How qualitative spatial reasoning can improve strategy game AIs

Kenneth D. Forbus

Qualitative Reasoning Group,
Northwestern University
1890 Maple Avenue
Evanston, IL, 60201
forbus@nwu.edu

James V. Mahoney

Xerox PARC
3333 Coyote Hill Road
Palo Alto, CA, 94301
jvmahon@parc.xerox.com

Kevin Dill

Qualitative Reasoning Group
Northwestern University
1890 Maple Avenue
Evanston, IL, 60201
kdill@cs.nwu.edu

Abstract

Spatial reasoning is a major source of difficulties for strategy game AIs. We conjecture that *qualitative spatial reasoning* techniques can help overcome these difficulties. We briefly review the relevant qualitative reasoning ideas, and outline four potential advantages of our approach. We describe two explorations in progress: How visual routines can be used to quickly compute qualitative spatial descriptions for war games, and how qualitative descriptions can help in path-finding.

Introduction

Perhaps one of the largest surprises in computing at the end of the 20th century was the extent to which entertainment became a major driving application. The power of the consumer market far outweighs the scientific or even defense markets. For example, 3D visualization hardware that would be the envy of any supercomputer center even a few years ago now is sold in discount stores, due to volume production made possible by the consumer market. Recently in the US the computer game industry reported higher gross profits than the American movie industry, a landmark indicating the widespread popularity and acceptance of computer gaming as a form of entertainment. It seems only natural, then, that computer gaming should be considered as a valid applications area for computer science research. As Laird has observed [9] artificial intelligence in particular has much to offer to computer gaming. The quality of a game's AI is now one of the leading product differentiators; poor AI can kill a game and great AI becomes an important selling point. From an AI research perspective, game-oriented research offers new, richer simulated worlds in addition to the chance to have significant impact on an important industry. After all, would you rather do your research with Blocks World, Truck World, or simple hydraulic systems, or with the simulated worlds of *Civilization*™, *Close Combat*™, or *The Sims*™? Consequently, there are now a number of efforts to build bridges between the AI research community and the computer game industry.[1]

While valuable contributions to gaming will come from many areas of AI, we believe that qualitative reasoning in particular has much to offer. For example, the conceptual understanding and explanations that QR systems can provide could lead to better opponents, advisors, and other forms of non-player characters[2], as well as technologies for rapidly building new game engines based on reusable domain theories [3]. In today's computer games, spatial reasoning is a major source of difficulties. For example, terrain is of vital importance in war games, and geography is key in *Civilization*-style empire/trading games. Creating good strategy game AIs is very difficult. Since today's strategy AI's are tightly bound to the underlying game world simulation, it is hard to start their development before the game world is up and running, and harder still to reuse the algorithms and representations in a new game, unless the underlying game engine is extremely similar to the old engine.

We conjecture that using *qualitative spatial reasoning* techniques [5,6] to build strategy AIs can help overcome these problems. We begin by briefly reviewing the relevant qualitative reasoning ideas, then outline four advantages that we believe can be obtained using this approach. After that, we describe two explorations of this idea that are in progress: How visual routines can be used to quickly compute qualitative spatial descriptions for war games, and how qualitative descriptions can help in path-finding. We close with some thoughts about next steps.

What is qualitative spatial reasoning?

Qualitative representations carve up continuous properties into conceptually meaningful units [7,8]. Qualitative

[1] AAAI for example has run several symposia on AI and computer games attended by both AI researchers and game developers.

[2] A *non-player character* (NPC) is an aspect of a program that is intended to be considered as another person by the player(s).

spatial representations carve up space into regions[2][1], based on a combination of physical constraints and task-specific constraints. In reasoning about a ball's motion, for example, identifying flat versus sloped surfaces is useful because of the different ways that colliding objects can bounce on them. In reasoning about mechanical systems, identifying ranges of angles where parts may come in contact versus ranges of angles where they may move freely is one important criterion for carving up orientation. Qualitative spatial representations for many tasks need to be firmly rooted in a quantitative, diagrammatic representation, for a variety of technical reasons [5]. For our purposes, this means that we can use quantitative information (e.g., specific coordinates of a unit, or distances between units) in qualitative spatial reasoning. For instance, qualitative spatial descriptions for trafficability support both finding routes in a general way, and carrying out time-distance estimates about travel over those routes (e.g., can they get there in time?). In other words, by identifying and using these conceptually meaningful units, reasoning strategies can achieve results that are more human-like.

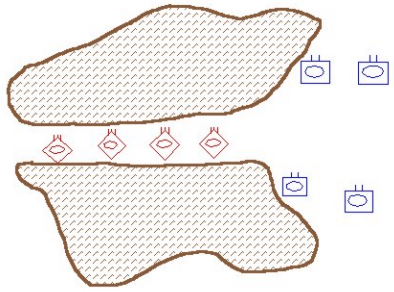


Figure 1: Red, heading east, is about to be ambushed by Blue

We believe that the spatial reasoning problems faced by strategy AIs can be better tackled using qualitative spatial representations. For example, one key spatial constraint for strategic reasoning is *trafficability*: The ability of a vehicle or unit to move across a specified piece of terrain [1]. In a *combined obstacle overlay*, US military planners divide an area of operations into *severely restricted, restricted, and unrestricted* terrain. Severely restricted means impassable, restricted means one can get through, although perhaps slowly or with damage, and unrestricted means that that unit can travel unhindered by that terrain. (We will see an example of this below.) Given a map and an echelon, the area of operations is carved up into regions of these three types. Further analysis of the unrestricted terrain identifies *mobility corridors* (i.e., paths over which sub-echelon units can move between interesting places) and *avenues of approach* (i.e., paths over which that echelon can move, based on the available mobility corridors). These spatial descriptions, and others commonly used by the US military, are good examples of qualitative spatial descriptions that have been evolved through human practice. Commanders and decision-makers use these tools constantly. If we want strategy AIs that can do as well as human players (without cheating), or that can communicate with human players in meaningful ways (e.g., as subordinate commanders or teammates or opponents), we need to exploit these and other qualitative spatial representations.

[1] We include edges and points, of course, as special cases.

Potential advantages

We believe that four advantages could be obtained by using qualitative spatial reasoning in strategy game AI's: *more expressive spatial representations*, *better communication of intent*, *better pathfinding*, *more reusable strategy libraries*. We discuss each in turn.

More expressive spatial representations

Many strategies exploit particular properties of terrain. Being able to describe spatial configurations and properties is thus essential to expressing strategies in a general way. For example, how can one characterize what are good locations for an ambush? One way of thinking about it is that an ambush is a place where you can focus far more combat power on your opponent than they can on you. One good type of ambush site, illustrated in Figure 1, is a "funnel", where your opponent is coming out of a tightly constrained space into a position where your forces are already arrayed, waiting for them. By providing more abstract descriptions of space, qualitative representations simplify recognizing and exploiting such configurations.

Better communication of intent

Commanders generally sketch when describing their plans. They specify their intent, so that units can choose their actions to be consistent with the commander's overall goals, rather than just blindly doing the concrete task assigned. They specify timing information and contingencies, so that their subordinates work as a coordinated force. Today's game interfaces, even with innovations such as waypoints and formations, are poor substitutes. Some might argue that forcing players to micromanage their forces is a good way to induce flow and thus enhance engagement in the game world. We believe instead that it leads to player frustration, especially when their units are being unrealistically stupid.

Consider a classic problem in strategy AI: Massed fires. If you assign three units to attack a specific enemy, they will chose paths to get to that enemy and then attack it. Many games are susceptible to the following problem: Suppose the quickest path to the enemy involves a tightly constrained tube, forcing your units to go down it single-file. The enemy, which would have been overwhelmed had your forces converged on it all at once, can then destroy each of them in turn as they enter the clearing (see Figure 2).

Good military planners solve this problem differently. They can specify paths that the units will take, and can specify synchronization constraints (i.e., "using these axes of advance (see Figure 3), be in position to commence the attack at 0400"). Good communication is essential to good coordination of forces. The bandwidth between human commanders and their subordinate commanders is massively higher than the bandwidth between a human player and the units under his command. Providing higher bandwidth between a human player and the game's strategy AI's could, we believe, provide a more satisfying experience.

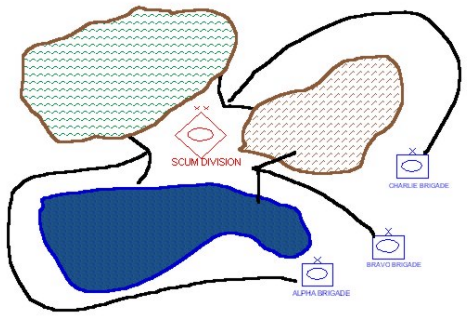


Figure 3: Enabling players to specify their intent by sketching paths should increase the sense of realism and immersion

Better pathfinding

One universal complaint from game developers during the 2000 AAAI Spring Symposium on AI and Computer Games was that path finding was still a significant problem across game genres. We believe that qualitative spatial representations can help, by reformulating space in ways that are more amenable to search.

Problems with pathfinding seem to run rampant throughout the industry, ranging from *Star Trek Armada*™ to *The Sims*. Even modern, massively successful games can run in to trouble in this area. *Diablo II*™ is a good example of an extremely popular game that has difficulty determining where the NPCs should move, and how and when they should get there. These problems are at the very least annoying, and in some instances cause the NPC or even the player to be killed.

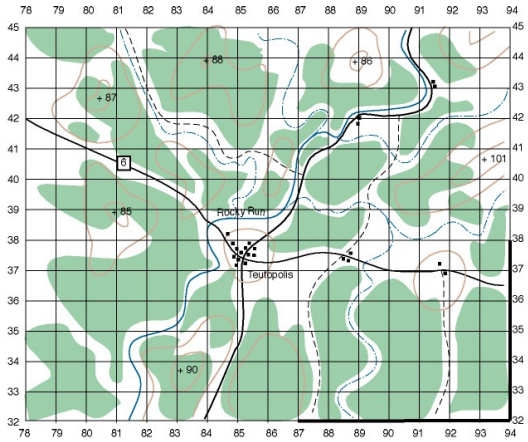
Diablo II's pathfinding is most visible when the player has an NPC follower. It appears that the game attempts to keep the follower fairly close to the player, but not right next to him. The NPC tends to wander continuously within some acceptable area. It seems that the game does not make any attempt to plan the NPC's path in advance or to consider how to get around obstacles. Instead, when player and follower are separated, the follower tries to move toward the player as directly as possible. In addition, there does not seem to be any conception of "safe" vs. "dangerous" spaces. The NPC will frequently wander into areas that the player has not yet explored.

Because the pathfinding algorithm doesn't have an understanding of obstacles and how to navigate around them, it is very easy for the follower to get "caught" in a local concavity (for instance the corner of a room you have exited). Many games have this problem for larger concavities, but it is more noticeable in *Diablo II* since almost any concavity can cause it. The solution applied by the folks at Blizzard is to magically transport the follower back to the player's vicinity if they get too far apart. Thus, when the NPC becomes stuck the player has the option of continuing without it for a while, or to go back and get it. Neither option is particularly appealing to most players, in our experience. In addition, if this mechanism were applied to a game which creates a more "realistic" world (such as *Baldur's Gate*) players might be much less willing to accept NPC's that can teleport in this fashion.

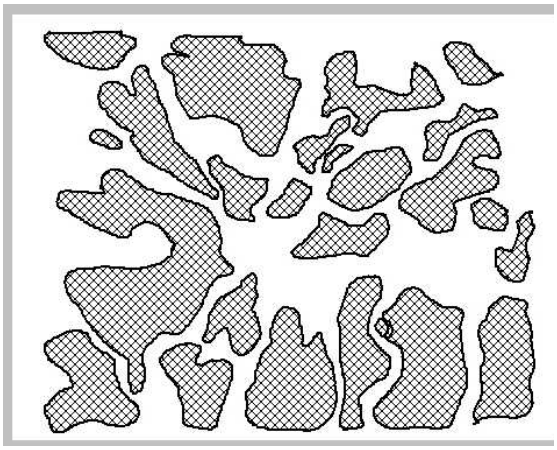
The NPC's inability to discriminate between safe and dangerous spaces can cause the player even more serious problems. When the player stands still (to heal, for example) the NPC tends to wander around him. As a result, it often becomes visible to the enemy, resulting in a fight for which the player may not be ready. There are even times when the player is fighting one group of monsters, and the NPC will wander off and draw more monsters into the fight, resulting in an attack which comes from several directions at once!

Reusable strategy libraries

Reusable components are an important way to develop new software. Reusability typically enhances robustness and reduces development time. Unfortunately, today's strategy AI's are mired in the ontological choices of the specific game engine that they are developed for. This sharply limits reuse. Consider an alternative. Qualitative spatial representations provide a layer of description above the specifics of the game engine, optimized to express the distinctions needed to carry out its reasoning. Suppose we can automatically compute qualitative spatial representations from the game engine spatial descriptions. Then we can express the strategy AI's spatial knowledge entirely in terms of the qualitative spatial representations. These representations would represent quantizations relevant for that genre of game (i.e., trafficability for war games with a land-based component), but would not rely on the specifics of the underlying game engine. As long as the "perceptual" component of the strategy AI were updated, the underlying game world could be completely reimplemented. Depending on how general the representations for other aspects of the game world are, the same strategy AI could be used with an entirely different game engine for that genre.



Snapshot 1: Map used in an on-line tactical decision game



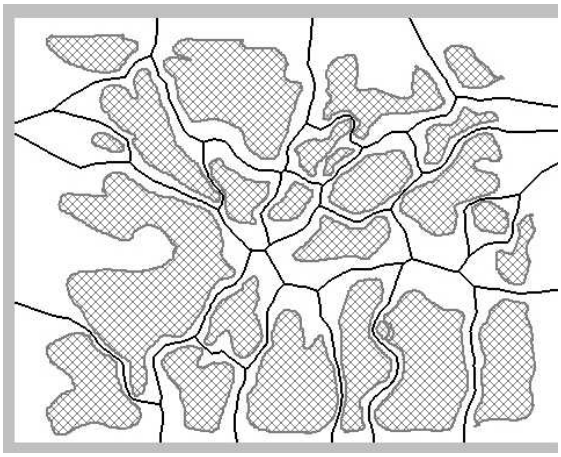
Snapshot 2: Severely restricted terrain, as drawn in the Tactical Decision Coach

Visual computation of qualitative spatial representations

Qualitative spatial descriptions are only going to be useful in games if they can be efficiently and automatically computed from the spatial descriptions in the underlying game world. Many aspects of these descriptions can be computed once for a given map, e.g., basic trafficability constraints due to terrain features. Other properties will need to be dynamically computed (i.e., changes in trafficability due to introduction of obstacles, visibility based on estimated enemy positions, fields of fire based on positions and available equipment). Figuring out the appropriate qualitative descriptions for battlespace reasoning is still very much research in progress. However, based on preliminary experiments on computing qualitative descriptions of free space, we suspect that these computations can be done practically in a gaming environment. This section summarizes those preliminary experiments.

Our work is based on Mahoney's Maps system for visual routines. Maps provides a computational model of high-level visual processing that exploits operations on bitmaps, organized into a high-level functional language that supports rapid prototyping of visual operations. We are experimenting with visual operations for extracting qualitative descriptions of free space, expressed in terms of areas and paths connecting them. We will illustrate the process via an example. Snapshot 1 shows a map used in an electronic *tactical decision game*, a computer-facilitated on-line game where players take on the roles of unit commanders, responding to orders from a designated commander. Snapshot 2 shows the severely restricted regions (for armor), given that map. (These regions were sketched in via our multimodal interface system, since the map was given to us in the form of an uninterpreted bitmap. Such regions can easily be automatically extracted from either ink-based or bitmap based models in game engines [1].)

Given the restricted regions, or 'obstacles', our goal is to extract free space regions and paths. The routines that accomplish this are composed from a small set of widely used primitive image array operations. This image-based, or iconic, approach may be contrasted with a symbolic or analytic approach, as exemplified by computational geometry techniques. One motivation for the iconic approach is simplicity of implementation: straightforward, robust algorithms for the primitives involved are well established in the computer graphics and image analysis literature and widely available. This simplicity derives from the discrete form of the input data; for example, numerical stability problems do not arise.



Snapshot 3: Voronoi diagram

The base representation for our free space analysis is the Voronoi diagram of the set of obstacle regions. In this

discrete context, by the Voronoi diagram we mean the set of pixels equidistant from boundary pixels of two distinct obstacle regions (see Snapshot 3). It consists of a set of curves meeting at junction points. Free space regions are defined with respect to the junction points, essentially by growing outwards from them toward the obstacles. In a similar way, free space paths are defined in relation to those curve points that are not subsumed by the free space regions. The results of these computations are shown in Snapshot 4. The regions and paths can be extracted without the use of any free parameters.

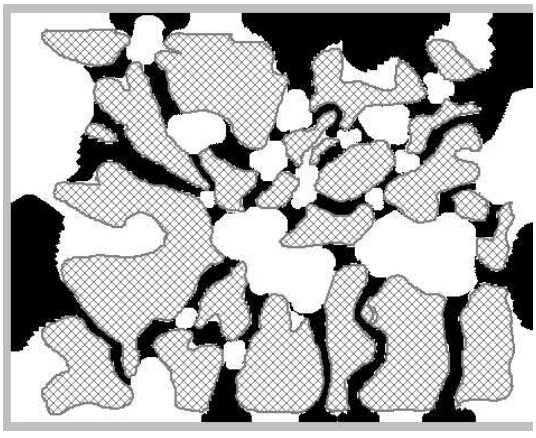
The primitive image operations required for this process are connected component labeling; the Euclidean distance and nearest feature transforms; binary edge and junction detection within a three-by-three neighborhood; and a handful of point-wise comparison and set operations. All of these operations take integer or binary array arguments and produce new array results. All array arguments and results are of equal dimensions. The primitives all run in time linear in the number of image pixels.

Connected component labeling assigns to each foreground pixel *p* the unique integer identifier of the connected component containing *p*. Background pixels are assigned a null value. This operation applied to binary image array *a* is denoted *labelcc(a)*.

The distance transform assigns to each background pixel the distance to the Euclidean nearest neighboring foreground pixel. Foreground pixels are assigned a null value. (A detailed description of a very efficient implementation is given in [11].) This operation applied to binary image array *a* is denoted *dist(a).*

The nearest feature transform [14], a minor augmentation of the distance transform, assigns to each background pixel a pointer to the neighboring foreground pixel. This operation applied to binary image array *a* is denoted *link(a)*. To make use of the feature transform, we define a *read* operation that for each pixel p fetches the value, in a given data array *d*, associated with *p*'s neighbor as defined in a given pointer array *l*; it is denoted *read(d, l).*

The edge labeling operation simply marks those pixels that have any unequal four-neighbor. This operation applied to integer image array *a* is denoted *edge(a)*. The junction labeling operation marks those pixels having connectivity number greater than two; it is denoted *junction(a)*. The connectivity number of a pixel *p* is defined as the number of background to foreground transitions in the border of the local neighborhood of *p*. (If junction(a) is defined on a three-by-three neighborhood, it is necessary to ensure that 'a' is thin, e.g., by first applying a standard thinning operator.)



Snapshot 4: Free space regions and corridors

We will make use of the following point-wise operators. *lt(a, b)*, marks locations at which the value of integer array *a* is less than the corresponding value of integer array *b*. *leq(a, b)* is defined similarly. The operation *gate(a, b)* is given an integer array *a* and a binary array *b*: it selects pixel values of *a* that are at foreground locations of *b*. (I.e., a pixel in the result is assigned the value of the corresponding pixel in *a* if the corresponding pixel of *b* is foreground; otherwise it is given the null value.) The set difference operation *andNot(a, b)* is given two binary arrays: it marks those locations that are foreground in *a* but not foreground in *b*.

Given these primitives, the Voronoi diagram of the obstacle regions is defined as follows:

```
VDiag(a) = edge(read(labelcc(a), link(a)));
```

where the foreground regions of binary array *a* constitute the obstacle regions.

The cores of the free space regions are then given by those locations that are closer to a junction point of the Voronoi diagram than to a boundary point of an obstacle region:

```
fsRegionCores(a) = lt(dist(junction(VDiag(a))),dist(a));
```

The complete free space regions are now given by extending these core regions to the obstacles. This is done by marking each location that is no further from a neighboring boundary pixel *p* of a core region *R* than *p* itself is from the defining junction point *j* of *R*.

```
Let j = junction(VDiag(a));

Let c = fsRegionCores(a);

fsRegions(a) = leq(dist(c), read(gate(dist(j), c)));
```

The free space paths are defined by a very similar process, but in relation to those curve points that are not subsumed by the free space regions:

```
fsSegments(a) = andNot(VDiag(a), fsRegions(a));
```

The analogous operations for extracting the free space paths are then as follows:

```
fsPathCores(a) = lt(dist(fsSegments(a)), dist(a));

Let s = fsSegments(a);
```

Let c = fsPathCores(a);

fsPaths(a) = leq(dist(c), read(gate(dist(s), c)));

There is a related, extensive literature on the find-path problem in the plane, which specifically addresses the problem of maneuvering a given 'target' shape through a set of obstacle shapes, usually for robot path planning. Often, these techniques employ polygonal or generalized cylinder approximations to the target and obstacle shapes, and employ a range of sophisticated computational geometric and topological methods (e.g., see [12,1]), including, often, explicit Voronoi representation (e.g., see [13]).

The technique presented here is very simple in comparison these methods, and is perhaps novel in its simplicity. It is designed to extract free space regions and paths in the limited case in which the obstacle shapes are given in discrete form (as foreground regions in an image array), doing so through the fewest and simplest possible off-the-shelf operations.

The resulting description of free space and corridors is still subject to refinement – for example, there are edge effects where the distinctions between regions and paths seem visually unnatural. However, we do find it encouraging, given that we have only just started exploring this space of algorithms. From a practical standpoint, it is important to note that these computations only need to be done once per map. The entire sequence of computations described here took six seconds on a mid-range machine, using a Java-based general-purpose implementation. We believe that this is already fast enough for most purposes, and that substantial further performance optimizations could be done if necessary.

Using qualitative spatial representations in pathfinding

We are exploring two uses of qualitative reasoning in pathfinding. First, we believe that we can improve the performance of pathfinding algorithms by better describing the space in which pathfinding occurs. Second, we would like to be able to add soft constraints to pathfinding.

More efficient pathfinding

Effective pathfinding requires solving two problems: Describing the space and searching that description for a suitable path. Much effort has been put into developing search techniques such as A*. Much less effort has been put into creating better descriptions of space to search through.

Historically, there are two schools of thought about how to describe space for purposes of path planning: *cell decomposition* and *skeletonization* [10,15]. Cell decomposition, in its simplest form, simply involves overlaying the world with a grid. Each square in the grid is marked as trafficable or non-trafficable, and then path planning is done on the grid. There is, obviously, a tradeoff between grid resolution and search speed. More advanced methods have been proposed that can deal with shaping the edges of cells to the edges of obstacles, or combining groups of small cells into a single large cell. Skeletonization attempts to reduce the world to a 1 dimensional skeleton along which units may move. Pathfinding is then reduced to finding a path to the skeleton, finding a way to move along the skeleton to a point near the goal, and then finding a way to move from the skeleton to the goal.

The method we are pursuing involves a hybrid of these two techniques. Using the qualitative spatial representations described above, we divide space into *tubes* (which can be used to get through constricted areas) and *open areas* (convex spaces in which we can move directly from any edge point to any other edge point). Concave areas can be handled either by subdividing them into smaller convex areas, or by moving along the wall as needed to get around the concavities. We believe that dividing space in this way will result in significantly more concise descriptions of space, allowing our search routines to more quickly produce a good path. Additionally, the resulting path should look fairly natural, eliminating the need to use post-processing techniques for smoothing. We have implemented a bi-directional A* pathfinding system that uses the place vocabulary computed by maps system, and are currently testing it on a variety of examples.

Soft constraints in pathfinding

Military planners consider at least three distinct sources of constraints when deciding how to move between two points. These are *trafficability*, *visibility*, and *fields of fire*. Trafficability, as noted earlier, concerns the ease with which a unit can move along the path. Visibility describes how easily the unit can be seen by enemy units, and how well it can see the enemy (often, but not always, there is a tradeoff between the two aspects of visibility). Fields of fire are the areas in which the unit may be fired upon by enemy units. Most computer games appear to consider only trafficability. In the *Diablo II* example given above, for instance, the NPC's failure to consider its visibility is often significantly detrimental to the player.

If we can qualitatively divide space based on these constraints, we should be able to devise AIs which act far more cleverly. They will be able to exploit terrain to hide from the player, figure out weak spots in the player's defenses, and generally give a better game, without cheating, than today's game AI's. Furthermore, we can add an element of personality to our NPCs by setting their utility for these constraints differently (or we can allow the player to set this for NPCs they control).

Discussion

In this paper we have described work in progress on using qualitative spatial reasoning to improve AIs for strategy games. The central idea is that, by using more human-like representations of space, we can build strategy AIs that can use terrain better in achieving their goals, take orders better (for subordinate commanders), and find their way around better. Moreover, by decoupling the representation of space in the strategy AI from the implementation of space in the game engine, we could take a large step towards making reusable strategy AIs, driving down development costs while improving their subtlety.

It should be noted that there are a number of other important spatial reasoning problems faced by computer game designers in addition to pathfinding. In *Civilization*-style explore/trading/empire-building games, for example, good strategies for exploring the simulated world and for designing the spatial layout of cities and transportation networks are two important problems for AI designers. There are also ample opportunities for qualitative decision-making, e.g., computer players must make classic guns/butter/research tradeoffs in their play. Formalizing such strategies could enable AI designers to more easily craft opponents with specific personalities and levels of skill.

While we are working on applying these ideas in computer games, the main driver of our qualitative battlespace representation and reasoning work is the creation of intelligent software to support US military needs. For example, we are creating a *Tactical Decision Coach* that uses analogical reasoning to critique student battle plans, based on comparisons with expert solutions to the same tactics problems. This coach is a first step towards software assistants and advisors that could be part of future command post software. By working closely with military experts, we aim to develop systems that will be capable collaborators with human commanders. However, we also expect that what we learn about strategy and tactics will be applicable in turn to strategy games. It is no accident that military thinkers tend to be gamers (although often not computer gamers); simulation games provide an invaluable forum for experimentation and learning.

We are currently proceeding in this research in several ways. First, we are continuing to experiment with the MAPS system, to create qualitative spatial descriptions to support path-finding and deeper terrain analysis (e.g., spotting good places for ambushes). Second, working with Cycorp, we are expanding representations of human activities so that effects of plans can be more realistically considered (e.g., if you force troops to march harder they will get tired sooner). Third, we are fielding-testing alpha versions of our software to US military personnel and military experts, using their formative feedback to help drive our development process.

As we improve our representations, we need to move beyond tactical decision problems and into interleaved planning and execution. One roadblock in doing this is the lack of good military simulations with APIs that would enable us to create software that could play on an equal footing with humans. (Existing military simulations, such as MODSAF, require human operators and substantial computing resources for anything but the simplest scenarios.) We are currently exploring collaborations with game companies to help them develop such APIs for research purposes.

Acknowledgements

This research was supported in part by DARPA under the Command Post of the Future program. We thank Jason Rosenfeld for programming assistance. We thank Major Robert Rasch (US Army), Captain Bill Turmel (US Army), Brigadier General Keith Holcomb (USMC, retired) and Colonel Michael Heredia (US Army) for valuable discussions and insights about tactics and military planning. We thank Will Wright and Ian Davis for valuable discussions and insights about the state of the art in game design.

References

1. Brooks, R. "Solving the find-path problem by good representation of free space." *IEEE Transactions on Systems, Man, and Cybernetics.* SMC-13, 1983.
2. Cohn, A.G. 1997. Qualitative spatial representation and reasoning techniques. In Brewka, G., Habel, C., and Nebel, B. (Eds) *Proceedings of KI-97*. LNAI, 1303, pp 1-30, Springer-Verlag.
3. Dobson, D. and Forbus, K. 1999. Towards articulate game engines. AAAI Spring Symposium on AI and computer games. (AAAI Technical Report SS-99-02)
4. Donlon, J.J. and Forbus, K. D. Using a geographic information system for qualitative spatial reasoning about trafficability. 13th International workshop on qualitative reasoning (QR99), Loch Awe, Scotland. June 6-9, 1999.
5. Forbus, K., Nielsen, P. and Faltings, B. "Qualitative Spatial Reasoning: The CLOCK Project", *Artificial Intelligence*, **51** (1-3), October, 1991.
6. Forbus, K. 1995. Qualitative Spatial Reasoning: Framework and Frontiers. In Glasgow, J., Narayanan, N., and Chandrasekaran, B. *Diagrammatic Reasoning: Cognitive and Computational Perspectives*. MIT Press, pp. 183-202.
7. Forbus, K. "Qualitative Reasoning". *CRC Handbook of Computer Science and Engineering*. CRC Press, 1996.
8. Kuipers, B. (1994). *Qualitative Reasoning: Modeling and simulation with incomplete knowledge*. Cambridge, Mass.: MIT Press.
9. Laird, J. "Toward human-level AI for computer games" Invited presentation, AAAI-2000. http://ai.eecs.umich.edu /people/laird/talks/Soar-games/index.htm

10. Latombe, Jean-Claude, *Robot Motion Planning* (c) 1991, Kluwer Academic Publishers, Norwell MA
11. Leymarie, F. and Levine, M. D. "Fast Raster Scan Distance Propagation on the Discrete Rectangular Lattice," , *CVGIP: Image Understanding*, Vol. 55, No. 1, 1992: 84--94.
12. Lozano-Perez, T and Wesley, M. "An algorithm for planning collision-free paths among polyhedral obstacles." *Communications of the ACM 22*, 10, 1979.
13. O'Dunlaing, C and Yap, C. "The Voronoi method for motion planning: I. The case of a disc." Courant Institute of Mathematical Sciences, New York University, Report No. 53, 1983.
14. Paglieroni, D. "Distance Transforms: Properties and Machine Vision Applications." *CVGIP: Graphical Models and Image Processing,* Vol 54, No. 1, 1992: 56--74.
15. Russell, Stuart J and Norvig, Peter, *Artificial Intelligence: A Modern Approach*, (c) 1995, Prentice Hall, Upper Saddle River, NJ