UAV Swarm Control:
Calculating Digital Pheromone Fields with the GPU

Bryan Walter, Adrian Sannier, Dirk Reiners, James Oliver
Virtual Reality Applications Center
Iowa State University, Ames, Iowa
bryan.walter@gmail.com, sannier@iastate.edu,
dreiners@iastate.edu, oliver@iastate.edu

ABSTRACT

Our future military force will be complex: a highly integrated mix of manned and unmanned units. These unmanned units could function individually or within a swarm. The readiness of future warfighters to work alongside and utilize these new forces depends on the creation of usable interfaces and training simulators. The difficulty is that current UAV control interfaces require too much operator attention and common swarm control methods require expensive computational power. This paper begins with a discussion on how to improve upon current user interfaces and then reviews a swarm control method, the digital pheromone field. This method uses digital pheromones to bias the movements of individual units within a swarm toward areas that are attractive and away from areas that are dangerous or unattractive. Next, a more efficient method for performing pheromone field calculations is introduced, one that harnesses the power of the GPU (graphics processing unit) in today's graphics cards by reshaping the ADAPTIV swarm control algorithm into a form acceptable to the GPU's pipeline (Parunak et al, 2002). The GPU ADAPTIV implementation is tested in scenarios that involve up to 50,000 virtual UAVs. When compared to its counterpart CPU implementation, the GPU version performed over 30 times faster than the CPU version. This gain translates directly into lower costs for training the future warfighter today and fielding the swarms of tomorrow. Finally, this paper presents a vision of how to combine these new interface ideas and performance enhancements into an effective swarm control interface and training simulator.

ABOUT THE AUTHORS

Bryan E. Walter is a dual major candidate for PhD degrees in Human Computer Interaction and Mechanical Engineering at Iowa State University. Bryan received his MS and BS in Mechanical Engineering from Iowa State University and anticipates graduation in December 2005. His professional interests include vehicle simulation, vehicle teleoperation, immersive command and control, and virtual reality.

Adrian V. Sannier is Stanley Professor of Interdisciplinary Engineering in the department of Industrial, Manufacturing and Systems Engineering at Iowa State University. He also serves as Associate Director of the Virtual Reality Applications Center. His research focuses on human/computer interfaces, immersive visual interfaces and their application to human performance and training for complex tasks, and internet-based collaboration and the implications of pervasive computing.

Dirk Reiners is an assistant professor for Computer Science at Iowa State University. He is a member of the Human-Computer Interaction Initiative and the project lead of the OpenSG Open Source scene graph system. His interests cover a wide array of topics in interactive 3D graphics, from software systems through efficient interaction methods for complex data sets to large-scale cluster-based displays. He has a PhD from the Technical University of Darmstadt and can be reached at dreiners@iastate.edu.

James H. Oliver is Associate Professor of Mechanical Engineering and Director of the Virtual Reality Applications Center at Iowa State University. He also serves as director of the Human Computer Interaction graduate program. His research interests span modelling, graphics, simulation, and interface technologies, and their application in science and engineering.

UAV Swarm Control:
Calculating Digital Pheromone Fields with the GPU

Bryan Walter, Adrian Sannier, Dirk Reiners, James Oliver
Virtual Reality Applications Center
Iowa State University, Ames, Iowa
bryan.walter@gmail.com, sannier@iastate.edu,
dreiners@iastate.edu, oliver@iastate.edu

INTRODUCTION

The military has a clear picture of the force it would like to field in the coming decades and this image has a single prevailing theme: the integration of manned and unmanned units. The addition of unmanned units will decrease the danger soldiers face in direct combat, and the DoD (Department of Defense) roadmap calls for an immediate and sustained increase in the use of unmanned units, starting with unmanned aerial vehicles (UAVs) (2002). By 2012, the DoD roadmap projects that F-16-sized UAVs will perform a complete range of combat and combat support missions, including Suppression of Enemy Air Defenses (SEAD), Electronic Attack (EA), and even deep strike interdiction. UAVs specialize in missions commonly categorized as "the dull, the dirty, and the dangerous."

Furthermore, the DoD roadmap requires that a single operator be able to control a UAV swarm. However, for this goal to be realistic, one of two research paths must be chosen and followed. Either the UAVs in the swarm will be totally autonomous and therefore require no human supervision, or the human interface to the swarm will need to be radically different from current UAV interfaces, which require multiple pilots for a single UAV. This paper makes the case for the second path and then elaborates on how to improve upon the current state of the art. The paper introduces two major research challenges: one a more efficient way to control the UAVs, and the other a superior way to monitor and manage a swarm's progress and minimize the number of operators needed. It then presents methods for attacking each challenge, the results currently obtained, and a discussion of the future research direction.

MOTIVATION

The Predator (the most common UAV in active duty) requires at least two operators, one to fly

the plane and the other to mange the camera mounted on it. More commonly, four people man the craft because controlling it is such a taxing task. The four people break off into teams of two and alternate controlling and resting using an interface like the one shown below (Figure 1).

Figure 1. Predator ground control station

One of the main reasons the Predator is taxing to operate is that the field of view afforded to the pilot is poor, resulting in a loss of situational awareness. Situational awareness, knowledge of location and what is occurring near that location, is critical when operating a vehicle in a hostile environment. According to a recent report on the loss of Predator aircraft during missions:

> "A good number of them were lost due to operator error, since it is hard to land the UAV. The operator has the camera pointing out the front of the plane, but he really has lost a lot of situational awareness that a normal pilot would have of where the ground is and where the attitude of his aircraft is." (Charles et al, 2001)

Predator operators liken piloting it to flying as though looking through a soda straw (Grant, 2002). If it takes at least two people to control one Predator it would take thousands of people to control one swarm of a thousand Predators. Clearly, this situation must be improved for

swarm control to be possible. This fact is reinforced by the DoD roadmap since it states that the ground control station must evolve as UAVs grow in autonomy. Specifically, UAV swarms "must be controllable by non-specialist operators whose primary job is something other than controlling the UAV." This demands "a highly simple and intuitive control interface … and the capability for autonomous vehicle operation of one or more vehicles being controlled by a single operator" (USDoD, 2002).

These requirements can not be met by current technologies. One reason is that current UAVs require too much operator attention, and this problem will be partially solved by UAVs with more autonomous flight capabilities. These future UAVs will be capable of flying to a given set of coordinates while automatically avoiding collision. However, even with such advanced UAVs, they will still require human supervision to know where to go and to confirm sensitive actions. As a result, further improvement of UAV ground control stations will be necessary as they are inadequate for controlling and monitoring the progress of a swarm of UAVs.

Due to the difficulty of maintaining situational awareness, it is tempting to suggest that people should not be involved in the control of UAVs. This argument calls for completely autonomous UAVs capable of taking off, flying their mission, taking pictures of or striking a target, and then landing -- all without human input. While not entirely technologically feasible at present, it may be a viable path for future UAV research and development. In fact, some UAVs are already capable of taking off, flying a specified path, and landing on their own (Adams, 2003). The trouble comes in the execution of the mission, be it either strike or reconnaissance. If this task is left entirely to the computer on the UAV, commanders will be unable to focus their swarm to rapidly changing areas of interest in the case of reconnaissance, and they will be unable to decide which targets get attacked in the case of strike. The latter case is of particular concern, because a glitch in the software a UAV uses to decide whether a building is an enemy HQ or a hospital could result in civilian casualties. Additionally, autonomous UAVs could confuse friendlies with enemies if their software fails.

These concerns have been expressed before and most recently in the New York Times article "Who do you trust: G.I. Joe or A.I. Joe"

(Johnson, 2005). This article outlines the dangers of letting artificial intelligence (A.I.) decide everything by taking the human out of the loop. Some might argue that if the A.I. were flawless than there would be no issue. However, it is unrealistic to expect flawless A.I. anytime soon and perhaps ever. After all, humans still make false positive errors, and we are still much better at pattern recognition and high level decisions than computer A.I. As a result, it seems that a human must be in the loop of control, be it for a single UAV or an entire swarm of them.

A human in the loop of swarm control requires three things. First, the human must have access to the control algorithm used by the swarm to enable either direct or indirect control. Second, the human must be able to monitor the swarm's progress on a global level without getting caught up in the state of individual units of the swarm since there are far too many to monitor. Furthermore, there will be hundreds or thousands of UAVs in a swarm, making direct individual communication with all of them expensive. Third, the operators must be trained adequately to manage swarms. This calls for a simulator to prepare commanders for swarm control.

In short, what is needed is a system that has simple controls with relatively uncomplicated units that perform their tasks without knowledge of the entire system, but whose combined actions exhibit complex aggregate behavior. Further, a simulation of this system must be made available to potential operators well in advance of swarm deployment to ensure the warfighter's readiness. Additionally, swarming UAVs should be kept as simple as possible to minimize costs and to enhance their robustness in the field. Despite this pressure toward simplicity and expendability, these groups of UAVs will still be required to perform complex tasks as a whole. While this may sound daunting, nature has provided a template for accomplishing these requirements in social insects such as ants. A swarm control algorithm can be created by making analogies between UAVs and social insects.

A key concept of insect-inspired swarm control is pheromones. Pheromones can be thought of as markers to tell units whether an area is attractive or unattractive for future exploration. In this way, UAVs can use local pheromone levels to determine which direction they should go. With such an algorithm, the commander could change simple parameters relating to

pheromones to influence the movement of the swarming UAVs. For this control to be effective, the commander needs a clear understanding of the entire battlefield and what the swarming UAVs should focus on. A fundamental challenge with insect-inspired algorithms is that they are computationally expensive for large fields.

A strategy explored in this paper for improving the performance of insect-inspired algorithms stems from recent developments in graphics programming. Modern graphics cards in commodity PCs have become complex enough to require their own processing unit. This unit, commonly called the graphics processing unit (GPU), is a parallel vector calculator. Until recently, the GPU had a fixed functionality – it took vertices in world coordinates, converted them to screen coordinates and applied the appropriate color to the pixel. However, now there are two steps in the pipeline that can be programmed, ushering in a new set of algorithms that use the GPU for non-graphics calculations. To run an algorithm on the GPU it must be turned into a fragment program, often requiring that it be written differently than if it were designed for the CPU because some of the GPU pipeline is still fixed. The GPU deals in pixels, vertices and textures so an algorithm must be modified to fit this paradigm.

GPU performs vector calculations much more efficiently than the CPU. This advantage stems from intense innovative pressure from the video game industry as well as the power of its specific design. Since the GPU does not have to handle general computation, its components can be optimized to calculate vector equations. Furthermore, today's GPUs have 16-24 parallel pipelines per card to perform these calculations (Harris et al, 2002). When these advantages are combined, GPUs can provide a considerable performance advantage. For example, a 3.06 GHz Intel Pentium 4 with Hyperthreading can perform six gigaflops with a memory bandwidth of 5.96 GB/s. In contrast, an ATI Mobility Radeon 9700 can perform 25.6 gigaflops with a memory bandwidth of 12.8 GB/s (Fernando et al, 2004). Further, each currently costs around $170.

This paper presents a vision for a system of systems that harnesses the power of the GPU to enable a simulation of human-in-the-loop UAV swarm control. Eventually, this vision could be expanded to control UAV swarms in the field. This vision involves a swarm of semi-autonomous UAVs under the indirect control of an operator. This swarm will be controlled with an insect-inspired algorithm. Each UAV in the swarm will be small and expendable while still providing invaluable reconnaissance by being able to locate threats and targets with on-board sensors and computers. This reconnaissance will be displayed in a virtual world representation of the battlefield (Walter et al, 2004). Further, the commander will directly manage a small number of large F-16 sized striker UAVs using the same virtual world. These strikers wait for orders to eliminate threats or targets. Additionally, the commander would be deployed near the field in a vehicle to minimize communication delay with the swarm. The swarm would gather information about the battlefield with only minor input from the commander. The striker UAVs would then use this information to determine the best path to the desired target or threat set by the commander. Finally, the commander would monitor this striker and could take over control at any time.

Sound like science fiction? While all of this is not currently technologically feasible, it is not far off the technological horizon. Prototypes and simulations of these future control systems can yield valuable insights on future operator interfaces. In fact, it is crucial that these future control systems be explored now, not only to spur UAV swarm development, but also to start training people on how to interact with swarms.

BACKGROUND

There are two main streams of research discussed in this paper that are important to implementing the vision of an advanced swarm control interface: swarm control and battlefield visualization. This section first covers battlefield visualization by discussing our prior work in this area: the Virtual Battlespace. Then it covers an accepted insect-inspired swarm control method, *Adaptive control of Distributed Agents through Pheromone Techniques and Interactive Visualization* (ADAPTIV) (Parunak et al, 2002).

Virtual Battlespace

The Virtual Battlespace immerses users in a virtual environment that provides them with greater context and awareness of the units under their control as well as the overall mission. By integrating radar tracks and UAV video feeds, the virtual world can provide up-to-date access to

the latest real time battlefield information. The virtual world is constructed from a mix of *a priori* information and real time sensor feeds to act as an organizing context for the operator. The result is a mixed reality system in which real world video and radar tracks augment a dynamic virtual world. Using real world data to augment the virtual world is an inversion of the more typical paradigm of augmented and mixed reality where virtual information is used to enhance real world data and imagery.

The research into Virtual Battlespace originated in 2000 when a research team at Iowa State University's Virtual Reality Applications Center (VRAC) began work with the Air Force Research Lab's Human Effectiveness Directorate and the Iowa National Guard's 133rd Air Control Squadron. The goal of this preliminary version of the Virtual Battlespace was to develop an immersive VR system for distributed mission training. It was demonstrated at I/ITSEC in 2002. Virtual Battlespace integrates information about tracks, targets, sensors and threats into an interactive virtual reality environment that fuses the available information about the battlespace into a coherent picture that can be viewed from multiple perspectives and scales (Knutzon et al, 2003). Visualizing engagements in this way is useful in a wide variety of contexts including historical mission review, mission planning, pre-briefing, post-briefing and live observation of mission training scenarios (Knutzon et al, 2004).

Figure 2 shows the Virtual Battlespace. The blue triangular wedges are aggregate units that each represent a squadron of aircraft. The red and white circles represent high threat areas near a SAM site. The commander can use this visual cue to see when units are in danger. The green
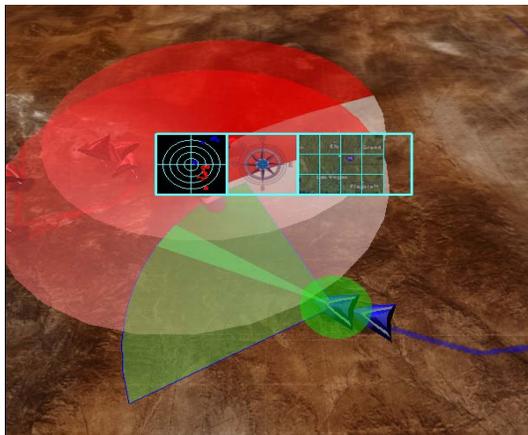

Figure 2. The Virtual Battlespace

wedge illustrates the sensor range of the lead aggregate's radar. Using this visual cue, the commander can determine what that unit can see on its radar. The commander is given more information with a billboard display that shows a compass, a map, and a traditional radar display.

The Virtual Battlespace can be displayed with a variety of display devices, from a traditional computer monitor, to a completely immersive VR device such as Iowa State University's C6. The C6, shown in Figure 3, is a six-walled CAVE display device with each wall consisting of a 10'x 10' stereoscopic screen (Iowa State, 2005). Recently, VRAC researchers received a $2.8 million grant to work with Air Force Office of Sponsored Research to continue development on the Virtual Battlespace for UAV control.
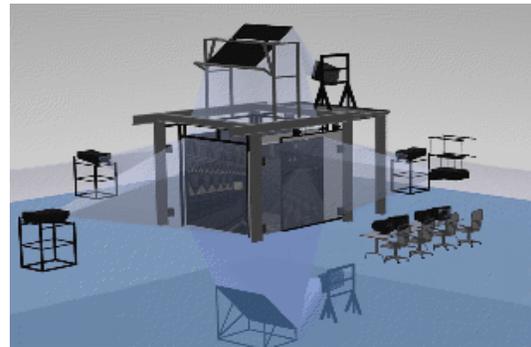

Figure 3. C6 Display Device

ADAPTIV

Social insects provide a template for swarm control as they can have a complex aggregate behavior despite being simple creatures. An individual termite does not know engineering or physics or even what its fellow termites are doing, but a swarm of tropical termites can build complex multi-level mounds that can be five meters tall and weigh ten tons (Parunak et al, 1997). These mounds have impressive overall rigidity, are made from a material that is fire resistant, and contain enough rooms and passages to house the brood and all of its food reserves. Individual ants gather together to perform complex social tasks such as aphid farming. These farming ants protect a herd of aphids (smaller insects) from predators so they can milk them of their honeydew (Holway et al, 2002). Also, most ant species form invisible roadways called "ant highways," where columns of ants follow one another without the aid of street signs or painted lanes. These complex behaviors are accomplished via pheromones.

Pheromones are chemicals produced in different flavors. Insects release them into the air to use as markers with each flavor conveying its own message. One chemical compound might signal food while another might signal danger. Insects use pheromone flavors to communicate with their fellows. If an ant finds food, it drops a pheromone that tells other ants, "food this way." As this pheromone reaches the receptors of other ants, they can use it to go to the food source.

Pheromone-based swarm control algorithms for man-made units have been explored by other researchers. Most pheromone-based algorithms use the concept of digital pheromones, data markers that are passed between units in a swarm with a network instead of being carried by the wind. The unit's pheromone receptors are packet readers. While in nature, the environment facilitates pheromone movement and interaction; in computer swarm control algorithms, some other entity must facilitate these pheromone activities. In this case, a global data structure serves that function. This data structure, a grid of all present pheromone flavors and intensities, called the pheromone field, is considered to be external to the units in the swarm.

ADAPTIV is a pheromone-based swarm control algorithm that uses a hex grid for its pheromone field. Each cell consists of a set of numbers that represent the magnitudes of the different pheromone flavors. The flavors represent the presence, as detected by the unit, of different things of interest, such as threats or targets. Alternatively, the flavor can be used to enable swarm dispersion. This type of pheromone flavor is called UAV repulsion pheromone and is dropped at the current location of the unit in the swarm. The threat and repulsion pheromones are unattractive while the target pheromone is attractive. ADAPTIV contains three steps:

- update the pheromone field
- propagate pheromones
- move the units in the swarm

The first step updates the strength of the pheromones at each field cell with Equation 1. It is a vector equation with each of its components corresponding to a unique pheromone flavor.

$$S(t+1,p) = E*S(t,p) + R(t,p) + Q(t,p) \quad (1)$$

Equation 1 states that the pheromone strength (S) at cell location p at time $t+1$ is equal to the S currently there (at time t) times the evaporation

coefficient (E) plus any pheromones added to this cell from units in the swarm (R) at time t plus pheromones that have propagated from cell p's neighbors (Q). E is a constant between zero and one and it represents the percentage of pheromone that remains within a cell after evaporation. This evaporation is important as it makes pheromones linger for only a short period after the target of interest is gone, automatically keeping the known state of the world up to date.

Two variables, R and Q, must be known in order to calculate S in Equation 1. When a unit finds something in cell p it places a pheromone of the appropriate flavor in the p location of R. R is cleared each time since it represents new pheromones input into the system. Pheromones dropped in R are added to the current value making pheromones of the same flavor at the same location additive. With R in hand, only Q is left unknown. The calculation of Q is done in step two of the algorithm by using Equation 2.

$$Q(t+1,p) = \sum [F/N(p')*(R(t,p')+Q(t,p'))] \quad (2)$$

Equation 2 describes pheromone propagation and is the weighted sum of pheromones from each cell's neighbors. The p' in Equation 2 denotes the location of a nearest neighbor of cell p. $N(p')$ is the number of nearest neighbors that cell p' itself has. In the case of a non-ending hex grid $N(p')$ would always be six. F is the propagation coefficient with possible values ranging from zero to one. This coefficient is the percentage of pheromones at a cell that will propagate to its nearest neighbors. In Equation 2, Q represents new pheromones added to cell p by its neighbors. The value of Q would then be the sum of the individual contributions of each of p's nearest neighbors, hence the sum in Equation 2. The part inside the sum is the contribution of each nearest neighbor. This contribution, dampened by the propagation coefficient F, is split equally among its nearest neighbors. As a result, $F/N(p')$ of the newly inputted pheromones (given by R and Q) in cell p' propagate to cell p.

The third step of ADAPTIV is to move the units in the swarm using the pheromone field. In ADAPTIV, each unit in the swarm uses roulette selection to determine where to go next. The bin lengths for this roulette selection are determined by the local pheromone levels. There is one bin for each of the cell's nearest neighbors and that bin is made longer by the magnitude of attractive pheromone flavors and shortened by the

magnitude of repulsive pheromone flavors. In this way, the swarming units move somewhat randomly with a preference for attractive cells. ADAPTIV was tested in 2002 in simulated scenarios and was found to be capable of enabling swarm control (Parunak et al, 2002).

GPU IMPLEMENTATION

As mentioned in the motivation, current graphics hardware is programmable. There are two separate levels in transforming 3D geometry into an image that can be programmed: the vertex and the pixel level. For our algorithm (and for non-graphics uses of GPUs in general) the pixel level is more interesting, as most of the computational power of a GPU is concentrated here. Using a so called fragment program the operations that calculate the final color triplet (independent red, green and blue components) from the input data like application-defined attributes or texture images can be changed as needed. The problem is reformulating a general algorithm in a form that is suitable for the vertex-stream- and pixel-oriented programming model of a GPU.

The first step in converting ADAPTIV is to represent a pheromone fields as a 2D texture. The different color channels in the texture are used to represent the different pheromone flavors; red for threats, green for targets and blue for UAV repulsion pheromone. Equations 1 and 2 can then be turned into fragment programs.

The first rendering step in the calculation of Equation 1 is to draw a screen-filling polygon with the fragment program enabled. The program includes Equation 1 using texture reads to get access to pheromone data. Both the S and Q fields are stored within textures, but the R values are not because the R pheromone may or may not be in a particular cell since it only exists if a UAV drops an input pheromone into that cell. Further the location of introduced pheromones can change rapidly. Therefore the R pheromones are not stored in a texture that would need to be updated on the CPU every step; rather they are additively drawn as points of the appropriate color (based on the pheromone flavor discovered by the UAV at that location). The first step fills the frame buffer with colors corresponding to pheromone levels $E*S+Q$. The second rendering step adds the R pheromones by enabling a simple additive blending mode and then drawing colored points at each UAV location. Finally, the frame buffer is saved to the S texture.

ADAPTIV's next step, pheromone propagation, uses Equation 2 converted to a fragment program. First, the previous Q values are put into the frame buffer by drawing a screen-filling polygon with the Q texture. Then points are drawn where UAVs drop pheromones and the current color is copied to the Q texture. This texture, containing $R(t,p') + Q(t,p')$, is then used to calculate $Q(t+1,p)$ with a fragment program version of Equation 2 applied to a screen-filling polygon. The implementation of this fragment program is made easier by making the borders off-limits to UAVs, threats, and targets. This restriction results in all cells having eight nearest neighbors, making $N(p') = 8$. Since $N(p')$ and F are constant, they can be taken out of the sum and the same fragment program can be used for the entire screen-filling polygon.

The third step of the ADAPTIV algorithm, roulette selection, involves a more complex fragment program. To perform roulette selection, a point is drawn for each UAV with the roulette selection fragment program active. These points are drawn in a line for efficient collection of the results since only a small subset of the frame buffer needs to be retrieved after running the fragment program. A different random number, UAV sensitivities, and nearest neighbor cell positions are also sent to the fragment program as vertex attributes.

The main task of the roulette selection program is to calculate the length of the bin for each of the cell's eight nearest neighbors and then select one of them. It accomplishes this task by using the color of S at each nearest neighbor of the UAV to create a corresponding bin length. A passed-in vector, called the PherSen vector, contains the UAV's sensitivity to each of the pheromone flavors. Since red and blue channels represent repulsive pheromones, these two components of PherSen are negative. A direct result of this is that cells that contain more green than red or blue will be considered "better" choices. However, both the amount of each color present and the magnitude of the components in PherSen determine how much "better" a nearest neighbor cell is over the others. The length of each of the bins is determined by the dot product of the color vector at a cell with the PherSen vector. The higher the magnitude of the numbers in PherSen, the longer (or shorter if the number is negative) the bins get. Additionally, there is a constant value called the "shift" added to each bin length. The purpose of

the shift is to make cells empty of pheromones (black cells) somewhat attractive. Without a shift, black cells would have a bin length of zero and have no chance of selection.

With all the bin lengths calculated, biased roulette selection can be performed. The random number between zero and one is multiplied by the total bin length. Then, the fragment program selects the first bin it finds whose cumulative length (based on the sum of all the bin lengths that came before it) exceeds the random number. The location of the cell whose bin was chosen is then returned in the red and green channels of the pixel. After all selections are done, a sub-texture containing the positions is read from the frame buffer. These positions are then assigned to the corresponding UAV. The UAVs then try to reach the positions and the process is repeated.

RESULTS

To assess the performance of the GPU version of the ADAPTIV algorithm, it was compared with a CPU version. The time for one execution of all three ADAPTIV algorithm steps was recorded for certain combinations of battlefield size and UAVs in the swarm. The battlefield size, measured in rows and columns of the pheromone field was tested at sizes 64x64, 128x128, 256x256, 512x512, and 1024x1024. The number of UAVs in the swarm was tested at 5, 50, 500, 5000, and 50,000. In each case, 120 data points were gathered. The test computer had a 3GHz Intel Pentium 4 with Hyper-threading and an ATI Mobility Radeon 9700.

In the cases where swarm size was investigated, the battlefield size was 128x128. Figure 4 shows the results for these cases. The GPU's performance advantage was found to grow slowly with increasing swarm size. At 5 UAVs the GPU was 30 ms (1.8x) faster and by 50,000 UAVs the GPU was 55 ms (2.7x) faster.

In the cases where battlefield size was investigated, swarm size was kept constant at 4000 UAVs. Figure 5 shows those cases. The GPU's performance advantage was found to grow quickly with increasing battlefield size. For example, at a size of 64x64, the CPU is actually faster by 5 ms (1.2x) but by a size of 128x128 the GPU is faster by about 43 ms (1.4x) and at a size of 1024x1024 the GPU is faster by about 440 ms (34.8x). The results of these test cases show that the GPU algorithm outperformed

its CPU counterpart in almost all combinations of swarm size and battlefield size.
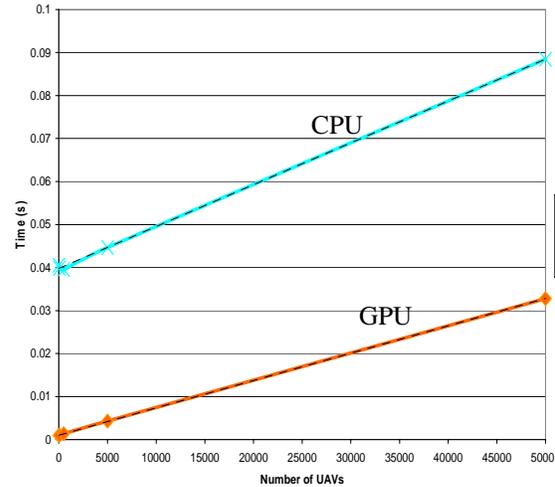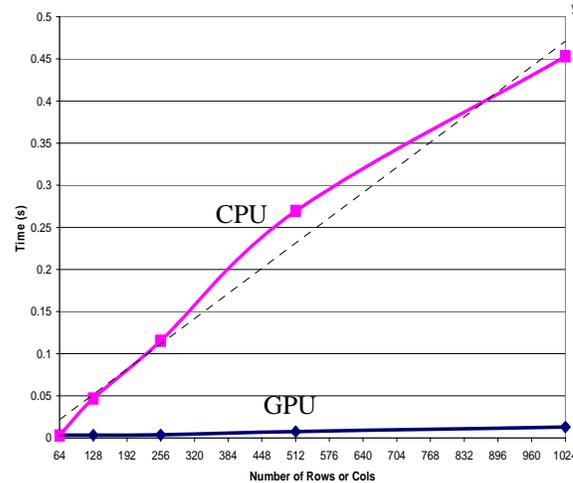

Figure 4. Performance vs. Swarm Size

Figure 5. Performance vs. Battlefield Size

SWARM CONTROL SIMULATOR

The GPU ADAPTIV algorithm and the Virtual Battlespace can be combined into a powerful training simulator. This simulator will require only two networked commodity level PCs. Because of the performance advantage of the GPU ADAPTIV algorithm, one computer with a 128M graphics card can simulate the behavior of a virtual UAV swarm of 50,000+ units in real-time with a time step of 0.1 seconds. The second computer would then run the Virtual Battlespace and the UAV swarm would be represented within the virtual world. Further, the pheromone

field (represented by the S texture) could be overlaid on the terrain in the Virtual Battlespace.

One user interface in the swarm simulator would allow the commander to manipulate the behavior of the swarm by changing its aggressiveness in exploring the battle and identifying threats and targets. Changes input into this interface would send a network communication to the computer running the swarm to modify the PherSen values of the UAVs to ones that have been seen to produce the desired behavior. Immediate feed-back on likely loss of aircraft due to the behavior setting could be displayed. This estimate of loss would come from prior runs with those settings.

A second interface provided to the commander through the Virtual Battlespace would be the pheromone field modifier tool. It would allow the commander to drop pheromones of a particular flavor into the field within an area designated by a user-defined rectangle. The pheromone flavor could designate the area that is off-limits to the swarm or it could represent a strong attraction, pulling the swarm to that area. This tool would allow the commander to directly manipulate the movement of the entire swarm. Further, the evaporation and propagation parameters for these new pheromones could be set from within the Virtual Battlespace. Different battlefield and swarm combinations could be easily developed and used in the simulator to test trainees in a variety of missions. Additionally, it would be straightforward to collect performance data that could later be used to analyze the commander's decisions utilizing the swarm.

In conclusion, fusing the Virtual Battlespace with the GPU ADAPTIV algorithm would create an ideal swarm control simulator. The Virtual Battlespace is flexible enough to handle new scenarios over varied terrain with minimal modification meaning that trainees could be tested in a variety of missions potentially involving both manned and unmanned units. Furthermore, with the inclusion of the ADAPTIV algorithm, missions involving a simulated UAV swarm would be easy to set up. This simulated UAV swarm would enable operators to be trained on how to interact with a swarm well before one can be deployed in the field. Additionally, since the swarm movements are simulated on the GPU, less expensive computing power is needed to run the simulator. All of this combines to make an effective yet affordable swarm control training simulator.

FUTURE WORK

The UAV research group at VRAC will continue developing the prototype of the swarm control simulator described in the previous section. We hope this work will uncover opportunities to improve the system beyond the vision laid out in this paper and will help in bringing UAV swarm control and its training closer to reality.

REFERENCES

Adams, Charlotte (2003). UAVs That Swarm, *Avionics Magazine*. Retrieved October 2004 from http://www.defensedaily.com.

Charles L. Barry and Elihu Zimet (2001). UCAVs Technological, Policy, and Operational Challenges, *Defense Horizons*, Number 3, Center for Technology and National Security Policy, National Defense University.

Fernando., R., et. al. (2004). **GPU Gems**, Addison-Wesley, N-Vidia Corporation.

Grant, Rebecca (2002). Reach-Forward, *Air Force, Journal of the Air Force Association*, vol. 85.10.

Harris, M., et. al. (2002). Physically-Based Visual Simulation on Graphics Hardware, Department of Computer Science at North Carolina University, *Graphics Hardware*.

Holway, D., et. al. (2002). "The Causes and Consequences of Ant Invasions", Annual Review of Ecological Systems, Vol. 33.

Iowa State University Virtual Reality Applications Center Website (2005), Visited June 2005 at www.vrac.iastate.edu.

Johnson, George (2005). Ideas & Trends: Who Do You Trust: G.I. Joe or A.I. Joe? *New York Times.* Retrieved Feb. 2005 from www.nytimes.com.

Knutzon, J., Walter, B., Sannier, A., Oliver, J. (2003). Command and Control in Distributed Mission Training: an Immersive Approach, *NATO Symposium on Critical Design Issues for the Human-Machine Interface*, Prague.

Knutzon, J., Walter, B., Sannier, A., Oliver, J. (2004). An Immersive Approach to Command and Control, *Journal of Battlefield Technology*, Vol. 7, No. 1, pp. 37-42.

Parunak, Van Dyke (1997). Go to the Ant: Engineering Principles from Natural Multi-Agent Systems, Altarum Institute, *Annals of Operations Research,* Vol. 75.

Parunak, V., Purcell, M., O,Connell, R. (2002). Digital Pheromones for Autonomous Coordination of Swarming UAVs, *American Institute of Aeronautics and Aerospace*.

US Department of Defense (2002). Unmanned Aerial Vehicles Roadmap 2002-2027. Retrieved June 2003 from www.acq.osd.mil/usd/uav_roadmap.pdf.

Walter, B et al, (2004). VR Aided Control of UAVs. *3rd AIAA Unmanned Unlimited Technical Conference,* Paper AIAA 2004-6320, Chicago.