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.

PROFESSOR: So before class today, you should have turned in your vision statement and your product backlogs. Wednesday, you should be turning in a copy of the sprint task list, and you should have a prototype ready for testing in class.

Remember that while we're offering you one in-class testing session for this project, you need to do at least one more testing session out of class for the two focus tests that are required for project three. And when we look over your focus test reports, we're not going to be impressed with focus tests that were done so late you clearly couldn't respond to them.

So the project is due Wednesday. Don't do a focus test on Tuesday, because we know you couldn't do anything with that information. So sort of plan out when you need a focus test accordingly.

So today is class, we're going to open with just a very quick two minute presentation from each of the groups. Since you've turned in your product backlogs, and hopefully you have finalized your vision statement, you can come and tell us, what is the goal of the game you're now going to make and what its major features are. That's all you need to do for that.

Then we'll have a fairly brief project management check-in. We'll sit down, I'll stand up here, and we'll just do a little bit of talking about what is your team doing for project management. Is it working? Should you change something? Then Philip will be giving the usability lecture. And after that, we'll take a break.

If anybody has something that they would like play-tested, after the break, we'll come back and have an opportunity for some play-testing and working on your projects in class. If you don't have something to play test, that's OK. This is sort of a hey, if you actually had your act together, get a little extra play-testing in. Wednesday's the day you have to have something that can be play-tested.

Any questions? All right, then. Let's move on to presentations. Actually up there? Yay. It is. First up we have Hardcore Dragon.

PRESENTER 1: OK, so I'm on the team for Hardcore Dragon. We've changed from Dragon's Lair to Hardcore Dragon. And the point of the change in the name is to reflect the slight change in the direction of the game.

One of the new elements we've added as a win condition is collecting an entire set of hardcore armor to be the hardcore-est dragon in the land. And so we're intending on adding a little bit of a cheeky, funny, dialogue flair to the game to make it a little less serious.

Outside of that, the main concept of our game is that you're a dragon, and you have a horde of gold that you are protecting and also trying to grow. So the player can make decisions between attacking villagers to gain more gold and also staying on their lair to be protected. And we're also adding elements of melee combat so that the player is not only making these decisions cognitively but also has to act on them and actually go and perform those actions, rather than something that's, like, just click, oh, I want to do this action, something like that. So we're introducing elements of strategy and melee combat and also collectible funness of RPGs.

And then elements of our backlog-- we have the player's desire, such as the playing wanting to be able to attack, defend, and also stay alive to fight, despite the attacking armies. And we also have a lot of focus on making our game usable and easy to understand for the player.

PROFESSOR: Thank you. DNA.

PRESENTER 2: Hi. So probably should have written on the board the project previously known as DNA, because we've completely reskinned our game. So it's no longer about a biology type thing. It's about exploring space.

So DNA is now the name of your mothership, And we're going to put you as a small spaceship-- RINA, so R-I-N-A, basically. RNA. And your goal is to travel across different planets and eventually make it back to your mothership before running out of fuel or exploding on a bunch of asteroids.

So our game has two primary mechanics that we're working on. First is a menu-type screen where you can see different planets that you can head towards on your way back to your mothership. And each planet would tell you-- if you hover over each planet, it will tell you, at this planet you can pick up X fuel, or you can get this jet boost type thing. And so the strategy

comes from the planning for which planet to go to relative to the other planets. And the trade-offs also come with which planet do you go to, or what do you pick up at each planet.

And the other mechanic is actually getting to the planet. So using a stat boost and/or fuel that you picked up from each planet, you have to navigate through a platform, or not really a platform. More like *Flappy Bird* or a *Copter* type level, where your stats increase, like how fast you go through the level or how maneuverable you are and how you can avoid obstacles.

So our game has two mechanics, and that's what we're kind of working on right now. We have the moving from one level to the next motion part down, so now we're working mainly on balancing the game and seeing what should the stat caps be, or what should the starting caps be, so it's not too boring at the beginning, but it's also not too hard at the very end. And our backlog has a lot of art.

PROFESSOR: Thank you. Ghost Maze.

PRESENTER 3: So for Ghost Maze, our design for the game hasn't changed much. We solidified our original idea, is that you're going to start off in the middle of a dark maze and you have very limited visibility. You're going to explore the graph, or the maze, and basically find the exit that leads you out of this dark scary place.

You have a sanity bar that goes down the more you are in the dark. And you have the ability to increase the brightness of your lantern, which would give you a higher radius of visibility or decrease it, which would make it lower. But the higher it is, the less slowly your sanity goes out, but the faster that your battery dies. So you want to optimize for that-- how fast can I explore this maze versus how slowly do I need my sanity to drop. And you can completely have it off if you want, because they're battery. But that would be chipping away from your sanity much faster.

There's ghosts in the maze, so you can't just do the bug around strategy, which would always guarantee you to get out of a maze. But there's going to be ghosts at strategic, random positions. These ghosts were originally intended to have AI that would have them find you or attack you or whatever. But we changed it so that they're just patrolling in a predetermined pattern so that you can focus on the one mechanic of the game, which is moving around and finding a good path, as opposed to escaping from things chasing you.

The one change we made, the biggest change we made so far is we switched from Flixel to

Unity 2D because we couldn't get Flixel running on most of the computers we had, and it just brought so many complications. So we ended up switching to Unity 2D.

PROFESSOR: Thank you. Score High.

PRESENTER 4: So Score High focused on simplifying our game. We realized we were probably overextending ourselves with the number of features we wanted. So even though our goal stayed the same, we kind of simplified it, and I'll talk about that in a minute.

So Score High's goal is to survive the week of school. If you can do well, even better. And that becomes difficult because as a player, you have your stats to maintain. You need sleep, and you need food. But you also need to not fail your classes and get kicked out, so you need to do your work.

So the main features of the game are you play your stats, so how are you doing as a person, your schedule, which is going to list your classes and what PSETs are due and when they're due, and then also the map. You, as a character, go around the map, and you have to go to certain buildings to do your work. And as I said, the trade-offs are managing you internally, so you have to go to your dorm to sleep, but your PSET might be due in two minutes, and if you don't run across campus, will you be able to get it done and pass your class? In terms of updates, I think that's about it. Thank you.

PROFESSOR: Thank you. Build a Space Car.

PRESENTER 5: So originally we were Build a Car, but we decided that the game would be better in space, so now it's Build a Space Car. The idea is that you have some kind of spaceship that you need to add components to, and then you send it out on a mission. Out on this mission, it'll run into random events, and you'll get a report of what happened. And then it'll come back and you need to iterate, add more components to either make it faster or stronger or have more attack. And then that way, you'll be able to succeed on missions.

The trade-offs come into what components do you add, and what effect does that have on the stats of your spaceship, and the fact that the more components you add to your spaceship, the more it costs and the less money you're going to make from going out on that mission. The main product backlog features that we have are the things like adding components to ships, coming up with stats for all of the components that we have in the game, and designing a system where the ship will go out on missions, get some results from all of that, and then

loop back into this initial building phase.

PROFESSOR: Thank you. And MIT Sim.

PRESENTER 6: All right. So the idea of our game is we play as an unnamed dean for student life at MIT. And you make and manage the decisions at MIT, and you have to trade-off all of these various resources that you have, like student happiness and the endowment and whatnot. And you have to please the MIT president above you and the MIT corporation above them. And then you also have to please the parents of students who might hear of some crazy things happening and you have to respond to them.

So for example, you might make shoes mandatory on campus. And suddenly East Campus revolts. But maybe it improves your standing with the parents or something. So that's the general gist for our game, yeah.

PROFESSOR: OK. So it's good to hear that people are both refining and changing and scoping down and kind of figuring out what they're doing with their games. That's a good start.

The next thing I wanted to talk about, ever so briefly, is-- I've been reading through the project write-ups from the last project. Haven't gotten all the way through them. None of us have. We're still working through.

But a lot of people are like, ugh, project management got inserted into the middle of the project, and it was a really hard. And it made things clunky and more painful. And yeah, I bet it did, and I apologize. What I'm wondering, however, is what are you doing for project management on this project, and is it working better So far?

I realize we're only partway through, but I'd kind of like to hear about how has your team decided to handle communication. I know that meetings were a big problem last time. Is the product backlog being a useful thing to sort of help define your design? Or are you doing it because we're making you turn one in? How are assigning tasks and sharing work going?

So I can either do this as a discussion, or I can call teams one by one. Do you guys have an opinion as to how you'd like to run it, or should I go ahead and make an executive decision, since I'm up here? All right, I'll call on people team by team, and we can just sort of go through. I'll start with MIT Sim, because you went last last time, so I guess you can go first this time.

You don't have to come down here. You can stay in the seats. But OK. OK, a microphone might help. I wasn't trying to make you do an extra report. I was hoping this would be sort of a class discussion where you're--

PRESENTER 7: Well, I'm already down here.

PROFESSOR: All right, go ahead.

PRESENTER 7: So I think one interesting thing about project management for our team as 611 teams is that the driving difficulties of working on a team like here are very different from what might be in the industry. So we have to adapt all these industry methods, which might work really well for a game company, where you have your eight hours a day of working. But here it's much more sparse and not necessarily-- it's not always one to one how you would manage an industry project versus this project. So I think just figuring out how to adapt these methods for students who are really crunched for time is a challenge.

PROFESSOR: Have you made any adaptations? Has your team made any adaptations or changes, or are you just trying to do as little of it as possible to help keep things moving?

PRESENTER 7: Probably the latter, yeah.

PROFESSOR: OK. If you can think about the adaptations you're making that are working for your team and share them, that would actually be really helpful for us to hear, and for actually everyone to hear, because I think every team is running into the same problems you are. But the questions is, how do you take these, and how do you make it work with a team you actually have and a reality you're actually working with, as opposed to, this is the way your should do it. That's actually the really important thing here to try and figure out. Thank you.

OK, Build a Space Car. Do you have a sacrificial victim? Scrums masters are a good person. Yeah, it could be a scrum master. Could be either way.

PRESENTER 5: Our producer slash scrum master is actually out of town, so I guess I'll handle it. But basically coming off of the last project, we saw a lot of things that we weren't happy about and that we saw a lot of problems with the way that we communicated. So this time we're trying to make sure to stay in communication as much as possible, let people know what we've been working on and the problems that we see.

PROFESSOR: What tools are you using to stay in communication?

PRESENTER 5: So we're using Trello for all of our tasks. We have them there. We have them color-coded for different groups of coding or UI or testing, things like that. And we have a Google Doc that's sort of all of the ideas that we had and sort of the design decisions and things like that.

We're also all programmers, which I think has been helpful so far, in that we sort of all understand what goes into the task and what's reasonable and things like that. Not to say that there isn't a place for non-coding people, just that it has definitely helped on such a time-constrained team.

PROFESSOR: Yeah. When you're all talking the same language, and you all come to the same estimate assumptions quickly, that's beautiful. Thank you. Score High.

PRESENTER 4: So I think one thing we realized right away was that meeting outside of class was pretty much impossible, especially when you have seven people with completely different schedules. So having the product backlog and that list of things was really helpful.

But at the same time, a lot of the other project management stuff is more just on the side. So for my team, I've been taking care of it and trying to let everybody else program. But the product backlog, I think, is really helpful. And the vision statement, having that breakdown of who's doing what, has been pretty useful. We use an email thread, just to keep everyone in contact.

PROFESSOR: All right, Ghost Maze. Two people can come down. That's fine.

PRESENTER 8: Yeah. So I guess as other people were saying, it's been hard to coordinate meetings in person. We tried to have one on Saturday, but only half of us could get together. So definitely we have to rely a little more on online communication and such. And people are working on different schedules.

So I think one thing we realized is you have to budget how much time people have at certain points in the project. So if someone has a lot more time, maybe they're more free on the weekend, then they could get started on the earlier stuff, whereas if someone has a big project due today, then we would give them more stuff for later in the day. Because when you develop a game, you can't do everything right away. Some other things are dependent on other things.

I've been taking the role of scrum master, project manager. So figuring out delegations and responsibilities. So I think it helps to have one or two people plan that.

PROFESSOR: Thank you very much. OK, and DNA. In space.

PRESENTER 9: That actually was the other working title. It might become Finding DINA.

PROFESSOR: Pretty much, whenever you tack "In Space" to something, it becomes infinitely better.

PRESENTER 9: I'll do that now then. So meetings are hard because people have lives. But we do little meetings where we eat lunch with each other, and only two or three of us is there. And the lunch is tasty, and we talk about cool stuff like game design and weird bugs and what exactly are we doing with our lives?

As you may be able to tell from the fact that things have changed drastically from the pitch to now, we kind of basically scrapped everything at one point and said, re-brainstorm. Shoot out all of the ideas. But of course, since we already said the word DNA, everything kind of ended up DNA deep down inside at heart, kind of sort of.

So I feel like we did a lot of email communication then, and we still do a lot of email communication now. It's just like, spew all of the things at each other. And I feel like, that's kind of nice. There are times when it's like, well, I'm really, really, really busy until Day X, so until Day X, I will have nothing to say if we have a meeting. And we're like, OK, then I guess we won't have a meeting until Day X. And it's like, well, Day X is the day right before Monday or something. So was it really the best idea to put that off, or should we have done it anyways and have you just sit there and feel lame about yourself? I don't know, but we did what we did.

PROFESSOR: OK. And finally, Hardcore Dragon.

PRESENTER 10: So looking up at that list, the first thing on the list is team communication. And we actually have quite a few different things in place. We're using Trello, so we do have that. We're using a Google Folder, which is just shared between all of us and all of the documents. And we're also using a chat, which I assume every single one of us just has open on our computer and is just sitting there. So everyone's looking in on it once in a while, people are commenting on it once in a while. That's how team communication is going. Design, I think, is probably the most solid point of all the things.

PRESENTER 1: Yeah. I think we came up with pretty solid ideas the last time we met in person. And since then, we haven't really deviated from that. It's been pretty straightforward that this is our goal. We might have to cut-- we'll probably have to cut features eventually, but for now, it's pretty

solid.

PRESENTER 10: I think we're fine without cutting features. I think it will actually fit right in with the time we have. I don't think we can add features, but we-- and I think I know exactly what the game is at this point. And I think that everyone else is in the same place, but I don't know. So I really think that figuring out what our design is is completely solid at this point. As for assigning tasks, sharing work, that stuff is sort of a work in progress.

PRESENTER 1: So we have a sprint task list already that we've been using to assign estimates and people to certain things. So far it's kind of been a free-for-all. But from looking at the task list the last time I saw it, it looks like everyone's completing things in a pretty good pace. So I think that so far it's worked out nicely.

PROFESSOR: OK, I guess you're done. OK, thank you. I think that's everybody. And now, Phillip.

PROFESSOR: OK. So just a reminder that this is project three, digital prototype two, user interface. So in addition to all of the project management stuff that we've just covered in the past half hour, there's also the intent to maximize usability. And we've already talked a little bit of usability when Sarah gave the last testing lecture. Now we're going to do a bit of a deep dive. So this is going to take a bit of time, but I hope you will follow me through this. You're supposed to use user feedback through user testing on your own, which means not just relying on the testing that's happening in class, but also doing it outside. So we're going to talk a little bit about that.

Before that, I'm going to walk you through an exercise about what's the intent of usability. The whole point of testing for usability and designing for usability is basically just to figure out whether someone can figure out how to play your game. Not so much whether they're having any fun doing it, although obviously you don't want them to be frustrated or confused. But you actually are just trying to figure out whether people can understand how to interact with your game.

The game could be improved through play testing to sort of say, oh, the challenge level might not be right for instance. You can use focus testing to be able to figure out specific design issues in your game, like for instance, do these armor upgrades make sense to somebody who's not on our design team? Usability testing is we have this thing that we designed with the intent of having it learnable. And can people actually learn how to play this game?

Now, just a straw poll-- how many of use speak or read German? OK. How many of you have

seen this game, or played this game or have heard of this game? All right. So if you do speak or read German, just hold your responses for a second.

And I'm going to start by just looking at this game. And this is a game that you'll find on a shelf, maybe in a speciality game store or an Amazon shelf. And just by looking at this, what can you intuit from what this game is or who it's for or how it might be played? Any ideas? Throw it out?

AUDIENCE: It's for young-ish kids, where they stack randomly-shaped blocks on top of each other.

PROFESSOR: Wow. How do you figure that?

AUDIENCE: Because it's bright and colorful and it has cutesy cartoon stuff on it.

PROFESSOR: OK, bright and colorful. Cutesy color stuff.

AUDIENCE: The animals are making funny motions, implying that they are precariously balanced. Also, the alligator isn't eating anyone.

PROFESSOR: The alligator isn't eating anyone. It is kind of nonviolent, bright color. What else on this stuff here, assuming that you don't speak German, gives you any other clues. I thought I saw a hand up. Yeah?

AUDIENCE: The type.

PROFESSOR: The font. The cartoonish-- yeah. I think, Matt, you had your hand up.

AUDIENCE: I was going to say something, that there's an exclamation point.

PROFESSOR: Oh, yeah, yeah. It's like, you can imagine there's something zany going on. Anyone recognize the brand? Has anyone heard of this brand before? No one has four-year-old toddler relatives or anything? Because it's a big kids brand.

OK, folks who do speak German, what's on this box? What's this box telling you?

AUDIENCE: The title is Animals on Animals.

PROFESSOR: Animals on Animals. So yeah, again, you're putting animals on top of each other. If you want to open the box, and I've actually gotten a box, just one second. You see these kinds of pieces, brightly colored wooden pieces. There is a crocodile.

But there's also this bright red die with these symbols on them. Any clues on how this game is

supposed to be played? I mean, we already talked about putting animals on top of each other. But here are the pieces. If anybody actually wants a closer look, you can come in and have a look. I'll go back to the previous slide for you to see. Any idea how this game is supposed to be played? Anything more specific than put animals on top of each other?

AUDIENCE: I feel like you're probably going to take turns, and every turn, you're going to roll the die and do something based on what it says.

PROFESSOR: OK. And that's based on? Making that assumption based on? The fact that there is a die and you've probably played games with dice before. And actually, in the previous picture, I think it actually says players-- is that a player number? I guess it's off in a corner. It is a multiplayer game. And if you could see that corner, you would've assumed, yeah, taking turns seems like a normal thing. That's based on your own experience of having played board games.

What else? Any idea what these symbols might be? Yeah?

AUDIENCE: Based on the hand symbol, it seems like it might be restricting which hand you can use to place the animal.

PROFESSOR: OK. So you are already assuming that this is a very dexterous game. It's going to require item manipulation, a hand connected to that. There is a picture of the crocodile. There's a question mark.

I'm sort of walking you through the process of somebody actually looking at this box for the first time. And sometimes when you see this kind of game in the store, you don't even get a chance to look inside the box. But if you had a display version, you'd be trying to figure out the game based on these things that you're seeing.

Now, there is a newer version of this box cover that makes the stacking animals on top of each other really, really obvious. That's the same cover that I've got over here. And this is the English version, Animal Upon Animal.

But if you wanted to figure out how to play a game and you couldn't look inside a box, what's the next thing you do after seeing it on the shelf? You look at the back of the box.

And there, everything that you've intuited from just looking at the components and looking at the front of the box is made even more explicit at the back. Now, it's in a tiny little font, but you've got a picture of two charming kids who are stacking things. Notice, they're stacking

everything in one plane. There are some pieces scattered around, but the only things that are being stacked are stacked within this single plane of the crocodile.

And of course, there are indications of how long the game is supposed to be, how many players it takes. And of course, if you were to actually buy the game, you would look at a manual, which happens to have six languages in it. And it would explain things, like what these icons on the die are supposed to be.

But ideally you wouldn't even get to this point. And the kid is not going to read this, because I believe this was targeted at kids age four to 99. So if you're age four, you're not reading this manual. A parent is going to be reading this manual, or some sort of older sibling is having to explain this to the kids.

So this isn't actually targeted at the target audience of this game. Maybe it's targeted at the buyer of the game, but certainly not at the people who are going to have the most difficulty playing this game even though the game is intended for them. By the way, this is a great game. This is a whole lot of fun to play with family.

So a lot of what I'm going to discuss going forward is just kind of a theoretical analysis of that process that we just went through. We just used a whole bunch of different cues, just looking at the box and the pieces, and the text on the box and the colors and cultural knowledge to be able to figure out how it is that you play this game.

Now, this is a game for kids. It's not a very complex game. If I gave you some sort of really, really abstract German board game, where everything are cubes-- and there are many, many games just fitting that description-- you'd probably have to read the instructions just to be able to figure out how to play the game. But ideally, you'd want the experience to be about as easy as the process that we just walked through with this board game. You look at this game, you're presented with information upon launching it or even before you bought it, and by the time you've actually started playing the game, you already have a sense of what you're going to expect, which sets people up to learn the more complex things that you need to teach them along the way.

Now, there's a book-- which I should have brought upfront. This book I strongly recommend, *The Design of Everyday Things*, which was previously titled *The Psychology of Everyday Things*, because that's really what this is all about. Usability, designing for usability and you

are testing usability, you are asking questions about the player's psychology. And you are trying to use that to your advantage to be able to help them understand how to manipulate your game and how to get the experience that you want out of it without being confused.

Now, there's this model that Donald Norman, the author off this book, goes into great detail. And it says, there are goals, there's an intention that the person using an HCI, a Human Computer Interaction system. And that person has some sort of goal. They are using Excel because they want to balance their budget or something like that. They have an intention. They're going to put numbers into the spreadsheet that reflect their actual spending. They have a plan, which is they're going to collect all of their receipts together and look at those bottom lines and start typing them in. And then they actually execute it. They're actually-- all right, I think this is the way how it works.

And then the computer gives them some sort of feedback-- visual, audio, vibration, if you're designing a game. The person using the system has to perceive it and then figure out what that means. OK, a dialog box popped up on my screen. I did something peculiar and didn't expect that, and then they have to evaluate, did that get them any closer to the goals that they had. And maybe then the goal changes to maybe I should learn Excel before I try to balance my spreadsheet. And then now they have a slightly different goal.

So you can translate that into games. You start off with the goal. How do I win this game? How do I lose this game? How do I avoid losing, and how do I win this game, or at least complete it? And then the player decides, what do I want to do and comes up with a plan, which is how am I going to do that thing that I want to do, and then tries it.

The game reacts, gives some sort of feedback-- audio, sound, vibration, whatever-- but the player still has to perceive it. And this is a big point right there. If you give the player feedback and the player doesn't perceive that feedback because they're too distracted by something else, because it's the wrong color or it's off in the corner of the screen where the player isn't looking, the player doesn't get that feedback. The player has to perceive the feedback and then figure out what that means. It's like, oh, a big giant exclamation mark appeared on the screen. What does that mean? I have no idea. And even if a player understands what that means, they have to figure out whether that's getting them any closer or further away from the goals.

So here's an example, back to my *StarCraft* fandom. You're playing *StarCraft*. You wanted a

giant army to roll over your opponent. And that's the goal. Actually, the goal is just to roll over your opponent. And what do I want to do? I want to build a giant army to accomplish that goal.

How do I do that? I'm going to build lots of buildings that is going to create all of these army units. And those army units are going to serve that purpose that I just described. All right, let's try it. Game reacts, and you get this.

What do I see or hear? The computer is yelling at me, saying not enough minerals. What does that mean? I guess I need minerals. And is that what I want? I guess. If I have minerals, then I can build those things that will get me the army that I want to roll over my opponent. All right, my goal hasn't changed at these long term, but my short term goal has. Now I have to figure out how to get more minerals.

So in *The Design of Everyday Things*, Donald Norman introduces this concept called affordances. And the idea is that there are certain kinds of interfaces, things around in the world, among the machines that you use every day but also within games, that encourage you to interact with them in a certain way. If you see, for instance, a glass wall, and your goal is to vandalize it, what's the most natural thing to do to that glass wall? You smash it.

All right. What if it was made out of plywood, and your goal is to vandalize it? You want to write on it. What else can do with a plywood wall? Concrete wall? You want to break it? Or spraypaint-- yeah, yeah because concrete holds paint pretty well.

So there are these materials that already suggest things that you can do with them. And you can take that concept further in a more explicitly designed way. So let's look at door handles. We've got some door handles around here. I guess there is this kind of crash bar.

You've got this sort of door handle, and it's just the right size for your hand. It's kind of curled in a pleasant way that affords gripping, that affords turning because there is a hinge on one side, and there's a sort of a lever action so you can use less force to be able to turn this thing. So you are immediately encouraged to gripping this thing.

But what does this door handle not tell you about the door? It's very important if you want to open the door. Yeah, do you push or do you pull this door?

So let's think of different kinds of door handles. This is something that you could grab and pull. But the designers of these intended to use this for you to push, because it's a lot easier for you to exert force going outwards than to exert force going inwards, using this kind of bar. That's

why you see these on doors that open outwards.

However, that's still a problem because there's a little rod all the way through, and your hand can go all the way around it. And it's still tempting to just grab the thing and pull it, which is a really bad situation if it's an outward turning fire escape, for instance. You don't want people trying to pull a door open when they should be pushing it when they're in an emergency situation. You don't want to have them do that kind of processing.

So that's why they came up with these crash bards. And eventually this one, which is very similar to what we have on our door. They have these very wide bars that you can push, and you can put your hand around it but not comfortably. It's much easier for you to use the flat of your hand to be able to push it outwards. And the version that we have on this store over there, the part that you push is off to one side, which tells you where you should be putting your hands so that you can get better leverage to open the door-- which way the door opens, left or right.

And that's the kind of thing that automates this process of how do I open this door in an emergency situation. These are affordances of the design itself. The material is mostly metal, although I guess the version that I've got up here has a rubberized grip. And the idea of a rubberized grip is that, well, maybe you want to put your hand on the rubber part, not on the metal part.

So you can design these things in your games, these affordances to sort of encourage people to, say, click on something. You can design something that's meant to be clicked on to look like a button, for instance, that's slightly raised. And Apple does this a lot, especially if you look at the older iOS design. Every single little icon kind of had this little glossy bump. It tried to give you this impression that this was something that was raised off the screen, even though of course it was completely flat. And it tried to encourage you that this was a thing that you can tap with your finger.

I'll go into more examples of this later. Anyone can think of something that you've seen in a computer interface that encourages you to use it in that way? It could be mechanical. It could be on screen. Look at your computers right now.

AUDIENCE: It's not like this on Mac anymore, but the scroll baller has a little-- I don't know what you call it, the lines that look like you can grab them and scroll.

PROFESSOR: Yeah, it had a texture, sort of three little lines that sort of gave you the idea that this was a raised thing for gripping, in the same way that a rubberized grip over a door handle encourages you to put your hand on it. It probably would make more sense nowadays, actually, given that so much of it has gone to touch screens.

The scroll wheel would be another one. I just broke my mouse scroll wheel this morning, so it's on my mind. Now, scroll wheels on mice actually do two different things. You can click them, and you can rotate them. But they're kind of really designed to be rotated. And that's why I broke my scroll wheel, because I kept clicking it. You are encouraged to roll that thing up and down. They're usually made out of some sort of rubberized material and a place right where your finger has easy reach.

You can flip that around and make things deliberately hard when you don't want someone to be doing something. Here's a screenshot of-- what game is this? *Plants vs. Zombies*. There are no zombies on screen, but it's very popular.

And early in the game, early in the development of this game, they had different values for these early items that you'll get right in the beginning. This is I think the very first level in the entire game, and they're just trying to get you to play this game.

Now, this is a tower defense game. You plant plants that do various kinds of offensive or defensive attacks against zombies who are invading your lawn and trying to get into your house. And there are things like sunflowers that actually create the energy that you need to be able to power the rest of your plants and to give you the resources to be able to put the rest of your plants down. But if the game starts like this, what is first a first time player who's encountering this game likely to do?

AUDIENCE: Put a plant in the dirt.

PROFESSOR: Put a plant in the dirt. Which plant?

AUDIENCE: A sunflower.

PROFESSOR: You think to put a sunflower first?

AUDIENCE: Or the pea shooter.

AUDIENCE: Yeah. The one on the left is--

PROFESSOR: The pea shooter.

AUDIENCE: Left to right.

PROFESSOR: Right. If they read it from left to right, then they'll probably start with the one on the left. Maybe there's a 50/50 chance at least that they will put the pea shooter first, which is the little green guy. And that's actually going to get the player into a lot of trouble real fast, because once they put that plant down, they can't do anything for a really, really long time, by which point you may have lost the game. They need to put the sunflower down first.

So designers recognized, oh OK, so we could just flip things around. But again, there's a 50% chance that they will just get it wrong. At it's the first level of the game. You don't want to turn somebody off the game that quickly.

So what they did was they reduced the cost of the sunflower and started you off with just enough money to actually plant that sunflower. They didn't actually flip the cards around because they figured, that doesn't really solve the problem. What they did was that they made it so that the only thing that you can do at the beginning of the game is plant that sunflower, because that's pretty much always the right thing to do in this game.

Now, the cost reduction might have been also out of other game balance considerations. Maybe they decided, wow, we really overpriced those sunflowers. We should make them a little bit easier for you to get. But this also serves a usability system. They want the people to understand, the first thing that you do on any level, plant that sunflower. Now, they could have told you that, but this is a constraint. They've made it impossible for a player to do the wrong thing right at the beginning of the game.

However, as the game goes on, this, and due to game balance issues, they started to come up with other plants. This thing is called a walnut, this brown thing over there. It makes a wall. It looks like a nut. And they figured through game balance, well, it's pretty much got to be about 50 solar energy, so we're going to have to price it the same as the sunflower. But we don't want players putting down walls at the beginning of the game. They don't actually need to. They should still be putting sunflowers.

So the solution that they had was, they have these recharge meters that fill up from the bottom. You can sort of see that this is right at the beginning of the game when you've got a

lot of different kind of plants that you can plant. But only the sunflower is fully charged up, and it's the only card that you can use. The rest of them are slowly filling up over time.

Right in the first, maybe, five seconds of the game, there is no other card that you can plant, even though the costs may imply that you can afford to plant the walnut right at the beginning of the game. You can't. You have to plant the sunflower.

So again, they've made it impossible for you to do the wrong thing. There's also that shovel, but you can't use the shovel right at the beginning of the game. The shovel right at the beginning of the game has no effect because you haven't planted anything. So that's a constraint. That's preventing someone from doing the wrong thing.

Now, if the decision of whether you do the right thing or wrong thing is a major game play issue and it's something that you want people to sort of agonize and to decide over, then you don't want to constrain it. You want to give them the option of making that mistake. But if it's an issue of learnability, if making the wrong decision makes your game hard to learn and hard to figure out how to even get ahead and how to make progress in the game, or even understand how your game works, then that's something that you should consider doing things like constraints and use affordances to sort of suggest, this is what you should do, constraints to demonstrate what you shouldn't do.

Now, that's affordances and constraints. Any questions so far? All right.

There are other things that you can do in addition to affordances and constraints that sort of suggest how something should be used. So this is from *Minecraft*. And how many of you have played *Minecraft*? OK, about a quarter of the class.

This is something that you see really early on in the game. The way how *Minecraft* is played, you are dropped off in a sort of wilderness environment. Everything's made up of blocks. So it's kind of like a wilderness environment made out of LEGOs. And the only thing that you can do is walk around and punch things. So you start punching the ground and you get some dirt blocks out of it. You start punching trees, and you start getting some wood blocks out of them, and they go into your inventory.

What you very quickly discover, I would say, in the first minute of game play, is that if you went into your inventory and looked at those blocks, you can move them into this little four-by-four square, just using your mouse and clicking and dragging them. And the square is named

crafting.

So OK, I've beat up 15 trees. So now I've got 15 blocks of wood. What do I do with them? Well, if I drag them into the-- I can turn them into planks. Is it planks? Yeah, planks-- just by dragging them up there.

OK. By dragging things into the crafting box, I can make other kinds of things. That's the craft part of *Minecraft*. The mining part is the punching part of the world. So you punch the world, and you make planks. You put planks into the crafting square, and you get sticks. And you can do all kinds of things, like make torches and make arrows and so on and so forth.

Later in the game, you encounter something-- you discover that you can make a crafting table. And you see a nine-by-nine square instead of a four-by-four square. Because you've already seen the first one, you've already seen this four-by-four square, the word crafting, and the arrow which points to the right, and now you're seeing this very, very similar thing over here, and inventory is still visible there, even though it's not just you anymore, it's you and this crafting table, the implication is that you should interact with this table pretty much in the same way that you interact with objects back when you were examining your inventory. So this is context that has been established to the previous things that you're seeing. And the things that you see around it, like the arrow and your arrangement of the boxes. And the word crafting-- you're sort of telling the player, treat this very much in the same way that you solved that previous problem. And then eventually you get to make things like pickaxes, for instance-- so giving some context to what the players are doing.

Now, there's one particularly powerful kind of context that you can set for any action that a player does in the game. And some of your games already have this-- and that's some sort of storyline or setting or narrative, some sort of fictional setting that your game is taking place in. And often this is done very, very poorly in games. You are being told your games are in space, and you're being encouraged to walk outside your spaceship without a spacesuit or something like that. And in your game, that's fine. You have no asphyxiation mechanic or something like that. But players don't necessarily assume that they should be able to walk outside in space and not die.

So when you choose your settings, when you choose your-- when you choose your storylines, you have to make sure that they are supporting those kinds of mechanics that you come in with. A lot of your games started off by what mechanic that you wanted to do. Make sure that

your storylines are actually supporting that. Can you think of any games, commercial games for instance that you've played, where the storyline was really at odds with what the players were actually being asked to do. I see some nods. Some hands? Let's call out a few bad ones.

AUDIENCE: Well, one this is that most-- playing games, especially Japanese ones, tend do have battle systems that have basically nothing to do with the story and have a lot of strange mechanics that aren't actually related. For example, the *Final Fantasy* games-- your character can summon these really powerful monsters, but for some reason you never actually use those to solve your problems in the storyline, for some reason.

PROFESSOR: Right. And I would say that's actually the difference between the better and the worse *Final Fantasy* games. In the better *Final Fantasy* games, they actually explain why that doesn't work. In the worse *Final Fantasy* games, you have all this power. Why aren't you just solving the problem instantly? That's a good one. Any others?

AUDIENCE: Well, I feel like the classic example is probably *Skyrim*, where you're a warrior fated to this grand destiny and probably just run around punching chickens for a while.

PROFESSOR: Well, I'm trying to remember. *Skyrim*, you start off like a prisoner or something, right? But I do remember some of the earlier *Elder Scrolls* and a lot of Western RPGs. It's like, let's go into the tunnels and kill rats, because you're the chosen one. It comes up.

That is actually, I think, partly fiction against fiction. It's like, you have the quest which tells you to kill rats. And you have the overall storyline that's telling you that you're the savior of the universe. And that's at odds. But there are mechanical issues as well. Why is your savior of the universe picking up all the cheese wheels in the world, for instance, which does happen in *Skyrim*.

Something else that you can do is expose more of your system. How many of you are playing *Destiny* right now, by the way? Just out of curiosity. A few hands, OK.

One thing that *Destiny* does that *Halo*-- how many of you have played *Halo*? OK, more. One thing that *Destiny* does that *Halo* doesn't do is that it actually shows you how much damage you're putting up with every single shot. A little floating number, in a very RPG-like way, showing you how effective your weapon is being. And that seems to be a necessary consideration by Bungie, who made both games, that, well, even though our games kind of play in the same way, we need to reveal this extra little bit of information, because the way

how weapons work in *Destiny* is a little bit less predictable than how they work in *Halo*. So we're actually just going to show you how much damage each weapon does every time you shoot it by showing you the numerical amount.

This is a screenshot from *SimCity*. And one of the neat things about *SimCity* is that you can kind of open the hood on all of the variables that the game is tracking at any given times, and they show you these lovely little bar graphs. I think this is, like, citizen satisfaction or maybe it's property value. I'm not sure.

But you can go into all of these different modes where you can look at your city through different lenses to see all of the different variables that the game is tracking and how they're interacting. So it takes a lot of work to be able to reveal this sort of information. And if you're not careful, you can reveal so much information that a player is just confused because there's so much data on the screen at once.

But being clear on this is the action that you're taking and this is the result, or here are all the things that the game currently cares about, is one way to help people better understand how your game is working, so opening up the hood a little bit. I don't mean giving them the source code. No one's going to read that. I mean putting things on screen or through audio to better reveal what's happening numerically in your game. Previous user experience, we focus-- yes?

AUDIENCE: What about the danger of you revealing that information, the player seeing how something works, and then them saying, I don't like how this works, I don't want to do this anymore?

PROFESSOR: Then I think you have a play-testing issue to consider. It's not like the player is confused about the situation. The question is-- that there are two possible answers. One, maybe our game system is badly tuned or badly designed in the first place. And yeah, once people realize how this game works, they're not going to want to play it anymore, which is a game design issue.

And then the other situation which is, well, if we never explained these numbers to them, they will actually better like our game. Even if they figure out how this game worked, by not showing them these numbers, they better like this game. I have that issue with *Destiny*, actually. I don't want to see those numbers.

And you can make that decision to hide it. It's like, this is complexity that we don't want players to care about. You guys have to be very, very careful, but that's why design is an interesting, tough problem. But those are the kinds of decisions you get to make as a designer. Other

questions?

We used this earlier when we looked at *Animal Upon Animal*, *Tier auf Tier*, And people immediately assume that this was a game for kids based on user experience, other kinds of games that you've seen-- the color of the font and the shape and how it's written. This is an illustration of how pull-to-refresh works on iOS. If you want to refresh all of the email in your mail client, you pull it down, and this little circle starts to stretch, and there's a little icon there. And then it starts refreshing.

That is not an interaction with a computer that anybody's born with. This is something that someone had to learn from somewhere. Anyone can think of when was the first time you saw this on a cellphone? On Instagram? OK. Any other earlier things? I think Twitter actually has the patent on this, but they don't seem to have sued anyone for using this, and Apple uses it now-- pull-to-refresh for your Twitter feed.

And now it's in many mobile apps. And mobile apps just assume that we don't have to teach anyone this. They just assume that if you are a regular user of mobile apps, pull-to-refresh is something that people will understand how to do.

Here's another example. This is early in *Half-Life 2*. And early in the game, you're just given a crowbar. And the dialogue even says, you know, assume you know what to do with this thing. Why can the game just give you a crowbar and not tell you how to use it? Why can *Half-Life 2* do that?

AUDIENCE: Because people know how to use a crowbar in real life.

PROFESSOR: OK. There is some real-life personal experience about how to use a crowbar. Although I would argue that how to use a crowbar in *Half-Life 2* is not actually how you use it in your life. In real life, you sort of lever it, right? In *Half-Life 2*, you kind of do that. Of course, having to use it in a game is using the mouse and keyboard. Why can the game get away with not explaining to you what this object is used for?

AUDIENCE: Well, it's a sequel, so you could assume that the person played the first game.

PROFESSOR: The assumption that they've played *Half-Life 1*. It's a game for fans. It's not that crucial a tool, although eventually they start to have puzzles where you need to solve it using the crowbar, but you can usually solve it using some other weapon as well. The crowbar happens to be an iconic weapon of the *Half-Life* series. And so they just assume, you're getting into *Half-Life 2*,

you probably realize that the crowbar is a thing in this series. So again, previous user experience.

There's cultural cues that come from outside of games. Like, this is a shot from *Grand Theft Auto*. And even though you may never have driven a car that looks exactly like that, you can make certain assumptions about this kind of car. What kind of car is that? It's kind of blurry, but maybe you can make it out. Does it go fast, or is it kind of a clunker?

AUDIENCE: Fast.

PROFESSOR: It's fast, because it's kind of a streamlined, sports car look.

So you know that you're in a fast car. *Grand Theft Auto* is a game. There's a lot of violence. If somebody shot at you while you're in this car, do you expect to be taking damage? Yeah. There's no windows. There's no steel frame.

If you wanted to get onto that bridge back there, how would you do it? Just go up the on ramp, right? I mean, you can kind of see it, but it actually curves off and then goes onto the bridge. But you kind of assume that players have seen real-life on ramps and sort of understand how urban geography works. In this case, it's particularly an American city. And they see a bridge in the background. And this thing that slopes upward, even though it curves away from the bridge, people can assume that it actually leads to the bridge. So they're using all of these cultural cues.

But you can take that idea too far. So this is a screenshot from the Magic Cap PDA user interface. So this was pre Apple Newton, I believe. And this was a user interface for a personal data assistant. And there are some things here that sort of make sense. What do you think the little postcard thing in the middle is supposed to be for?

AUDIENCE: Writing postcards?

PROFESSOR: Hm?

AUDIENCE: Writing postcards?

PROFESSOR: Writing postcards? This is a digital device. Do you think you're actually using it to write postcards?

AUDIENCE: Internet postcards.

PROFESSOR: Internet postcards, OK. What do you think this thing in the lower right corner does?

AUDIENCE: Recycle bin.

PROFESSOR: It's a recycle bin. It's a trash bin. You've seen it in a lot of different desktop-- this time it's a trash truck, I'm assuming because all of the other recycle bin and trash bin things have been copyrighted or trademarked by Apple and Microsoft, respectively. What does the magic lamp do?

AUDIENCE: Grants wishes.

PROFESSOR: Grants your wishes. Programs?

AUDIENCE: [INAUDIBLE].

PROFESSOR: Like, apps. Yeah, OK. Some other hands? Some other ideas?

AUDIENCE: Help.

PROFESSOR: You think help? Yeah, you rub the genie for help. So we have three fairly plausible entirely different interpretations of what that magic lamp is. This is an indirection metaphor. Everything on this desktop-- and we still use that terminology today. This is from the early '90s. Every single one of these things is a metaphor for something else that you can use on the PDA. You're not going to be writing postcards using a digital device because that's a medium that exists outside of digital technology. But you're going to be sending emails, so you're going to use that postcard icon in the middle to send messages.

Are you going to be sending faxes exactly? That's a fax machine. First of all, in 2014, I'm not entirely sure whether anyone recognizes a fax machine anymore. Are you going to be using that thing to make phone calls, or are you going to be sending pictures with it? The clock is not such a difficult thing to understand. So some of these interaction metaphors work, and some of them don't.

Again, if you put story in your game or even some sort of a fantasy setting or fictional setting, that's already establishing a metaphor for your game. If your game is about child grows up and learns to battle the evils of the world, for instance, that maps on pretty closely to the mechanics of a lot of games. That's why the *Zelda* franchise works kind of hand in hand, the

storyline and the mechanic. You start off as this kind of helpless kid, and then you get all of these tools and take on bigger and bigger challenges. It's a big metaphor for growing up.

But you can also confuse, or at the very least, date your game by over-relying on metaphors. So if you, say, have a big stop sign or something that, well, that's probably pretty timeless. But it's going to assume that everyone has seen a stop sign, and your users will be unclear whether that means stop the current action or stop forward velocity. Those are two different things. So I'm just saying, be careful of using too many metaphors to do the heavy lifting on getting your concepts across. Often you're going to have to rely on multiple means of feedback to help people understand how your game works.

Now, one thing that you're going to be keeping your eye out for when you're designing for usability errors-- mistakes that people are making, but I want to clarify there are two different kinds of errors, and mistakes are only one of them. Slips are skilled behavior errors. Someone who knows what they want to do and accidentally did the wrong thing. How many of you have gone to class when you had intended to go to a friend's dorm room, and you just ended up in your class? OK, yeah. Or driven to work or something while intending to drive to your friend's house?

So it's not like you didn't know where your friends were. You knew where your friends were, and you knew that you were going there. But somewhere along, your skill behavior of getting to class just automatically took over. And you did the wrong thing.

I have calling your boyfriend by your last boyfriend's name on my examples here. That's a tricky one. But things like clicking a button that looks very similar to the button that you actually wanted to click. This is, by the way, something that you find in *Dark Souls* and many, many different fantasy games, is you have a monster that happens to look like a treasure chest. And this was intentional.

So the game designer is using this kind of player error deliberately to sort of trick you into a powerless position. Ooh, a treasure chest. I'm going to open it. And it bites you. So it's going to mimic.

And those are things that might be OK to leave in your game. If it happens too often, it can be frustrating, especially if you're sort of designing your user interface in a way that leads people to confusion, and you want to be able to catch those errors and fix those. But a little bit harder to fix are mental model errors, where somebody actually thinks your game works differently

from how your game actually works, and they're going to make a mistake because of that.

So how many of you play *Portal*? All right. Up until you meet this thing, is there anything in the game that can actually hurt you? Oh, yeah. You can fall into water, that really, really icky-looking water. So that's the first time you die. But that's something that you fall into. That's something you kind of like-- yeah, you can kill yourself, but you have to kill yourself.

Then you meet these things, and you haven't met a single thing that is out to get you. And then you walk around the corner, and what does this thing do?

AUDIENCE: It gets you.

PROFESSOR: It gets you. It shoots you. It's a turret. It says some really, really cute things, and then it's like, are you still there? Bum-bum-bum-bum-bum. And then you suddenly realize in this game that there are things out there who are actively going to be trying to hurt you even if you're not moving, for instance.

So now, up until that point in the game, the player will be perfectly legitimately thinking, the only way that I'm going to lose or die in this game is to make a mistake myself or to deliberately cause myself to be in a position that's going to kill me. But then they find these things, and they realize, no, actually, there are hostile elements in this game that are out to get me. And now they have to revise that understanding of what this game is all about.

Now, this is the challenge. You as a designer, you are the people who are making the games. You are making these systems that players are going to interact with. And when players are playing the game, they're sort of having a little conversation with the system.

But your conversation with the system is kind of one way. You are making this system, and then the system isn't there to tell you, by the way, the players are doing this thing that clearly indicates that you don't understand what's going on. But the system isn't going to be able to give you that information. Actually, if you take our design class in the spring, it gets even worse, because what you're really in control of are the game mechanics. And those are going to result in emergent dynamics-- strategies, for instance, things that are optimal, things that are going to lead people in a downward spiral.

And that's going to create an experience for the player, the aesthetic which we will get a little bit into later in the class. They're just what the player actually experiences. So the player may

think your game is very threatening and dangerous and it's out to get them, when really to you, it's just a perfectly harmless logic puzzle, because this is what you have control over. But through all of that interaction, what you've got here is kind of a second or even third order problem, where the thing that you get, where you can actually create, is not the thing that the players are actually experiencing.

So that's why you've got to test. That's the only way that you are actually going to get information about what the players are actually experiencing. You can't intuit it just by looking at your design. Maybe with a lot of experience, you will eventually start to get some best practices, but you still have to do usability testing, even if you're extremely experienced.

So one way to do testing is to actually set goals for the player. Sometimes when you're doing this in the middle of play-testing, the players can come up with your own goals. But what I like to do is actually ask players, all right, I want you to create a new game with a new player and give it a name of whatever you want. Give them a specific goal, and see how they go about that process of actually accomplishing that.

And of course, while they're doing that, you observe them. You encourage them to talk aloud. You observe where they're moving their mouth, or if they're doing a tablet game, observing where the fingers are looking. Try to figure out what they're looking at. In big companies they can do things like eye tracking-- Riot Games talks a little bit about eye tracking-- to actually figure out what they're looking at.

But sometimes if they're talking aloud, they will say, well, I see all these buttons. And they all have icons on them. And this one looks like it might mean delete game, so I'm not going to click on that. This one looks like a disk drive, so I think it's a save game, so I'm not going to click on that. And then while they're saying all of these things, you start to understand how your players actually understand the things that you put into your game. And you can figure out why they're making these errors.

So you're looking for them to make these errors, and you want to record them down so that later on, when you go back to design and go back to implementation, you can fix them. Or you can give players the necessary feedback to avoid those kinds of errors. But it's important-- when they make an error, you have to try to identify what the player's reaction to that is. Say something happens that they didn't quite expect. Are they surprised, or are they confused? I thought I was creating a new game, but I ended up opening an old game. That's probably a

confusion situation.

I thought this was going to take out one enemy, but it took out five enemies. Pretty nice surprise-- players might like that. So even though it's an error, you might be able to just live with that, or even amplify that. Let's give these players these weapons that occasionally do critical damage and surprise the player by how bad-ass the players are.

Are they frustrated, or are they engaged? I chose *Flappy Birds* in particular as an example here because when people are playing *Flappy Birds*, they get very angry, right? But then they go back and play it again. Are they frustrated? Well, yes. But are they engaged? Yeah, they are. For some people, *Flappy Birds* is an extremely engaging game, and it's kind of a pity you can't get it anymore.

I also have two different versions of *Flappy Birds*. One is kind of an Arduino powered version, but it's really made out of paper and cardboard. And one is the actual digital version, so to remind you, you can do this without having to actually implement a digital version. You can do this with your paper prototype. You just have to create a paper prototype of a user interface instead of just game play. So you can do testing even before you've written a single line of code.

So I've talked a lot about constraints. Now, so far everything that I've been talking about has been about informing the player and then trying to catch those errors and then feed that back into your design of how you provide more information to the player. But you can also prevent certain problems. Constraints is one way, where you just make it impossible for them to do the wrong thing, or at least early on, when you had multiple choices, you sort of take some of those choices away and maybe later reintroduce those choices when you are fairly sure that you've understood what the consequences of those are.

You can do conformation, which is you take an action, and then the game asks you, are you really sure you want to do that? However, this is really sub-optimal. Any idea why? Why is it bad to get a little dialogue box asking you, are you sure you want to do this?

AUDIENCE: If you wanted to get to it, it might be really annoying.

PROFESSOR: OK. It gets annoying, yeah.

AUDIENCE: And it might make you doubt what you were doing in the first place.

PROFESSOR: OK. Are you really sure you want to close this web page? You really sure that you want to save this game. Well, maybe not, but maybe that was the right thing to do.

You can also get really automated on that. Players can sort of learn to just click yes every time that thing pops up, and then kit kind of defeats its purpose. If all you're doing is throwing up a dialogue box and they're going to say every minute or something like that, and the answer 90% percent of the time is yes, then players start to learn to just click yes every time they see the dialog box, whether or not they actually intended to click yes or not. And that's bad. You're training them to sort of defeat the confirmation process.

And then you get dialog boxes like this. I don't know what the buttons do. So this is Skype, obviously. This itself can be so confusing for players. The confirmation dialog itself is a piece of user interface that can go right and can go wrong. So you've just introduced a new element that can be misinterpreted in the design of the game. So confirmation is something that exists, is often used, and can backfire, and we need to be aware that it can backfire.

You can also do checks. If you've ever entered a code into Xbox Live or PlayStation network or the Apple App Store, the numbers that you're typing into the system have to fit some sort of checksum or error correction protocol. And you can prevent certain errors by doing checks on the data that you're entering. Now, this is probably very familiar for anyone who's done any kind of user interface, software engineering class.

But this is one from Microsoft Xbox Love. And if you type in a code that you might have gotten from a giveaway or a free coupon from a game developer, or maybe a card that you bought at the store, they're always 25 characters long. They're always separated by these dashes in chunks of five. I don't have to type those dashes in. I just start typing, and those dashes fill themselves in automatically.

If I do type the dash, the input box will just ignore the dash, and it will only allow me to enter in those characters. So that makes sure that the dashes are always exactly five characters apart from each other. And that's a visual way to provide a check while I'm entering this long string of characters. I can do this sort of visual check of, are my characters appearing in the right position in the string as they are printed on my card, and as it is being displayed on my screen.

You can do other kinds of checks-- like, if you try to do the same thing on the Apple App Store, I believe it could be numbers, it could be letters, or whatever, but you can hit OK as soon as you enter it in, and the system actually does a quick check before it even sends the code off to

Apple on whether that's actually a valid string, that fits a certain kind of checksum algorithm. Before it sends it off to Apple. And if it's a wrong one, it just tells you right away, no, you actually made an error. You can fix that now. So you don't have to wait for it to hit the server and come back.

But say the error does happen. The player has made a mistake, and it happens to be a mistake that you know about as a designer. And you know this mistake is going to come up often. This kind of error is going to happen. And maybe it's a slip, maybe it's a mistake. And you want to provide some feedback.

So first of all, make sure that you are informing the player that an error has happened. And you need to test whether the information that you're providing, whether it's audio or visual, some combination of the two, is actually something that people notice and see, and then they realize that, oh, I've made a mistake somewhere. And then you've got to get them some tools to be able to fix that problem.

There's backwards, which is basically undo. This is *Words With Friends*, for instance. If you enter in some sort of word-- this is an online game, by the way. It's a multi-player game that you play on your phone. And because it's *Words With Friends*, you're basically playing Scrabble against somebody who's geographically separated from you. And if you've taken a turn, you've entered in a word. And then you say, wait a minute, I've come up with a better word. You can hit Undo. I think it's called Recall. The button is actually called Recall. Take your move back and then just replay your turn before the other person has taken their turn.

But that's backwards error recovery, very good for things like turn-based games, games that you play by email for instance or games that you are sort of-- like *Final Fantasy Tactics*, or strategy games. A lot of strategy games which are turn-based allow you to sort of undo your decisions. Even if you've said, yes, this is what I want to do, wait a minute, this is not what I want to do. I want to undo that.

There's also forward error recovery. This is a screenshot of Mario where Mario's mid-jump, and based on player experience up to this point, you realize halfway through the jump that you're not going to be able to make that jump. What can Mario do?

You can go backwards. You can change direction in midair and say, oop, not what I'm going to do. It doesn't rewind time. You're still moving time in the game. And time is important in a

Mario game. But the player has the tool to say, I'm going to try to jump. No, I'm not going to complete this jump. I'm going to go backwards before it's too late.

There is a consequence to that, and the player is losing time. And in many of your games, you're going to want to allow those errors to happen but then give players tools to be able to take a new action to compensate for the mistake of the previous action. And that is called forward error recovery. Now, if it's a mistake that happens near the end of a game session-- say you're making a multi-player game and somebody's about to win. Forward error recovery may not be something that you want to make too easy, because somebody's spent a long time actually getting themselves in a position where they can win the game. And then the person who's losing uses whatever forward error recovery tools you've given them to say, ah, let me just take back all of those mistakes I did and then come back from behind.

Well, if you were explicitly designing a game where someone is supposed to be able to come back from behind, like *Mario Kart* does something like that, then that's great. But if you're trying to make some sort of game where long-term strategy is supposed to pay off, then yeah, you don't want to make those forward recovery tools too powerful. Let people live with their mistakes.

One last point, and I save this for last because I want you to be thinking about this in your team, so something for you to go into discussion about after this talk, is accessibility. One challenge is that when people think, my game is for everyone, my game is for anybody who plays games, is that that's not actually what the game ends up being. Often it becomes this, which is you're going to assume that your player is focused and has full use of their hands and full visual acuity, that they speak English so they'll understand any text that you give them, that they know how to install stuff in a computer and they've played games before, that they are around about your age, that they have desktop computers that have lots of processing power, especially if you're doing a 3D game, that their mice have more than one button, that they're actually using mice and not a trackpad, that they've got headphones so they're going to listen to every audio cut that you're going to give them without distraction.

So these particular red line things are the things that I want you to be willing to question about your own game. There are some things like-- they're speaking, computer-literate young adults. It's great to be able to design games for people who are not like you. It is a little bit beyond the scope of this particular class, although you do have to keep in mind that we are also designing this games, your final fourth project, for people who may be working for the Red Cross Red

Crescent Climate Centre.

So you want to keep them in mind and consider-- many of them actually speak English. They may be foreigners, people who work in other countries, who will appreciate a game that is a little bit more language agnostic or even localized for them. Shall we assume that the computers are capable of handling your game? OK, I think for this class that's all right. But shall we assume that the computers are only running your game and nothing else, that they're not running any IM clients-- or a lot of the games are going to be browser games so that they don't have any other browser windows open, for instance.

I think that might be asking too much of your player. You might want to assume that they have other things running around in the background-- their iTunes playlist, for instance-- and which is going to interfere with your sound. You should be able to design your game so that you don't assume everyone has a left mouse button and a right mouse button, or necessarily a mouse. For one thing, it'll make testing easier. A couple of you discovered last time you did a visual test, that if you don't have a mouse, a lot of your games actually became difficult to play. You might want to have that discussion-- at least have that discussion with your team, and if you decide, no, a mouse is essential, make it very, very clear that's what you're saying. And put it in your game. Say, this game is best played with a mouse and headphones. There's a lot of iOS games out there that require headphones, and they will state right upfront, this game is best played with headphones.

And I would like you to at least consider colorblindness. Actually, yeah, there we go. When you are designing art for your game, buttons that need to be distinguished from each under or visual cues, consider that there are people who can't tell the difference between certain color pairings. What are common ones? Red, green. Red, blue?

AUDIENCE: Blue, yellow.

PROFESSOR: Oh, blue, yellow. And red, blue. Red, blue. Blue, yellow. And red, green. Those are the most common ones, but there's a lot of colorblindness out there. And it affects, I think, 20% of the population. 10? 10 to 20%?

AUDIENCE: I think it's as much as 10% of the population.

PROFESSOR: 10% of the population. So there's a lot of people out there who might well want to be playing your game but can't because they can't tell apart those visual cues that you're giving them. So

make sure that they're also taking advantage of things like shape and brightness. These are things where if somebody is colorblind, they can at least still distinguish shape from each other. So instead of having red diamonds and blue diamonds, you have red diamonds and blue stars or something, and that makes it easier to tell that there's a difference.

I mentioned other application windows, touch pads, or the possibility that somebody might be playing your game muted, which means make sure that not all of the feedback that you're giving them is audio only. Make sure that you're also giving them some sort of visual feedback so that they understand that if they're playing in a cafe or something like that, they will still be able to understand that they're making mistakes, or errors or slips.

All right. That went way longer than expected, but I did warn you guys it was going to be long. Any questions before we wrap this one up? OK. All right. Well, now is a break time, I believe?

PROFESSOR: Yeah. Take a 10-minute break. At 2:47, come back here. If you want to play-test a paper prototype or have any kind of testing in your game, let us know. Otherwise, working in teams.

PROFESSOR: Yeah. A lot of you have mentioned how hard it is to schedule a meeting. You've got the rest of this class to be able to meet with your team. And either test something, or you can just work on your game. If anyone wants to look at these books or this game, come up to the front, have a look. Thank you.

[SIDE CONVERSATION]