



2.1 Company of Heroes Squad Formations Explained

Chris Journey—Kaos Studios

journey@gmail.com

The squad formations in *Company of Heroes* are designed to provide very complex and believable motion for groups of soldiers moving through a highly destructible environment. This article explains how accomplished in enough detail that you can implement a similar system or just steal a few of the tricks for your next game. The techniques primarily apply to games with an overhead view where the positional movement of units is of high importance.

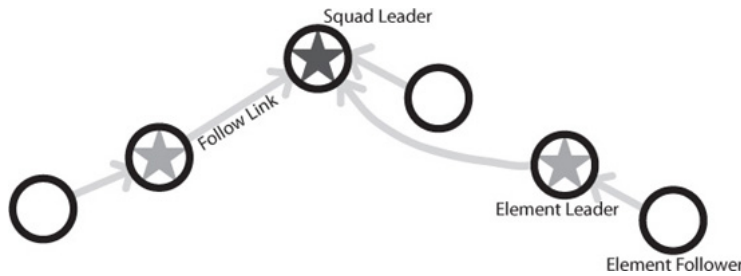
Squad Makeup

In *Company of Heroes*, infantry units are grouped together in squads. Players are only able to issue orders to squads, so it is up to the squad AI to make the units look smart while executing orders. Squads are divided into three elements: core, left flank, and right flank. Each element has a leader, and the leader of the core element is the leader of the squad. These roles are assigned to soldiers and updated whenever one unit in the squad is reinforced. The assignments are made based on an extensive set of game-specific rules, for example:

- Put squad leaders in the core.
- Allies put heavy weapons in the flanks; Axis put heavy weapons in the core.
- Put an even number of soldiers in the left and right flanks.

These assignment changes are “stable,” meaning that the system does the minimum number of swaps necessary to obey the assignment rules. Extra swaps are avoided because it looks awkward when a soldier randomly runs from the left side of a squad to the right when a reinforcement arrives. The hierarchical structure of the squads is shown in [Figure 2.1.1](#).

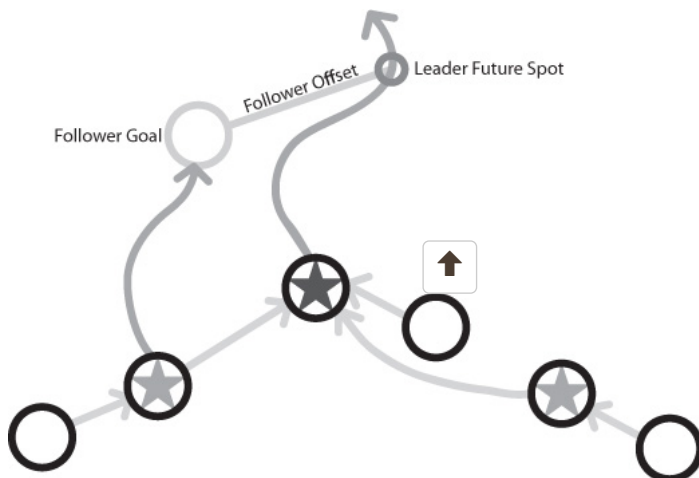
FIGURE 2.1.1 The formation layout is hierarchical. At the top of the hierarchy is the squad leader who has an immediate follower represented by the hollow circle and two element leader followers that are represented by the circles filled with gray stars. Each element leader has a single element follower, which is represented by a hollow circle.



Move Out

Now, let's move on to the basics of movement. When a move order is issued to a squad, its leader receives an order to move all the way to the move order's goal. He immediately computes a path and starts moving. To generate goals for other squad members, we predict the future position of the leader along his path roughly two seconds ahead. We then take an offset from the leader's future position and orient it using the leader's future heading to generate goals for element leaders. The formation is hierarchical, so element followers make predictions for their element leaders' positions and move to offsets from there, as shown in [Figure 2.1.2](#).

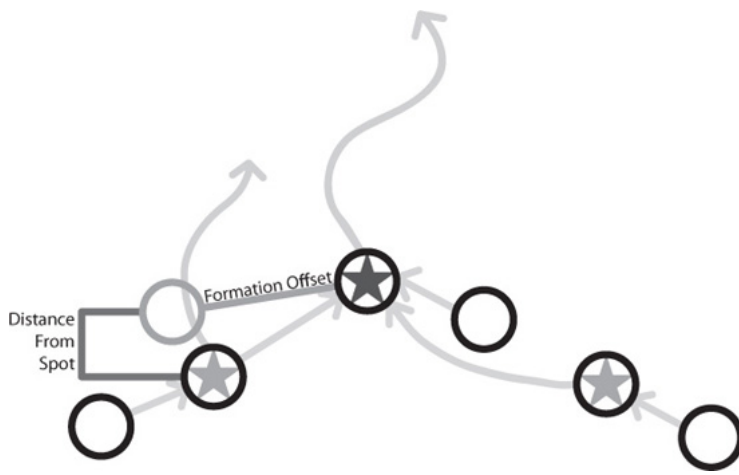
FIGURE 2.1.2 Followers move to an offset from a point in the leader's future. The goal for the left-hand element leader is computed using the predicted future position of the squad leader and the offset between the element leader and the squad leader rotated according to the predicted heading of the squad leader.



Now the units are moving, but there is a problem. Without some control of the units' speeds, the formation will quickly lose its shape as some units fall behind and others pull ahead. To overcome this problem, we introduce a speed modifier that is used to adjust the speed of movement of individual units. To get the speed modifier, we take the same formation offset we used to calculate the follower's goal, rotate it according to the leader's current heading, and apply it relative to the leader's current position instead of his future position.

If the follower is ahead of the offset position in the axis of the leader's motion, we slow him down proportional to the distance he's ahead; if the follower is behind, we speed him up proportional to the distance he's behind (see [Figure 2.1.3](#)). We don't really care if he gets out of place in the axis perpendicular to the direction of motion of the leader because the follower's goal will bring him back in line eventually.

FIGURE 2.1.3 The distance of a squad member from his current offset position in the current direction of motion of the squad leader is used to adjust his speed.



Softening the Shape

Now the units are moving out, but they will occasionally run into problems. Sometimes when using fixed offsets, the followers' goals will be inside or on the far side of obstacles, which is particularly problematic when the pathfinding distances to those goals are significantly larger than the straight-line distances.

To resolve this, the system never uses offsets directly. Instead, for the goal offset points described previously, a cheap A* pathfind with a search step limit of around 50–100 nodes is run from the leader's future position to the follower's ideal future offset position. Whether or not the search succeeds, the offset that will actually be used by the follower will be the point touched by the search that was nearest to the ideal offset position. The net effect of this approach is to make the formation squeeze in organically at chokepoints and route around small obstacles that don't divert too much from the leader's path. The behavior of this technique is shown in two different scenarios in Figures 2.1.4 and 2.1.5.

FIGURE 2.1.4 The actual follower offset goal is picked by pathing from the leader's future spot to the ideal offset position. The nearest point found by the pathfinding algorithm to the ideal offset position is used as the actual offset position.

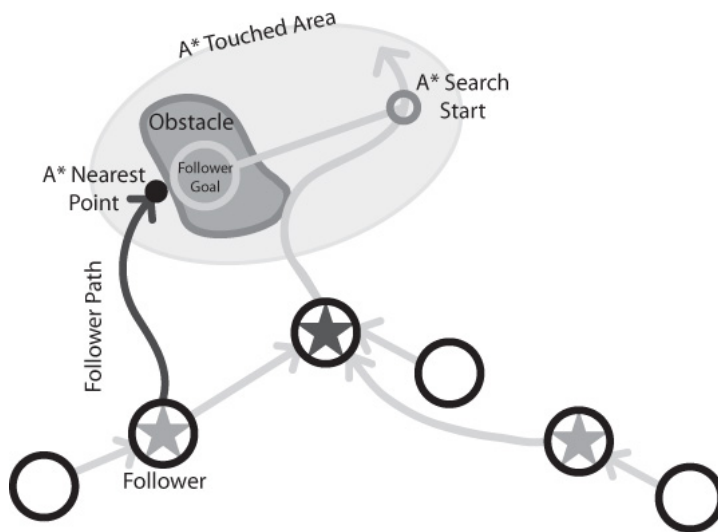
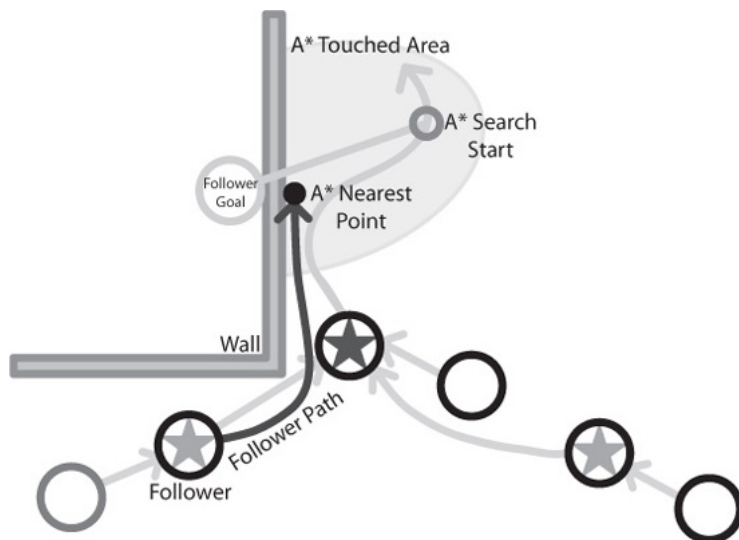


FIGURE 2.1.5 Using a search to choose a follower's goal produces good results in nearly every scenario.



The formation is now handling movement amid obstacles and chokepoints, but it will look a bit robotic because the combination of the fixed offsets, speed control, and obstacle avoidance code is too good at keeping units in their formation-mandated spots. To counteract this, we have each unit store a personal drift variable that is added to its formation offset for all the formation calculations. Each tick of the simulation this drift value floats around via a random offset within a maximum radius.

All the systems that have been described for controlling the movements of individual units in formation have one major drawback: performance. The cost of the pathfind searches that are used to calculate goal positions are inexpensive because they can be accomplished with a very low search step limit, however, pathfinding from the followers' current positions to the followers' goals is slightly less constrained. Because we are giving slightly different move orders to each unit each tick, we are asking each unit to repath each tick.

Even though the pathfinding is only performed over relatively short distances, it does add up. To counteract this, when the system calculates the final goal position and speed modifier for a unit, it compares these to the unit's current movement. If the difference is within a small delta, then the new order is not given because it would have little impact. By tuning this delta, we can have our follower units repath only every half-second to three-quarters second during formation movement, which is enough to keep their pathfinding from showing up significantly in the profile stats for the game.

Formation Context

Not all formations are appropriate for all situations. When in a wide-open field, for example, a broad wedge shape looks very natural, but when navigating along cramped city streets, the same shape seems unnatural. To fix this, we decide which of a set of formations to use based on the terrain the squad leader is passing over. Specifically, we used a wide wedge for open areas, a tight wedge for confined areas, and a staggered column for roads.

Leapfrogging

Now that the soldiers are moving in formation, it would be nice if they could leave the formation occasionally to leapfrog to cover or some other interesting feature of the game world. Specifically, *leapfrogging* means that one of the elements is going to leave his normal place in the formation, run to some interesting spot in the world, hold for some amount of time, and then resume moving in formation. Such interesting spots are selected by searching for points of interest in the area of a follower's goal. When we find something interesting, we change the mode of movement of the element to have it follow the leapfrogging pattern. Only element leaders perform this behavior, and element followers are simply along for the ride. Occasionally, a leapfrog is performed even when there is no interesting feature, just to keep the squads looking interesting. The leapfrogging pattern is shown in Figures 2.1.6 and 2.1.7.

FIGURE 2.1.6 Followers search for points of interest by leapfrogging in the area of their goal.

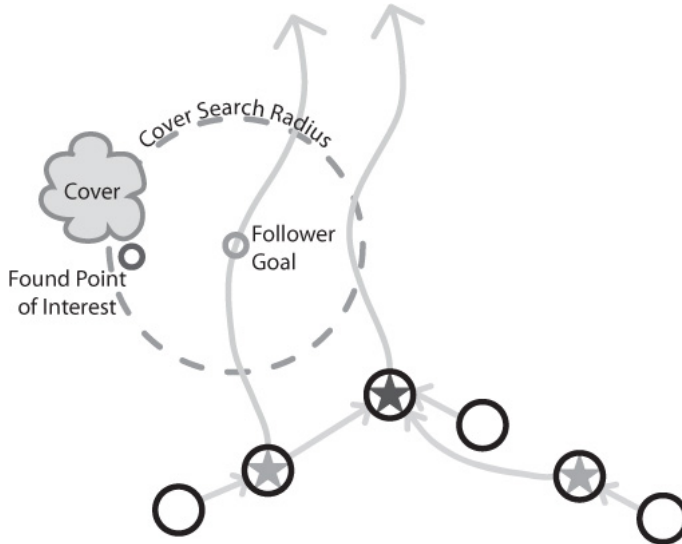
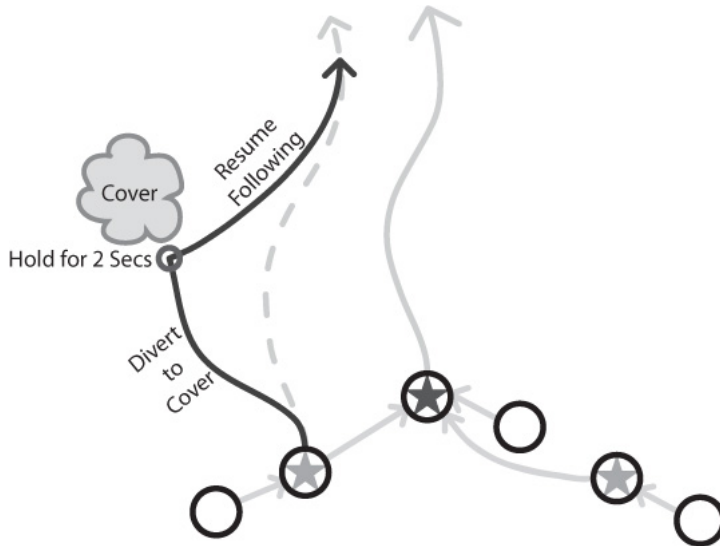
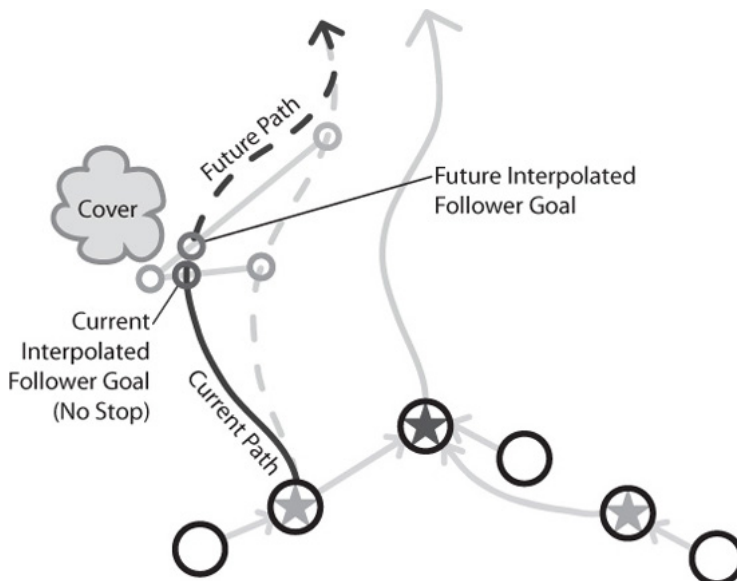


FIGURE 2.1.7 Leapfrogging units move to the point of interest, stop, and then resume following.



This system works, but in some cases, putting a hard stop in the middle of a leapfrog looks wrong, especially when the squad is in a hurry, such as when it is retreating. To fix this, we add a second type of leapfrogging called a "soft" leapfrog. In the case of a soft leapfrog, we calculate both the normal formation move order for the element leader and the leapfrog order. We then send the element leader to a point on the line between the two orders, about 80% on the side of the leapfrog order. This results in the character veering out of formation obviously toward cover and slowing a bit but never stopping. The effect is a very intelligent-looking unit who is in a hurry, but not so much as to totally ignore his own safety and his environment, as shown in Figure 2.1.8.

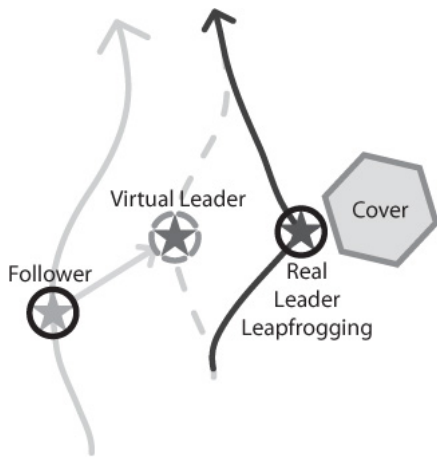
FIGURE 2.1.8 Soft leapfrogging modifies the follower's goal by pulling it in the direction of the point of interest.



Virtual Leader

Adding leapfrogging into the formation system creates a new problem. When the leader of the squad and of the core element stops heading for the goal and heads for some cover in a leapfrog, the entire squad will veer off to follow him. To fix this problem, we add the idea of a virtual leader that the formation will follow instead of the real one. Most of the time, the virtual leader is snapped to the position of the real leader, but when the squad leader leapfrogs, the virtual leader continues on the original path to the goal. When the real leader finishes leapfrogging and resumes heading to the goal, the virtual leader interpolates back and eventually snaps to the real leader. The interpolation helps prevent jerks in the motion of the units that would be caused by sudden changes in goal positions. The effect of the virtual leader is shown in Figure 2.1.9.

FIGURE 2.1.9 While the real leader is leapfrogging, followers use a virtual leader that continues on the real leader's previous path.



The reason for not using a virtual leader all the time is that the virtual leader is not a real unit dealing with all the obstacles of the dynamically changing world and can therefore get significantly ahead of, or behind, the actual units. When the leader is a significant distance from the followers, follower behavior will become less intelligent because their goals and any leapfrogging behavior will be based on objects too far in their future to appear relevant. Initially, the formations implemented for *Homeworld* had issues with this, so we avoided the problem in *Company of Heroes* by keeping the virtual leader as nonvirtual as possible.

Destination Formation

A separate formation system is used when a squad gets near its destination. This formation has the same shape and uses the same drift values as the moving formation to avoid any jarring shifts. In the separate system, each soldier marks his destination spot with a reservation to make sure that no one else is standing there when he arrives, and no one else is allowed to use that spot as a destination until he gives it up.

The user can set the facing of the formation in *Company of Heroes* by right-dragging the mouse instead of right-clicking. If he doesn't do that, we try to help him out by making the formation face any enemies in the area; if there aren't any, units simply face the average direction for the move order, defined by the vector from the squad leader's start position to the position the user clicked.

Each individual soldier in the formation does a search in the area surrounding his formation spot looking for cover to protect him from enemies that lie in the direction of the formation heading. If he finds cover, he'll go there instead of his formation spot. Just like in the moving formation, we find a spot for the leader first, then we use cheap pathfinds to validate any spot chosen by the destination formation, and we use the closest available if it isn't reachable. This keeps squad members from ending up on the wrong side of a long wall or building at the end of a move but allows them to go to the other side of small obstacles.

Handling Destruction

Fortunately, no additional code is required for this system to handle a highly dynamic and destructible environment. Because everyone but the leader is recalculating their paths every four to six ticks, any changes to the environment are immediately reflected in units' routes. As long as the leader also periodically recalculates his route to the goal, he will also react to any changes. In practice, having the leader repath every time he leapfrogged was sufficient.

The performance impact of all the repaths required by this formation system is mitigated for the followers by the fact that their paths are all very short distances and almost always direct due to the fact that their goal positions are close to a future position of the leader and hence almost always directly reachable. Color Plate 1 shows a formation from *Company of Heroes* with examples of these short paths. The performance impact of the leader repaths is mitigated by the relatively smaller number of leaders and by using hierarchical pathfinding.


Conclusion

The motion produced by the system that is described in this article played a significant part in making the soldiers in *Company of Heroes* into believable characters. Using the system, you can achieve tactically and visually interesting motion. Even if you can't use the entire system, you should be able to use the individual pieces to improve the quality of your own group movement. The parts described can be developed iteratively to build up advanced behavior gradually. After you apply them all, you will have squads that will impress your players with their incredibly tactical and life-like motion.



PREV
SECTION 2 MOVEMENT AND PATHFINDING

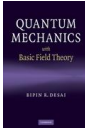
BOOK SECTION

 **Chapter 18. The Façade Pattern**
from: [C# Design Patterns: A Tutorial](#) by James W. Cooper
Released: September 2002

29 MINS
Design Patterns

ed: 2.2 Turning Spaces into Places NEXT

BOOK SECTION

 **Dirac equation in the presence of spherically symmetric potentials**
from: [Quantum Mechanics with Basic Field Theory](#) by Bipin R. Desai
Released: December 2009

12 MINS
Math & Science