PROFESSOR: OK. Welcome to class. The theme of today's lecture is all about testing. So I'm going to give the intro to testing lecture where I talk about testing and how, what it is, and why we do it. Give you some basic testing tips, and let you know what we expect you to be doing in terms of testing for your projects.

And I'll spend a little more time on focus testing, because that's the part people are frequently least familiar with. We have Genevieve Conley from Riot, and she's going to give a lecture related to testing. But more about what Riot's doing and stuff like that. So hopefully that will be fun.

We'll take a break. And then since we just told you all about testing and how important it is, we will go ahead and run a focus test in class. We'll give you some time to get things set up, probably about 10 minutes to plan your test. Then we'll set things up and we will run testing.

We're going to ask that everyone on your team test at least two games and that everyone on your team observe at least one person playing a game. How exactly you're going to make all of that happen in flow is your problem. And then we will come down.

We'll give you some time to let your team compare observations and talk about results, and then we'll ask you to come down for a standard in class, two minute no visuals presentation. Monday, project two is due along with your design change log and focus test reports and your individual write-ups, and the project two presentation.

This is a five minute presentation. There are presentation guidelines on Stellar. You can download and look at them. Once again, we're primarily looking for postmortem and reflection. We want to hear about your process.

We want to know what you learned. We want to know what you did right, what you did wrong, and what you're going to do in the future. And we want it in five minutes. Grades for project one are up, along with comments about your individual postmortems and some comments about the overall project stuff. Overall, we were really impressed with the quality of project

one. So keep it up.

But do you stop and read the comments so that you guys can make sure to do good jobs on your individual postmortems going forward. Anything to add? All right.

So people say testing. People say quality assurance. What are they? Quality assurance is actually just a whole process. It involves a whole lot of different types of testing. But in general, it's the systematic process checking to see whether a product of service being developed is meeting its specified requirements.

The most important part of that definition really is systemic, which means you're doing regular, repeatable testing. Looking at making sure that all the portions of your game or project are working. And requirements, making sure you're meeting that it doesn't just "work" as far as you think it does. It works according the requirements of the game, of the project, of whoever you're shipping to and et cetera and so forth.

So you need to know what your standards are to be doing good quality assurance. Without testing, your project will fail, and will take more time, and will take a lot more effort. You can make games without programmers. There's a lot of graphical user interface engines out there. Some of them are getting older and older, but they're out there and you can use them. GameMaker, for example.

You can make games without artists. I bet several of you are figuring out how to do that right now. But you can also make games completely without assets using text engines like Inform 7. You don't really need designated producer or distinct game designer on a small team. One person can really do it alone.

But you're never going to make a good game without testing, and the more systemic and planned your testing is, the more results you'll get from less effort. So who gets to do all this testing? That's really everyone involved in the project.

Do not assume that because you are the key programmer or the key designer or the key whatever that someone else will check your code and make sure it's working after you've integrated it. You really want everyone on the team to consider it to be their responsibility to keep the game working as well as possible all the way through, and also to take the time to go in and check other people's stuff to make sure that it went in.

Remember that this is one of the cornerstones of using Scrum as well, where before a task is actually done, someone else has gone in and verified it. Well, someone else is going to be someone on the team, which means you're all going to have allot some time to doing it, and you're going to have to think about that. Speaking of which, anybody have testing as one of your tasks on your sprint task list?

Anybody? Woo, you win. It's going to take some time. And it's going to take some time that probably otherwise could be used for making code. So go ahead and remember that now and think about that as you're going into your last week of work, that you need to allot some time for testing.

So types of testing. What is the terminology I'm going to use so that we all understand what I'm saying when I say it? I divide testing up into three major areas, and there isn't a lot of formal methodology on this, especially not in the game industry. So I tend to talk about technical testing, play testing, and user testing.

When I say technical testing, I mean getting your game to run. Specifically, I often mean getting your game to run exactly as how you intended it to run. So if the designer claims that the peach mobile should go up to 30 miles per hour and fire eight peaches per second, if you go into the game and you discover that it goes 70 miles per hour and fires two peaches per second, that's not working even if it makes game awfully fun with it like that.

When you start thinking about whether it's fun or not as a developer, then you're moving into the area of play testing which is where you, the team, are playing your game trying to figure out, hey, all those requirements and constraints and plans we made, are they working? And are they working well, and is a player going to enjoy it?

You're play testing your game as a developer. You're checking to make sure it works. In that case, yes. The fact that the peach mobile is currently going 30 miles per hour and it's so slow that no player is ever going to want to use it. That's a bug. That's a really big bug.

But it's a play testing bug if your specs say it should be going 30 miles per hour, not a technical testing bug. And it is good to know the differences so that you can know what you're talking about and why you're talking about it.

Finally, there's user testing, which is getting someone who is not on your team to play your game. You, the developer, while you do play testing, can pretend to be an actual player. But

you are not an actual player. You will never be an actual player for your game. You know it too well. You are too close to it.

You have spent too many hours playing it. So you really need to get someone who has never seen your game before, ideally, to come in and sit down in front of your game and play it. We're going to make pretend when we run our focus testing workshop that your classmates here are actually good testers. Tell you a secret, they're not.

They've seen your game because they've play tested it before. They've heard you talk about it. They heard you pitch it. They might even have worked on it if they switched teams.

So they're not actually your ideal testers, but they are very convenient testers. So we're going to use them today. And for your next projects, we're going to encourage you-- and, in fact, require you-- to go out into the cold, cold world and find actual testers who haven't seen your game.

People in your dorms. People in your other classes. Friends, people at the coffee shop. Whoever you think will do the best job of giving you the information you need. User testing is another one of those terminologies that means a whole lot. So for my convenience, I tend to divide it up into two kinds of things, focus testing and "user testing."

Where "focus testing" is specifically when you're sitting down and you're playing the game, looking at how it plays. Is it fun? Are people enjoying it? Is it challenging enough? Are the mechanics interesting? Is your game a good game, and getting that information back from your players.

When user testing, I tend to use more for looking at the usability of your game. You may have a really, really fun game, but if it takes your players 12 minutes to get off your starting page because they can't figure out what they're doing or how to do it, that's a pretty serious usability issue. And you need to figure that out as well.

So just so you know that those are those two user types. It's all user testing, but having some granularity to find the difference is a little useful. So the basics for technical testing, which hopefully we can go through pretty quickly. I suspect that if you guys have done programming, you've done this before.

If you've had classes that wanted code that worked, hopefully they've helped teach you what the good things to do are. A lot of this sounds an awful lot like good software practices lecture

that Andrew gave, because a lot of good testing is, in fact, a good software practice. So have some technical standards.

Have a bug database. Or, given the size of these projects, at least a bug Excel spreadsheet with a list of everything that you've noticed. Build and test before checking it in. Have daily builds. Have a daily play-through.

Planned feature cuts. Scope. When you realize that you're not getting everything done on Friday night as you're working on your game, start cutting then. Don't wait until Sunday night to decide what you're going to cut.

Cut as soon as you realize that you're blowing your estimates. That will allow you to choose what you cut and to adjust changes in your game to reflect the things you cut. It will also mean that testing your game to make sure-- it will introduce a lot fewer bugs when you do it in a controlled manner than when you do it in desperation, which is why I put it under testing.

Finally, code freezes and asset freezes. These are great, and no one likes to use them because it means you're going to waste an entire day not working on your game. Who wants to do that? On the other hand, has anybody here turned in a project that they've made a last minute change to Sunday night, it was due Monday morning, and then it didn't run at all? I got some honest people out there. Good.

The advantage of taking that whole day to test your game is you turn something in that works on Monday. Often, you turn in something that works a lot better on Monday because you've taken the time to test it. You've found the bugs. You've had a friend help you review your code changes, and you really got it working.

If you want to make a good, polished, solid game, you will think about it and give yourself a cushion for project two. That's probably maybe a day. It's a two week project. If you give yourself much more cushion than that, you're not actually working on it. But actively give yourself that deadline and give yourself some time to play your game.

So that's the basics. Some advanced tips. It's all pretty simple. I think you've all heard this all. How we maintain your feature list. Agree on how features are going to work, so talk with each other. Test those features as soon as they go in.

This may not be worth it on these games, because they're small. When you hit your project four game, you may want to have one of these. Think about having a standardized test plan that runs through all the buttons and the branches and the pages and the options of your game. I realize that when you've got randomness as one of your core design goals, you can't predict how your game is going to play.

But you should be able to take a look at your game and know most of the states that it's going to get in. And you can create a checklist that says, hey, have we exhausted all the possibilities that a player will encounter? Have we tested them all in the last two builds, yes or no?

And the answer should probably there be yes because one of the things that's very common with projects that have a lot of highly integrated systems working together and depend on emergent behavior-- which is most games-- is you can introduce a feature over here that changes the way that a feature that you implemented two weeks ago works. And you'll never know that it did that until you go through and you test and you find that something that was working is now broken.

So it is worth it to take the time to go back and check those features you know nobody touched. Bug reporting. Because so many people are so bad at this, I'm going to take the time with a captured audience to tell you how to do it. Be kind to your fellow engineers. Be kind to yourself, if you end up getting your own bugs.

Write good defect reports and not bad ones. Good bugs include what happened, what should have happened. Because it doesn't do me a lot of good if you tell me that the peach mobile is going 30 miles per hour if I don't know why that's a problem. How do I reproduce it?

If it's obvious to reproduce, you don't need to include it. But if it took 38 steps, include a text file. How common is the bug? How serious is the bug?

Is this something that the team needs to jump on right now, or is this something that Sunday afternoon you're going to look at and say, eh, ship it. Because some bugs you look at and you say, eh, ship it. But you want to know that it's there, and you want to know how it's going to affect your game before you just decide to ship it.

Include some supporting data if it's going to be helpful, and it often is. If you're trying to describe something visually, give them a screenshot. If you can run it in the debugger and catch some debugging code, great. Save the files. Know the version of the game you were

using.

Be ready to help your team member who has to fix your horrible awful crash that happens one out of every four times, and you're not sure how you made it. Be willing to help them recreate it, because it can be really hard to recreate bugs.

Finally, know the difference between a criticism and a critique, and make sure that your reports are critiques of the game and the bugs and not criticisms of your fellow developers and their ideas. And when I say criticism, what I mean here is an unsupported opinion usually delivered in an unprofessional tone. These purple elephants are junk and it's totally stupid that they fly.

A critique is a statement of opinion about the game backed up by examples and hopefully facts. The purple elephants make the game too easy because they can fly. I can skip almost a third of the game play in the bottom area of the screen. There's a problem. There's a reason why it's a problem. And there's a fact that you can talk about, and you can have opinions.

And it doesn't matter that the purple elephants go in the game or not. You can just talk about what they're doing in the game. You don't have to talk about the person who put the purple elephants in the game at all.

So user testing. I think we've actually covered this one to death. But if I did a lecture about testing and didn't stop and talk about why you're doing it, it would be a little odd. Once again, every time we say iterative design, that means you make a design, you test it, you get the data, you move on without actually testing it with users. You don't get real data, et cetera and so forth.

So I think that drum has been well and truly beaten. But here's a warning we haven't really talked about yet, which is test-driven design. Does not mean do what your testers tell you to do. Test-driven design means gather data from your testers, analyze it, and act on it.

90% of all MIT students will tell you that your game is too easy. They're wrong. 90% of all people who like to play first person shooters will tell you that your game needs a gun. If you didn't plan on having a gun in your game, it probably really doesn't need a gun.

Testers are often full of solutions and suggestions to make your game just like the game they wish they were playing. But you're not trying to make that game. You're trying to make the game you're trying to make. So when you gather your data, put it through the filter of what

your vision of the game is and what you intend your game to do, and don't let your testers drive your game. Let your vision and your data drive it.

So how. How do you get ideas from testers? And how do you actually turn them into something useful? So you all remember the experimental method. It's actually really pretty similar.

Going to form a hypothesis, or ask a question. Figure out how to get the data to that question, collect it, and analyze it. So forming a question. What do you need to know about your game that your team can't answer? The more specific you can be, the easier it is.

Sometimes it is hard to get specific because you've got a game and you've got a whole lot of interlinking systems, and it's hard to figure out which one is making things not quite right. But do your best. In there a particular mechanic? Are people having a hard time or an easy time getting started on your game?

How engaged are people in your game play? Are they playing the game and grinning, or are they just sitting there and hitting the keys? How many of your players complete your game in about the time you expected them to? How many of them complete too quickly or take too long?

So when you're running scientific experiments, you try to standardize your conditions as much as possible to get data that can be compared with each other. When you're collecting qualitative data, which is what most focus testing data is, this is actually pretty hard to do but you want to try to do it anyway.

So some of the ways you can do that is by make sure you're giving each player the same set of starting information by having a script to introduce your game, to give them any information about the game they're going to need because your game is in too early to have a UI. Games often don't have UIs that are particularly useful while they want to be tested early.

That's OK, just tell your players what your UI "should" tell them. You can double this up with user testing by giving them pictures of the intended UI, and you can see if they can figure it out from there or not. But that's advanced testing tricks.

Have a dedicated observer. Don't depend on their memory for what they're looking for or what they see. Give them a list of specific things to look for that you think will answer the questions

you need to answer. Give them space to write it down with. And give them the autonomy to stand back and just take notes while someone else offers your player assistance or introduces them to the game.

And finally, you really can't observe everything. Think about what your players might be able to tell you that you can't see. Write down a few questions for them to answer afterwards if you need to. Keep it short. You don't really want to make somebody write an essay while they do your testing.

If you can encourage your players to talk to the game or to talk out loud while they play, that's the greatest thing ever. The best people I've ever heard play games are middle schoolers because they connect their brain straight to their mouth without any filter, and they're just there.

They're telling you what they're doing. They're telling you why they're doing it. They're telling you why this particular aspect of your game is the worst thing ever, or why this particular aspect of your game is the greatest thing ever. And you can just sit there and listen to them and you know why they're doing things.

Adults are a lot quieter. We often don't want to explain why we're doing something. Often it's because we don't know we're supposed to be doing. And we don't want to admit that I don't know what I'm doing. I'm just going to poke stuff until I figure out what's going on.

So that's where you want to watch people. And if you really need to, you can lead and say, why did you do that thing? Or even better, what are you thinking? People will often answer those questions. But you really want to keep those prodding questions neutral.

Don't say, why did you do that crazy thing? Or, what are you looking for back there? But just, what are you thinking? What are you doing? What's your plan? That sort of thing.

Once you've got your data the team can review it together to notice trends, spot common problems, and so on. And that's where the things that four or five or a majority of your testers did or said or ran into, that's what you're going to be pulling out and using to think about how you're going to fix those problems. Or if you noticed a trend that you hadn't noticed before that people were really liking an aspect of your game you haven't thought about, it's something you can work on magnifying so that you can share it with everyone.

So for those of you who fell asleep while I was talking, the too long didn't read version really is

who are you testing with, because your audience will affect your data. What are you testing with? The state of your game will affect your data. That shouldn't stop you from testing with early builds, but it does mean you need to adjust your expectations and make sure your players are ready to play with it.

And finally, why are you testing? Only you can answer that question. We can give you a lot of reasons or suggestions, but if you're running a focus test it really needs to be because you're looking for data that your team intends to use. And you can always use data to support your game.

You really can. Even if you don't realize that you need it, you do. And I keep saying "and gather data." And I talked about observations. I talked about surveys. I talked about interviews, which are our three major tools unless you are going to go to all the work of videotaping and audio recording and getting releases from your testers, which I really don't recommend. That's more work than you should be doing.

But the three major tools really are observing people and taking notes, asking them a standard set of questions afterwards-- either on paper or in person-- and thinking about what kind of data you can gather best with each of those methods. If you have 20 different levels you want someone to rate at a one to five as to whether it's good or not, you should just give them piece of paper and when they finish each level, they can check a number between one and five.

If you want to see how long it takes someone to get started playing the game, you probably just want to watch them with a timer. How long does it take them to get from that first screen to getting into the game? And finally, if you want to get a general impression, sometimes talking to people helps bring out things that you didn't think of.

You can get some insights you wouldn't have thought to ask for by talking with people. But be a little careful talking with people, because people don't want to tell you bad things to your face. They'd rather write the bad things down on a piece of paper where they don't have to admit that they said bad things about this game.

That covers less for middle school students, as I mentioned earlier. But most people don't really want to say mean things to the person who made the game, because we don't want to make people feel bad. So you need to balance that out when you're asking people questions about well, how did you like the game?

The answer is usually, oh, it's great. Yeah. So having run through all of that, let me fight with my computer and bring up a couple of examples. That's not the right form yet.

So this is a script created by students by a team that had a dedicated QA lead person. I do not expect your sheets to look anything like this, but I want you to see an example of what someone did when it was their job and they were taking time to get things ready. So you can see how serious we are.

And so you've got your observer instructions. And what's day's goal? Finding out if the feedback in the game works, along with a couple of other things. And there's a testing script written out pretty clearly along with some basic instructions.

Since these teams had a dedicated focus test lead, they had someone who'd spent a lot of time thinking about testing and a lot of people who'd spend a lot of time working on the game. So the lead tester was in the position of trying to make sure that the team understood how to run tests, because they weren't experts at it either. So I just embedded all of the good advice right into the script.

And there you have it. And then there's a sheet to write notes down. They've got the big questions that they're worried about, so they can write down. They've got space for other notes. And it means that whatever they've got, they've got.

And then finally, it follows up with a fairly short questionnaire trying to get some answers to questions they're wondering about. The specifics are always going to vary from game to game. Looking at the focus test sheet from one game isn't going to help you wildly with yours. But you can think about the way the questions are worded and the number of questions so that you're not overloading people.

I believe these are up on Stellar as a sample if you want to download it and look at it as an example. Finally, the paper we expect you to do. So this is the focus test report which is also download-able on Stellar, I think in either Word or Rich Text Format so you can use it. When we say we expect you to turn in a focus test report, this is what we're expecting you to turn in.

If you take a look at it, it walks you through the whole process and there's a space for most of this stuff. So what's the goal of the focus test? Record what happened at the focus test and what are you going to do afterwards.

When you are planning your focus test for this afternoon later in class, I recommend you download this and use it as your sheet to record notes on, because it'll help make sure that you get the data that we are asking you to get. And I think that's actually most of what I have to say about testing. Are there any questions?

PHILLIP TAN: I have a question. What's currently using some bug database, and if so, what are folks currently using? Or let me rephrase that. Who has used a bug database before? What are some of the ones that you're familiar with?

AUDIENCE: Chehra.

PHILLIP TAN: Chehra?

AUDIENCE: Bugzilla.

PHILLIP TAN: Which one?

AUDIENCE: Bugzilla.

PHILLIP TAN: Oh, Bugzilla. Yeah.

AUDIENCE: Buganizer.

PHILLIP TAN: Which one is it?

AUDIENCE: Buganizer.

PHILLIP TAN: Buganizer?

AUDIENCE: Yeah.

PHILLIP TAN: Oh.

AUDIENCE: I use [INAUDIBLE].

PHILLIP TAN: Yeah.

AUDIENCE: [INAUDIBLE] That's the [INAUDIBLE] database, yeah.

PHILLIP TAN: And as Sara mentioned, you can also just use a Google spreadsheet or something.

AUDIENCE: [INAUDIBLE] is a great bug database too.

PHILLIP TAN: There is always this intermediate step between someone entering a bug into a bug database and then deciding that they're going to fix it, right? So priority, somebody actually says I think the bug that the characters are too short or something, and then the art director says no, not a bug. Will not fix that.

So just because you're entering something in a bug database doesn't necessarily mean that your team's committed to actually spending time on that. But when a team does decide that, OK, this is a bug. Maybe because it crashes the game.

Maybe because it's actually not how we want the game to perform. It needs to be entered into your task list. So that means your task list is changing and you need to be able to put that in a priority.

Possibly, this bug is more serious than some of the planned tasks that you already have on there. So you need to rearrange that so that [INAUDIBLE]. Something else that Sara mentioned very briefly, but I want to stress, MIT students are actually horrible testers. I've heard this many times from many different companies here and [INAUDIBLE].

MIT students are not a representative population of the general public. You look at systems differently. You look at screens differently. So even though most of your testing that we're going to be doing this semester, I imagine that a lot of testing you're going to do on your own is going to be with other people in your dorms, for instance, and that's fine for this class.

But don't think that it's normal to use even university students as your only test audience. That's really not representative of the general public. MIT students in particular.

PROFESSOR: MIT students do [INAUDIBLE].

PHILLIP TAN: The final thing, as Sara mentioned, user testing in two contexts. One for overall category of you are testing with users and then it's broken down to more specifically, focus testing and user testing. And you may be wondering why user testing is used and that [INAUDIBLE] two different things, as Sara said, it's good to be more granular.

What we normally refer to as usability testing, as Sara mentioned, has only fairly recently become the standard name for it.

PROFESSOR: Yes. Yeah.

PHILLIP TAN: Back in the '90s and even the early 2000s, user testing and usability testing were basically the same thing, because we were talking about websites and enterprise software. We were not talking about games.

So the only kind of user testing that you'd be doing, putting a piece of software in front of someone, getting feedback, was usability testing. That's why they had the same name for so long. But now there's focus testing, which is very, very applicable not just for games but also for things like user experience. I had not heard that term user experience [INAUDIBLE].

So where it's not just can the player figure out what to do, but the person actually having the desired experience for playing the game. That's why now there's multiple names for that.

PROFESSOR: Thank you. Then I think I'm out of here. I'm going to sneak off with my computer.

GENEVIEVE CONLEY: Hi, guys. My name is Genevieve Conley. I'm a researcher at Riot. And some other terms that you might have heard for that, user experience researcher, user researcher, or the way we were just talking about it, like user tester basically.

And a little bit about me. I graduated from MIT in 2010. I got my degree in Course 9. And I actually worked at the MIT Gambit game lab on a couple games there. So that's actually where I got my start doing user testing, was in the summer projects we would run play tests with local community members, have them come in, and that's where I actually did my first user tests.

So I work at Riot Games. How many of you guys have heard of *League of Legends* before? OK. How many of you actively play *League of Legends*? OK, good. So I'll give a little bit of context on the game just so I'm not losing anybody. And I'll try to make sure I explain all of our jargon, because sometimes when you're in a game company you just start using the language like everybody speaks that language.

But a little bit about Riot is we make *League of Legends.* And our core mission is that we aspire to be the most player-focused game company in the world. And this is going to be relevant to what I'm talking about today, because you can't really be player-focused if you don't talk to your players.

So what this means at Riot is we think about all of our decisions in terms of how much player value it adds, or how much player pain something negative causes. So this is like-- I don't

know if any of you familiar-- we have an eSports scene, and right now we're in the middle of our world championships. And we created a video and a song that would basically, we hoped to be the anthem of the world championship.

And this is something that a couple people at the studio thought was really important, so they just went out and made it. And the point was just to make the players pumped up about worlds and to give them a song that represented how they feel. And this is just a small example of something that happens at Riot where we really just want to do something that we think players will think is cool.

And one of the ways if we do that is through play testing and user testing. And I'm going to use a little bit different terminology, so I'll make sure that I tie it back into the way that Sara's been talking about it as we go. So a little about *League of Legends.* It's generally a 5v5 game. It's a team oriented game.

So you play as a member of a team. And you usually play a specific role or position on that team, just like basketball. And you play as one of many champions. And it's online and it's competitive. So like I said, we have an eSports scene where you play against another team.

And you go basically across this map, try to take objectives, and ultimately blow up this thing called the Nexus which is a giant crystal on the other side of the map. And you play as a champion. And these are player characters that each have an individual set of skills, different play styles, and there's over 120 of them in our game.

So even though each game is session based-- you don't level up your character necessarily over the lifetime of playing the game, each session is a new session-- there's lots of different ways to play because we have lots of different characters, which I'll talk to in a little bit. So one of things that I talked about is it's really hard to be player-focused if you don't talk to your players.

And one of things at Riot is that we all play *League of Legends.* In fact, some of us play it a lot. Some of us play a lot at work. But just because you play your own game doesn't mean that you actually know what it's like for all of your players. And using MIT students is a great example. If you guys just play with MIT students, you might not know what it's like for people in other countries, for example.

So one of things we try to do is make sure that we understand our players broadly and know

that we are our players, but we are a small sub-segment of our players. The way we use research is it equips us to understand player beliefs, their needs, their desires, and their relationship with the game and with other people in the game. And what this allows us to do is design experiences.

So there's specific experiences, interactions that we're hoping to design. So in general when we're creating a feature, we have a set of goals that we're trying to achieve. And what we want to do is we want to help designers. developers, QA, artists, we want to help them achieve these goals. So we want to know exactly what they're trying to do and then get them feedback on how well those goals are being communicated to players, how well they're actually able to interact with those, and then what their reception is as a result.

So the way I'm going to break this down is play tests and user labs, which is essentially the way we were talking about it before, is play tests and user tests. So play test is the backbone of creating your game. It's really, really difficult to create a game without play testing it. So I'm glad that's really part of how you guys are designing your games.

And play tests basically allow us to quickly gather information about a game or a feature to see how it's working. And typically, this is done with resources that are readily available. So people in your dorms, people in your class, or at Riot, we often do it with other Rioters, and even on the team.

And some of the benefits we can get out of this is we can identify technical issues. Now, Sara talked about a specific type of this, which is quality assurance testing, which is less what I'm talking about. I'm more talking about testing with your team to just quickly identify technical issues that might prevent you from playing that game.

You can also look at things like balance. So in a competitive game like *League of Legends*, you play against other players. And if one type of character is really overpowered or has a significant advantage over another type of character, we can quickly identify that through play testing.

So for instance, our designers have two play tests a day where they test their in-progress champions to see if there's anything basically game breaking like that that they can quickly identify and iterate on. We can also look at things for fun. I mean, this is the most obvious one. But it's really important when you're making a game, if the game is supposed to be fun, that it is actually fun.

And in our case, we do want to have a fun game. So we have a whole group of designers and developers that have a team which is known as the play team and they make temporary game modes. So I don't know if anybody played recently, but we had a big in-game event called Shurima, and we had this game mode called Ascension.

And when they were developing that, they were often several times a day-- many, many times a day, in some cases-- testing every iteration that they made to that game mode to make sure that it was actually fun. And a lot of times what you'll find is that your game isn't necessarily fun sometimes. And that's OK, because each time you make a change, you're getting closer to that fun experience.

One day, you'll be playing your game and like, oh my gosh, it's actually fun now. This is great. Let's do this. So it's a really good way to get that feedback.

And another specific area that we're looking for a lot when we play test is clarity. And clarity is a word that we use to mean basically how clear is the information-- either the visual information or the mechanical information-- how well is that being communicated to players? So this could be something like with UI, if you just have a picture and you show it to players or to your dorm-mates, your classmates, how easily are they able to identify what a button means?

Or which things are clickable and not clickable. Those things are all really, really important in a game like ours where we have five people on a team, 10 people on the map, each with their own set of abilities and visual effects. And there could be a lot of visual clutter if we didn't think about this type of thing.

And then the way we typically get these, at least at Riot, is like I said, team tests. That's the example I used with the designers and the play team. They're just playing within their team frequently to make sure that they can quickly and easily give each other feedback.

Quality assurance test is another type that's a very specific implementation of this, which Sara spoke to. And in house "guerrilla tests"-- and I always have to be careful how I say "guerrilla," not "gorilla." "Guerrilla."

This is when we bring other Rioters who aren't part of a team who maybe haven't had exposure to that particular feature. This is particularly useful for things like clarity or usability

tests, because we don't necessarily need to have that perception or that sentiment that is really unique to players. What we need here instead is just see, hey, how easily is this usable for a reasonably tech savvy person?

But even then, of course, there are some cultural differences. There's some hardware differences. So it's a first phase when you do them as guerrilla testing. And you always want to make sure that you follow up with real players after the fact.

So user labs, this is what I really focus in. So I use a lot of different methods at Riot, but one of the key ways that we're able to get player feedback is by bringing players to Riot to have them play the game. So this is a way to help us understand how real players will actually approach a feature. What we get out of this is really, really important.

We get reception and interest. Like, how stoked are players going to be about something, or what we call the table flip reaction. Which is like, we're working on an update to *Summoner's Rift* right now. And it was early on. It was really important for us to know, were players going to be really upset that we were changing something that's so key to the game.

So we made sure to loop them on very early in on the process to see how they were responding to changes we were going to make. We also look for things like usage patterns and errors. And this is important that we get a broad swath of different types of players to look at this type of thing, especially with things like we just updated the client. Making sure that the changes we made to the interface weren't going to confuse players-- experienced players who were used to our old version and new players who would be coming into our game for the first time.

We also look at needs and desires. And this is not like Maslov's-- Maslow's hierarchy of needs. It's more like, what are things that players really want from our game? And understanding that can be difficult sometimes because, like I said, we all play our game.

We're very entrenched in this experience. We have our own experience in the game that we feel very strongly about. But understanding that other players might have different needs and desires, and understanding what proportion that represents within our broader player base, these are really important things to understand.

How we get this is a lot of different ways that I'm actually going to go into in a little bit more detail. And each one has a specific application, but there are a few that we use pretty much in

every user lab, which I'm going to go through right now. So the first is something we just talked about, which is just observation.

And I bring this up because it's something that anybody can do. You don't need a dedicated researcher like myself. You guys should and will be doing this. And so in this case, I have Chris, who is a developer at Riot, sitting in for a player. And I have [? Jisan, ?] who is another researcher at Riot, sitting over his shoulder watching him play.

In our case, we typically just have one researcher in the room both taking the notes and doing the observation, just because of resource limitations. And because we've done this a lot and it's our full time job. So in this case, [? Jisan ?] is watching Chris interact with the client. And you can see in his hand, he's actually got the script-- or what we call protocol-- in his hands so he can remember what things he should be looking for, what things he wants to say.

In this case, we don't generally try to use it as a script. We try to use it as a memory refresher. That way, players can feel really comfortable with us and feel like they have a relationship and can have a comfortable conversation. One of the ways we achieve this is through what we call the Think Aloud protocol, which is what Sara was referencing, which is basically we try to make sure that they have this direct link from their mind to their mouth.

And there's a couple ways to achieve this that I'll speak to, but the goal of this is we want to know exactly what the game is communicating to players, and then how players are processing that and creating an internal model. And the best way to do that is just to step back and let them tell you exactly what they're seeing. And if you aren't quite sure what they're seeing, then you can follow up, which I'll get to in a second.

So here's an example of a player who's seeing our character Lucian, who is a champion who was released fairly recently in the game. She's seeing him before he had been released in a work in progress state and giving us feedback on what she sees.

[VIDEO PLAYBACK]

-I can see that he walks faster than the normal--

[END PLAYBACK]

GENEVIEVE CONLEY: Sorry, we forgot to plug in the audio. I think you guys could hear it all right. All right.

[VIDEO PLAYBACK]

-[INAUDIBLE].

GENEVIEVE CONLEY: Oops.

-I can see that he walks faster than the normal character would, but his animation doesn't make it seem so. It seems like he takes a really big stride, but I was imagining him to walk really fast. But it seems like it looks like that way.

[END PLAYBACK]

GENEVIEVE CONLEY: So this is an example where the player has just brought this up to us that she expected him to move one way, and he's actually operating in a different way. You can think of this is as like, basically a violation of expectations, which is often what we're looking for especially with our characters. Does what you see and what you hear about a champion match with how they actually play?

So one way you can do this is to just let them talk. And it is really important not to be giving feedback on whether or not you think they're correct, especially if they do have a question. It's also really important not to be judgmental, right?

So the way you phrase your questions, even if you aren't actually being judgmental, sometimes players might think that you're judging them. So one of the contexts we give players upfront is always for us, since we actually didn't work on the game or the feature that they're seeing, we can say, we didn't work on this. It's really important that you give us honest feedback. I'm not going to go home and cry at night, so I want you to give me totally honest feedback.

And we also let them know that sometimes we're going to ask them questions, and we're asking it so that we can understand what they're thinking because that's what we're interested in understanding. We're not asking it because there is a right or wrong answer. So one of the ways that makes it easier to make sure that we understand what their mental model of the game is is a follow-up with some probing questions.

So the player in the previous example was pretty clear about what she was seeing and why it violated her expectations, but sometimes players don't necessarily know why it doesn't actually

fit what they expected. Or they might know that something's frustrating, but they're not necessarily giving you the answer why.

So one of the ways you can follow it up is either with probing questions, which you ask in the moment, or you can follow up with interviews after the fact. So here's an example of [? Jisan ?] following up with a player who just saw Lucian for the first time.

[VIDEO PLAYBACK]

-How was that play [INAUDIBLE]?

-I thought it was fun. I thought it was [? different. ?]

-What makes it fun and something different about it?

-So he, I guess, normal [INAUDIBLE] skills, just increased the intensity. A lot of his skills based off of move speed and [INAUDIBLE] speed, and I guess he feels [? confident. ?]

[END PLAYBACK]

GENEVIEVE CONLEY: So this is an example. A lot of times you'll find this, is that when you ask that initial question, you get a short response. And what we're always trying to do as researchers is invite a conversation where they're just really comfortable telling us exactly what they feel and giving us a lot of detail.

So in the first question, he asked a really broad opening question, which was "how was that?" He didn't even ask, "how did that feel," which might lead in a certain direction-- although that would be an appropriate question as well. He left it wide open for the player to take us where they wanted to talk about it.

So the player mentioned that it was fun and something different. So [? Jisan ?] wanted to get more depth into what does that actually mean, because that alone doesn't give us enough direction to be able to provide to our designers to say what was good, what was working. So he mirrored exactly what the player said back at him.

He said, oh, what was fun and what was the thing that made it feel different? And then the player, that way we're not actually putting our own spin on interpreting what we think they're saying is fun or maybe what's different. We're just mirroring exactly back what they said so

that they can give us more detail.

You have to be careful with that, because you also don't want to sound cold and clinical like you're sitting in an office somewhere writing down everything they say, which you will be. You want to make it feel like it's a natural conversation.

So we're really lucky because we get to work with *League of Legends* players so we already have a connection built up with them where we can talk about it, but it's really important that you make that as hey, I want to know what you think. You're really valuable. Let's just have a conversation about it. And it can be really challenging sometimes.

So this is one that you guys are already pretty familiar with, too, is part of interviews that I've clumped in here is basically the questionnaire. Sara gave you guys some great context about when this might be useful. Here's an example of when we use it in *League of Legends* research.

So this is an example with Azir, who is a character who just came out in *League of Legends*. This is actually from quite a long time ago when he was in the concept phase. And this is something we do actually with most of our concepts, is we put them in front of players before they go into development to see, hey, is this something that feels like it fits in the world of *League of Legends*?

Is it clear where maybe the source of power is? And does this look like a champion you would actually want to play? Now, we don't use data to drive decisions at Riot. I want to make that really clear. We use it to inform.

So just because something wasn't resonating with players in our labs wouldn't mean that we would cancel a character. It's just a way for us to get feedback for our designers who already have their experience and the experience testing with other Rioters to help them make informed decisions. So what's useful about a questionnaire in this case is that we can compare across champions over time and across different roles or positions and see what is resonating and what does good look like.

We also use this as a jumping off point. So we start with a questionnaire, and then we go into a more detailed interview about it to make sure that we're getting enough detail to be able to provide useful information to the designers. So you guys are also probably pretty familiar with the idea of experiments in general. But part of research testing can be devising experiments.

So we often hire people from psychology backgrounds or research backgrounds in academia because they have experience setting up experiments. So this is like the classic one. I don't know if you guys have already covered this, the Engelbart experiment. OK.

So this is foundational human computer interaction. It's one of the most cited studies. And it's English, Engerlbart, and Berman basically were looking for a way to improve a way to interact with a screen or text manipulation. And what they proposed was something that was radical and was called the "mouse." Maybe you've heard of it.

So that thing in the right, I know it's not a really clear photo, but that is one of the first early prototypes of a mouse. And what they wanted to do is they want to see, is this better than some of these other interactions out there. Qualitative experience alone might not be a very strong way to prove it in this case, especially when you can collect quantitative data to back it up.

So what they did is they used the time that it took to complete a task, a very specific task that was given across all these different ways of manipulating text. And they saw how many errors the user made in this case. So you can see that the mouse did pretty well in terms of time overall. It didn't take the participants very long to be able to complete the task.

But it had very few errors, which made it overall the best out of these different ways to select text. And you can see some of the other ones on here were things like knee control, joystick, a graphicon, which is like an arc drawer, a light pen, and a joystick. And all of these were ways that we could have gone instead of the mouse.

But based on some of this research, they were able to improve their design and now you can see I'm using one right now. And as part of the experimental approach, I broke it out, but there's a specific way you can look at it, which is using physiological techniques. And there are many, and I'm just going to cover two.

And typically, these are used for very specific types of research. They're very exciting. So oftentimes, you'll see people want to use them on everything. But they're very particularly useful for certain cases.

So this is an example of eye tracking, which is basically fundamentally, we put a camera up that's able to specially track where eye movements are. And there are more sophisticated ways you can look at this-- and I'm sure a bunch of researchers at MIT could tell you a lot

more about it. But in this case, it was pretty simple.

We were just looking at where people were looking on the screen and how often and how long they were looking there. So you can see the red parts of the map here are the parts where there are more intense gazes or more gazes overall. So unsurprisingly to probably most of the *League of Legends* players in here, or you guys as gamers in general, the mini map has a lot of gaze.

The bottom part, which is where the abilities are displayed, also has a lot of gaze. That has a lot of information that's really relevant to players like the cool down timer, basically when the ability is going to be ready again. And the center of the screen is where most of the action happens.

So this type of information is helpful for us. If we want to, for instance, introduce new information on the screen, we'd want to know, hey, are we going to introduce this in a place that's going to get a lot of traffic naturally. But also is it going to interrupt things that that gaze is there for a very important reason, and maybe we shouldn't interrupt that. So we can help us to inform those types of decisions.

There's also many other types of physiological, and I'll just hint at one right now which is things like galvanic skin response, anything that has to do with physiological arousal, either for stress or excitement. These can help with situations where maybe players wouldn't be able to vocalize why something was frustrating or exciting.

Or it can also help combat some biases like recency effect, where if you ask a player at the end of the lab or at the end of a play test what was interesting or what was frustrating, they might preference things that were at the end. Or they might preference things that were really emotionally significant for them, but might forget smaller things that are still maybe small frustrating things that build up over time.

So none of this is really that interesting if I don't give you an example of how we actually do at Riot. So here's a case study. And I actually presented this last year as part of a talk that we did at MIT. So I apologize if a little bit is re-run for you guys. I'll try to spice it up a little bit as I go.

So I'm going to be talking about Lucian, who I introduced to you guys earlier. He is one of the player characters or champions in *League of Legends*. So where we always start as researchers, and part of the job that I haven't really alluded to yet is we spend a lot of time

talking to our designers, to our developers, to our artists, to our producers, to make sure we understand the goals they're trying to achieve.

And I don't know how many of you guys got to catch August Browning's talk last night, but for the champion designers, they start with a very specific idea of what goals they're trying to achieve. What is this player character going to feel like? What are you going to feel like when you play it?

What is the mechanics supposed to feel like? What is the theme supposed to feel like? How does this resonate with players? How does it fit in *League of Legends*? These are all really important questions for them that they're constantly asking themselves.

And in the case of Lucian, he was supposed to feel fast, sleek, speedy. He was like a gun ninja. Like pew, pew. No? All right. Trying to see if you guys were awake there.

And he's a range damage character with finesse game play. And he basically has the high skill ceiling. And what that means is basically he's harder to pick up, but when you master him it's a really rewarding feeling. So that's a trade off between accessibility, maybe how quickly a player can pick him up, or what level a player can pick him up, for a reward of how rewarding that feels when you do master that champion.

And some of the challenges in this were, this guy, his theme is pretty different. His equilibrium style, if you've ever seen that movie. And that's not something we had necessarily done that much in *League of Legends* before, so we wanted to make sure, does he even fit. And we also wanted to know how do you let a player feel super fast and speedy in what is essentially an RTS style game. It's a challenge.

So this is what the test plan eventually looked like. It's not necessarily where we started from. But we started with concept testing, and then we went through actually three lab studies with iteration in between based on the feedback we got from players. And after he launched, we did some post-launch evaluation.

So step one, just like I showed you earlier, was to do this concept testing. And we got some pretty good feedback based on that. His fantasy was good with players. They liked the mood.

[THUD]

They felt like he fit. Are you all right?

AUDIENCE: Yeah.

GENEVIEVE CONLEY: And he made sense for the role of an AD carry, which again is like a range damage dealer in this case. So he had high appeal scores and the theme felt really strong, but one thing we very quickly identified-- it's a little hard to see on this one-- but his guns were different colors.

So players expected the guns to do different things based on their colors, which was not what we were planning. So this was very good early feedback before we got very far into development where we could pivot and make sure that the idea of the two guns doing the same thing was conveyed. So based on the feedback we got, we were able to give that to the designers and they continued their development.

And then after a little bit of time, we were able to take it into the lab. So we started with a Lucian that was designed to be a speedy range assassin. He had a double shot and a utility kit. And when I refer to kit, just to be clear, that means his set of abilities. And characters in *League of Legends* generally have three abilities and then an ultimate ability, where the ultimate is like the super big bad power that you get after some leveling of the character.

So I blew the reveal here, but in the first lab we found out that he didn't really feel that sleek and fast. He felt stiff. He planted and then fired his double shot, which made him feel like a turret. And people did like his visual theme. They thought it was compelling, but it just didn't match his play style.

So we took this back to the designers we said, hey guys. Here's the videos. Here's the notes. We usually make a highlight reel based on what we see to support our evidence so that the designers can get enough feedback from that, as they are our subject matter experts and really know how to interpret what they see. And we said, OK. Let's go back.

You guys take some time, do some changes. We'll take it in the lab again. So we did lab study two. And this time, we had a different Lucian to bring in. This time, he was more mobile. He was basically a range [INAUDIBLE], running gun, double shot guy.

Well, we had a little bit of a mixed bag of results here. He did feel a little bit faster. But some poses and animations felt awkward. And he was high speed in the numbers-- which I'll talk about in a second-- but he still felt slow for some reason. But worst of all, we got quotes like this one about the ultimate, which was "It worked in the way I did not want it to work," which is a pretty good sign that something's wrong.

So we took all this back to designers again. We said, hey guys-- and the animators and the artists in this case, of course-- and we said, he feels slow. His ultimate doesn't really have the usability that people expect, but the double shot theme is working, so we're definitely on the right track and players are still very excited about this champion.

So now we're in a bind, and this is often the case when you're working with research from players because they're not always able to articulate exactly what's wrong because they don't have the context that you do. So this is why it's so important for us as researchers to pair with our designers, our artists, our producers, and our players to be able to get the right kind of data to make informed decisions.

So here was our problem. The frame delay between his basic attacks-- which is actually very, very low in this last lab. Usually that means things feel fast. It's a pretty standard way when we're developing characters to be like, OK, this will feel faster. He also had a very high base movement speed, which means that the time it took him to run from point A to point B was pretty high relative to the rest of our champions.

And if that weren't enough, he actually had a dash ability which means that he could just jump basically from one point to another. All of these things should have made him feel fast. But players were saying he didn't feel fast, so there was something wrong.

So our designers and our artists and our producers took all of this information back, looked at the videos, play tested the heck out of it, and came to this conclusion. They increased the ability responsiveness and they changed how animation flowed from one state to another. So I'm not an artist, but what this means to me is that they are basically able to make the transitions from animation a, animation b, or ability a to ability b, and make those feel more natural. Not like me.

And they also gave more feedback to the double shot. And again, this is where subject matter expertise is so important because our animators, our visual artists, our designers knew this bag of tricks to be able to apply to it. And finally, they were able to make the ultimate ability clearer.

So they were able to the telegraph better what it was actually supposed to do so that players would have more accurate expectations. So we took it to lab study number three. We put it in and finally, the game play anesthetics fit the concept, and players were quite happy. But not

everything is always happy ending, everything is perfect.

There was still some confusion about how the ultimate was supposed to work. We found that players wanted more control of it. Basically how the ultimate currently works is he locks his arms forward and shoots in that direction. So if you stray left to right, he continues shooting in that direction which, for some players, was pretty confusing. Now what they expected based on some of our other champions, and it wasn't able for them to learn it in one session.

So one of the things that we have to take away from that is labs have some external validity problems, which basically means that we're sitting in a lab. There's not other players around. They're often not playing on live. There's a lot of things that are different in our test situations than real life.

Usually, you wouldn't try to master a champion in one session. So this is one of those cases where we put it out to our designers and they decided, and we supported, that he was going to be a high skill ceiling champion, which meant that players were going to need some time to learn how to play him. And that just because they didn't learn him in one session didn't mean that they wouldn't learn him over time.

So this is what Lucian looked like when he did go live. And it helps if I play the video.

[VIDEO PLAYBACK]

[WARFARE SOUNDS]

GENEVIEVE CONLEY: So he's in the top there. Pew, pew.

[WARFARE SOUNDS]

GENEVIEVE CONLEY: So you can see his dash there.

[END PLAYBACK]

GENEVIEVE CONLEY: So yeah, sometimes a little confusing if you don't play the game to have to watch 10 champions all in one little team fight like that. But ultimately, the important part's there. He's using his ultimate and he dashed at the end.

Ultimately, ended up actually delivering on player expectations for this champion.

[VIDEO PLAYBACK]

[END PLAYBACK]

We don't need to watch that again. So what did we learn? Well, we knew going into it that Lucian was going to be a difficult champion to get right. And we also learned that, you know what, it's OK to take a champion back to the lab multiple times if we really feel that it's worth the effort.

We knew that his theme was compelling, but there was something that wasn't just matching up between his mechanics, his visuals, and his theme. And this is one of those things is that everything on paper can be totally 100% right, but if players don't agree with it or players aren't feeling those goals that you're trying to hit, there's going to be a problem.

So we're able to collaborate with our subject matter experts, with our players, in order to get the experience right. And I mean, I'll say it again. Just because the mechanics stacked up correctly does not mean that it necessarily will in the feeling, which is why you play test it in house, why you play test it with other people at your studio or at your school, and ideally, why you test it with the people who are actually going to play your game.

And then we also learned that you can sometimes have these trade-offs. It goes back to what you hear from players. Sometimes they're going to make suggestions and you have to look for the note behind the note. It's like, what is actually the problem?

You guys are the subject matter experts. You're going to be able to look at what they're saying and never take it as, from players, a criticism. Always think about it as if they said something wrong, there's something that has to be causing the problem. What is it underneath their note that's actually causing that problem? Let's look for that thing.

And we're still iterating after launch. That's one thing that's typical with our champions is that yes, we do play tests with players. Yes, we do play tests in house. But once we launch the champion, there are thousands and thousands of play tests with real players on live. And that's when we learn about things that happen maybe at scale or under very specific situations, or for things that we didn't account for, for these external validity concerns.

So we keep iterating. And in the case of Lucian, we actually continue to collect metrics and user sentiment through things like surveys and through game data to understand how he was doing. And based on this, we're able to make more iterations, fine tune, and hopefully make him a little bit more accessible.

And today, Lucian is well regarded, widely played. And if you watch the professional matches in the world championship, he's use quite frequently. All right, you guys, I hope I didn't bore you. My name is Genevieve, again.

If you guys ever have any questions, you can feel free to reach out to me at gconley@riotgames.com. My *Summoner* name in game, if you play, is RiotCapernoited. And you can also find me on Twitter there. I'm always happy to answer questions. Thank you guys.

[APPLAUSE]

AUDIENCE: We get a 10 minute break.

PROFESSOR: So 10 minute break. Come back at 2:20 and be ready to start working on your focus test stuff. OK, welcome back to class. Now that you've had a chance to be talked at a whole lot, we're going to go ahead and ask you to actually do.

The plan is to give you about 10 minutes to meet in your groups, plan your test. I actually strongly recommend you just grab a copy of the focus test, either in the Word version or the RTF version of off Stellar so you've got that as a guide. Sit down, plan your test and how you're going to run it.

We'll give you 10 minutes for that. Then we'll have about 45 minutes for people to circulate and test. And one of the things you should also be planning is when your team members will be out testing other people's games and when your team members will be observing.

Just given the number of people in class, the number of games we have, and the number of people on the team, chances are good that you aren't going to be either observing or testing all the time. You're probably going to have to wait. But go ahead and circulate, take the time to watch other teams doing focus testing.

Try not to watch a game in progress that you're going to focus test. If you realize that you're just sitting around waiting for a chance to focus test, let the team know that you're in line next.

I'd rather you pulled out a book and read it than watched someone else play a game you're about to focus test. It will be more useful for the poor people who you're testing for if you haven't already seen somebody do all those things.

That doesn't mean hide in the corner and read a book because there's too many people wandering around testing games. We do expect everybody to participate both sides. Observe and test. So try to balance that out a bit.

We'll circulate and keep an eye on things. And if it looks like at some point nobody's actually testing, you're all sitting around twiddling your thumbs, we'll end it early. I don't know how long this testing session will take because we haven't had such large teams on this project before. So it's a little harder for me to know what the amount of time is.

After that, we'll give you some more time to talk in your own teams and come up with the what are the results of your test to analyze your data. And then we'll ask somebody to come down and for each team give us the summary of your focus test report.

How did your focus test turn out? What did you learn? Do you have a plan for what you're going to do with it, or are you still figuring out how you'll deal with it? Those are all reasonable things given you only had 10 minutes to look at your data.

Thank you. It is 2:23. At 2:33, we'll go ahead and start focus testing. Actually, I'm going to give you 15 minutes to get set up, because I just realized you probably need 10 minutes to talk and then probably five minutes to make sure you have a machine up and running. So I apologize, it's going to be 15 minutes to plan and prepare for your test. So that's 2:38 at this point.

[CHATTER]

AUDIENCE: Yeah. I think he just wants to--

PROFESSOR: OK. You should be ready to go now, I hope. And actively go ahead. You should have a computer ideally set up to run your game. Who is actually ready to run a focus test?

PHILIP TAN: Everyone who is manning a demo computer, a computer ready to run the game, put your hand up. All stand up so that we can see you.

PROFESSOR: Yeah, go ahead and stay up so we can see where you are.

PHILIP TAN: I'm not seeing any hands from these teams over here. OK. All right. All right. All right. As I said that, someone is standing up back there. How about the Sparkly Redemption team? OK, all right. There's someone there to run a game. All right.

PROFESSOR: So close all the laptops that are not actually running the game would be my advice, and then let's go ahead and get started.

PHILIP TAN: All right. Start playing each other's games. Go.

PROFESSOR: Yep.

[CHATTER]

PROFESSOR: Next time, we'll ask them to have two or three.

PHILIP TAN: Yeah.

PROFESSOR: As part of the Stellar thing. He was talking about the--

AUDIENCE: I got the FOA for free.

PROFESSOR: OK. It's OK if you don't have a plan of action as you're looking at your results as long as you've gotten ideas to what you're seeing out of your data. It's hard to look at your data, get it evaluated, talk about it, and figure out what you're going to do about it all in 10 minutes. We are giving you a short amount of time because we're running out of time.

That said, let's get started with *Sparkly Redemption.*

[APPLAUSE]

AUDIENCE: So we had a small hiccup when we started where we didn't have-- part of our core game is collecting sparklies, and we didn't have a version with that when we started play testing. So we actually got to test out some different movement controls that we were trying out, which actually turned out to be good because the controls that we liked, which was right clicking with the mouse and then left clicking to shoot, a lot of people had a lot of trouble with that.

So it was a good thing that we messed up and then got some information from that. Then we got a version with sparklies working, and there was a big bug where if you killed all the monsters, they never respond. So that bug has now been fixed in the last 10 minutes, so that

was good.

Other than we, we got just random things like people staying in the corner and just shooting, as opposed to just moving around the map like we want them to. So we have a few problems that we have to figure out that. Right now, we don't know the answers.

[APPLAUSE]

AUDIENCE: Hi. I'm with *Lost Underground.* As a refresher on our game, the idea is that you are lost underground in the darkness trying to find an exit. And so as part of our game, we have this sphere of visibility around you. And as we play tested, we got a lot of good feedback on what that visibility means to people and how we can actually use it to make our mechanics and our [INAUDIBLE] design more compelling.

Another big issue we ran into with people play testing was our collision boxes are basically pixel perfect. And so people would get caught trying to go through the map, which was really helpful for us, because although we already knew that this problem existed, we actually learned a lot about how people were actually trying to make these decisions.

And we learned that instead of using the idea of the entire [INAUDIBLE], a lot of players were actually looking at where the feet were planted and trying to move based on that. So when we go and fix it, we'll make sure to put the collision box around where the feet are. We also got a lot of good feedback on the items that we have in our game and where people are actually focusing their attention.

[APPLAUSE]

AUDIENCE: [INAUDIBLE].

PROFESSOR: Oh, *Plunder Winds.* Yeah.

AUDIENCE: So our game involves moving around a grid and running into random encounters. The favorability of the random encounters are indicated to the player by numbers and colors on the board. So one of the biggest things that we got from testing was that people didn't understand the role of the numbers and colors played or how they were related or if they were related.

And they began by moving around randomly and just waiting for things to happen to them.

There's also a mechanic in our game that involves moving with the wind and against the wind. And so moving with the wind doesn't count stamina, but moving against the wind does.

Or moving sideways of the wind does. Moving against the wind is not possible. People had trouble figuring that out as well. So one of the biggest things we're going to try and do is more visual feedback to the player, help them figure things out as it goes.

Also, people felt like there should be more long-term consequences instead of just planning for the next move or two. And so we're going to see how we can integrate that into our game.

[APPLAUSE]

AUDIENCE: Hi. *Blind Aliens.* So we got a lot of great feedback from the use play testing. So one of the main things we realized is we needed more clear goals and more long-term goals.

So for example, our gun automatically reloads, and that was a little confusing for people because they didn't really see that it was reloading. But instead of doing something like that where the gun automatically reloads, we're thinking of adding bullets around that you have to go and collect so it's more short-term goals and also long-term strategy. How do you save your bullets up, when do you use them all.

Another thing we noticed was that our game was way too hard at first, and way too easy after five minutes. So we decided we need to focus on ramping up the difficulty. Instead of starting with three aliens attacking you, maybe only one. And then maybe after five minutes, instead of only having three aliens, maybe 10. [INAUDIBLE] difficult.

So another thing we focused on was our AI. Not AI. User interface. So I talked about how there was a reloading thing, it was flashing in the corner, but no one saw it so no one knew that their gun was reloading. They thought it just wasn't working and they were frustrated that they were clicking and nothing was happening.

So we need to make that more visual. Along with our aliens, when they're half-dead or they're still attacking you. People thought they were dead. They'd run into them and they'd die and be frustrated.

So instead of maybe turning them darker, maybe turn them red or something like that to make that clearer. So thank you guys for all the feedback.

[APPLAUSE]

PROFESSOR: *Modudice.*

AUDIENCE: Hi, we're *Modudice.* So a lot of the feedback we got was really based on our UI. And it's because we're modeling a 3-D die using 2-D graphics. So a lot of people have different opinions about how that should look, and no one thought that the way it does look is OK. [INAUDIBLE].

And so we really got a lot of opinions. Maybe you could do it this way. Maybe you could do it this way. And that really helps, because now we can really build up a discussion of how it should look. How we can make this mental model in people's heads match what they're seeing on the screen, which is the real hard challenge that we're facing. So thanks for the feedback.

[APPLAUSE]

PROFESSOR: *Beaver Evolution.*

PHILIP TAN: I told them it was going to be the problem, someone needs to go up.

AUDIENCE: So through play testing, we really realized that we weren't conveying to the user exactly why all the instructions worked clearly and what was happening at each turn. But otherwise, the game mechanics worked pretty well.

So in *Beaver Evolution,* you can choose whether to build, populate, or evolve, but it wasn't clear to the player exactly how many beavers you can populate or how many dams you could build, or what natural disasters could happen. And then between each turn, when a natural disaster happened, there wasn't that much engagement between the player and the game being like, oh, this drought happened and then this is a consequence of that.

So I think going forward, we're going to really try to make it easier for players to understand what happened during that turn, and then what their choices are in terms of picking it. So a lot of it is clarifying what the instructions are, what the game mechanics are.

[APPLAUSE]

PROFESSOR: And finally, *Comcastic.*

AUDIENCE: Well, the two main pieces of feedback that we got from players were about the UI of the game. And we had some issues with that because our game has a lot of complex UI requirements and we didn't have time to implement all of those prior to this test session.

But we did get a lot of feedback about UI improvements that would make it easier for players to see what was going on with the game. For example, highlighting houses that are currently being served by a [? source ?] center. The other main complaint that we got was that the game is nebulous right now.

There's not a whole lot of challenge or clearly defined goal. And so one thing that we're really going to look at going forward is tightening the focus of the game, addling more tension, and making the goal of the game more concrete. And since it doesn't fit very well into an established genre, we had some more complex design decisions to figure out how to make this a fun and engaging game.

And so that's pretty much what we're going to be focusing on going forward, would be the UI and adding more challenge and tension to the game.

[APPLAUSE]

PHILIP TAN: I have a few words. Do you have anything to say?

PROFESSOR: You go for it.

PHILIP TAN: OK. So I just want to remind everybody that you started on this project less than 10 days ago. OK. And you just finished your first playable prototype, and you did a good test. So give yourself a hand.

[APPLAUSE]

Let's see. Things to keep in mind. For those of you who are making mouse-driven games, you want to make sure that you have enough mice to have for your next play test which will be probably project three. And you've only got another five days, if I'm not mistaken, when this is done. This is exactly a two week project.

So if you haven't started cutting features, this is probably the time to do it. Let's see. Something that, of course, we did notice and you all felt it was that there weren't enough stations for playing the games. For future play tests, for project three onwards, we are going to

expect at least half as many game computers as you have at the average team size.

So I'm not sure whether we're doing with teams of six or teams of eight. We're doing teams of six for project three, which means each team should be prepared to have at least three computers running with the games. Something else that we're also going to make it a lot clearer where the game stations are is that the computers that are not being used for game play should actually just be closed.

Take notes using paper, because it gets confusing to figure out which are the game stations and which ones are just like people taking notes and which ones are the ones where people actually coding on the sly trying to fix last minute problems. You really should have all that sorted out before the play test, not during the play test.

So next time when we're doing play test, just make sure that half your team, basically, is ready to go with the game. And I would suggest more than half your team should be able to run the code, just in case someone falls sick, right? And doesn't bring their computer or their mouse.

Rest of the class is for you to work on your own team. If anyone hasn't signed into the attendance sheet, we still have it up here. And I want to have everyone give the folks from Riot a big hand for coming and giving their feedback.

[APPLAUSE]

Thank you.

GENEVIEVE CONLEY: [INAUDIBLE].

PHILIP TAN: Oh, yes. They have lots of information about internships. Oops. Oh, whoa. A very, very large stack of documents and positions that are available. And a deadline, October 15th, which is not that long from now. So definitely check those out as soon as you can.

Let's see. Actually, while we're on internships, there's one other thing that I wanted to mention. I sent out a message for Europe over in AeroAstro. They are looking for game developers. Even better if you also know MATLAB, but they just want game developers for a high school science competition that they're running.

It's like first robotics. It's a robotics competition, only in zero gravity. So if that sounds cool, you

should definitely respond to them quickly because Europe deadlines are coming up. And that's it. See you [INAUDIBLE].