PROFESSOR: Are you guys excited about Battlecode? I'm so excited to see this many-- Yeah! Yeah, let's hear a hurray-- no, no a yeah. Are you excited?

AUDIENCE: Yeah!

PROFESSOR: All right, that was a strong yeah. I actually mean that. I have a strange thing where when I say something it often comes out like I meant the opposite of what I said, like I didn't believe it. Like that time that Drew Houston-- from Dropbox, he's the Dropbox CEO-- do you guys remember this? He was giving his sponsor talk at the finals of Battlecode last year and he was just wrapping up, and everybody was applauding, and I came to the podium I was like, thanks, thanks Drew. That was really good. That made me want to work at Dropbox. If I didn't have a job already I would. And nobody believed me, they thought I was just being facetious. So, now he hates me. That's too bad. That happens to me.

OK, let's get started soon. So you're all here because you're interested in Battlecode. These lectures are optional. They're geared at beginner to mid-level coders, but they'll be relevant even if you've had coding experience in a million languages. Because we're going to be talking a lot of the time about Battlecode, the game itself. And it could just be a way to keep yourself up to date on the latest metagame, for example.

So we're going to go over the specs today. I'm going to introduce the competition. And then after lecture today, we'll be having a lab, during which we'll help you set up. So I see a lot of you brought your computers, that's fantastic. And that's also great because it means if you get bored, you can just watch YouTube videos. Yeah, so make sure to bring your ear plugs so that you don't bother the guy next to you,

because he's already heard Gangnam Style. All right so be respectful of those guys.

So here's our introduction. My name is Maxwell Mann, I'll be your lecturer for the next 10 days. It actually is minus 4 days because of various things. So it's like the next two weeks, you can come here at this time-- or slightly earlier, so you can hear a lousy joke --you can come here, and you can hear me talk about the game, and hear what the devs think about what you might need to know. And then you can eat food at 6:00 PM, when we have massive quantities of food coming, and then from 6:00 to 7:00, you have the option of staying for the lab session, which has been described, in anticipation, as a social utopia. And I really believe that it could be one, because everybody here has, sort of, some similar interests.

So to introduce us, we developers, are students, or alums, who have worked on developing Battlecode. I'm Max. Sitting here in the front is Aaron Epstein and Steven Valdez. You'll be using them for help, all the time, if you need it, or just to bother them because you feel like it. They're here for no other reason, and to maintain the code base and these kinds of things.

The website is battlecode.org and we have a forum there, where you could also go to get help. And I also, particularly, would like to direct your attention to the IRC channel. It's #battlecode on Freenode. So I don't know if you know what that is, but it's basically a way that you can talk to us on a real time chat. So you can do that at pretty much any hour of the day, because the way we're setting it up this year is we'll all be available, pretty much the whole time. OK, make sure that if you are an MIT student, and you want credit for this class-- it's a six credit class --you just have to sign up on WebSIS for 6.S187. How many people have signed up on WebSIS already? All right, that's like everybody.

Let's talk about the goals of the competition. What is a game? What are we doing? I'm sure a lot of you are thinking all right, well, here we go. I've got finals tomorrow but I have to decide what IAP classes to take. Gosh, this is a terrible combination of deadlines. Then you thought, should I take-- should I to do battle code? Should I do one of these other things, like web programming, or game programming? What is

the operating systems whatshumadoosits? Should I do this or that? And you're probably thinking to yourself, what is fun? What is valuable for me? What am I going to learn? And we ask ourselves the same question.

A couple months before then was, what do we want the Battlecode competition to be like? And what we ended up deciding, was that the game needs to be fun and it needs to contain classic battle code areas of challenge in artificial intelligence. So this is a game that is a cross-- we've always said in the past that it is not StarCraft -- but it is a cross between a real-time strategy game and an AI programming competition. So what we wanted was the game to be easy to learn, but hard to master. So what you'll see when I describe the game specs-- and I'm going into it just momentarily --when I describe game specs you'll see ways to play the game, and then ways to excel at the game. So it's like you've got this infinite ramp, that you can run wild on, of how much effort you want to put into developing the best player. Or better yet, because what is the best? It's comparative. That's comparing to somebody else. But that's kind of irrelevant. What you want is to have fun. Anyway, you know that. So there we go.

Our idea was to not have sort of skill caps and limiting factors, like maps that are just covered in maze so you can't do anything unless you can get past that maze. Like, OK, you didn't pass the test, I hope you enjoy your starting area. No, no. So not that. So let me show the game. How many of you have seen Battlecode before? A show of hands. So that's a good number, but it's not everybody.

So I'm going to show you a game. So this is when you've made your code and you've put it into the client, you can run a game like this, and we'll go over how to do that. And I'll just hit play here, and you'll see the red team in the upper left, and the blue team in the bottom right, playing a game of Battlecode for this year. The player on the upper left, in red, is placing mines, and capturing bases. His soldiers are moving around and fighting with one another. Fighting the enemy, hopefully. And the blue player, here at the bottom right, is just sort of shooting his guys off to die at the enemy army. On the left you'll see indicators of various game constants and values that are changing. If I pause this for a second, you can see by clicking on

any robot, you can see his ID number, what robot type it is. You can see his hit point level, which we have characteristically had a very difficult to remember name for. We always have different names for their energy and their head points this year it's going to be, "energon," for hit points. I think this flux number is actually no longer relevant, and you can ignore it.

[AUDIENCE LAUGHTER]

PROFESSOR: See, we worked so hard on making this polished and perfect, that we weren't finished until-- actually we're not finished yet, they're still fussing with something.

[AUDIENCE LAUGHTER]

PROFESSOR: The bytecodes used is going to be a key factor that you're going to look at so that you can determine how much computational energy those robots are spending. This year, there's a very big impact on the efficiency of your code, and the performance of your player. If you have efficient code, you could have twice as many robots on the field then you can sustain if your code is inefficient. Because the actual computational cost is recorded, and is then implemented into the upkeep of each unit. Finally, you'll see the location of the unit, and three indicator strings that you can use for debugging and control purposes.

So what we'll do today is talk about what kinds of things are on this map, and how that's different and amazingly great, and how this year's new gameplay will be even more fun than it ever has been. And I can say that with some confidence, because it's quite different and demonstrably better. Yes. So what we think of as skill areas, areas that make a fun game for Battlecode, are, navigation-- but not that you so much require it --see, in this example, they didn't really need any complex maze solving algorithms. What they were just doing, is there are these neutral mines that are here, and you could either walk around the neutral minefield, or you could just clear yourself right on through it. So there's navigation, but you don't necessarily have to do it.

[AUDIENCE LAUGHTER]

PROFESSOR: There's group pathing-- and that's a good point because, you know, OK, anybody can take one unit and after enough time, figure out how to implement any of the existing pathing algorithms. But let's say you've got multiple units. They start to get in one another's way. Let me give an example that's very concrete, and that hits really close to home, for me. I did this all the time-- I've done Battlecode, I guess, like four years now.

And so what I'd always have is, I'm here, and I want to go to ice cream-- something desirable --and I'm trying to get past this wall. So I go here, but then all my friends take up all this space because they're going the same way. And then I'm trying to turn around, and go back, and they're trying to go this way because they think it's closer to the ice cream. And we just end up here all day, and then the enemy comes by and shoots us. And then we've got holes in ourselves, how are we going to maintain that stomach area to keep the ice cream in? It's just not going to work.

So we like navigation, we like group pathing. We like messaging. This year, the messaging system is totally new. In the past, what you'd need is-- I'm going to go back to the screen here --is your robot would only be able to see a certain area around himself. And so every other robot would not know what's going on, on the whole map, except by messaging. But even then, they would have to receive messages that are broadcast within a certain range. So you end up having to make these chains where you'd say, all right, I've got, let's see, Joe, and James, and Jalopy. They're all in line, and these three robots are in line, and they each have some broadcast radius. And so maybe I can, sort of, set up a relay chain, and they're broadcasting back. And that was how you used to communicate information about where the enemy is.

So here's the ice cream, here, and here's me. And I would have to hear it from these four guys. But that would end up being a lot of work for me to implement. And we like that, we think that's really cool. But at the same time it ended up meaning that if the ice cream were an enemy, and I needed to scout that the enemy were coming with some rush, that what would often happen is that there wouldn't be a chain established. Because this kind of stuff's kind of hard do. So this guy would see

the enemy right here-- and here's me, back here, I'm just minding my own business. Maybe I'm building a staircase, which we did in one year. Or some other very useful expenditure of effort and time. So this guy sees the enemy and I don't know about it, I can't prepare until he comes back. OK, so then he comes back to within broadcast range-- he's here --but by then the enemy is right there. I can't do anything.

So what we've done this year, is we've said all right, realistically we've got these radios, right? Let's communicate, on these radios, across the whole map. So now your radio messages can be heard across the whole map. But, instead of being messages, like they used to be-- and these messages were kind of annoying, they were these constructs that you had to assemble. You'd say, oh, I'm making a new message and I'm applying these parts to it. And not every message has every type of part. And not every part as every type of message. And every time I receive a message I can receive an infinity of messages. And then my robot explodes because he's being overloaded with messages. That's like spam, happens a lot, it happens to me.

So what we get instead, is we said, all right, we've got a uniform message channel. We've got 10,000 channels. And any robot can post an integer to any one of those channels, at any time. So then my robot could be posting-- maybe my code is using frequency 567. And I'll be communicating on that frequency specifically where I'm going, or what direction I'm going, or something like that. And everybody on my team can see it. So that's a really useful sort of centralizing feature. So if the enemy is coming, I can just look on integer number 567, and if the enemy is coming, maybe that's a flag, maybe I make that, I guess, 666 because that's really bad. Right? So I make that my token for enemy is coming, and then bing, everybody can start preparing for the enemy coming.

But there's more, I mean, let's say you're going to metagame them. You're going to think about what could happen from a high level. Like when would the enemy just invade? When would they attack me? I mean, I don't have to see them coming to know that they will. For example, maybe the map is really small. So this year, you

can tell how big the map is. I mean, how does it make sense-- this happened again and again in previous years --how does it make sense to have a game, where players are expected to operate intelligently, in the absence of information. OK, I understand that sometimes a little bit of uncertainty can be great. Like how great poker be if everybody knows what everybody else's hand is? That would be really lousy, right?

But here, we just had almost no information, so you'd have build-order losses. This is sort of a big problem, in my eyes, with StarCraft. Which is that you'd start with a strategy that could lose before you have any information on how to fix that. So at least in StarCraft you know how big the map is.

So here we're giving them the following centralized information. So here's what I'll do. I'm going to list out the centralized information that you can use, that you didn't really used to have access to. Now you might ask, all right I'm not going to memorize this, and I'm not taking notes-- even though I did bring a thingamabober, which is a computer, or a notebook. You might say, I'm not going to memorize this, I'm not interested, I'm here to have fun. Well look here, you can look in the Java docs and you can see all of the robot abilities. They're listed in this giant list here, and you'll have time to peruse those. And we'll sort of go over them over the course of the few lectures.

But having them sort of in a condensed form or organized into topics, like what is centralized information? You know, what information can I use? Ends up probably being a bit of a convenience. So I have a couple of convenient organizations of data. And that's really all I'm here for is just-- you can get all this information in our documentation, which we'll supply you when we help you set up the game. And/or you can sort of hear it from me, where I've got it sort of organized into a slightly different way.

The centralized info that you get includes, you can sense the positions of all allied robots. So there's a function, and again, we have very good naming conventions so that all of the names that we pick let you know exactly what we're referring to. And

for this function we call it-- it's a method called, "Sense Nearby Game Objects." And that's great because it can detect game objects across the map, anywhere, they don't have to be nearby. So that that's really nice. And it senses these things, which of course, you would want to sense. They're objects! You know, I was playing chess the other day with my friend Rick, and I said to him, you know what I could use more of is game objects. I can sense all of your game objects. And he didn't get it.

[AUDIENCE LAUGHTER]

PROFESSOR: So Sense Nearby Game Objects is an overloaded function, which means that you can call it with multiple arguments. And that really does help you. So the point is, this is a global function, because you can see any game object if that game object is of a certain type. So all of your allied robots you can see. Let's go back to our example, let's say that I am back here, that's me, over here at the bottom. And this guy up at the top, I can see where he is back from here. But my sight range is actually only here. So I've got different detection abilities. Allied robots, allied mines, and neutral mines, are all visible no matter where you are, but enemy robots are only visible within your sight radius. But it's within any allied robot site radius. It's harder to explain than it is to think about. In this picture, this robot can see these guys, because he's really close. But this guy can also see them, because this robot is nearby. Right, that's pretty straightforward.

One person was nodding but he was also yawning, so he may have been nodding off. No, I'm only joking. Yeah, zing!

[AUDIENCE LAUGHTER]

PROFESSOR: Sense Nearby Game Objects is a kind of useful thing to be able to do, but it's not always obvious how to use it. So, later on, I'll be talking about how one of the inputs to this is what type of game object you want. I still haven't-- I've been saying game object over and over, but none of you care about game objects. You care about enemy robots, I want to shoot at the enemy robots. So you would end up writing, oh, I want a robot please. So you tell it, give me game objects of type Robot.class. And you would say, all right, this is getting kind of confusing. I'm feeling a little bit

overwhelmed. Well don't you worry about it, because here in the Java Docs, under package-- and you might notice that this is the Java docs version 7, for Java version 7. It's real sleek. We're distributing this at absolutely no cost. Yeah. Let's see here, oh, look! It's game object. There it is. And oh, look, subinterfaces include encampments and robots. Well that's great, that answers my question. So now I know that I can put Robot.class or Encampment.class and it will give me that type of game object. Yeah. Yeah! All right. [LAUGHS]

So that's a piece of centralized information. Another piece of centralized info-- so we've talked about how it's allied robots. There's allied mines. I'll talk about the mines and how you can lay them and defuse them, and how you get these things onto the game board. Allied robots, allied mines. You've got the radio system, which is entirely global. And that means that you can disrupt somebody else's radio program.

I have a hunch that there are several people out there who would broadcast random numbers to my favorite channel, 567, and it would just totally ruin my player. I wouldn't be able to work at all. Now, in order to limit this sort of-- there's sort of a back and forth between sending messages that you want to send, receiving the messages with a certain amount of, sort of, redundancy in there, so you can be safe against message jamming, versus actual jamming itself. How do we assess the balance of these competing objectives? Well we've got a cost associated with everything.

There's no free lunch. Except dinner for Battlecode. And that is actually free. We ordered so much pizza for today. Now I'm sure not many of you ordered that much pizza before. But let me tell you that I expected that-- there's like 120 people here in the room, and we thought how much pizza does everybody eat? Well unfortunately Battlecode is a male dominated class. So we'll have slightly more pizza consumption than otherwise. Is that sexist? And so--

[AUDIENCE LAUGHTER]

PROFESSOR: So we decided to get 50 pizzas. And we were like, well what's awesome pizza? We

like Bertucci's. You guys don't like Bertucci's. Yeah! He likes Bertucci's. Yeah, so that was like $20 a pizza. So it was like $1000 for pizza.

[AUDIENCE LAUGHTER]

PROFESSOR: My goodness. Well anyway, it's going to be really good. And it's coming in like a half an hour. That's fantastic-- I've totally lost my train of thought.

[AUDIENCE LAUGHTER]

PROFESSOR: Right I was

AUDIENCE: --message him.

PROFESSOR: Yeah. I was saying that there's no free lunch, and as such, we have a bytecode cost. Now, I think in the past it's not been that obvious to me what the cost of doing a certain computation is. Can I jam the enemy? Is that really going to work, or am I going to run out of money? Is a question you could ask.

So here, in the-- let's see here, I have a place where I have installed Battlecode. And you're going to do something just like this, where you'll have that Battlecode-- that's OK, he had to go to the bathroom. Don't worry about him.

[AUDIENCE LAUGHTER]

PROFESSOR: Well it's a reasonable supposition, he is human, isn't he? All right, and so here, in the Battlecode directory where I've installed Battlecode, there's going to be a text file that's called method costs. Bingo.

All right now it looks disorganized because everything-- the strings are different lengths. But each string represents a function that you can use. Like for example attack square. Or maybe broadcast a message. Ah, that was the one we were interested in. OK, let's look at that.

Broadcast costs 25 bytecodes. All right, let's do a little bit of math-- arithmetic, it's not math, is it? 25 bytecodes, all right? There are 10,000 strings, which I wrote and

erased. 10,000 places to write a message to. There are 25 bytecodes to broadcast a message.

OK, so that means that I can broadcast 400 messages. Is that right? Is that this number divided by that number equals that number? Is that-- Yeah, yeah. Yeah, let's go with it. Yeah, let's go with that.

So you have 400. No, but maybe you have more of your robots jamming signals. OK, how many robots can you have? All right, let's do that computation. I'm pretending that I'm you, and that I'm just like-- OK I'm not doing a very good impersonation. I recognize that. You'd be much faster at this.

So you would want to find out how many robots you can sustain. Can I get somebody to suggest one way to figure out that number? Yes! You, you, you. Ah yes you could make a player and just spawn robots until they start dying.

Let's say you wanted to do it in a different way.

[AUDIENCE LAUGHTER]

PROFESSOR: I have the example here. What is It? Game constants, there we are. So here I'm looking at the Java docs, and I can click on game constants. I'm looking at a different version of it that didn't actually show the game constants in the game constants page. But that's because we are well organized. And this is to show that we're indomitable, indefatigable, as it were.

These constant values indicate, here, that your headquarters is capable of producing quite a few resources. This shows that it produces 50 resources per turn. This resource production enables you to have an upkeep of robots.

So in the past, we've had names like flux, and so on. This year, the name for it is power. That's tough though because we have time as well. And as you well know power times time is energy. And so now we have energon and energy, which is a multiple or a calculated quantity from power. But don't you worry too much about it. It's just a number. And the usage of this resource, or power, by robots, is displayed

in unit upkeep. Which here, is written incorrectly. I'm going to be bring up the most recent version of this document.

What's nice is that you guys will get one version, and then you'll have one version on your computer and not, like, 16. And your one version will be up to date. And when you want to update it, the client will automatically tell you that it needs updating. So you'll just have the one version, and you won't have disaster area.

Yeah, constant value. So this is the more recently updated one. And you can see here the unit power upkeep is one. All right, fantastic. We're ready to do a computation. So I'm going to put here 40 divided by-- so this is the amount of energy, or the amount of power, that's produced per turn by your headquarters divided by the usage of that power.

Now you're going to say, all right, I'm bored. I can do this in my head. Well maybe you can but some of us can't, all right? No, so that would be the number. But that would say that the number of robots you get is independent of how much work they're doing. And that doesn't make any sense. You should get to have more robots if you're doing less computation.

So here you can see, there is a cost. Power cost per bytecode, which is here. And it's 10 to the minus 4. So let's say that I'm using-- 10 to the minus 4 times the number bytecodes. Well, if I'm using zero byte codes, then I get 40. Because that would be 1 plus 0. And 40 over one is 40. OK and if I were using 10,000, which is also known as 10 to the 4. Then I would get 20. Wow, what a big difference.

So that tells you that if you want to go ahead and spam the message system-- let's say you're spamming, so you're using all of your bytecodes possible, and you have 20 robots. So 20 robots are spamming 400 message integers per turn. So you're doing nothing but spamming the messages. And you're getting eight, zero, zero, ze-- Oh, ahh, ahh. Oh, it's not even enough. It's not enough.

Well, so that shows that you could jam the enemy. And maybe you should, if you've got nothing better to do, if you derive a personal satisfaction from it. But this is worth

saying. These kinds of calculations are important for strategy. But don't make them totally fixed in stone. Because we change the game, we tweak it a little bit for balance. So if it ends up being that 10,000 is just so much radio space that we just would rather not have it, then it could just get knocked down to 1000. And then all of a sudden your player, which depends on this or that messaging construct, may have to be somewhat rethought.

So this is a good time to talk about the organization of the competition. Is it? I think it is. Let's go to the Battlecode website. And let me show you something. I don't know I knew that you could do this. But previous years we've had like a list. And it'll be in text form, and it'll say on this date we have this competition, and this one.

And then you have to open up a window of this. And then every single one of you opens up another window with Gmail or Google Calendar. And then you like manually enter in the-- do you not do that? Who has to do that whenever you have a competition, you want to know when things are? Yeah, yeah there's like two people who also have to do that.

So I figured, in light of saving those two people some effort, we made this Google Calendar. And you could see, here's today's classroom session, here in 1-190 from 5:00 to 6:00 PM. And it's got the lab written there. So the specs and software are released, as of 37 minutes ago, roughly speaking. And what we'll do, after this, is we'll talk about how to set it up. So there you go, specs and software are released.

The sprint tournament is on the 14th. And what that means is you can submit a player for that, and you can be entered for a relatively small prize pool. Relatively small, the actual prize pool for the finals is I think $50,000. So relatively small is just some fraction of that which is still non-negligible.

Anyway, so that's going to be really exciting because that will give you the opportunity to test your code out to see what other players are doing, because to some extent winning is going to be related to what other people do. But more importantly, having fun is related to what other people do, and talking to the other people.

Like this social utopia of mine, the idea of this lab, is totally new-- as well as the Google Calendar --to try to make things more easy and fun and together, among us as Battlecoders. Because we do all share that common link. Yes.

So the point is that if you didn't know you could do this, what you do is you go to-- if you want to add this to your own calendar, there's like a button where you can go and do that. I think the button isn't available because-- Yeah, there. There? At the bottom. At the bottom? Yeah, there you go. And then you just add it right in, so I've already got it here, in my personal calendar. And there you go, so you can see that I'm having lunch on the 24th. And OK, so you get the idea.

So the sprint tournament is-- the submission deadline, you can see here, is on the 14th. On the 16th, on that Wednesday, in class, we'll show some of the games from the sprint tournament. I don't think we'll show all of them. See because what used to happen, is we would show all the games. And there's like 100 teams. And that ends up being a relatively large size round-robin bracket with, I think, double elimination.

So it's a ton of games. I think it generally takes like five hours. But it ends up bumping up to six, seven, or eight hours because we end up showing up there, and the tournament code that sets it all up doesn't work. And so it takes forever. So we're going to do this year is we'll just sort to distill it down, we'll have something simple to show that's fantastic and exciting on the Wednesday.

And in the future, when we have our later-- our seating tournament, which is sort of the same kind of thing, but it's more important, because it determines your positioning for the final tournament. When we have this seating tournament, we're going to live stream it on the internet, so that you don't even have to-- I mean, you guys all trekked out here in the dead of night and 5:00 or 6:00 PM in the freezing cold. I mean there could be people out there dying because they tried to make it here and they didn't make it. That would be horrible, that's not something to laugh at.

So you came all the way out here, but we don't want to make you have to do that

much work. Because we want you to have fun. So there you go. We've got a calendar to save you that effort. We have lab sessions to save you the effort of trying to make friends, and we'll have live streams in the future, which will make that much better.

So as you can see here, it says the seating tournament submission deadline is on the 21st. And the seating tournament will be on the 22nd. So what that does, is it gets you your position in the finals tournament. The finals submission deadline is different, though. So you can continually update your player.

My point is that you only have three weeks. It's the 7th today. And the finals submission deadline is the 28th. You have three weeks to make an amazing player. And each week, you'll have the chance to-- you could upload your player at any time, you can do scrimmage matches with other players at any time. And then finally, you'll be able to submit to the finals.

I mean, I don't think it's worth getting too hyped up about this final thing because after all, in my experience, game changes at the last minute screw you over at the finals, and ends up being a crap shoot. No, no, I exaggerate. But the point is, the fun should be had as you go. And I am here to help with that.

So what you need-- so just one more thing about this administration is that I think that it's required that you upload a resume, and that you be an MIT student if you want to earn-- no, that's not right. We have international competitors, this year. We have sort of a thing going on with the University of Sussex.

If you have friends at other universities, they can feel free to join you. And they don't have to be MIT students to either play or win money. And if they're in Honolulu, or Alaska, we'll fly them in here for the finals because we expect the finalists to show up at this dinner that we have at the Hyatt.

We have this fantastically lavish dinner. I think we paid like $5,000 for it. Aaron, could you come up here and show what the sponsors are by just turning your back to the audience? No, actually there's a button here, I can do that.

AUDIENCE: [INAUDIBLE]

PROFESSOR: Yeah, that's a good point. OK, so here are our sponsors. We have these fancy company reps show up at this dinner. So if you're among the finalists, then you show up at the fancy reps dinner and they offer you jobs and stuff. It's apparently very nice, I've never been.

AUDIENCE: [AUDIENCE LAUGHTER]

Yeah, that's because I was busy. Yeah, that's why. Yeah, that's why. That's why. No. So these are some of the sponsors. Our main sponsors and some other ones. And you'll get a chance to meet them, as well, at the finals. So you don't have to be the best of the best by whatever stupid metric we come up with. What you have to be is-- anybody who does Battlecode is an attractive person to these employers who will meet you and give you business cards. And it'll be a good time for everyone.

And so in that note you have, once again, on the calendar page, the finals are on the 2nd of February. So that'll be when that happens and it'll be in 26-100. It'll be a blast, and you should be looking forward to that very much.

I feel like I've just been droning on, ranting about what I don't like about StarCraft. And I haven't talked enough about the things that are related to gameplay this year.

This year, you have these robots. It's astounding to me, that I-- Oh, is that why it didn't work? Because I was using this one instead of this one, and I meant to use this main client? Yeah, that's the one. OK, yeah.

This was the point I was trying to make earlier. Where I was going to take compute time bot, playing against compute time bot, on this map. This is sort of how to do it, but it'll be done a little bit differently. And so the goal here is to see how many robots can be built. But this guy-- Ah! I think team A and B got switched. So what I'll do is I'll just briefly switch around the code.

So this is sort of the structure of a robot controller. What happens is this code is run

on each robot on your team, and that's what plays the game. I'm sure most of you know that already. And so what I'll do is I'll just check that the team is B. So I can switch the team, hit Save, run it again.

Well it doesn't work as well as I was expecting because these guys are just dying when they go through the minefield. That's interesting.

[AUDIENCE LAUGHTER]

PROFESSOR: You know, you can make your own maps in this. This is a really good example of debugging. And you can say that I did this on purpose. Although if you have a strict adherence to honesty, you might not.

So the maps are located where? You're looking at what, here? So you'll have a directory where you can see your maps. And you'll be also able to see them here. So here's simple. And what if I want to edit this map. Can I do that? Can I just double click it? It's an XML document.

Oh, look. Is it? Oh, data, OK. Here's some data. I'm beginning to get to the part-- see, this would be really easy if I could just do edit with notepad.

AUDIENCE: --source at the bottom

PROFESSOR: Source! No.

AUDIENCE: [INAUDIBLE]

PROFESSOR: Tab.

AUDIENCE: No, down.

PROFESSOR: Down? Down.

AUDIENCE: Where design is.

PROFESSOR: Oh! Bingo. All right! Yeah, thank you. So then, you can see that I can make my own maps using these codes. So the character "3" indicates a neutral mine. So I can just

do like a find replace, were I'll replace "3" with ".", which is an open terrain. I'll just do replace all. And then I'll just make sure to change this one or else the whole map would be mines. I'm sure two of you thought that I wouldn't do that, but I assure you that I'm operating at full capacity.

All right, so this is an example where you can see I've just built bunch of robots. And you can see that this is my power. Your power stored in a reserve bank. And what I'm going to do is suddenly at round 1,000, I'm going to start computing a lot. So right now you can see that this robot is using 38 bytecodes. Not very many.

If you're amazing at counting you'll see that there are about 40 here. And the new robots-- oh, boom! Look at that. The new robots died because they don't have enough-- these guys don't have enough compute-- they don't have enough power to run.

So they lose hit points until they're dead. And then if they all start using code, boom, when we cut down to half the number. And so you could see quite visually the effect of using 10,000 bytecodes versus using no bytecodes at all.

So from here, down to there. Bang. This power quantity can be affected by various things. So there are some equations that I could show for what that is. But maybe I've just gone a bit far field here because I've been talking about some details.

Here's the general overview, right? You don't really need too many of the specifics, because all you've got to do is find the enemy. And there's a function that does that. Oh, look at that, I'll just add that to the global list. There's, "Sense Enemy Headquarters," bingo. Done.

If you were interested in a little bit of masochism, coding masochism, and just doing something the hard way, you could do that. Because what we've also provided, is the dimensions of the map, and you know that the upper left corner of the map is 0,0.

So with a symmetry operation you could say, my headquarters is here. The map size is this. My location is here. Therefore, the enemy must be at this other location.

So more than one way to skin a cat, as the case may be.

So you locate the enemy, and all you have to do is walk up to him. Because this year attacking happens automatically for soldiers. So right now there isn't any fighting going on. But I'll show an example that will show that all they've got to do is go near one another, and then their attack power is distributed among all of the robots they can see.

So this is some old code that we were writing when we were testing the game. And what you'll find is that when robots come close to one another, they make these circles that indicate their attack range. So they're attacking-- this guy is distributing his attack among this one and that one. Similarly, this one is distributing among these three. So his attack power in this version of the game was different.

But right now I believe the attack power is six. So he would be dealing two damage to each of these. What you also see in this picture, which I believe is quite instructive, is the shield robots. Yeah, any robot that's standing next to this shields robot will be given this additional shields parameter.

And so you can see it here in the user interface. There's a green bar indicating health, and a blue bar indicating shields. In the latest implementation of Battlecode, shields serve as a protection only against a certain type of encampment that can be built. So is these encampments work in the following way. You start with just a headquarters, and that produces these soldiers. So you can produce a soldier at any tile around you, if that tile is unoccupied.

That soldier can then be used to construct an encampment. And he can construct any of a number of types of encampment.

So I've made this nice, handy, useful Battlecode data sheet. People from last year might remember that I made sort of a handy unit sheet at that time. But this year it's 100% more accurate than last year's. That's right. We've checked and double checked, but this is the latest version.

So you can see here all of the different types of units. And there's a Boolean for

whether it is an encampment, in terms of just a description. Although, you know, that's kind of not a fantastic description. Because all of these units-- you know, because we like to be straightforward.

It's a realistic preparation for your life as a professional coder, that the database and documentation you receive will contain information opposite to the way that things are run. So you can see here that they're all listed-- some of these are listed as encampments. But in fact they are all robots. Yes, I believe that's correct. They are all robots.

So the headquarters is what you start with. It generates 40 power turn, and it can spawn these soldiers. It takes 10 turns to spawn a soldier. Or it can research upgrades, and I'll talk about the upgrades below, in just a minute.

The soldier has an upkeep of 1 to 2 power per turn, depending on byte code usage. It automatically distributes its attack power among adjacent enemy units. And it has the capability of laying mines, which take 25 turns to lay a mine. Defusing mines, which take 5 turns, and capturing encampments, which takes 50 turns.

Now you might ask, all right, where am I getting these numbers? I'm assembling this sort of convenient diagram but what if the specs change? You know, where do I get these numbers? Well they're all under game constants under robot type enum, or the upgrade enum, and you'll find them all the Java docs.

Alternatively, you could all always ask us where they are by going on our IRC chat, or talking to one of us in person. So you can see here they're laying mines, defusing mines, capturing encampments. And in the act of capturing one of these encampments, which is a pre-existing location on the map, these black circles.

In the act of capturing one of those-- there's nothing magical about it. Because here, among the robot controller methods, you'll find capture encampment. And what you'll do is you'll specify what type of enchantment you want it to be. And you might ask, what types are there?

Well, you could either look at handy chart, or you could click here, under robot type. And this enum robot type will show you the types that you can build. You can build artillery, generator, medbay, shields, and supplier. The headquarters and soldier are different. They're not built from encampments. I don't know if that's written anywhere, but I've said it, so now you know it.

Yeah. So that's how you're going to be able to build these units, that are shown here. So the medbay heals adjacent robots, 2 hit points per turn. You can see that's their quote, "attack power," because attack is a lot like healing.

You know, modern surgery, you can use scalpel and such. These are also considered weapons. I'm sure you couldn't bring a scalpel on a plane. So attack is a really good word for healing-- a healing unit. And if you don't believe me talk to any medicine man you know.

Shields will give all adjacent robots 5 shield points per turn. So if you cluster 8 units around them, then that'll go ahead and give 40 shield points distributed among them, to the extent that 5 times 8 is 40. The shields only protect you from artillery, though. They don't protect you against other soldiers.

Now, these robots that are shielded will lose 1 shield point per turn, as it wears off. But there's no cap on the amount of shields they can have. So they can have, I think, 10 to the 8th shields. Which is like a lot more than their total hit points of 40. Right? So that's effectively infinite.

The artillery is the one unit that can shoot at a far range. It could shoot at a range of 63. Now we always talk about squared ranges, because that makes it easier to do comparisons without having to take a square root. Although I guess if we did give you ranges, and not squared ranges, you could just square it and then compare it to the range that you had just calculated. this way works really well.

So what I can do is I can demonstrate for you what that looks like by opening up-- oh my goodness, I'm almost out of time. I can open up this stockpiling-- no, that's not right. I made it in the map maker program, down here.

Yeah, so this shows you what 63 looks like. So if you're here in the middle, you can see all of these tiles but you can't see the ones outside. But its site range is only 14. Every robot has a site range of 14, As you can see here, in this document.

So the attack range is 63, the site range is 14. So, this is one of those cases where you'd need scouting units that would know what to shoot at. And it's really helpful that you have this global information. Because you don't need to post to radio to tell the artillery what to shoot at. Because he can already see all of the unit that your allies can see. He can't necessarily see the enemies, unless your guys are nearby.

So, yes, the artillery does 40 damage to the target that it specifies, and 20 splash damage to adjacent tiles. And that does include friendly fire, so you've got to be careful about where you shoot.

The generator generates 10 resources per turn. And that resource is power. So that's added on to your headquarter power. So if you build a bunch of generators, you can support more soldiers.

And finally, the supplier will reduce the build time for soldiers by 5%. And that's multiplicative. So if you have two suppliers, then it's 0.95 squared times 10 turns.

This year, we give you the opportunity of building upgrades as well, researching upgrades at your headquarters. So your headquarters is your central unit, the goal is to defeat the enemy headquarters. To do that, you could just walk your robots on over and kill them dead.

It's not required that you research upgrades and built encampments. And that's one of the ways that we make the competition somewhat easier to enter, easier to get something that works.

But if you want to research upgrades, to make a strategy that's tailored around a certain approach, then you can get these. The future upgrade reduces your energy decay because the way that you hold power, is that you can't just mass it up indefinitely.

You get a power income from your headquarters, and you spend power to upkeep your soldiers. But we don't want that-- if you have no soldiers and you're just getting income, we don't want that to go up indefinitely. So every turn that gets multiplied by a number that reduces it. So the energy decay, or the loss is 20%. But if you research fusion, the loss is only 1%.

So then you might think of stockpiling energy and then spending it all at once to get a larger burst of robots, which couldn't be sustained for a long time. But then again, you only have to win one second.

Finally, the-- OK, so then there's more upgrades. Vision upgrade extends your squared vision from 10 to 32. This should say 14, but as you know everything is always up to date 100 percent of the time.

So here's a demonstration of what that would look like. Here's 10 and here's 32. Boom, what a big difference. That's quite a lot more vision.

That's important because you can defuse any mine within your site range if you have the diffusion ability. Which is not to be confused with chemical diffusion, which is something else. Yes, this is for defusing things, and it is spelled correctly.

[AUDIENCE LAUGHTER]

PROFESSOR: So yeah, that extends your mine defusing range. That's pretty important, but keep in mind that you can still only defuse one at a time. So you have to select which one you're going to defuse. And then you defuse that.

Now, it's important to note that you can see where-- I've said that you see your allied mines. I haven't said that you can see your neutral mines as well. But you don't see enemy mines until one of your robots has stepped on it.

So that means that you can't-- I don't think you can defuse something you can't see. But actually I don't know, I'll have to check with that. Because it may be that you could just spam defuse on terrain that you don't know. And that might actually defuse things that you can't see.

At the same time, that's not a fantastic strategy, because defused takes 5 turns. And during those 5 turns you cannot attack. So that leaves you-- or can you attack? I'm thinking maybe you can attack.

[AUDIENCE LAUGHTER]

PROFESSOR: You know what, you can't-- it's a very vulnerable position, to be defusing something. Because you're concentrating on something else. How do you going to perform as effectively on a different task?

Pickaxe is the tool that you really need if you want to lay a ton of mines. Because instead of laying just one mine, where your robot is-- I guess I'm going to have to come up with a system for describing robot-- that's sort of like a robot. Maybe it's a Chinese character.

So that's a robot, and ordinarily he would lay a mine here, but if you have pickaxe, he will lay mines in all of these positions for the same cost of laying just one mine. Pretty useful to have.

And it takes the same amount. So laying a mine cost 25 rounds, defusing a mine costs five rounds. So you could do the math. If you have pickaxe, you could lay them as fast as the other team can defuse them, provided you can get into an open space where you can do it. Realistically they might be able to encroach, and get you during that time.

And why do you want to lay mines? Doesn't that slow the game down? Doesn't that mean that things will just move to slowly, make it a boring thing to watch? I would say no, because during that time you could be researching the nuclear missile. That's right. That's right, we knew that it would be interesting to watch players build nuclear missiles.

If you research that, you win immediately. So--

[AUDIENCE LAUGHTER]

That's yet another fun way to win, and I'm going to show you an example of somebody attempting to win that way. This will only take as much time as it took me the last time I tried to do it. It's under-- golly, you won't have nearly-- oh, it's right here. It must be so hard to watch me, because it's like, oh man, he's looking right at it.

Let's see, so let's put do nothing bot-- because that's a decent match for the nuke bot. Where is he? Is he not here?

AUDIENCE: No, he's in the middle.

PROFESSOR: He's in the middle. Oh, I'm looking right at him. All right. All right, well it's by no means a foregone conclusion who wins this match. This is an old version of the code where nuking took 1,000 turns instead of 400 turns. But that's not to say that the red team will necessarily win on turn 1,000.

Oh wait. It could only take 400 turns wait a minute. Oh, that's 1,000 and this was-- OK yeah, hey! It happened. There you go. So yeah, the nuclear missile has-- I mean, it's not a very impressive graphic.

[AUDIENCE LAUGHTER]

PROFESSOR: It's technically the same one we use for every other explosion.

[AUDIENCE LAUGHTER]

PROFESSOR: But I think what we'll do is for the finals, we've got a new match visualizer. So this was an old version of the code, because of course I wouldn't show you something up to date. That would be that help you instead of hinder you. And what we're trying to do is train you to be amazing. And it's working, I swear.

So what I'll do is I'll show this version of it, which is great, because it shows the thing researching. All right, well later versions of the client that you're going to receive will show what things are researching in the client window. And I think it really would not be in an excessive-- the pizza should be arriving any moment, so I know that the gurgling of your stomachs is totally reasonable and will be resolved imminently.

If I run this version, then I think-- let's see, is there-- so let's put do nothing against the nuke guy. Is the nuke guy still there? He is, OK. So this version of the map viewer shows researches that are going on. Yeah.

And you can see he's researching pickaxe, diffusion, sight range, the fusion thing that reduces the amount of energy loss. He's researching everything. And whereas the other do nothing guy-- you know, you could describe him as a pacifist.

The philosophy that he is developing, using three bytecodes per round, may change the world. So I see no reason to hate on his choice of vocation. I would definitely not-- yeah, no, definitely. OK.

So you can see here that nuke is researching. So in a tournament scenario we could all start getting excited and going ohhhhh! What's going to happen? Is it going to be close? And that sort of thing.

I mean of course at the end it's going to be a bit of an anti-climax because it's going to show a little puff ball, like somebody made popcorn on one tile. But we'll be able to overcome that because we can just switch to YouTube and show a clip of some giant nuclear test.

All right, that's all I have for you now. I mean to say that we're going to show you a whole bunch of things. You can get up and go, I know a lot of you have appointments with very attractive young women and men. And that's totally OK. But I recommend that you stick around, eat some pizza, hear about how to set up Battlecode.

And know that we will have our developer testing code up. The team name, here on the Battlecode website. You can see here, team list. We just made a team that you can use to test your code against. That's down here at the bottom.

It's test, somewhere. Yeah, Teh Devs Test, there you go.

Did you guys make this many teams in the last 2 seconds? You must have. You

know, it's important to multitask.

Yeah, so you can upload your player and test it against Teh Devs test player. And that'll be available all the time, and that's probably sort of your first step on your road to success.

Thank you very much for coming, and if you have any questions you can come to us after.

[APPLAUSE]