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 view additional materials from hundreds of MIT courses, visit MIT OpenCourseWare at ocw.mit.edu.

SARA VERILLI: OK. So I get to do the introduction spiel for today's class. I'm going to start by just saying what we're doing in class today, so you've got sort of a framework of what we're doing, because we're doing a whole lot of things. And we're going to be jumping between speakers, and workshops, and lectures, and in class work. So we actually have a lot of stuff to try and get through today to get you guys started on Project 1.

So we're going to introduce Project 1. And that's actually what my brief lecture's going to be after I get done with the class spiel. We're going to hear lectures on how to do brainstorming. And then we'll ask you to brainstorm. We're going to do lectures on prototyping. And then I'm going to give a more pointed lecture on what we mean by low-fidelity prototype, which is what Project 1 calls for. And we are going to do all those bits of in-class work, hopefully, scattered along the way.

OK. So moving on to introducing Project 1, so that you've got-- and the reason we're introducing Project 1 before we give the supporting lectures is so you've got something to frame. You're going to be hearing lectures about prototyping and brainstorming, which we want you to do in Project 1.

We want you to be thinking about the project as we give you the lecture, so you can try and figure out how you're going to be applying that information to the thing we're going to be asking you to be doing, OK? So the overall goal is-- and I have to actually stand back so I can either see my notes, or I can see the slides. But I can't see both-- to demonstrate a working mechanic via a low-fidelity paper prototype, to service the prototype for a digital game while tracking and understanding how your game design changes over time.

Any words there you didn't understand? Because we've got a lot of jargon, actually, up there, a lot of game design jargon up there.

AUDIENCE: [INAUDIBLE].

SARA VERRILLI: OK.

AUDIENCE: What is actually a mechanic?

SARA VERRILLI: A mechanic? That's actually a really good question. A mechanic is usually a key chunk of a game by which the player takes actions in the game or the game takes actions. So if you're playing like a basic platforming digital game, one of your basic mechanics is your character can jump. And that's what your character often uses to solve a lot of problems.

A very common mechanic in games is shooting. So a mechanic is basically an in-game action. It's a game design term for that. Any other questions? OK then.

So we're hoping that this project will teach you to do prototyping, which is a skill we will hope you will use again in Projects 3 and 4, and we expect you to. And we're also going to introduce our very first project management tool, which is the design change log, which is, hopefully, a reasonably quick and concise way to record the changes your team makes to your design as your design evolves and helps you keep a group memory of what you've done, so you don't end up doing the same things again.

But that's not actually all we're hoping to do. That's just the official goals. We're also using it to give you a small group experience working in teams on a creative project. This is the first and last time you'll be in nice, small, tight teams of three people. Project 2, we boost you all the way up to six people. And by the end, we're going to be in eight-person teams. And you're going to get to see just how more complex it gets working with bigger and bigger groups. But we're going to start you small for your first project, so you get a chance to learn how to communicate.

We're going to teach brainstorming. We're going to expect you to brainstorm again for Projects 3 and 4, although we're not going to formally tell you where to do it or when to do it. We're just going to expect you, when you form your groups, to start going over ideas to use the techniques we've taught you in this project to do that on your own.

We're also going to teach you-- and I think the most important thing here is-- working fast. You've got one week to do this project. The final version and your project reports on this are due next Monday. You're going to get some time in class. But unfortunately, we have a lot to say, to teach you everything you want to do in this class. So you're not going to get all of class. You're going to get a small portion of class Monday and Wednesday.

And we're going to expect you to be able to play something on Wednesday in class. Or hopefully, you'll be running play tests. We expect you to have something working. So you're going to have to learn how to work fast. How do you get those ideas out of your head, onto paper, and usable quickly? And finally, the prototypes you're using here are going to move on to Project 2, probably about half of them. Your team sizes are going to be shrinking, so we'll be doing fewer of them.

We, the instructors, are going to take a look at the prototypes as they come in, both during the presentations and during the playable playtesting sessions. And we're going to decide which of those projects we think really do have the potential to go on to be Project 2. If your prototype isn't chosen to go on, that doesn't mean it's not a good prototype.

We may choose to exclude things because we think the scope is too big for a two-week project, we think it's a little too complicated to try and communicate the ideas. We may also think that there are other projects that sort of nailed the requirements better. So your prototype hasn't failed if it doesn't move on, it's just that there were other ones that we think fit all the constraints better. And all the prototypes are important to do.

All right. So that's that. OK. So the breakdown of the project is, today, we'll be forming teams, doing brainstorming, and getting started working on things. We will do the first formal playtesting session on Wednesday, Monday, everything gets turned in, and including your change log and a presentation.

The vision document, we will be making in class. So don't worry about that, because we'll be explaining what a vision document is and how you make one in class. So don't worry too much about that particular requirement.

The presentation, the one-minute pitch, will be one of the things that helps sell your classmates on joining your team and will be one of the things that we use to decide whether or not we think you guys have a good enough core idea and enough core excitement to move that prototype forward. So it's actually pretty important. And remember, it needs to be one minute. We're going to have like 10 groups up pitching, so it's going to have to be fast.

Finally, design change logs, which some of you may never have heard about. Whoa. Oh, no. My slide, it isn't there.

PHILLIP TAN: Oh. Because that screen's [INAUDIBLE].

SARA VERRILLI: Because this one isn't-- aha. Oh, boy.

PHILLIP TAN: Let me restart this presentation.

SARA VERRILLI: So who here has used a design change log before? Because if you've taken 608, I know you have. All right. So those are your experts. If you have one of those people on your team, you're in good shape. Talk to them.

But essentially, what a design change log is, it's actually really simple. It's a table, three columns. You've got a date. You've got some action you made on your design to change it in some way, and then you've got the reason why you made the change.

And every time you make a significant change to your project, and/or every day you work on your project, you make another entry so that your team and the people working on it and the people looking at your project to see where it is now can see why you made the decisions you made and what decisions you made, what features you dropped, what features you added, what problems you were trying to solve when you made those changes.

Yeah. I was probably-- anyway. So that is, in fact, a design change log. There's an example of one at the bottom of the Problem 1 handout of the-- there's an example of one at the Project 1 handout, so you can see an example there. Are there any questions? Hearing no takers, I hand you over to Phillip for brainstorming.

PHILLIP TAN: OK. All right, let's talk about brainstorming. How many of you have done brainstorming in any class? High school? Before high school?

AUDIENCE: [INAUDIBLE].

PHILLIP TAN: Wow. OK. All right. OK. So it's a term that gets bandied around a lot. Let me see. Awesome. However, there is a problem with brainstorming. And a lot of people have practiced it without actually knowing why certain things are done in a certain way and, as a result of that, may actually be doing it incorrectly.

We have an excerpt of an article-- I'm sorry-- a chapter from a book called, *Applied Imagination,* which is written by Alex Osborn. This is a guy who actually came out with the term brainstorming and the practice of it. He worked in advertising.

And he has lamented since he wrote that book that a lot of people kind of liked the output of brainstorming, but didn't really understand the practice of it. So I'm going to try to start from first principles so that you understand why brainstorming is and how it's supposed to be working. And hopefully, you can sort of get the benefits of it, without maybe, say, overstating its power.

Now, the main reason why we're doing brainstorming in groups-- you will hear people out there in the game industry saying that brainstorming's not necessarily always the best way to be able to come up with an idea for a game. However, in this class in particular and in a lot of video game design outside of this class, game design is coordinated, creative collaboration involving different skilled workers.

And if this is going to be the first time you're going to work with a team, you need to understand who are the people who you might be working with. Brainstorming, if nothing else, can be a really, really good way to just be able to get a sense of who it is that you're going to be working with. And when we start forming up teams, you're not going to necessarily be able to choose who you're brainstorming with. So you want to be able to get a quick read of their ideas and what their interests are and maybe decide, hey, this is a person I'm going to do Project 1 with later on.

However, there are very strong social pressures of being imaginative in a collaborative environment, especially here in MIT. There's a lot of pressure to hold back your ideas or maybe not say what's coming to your mind because you think that's kind of dumb, but other people are going to think that I'm stupid, or anything like that. And brainstorming has certain practices to be able to get you past that.

So there are four principles of brainstorming. The [INAUDIBLE] just do this thing. This was the four principles laid out by Alex Osbourne. And he was Advertising Manager at BBDO, which is a very, very large advertising agency.

And there are certain similarities at marketing and video game design, right? You can produce ads, you can produce marketing, you can produce games, without necessarily saying, yes, we solved the problem. There's a lot of different solutions for exactly the same problem. And advertising is trying to get people to pay attention to the ad, and maybe you'll buy stuff.

In video games, it's trying to create an entertaining experience for somebody else. Millions of ways to be able to accomplish that. So we're not looking for like, here's a engineering solution to a engineering problem where it does, you know, if the thing doesn't explode, you know you've done a good job. That's sort of like a very easy metric to be able to measure success. For games, for advertising, not so easy.

So here's the first principle, no criticism. How many of you have been in a brainstorming session where you said your idea, and then somebody said, oh. Yeah, OK. All right. I see some hands go up. All right. OK. So this is the most important rule of a brainstorming session. If you do not stick to this rule, if everybody around the group doesn't stick to this rule, you can ruin the entire session. The whole session can just become worthless.

You can give positive feedback that says, thanks for the idea. Just simply writing it down is positive feedback of we've received your idea. But you want to hold back on adverse judgment or any kind of negative reinforcement. You can share that after the brainstorm session. You just don't want to do that during a brainstorm session, because it inhibits the process of creating new ideas.

Again, that's sort of like professional shame, especially in a place like MIT, but also if you go out into the corporate world, this idea of you don't want to look bad. And then you can't have that in your head to have a proper brainstorming session. If you're worried about how your ideas are going to be taken, then you're not actually generating ideas at the speed that you need to be. So you need to be able to let your entire group know that we now have total freedom to explore all territories. Whatever idea you generate, we're going to write down. And then we'll figure out which are the good ideas after we've generated all of these ideas.

Secondly, you kind of want to be freewheeling. Typically, the crazier the idea, the better. It's easier to be able to tame down a crazy idea than to take a fairly unambitious idea and scale it up. So typically, you will get your best ideas from some sort of pipe dream or blue moon idea, something that's out of scope, something that's unfeasible. And then you sort of scale it down into the thing that is actually feasible.

So go ahead and shoot for the moon. Shoot for the game that you can't possibly develop in the next week or in the next couple of weeks. And then, after the brainstorming session, figure out how you're going to pare down that idea into something that you can do in a week or in two weeks.

Now you can expand on simple ideas. And I'll talk a little bit about some strategies on doing that. But usually, reworking a wild idea to become feasible, there's usually a nugget there that's a lot easier to identify when you start big and then shrink down.

What are you going for? You're trying to come up with as many ideas as possible. Quantity. The greater number of ideas, the more likelihood that some of those ideas are actually going to be useful. You're not actually trying to solve the problem in brainstorming.

Say, if the problem that we're asking you to solve is create something playable in a week, we're not asking you to solve that problem during the brainstorm session. That comes after the brainstorm session. That comes with things like prototyping, which we'll get to later today.

Now, that's a simple metric that you can use to gauge the success of a brainstorm session, how many ideas that you generate, at what pace are you generating them at. Again, that's why you don't want to be judging ideas while they're being generated. All ideas should be included. Nothing is excluded. And you're going to need to reinforce that within the group of people that you're brainstorming with. There's somebody who's actually got a job to specifically do that, but you've got to help with that.

And you can build on ideas. You can refer back to ideas that have already been mentioned. Take the notes off, and then combine them with your own idea, even if ideas that other people have come up with, to be able to come up with something synthetically more complex. Great ideas are often generated when disparate ideas are combined, LEGO plus Star Wars equals LEGO Star Wars.

[LAUGHTER]

Which is awesome, you know? Marvel plus Capcom, Marvel vs. Capcom. OK, great ideas there. And often, that's actually where a lot of your really good ideas are going to come from, because you're going to come up with a whole bunch of really mundane things that basically refer back to games that you've already played or ideas that a lot of people have encountered, but maybe not encountered together in the same context.

So this is how you're going to do it. You're going to start off with a really, really-- try to create a little casual spirit. This is not like a really straight laced, formal affair. Start with giving your worst idea, the worst idea that you can possibly think of, the most useless, terrible. If you have to do offensive, sure, but usually there's just an idea that you know is dumb. Just tell people that. Everyone put it down. Write it down.

While someone is speaking, don't interrupt them. But do designate a facilitator at the beginning of the session. And it's going to be that person's job to keep things moving. So you don't really need to explain so much, because again, you're not trying to solve the problem here. You're just trying to be able to get enough ideas out so that whoever's performing the role of the secretary can write it down.

And again, before you start the entire session, make sure everybody understands this process. If there's somebody who walked in late in the class or something and is just joining your team, make sure they understand all of these rules, these four rules that I went through. And write down everything.

Now, don't just write down the title of the idea-- turtle tower defense, I don't know-- those are three easy words to write down, but it might be difficult for you to then remember exactly what that idea-- were they like anthropomorphic turtles? Are the turtles the towers? Are the turtles marching into towers? You know, it's like, what was that again?

Let whoever is taking down notes take a little bit more information, maybe like a line of information, just so that, when you come back to that idea, you don't look at a title as, that looks like a-- what was that idea again? Does anybody remember what his idea was? Just take the time to be able to write things down.

Now, this is more general. Today, we're asking you very specifically to brainstorm about the game that you're going to be prototyping for Project 1. But in general, you can apply brainstorming to a bunch of different ideas, even in the middle of a project, Project 1 or later or to other domains. But it's very, very important to be able to know what kinds of problems are good for brainstorming and what kinds of problems are terrible for brainstorming.

You need to be able to find a problem that is very clearly defined. The problem should be simple. If you've got a problem that's really, really complex with lots of very different opposing factors that you have to be taking into account, optimize, something, you can either try to divide those complex problems into subdivisions where each one of those subdivisions could have a separate answer, or you could try to use a different problem solving strategy entirely. You don't necessarily need to apply brainstorming to everything.

You should be able to articulate the problem simply and clearly. But just because I can articulate the problem doesn't necessarily mean that it's appropriate for brainstorming. Osborn, for instance, gives you an example of get married. That's your problem. All right. OK. And then you're trying to solve that.

Brainstorming's a really terrible technique for that. Here are all the ways that you can get married. It's like that doesn't help. Maybe that doesn't identify the problem in the first place. It's is a complex problem that's a little bit too large.

What's your game going to be about? Or what is your player going to do 80% of the time? What's the one action that your player is going to do 80% of the time. Where is our game set? Who is our main character? These are sort of well-articulated problems that brainstorming does work for.

Now, responsibilities. Again, I said designate someone to be a facilitator. And this person's job is to basically keep this whole session moving. We don't like to use the word "leader," because their job is not to be the head of the brainstorming session. Your job is to facilitate it, is to be the clerical equivalent of moving the process along.

So you need to understand the problem and then make sure that everybody around the brainstorm group also understands this problem. And then, when the session actually goes, every time, you sort of sense that there's some criticism coming out, and you say, all right, we'll put that aside and move on to the next idea. Try to defuse any criticism when you see it coming.

And encourage people who aren't speaking up to speak up. Just simply, it's like what's on your mind? Or you'll call out someone's name and just say, any of these things give you any ideas? Try to sort of call out those people who aren't already very much involved in the process to try to get them into the process. Because the more people riff off each other, the more ideas you're going to be able to generate.

Encourage people to join ideas together. Sometimes, you'll just do that yourself, just like look at two random things that you wrote down and just try to slam them together, just to get the process started. And maybe other people will do more logical compositions. Make sure that no voices are lost in this process.

It's not the facilitator's job, by the way, to identify what's the best solution on the board or on the computer when you're taking down these ideas. That may not actually be the job of brainstorming at all. That's actually a problem that you come to later when you see the whole pile of ideas that you generated. And now, you figure out what it is that you're going to choose for, say, Project 1.

Now, there's a second person who's involved. This is the person who either has a giant Post-it pad, or a chalkboard, or a computer usually connected to a projector. And that person is the secretary whose basic job is to record every single idea in a way that everybody around the group can actually see those ideas. You don't want to just work on your own personal laptop, on your own screen where no one else can see it, because then other people can't, say, look at ideas that have been mentioned before and combine them. So make sure that you're writing things up on somewhere where everyone can see.

We've got some giant Post-it pads right here. And you can grab them. And we've got some markers. Stick them up on the wall and write on them.

So you want to try to just report on what people are saying. Don't try to be editorial about it. Don't say, good idea, bad idea. That's your job either.

You do want to keep an eye on the time, especially because we're doing this in class and time will run out. So make sure that you actually have enough time to get down the to those last couple of ideas before we go into our next part of our class, which I think is prototyping.

The secretary can participate in a brainstorming. It's sometimes a little bit tough. It's tough to write down and listen to what other people are saying while trying to generate your own ideas. So what you might want to do is split up your brainstorming session into, basically, two halves. How much time are we giving them for brainstorming?

PROFESSOR: 30.

PHILLIP TAN: 30 minutes? So split it into like two 15-minute chunks. And either the facilitator and the secretary can switch, or just get two other people to be secretary and facilitator. So the people who were facilitator and secretary can sit down and have the space to generate their own ideas while someone else is taking notes and making sure the process is running.

So that's what we're asking you to do. Any questions about brainstorming? Or anything else I've said? Yep?

AUDIENCE: Is there a limit or a minimum number of ideas we have to come up with?

PHILLIP TAN: Is there a minimum of ideas? No. In fact, that's probably counterproductive. Don't say, oh, we have to hit a least 30, no-- it's 30 ideas and we're done-- because that implies that once you're over 30, you can stop. You don't want to stop.

You want to keep generating ideas until you run out of time. And if it takes you 30 minutes to come up with two ideas, that's fine. That's the group. That's what your group was able to come up with. Generally, you want to be generating like a lot, several dozen ideas in a 30-minute session.

SARA VERRILLI: I'm going to ask you to form groups of six people. Grab the people who are nearest you. These groups of six are going to like-- do us a favor and everybody stand up. As you form groups, sit down. You've got three minutes to form groups.

[SIDE CONVERSATIONS]

PHILLIP TAN: So we just finished the brainstorming session, and I'd like to get some people's impression about what worked. Do you need some time to get started? Were people kind of feeling it went a little slowly at the beginning and then picked up or died off halfway? Tell me some results. Yeah?

AUDIENCE: I think it's like the sessions were like, instead of two-minute, three-minute sessions, they were like even shorter. And like broken up into kind of a mini theme [INAUDIBLE].

PHILLIP TAN: OK.

AUDIENCE: And it felt like it ran out of steam.

PHILLIP TAN: OK. All right. So sort of like breaking up even further into smaller chunks and that. OK. Sure. That's a good idea. How about the note-taker as a facilitator [INAUDIBLE]?

AUDIENCE: I think starting with words and themes helped us get just basic concepts out there and then sort of applying them, rather than jumping into game ideas.

PHILLIP TAN: Right. And one of the reasons why we did that will also give you an idea of what's a theme versus whats a mechanic, right? Yeah.

AUDIENCE: It gets hard to avoid the scenario of someone writing an idea, and then everyone's like, oh, that's cool. Let's talk more about that thing, instead of just throwing out completely different ideas.

PHILLIP TAN: Right. Oh, so switching to a different idea, right? Oh, you know, that's not necessarily a failure of the process either. But you're right. You're trying to generate quantity. And instead, people trying to improve the quality of the idea. So yeah, but still that's a happy problem to have, right? It is a good point. Yeah.

AUDIENCE: I think, for our team, jokes helped us a lot.

PHILLIP TAN: Sure.

AUDIENCE: Our first situation was more tense. And then, after one or two specific ideas, everyone became more casual in saying more ideas.

PHILLIP TAN: And that's why the throwing out the bad ideas part helps as well. It helps you get into that mood of we can be silly now. Something that might help with the switching gears problem is the act of writing it down, to begin with, of course, sort of says the idea is on record. And you can always say, let's get back to that later, because you will get back to it later anyway. And there will be time for that discussion. It doesn't always work, but it's worth a shot.

Cool. OK. Let's talk a little bit again about Project 1, just as a reminder. There is a theme for Project 1, and that's planning for randomness. You generated a lot of ideas. We didn't really over-emphasize that constraint right at the beginning of the brainstorm session, because we wanted you to generate tons and tons of ideas. But now, you should be looking at that theme and the ideas that you've generated and figuring out which ones of those kind of match that theme.

Now, that's sort of your overall constraint, over-constraint of what we want the players to be doing is planning for randomness. But is it in space? Is it some sort of like dice rolling mechanic? Is it some sort of race mechanic, are just a branching decision tree or something like that? That comes into themes and mechanics. So you want to be identifying that, as well, as part of Project 1.

So what I'm going to be asking you right now is that, from your brainstorming group, we want you to split into two groups. That's your prototyping group for Project 1. You only have to work with this group for one week, you know? This is not a long-term commitment.

And there were eight brainstorming groups, so basically, we're looking for 16 teams. And so we're encouraging everyone to be in groups of three. Not only does that mean that we will get the right number of groups, it also means that if somebody falls sick in the next week, your team still has at least two people to work on it.

So for each group of three, first of all, you figure out who you're going to work with. And then you choose one item from your list, one of these ideas from your list, keeping in mind those constraints that I just mentioned. And we will be using that idea for the upcoming prototyping workshop, which I will be leading.

So once you know what your idea is, you know who you're working with, I will go into the prototyping workshop to tell you what you do next. And the goal will be to then test that idea through the process of prototyping. And if it doesn't work, you are free to choose a different theme. But you still have to stick with your team. OK? You stick with your team, you can choose a different theme, just to be clear. All right.

AUDIENCE: 10 minutes?

PHILLIP TAN: We have 10 minutes to figure out who you want to work with and what your initial idea that you want to work with, so go for it.

[SIDE CONVERSATION]

And your team was six, right? Cool. You used the computer control, so it was really voice-activated. It just didn't work.

AUDIENCE: [INAUDIBLE].

PHILLIP TAN: How many of you have seen me give this presentation before? OK. How many of you are thinking does Phillip give his presentation every class? Correctarino. I do.

[LAUGHTER]

So I apologize to everyone who's seen this presentation. Maybe there's one or two little points in here that you might not have seen before. But for the most part, it is the same, because it is the fundamental skill that we try to teach in every single game design class, because it is basically the first skill that you're going to need to be able to figure out whether an idea works.

How many of you have made prototypes in any field, say engineering, software, whatever? All right, why do you make a prototype?

AUDIENCE: Because it's cheaper.

PHILLIP TAN: Because it's cheaper than making the real thing. Sure. I thought I saw some hands up there. Yeah?

AUDIENCE: It's faster than making the whole thing.

PHILLIP TAN: It's faster than making the whole thing. Yeah. But that kind of goes hand in hand too, right? Time is money.

AUDIENCE: Making drastic changes is a lot cheaper on paper than it is the built system.

PHILLIP TAN: Ah, OK. All right. You can sort figure out what's broken and then make changes.

AUDIENCE: It shows the proof of concept.

PHILLIP TAN: It shows the proof of concept. You can show it to other people, and not just prove it to yourself. Right.

AUDIENCE: It's a lot easier to experiment and make changes that are necessary when you're not committed to something like [INAUDIBLE].

PHILLIP TAN: All right. You can try a lot of different ideas, right? It's like, I'm not quite sure which way I want to go, so I'm going to try a lot of different alternatives.

AUDIENCE: It helps you realize, maybe, your flaws [INAUDIBLE].

PHILLIP TAN: Catch the big problems that are going to break your design. Yeah.

AUDIENCE: To move from 2D to 3D?

PHILLIP TAN: To move from 2D to 3D? You'll make a 2D prototype, before taking it into a more complex design. All of those are absolutely true.

You know, just for the general sense of why you want to do a prototype at all is to get feedback from other people. And you get it early, you get it cheap. And of course, we talked about making changes early on in the process.

Usually, it costs less money because you haven't committed anything. You haven't invested a whole lot of resources in something just to have to change it again. If you're not quite sure which direction you want your design to go, you want to be able to have lots of different ideas on the table and actually see how all of them work in one way or another.

We talked about how easy it is to be able to change our ideas when you're prototyping. But also, it's easy to discard a prototype. It's easy to just say, well, that was a bad idea, because you haven't really spent all that much time on it. And in many ways, that's something you're going to need to do in game design a lot.

You're going to pursue some of these ideas, some of these ideas that you've already been discussing. And you're going to start prototyping it today. And then you're going to say, well, that was a bad idea and shove the whole thing off the table into a trash can somewhere and start afresh, because you've spent what? An hour designing this thing? It's not going to be a whole lot of time. And then you can learn from that process and build something new.

So that is very good mindset to be able to have in early iterations, that this thing that you're making is meant to be informative. It's meant to teach you something about this project that you're working on, but it's also meant to be disposable. It's something that you do not have to make look good, or you don't have to make it last. You don't have to make it perfect. You are just trying to learn something through the process of prototyping.

Now, when you're making games, this is what you're trying to figure out. You're trying to find the fun. And we've talked about games being a series of interesting choices. And when you're prototyping, what you are trying to do is you are trying to identify what are those few game mechanics that are going to be able to sustain the interest of your players over a significant amount of time.

So the nice thing about prototyping when you're, say, trying to design a computer game using paper materials or something is that you have to define those mechanics down to the very purest form. So for instance, I could say, oh, this is going to be a game about space aliens. And you're going to run around shooting different kinds of weapons and occasionally throwing grenades and taking cover. And you get to fly ships and drive things around.

But what I can do when I'm prototyping is that, all right, I'm going to make a game where you come out and take a few potshots, go under cover, wait for your [? shoots ?] to recharge, throw a grenade, wait for [? shoots ?] to recharge, come out and take a few potshots. That's basically what game?

AUDIENCE: *Halo*.

PHILLIP TAN: That's basically *Halo*. That's the thing that you do every 20 seconds in *Halo* all the time. And that's really the thing that you have to nail and get absolutely right.

So when you are making a prototype, you can identify, oh, this core little loop is the thing that's interesting. And if we can actually get it right and tuned perfectly in [? additional ?] game, then we know players are going to do this over, and over, and over again, and it's gong to feel good.

So you're trying to find that fun. So you're going to focus on the small handful of choices that a player must make in order to play your prototype. And you model the system with a few basic rules that creates interesting choices for the player.

Always ask a question about your game. Make sure it's falsifiable. Anyone heard that term, "falsifiable hypothesis"? No? Yep. What's that mean?

AUDIENCE: Basically, it means some kind of opinion that can be proven wrong.

PHILLIP TAN: Yeah. You could say, well, say the question is that we think the players are going to enjoy making these kinds of decisions. And the answer could be, no, they don't. You want that information.

It could be something like this particular design is going to save time, right? It's going to be less of a cognitive load. Or it's going to be easier for players to understand. And it should be that kind of question that you ask when you're making a prototype. Your prototype should be able to be designed in a way where you can sit someone in front of it, play through it, and then tell you yes or no, was that hypothesis true or not.

Now, physical prototyping, paper prototyping, doesn't give you insight into everything that you can achieve in a digital game. It doesn't say, for instance, tell you terribly much about the sensory experience of playing that game or how feasible a certain feature is going to be. So you don't discover everything through making a prototype, but you do get a lot.

Now, the other thing that a few people alluded to is the ability for your prototype to communicate to other people, especially other people on your team. So imagine getting into a room full of developers. All of you are game developers. And you don't know what's in each other's heads right now. You maybe have come down to a certain theme. And all of you have different versions of that game floating in your head somewhere.

Now, if you want to be able to explain that to somebody else, it can be near to impossible to communicate that. You can try doing it verbally. You can try writing it down. But all of those methods don't really compare very well to something you can actually set somebody in front and play.

So what you are trying to do is you're trying to give you own team, to start with, something solid to work from. It may not necessarily be-- it will almost certainly not be the idea that you're going to end up implementing in the long run. But you're going to start creating some sort of framework for you to then say, well, I didn't like that. I should change that. Or I really like that, and we should try to polish that a little bit.

So it's a communication tool. And any kind of game design definitely benefits tremendously from early stages of physical prototype, because it forces you to think through how your game is going to work. Every little detail about how your rules worked, at least. Maybe not how your game looks, maybe not what your input mechanism is, but certainly what are the decisions you're asking for your player and what are the consequences of those decisions.

You have to define them. You have to write them down in rules that your players, as well as your design team, actually understand. And then they're going to try it out. And you're going to see all kinds of emergent little dynamics pop up when all of these rules are put together that you didn't anticipate, and could be really cool or could completely break your game. And you want to find that out.

I talked about bringing in testers. And as we keep harping on this class, and again, every single class that I teach, testing's incredibly important. But the nice thing about paper prototyping is that you don't really need special skills. You need third grade arts and crafts skills, really. Maybe MIT math. Sure. But actual, we're talking about using markers and scissors and kindergarten counters and dice.

And the nice thing about it is that, as your team gets larger and larger, everyone can still contribute to the paper prototype. It could be an artist. It could be a designer, a producer, a [? quoter, ?] a QA tester, even someone that you just brought in for a day to play your game, can all contribute to the design of the game, because it can give you feedback of, well, what if, instead of rolling this dice, I roll that dice. Or what if the grid was a little bit bigger? What if my pieces were a little bit larger and took up more space, for instance?

A paper prototype itself is not just interactive in its play, but is interactive in the design. You can just open up the hood of your paper prototype and change a rule. Do not do that when you're working with code. In the middle of a playtest, do not change your code, for god sake. But you can do that with the paper prototype.

And many times, you might want to do that in the middle of a test. It's like, you're playing a game. 10 minutes into the game, it's like, oh, my god, this game is hurtling downhill into a pit. And you can see it. And the player sees it. And you know there's a problem. You can just change a rule and just say, what if we did this instead, mid-game even, and then try it out. And says, oh, that seemed to maybe address the problem. Then start the game from the beginning and see if it actually did. But you can do that with a paper prototype.

Paper prototypes, in particular, are very good at revealing usability problems. The simple issue of a player has a decision, knows what he or she wants to do, and has no idea how to actually execute that. It's like, does that mean I roll first and then move, or do I move first and roll, which piece do I move, things like that. Paper prototypes are very, very good in sort of just laying that bare. Your team will see where those problems are, as soon as somebody who's not part of your team sits down and tries to play it.

Let's see. Now, players do tend to more readily give criticism when they see something that's very clearly incomplete, very clearly something that you're still working on. And if you've got a hand-drawn cardboard paper prototype in front of them, they know you're right at the beginning of this project. They know that they can tell you something about, I really don't like how this character looks. And it's like, that's fine. We spent two minutes drawing that character. We can completely redo that. And they'll tell you that, if it feels like you haven't spent a whole lot of time in it.

And that's a useful trick, by the way, if you are doing, say, the digital version of your games later on. It can actually be easier to get feedback if you deliberately insert some things that look incomplete, a giant box that says Place Order right there while you're doing your testing, just to get a little bit more feedback from your testers, so they feel, OK, yes, they're working on this. I can tell you that character looks terrible. And it's like, we spent three days on it, but you don't say that. You don't say that in front of them, because you want that feedback.

But when they give you feedback, you take everything that they're telling you with a grain of salt, because, in the paper prototype, it's very, very easy for a tester that says, I don't like pieces that are this color. I want pieces that are that color. And it's, OK. You could just make the substitution. But what you should do is you should think back a little bit and try to figure out why they're giving you this piece of feedback. They may have a personal preference, and they think this is the way how it should be executed. But probably what they're say is that they have some sort of actual problem with your game, and that's the only solution that they can think of.

So for instance, why does this person want to change the color of the pieces that they're moving around? Is it because they're having trouble telling the pieces apart? That may be the real problem. And the solution may not be change the pieces to a different color, it may be like change the piece into a completely different shape, stick a little sticker on them. On a computer, it's going to be different kinds of solutions.

Vet every little bit of information that comes into you. People can make suggestions on how you're going to change your design, but it's up to the team to actually provide some sort of unified direction on where you want the game to go. So the feedback is valuable. The solutions they're providing may not be the solutions you want to use.

Any questions so far about why we're doing prototyping? No? OK. We'll have more questions later.

So here are the kinds of tools that you're going to be using, very large sheets of paper. This is probably the smallest sheet of paper-- well, we're going to be using index cards as well. A lot of you have seen this before in my other classes as well. This is a square grid on one side, hex squared on another.

For spatial games, you can use this. But sometimes, it's much better to start with a blank sheet of paper. So don't immediately start designing grid games just because we happen to have sheets of gridded paper. But we do want them to be large, so that everyone around the team can actually see what you're working on at the same time, as well as the people playing the game.

If you're going to be designing something like an iPad game or something like that, don't use something that's literally the size of the iPad. Use something a little bit larger, so that everyone around the table can see what the tester is doing while they're testing your prototype

We use index cards for a lot of different things, writing down rules. You can shuffle them and use them as playing cards. You can record data on them. They're very disposable, more accurately, very recyclable. Please, please recycle any paper material.

Dice, of course. They can be tokens. They can be used to help keep track of stats. But something that I've seen a lot, that students like to use a lot, is they use two 10-sided dice to keep track of a number between zero to 99. That's a really bad idea. One brush of the hand, and that variable is gone, right?

You can swap the tens and the ones really easily. It's like, was that 91 or 19? I can't remember. So if you want to record variables, just gram an index card and just write down the variable. And then, whenever the variable changes, cross it out right in your variable. It's much better for you to use a dice either as actual randomizers, or you can use them sometimes as tokens that move around on the board.

Let's see. What else? Game bits, pieces from other games are wonderful, but sometimes a little irritating because, if you took them out of the game box, then you have to put them back into the game box, otherwise, you've just ruined your copy of the game. So I like using disposable kindergarten counters a lot, because I don't really care about returning exactly the same number I took out.

And there's usually less assumption about what a kindergarten counter is supposed to be. Is it a number? Is it a person? Whereas, if you use game bits, that's kind of person-shaped, [? meatball-shaped. ?]

Post-it glue and notepads-- well, I don't think we actually have any Post-it glue right now, but they're basically glue sticks that don't stick permanently. So you put that on an index card or on a piece of paper, and suddenly, it becomes a Post-it. But what we do have are a lot of Post-it note pads, both big ones and small ones. And these are useful for you to simulate displays, for instance, or variables and information that a player is supposed to have. And you don't want that to go flying off if somebody sneezes.

Of course-- oh, right. I have a picture of some Kindergarten counters up there-- pencils, pens, markers, scissors, tape, just to be able to hold things together, write down information. Something else about Kindergarten counters. We have a particular kind of Kindergarten counter, which is an interlocking cube. You take it out?

PROFESSOR: Every year, we've--

PHILLIP TAN: Well, it is possible that we took them out because they're very problematic. Because once I give you interlocking cubes, you make games about interlocking cubes. It's like handing LEGOs. And then, the first thing everyone thinks of is let's make it a game where you put things together and assemble them.

AUDIENCE: Like stacking counters.

PHILLIP TAN: Stacking counters.

AUDIENCE: So you make games about--

PHILLIP TAN: Stacking. Yeah. The pieces that you end up prototyping with sometimes ends up shaping the direction of your game. And then, sometimes, if you are dealing with a designer's block, that can be helpful. That gives you a path to follow.

But in many ways, that's a distraction. You start becoming constrained by your tools, instead of the idea that you're trying to work with. So if we managed to get rid of the interlocking cubes, that's awesome. But if they're still in there, by all means, use them. But don't automatically assume you have to make a game about interlocking cubes.

Now, don't forget to keep a record of everything, including all of the notes that you've had. Use your phone camera doing the session. At the end of your project, make sure that you try to get them scanned, photocopied in some way.

There are a lot of photocopy machines around campus now. There are actually scanners. And you just put the raw materials on top. And instead of generating a copy, you can have them email a color scan to your email account. Those things are really useful for your final assignment.

OK. So those are the tools. Who are the people sitting around the table? When you've actually made a prototype, what happens is-- I guess I got some of my slides mixed up-- this usually happens at the end of having designed a workable prototype is that there is going to be a bunch of people sitting around the table. And each person has a different role. That's, of course, the person who's actually going to be playing the game, preferably somebody from a different team or somebody who's even outside of this class. It could be one of us instructors, for instance. It could be one of the OCW people.

And somebody in your team is going to be a facilitator. This is the person who's going to present the rules to the person who's going to play, you know, tell them that we're looking for your feedback. Make them comfortable. Make sure that they understand what they're supposed to be doing and basically making sure that the session runs according to script.

That person, or somebody else, can also be the computer. This is the person who is actually making the prototype work, updating variables, moving the positions of pieces on the board, revealing new dialogue boxes, or basically making the board respond to what the player decides to do.

And the computer is trying to do all the computation that your actual computer will do, if you're making a digital game. The computer should not do anything that a real computer wouldn't. So for instance, if somebody doesn't understand a rule, the computer person shouldn't be correcting the player.

The facilitator could. But sometimes, what I find is really interesting is that somebody thinks a rule works differently. I let them play through it. And that becomes like a different iteration of the game. And sometimes, it turns out to be better. Then you can run through the iteration with the correct rule. Yeah?

AUDIENCE: So the person who's the computer does have to make a choice at that point of whether they would let them play according to what they think the rule is?

PHILLIP TAN: Yes.

AUDIENCE: Or what the computer-- like if it were a real computer, it would just think [INAUDIBLE] something that doesn't work [INAUDIBLE].

PHILLIP TAN: And it puts it in play. And the computer says, nothing happens, you know? It's like you can do that, right? It's like if the person really doesn't [INAUDIBLE].

But if you notice, say, this person is clearly playing a really sub-optimal strategy, just respond like a computer will, lead them to [? that loose ?] condition or whatever it is that happens. Because in many ways, that's how an actual player is going to learn how to play your game in the long run. They're going to make a lot of bad decisions. The computer is going to respond, and then they're going to learn from the mistakes that they made and then try it differently.

Let's see. Something else that the facilitator can do while the game is going on-- say, the player is actually playing-- is to encourage the player to think aloud. So the person that you brought in to play your game is trying to understand your game and is just like sitting there in silence looking at the board. And you don't know if they're bored, confused, or really into it. So you ask them, what are you thinking now? And if they're saying, what are all these pieces? All right, then. You know that they're confused. The facilitator can help, can say, all right, let's start again, and I'll explain what all these pieces are.

They could be, ah, I've got like five different strategies, and I'm trying to figure out which-- you know, I'm thinking, maybe, if I move this here, [? I move this out, ?] OK, that person's into it, right? That person's really thinking hard about this problem.

Everybody else in your team watching that board. And this is the reason why we have a big board. As an observer, you want to be able to see what's going on in your game. You want to be able to write down the information in a note book or on your laptop and try to record whatever information. Most importantly, look at the face of the person who's playing the game. Again, that helps identify whether they're confused or whether they're really into it and trying to make a tough decision, or they're just bored.

Don't offer help to the user, especially if you're on the development team and you know all about this game. It can be a real temptation to basically just tell the user, this is what you want to do. Don't do that. Sit on your hands. Bite your tongue. It can be hard to take notes when you're sitting on your hands, but, you know, take notes.

And the person who's being the computer and being the facilitator really doesn't have the bandwidth. Those people do not have the bandwidth to take notes. So someone has to be officially designated with the job of actually taking notes and observing.

What we're going to be doing is a Wizard of Oz test for your games. And that is someone's playing the computer. It's very constrained in terms of communication between the computer and the person who's testing your game.

The rules are usually pretty rigid. And the nice thing about it is that you can do a pretty deep analysis of a game. You can make a game that sort of goes through multiple decisions into sort of long-term consequences. That's not necessarily what we're asking you to do for Project 1. I'm just telling you this because you might want to do this for Project 2, 3, or 4. A Wizard of Oz test allows you to really do a deep dive into how a particular feature works, for instance.

And since it's a human simulating the back end, simulating the computing, you can be very high-fidelity with very little cost. That's not, again, we're asking you to do for Project 1. We're asking you to do a low-fidelity prototype. And we'll go into a little bit more detail about what that is.

Very important. Don't let the player know what the computer is thinking. The computer only displays information. You do not reveal how the computer is coming to those conclusions.

There are a couple of other kinds of prototypes that you could design. I'm going to run through them very quickly, because that's not what happens in this class. Player versus player, what you'd expect in a board game or a card game situation. We have a class just for that.

And you can make cooperative games, competitive games. You can make games where everybody has the same role, a symmetric game, or asymmetric game where everyone has different roles. The problem about that is that there's a lot of loose miscommunication happening between the people who are actually playing your game.

And they may decide on a house rule that was totally not what you intended or designed as developers. And that turns out to be the rule that you end up playing with. Again, what I like to do is I like to see them play through that rule. Then, at the end of the game, I correct them, and I have them play through the rule that I originally designed, because sometimes, the rule that they came up with is better. The facilitator, again, helps to explain the rules at the beginning of the game. And then you kind of just watch after that.

It's hard to get consistent results across tests when you're doing a multiplayer game, because it's very hard to duplicate exactly the same situation. Whereas, you can do that on a Wizard of Oz test pretty easily. But if you're making a multiplayer game, you have to do a multiplayer test.

So this might actually happen-- actually, are we saying that you can only do single player games this semester? I'm not quite sure what the constraint is.

You can also do live action, not for today. But I actually enjoy these where you actually have predetermined rules that are explained between the game. And then you just let real human beings walk around and do stuff.

The problem with this is there's a very limited communication between people who are playing a game or even the people who are running the game, because everyone's kind of physically spread out. But it can be really useful for a game, if your game is really about a human being walking through a space, such like a stealth game or something like that. Yes, that's actually a person in that U-Haul box.

All right. So when you're actually making your prototype, here are the things that you have to make sure you do for every single prototype. Keep it hand-drawn, because hand-drawing is fast. You want to keep it really, really sketchy and look really sketchy and rough, again, because that actually gets you feedback from the people who are testing. You make it look too pretty, people are going to hesitate on commenting on how your game looks, for instance. Or they may actually hesitate on commenting on how your game works, even though that may not be the prettiest part of the game, because they can tell you spent a lot of time on this game.

Again, you want it to be big. Everyone needs to be able to see what's going on during testing. And just use one dark color, if you can for your inks, for your markers. It needs to be dark, high-contrast, so that everyone can read it easily.

You can use different color cards, for instance, to be able to tell this card is a different kind of card from that card, but don't use multiple colors on the same card. That's just not worth the effort. And again, it gives people the impression that you spent a lot of time on this prototype. And even if you have, you don't want to give them that impression.

You write your rules on cards. So keep track of your rules. Eventually, you're going to end up typing it down on your laptop, because that'll make it easier for you to turn in your work for Project 1. But while you're working on a prototype, try writing your rules on the cards instead, because that allows you to do things like rearrange the steps in which certain rules operate. It allows you to change out a rule really easily, while keeping the old rule around. And you can just swap things in and you can try things differently.

Every time you change a rule, change the card. If you are changing the number of rules, instead of taking, say, take three steps every turn, you say take two steps every turn, just cross out the three and write two in there.

Now, periodically take photos with your phone cameras. And always try to simplify. Always try to get down to the smallest set of rules that is necessary for you to test your idea. Part of that is just to speed up testing.

You're going to test within your team first, just to be able to see whether this basic mechanism works. And eventually, you're going to end up testing with each other. And the simpler your rules, the less time it is to be able to start a new playtest, because there's less for you to explain to somebody else who's never seen your game before.

So try to get your rules to be as simple as possible. Also, if you have too many rules and something breaks in your game, it's sometimes harder to figure out which rule was the culprit. If you have fewer rules, it's usually more obvious for you to spot.

Now, we talked about iterative design a lot. And this is basically the loop that we're using while we are paper prototyping. You start with a question, a falsifiable question, a yes/no question. And what is this prototype going to achieve? Maybe in this particular case it's going to be, does this theme that we picked actually have randomness in it that you can plan for? Or is it the randomness so random to the point where you can't plan for it? Or is it really just very deterministic, and when you play the game, you can really tell, yeah, here's the optimal solution?

You can look at axiomatic design-- anyone heard of this term? [? Mackee, ?] in particular, uses this a lot-- which is, basically, you establish axioms, which are things that you are taking to be true, a baseline that need to be fulfilled by any workable design. So it could be we assume that this game needs to have some sort of random condition that's generated that changes during the game. And we need the player to identify that this number is changing and anticipate where that number might end up going.

And if your prototype fulfills both of them, then you know that prototype worked. And you can build a lot of different prototypes that fulfill those same two conditions. That's another way to start with a question.

And then you actually start designing, grabbing those pieces of paper, writing out your rules on cards, brainstorming on what your game's all about. But you want to do this as quickly as possible. The less time you can spend talking about this, the more different kinds of ideas you're going to be able to run through. And that's what we're trying to do during a prototyping process.

If you're ever in a situation where you've got two different options, this might be better, or this might be better, or you're not quite sure, it's usually faster in paper prototyping to just start making both prototypes, you know? All right, you do this version, you do this version. Generate the cards, play through both of them. And you'll get actual evidence, actual playtesting evidence on which idea seems to work better.

After you design, you play through it. First within your team, and then you grab somebody from some other team, or one of the instructors. And we'll sit down and we'll play a game and see where the strength in this design, where are the weaknesses. Whatever you design within an hour is going to be clunky and broken and really, really not quite where you want to be. That's find. Do a playtest with that expectation that there are going to be problems. And you'll get a lot more information out of that playtest than you will working on your own.

And then you use that information to revise your design. Make changes, and then you repeat this entire process again. Either you amplify it on the strengths, you figure out what your weaknesses are and you make big changes. So you repeat that.

And here are a couple of tricks that can work, sometimes, if you have trouble figuring out what sort of changes to make in your design. Killing a rule, just like taking a rule out sometimes fixes problems more than trying to fix the rule itself. A certain thing is bogging down the game, a certain thing is too confusing. Just does the game work if you just take that rule out? Sometimes it does.

You can make a resource that's limited, unlimited, or make an unlimited resource limited, limited health, unlimited health, limited ammunition, money. Player interference is more for multiplayer games where, sometimes, if everybody is playing their own little solo game and they're not interacting with each other, you'd come up with ways to mess up each other's plans.

That doesn't always apply with single player games. But if you've ever played a game like a role playing game where you can delay an enemy. Has anyone seen one of those where you can like cast slow or something on an enemy, and then they can't attack you for another turn or something. That's player interference, right? Only the player that you're interfering with happens to be the computer. What can you do to basically get into somebody else's plan and change it? And the computer can have those rules too.

Mess with the play order. Mess with the order of your rules. Just try rearranging the order of rules. And sometimes by grouping things into logical chunks, it can make the game easier to understand. Instead of saying, first, you decide whether you're going to do this. Then you decide whether you're going to do this. Then you're going to decide whether to do this. Maybe you just say, make two of these three decisions, right? And every turn, you get two out of these three. Make them any order you want. That's a way that you mess up play order.

If you're going to change variables, either multiply or divide them by two. There is not enough granularity in the prototypes that you're making today to do like a 10% change in a certain variable and then be able to detect whether that change was the right change or not. That's not going be obvious enough.

Do like big multiples of two, of three, to be able to see whether that's the change that's necessary. You can tune things later when you are preparing the project for the playtest on Wednesday, for instance. But when you're in this class, do things by big multiples.

If your game's really, really, really kind of getting bogged down and taking too long to play, try to get down to the core set of rules, the minimum set of rules that's necessary even just for your game to work. Just for the game to run at all, try to figure out what that core is, the minimum set of rules. And then you add in each rule, one by one. Reconstruct your game. That's one way to identify where the main problems are in your game because, as you add the problematic rule, things start to break.

And finally, be willing to just throw away everything that you generate in the next hour, OK? Anything that you make in the next hour might, in fact, just be terrible. And you're going to learn that. And that's fine. That's why we're making prototypes, so that they can be disposable. That's how I started this talk. So any questions before we actually start making stuff? Coming up with rules?

You're going to grab pieces and create some sort of a play surface or cards that people are going to play with. You don't need to put anything on a board. And before everybody runs down and grabs things, I guess we're going to go back and remind people what a low-fidelity prototype is.

[SIDE CONVERSATIONS]

PROFESSOR: --terrible as long as you're aware of it.

AUDIENCE: I thought I replaced it with a stacking counter, so I guess it didn't work.

[SIDE CONVERSATIONS]

[? ANDREW GRANT: All right. ?] So we just have a couple more minutes left of class. A couple of things to remember before you leave. Take pictures of all of your brainstorming notes. And keep your notes with you, so take them away with you. Make sure everyone on your team can access those notes.

Make sure you know the email addresses of the people you're on your team with, so you can talk them again. And before you leave the classroom today, start up a document and enter your change log. So put in what you designed today and what problems you found by the end of the day.

So I'm going to repeat that one last time. Before you leave class today, start up a document, open it up, and start up your new design change log. Put the date. Put what you did today, so a brief, two or three-sentence description of what you designed today and what are the problems you're facing now.

That'll be important to you both for when you meet as a team again to remember what you did, but also later on to have a full history of everything that you've done. OK? One more thing. Yes?

SARA VERRILLI: One more thing. Tomorrow--

[? ANDREW GRANT: Wednesday. ?]

SARA VERRILLI: Wednesday, rather, we are going to have playtests. We're going to be playtesting all the prototypes. So you need a playable prototype for Wednesday, 3 o'clock.

[? ANDREW GRANT: And ?] then we're starting Wednesday's class off with discussions about the gaming tutorial, so bring your problems, bring your questions about the engines that you were assigned. And we'll talk about them. All right. See you Wednesday.