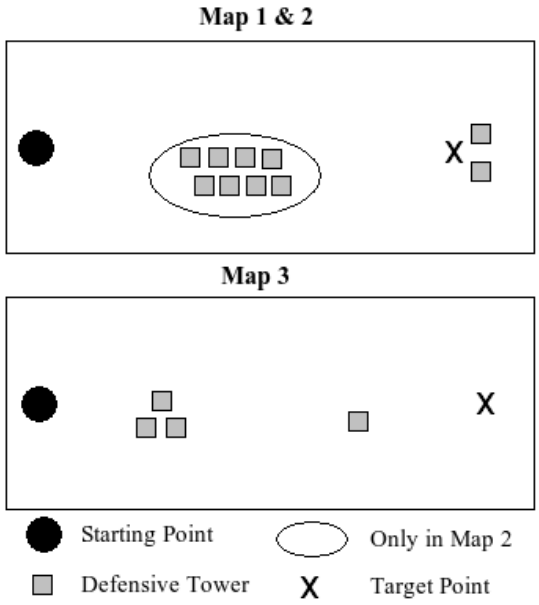

Fig. 4. Underlying maps for the three scenarios tackled. The encircled defensive tower position in the middle of the upper map is only available in scenario 2.

we measure the performance of the group by counting the number of lost combat units and defeated turrets, as well as by calculating the lost overall hit-points of the combat units.

To give a careful and well structured report on the experimental results they are presented according to the guidelines suggested by Preuss [14] which are based upon the framework of sequential parameter optimization by Bartz-Beielstein [15].

A. Experiment 1: Properties of Flocking

Research question: In this experiment, we investigate the properties of flocking by analyzing the performance of a group of combat units. We want to demonstrate that flocking either improves or worsens the performance of a group compared to a normal group behavior. By discovering the drawbacks of flocking we learn in which situations flocking needs to be improved. One of such situation is the second scenario and IM are introduced to improve the units behavior here. This way, the first experiment deals as the pre-experimental study for the second one.

Preexperimental planning: To show the benefit of flocking compared to a normal group behavior, we constructed a setting in which a group loses the final combat without using flocking and wins in the other case. We also give a simple counter example of a typical game situation where a normal group wins but flocking causes a deterioration of performance. This scenario is quiet robust because different flock parameter settings do not change the result of the experiments.

Task: We hypothesize that flocking is helpful in some typical situations in RTS games while there are others where

flocking worsens the performance. This has to be statistically confirmed.

Setup: In our settings a group of combat units is send to two hostile defensive towers (high combat strength, 7000 HP). The first map (cf. Fig. 4) is empty apart from these entities and features no obstacles like mountains or rivers, and it is rather small (64x64 tiles). The group consists of six units with different abilities: two demons (average speed, weak combat strength, short-ranged, 700 HP), two battlemages (average speed, weak combat strength, long-ranged, 700 HP), and two drakeriders (fast speed, high combat strength, medium-ranged, 1300 HP).

In the second map, (cf. Fig. 4), hostile turrets are additionally located along the way the combat units have to walk to get to their target point. A new unit, an initiate (slow speed, weak combat strength, medium-ranged, 450 HP) is added to the group. Remark that the initiate is the weakest unit in the game so that the group's strength does not increase significantly. The units have to pass along eight defense towers which have a sufficient attack range to injure or even kill the units. The chosen parameter settings regarding this scenario are: `ViewAngle` = 90, `SeparationWeight`= 3.0, and `SeparationDistance` = 4.0. For every map and every type of movement we test 500 games with different random seeds. Every unit causes damage within a certain damage range which is affected by random variation.

Results/Visualization: The results received are first clustered in wins and losses for the attacking units and are presented in Fig. 5. The two graphics show bar charts for the corresponding numbers received on map one and map two. Both are placed on top of each other with map one above. Each depicts the charts for four scenarios, namely the normal movement and the flocking with and without the help of IM to calculate the paths. The charts incorporating IM are discussed in experiment two.

In contrast to Fig. 5 Fig. 6 presents the number of surviving units within the different scenarios. Note that the bar which represents zero surviving attacking units indicates that the attacker lost the game.

Observations: In the following, the different settings and their influence on the resulting movement behavior are described.

Without flocking. Fast units break away from the slower ones and the group splits up. On the open-ranged map, the units reach the two hostile turrets with long delays, so that the turrets have to fight only a few units at the same time. In most cases, the turrets are able to shot each of these units before the next ones arrive and thus defeat the group. As a result of this separated movement behavior only approx. 8% of the games are won (cf. Fig. 5). Nevertheless, these games are usually won by a narrow margin, only a few units survive (cf. Fig. 6).

On the second map, the calculated path for every unit passes through the range of the eight defense towers located in the middle of the map. So especially the slow units get hurt by these towers. Since the units are killed or at least

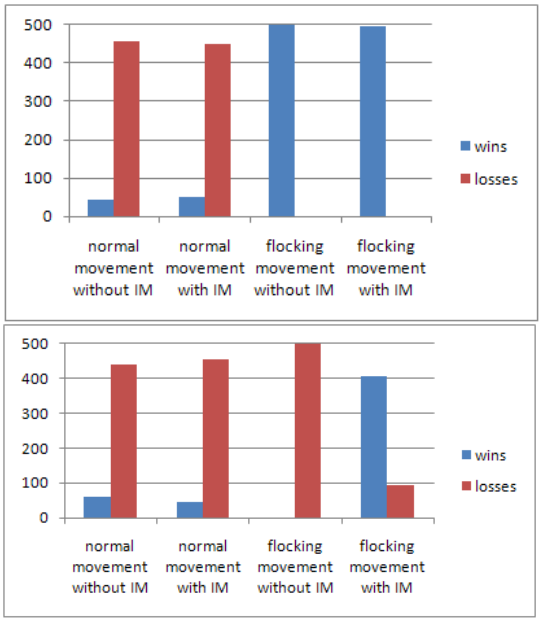

Fig. 5. Bar chart of four different moving strategies depicting the number of victories and defeats on the first map (above) and the second map (below). All combinations of normal and flocking movement with and without incorporation of IM are presented.

weakened on their way to the two towers on the right, their combat strength is even worse and they hardly win in this situation. Nevertheless, the winning ratio is roughly the same compared to the one on map 1 because the damage caused by the eight towers does not really affect the already bad results. The faster units are hardly afflicted by the towers because these are too fast to be injured. Only initiates are killed in most games because they move too slowly (cf. Fig. 5).

Using flocking. The units stick together by adapting their movement to the speed of the slowest group member. So in the first map, the units of the group reach the two defensive towers at the same time. As the group outnumbers the two towers, these are hardly able to defeat the group. As a result, most games are won by the flock (approx. 100%, cf. Fig. 5). In most of the games, four units survive the fight as can be seen in figure 6. This chart strongly indicates that the flock dominates the two defensive towers.

On the second map, flocking causes a significant drawback. The flock sticks together but moves rather slowly due to the slow units. The path of the flock leads through the range of the eight towers in the middle of the map. Compared to the results without flocking, the flock is under attack for a longer time and many units are injured or killed. None of the games is won (cf. Fig. 5).

Discussion: The bad results for the normal game behavior stem from the different speeds of the considered units. Here, these units reach their common target one after another, which enables the towers to kill them easily. Incorporating flocking, all units stuck together in a group and reached the target at the same time. In this situation, the flock is strong enough to destroy the towers.

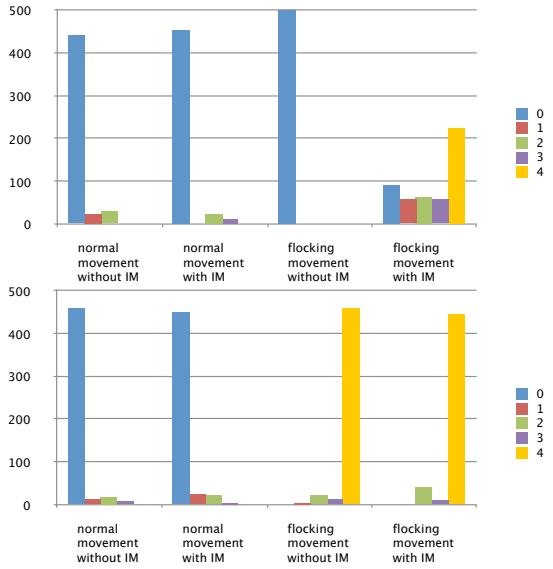

Fig. 6. Bar chart of the number of surviving units on the first map (above) and the second map (below) for four different strategies with 500 runs each. Again, all combinations of normal and flocking movement with and without incorporation of IM are presented.

Due to flocking, the speed of faster units is decreased. The whole group adapts the speed of the slowest member in the flock here. This way, the flock reacts slower and the duration of remaining in critical situation increases. This causes fatal drawbacks in special situations as can be seen on map 2. The winning probability applying flocking here is 0%, although the winning probability on the first map is about 100%. Here, in most case, 4 of the 6 units survive the attack as indicated by Fig. 6. This underlines the need to improve the flocking method.

B. Experiment 2: Properties of IM in combination with Flocking

Research question: Because of the dramatic drawback of incorporating flocking on map 2, we investigate the interaction of flocking and the IM technique on the two maps. We try to observe how the information from the IM affects the path-finding and the group's behavior for the flock and the normal movement.

Preexperimental planning: Setups have been developed depicting typical situations for RTS games. These demonstrate the improvement based on IM in situations, where flocking fails, i.e. the first experiment. Therefore, the first experiments may be seen as the preexperimental planning for experiment two.

Setup: The setup is quite similar to the first one again aside from using the modification of the A*-algorithm by the IM to calculate the moving path.

Results/Visualization: Here, we refer to the figures 5 and 6 described in the corresponding part for experiment one. The results have already been presented, but not discussed there.

Observations: The activation of IM for calculating the units' paths can well be recognised within some of the results received. Other results do not show a meaningful difference whether or not the IM is invoked.

While differences for the number of wins and defeats cannot be recognized for the normal movement on both maps, considerable differences can be detected having flocking activated for map two (cf. Fig. 5). Here, the results change from 100% losses to approx. 80% wins. Invoking IM obviously does not influence the results in a positive way on the first map because the attacking units have already won all games.

A similar behavior can be observed with respect to the number of losses and surviving units presented in Fig. 6. While activating the use of the IM does not affect the behavior on map one at all and map two with normal movement, it significantly changes the results on map two featuring flocking. Here, the situation changes from 100% defeats for not considering the IM to less than 20% defeats and more than 40% wins with more than four surviving units considering the IM. Even more evidence for the influence of IM in particular in this situation can be found in Fig. 7. The mean HP values for the attacking team overtakes this value for the rival team, while no value can be measured at all not having the paths calculated with the help of IM.

Discussion: The group behavior in the first map is not affected by the IM path-finding since there are no additional units whose influence could affect the path-finding. The group supported by IM path-finding without flocking still loses against the two turrets for the same reasons like in experiment 1 (cf. Fig. 6). The flock combined with the IM path-finding still wins just like in the first experiment with minor losses of hit-points (cf. Fig. 7). On the second map, the behavior of the group changes significantly with the IM path-finding activated.

Without flocking, using IM path-finding. On the second map, the calculation of the path is affected by the input from the IM. Each unit calculates a different path with respect to the influence of the eight towers. In general, the path length is increased, which leads to the attacking units splitting up even more than in experiment one. Based on this, the attacked towers have more time to fight arriving units and the winning probability of the attacking units decreases.

In contrast to the behavior in experiment 1, the attacking units do not lose any hit-points on their way to their target. This is due to considering the IM for calculating the path. But this does not have a major influence on the final result, because the attacking units are defeated in any case (cf. Fig. 5).

Using flocking, using IM path-finding. The combination of the techniques is the most successful alternative for the second map while not losing performance for the first one. Here, flocking makes the group stick together and the IM takes care for secure paths around opponent units. Even passing the eight towers in map two does not effect the flock in a way that it regularly loses the final combat. This is due to the path calculated based on the IM, heading through an area, where the units of the flock cannot be injured from the

towers.

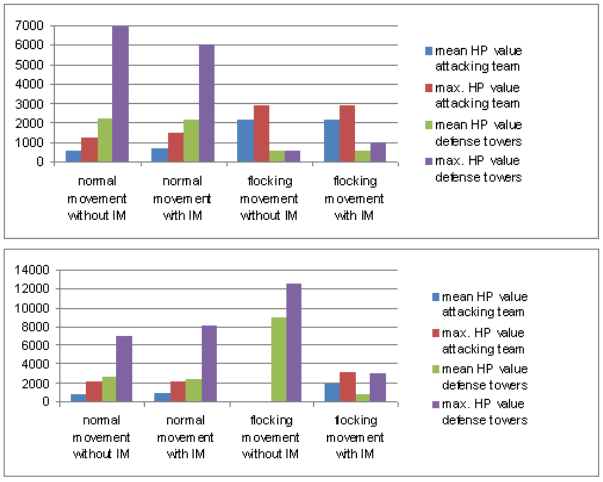

Fig. 7. Bar chart of the mean and the maximal HP value of the attacking team and the defense towers on the first map (above) and the second map (below) for four different strategies. Again, the four combinations of normal and flocking movement with and without incorporation of IM form the different strategies presented. The starting HP value of the attacking team is 5400 on map 1 and 5950 on map 2. The defense towers own 14000 HP together (7000 each).

Compared to the unsupported group in the first map no difference has been expected since there are no other units whose influence could be stored in the IM.

The combination of the flocking method and the path calculation with the modified A*-algorithm is a great improvement in the shown situation. The profoundly drawbacks of flocking could be invalidated with the help of the IM path-finding. This possibility of an intelligent moving behavior represents a real alternative to the normal movement in RTS games.

C. Experiment 3: Properties of IM and Combination with Flocking and the influence of the SEW

Research question: How does the parameter SEW influence the attack behavior of a selected group of units? Is it possible to achieve a more defensive or aggressive behavior through using different values of the SEW?

Preexperimental planning: Within the first two experiments, the combination of flocking and IM has proven to be an excellent movement behavior in RTS games. For the scaling of the combat strength, a scaling factor was defined, which has been kept constant until now. This parameter will now be varied in order to check how it influences the group's aggressiveness.

Task: Check whether the behavior of a flock can be controlled via the parameter SEW. How do the units behave for different settings of SEW and can the requested control of the aggressiveness be provided by the variation of it?

Setup: The third scenario should evaluate the influence of the SEW on the attack behavior of a group of units using the flocking movement and the IM for calculating the path. The

TABLE I

PERCENTAGE OF GAMES WON WITH DIFFERENT NUMBERS OF SURVIVING UNITS FOR THREE VALUES OF SEW.

	surviving units							
SEW value	7	6	5	4	3	2	1	0
4	0	0	0	5.8	27.8	17.4	10.8	38.2
8	0	0	100	0	0	0	0	0
12	100	0	0	0	0	0	0	0

TABLE II

PERCENTAGE OF GAMES WITH SURVIVING DEFENSE TOWERS (SDT).

SEW value	sdt team 1				sdt team 2	
	3	2	1	0	1	0
4	0	16.8	21.4	61.8	100	0
8	100	0	0	0	0	100
12	100	0	0	0	100	0

SEW should control the aggressiveness that can be observed in either attacking stronger enemy units or passing around these with possibly attacking weaker ones. A third scenario is defined that differs from the prior ones in two enemy positions on the way to the group's target (cf. Fig. 4). The first enemy position consists of three defensive towers (same values as in experiment 1 and 2). The second one is a single but stronger defensive tower featuring a high combat strength with 13000 HP. The distance between the two positions is big enough so that their influences do not overlap each other. The group is the same one like in scenario 1 with an additional archmage. This allows for both parties to win the combat with roughly the same probability.

Results/Visualization: The attacking group starts with seven units within all 500 experiments executed. The values in Tab. I are calculated for the three values of SEW given in the left column. The numbers in the table display the percentage of games in which the number of units in the second row reached the target point. For example, with a SEW value of 4, in 27.8% of the games, 3 attacking units reach the target point.

Tab. II depicts comparable values for the two groups of defensive towers in experiment 3. The table presents the results for the group of three towers in the left, indicating in which percentage of games, none, one, two, or all three towers survive the attack. The results for the stand-alone, stronger tower are given in the right part of the table.

Observations: Tab. I shows that all attacking units survive using a SEW of 12. Using a SEW value of 8 exactly 2 units are killed in each game played. Playing with a SEW value of 4 results in losing approx. 40% of the played games (0 attacking units left).

Concerning the attacked towers, all of these survive by using a SEW value of 12. Playing with a value of 8, only the second tower is destroyed in every game, while the group of the three weaker towers survive. SEW value 4 results in the stand-alone tower surviving every game again. Two out of three towers in the first location survive in every sixth game (16.8%). Moreover, in about 60% of the games the whole group of towers is destroyed by the attacking units.

Discussion: Depending on the different values of the SEW we have to discuss three different situations. A high SEW value causes a more defensive and cautious behavior of the group, what is reflected in all attacking units as well as all towers surviving in this situation. The main aim here is to achieve the target point and not to attack any enemy position. The high SEW provides significant compensation for the cumulated group power. Note that the high value of 12 does not guarantee a complete defensive behavior with other compositions of groups. An extreme powerful group might still attack a weak enemy position in other scenarios.

A SEW value of 8 shows a slightly different behavior. The group does not take the risk to attack the first strong enemy position, because its cumulated power is still underestimated. Instead, the group carefully passes around the strong position, and attacks the single turret, which has a higher HP but is not assisted by other towers. Slightly surprising is the fact that exactly two units die in all 500 recorded games (cf. Tab. I). But the reason can be found in having a closer look at the composition of the group. The tower kills the units with less HP first and therefore the two battlemages will not survive the battle. This provides a little variance in time, where the other units are able to destroy the tower.

The group starting with a SEW value of 4 attacks the first emplacement directly. Its synergy effect is now strong enough and the group acts in that aggressive behavior. This way, enemies with comparable HP value will be attacked. The results show that at least one out of three towers is destroyed by the group (cf. Tab. II). Nevertheless, a big variance remains caused by the diversity of the damage. All possibilities – between two towers and four attacking units survive – occur.

VI. Conclusions and Outlook

As shown in our experiments, flocking in combination with IM path-finding improves the performance of the units in every game situation. Especially the disadvantages in certain situations of each technique are overcome by their combination. Although this result is great, there are some points for further studies. On the one hand, we can improve the calculation of the influence by regarding more characteristics of the units and the game mechanics. On the other hand, it could be possible to enhance the flocking parameters, e.g. by adapting the parameters using an EA or by changing them due to the current game situation. The flocking group could employ a more scattered spreading if an enemy is able to attack the flock with burst damage for example. Alternatively, the IM path-finding could calculate the paths with a certain buffer to the enemy territory so that a scattered flock is completely safe and not only parts of it.

Furthermore, an advanced strategy game player sometimes wants to fool his enemies by sacrificing one unit to prevent several other units from being attacked. If our actual implementation is used in RTS games now, this strategy will not work in some cases yet, because the path-finding combined with Influence Maps prevents units from committing suicide when the enemy territory is large and of strong influence. On the other hand, the famous problem of the workers being killed by an enemy on the way to the resources they have to collect belongs to the past because we provide a simple remedy for the workers to follow a save path around the enemy.

Acknowledgements

This work was kindly supported by the *Deutsche Forschungsgemeinschaft (DFG)* and the *Federal Ministry of Economics and Technology (BMWi)*. We thank the student project group 511 for developing the basics of this work.

References

[1] M. Deloura, *Game Programming Gems*. Rockland, MA, USA: Charles River Media, Inc., 2000.

[2] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics SSC4*, pp. 100–107, 1968.

[3] GNU Software Foundation, "Gnu general public license, version 2," June 2001, november 18, 2007. [Online]. Available: http://www.gnu.org/licenses/gpl-2.0.html

[4] M. Figueroa, J. González, T. Fernández, F. Menéndez, and M. Caruncho, "Glest, a free 3d real time strategy game," 2007, december 16, 2007. [Online]. Available: http://www.glest.org

[5] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM, 1987, pp. 25–34.

[6] P. Tozour, "Influence mapping." in *Game Programming Gems 2*. Ed. M. Deloura. Hingham, MA: Charles River Media, 2001, pp. 287–297.

[7] Z. Shen and S. Zhou, "Behavior representation and simulation for military operations on urbanized terrain," *Simulation*, vol. 82, no. 9, pp. 593–607, 2006.

[8] A. Davison, *Killer Game Programming in Java*. O'Reilly Media, Inc., 2005.

[9] H. Danielsiek, C. Eichhorn, T. Hein, E. Kurtić, G. Neugebauer, N. Piatkowski, J. Quadflieg, S. Schnelker, R. Stüer, A. Thom, and S. Wessing, "Final report of PG 511," Technische Universität Dortmund, Technical Report of the Collaborative Research Centre 531 Computational Intelligence Reihe CI 252/08, SFB 531, 2008. [Online]. Available: http://www.ciingames.de/publications/pg511finalreport.pdf

[10] A. Kirmse, *Game Programming Gems 4*, ser. Interdisciplinary Systems Research. Basel: B&T, 2004, vol. 26.

[11] J. Flensbak, "Flock behavior based on influence maps," 2007.

[12] C. Miles and S. J. Louis, "Towards the co-evolution of influence map tree based strategy game players," in *CIG*, Reno/Lake Tahoe, Nevada, USA, 2006, pp. 75–82.

[13] ——, "Co-evolving real-time strategy game playing influence map trees with genetic algorithms." in *Proceedings of the Congress on Evolutionary Computation*. Vancouver, Canada: IEEE Press., 2006.

[14] M. Preuss, "Reporting on experiments in evolutionary computation," Dortmund University, Tech. Rep. Reihe CI 221/07, SFB 531, 2007. [Online]. Available: sfbci.uni-dortmund.de/Publications/Reference/Downloads/22107.pdf

[15] T. Bartz-Beielstein, *Experimental Research in Evolutionary Computation – The New Experimentalism*, ser. Natural Computing Series. Springer, Berlin, 2006.