The following content is provided under a Creative Commons license. Your support will help MIT OpenCourseWare continue to offer high quality educational resources for free. To make a donation or to view additional materials from hundreds of MIT courses, visit MIT OpenCourseWare at ocw.mit.edu.

PROFESSOR: So I'll be sitting down here and showing you a whole lot of really cool videos on YouTube, and battlecode players that I have programmed especially for tonight. So here we go. What's the point here? The point here is to talk about the things that have been done before in swarm, and thinking about swarm.

So let's go straight to a YouTube video and look at this animation. So you can see here, there will be these birds that are following the green blob, and they avoid the red blob. And what they're trying to do is match one another's direction, try to get a certain distance from the other birds, and try to avoid obstacles.

And so you can see how the combination of a few simple rules can create a behavior that's sort of complicated looking, or a behavior that's somewhat fluid. Whereas if you had manually tried to get them all to go in the same direction, it might have looked somewhat more rigid. And it might not have been able to handle things like obstacles.

So let's look at another video of how this is done in nature. Now, this is a massive number of birds. I think in the video description it says this is 300,000 birds. Wow. And it's so pretty how it undulates and smushes around. Here's another view of the same thing. I think they'll all take off in the distance. Look at how pretty that is.

And the all sort of do it once, but there's certainly nobody in control. What a pretty-- ah, you've got to love pictures of nature, in the evening, in a lecture hall. We're about as far from nature as people can get. My goodness. Ah, that's so pleasing. When are they going to show the birds? I'm referring to the birds that aren't in the water. The birds in the air, yeah.

So what we want to do today is just talk about the things that are recognized as

elements of swarm behavior. And then we'll show how our robots can do much better than the enemy robots if we do it right. Look at this. It's almost like two separate swarms in certain areas.

And it's an interesting problem to say, at what point is it possible to split into two swarms? Because maybe one of them got too far from the other, or maybe it's just too gosh darn big. Or maybe one half of them are Democrats, the other half Republicans. There are a lot of reasons that they might want to split into parts. So we'll talk about that too.

Next, we'll talk about crowds. I mean, this is a thing that you deal with every day when you go to eat battlecode lecture dinner. You come up here, and there's people trying to get to food. And I guess the food is there. No, this is a bad example, because these guys are just going to walk through one another. And the goal is trying to get them to anticipate who isn't going to be in the way, and to form lanes. That was the problem that these guys were trying to address.

And I guess nowadays it's becoming popular to post your thesis material online and to animate it in 3D. I don't see why this is animated in 3D. You could just show the same thing with a two dimensional map. Now here you're going to see ripples form. And that's the expectation path. And so a small perturbation in the position of someone is amplified into like avoid this lane.

And here he shows it again with some lines. He does it with more people, because every computer scientist knows that more is better, especially when it comes to simulations. And so he shows a bunch of them, and how these guys are going up, those guys are going down there. And they sort of make this vortex.

And I think-- oh, yeah. I remember. This is really silly. The end of this video, you're not going to see it coming at all. You're thinking, all right, where can he go from here? He can show-- so there's their model, and there's some alternative models showing that their model is a lot better because people aren't bunching up. So there's these high ES low and high distance models. Which basically, the guys are just avoiding one another.

But in their model, they're sort of creating a density field for an expected position. I think this is supposed to be simulating armies. I guess they want a job at Pixar or whatever. And so they just sort of put this one in here, like oh, look at how realistic your armies will look if people at the back move 10 times as fast.

[LAUGHTER]

PROFESSOR: And then another army is over here. Because just like I said, these computer science people like nothing more than to have a lot of things. I once saw a simulation. Oh, this was crazy. It was a simulation of neutronics. So this is like a college campus. I mean, you could really tell where this guy's coming from. Movies, campus. And it's like people are coming from certain doors and going to others. And I think the next shot he'll show the paths of these people, showing that they're not bumping into each other that much, and that they're kind of straightish paths. Which is supposed to be very impressive.

But back to my other story, I was watching a neutronics simulation. And it had like 2 billion particles in it that were all in a lattice. OK, I'm going to pause this here when it gets to the city. I mean, my goodness. How more complete do you need? Once you've seen it, you've kind of seen it.

But anyway, this 2 billion particle lattice, which took like 10 hours on a giant supercomputer to do this simulation of damage from radiation, from like high energy particle radiation. And the final result of this massive simulation that I saw, it was to show that this one incoming particle had ricocheted in a couple of ways. And the diagram of this final result from this giant computation, which was like the very end of his presentation, because for some reason people don't like to show the graphics first. They like to tell you all the words first.

In any case, he showed me this thing. And it's like a little hook or something. It was really stupid looking. Oh, my goodness. Can't believe it. But no, this guy's final result is going to be much more exciting.

So he's got these people, and he's like, well, this isn't realistic. In a real city, there

are cars. So you've got to only walk on the crosswalks. And you've got to avoid certain things. Maybe you form lanes of traffic on these crosswalks kind of automatically. Oh, now he's getting more. He's like, OK, I've got too much time on my hands. I better implement stop lights.

But it's still not that realistic, because what if people get hit by cars? There are no cars to hit people, or to almost hit them, or to scare them and get them to jump out of the way. So now we've got a 3D flyover, because the only thing cooler than having a 3D environment is a slow pan and zoom.

So that's what he's sort of doing here. And he puts the cars in. Obviously, he has excellent taste. All the cars are from 1970. And I guess there's jaywalking. Oh, right. This is like a chemical evacuation or something, so all these people are covered in yellow stuff. And they're running away. And that's supposed to be really useful for planning evacuations and such.

Yeah. And everybody else is just like, OK, I'm going to worry about that three, two, one, now. And then the other people start running away, too. They have really bad reaction time. And then this is like, people are walking in the street, and how it's messing up traffic. And it's like, wow, this traffic is almost as bad as New York City traffic. Almost as bad.

And then, I mean, what else can this guy do after he's done this many things? Well, the last thing that he has is aliens.

[LAUGHTER]

PROFESSOR: So there you go. Just to show that everybody likes to have a little bit of fun with pathing and navigation.

So let's talk a little bit about the things that we saw in those videos. and how they apply to battlecode. So there's a whole question of centralized versus decentralized control. And in the ones that we saw, they weren't centralized. So I'm going to show you examples in battlecode of both of those.

Now, those can have impacts on whether there's a radio attack possible, whether your strategy's going to be vulnerable to that. Whether it's like a map dependent strategy, because maybe on a centralized controller, it works really well if it's a tiny map. But maybe if it's a huge map, you've got to start having multiple swarms that are independent. And talking about distributing information is another thing to worry about it.

So let's consider that in the boyds model, which is supposed to sound like birds if you have a Brooklyn accent, they considered several aspects for computing the next position for each actor in the swarm. The separation of each one to its neighbors. The pushing together of each one, which they call cohesion, because the smarter you sound, the more Ph.D. Money and grant money you can make.

They talk about alignment so that everybody's facing the same direction. Now, that's somewhat different this year than it has been in past battlecode years. And then I guess they don't talk too much about it, but they considered the later effects of group splitting and lane creation, which we saw today.

So let's see here. I showed some examples of these things. Alignment this year is worth noting that there used to be a method where you could take the robot info and do dot direction. You could get the direction of any robot that you can see. Now, in this year's battlecode, there's no direction. So what are you going to do?

So let's see here. You can get the distances for separation and cohesion. I mean, these are just a fancy way of just saying there's like a spring between you and the others. And this one is just saying that you're headed the right way. There's really almost nothing complicated to it.

And we're going to end up seeing some pretty neat looking behavior from very simple rules. So yeah, let's talk about how you're going to know that you're facing the right direction. I mean, you could check all the neighbors. Let's see here, face the right direction.

You could check all the neighbors against all the other ones. So you'd be checking

to see if robot ID number 97 was from this position, and then that one. And then you would try to find where he was, and correlate that, and get the direction. And then take an average for all the robots that you can see. That would be really computationally expensive.

Another option is to say, all right, take average of all allied robots. Or all within a range. And then see how that average moves. So that's like an implicit direction for the group of allied robots. And then an individual robot, by doing that, could see where all of them are.

Another option is just to radio broadcast so that somebody somewhere is deciding what direction to go. Or maybe all of them vote on the direction that they want to go, because maybe some of them are close to enemies and other ones are standing on mines. And maybe they want to put it up to a vote so that they can wait for 50 turns before finding out which direction they're going to go, and end up going in the wrong direction anyway. Or they could have a recount, and anything. It could be really fun. You could do like a whole political simulation.

So here are some considerations when you're talking about central control versus the group mind. Where the central controller, you'd need a leader. And so leaders have to have elections, and they also can die, and sometimes they get lost. So those are things to worry about when you have a leader. That means that the headquarters is a good choice.

And so if you can do all of your computations in one place, that's much more efficient. Imagine if you had to-- every robot has all the same information. But if all of them make the decisions, then they all have to do a lot of thinking. And in this game, if you use up 10,000 byte codes, each of those robots is going to have twice the power upkeep. It's going to be pretty significant.

So you could have the leader broadcast all kinds of things. And later on we're going to talk about some more little details of what the leader can broadcast besides just what direction you're going in that can really save you a lot of trouble. If you have a group mind, it's a little different. You can have a better site range, which is to say

that in previous years you were limited to just one robot site range. So having a group mind sort of gave you more access to the edges of your cloud.

And you could also make it so that the front guys are leading. Which would be really useful so that you could have them be the first ones to note. This year it's not so important. But in like a real flock of birds or something you might develop after this, it would be.

You can also broadcast of the enemy locations in a group mind. There's no reason why a group mind can't use broadcast. It's just a question of whether you expect to receive something from a single entity, or whether you expect to integrate a bunch of broadcasts from a lot of entities. Broadcasting is becoming cheaper and cheaper now that the read cost has been decreased from the 0.01 to 0.003.

We've also increased the number of channels in the latest release so that your broadcasts are more secure against radio attacks. At least those spanning attacks. It is worth mentioning that you should be moving your channel somehow. Come up with your own algorithm for shifting the channel that you're using. Something, for example, that uses the clock round number so that you could always be looking in a different place, and the enemy won't locate which channels you're using on the radio. Which is to say, which frequencies.

So you might also want some other features. So these boyds things with their separation and such. What other features do you want that aren't in boyds? So you might want some in and out motion so that everybody takes a uniform amount of damage. You might want some cycling behavior. You might want to heal the weak by going back to the med bays.

You might want to apply shields at shield generators. Those are encampments that you could build. And you might want to make sure that each robot knows the size of the pack. And if each one only has so much information, especially about the enemy, then they need to be able to get around. I mean, this year it's not so much of a problem again.

So let's see here. There's sort of an interesting trade off here when you think about the density of a group. But we're doing too much talking and not enough watching. Let's watch an example. So I have some players here. And swarm one, I'm just going to pick any old map here. Let's go to chunky. Chunky is a map I made which has just these big chunks of mines.

So here you can see, I've got my swarm one player. It's very simple. It's quite close to the other players that I showed you. It's not that exciting, because the robots are just sort of-- they rush at one another. They come together in the middle. They go toward one another. There's not that much swarm-like behavior.

And when they come together, they're not really doing anything that intelligent. Like at this moment, neither side is really being that smart about what to do with the other. Here I'm going to do it slightly differently. I'm going to have the robots, instead of just choosing a rally point at random at the very beginning of the game, and having each robot compute it individually, here in swarm one, I'm going to show how in swarm two, which is this code, we've included the ability for the headquarters to tell the robots where to go.

So you could see-- let's see here, let's change the font size so that you can see what I'm saying and follow along in the code. Font 16, OK. The headquarter code here is what the headquarters is doing. So it does some spawning soldiers. And I've got a nice code documentation here. And it moves the rally point from time to time, which is to say that at round greater than 500, it just sets the rally point to the enemy headquarter location. And it messages allies about where to go.

So first it gets the channel from my function called getChannel. And when I do that, I say that if I'm team b, I'm going to have this multiplier. If I'm team a, I'm going to have this multiplier. So that when I calculate the channel, I do the clock round number times the multiplier, modulo the game constants, broadcast max channels. So that means that the channel that I'm broadcasting on is continually rotating, but because it's just based on my team and the clock round number, that means that every robot can still read off that channel. Still knows where to look for that channel.

And rotating frequencies is actually a common practice in modern military communications. After all, it might not have occurred to you that in the modern military, they don't just flip open their cell phones and then say, hey General Saxony, do you want to sort of attack tonight? And the general goes, yeah, maybe. I might. I'm not sure. Maybe I'll send you an email.

They often use radio, because it's there's not internet connection out there in the middle of nowhere. And they have these rotating radio frequencies. So yeah, that's a good thing that you'll have. So once you've got the channel, you need to have a message.

So here I've written the message builder that converts a map location to an integer. So you can see that function here. All I do is I multiply the x value by 1,000 and I add the y value. So then both x and y values are included.

Finally, on that unpacking side, I use this function to turn the integer back into a map location. It makes life easier. So then at the robot side, you can see the result. So we're going to show swarm one versus swarm two.

Swarm two has two advantages. So here swarm two is in blue. And swarm two has two advantages, the first of which is that it receives the messages from the headquarters. So it'll go attack at round 500, rather than the round that these guys are attacking, which is round 200. So I guess it has sort of an advantage there.

And second advantage is, it doesn't defuse mines in combat. Some of you may be familiar with World of Warcraft or those type games. Defusing mines in combat is like looting in combat. It's definitely a no-no. You don't want to-- that's not how to be a team player. You could see the result is that just because you aren't defusing mines in combat where the other team was, you end up with almost your whole army remaining. What a huge difference.

And then quickly at round 500, you can see that these guys are successfully receiving a goal. And here in size 0.001 font, you can see that it knows its current location. And it has a goal location which has been broadcast from this guy.

So there you can see a little example, but that's not as exciting as it might be. Let's consider adding some distance tolerance in go to location. So let's do that so that when they go to a location, they don't just keep bobbling around in there. So that they can actually start to get work done.

So here, they're going to location. And then they just calm down. And they take their spot, and they sit there. Really nice. And if they're on a certain tile, I even have they lay a mine. So this is a neato thing that I thought of is if you lay mines not in a fully dense pattern, then when the enemy attacks, you have the opportunity to sort of whittle them down.

Because you might have this situation. It's kind of hard to see on this map. I might take a screenshot and then work on it. So let's say that we've got these mines here, and that we intend to use them in a really clever way. Yeah. So let's go here. Super leet. Super leet.

So maybe the enemy is coming this way. Now, if they come here, they can either start to defuse the mines, and they'll be just waiting on that corner, or they can infiltrate inside. Because it doesn't really present a giant obstacle to them. And once they're in here, I'm going to have a big advantage because I can confront them with a lot more area. Because they're all so sparsely laid out.

So I think this will work really well. And if they want to retreat, there's definitely no way that's going to work for them. Because they can't do any real cycling in this configuration. I think it's going to totally mess them up.

So here in this one, I've got these guys who are cleverly not-- they're not going to defuse mines in combat. And they'll even sort of back up and fight over the mines. So here they're backing up to the mines. And they're trying to bring the fight back to home.

Let's show it with swarm one versus swarm three, because then they won't both have that part. Oh, wait. That was pretty one-sided. Right. So this is something that happens. So here the enemy is going to show up and I'm going to retreat off the

minefield. So then they're sort of busy, and I come back. And then I retreat a little bit more. And they sort of start defusing and they think that's a good idea.

And then I sort of go this back and forth type motion. And that kind of thing can help to string out the enemies in a line. Give me a little advantage. But even still, it's not looking that much like swarming.

Now, I want to make the point that a lot of these maps are absolutely filled with these neutral mines, especially to begin with. In this map. there's so many neutral mines to get through. If you were really intelligent, you could take this path. And I'll show an example later of how you could write a player that really efficiently paths through this space for a whole mess of troops.

So here I'm retreating past my little minefield. And then I'm sort of bopping on forward again. And he's sort of poking the dragon here. He's making me irritated. And so then-- oh, I'm just going to kill him. Hmm.

But the killing him was less effective than it might have been because a lot of my guys got stuck back there. And they didn't know what to do because I told them, please don't defuse mines in combat. And it was like combat because they were going to attack the enemy headquarters. But nevertheless, these guys got stuck. So what am I going to do?

Well, here's an alternative where you could say each guy has a certain amount of patience. And if he's stuck in one spot, then he's going to go ahead and defuse those mines in combat. So here, once again, this pulling back and going forward strategy is very effective against an equally sized enemy. These minefields are a wonderful defensive advantage that you should definitely use. So then here, when these guys get annoyed, they can start digging through. And eventually, they'll make it through to the other side. Which will be pretty useful in that case.

Let's do another example versus basic player. I think this example demonstrates that tunneling effect a little bit better. Now, basic player is building all kinds of useful things, like suppliers and generators. And that's going to make him especially

powerful. So it's going to be-- no, actually, I beat him.

[LAUGHTER]

PROFESSOR: So that's a really good example. Let's move on, and I'm going to try to beat swarm four. So swarm four was like-- well, maybe I'll continue beating swarm one. What I'm going to do is I'm going to use guided swarm, which is a new player that I made. So this guided swarm's really neat. This is a lot more like birds.

And the guided swarm's goal, his goal is to be a little bit more puffy. Yeah, that's him up there. Look at him. He's like all cloud like. And he's keeping it real. What a guy.

So you can see here that the guided swarm's kind of OK, but he's so spread out that he doesn't do well in combat. What can we do to make this guided swarm better? What can we do-- I think one thing that works really well is to put him on a map that's better for him.

So I'm going to put him on this one. Now he's a bit farther away from the enemy. And my hope is that he'll have time to produce enough units that he'll start laying a big minefield. Yes, I think that will help him out significantly. Oh, come on. Please lay that minefield. Please lay that minefield.

Oh, he didn't have time to lay the minefield. I think what I'll do is I'll tell him to build the minefield just a little bit earlier. I'm going to make this-- right now I've got it-- ooh, don't let me change the wrong code. It's so easy to do that. You think robot player got Java is the code you're working on, and it is, but you're wrong.

[LAUGHTER]

So I was working on guided swarm one, and I want the headquarters code to start building these robots. Oh, yeah. I want him to research pickaxe. And once he's researched pickaxe, I told him to tell the other guys on channel plus 1. So here you can use your get channel function and then add one, and now you've got sort of a set of channels you can use.

So I want him to tell the others to please start laying mines as soon as I have the

pickaxe upgrade. All right. So that's going to happen. I'll just make it at this point, and hopefully that will give him enough time to do it.

Yeah. So now he's going to build fewer allied units before he starts researching. Gosh, this is a tiny map. If you can see this accurately, you don't need glasses. Yes, that's probably true. Ah, right. So they started laying mines. And look at that pattern. Look at the pattern they used.

Oh, my goodness. They must be somewhat skilled. Can't believe that. Ah, ooh.

[LAUGHTER]

PROFESSOR: But you see the point. The point that I made was that I found a really neat way to make them lay mines in the right pattern. Now, maybe you can't really see that well. So I'm going to open up good fashioned Mathematica. Not that. Good old fashioned-- not that. Good old fashioned-- there we go. Mathematica.

So you can see that this arrangement, this disposition of mine positions, which is like over one, up two-- I've got to zoom in a little bit. Yeah, if you go over one, up two, over one, up two, this arrangement is going to totally fill space. So this is where I've colored all the adjacent ones in. And this is going to mean that if you lay mines at these centers, and you have the pickaxe upgrade, then you will be fully tiling space. Really useful.

And here I'm going to give away the trick. You'll be on the right spot if mod of 2 plus your x val, plus 1 times your y val equals zero. If mod, this number 5-- see if I change this to like a 7, then the whole pattern changes. And you can investigate different patterns by playing around with it.

And these patterns may have different effects, depending on your upgrades and the positioning of your mines. And it made me get kind of a neat idea. So this is fully dense. And you could do partly dense, which is really simple. You could just do mod i plus j, where i and j are your x and y values. You just do mod 2. And if that mod 2 is equal to 1, then that'll get you every other tile.

So no matter where you are on the map, you don't have to start comparing to your allied tile locations. You don't have to start doing a whole bunch of really complicated operations. Just say, if my robot is on this nice tile, then go ahead and lay a mine. Really nice.

And here's an interesting alternative. So this one I slightly farther spaced apart than the one above. This is sort of like knights in chess, where they can move up and up and up, or they can move over and over and over. Well here, I've gone and done it a little bit farther apart. And the result is this neato looking structure.

So it's like the one we did before, where we had only done a checkerboard pattern. Only now, we have a lot fewer mines to lay, and we still get sort of this interpenetrated mix. This can be really useful for fooling the enemy team. The formula for this system is 3x plus 1y. And as long as you mod that with 8, you'll get the answer you're looking for.

Now, this is sort of an interesting system. If I change this to 1, and I change this to 5, I get the same result but sort of mirror image. Just playing around with this kind of problem can be quite enlightening if you're not the kind of person who instantly sees how these systems work.

So let's do some summaries, and then at the end of the summaries, I'm going to show you a really awesome player. A really amazing version of this thing that has the headquarters telling it where to go. So this is going to be at the very end of lecture, I'm going to show you. It's the same as this player that just lost this match, this one up here, only instead of being dumb when the enemy arrives, it's going to be smart about where it puts itself with respect to the enemy. Which is to say that it's going to avoid the enemy unless the number of nearby allies is a really high number.

And at the same time, the central headquarters is going to tell it to capture encampments and to build suppliers and generators on them. So you're going to start to have a player that, just from few little pieces, is starting to do everything it needs to do to start winning matches. So I'll show that at the very end after I

summarize some of things that I've done, and some of the things that I think you ought to try out. Because I can put this code together in a certain amount of time, but I can't try every possible combination.

So let's go straight on over to here, and I want to just talk about two categories. Let's talk about the shape of your system, which is to say the cloud. Let's talk about how you achieve the shape. And you can achieve it either using a headquarters, and I'll talk about all the amazing things that you can have the headquarter tell the group that you might not have thought of. And you can achieve it using this group mind.

So let's do the group mind first, where you could check the neighbors and go to the area with the fewest. I sent around some code. I don't know how many of you used it. And it looked like this. Let me show you my lecture three robot player release.

It's not incredibly good code, but you can see the point. So when these guys get close to one another, each one has this list. And this list shows the number of adjacent enemy and allied units. So that list is absolutely gosh darn tiny. So I'm going to go ahead and do a fancy way of zooming in.

Oh, my goodness. That's even less legible. That says 22, 21, 2, 0, 0, 1, 12, 23, me 12. So me 12 means that I have two allies adjacent and one enemy adjacent. And adjacent this number, these are eight different numbers indicating north, northeast, east, southeast, and so on, around the clock directions. Although not with 12, but with eight. Yeah.

And it has this many enemies and this many allies. And so I've written the code for you that will do this. And so the robot can sort of start to make decisions about who is adjacent to it. All you've got to do is start sorting how good it is to be next to allies or enemies. Like maybe it's really good to be next to a lot of allies and very few enemies. Otherwise you want to be close to few allies so that you spread out.

There are a lot of things that you can build on that. So that's why I included that code. So you could check neighbors and go to the area with the fewest. You could

compare to neighbor repulsion. So the repulsion is accomplished in the following way.

So I've written this code here in guided swarm. And here's where you are repelling from the enemies. So it isn't just one way of doing it. A lot of ways of doing it. But I've called this function freeGo, and it lets you sort of freely float among your various alternatives of places to go.

What it does is it says, all right, here I'm starting at my location. I'll get a direction to the target location, which is like the waypoint. And now I'm going to apply a target weight, because maybe I really want to get there if I'm far away. But if I'm kind of close, I don't really care. So now I have this function called target weight, where if my distance squared is greater than 100, I'll call the weighting 5. If my distance squared is greater than 9, I'll call the weighting 2. And if I'm really close, then I'll call the weighting 1.

So that's sort of giving me this spring-like behavior to get to where I want to go. And it's going to give me this gassy behavior. So what I'll do is I'll make a goal location, which is not the same as the target. So that goal location, I'm going to just add to my location this direction to the target. So let's do it on Paint to give an example. Because saying words is not always incredibly clear.

So here's me. Here's my waypoint. And maybe here is the closest ally. So what I'm going to do is I'm going to repel from the ally, and I'm going to attract to the waypoint. Which means I'm going to end up having a bunch of guys that are having competing requirements.

Competing requirements is like everything in these kinds of dynamic systems. Because in the end, you'll want to expect them to find a balance between these competing things. Because frankly, they want to do things that are inconsistent. They want to be at the waypoint, but they don't want to be next to their friends. So that automatically means that they can't be satisfied, like a lot of people I know.

So you could see here the repulsion term. So if there are allies, then it will locate the

closest ally using this function that I wrote. Very simple. And then it will add to the goal location negative 3 of the direction to the closest ally. so what the heck does that mean? That's just saying repel.

So my goal location started here. And then I moved it toward the waypoint some amount. And it would be more or less, depending on where the waypoint was with respect to me. And now I'm going to take the ally direction, which is this direction, and I'm going to subtract that from this. So now I'm going away from the ally and toward the waypoint, which gets me this x.

So in the final analysis, this is my new goal location, which is the sum of a bunch of weighted added directions. And now I'm going to go that way, which is going to get me both toward the waypoint and away from my ally. I mean, on a grid, this direction may have been quite close to or the same as the direction directly to the waypoint. But when you keep doing it over and over, those little edge cases where a small difference makes you go one way or the other start to become significant.

And so you can do it just simply like this, where you only care about where the ally that's closest is. And you only care about the weighting of where you're going. But wait just a second. There are so many things you can add to it, and that I did add when I wrote my guided swarm two, which is going to end the lecture.

So let's do all the things that you can repel or attract. There's enemies. And you repel or attract to them based on how many there are. There's mines . And if they're yours, do you want to be right next to your mines? Do you want to be behind them? Do you want to be sort of in a clump of them? Do want to keep laying mi-- I mean, if you're in a laying mines mode, then you don't really want to stand on a mine, because then you can't lay one. You're on one already.

So [INAUDIBLE] on your current goals. So you could imagine making the following-- this would be so cool. Somebody in this room needs to do this. I was going to do it for today but ran out of time. You do the following. So here's what always happens when I write players, is I've got a center, and people start laying mines around the center. I think you saw this today, where they were laying these mines. And the

mines reached a certain amount, and the robots were only out to here. And so that was it. I didn't get any more mines.

But here's what you could do. You could say, have the headquarters, which might be, say, down here. Have it say, if-- let's do this. Oh, my goodness. Watch this. Oh, uh, uh. It's going to work. It's going to work. Headquarters. If there are a lot of mines near the rally point-- and it can do this very easily.

You know how to do it? There's a really, really handy function. Don't do the following. Here's years what you were probably thinking. What you might have done is sense all the allied mines, or whatever the function name is, and then parse through that list and find the ones that are close the waypoint. You don't have to do that. You don't have to do that at all.

You can see here in the documentation under Robot Controller, there's a method that really helps. There's senseMineLocations and it will let you specify a center, a radius, and a team. So let's put that on here. So you definitely want to be using this so that all you've got to do is have the headquarters.

And why would you use the headquarters? Just so that it doesn't have to be done again and again. So the headquarters is going say, how many mines are there when this x is the center, and I'm looking within some radius? And you could just expand the radius turn by turn to see how many mines there are until it starts to drop Off or you could just like set it at a given radius, or the previous radius.

So yeah. Let's do it that way. And you would say, how many of my mines are in this circle? And so you'd say, if there are full mines in the circle-- and you can do that pretty easily because the circle is defined by a radius squared, right? So expected mines is like pi r squared, right? That's a number of tiles. But r squared we already have, and pi as an integer is equal to 3.

And so we can just say the number of expected mines equals 3 times the radius squared. And so that's also nice, because it's a little bit less than pi. And so if you're like mostly filled of mines in that circle, then-- so if the detected mines is greater

than or equal to expected mines in a full area, then expand the mine radius.

And you might say, well, wait. How am I expanding the mine radius? Well, here you go. Here you go. Right now, the function looks like this for the repulsion and attraction. It looks like the following. Oh, I'm going to use this. Yeah. Like that.

So if this is the distance x from center, and this is the repulsive force-- or I think we were attracting. Yeah, we were attracting to the center. The attractive force was greater the farther you got from the center. But you could imagine-- So that's going to push everybody toward the middle.

But you don't have to do that. You could just as easily have the headquarters-- the headquarters could be telling everybody by message systems what this looks like. And they could very well tell them to make it a little bit more like this. So let's say that you already put mines out to this range. So you've already put mines here and you've detected them using the headquarters.

So now you want everybody to show up here. Well, you could very simply do that. All you've got to do is make the forces look like this. Oh, is that right? No, we would want-- Yeah, yeah. Well, that's the energy. And the force is this. The attractive force is this. So this is exactly a spring. That's a spring from physics.

And that will make them go toward it if they are inside, because they will go opposite the direction of this vector from the origin. And they'll make it go away if they are in the other direction, and you get the idea. So that's a pretty neat idea, because you could have them spread out in a circle and maintain that sort of positional authority, as it were. Very, very cool.

We are also going to talk about some other awesome things the headquarter can tell them, even though our mouths are watering. Check the neighbors, OK. We can avoid this. We can avoid mines. We can avoid the enemy. We can do those things.

There's no more direction. We've talked about that. So now let's go back a little bit and talk about how the headquarters can help you achieve a given shape. The headquarters can signal when the enemy has artillery.

Now, how's the headquarters going to know? Well, it could receive the message from somebody else that notices his hit points have dropped by 20 or 40. Or it could just locate an artillery. Like if you actually physically walk past one, then you could say there is an artillery there. I should start repelling my other units. They should start spreading out, because if they are all clumped together, they're going to take a lot more damage.

They're going to take, instead of 40 damage for just one hit, they're going to take eight splash damage as well. And so that's going to end up being an additional five times worse. Yeah. Well, it's going to be four times worse. Five times as bad, as it were. That's definitely no good. So yeah, I mean, that's how you're changing the shape of your cloud using the communication that you're giving from the headquarters or from somewhere else.

So you could do that. You can say, when the group-- and you could say like average position-- has arrived at a waypoint, like at a waypoint that the headquarters has specified, then go to the next one. And right there you can start to fill the whole map that way.

Now, this is a really crazy trick that I think is the bee's knees. And it's just going to make the difference between amazing code and non-amazing code. So everybody is going to have had this situation. And I've had this situation for a long time, but I very rarely-- in fact, I've never come up with a solution until just recently. And it was the following.

OK, so here's me. Here's me and here's the goal. OK I, want to go from one to the other. And a lot of times what happens is I've got a large group of people, and they end up splatting. They just splat, and then they trickle. Like this is a little thin line of trickled units. And everybody else is just splatting there, and they're really unhappy.

Yeah, it's no good. I think this guy here has gone to sleep. I mean, this guy started to tunnel. And so he's like mining in there. And then when these guys arrived, he's like, I'm tired, I'm going on a workers strike. So I mean, it's just terrible. So here's an

alternative to that that may make the difference between winning and losing. This may just become the dominant strategy. I couldn't be more infatuated with my idea.

And it works like this. It works like this, where I'd say, OK. I'm headquarters. I'm headquarters back here. And I'm telling them what to do. All I've got it do is look here, here, and here, and I can use the senseNeutralMines. I'll use the senseNeutralMines, and I'll put these as the centers. One, two, three, four. OK, five.

And these are the directions that the whole swarm could move. And I'll say, OK, this has seven mines and this has two. No, this isn't realistic. They all have zero. OK, they all have zero. Right? But then once I move down a little bit-- come on Paint, come on Paint. Help me out here.

OK. Now I moved down a little bit, and now some directions really look bad. So what I can do is tell the whole swarm to follow a waypoint set that looks like a giant robot. Not like a gundam, no. Like one unit of robot. So the whole swarm becomes one big thing that can all path around the object, because I've down sampled the terrain.

This down sampling idea can extend to full search and pathing algorithms. So you'd split it up like this or so, something like this. They can be overlapping or non-overlapping down sample boxes. You can use blur or other features as standard concepts in computer science. So that you can start to do pathing on a much simpler matrix.

And this can get you to your enemy like leagues faster than you would otherwise get. And if you use a heuristic, then it doesn't even matter. Let's say there's like a few mines here. Well, maybe you'll just barrel right on through. If you can just compare to the amount of mines that you're going to find here versus here, what better way to make that comparison than to count mines in a giant area like this?

And these giant areas are going to be so computationally simple for you to compute, because it's just one function call. And I believe that function call only costs you like $100. And by dollars, I mean byte codes.

So this has been a summary of a bunch of types of swarm, but it hasn't been a

complete summary yet, has it? Because I told you I was going to show you that amazing code that I wrote. And this time I may or may not be being facetious. And that amazing code where I've made this guided swarm two, and I'm going to play it against a basic player on maybe a couple of maps.

Basic player is not horrible, I guess. I don't really know how good or bad we are expecting to claim that it is. But let's go ahead and show this match. So here I've got guided swarm in red. And he's starting out by building a checkerboard pattern of mines.

And what he's going to do is when he encounters the enemy, he's sometimes going to just rush straight at the enemy. Other times he's going to sort of back up and then only fight at the right time. In this example, it looks like he was destroy-- oh. Well, the enemy built artillery right next to my base. It doesn't count.

[LAUGHTER]

PROFESSOR: OK, OK. This one, here I am down in the bottom left. I'm going to get ready to research pickaxe. I'm going to do it. And then when I research pickaxe, their mining pattern is going to change. They're going to spread out, I think. Or maybe-- I might not have implemented it.

Well in any case, it's there. It sort of exists in my mind that I'll do that sometime. So what I'll do is when they arrive here, I'll totally mess them up. They're going to show up. I'll back them up just a little bit. And then they're going to be on my mines and they're going to have nothing to say about it. They're going to just-- they're going to-- oh, so there I go. There I go. I'm starting up to move out. I'm starting to continue to build mines. And I'm going forward.

I don't know that-- I think he doesn't have really any robots. Yeah. Yeah, super effective. Oh my goodness. That was fantastic. I'm actually going to do-- I wonder if it is working as I was expecting. So in this one, I'm going across. And when the enemy shows, what I'm hoping is-- oop. It already ended. Yes. I'm hoping to sort of back up just a little bit, and then take them on. No, it looks like I just run straight in.

[LAUGHTER]

PROFESSOR: Well in any case, I sort of spread out. I made this neat little pattern. I'm starting to take these encampments. It's getting done. I'm doing it up. Making it happen. I take the other side. I take more encampments. And then I just sort of-- because these are like, now I've got better production than he has. I'm going to cast these games this well when we have the finals. It's going to be great.

I'm making more units. Oh, look at this. Do you see this like micro? I'm backing up a little bit, letting them do it. I'm do some micro here, making sure that I outnumber the enemy before I go in. I'm not keeping it so spread out as I was before. Some guys are doing some important philosophy.

[LAUGHTER]

PROFESSOR: And those philosophical results just very well may make the difference between our salvation as a species and not. Thank you very much for attending this swarm lecture. Next lecture will be about strategy. And I hope you enjoy the Indian food.