PROFESSOR: All right, so it looks like we've got everybody here. It's about 1:11. Today in class, we are going to end our project twos. Yay. We feeling good about this? Kinda? I'm feeling good about this. And we're going to start project three.

So we're going to start off with our project two presentations. Again, we're going to go through here. Each team has five minutes to give their presentation. Slides. You can set up your slides up here. We've carved out enough time for today to be able to swap out between computers and for us to ask a couple of questions after each presentation.

After all the presentations are done, we're going to take a break. Instructors will kind of talk to each other about what we saw in the presentations, and we'll address some of the feedback that we heard, some of the things that we saw in the presentations. As a reminder, presentations, we want to hear about what went right, what went wrong, and also what would you do differently?

Then we're going to start project three. And we're going to do brainstorming in class. Basically, the first week of class we're going to compress into about 30 minutes of this class. We're going to brainstorm, we're going to do our elevator pitches, and then we're going to form teams. So we're going to do that really, really quick. And it might require rolling the die again. It might require moving you from project to project. And that's going to be OK. We'll talk about why we're doing that when we go.

So I think that's everything from me. Any questions? And if not, Sparkly Redemption, come on down.

PRESENTER 1: So what went right? The game got made. It isn't broken or sad or dead. Even though we had to cut some things, I feel like the general idea of you are a player and you pick up sparklies and your sparklies change monster behavior slash your power-up slash whatever got implemented pretty well in a way that carried across our original idea.

What went wrong? I guess we had to drop some things. We had a person on our team not get

hit by a bus, but they did basically not have as much time to work on the project as much as we thought. So we had to adapt to that. On the other hand, I feel like we adapted to that pretty well. So not good, but still good, I think, personally.

We were told to test early and test often, which I think we actually kind of did-- maybe not as early or as often as we could have. But we still tested even when we had something that we felt wasn't really testable but was still touchable. And I feel like that was kind of interesting to do because people kept telling us, you should totally be able to die. And we were like, yes, we do think you should be able to die, but we haven't implemented that. And so on and so forth.

But on the other hand, that meant we still got lots and lots of good feedback and things to think about, like stuff that we probably wouldn't have thought to have implemented but still implemented because we caught that early on. So yes, test early and test often, even if it's annoying, is what I learned, and I feel we all learned. And that might be all I can think of.

PROFESSOR: What would you do differently next time?

PRESENTER 1: What would I do differently next time? Meet up with people. Well, not necessarily physically, because we scheduled meetings but then we decided maybe we don't need to physically meet up because it's a pain getting people onto campus if they don't live on campus, which would have been fine. But then there were still times we were like, well, we meant to meet up but then we got iffy about scheduling a time, and everyone got confused.

And I feel like we managed to work our way through it. But it was still more of a hassle than it probably needed to be if we just sat down and scheduled out a chunk of time and listed things that we were going to get done. So differently would be better about meeting with each other, either physically or through the interwebs.

PROFESSOR: Any questions from anybody? OK. Which game engine was that again?

PRESENTER 1: Sparkly Redemption. It's the one-armed lady.

PROFESSOR: What game engine did you use?

PRESENTER 1: Oh. Phaser.

AUDIENCE: Anything to say about the experience of working with Phaser?

PRESENTER 1: So you do the thing where you have a bug and you think it's Phaser related, so you Google

vague terms. And I've worked with Flixel and Unity, so I know personally I'm used to having long form posts pop up that you then dig through. But I feel like with Phaser it was a bit of a mixed bag. And sometimes you got relevant posts. Or sometimes you just went to the Phaser examples and kept digging through it until you found something vaguely relevant. So it wasn't terrible, but it was still not quite what I was used to in terms of figuring things out.

PRESENTER 2: So we're Lost Underground. One of the biggest things I felt our team had a problem with was communication. Because at the very beginning, we were really good. We all got really pumped for this project, and we got together. We had the paper prototype. We edited it a bunch. We quickly got that over to a digital prototype.

And then it kind of stagnated. We had the play testing, and we learned a lot from that. One of the most interesting things is we added the darkness mechanic, where you can only see a certain zone around you. And some people felt like you shouldn't be able to see as much. People felt you should be able to see more or less. Everyone had different opinions.

But we liked the way it worked. But it didn't work the way we wanted it to. People weren't concerned about walking around at all. The whole point of the dark is to be afraid of it, like oh, is there a monster over there, but there could be something to collect, and people weren't feeling that.

And so we found out from the play testing we needed to change it. And the best way to do this was to shrink it so you could see even less, to zoom out the camera so you feel really claustrophobic and far away from everything. And I think that really helped us.

We also learned-- what was really funny is we had just a placeholder, a little coffee cup, instead of a coin, which is what we originally went for. But every play tester was like, oh, coffee. That's the coolest thing. I love the idea of collecting coffee.

And so one of the cool things about play testing is if you don't say anything, they will give you really useful information, even though that's not what you intended at all, which was really cool to experience firsthand. And so we decided to stick with the coffee mechanic, and it ended up being a little more silly than what we originally planned for.

But like I said, communication was really hard. Planning out who was going to do what with a digital prototype was much more difficult than with a paper prototype, I personally feel. Because code intermingles in a way that paper doesn't. And so it becomes really difficult when

people are pushing things and pulling things and then things break.

And so that was really hard. A way to change this, I feel like, would be to just make sure everyone replies to their emails, and make sure people are talking and everyone knows what's going on. And I don't know the best way to do this. Maybe an IRC chat, where they constantly have that chat channel there and people can look at it when they want to. Because emails kind of get lost. And text messages people forget to reply to. So something like that, where it's constantly there and you can go look at it and hopefully reply to it in a timely manner.

Or maybe just have-- one of the things I did in my other classes is the very beginning, everyone had to write up a contract of when you'd be working on things, when you reply to things. And so we had to sign and say, I will reply to any emails about this at 9:00 PM every day. And so you can expect a reply by my by 10:00 PM.

And so because we had this contract, we knew-- if I send an email to this person before 9:00 PM, I will get a reply an hour later, which was really useful when people were a little hectic and they had other things to do. At least I knew I would get that email back. So any questions?

PROFESSOR: So same question as before. What game engine did you use?

PRESENTER 2: So we used Phaser. And any coding questions, I'm going to direct to another team member who was more heavily invested in it than I was.

PROFESSOR: Anybody want to speak up about coding?

AUDIENCE: [INAUDIBLE] curious about Phaser.

PRESENTER 3: I thought Phaser was an interesting engine to use. I think in terms of the organization of the code, you have to do a lot of thinking beforehand if you want to make it work well. I guess it could be a little finicky when you just start building, because we thought, OK, well, we'll add code, and then we'll sort of architect it based on what we see happening. But then we came to realize that our code wasn't going to be as separable as we thought it might be.

I also ran into a couple of bugs that were in Phaser itself that I had to sort of dance around. So I would say watch out for the fact that there are still some bugs in it.

PRESENTER 4: Yeah, just callback nastiness and having to think about something will happen now-ish. Time isn't really now-now. It's like, now kind of now, then kind of then, floating point numbers that

are somehow always floats and nothing is ever actually equal to anything else.

PROFESSOR: Is any of that documented?

PRESENTER 4: This is just found out. That's JavaScript. But I thought we could get away from JavaScript by wrapping that all in an engine, but no.

PROFESSOR: Thank you.

PRESENTER 5: Hello. We were Plunder Winds. So starting off with the good. I think our team was actually pretty well organized, and our roles were well-defined. So from the very beginning, we set up task lists and lines of communication. We set up a mailing list and then assigned everybody specific roles.

And then from that, we were able to-- none of us duplicated work. And we set up a good architecture early on. And so we used Phaser. And unfortunately, there's this one part of Phaser that's not really well documented in any of the examples and tutorials which is the state machine that Phaser has. Surprise, there's a state machine in Phaser.

And then once we discovered that-- we discovered that early on and used it. And so that allowed us to really architect our code well and separate out modules. And we also used a JavaScript module thing, which basically made it so that all of our coding efforts were really-- were really efficient, I think.

And as for the bad, five out of six, or basically our entire team, came down with the flu the first week, or over the first weekend. And that really impacted our schedule. So originally we had wanted to get a playable prototype by the first weekend so we could get people touching it and start getting feedback. But the plans best laid, it doesn't help if everyone gets sick.

So we had a large schedule impact. And the way we had to deal with that was, like, there's one guy who was able to code throughout the entire first week. And then the rest of us were playing catch up. And so we didn't get anything playable until the in-class workshop on Wednesday. And I think that really hurt us because we weren't able to get a lot of important feedback until very late in the process.

And so there were some features that we kind of added. And we were like, this may address player concerns. But we just didn't get enough feedback or data to figure out whether or not those concerns were actually really addressed.

PROFESSOR: And anything you'd do differently?

PRESENTER 5: I think we would not get sick. But also, we set up Asana and everything on Monday. But then we learned, like, oh, we have to do a sprint task list and a product backlog on Wednesday. So I think there was a bit of a weird issue there, where we already had one task management system, and then we introduced another one. And then we introduced scrum boards and we were like, well, which one do we use? And then we just didn't end up using any of them.

PROFESSOR: Any questions? So my question, I think, is you had the flu. A bunch of people were out. You didn't cut features. You tried to just catch up. Did you think about cutting features? Do you feel like you ended up cutting anything at all?

PRESENTER 5: So I think our core game was actually, like-- it was pretty simple to implement. And I just want to give a shout-out to our team. You guys pulled off an awesome job and implemented everything by Wednesday. And so we did actually manage to get a feature complete game by Wednesday. And then we started adding tweaks, like interface tweaks and maybe a visual cue here and there throughout the rest of the week. But in terms of cutting features, we didn't really have any complex time-intensive features that we would have benefited from cutting, I don't think.

PROFESSOR: Thank you.

PRESENTER 6: So our project was Beaver Evolution. And so I guess I'll first go through three of the bigger design challenges we came up with or that we encountered during these two weeks. And our [? system ?] was working with Phaser, which is kind of design, kind of the project challenge.

And so one thing we ran into is we didn't realize there was this thing called stages, and I think you guys just mentioned that. But since everything is turn-based, and in each turn you can make a choice between building, populating, and evolving, and then disaster might strike. So there was different things that happen. So we thought it could be very modular and we could use stages.

But then we learned that stages really start and stop every time you switch stages. So it doesn't take the info about what happened in a previous stage, which was really important to our game.

So then we ended up having to switch everything over to Phaser groups, so we could hide

groups and show groups. So that was one of our first challenges, just kind of figuring out how Phaser worked and what would be best for our code.

It turned out our code was very modular because there were so many different elements to each turn. So it was very easy to split up the work. And that was helpful. Overall, we really liked Phaser.

Our second challenge is balancing the game, so making it not too easy to win and not too hard to win. And we kind of changed that by changing some of the winning conditions and then also adding a few mechanics here and there to make the game harder. Because often it was just too easy.

So for example, you could just keep evolving. Or you could keep populating your beavers. And then once you reached the number of beavers you needed, you'd win. But we kind of changed that by implementing a rule where you can't populate every turn because you have to [INAUDIBLE]. So I guess the slide is not really working.

PRESENTER 7: Yeah. So I'm not sure why the screen won't show our slides, but we can keep talking about the game. So Rachel talked a lot about the design challenges we ran into for the game. And so I'll talk about some of the project challenges that we ran into.

I was assigned the role of the product manager. So you remember the product manager in the scrum roles is the person who's supposed to sort of lead where the game is going and make sure that all the design implementations are the way that they're supposed to be and handle any changes that we need to make to the code.

So unfortunately, we didn't explicitly assign a scrum master. So by taking on the role of being the product manager, I also assumed some of the roles of the scrum master. And I know that this was stated to us in the lecture about scrum, how we should definitely not do that. And we now see why.

I unfortunately had to run some of the meeting-- I unfortunately had to focus on running the meeting as well as on making sure that the design was going in the direction it was supposed to go. So I definitely lost some efficiency there. And we started off very efficient. We were able to assign tasks, and our product backlog, our sprint task lists, were all very laid out. We had an early meeting to make sure that everything was going well.

But then by the end, I just sort of started running out of time. We were unable to schedule a

meeting after the play testing session due to everybody's schedules being very tight. And so some of the changes that we wanted to implement after the feedback from the testing session we were unable to implement. But we were still able to get a good game together. Our organization just sort of fell apart towards the end.

So those were probably some of the biggest design problems we ran into. I would definitely next time, for our next project, make sure that there's a clear product manager and a clear scrum master so that those two roles can be done very efficiently and not overlap.

PRESENTER 6: Yeah. And I'll just add on, so our testing we really realized that user experience could be improved between having all the players know all the game mechanics and rules coming into the game. So yeah, in addition to not being able to meet up, not being able to play all the features, we really just focused on improving game experience by adding audio and by adding smaller things. But I think earlier in the game, we should have thought more about user experience and making sure that was a good flow.

PROFESSOR: Are there any questions? I think you covered everything. Thank you.

PRESENTER 8: All right. We're Comcastic. So as you'll recall, our game involves placing service centers on a map and trying to cover all of the houses. So what went wrong-- setting up Phaser took a long time. It was really hard for us to actually start writing code. We were using TypeScript, which was important, I think, for us, especially as a documentation. But it took some time to get a good set-up where you had a posted Phaser where everybody could develop and you had TypeScript integrated. And it would be nice if-- future projects, I think, will be set up for that. But for the first project, that was hard.

Meetings-- it was really hard to schedule meetings. We had seven people when we started, and we went down to six. And even at six, it was still hard to schedule meetings, mostly because of PSETs. I had one PSET, but everybody else had a lot of PSETs, and it was a lot of conflicting times.

What went right-- so Phaser itself, I think, really helped us develop fast. Once we got the hang of Phaser, it was a lot easier to develop. It was pretty consistent. The code is pretty good. It was not hard to debug by just looking at Phaser code for us.

Also, I think we got the synchronous communication down, because we really didn't meet that often, especially as a whole group. So we were using Trello, and it was really nice to just sort

of bounce an idea off Trello, just put in [INAUDIBLE] feature, and then we would sort of discuss it via Trello. I think that that's something that we want to keep doing.

Also balance-- I think our balance is actually pretty good. It was sort of a coincidence that our pieces-- so the components of our balance are basically how the map looks, where the houses are, and then also the service centers you can place are. And you can imagine it'd be kind of difficult to make these two match up, but I think our first random attempts ended up working pretty well.

What we would do differently-- we would definitely stick to one communication tool and stick to something simple. I think Trello was fine, [? but even ?] GitHub issues would be good, and they would integrate with the code. Definitely we want to keep scheduling partial meetings with some of the group, just the programmers, just a handful of people, whoever can make it to one meeting. I think with students that ends up being the best thing to do.

And getting a [INAUDIBLE] viable product out early on-- I think we underestimated how important that was, even just getting a Phaser set up where you have an empty game running quickly. Even that, if you get that done in two hours versus two days, it makes a huge difference.

PROFESSOR: Any questions from anybody?

AUDIENCE: [INAUDIBLE].

PRESENTER 8: Yes, face-to-face meetings. Even Google Hangouts actually weren't easy to get everybody at one time.

PROFESSOR: All right, thank you.

PRESENTER 9: Hi. We're Modudice. That is our game. For those of you that played it, it's sort of like a number-puzzle type game. You move around a dice. You add numbers, a module of seven. And you want to make as many sevens on the board as you can, and you get points. And so what went well for us is our design meetings went quite well. We had two or three of these that were basically dedicated to figuring out, how do we do this better? And how do we answer some of the questions that came up during testing?

And I feel like those went really well. We were able to get good discussion. We actually came up with, like, solid answers. And we were convinced, like, OK. This may not be the best way to

go, but it's definitely comparably good to anything else. So we're going to just stick with this.

And we were able to make task lists as a result of that. So we decide on something and then say, OK, what do we need to do to actually do this? And I think that went across fairly well.

And we did a lot of good testing, especially the testing we did in class. And on top of that, like, all of us just tested the game with our friends, since it's, like, fairly easy to play and sort of fun. So yeah, the testing was really useful. And we actually did change a lot as a cause of it.

PRESENTER 10: I'd like to add to the task list that I think we also did a very good job in terms of prioritizing what tasks were important. I'd like to say that we cut two features-- the dice rolling animation and sound. And I think those were really good feature cuts because they weren't central to our game-play as a very fast-paced browser game. And cutting them actually saved us a lot of work and let us have the time to spend on other features and stuff.

PRESENTER 9: And Phaser was good in the sense that it was something that went well, because it was easy to start with and it was fairly easy to use. Like, once you knew what you needed to do with Phaser, it was pretty simple to get it going.

But it was also a bad thing because it's Java. At least we did it in JavaScript. So JavaScript is hard to work with, hard to fix bugs and stuff. And I could see-- like, Phaser was fine because it was a fairly small project. I could see Phaser being absolutely horrible for large projects. Like, I would never want to use it ever.

PRESENTER 11: I actually thought that it actually did not have a lot of good documentation when we were trying things. I thought the community was not the best for Phaser.

PRESENTER 9: And so Git, I think, is awesome. But a lot of people weren't familiar with Git. And so the branching was confusing. And that became an issue. So if you're going to use a [? version control ?] like Git, make sure everyone understands it. That's something we learned.

And getting tasks done separately-- we thought that would be most efficient, since we're everywhere across campus, it's hard to meet up and stuff. But I think it's actually better to just at least start working together. When you have a set of tasks, begin the work together, do as much as you can, and make sure everyone's clear on how to do what they need to do and what exactly they need to do so that everyone feels like they are actually contributing and working together on this.

PRESENTER 12: I want to add one more point on Phaser, in the sense that-- I feel like Phaser-- we're trash-talking it right now, but I think Phaser is good for a one-person project because it's very easy to manage the stuff on your own. But with larger groups, I just don't like reading through JavaScript spaghetti code. So maybe we needed to find a better way to organize our code with more people. But I just didn't like-- like, I edited a piece of code, someone else edited it, and I was like, wow, it's totally different from what I expected it to be.

PRESENTER 9: And so yeah, next time it would definitely help to have scrum meeting and work physically together, so basically just spend more time together as opposed to messaging on email and stuff. Make sure everyone understands the source control and the engine. And like, have more people really think about-- like, have everyone on the team really think about design decisions. And then communicate, communicate. More communication definitely would hurt.

PRESENTER 10: I would like to add to that in terms of the communication. Because as a new member of a team who joined the Modudice, not the original Modudice team, we didn't actually host an information session about what were the critical things about the game, that we made it the way we made it. So the new members, such as me, had difficulty to put in variable input, at least for the starting duration of the project, and then we picked up later on. But if we had an information session, something like that, it would be very helpful.

PROFESSOR: Any questions from anybody?

AUDIENCE: Well, [INAUDIBLE]. When you said that your design meetings-- this was all the way on your first slide, were those online? Or were those in person?

PRESENTER 9: It was in person. But those were mostly focused, like-- so we had a lot of questions of how to represent the dice in 2D and stuff like that. So the design meetings were mostly focused on big questions like that.

PRESENTER 11: In addition, I think we had [INAUDIBLE] meetings. And I thought that we should just have multiple smaller ones, [INAUDIBLE].

PRESENTER 9: It ended up going, like, two hours.

PROFESSOR: Thank you.

PRESENTER 13: So what was bad? We're going to start with the bad things. The first decision we made as a team was using Unity 2D, because that was just the majority vote. Most people had worked

with Unity for the project before. But it ended up having too many complications.

And it was actually a good thing that we switched to Phaser very early on. Like, two days into the project we switched to Phaser. So Unity [INAUDIBLE].

Second thing, we weren't paying too much attention to the assignments. Everyone was focusing on the game development and making the game look good. And we weren't updating the product backlog, and a change log [INAUDIBLE]. So we had to go back and remember what we worked on in the past week. And it was very asynchronized with that.

Another thing is we put off a lot of the work until the day before they were due. Like, the last project we had a lot of things that could have been done during the week, but the night before, we always have hundreds of emails to sending the emails around, trying to get everything to actually work. And that was a thing that cut off on our productivity.

Another thing is that we didn't really have a lot of meetings. We only had two meetings throughout the entire project, which had a downside that we weren't very synchronized. Even in some of those meetings, some of us couldn't make it. So we had to update whoever wasn't there with what happened in the meetings and things like that. And we definitely could have used more meetings to keep our work synchronized.

The last part is CoffeeScript. CoffeeScript was a good and a bad thing. I think it was a good decision overall, but it definitely had bad parts to it, which is that we had to sort of-- there's a hack-ish with Phaser that had to compile CoffeeScript to JavaScript, which caused some problems with the [? gulping, ?] where it would sometimes not work. And we had to run it twice. It generated JS files which we had to not commit into the source control because they were changed when the CoffeeScript files would change. And so there was some problems with it.

But once we had it up and running, it was so easy to write code and just iterate on the code. It was also easy to read other people's code, because it simplified 100 lines of JavaScript into two lines of CoffeeScript. I'm exaggerating. And then finally, also, it optimized the JS when compiled, so it would run faster.

PRESENTER 14: So talking about now some things that went well in addition to finding CoffeeScript, we had really great group direction from the get-go. Even though we didn't have a lot of meetings, I think one way we still managed to make that work was we divided bigger tasks into the sub-

categories really quickly. And we also set up a Google folder in which we kept everything.

We didn't end up using Trello or anything like that because we really found that a Google folder with everything in it was the simplest for us-- that along with just one chain email where we sent everything really kept everything focused and everybody in communication, in the loop, as best as possible.

So source control, we used Git and GitHub. And we actually had a great experience with that. I'm sorry some people didn't. Nothing ever broke, and we barely had to merge because I don't think we did very much branching, which is the way I think we avoided any problems there.

So iteration-- we iterated our game a lot. We changed a lot of things, because we'd play it and be like, well, people keep dying, so how can we make that better? Another example is with our sprites, as you can see up there-- so the initial designs and then what we changed them to. We actually had a problem where somebody looked at our first alien design and said, that looks like a piece of the female anatomy. That is distracting. You need to change it. And we said, OK, you might have a valid point there. Even if we don't think so, maybe it does. Let's change it. So we iterated a lot.

And then focus testing-- we really actually implemented-- talking about iteration, in our focus testing, the two things we found were that new players were dying really fast. It took them, like, five times before they realized how to play the game. And then, in addition to that, after five minutes, they were getting bored.

So we actually took that to heart, and we changed one of the main premises of our game, which was this reloading thing, that it would take a couple of seconds before you could shoot again. And instead, we added a short term goal of actually picking up ammo. So you can see-- it's kind of hard to see. But the ammo-- the big thing is just the animation. But if you can see on the screen, there's a player, there's the ammo, and there's a bug.

And basically we changed that whole way the player actually shot to make the game a little more difficult over time but also starting easier, by having everybody smaller and then the bugs get bigger. So a lot of changes and iteration even based on our focus testing.

What we would change-- we'd start earlier. I know we talked about a lot of changes we made were the night before things were due. Pretty much all of our documents changed the night before they were due. And it's kind of frustrating, you know, having to stay up really late the

night before it's due when you've had the whole week to do it. So just start earlier. I mean, we started pretty early, but finish earlier, definitely. Hands off for the last 12 hours, kind of thing.

And then we also talked about this-- being focused on what's actually due. So one thing that was good about having a lot of focus on our game was that we had a viable product really early. But a lot of our other documents kind of fell by the wayside, and they kind of came up to one or two people to just fix them all the night before they were due. So definitely focus on those things and more accountability with that.

And in conclusion, even though we had some frustrations, it went pretty well. Our game ended up pretty clean and creep and fun. So we really like it. Thanks, guys.

[APPLAUSE]

PROFESSOR: Questions?

AUDIENCE: How did you stumble upon CoffeeScript? What was the decision when you were trying to [INAUDIBLE]?

PRESENTER 13: Jen, you want to take this one?

AUDIENCE: If you could come down [INAUDIBLE].

PRESENTER 15: I used CoffeeScript a lot before I was really familiar with the build systems and stuff. So I figured it'd be best to kind of set up a build pipeline that optimizes the [? alpha ?] JavaScript to make the game run faster because [INAUDIBLE].

PRESENTER 13: And CoffeeScript also has a lot of shortcuts. It's like Python. It's like using Python to write Java. It's so good. Like that.

PRESENTER 14: Any other questions?

PROFESSOR: Thanks. OK, feedback. And instructors, please feel free to jump in if I'm missing anything. Oh, and I forgot to copy all my notes. All right. So yeah, I was wrong. Ha, you caught me.

Slides were intended to be required for this project, but they were not. So you're not going to get penalized for that. The issue for that is slides and visuals are recommended to help your presentations stay on topic. Basically, you had some issues sometimes with keeping track of changes. We had some issues with keeping track of our changes to our documents. So QA is

important in everything that we do in life.

But for project three, slides and visuals are recommended to help your presentations stay on topic. And what we saw presentation-wise, slide-wise, was perfect for these five minute presentations. Quick, dirty, gets the point across, gets the bullets across. Visuals, like images or something, if you need it. For project four, that's a 20-minute presentation. Visuals are more important there. And we'll talk about those presentation requirements later when we start project four.

So discussion on meetings, things you should do in project three-- address how you're going to meet in your initial design meeting. The most important thing you have, before you even start designing your game, is to talk about who can meet when, what is your schedule. You all have syllabuses from your other classes. When are your PSETs due, when are your recitations, all that stuff.

Put it in your high-level design document. It's actually one thing that we use often in that one section where we talk about team roles, is team meeting times. And I think I heard this team mention that, that you've used in previous classes. That's a very good tip. Use that. If others can't make it to your meetings, is there a way you can take video? Is there a way you can archive what happened in the meeting for those people who couldn't come?

Here's something that I've seen some local companies do-- drop in IRC. Basically, even something as simple as keeping Skype open. Even better-- actually, my notes weren't put in there. Even better, an IRC solution that has some kind of archival process in, so all the previous chat can be saved. I know there are some solutions out there that actually hook this into your GitHub or Bitbucket or what you're using.

Those solutions are out there. Try one of them. But again, even something as simple as having Skype open and online, you can always tell if one of your team members is online. You can throw a text and say, hey, can I get your feedback on this? Can you take a look at this code? For your daily scrum, 15 minute Hangouts probably is-- all you need is 15 minutes for doing that kind of daily scrum meeting, so trying that out.

Problems we saw with backfilling your product management. Integrate your project management from the beginning. We know that this was a problem for this project because we introduced it to you midway through. That's OK. You actually did-- we mentioned this, you did some pretty good use of product management.

The goal for any project management tool that we're teaching in this class is that it should be helpful. It shouldn't be a hindrance. That said, we'd like you to get some practice with it first before you start throwing it out. So for project three, please use the project management tools that we described as we've described them to you. In particular, product backlogs, sprint task lists, and the design change log, and the vision statements, those high level design documents. Scrum boards, mostly optional, but we found them useful. They might not be useful for how your team is distributed. It's up to you to figure that out.

For project four, we'll give you the ability to create the system that works for your team, so long as you give us what information we're asking for. So you know you need this information. You know you need a way to manage your tasks. You know you need a way to manage your features and to prioritize your features to estimate. But how you actually do that and how you actually save it, that can be up to you for project four. We will want to see it actually cause some improvements. And if it doesn't cause improvements, let us know why and what kind of problems you found with that.

And again, the design change log. This is a design diary. It's even better. It's your meeting minutes. You should be able to fill it out in five minutes. Along with your task list and backlog, it kind of shows the history of the project. Take the time just to spend those five minutes at the end of each meeting to just throw something quick down so you know what has changed in the past.

Phillip, you want to chime in on this one? Modular code doesn't mean your tasks are actually separate.

PHILLIP: Yeah. This is just something that I got a sense of from watching everyone's presentations today. A lot of folks, even the folks who mentioned that the code was well separated and people could code independently, it doesn't mean that there are no benefits to actually still working side by side, even if you're working on different parts of the code. And some teams mentioned that.

So actually, again, being on Google Hangout or Skype or IRC or whatever, if you're working in your own rooms, or even better, a lot of people mentioned scheduling those face-to-face meetings-- it's OK for you to have a face-to-face meeting, even if you're not planning on working on the same batch of code. And you can actually still save a lot of communication

time-- if nothing else, just to tell people what you're doing. So maybe it's not speeding you up, but it might speed everybody else up, and suddenly it helps the communication.

PROFESSOR: Form strike teams. You've got six people, maybe seven people, and eight people on project four. Not everyone needs to be touching code. There are other tasks that need to get done. You can form up into smaller teams. One team might just be working on some kind of-- again, if it's modular, it's going to have some dependencies. But if you can modularize it out, great.

But even better, especially with project four, when you have longer time, one team doing paper modification, doing UI elements, running focus tests, while another team is writing code. Think about how you can better utilize all the personal resources and all the man hours you have without requiring all these clashes with so many people touching the code. It's that too many cooks in the kitchen kind of thing-- not everybody needs to be doing that, need to be touching that.

Here's another thing we noticed. A lot of what you asked for was basically more time-- just various different ways of saying you wished you had more time, you wished you had started earlier, you wish you had iterated earlier, the flu hit. A lot of what you're describing there are things you're not going to have any control over ever. You never have control over that stuff.

What you need to do is focus on those things that you can control. So if you have illnesses, if you have major technical problems, the game is whatever you decide the game is. You've actually got some freedom in this class in that you've told us what the game is, and we're actually even telling you in the last week, again, tell us, what was this game actually supposed to be, so that when things change midway through the project, be flexible. Change what you're delivering to us.

Make it so that whatever you're delivering to us runs, works, is playable. This isn't a design class. We're trying to teach you some design skills. But it's not focused on design. So if the final game isn't fun, it's sad, but it's OK, so long as it runs, so long as it's playable, so long as it fits all the requirements that we're asking of you from the beginning. I know it's weird. This is the only class we do this in. Trust me.

All right. So that's it. Any other questions about project two? I think you all did really, really well. Really happy to see how the presentations went out. Yes?

AUDIENCE: Is there any way we can see what everyone did?

PROFESSOR: What's that?

AUDIENCE: Is there any way we can see what everyone did?

PROFESSOR: Oh, yeah. So one thing I highly suggest, somebody on your team email the video game mailing list. Video games, yeah. I did the plurals wrong. I never do that right with mailing lists. videogames@mit.edu-- email out a link to your game to your classmates if you want people to play it and give you feedback on it. At the end of every project do that.

Actually, use that mailing list as a resource for you to talk to each other across teams. Hey, here's our game. We need a little bit of feedback on this. My team is all sick. Can somebody please play this and tell me what's wrong? Can you play this on a different browser and tell me if it's broken? Take advantage of that mailing list. Any other questions? Yeah.

AUDIENCE: So [INAUDIBLE] postmortem, will we get more feedback on project one and project two, what the group grade was?

PROFESSOR: So there are no group grades. So what you're getting is-- your grades, if you look at the grading rubric, 20% of your grade is individual. The other 80% percent is your group. So the grade that you get is basically adjusted based on the quality of the write-up.

AUDIENCE: OK. And on project two, will we get more feedback on everything we've done, so we can better plan for project four?

PROFESSOR: Yeah. So for project two, you're going to get feedback on your design change log again. You're going to get feedback on your task list and product backlog. We just gave you feedback on the presentations and on those skills. You're going to do that again for project three. You get your feedback on the individual write-up.

Is there anything I'm missing there? And then game functionality-- we're going to give you more feedback on game functionality, because last time we did that in class. This time we're actually going to play your digitally released games. We'll give you some design feedback there for product three.

PHILLIP: Yeah. Because we could just show the actual numerical grading proclamations that we do, but that's meaningless without actually explaining what it is. So what we'll try to do is focus more on actual text feedback on what we thought were the strengths or the weaknesses of whatever you've given us.

PROFESSOR: That answer? Cool. Any other questions on grading? All right, give yourselves a round of applause for finalizing project two.

[APPLAUSE]

Let me double check to make sure I didn't miss a break. No, we just took a break. Ha, ha. All right.

So project three, you're going to practice project management, meaning you're going to use those tools that we talked to you about in the middle of project two from the get-go, with a slightly larger team. Some of your teams were five people this time, minimum is going to be six. We're going to make sure we leave class today with at least six people on each team.

And again, that's just to get used to it. Project eight, your minimum is going to be eight people. So with a larger team, you're going to focus on design iteration to maximize usability using user feedback through independent user testing. So the keywords there-- users, feedback, usability. Those are things we're going to be looking at when we're grading the functionality of your games.

So for project management, we mean create and use a product backlog and a sprint task list. There's few turn-ins for that. For design iteration, you're going to conduct focus testing and user testing on your own. And I'll touch on this again, but basically you're going to turn in two focus test reports, one of which can happen in class on a scheduled day. The other one must happen outside of class with a group of people who are not in class. So get people to test your game and report back to us on that.

And again, maximize usability. All the iteration that we see you do on your games, you should be iterating on that user interface or that user experience. So we are going to ask you to make a little bit more complex game than before. But really what we're trying to get you to do is make a complex game that has a possibly problematic user interface. Give us an interface that's good. Give us a user experience that matches the user experience that you want to strive for when you're designing the game.

PHILLIP: Just a quick thing. We do have a couple of lectures coming up on designing for that. But as Greg mentioned earlier, this is not actually the design class. There is actually a different design class that some of you have taken and are going to be distributed among the teams.

Or you might have taken, say, Rob Miller's user interface class, which also has a lot of those same sort of concepts.

If you've taken any of those classes, you should be identifying yourself to your teams, because you probably have actually practiced designing for usability a lot more than a lot of the other people in your group. And that's what we're going to be looking for. But most importantly, the testing that we've been talking about, we are expecting you to be able to do that. And that anyone of you have already got the basic foundation of.

PROFESSOR: So our design constraint-- we're leveling up from planning for randomness. We want to see trade-offs in decision making. So every decision made by the player must have a positive and negative outcome in their play state. So first off, what does that mean to you? What do you think that means? Jenny?

AUDIENCE: Good things and bad things happen when you take steps. So you have to decide whether the good things outweigh the bad things at your current point in time?

PROFESSOR: Yep. Basically it-- opportunity costs. Risk management, future risks, unknown risks. There should be some unknowns and knowns there. So while the players should know that there's going to be some positive and some negative, they might not know the values of those positive and negative outcomes. So there's some kind of uncertainty going on in the game there.

And then some of the side effects-- sometimes when you make a decision, there's going to be an immediate change in player state. Sometimes it's going to be a long term. It's going to chain or combo up from previous decisions, causing problems in later things. So easy way to think about this, again, is of course strategy games. But you're also welcome to apply this to any genre that you might like to experiment with in this project.

So here's some suggested goals. Think of project-- yeah, go ahead.

AUDIENCE: Do we still have to use planning for randomness in this game?

PROFESSOR: Yeah. So there should be some amount of randomness in this game. So it's building onto it. So there should be some kind of uncertainty. It's not the focus of it anymore. So project two and project one were focusing on just that randomness. Project three should have some of the randomness in there. But really, what we want you to focus on is trade-offs.

So that's a tool in your toolbox. Try it out. If it doesn't work, if there's a reason it doesn't work,

then don't use it. But there should be some uncertainty there. And again, all this is building up to project four.

So things you should try out in project three-- work with new people. The method we're going to have to put you in teams is going to try to do that. So you'll be working in different teams with different people. These are basically auditions for project four.

Use a new game engine. Did everybody use Phaser? Yeah. You're not using Phaser for project three. So we're going to basically want you to go back to that game engine tutorial, think about the other game engines you have available. We had one team to use Unity, and they had some issues why they wouldn't use Unity. And actually, those issues will probably apply for project three as well. Maybe you don't want to use Unity. Maybe it's too scary. Maybe it's a challenge. Figure it out.

And the last one-- I posted this to Stellar. It's a light read. *Games For a New Climate* is basically our handbook for project four. Our client presented us with this information. This is basically how games are used for game based learning, for climate change, for talking about planning for future risks, the complexity of future risks.

Take a flip through the book. Read it now. Start it now. Have it read by project four. Again, we don't have required reading here, but you'll have a better game, you'll have a better product if you take a look at this and maybe even do a little bit of research on your own, which is actually going to be part of project four. We'll talk about that later.

But if you want to make a game that does this, you can. If you want to make a game for project three that is a game to help a policymaker understand the need to spend money, time, or resources on disaster preparedness as a result of climate change, feel free. Part of that trade-off design challenge is actually inside of that statement. The uncertainty is inside of that statement. Project three actually would turn into a possible springboard and a way to recruit people for project four. But again, you're not necessarily going to have the same group. So it's up to you to figure out if that's going to be useful for you on this team or not.

All right. And then the hard requirements-- please read this handout on Stellar. I'll actually read the handout on Stellar too this time. Sorry about that.

Maximum play length, again, is five minutes. Again, single player game. User interface tested for legibility and usability. It must use and play audio for the player. And then, like previously,

players can pick up and start playing the game with no external instructions. So if there are instructions, which there can be, it must be inside of the game or at least inside of the frame of the web page that you are using as a link for us to play within. And then, of course, must be delivered as a browser game, running on Chrome. And also, only in those engines that w e used in the tutorial.

Deliverables-- so actually, this is Monday the 29th. Wednesday is the 1st. On Wednesday the 1st, we have a guest lecture coming in. Swery65 is coming in to talk to us about his new game. His lecture is probably going to last about an hour. That means you're going to have two hours in class on Wednesday to work in teams and create a low fidelity prototype. It is not something you're required to turn in.

We are asking you to have a playable game on Monday, the 6th. Whether that's digital or paper is up to you, but we highly, highly, highly recommend making a quick, low fidelity prototype on Wednesday in the two hours you have, and then based on that prototype create the product backlog. It's the best way that I've found to make product backlogs. You might find a different way. That's OK, as long as you make one.

But on Monday, on the 6th, turn in the Stellar, your high level design doc, and what your product backlog is. So what are all the features that have been estimated, that have been put in priority order? In class, you're going to give a really quick--

AUDIENCE: Next slide.

PROFESSOR: What's up?

AUDIENCE: You're talking about stuff on the next one.

PROFESSOR: Oh, wow. I'm talking about stuff on the next one. Thank you. That's weird. In class, two minute presentations. Basically, just the core of your game design idea. Let us know what you're making. And then we'll be able to test your prototypes. I think-- let me look ahead. Yeah, so testing prototype is not required on the 6th. But if you do it, please we'll test them then.

We will also test on-- and this is where we actually do want you to have a playable, digital version. Basically, let us know that you have a playable game by October 8. And turn in via Stellar your sprint task list-- basically all the tasks with time estimates that are going to happen between Wednesday, October 8 and Monday, October 15. And the project due is the same kind of stuff that we did for this project-- your game prototype builds, your post mortems, your

design change log, updated design document if you have an updated design document. And again, two focus test reports this time. One can be created in class on either the 6th or the 8th. The other one should be created outside of class with external testers.

AUDIENCE: Question.

PROFESSOR: Yes?

AUDIENCE: Is that Monday the 13th or Wednesday the 15th?

PROFESSOR: Oh, my lord. Did I really make that mistake? Can somebody check on Stellar for me? So it should be Monday the 13th, right?

AUDIENCE: No, we think it's Wednesday.

PROFESSOR: Wednesday the 15th is when we're having people turn stuff in?

AUDIENCE: Yep.

PROFESSOR: All right, holiday. Right? Is there a holiday that week?

AUDIENCE: Yeah.

PROFESSOR: OK. Stellar is the right place, and I will update these slides before I post them to Stellar.

AUDIENCE: Let me check on the big schedule. It's Columbus Day. So Monday, there is no class.

PROFESSOR: So Monday, no class. So Wednesday, the 15th. So you got a little bit of extra time. But it's a holiday, so maybe you don't have extra time.

PHILLIP: But if you get it done by Monday, then you can let your code soak a little bit [INAUDIBLE].

PROFESSOR: All right. So we are going to do brainstorming, and we're going to brainstorm in groups that you are not currently working with, so you can meet new people. You're going to get a card. It's going to have a number on it-- one, two, three, five, eight, or 13. I'll hand out the cards really quickly. Go to one of these stations, erase whatever's on the board if you need to erase on the board.

If you're in five, eight, or 13, you're going to have these easel pads. We are going to give you two timed quick brainstorming sessions. Five minutes, small break, another five minutes.

When we break, change whoever's writing things down so everybody has a chance to speak. After this, each brainstorm group is going to be allowed to make two pitches to the larger class. What we're going to ask you to do is write a title for the game on a large Post-It, and we'll hand these out. One of these, yeah. We'll hand these out when we get there.

In your elevator pitch, make sure to address the core mechanic of the game and how it's applied to the design constraints. Games from project one that were not selected for project two can be pitched if your brainstorming group says that was actually a pretty cool idea. Let's pitch that too. But it must be altered for design constraints.

And I'll give you a break after that second brainstorming session to come up with these pitches-- about 10 minutes, I think. Any questions about this? Psyched? Pumped? All right, I'm going to hand these out.

[SIDE CONVERSATION]

PROFESSOR: Team one, pitch one, come on down. And just like before, you've got about a minute. Describe-- oops, wrong ones. Give us the name of the pitch, the core mechanic, and how this pitch approaches the design constraint. Pitch, go.

PRESENTER 16: So our first pitch is called Dragon's Lair. It bears no relation to the previous Dragon's Lair idea. The idea we came up with is that as a dragon, when you're acquiring gold, the more gold you have, the more heroes want to come in and mess you up to try and steal your stuff. So we thought that there's a fun trade-off there between acquiring more gold and becoming a more powerful dragon, but at the same time having more and more heroes wanting to come in and attack you.

We also thought that just in that, in itself, you don't quite get the idea of meaningful decisions, because everybody's just going to want to be a more powerful dragon and prove themselves as, oh, I can handle more and more knights. So we thought it'd be interesting to have this idea of not just being an enemy to everyone. You can actually ally with some of the villages in the game so you'll be making less money, but then you'd have more people on your side to help you defend. So the trade-off here is between being nice to people in order to get more defense or slaughtering them or just steal their money.

PROFESSOR: Nice pitch. Number two.

PRESENTER 17: All right, so our second game is called caffeine worker. And it's based on the idea that caffeine will pretty much simulate your [INAUDIBLE] where you can actually get more work done under less amount of sleep. But it also tends to have side effects.

So our game pretty much revolves around you are a worker, and you want to get these tasks done. And you want to get these done as efficiently as possible. So we're going to have something like a 24 hour clock. We're going to have night and day. We're going to have a lot of different sort of actions that you can sort of, like, decide on executing, such as you can figure out how many hours you want to sleep, how much caffeine you want to take, and also you can also figure out your eating schedule as well.

So we don't really have the nitty-gritty details, but we imagine that this game will involve a lot of scheduling, a lot of sort of-- I don't know, the trade-offs between sleeping and eating and drinking coffee. But in the end, I think what we want to do is that we want to make this game sort of like a platformer, where you are actual in control of the character and for [INAUDIBLE] your tasks, something you need to go from point A to point B.

But these different sort of attributes, how much caffeine you have over time, how much sleep you have, whether you're eating enough, whether you're happy or not-- those sort of effect your motor skills as well as some other implementations that we have yet to figure out.

PROFESSOR: All right. Thank you. Thank you group one. Group two. Pitch one, category A. Remember, it's one-minute pitches.

PRESENTER 18: All right, so here's the idea. You are an administrator at MIT. And your goal is the management of MIT and promote MIT's ideals and maximize their endowment and all that. And your goal is to carefully trade-off the short-term versus long-term policy. For example, you've got this plot of land. You could lease it to Pfizer for 60 years and get 2 billion straight to the endowment. But then you can't expand it till later.

And do you want to appease the students to get more in tuition? But maybe you need that money, so you raise tuition, but you also get rid of your unpopular flag policy at the same time. So you can play as many different administrators. You can play as, like, Dean Colombo. You can play as Kevin Kraft from the Office of Student Citizenship. You can play as the shady MIT corporation with this weird board of self-voting members that no one really understands. That's our pitch one.

[APPLAUSE]

PROFESSOR: What's the name of that one?

PRESENTER 18: MIT Simulator 2015.

PRESENTER 19: Our second idea is build a car. And we thought that you could start out with a limited amount of money, and we will send the car you build after a turn through a simulator. And so you don't exactly know which maze it's going to get. So depending on the result of that, you have trade-offs of how to spend your money, what parts to buy, and build a car.

PROFESSOR: Great. Group three.

PRESENTER 20: Hi, everyone. I'm going to tell you about Fight or Flight, which is from our first project. And if anyone played the paper prototype, it's going to be completely different.

So the idea is that you're playing this platformer, which is a map that has a bunch of separate little rooms. So any given little room might have four or five platforms. It's not really big. And so this whole big map has a bunch of these little rooms, all connected by different passageways. And you can get from different places to different places.

And all of these rooms are potentially exploding. So this is some sort of spaceship, and there's a meltdown. Or, I don't know, this is, like, the tavern. And near a volcano, things are filling with lava, or something. However you want to skin it.

So each of the rooms will give you some sort of warning about whether or not it's about to explode. And then some time after the warning starts, it explodes. And then if you're in there, you're in trouble.

So the trade-offs in this game are that-- so the overall goal is to get somewhere, is to get out, is to get from one side of the map to the other. And in between, there's a whole bunch of other people who want to fight you, just for the heck of it. So they want to fight you, you want to fight them. But the more you fight, the more tired you get. And the more tired you get, the slower you move. And the more slowly you move, the harder it is to escape a room once it starts exploding. So you can choose to fight, and you're probably going to kill them, you could kill the enemies because they're AI written by people in this class. Versus, like, the player who can actually just implement strategies and things.

But if you do go and have fun and try to kick everyone's ass and you succeed at that, well, then you're panting in the middle of a room that's about to explode. So fight or flight everyone.

PRESENTER 21: The second one is Jigsaw World, another platformer where you are inside a jigsaw puzzle, where every time you get to the end of a block section square thing-- they're oddly shaped because it's a jigsaw puzzle-- you get a choice of a new set of tiles to put on the end of the one you've already gotten. Sometimes you have to go back and pick one up off the beginning of your puzzle and put it at the end.

And various power-ups make certain ones incompatible, like being able to fly. That means you can't swim. Being able to swim means you can't jump-- all those kinds of things. You're kind of creating your own world that you then have to navigate differently. Oh, yeah. And deconstruct it. You never get to see the whole thing at once because it's falling apart as you're taking it and moving it somewhere.

PROFESSOR: Thank you. Group five.

PRESENTER 22: So our first idea is a game called The Gun Wars, in which you're managing a defense company. And you're trading off who you're going to sell your product to amidst a bunch of different clients. So you could have somebody go up to you, you get some information on who they are, say, hey, this is James, seems like an OK guy, and he wants to buy, like, 10,000 missiles, air-to-surface missiles or something.

You have to surmise him, figure out if that's somebody you want to sell to, figure out at what price you want to sell. And then there may be some consequences for that. Maybe you sell to Jimmy, and then he launches an attack on some country you like or don't like. But more importantly, maybe he does something your other clients like or don't like. And so you lose other business.

Or you get new business from people that are Jimmy's friends and realize they can buy weapons from you. So it's sort of a social responsibility game. You're a defense contractor. You're managing some pool of money, trying to make money off of contracts. And the trade-offs come with who you're deciding to buy and sell from.

PRESENTER 23: Our second idea for the game is Bullet Craft. Bullet Craft is a fast-paced, top-down shooter game that follows the simple formula, the classic formula, that the player controls the ship, moves around, [INAUDIBLE] that's coming at him, and shoots the enemies that spawns from

the top of the screen.

Well, you might ask, where does trade-offs come into play for Bullet Craft? So there are actually various ways we can augment the game play so that we will incorporate trade-offs. For example, on the micro level, we can have ammo limits or overheating guns. And we can have a fuel limit which if the player moves, he consumes fuel which pollutes the environment, and the enemies become stronger because of that pollution.

At the macro level, we could also have the player choosing different types of weapons, spending their money on different types of ships and even a crew. What I like about Bullet Craft is it is very easy to implement, and it's guaranteed to be fun, so that if we make Bullet Craft, we are sure that we will be able to make a successful game.

PROFESSOR: Group eight.

PRESENTER 24: So the first idea we had is Daydream Inc. So in this game, you're like a regular office worker. And you have to try and not get fired. But the goal of the game is to succeed in your daydreams. So there's basically two mini-games within the game. There's surviving the office life, and then there's also your daydreams where you're fighting your boss or something.

And it's interesting because you use tools from the office to build weapons for your daydreams. So you might get a stapler and a rubber band and make a staple gun or something for your daydreams. So you have to spend both time and research from the office life in the daydreams without getting fired, so you can eventually win the game. Also, if you become CEO in the office life, so if you do really well there too, you lose.

PRESENTER 25: So our second idea was Score High. So basically, you're an MIT student. And you have a list of maybe three classes. And you have some stats-- like, your food, sleep, happiness, health. And the thing is, each of your classes has PSETs and tests. And you have to get to the tests on time, and you also have to turn in your PSETs on time.

And so how do you do that? So you run around the maze, and you have to get to the right building at the right time. But you also, when you're there, actually have to take the tests. And depending on-- you pick up different items. And say you drink 10 cups of coffee, you're really jittery.

Instead of seeing all of the options for the multiple choice question, suddenly you can't look at it, and you only see one question. And somehow you miss the back of the page. So you fail

the test, right?

So the whole game is about choosing-- you know, when am I going to drink coffee? Or maybe I'll buy my friend coffee, and he'll give me the PSET answers so I don't have to spend five minutes doing the PSET. And managing those trade-offs while also running around this maze and getting everything done on time. And if you fail any of your classes, you lose.

PROFESSOR: Group 13, come on down.

PRESENTER 26: Our first idea is Ghost Face. The title's a working title. The idea is you're in a maze, actually a maze. And you have very limited visibility. You can only see a certain radius that is much smaller than the actual size of the maze. And there's ghosts coming at you.

And basically the idea is to get to the other side of the maze, solving the maze, but you only have limited disability. So you have to explore the maze and remember which path was a good decision and which path wasn't. So you have to trace back a lot.

You can also add in more things. Like, you may have a certain number of beacons, light beacons, that you can place in different areas, but a limited number of them. So you want to be-- you want to make a decision as to what would be a strategically good place to place a beacon and reveal an important and critical part of the maze so that you can see more of the maze. So that's our first idea.

PRESENTER 27: So our second idea revolves around DNA. We haven't found a good title for it. But the idea is, let's say there's an organism. And you have a certain set of diseases that you want to try to get rid of. And your goal is to somehow go around finding other organisms that have their set of traits. What's the diseases that you don't have, some other diseases that you have.

And play a game of chance-- so whether it's worth it to spend your time and energy eating or mating with the other organism and seeing if those genes come across to your [INAUDIBLE]. And the trade-off is-- the trade-off is, do you want to mate with this organism or not? We're still working on the details of this, but the idea is this gene competition [INAUDIBLE].

PRESENTER 28: And random mutation.

PRESENTER 27: Yeah, random mutations [INAUDIBLE]

PROFESSOR: OK. Thank you. All right. We're actually really good on time today, so you'll actually have some

time to work in your teams after we get you all sorted out.

So we heard about Dragon's Lair, Caffeine Worker, MIT Simulator, Build a Car, Fight or Flight, Jigsaw World, The Gun Wars, Bullet Craft, Daydream Inc., Score High, Ghost Maze, and DNA.

12 ideas-- actually, a few of them could probably combine pretty well together. So we're not really throwing the ideas away. We're really just trying to get together into groups of six around a base idea we can start with. So if you've got a game that you're really, really excited about working on, come right down and put your name on it. And make sure your Post-It has your name and the game engine you used in the tutorial assignments.

Let's see what we got. All right, raise your hand if you still have your name tag, your Post-It. One, two, three, four. All right. We have Caffeine Worker only has one person on. [? Zigamantias ?], you have your name tag back. Nope, sit down. Just sit down.

Jigsaw World. Sorry, Kathleen. All right. We've got one, two, three, four, five, six. DNA is done. So DNA is Derek, Eduardo, Ava, Harry, Trisha, and Lauren.

One, two, three for Ghost Maze. Three for Score High. Two for Daydream. None for Gun Wars. Two for Bullet Craft. Three for Fight or Flight. One, two, three, four for Build a Car.

Two, three, four, five, six. Oh, cool. MIT Simulator. Don't get me in trouble. Norman, Jordan, James, Peter, [? Szeun, ?] and Sam.

Bullet Craft and Daydream-- first off, does anybody who has their name tag on them, are they all excited about either of these two games?

AUDIENCE: I am.

PROFESSOR: You are? Is your name-- do you have your name tag in your hand? OK, raise your hand if you have your name tag in your hand. All right, come down here and put your name on a game.

I'm going to get them all. I think we have enough to get every game to six. 10, nine, eight, seven, six-- hey, five. Cool. Thank you.

All right, Fight or Flight, Bullet Craft, Daydream I think are all either going to go away or combine somehow. Score High, one, two, three, four. One, two, three, four, five on Ghost Maze. Just needs one more person.

Five on Build a Car. Just needs one more person. Four on Dragon's Lair. One, two, three, four, five, six. OK. Daydream is gone, Bullet Craft is gone, Fight or Flight is gone. Move yourself to another team. All right, Ghost Maze has six and is done. Score High has five, six. Done.

AUDIENCE: Everything is full, isn't it?

PROFESSOR: Yes. So put yourself on one, two, three, four, five, six. Put yourself on any of them. Five, four, three, two-- thank you.

So here are our final teams. We're going to do one last shuffle. If anyone is like, I don't want to really be on that one, they can move to another. For Dragon's Lair, we have Roy, [? Mikael, ?] Liz, Zigamantis, Devon, Caleb, and Kathleen. You have a good mix of Unity and Flixel, so I imagine you might use Unity or Flixel.

Build a Car, we have Rodrigo, Anderson, Matt, [? Taj ?], Jeremy, and [INAUDIBLE]. We have Unity, Unity, Unity-- three Unity, three Phaser. I wonder what you're going to use.

Score High, six. Miriam, [? Sen ?], Megan, Kevin, Sabrina, and [? Zinue ?]. And we have Haxe, Haxe, Flixel. Oh, wow, lots of Flixel. And two Unity. So maybe one of those.

Ghost Maze, [? Jutoshka ?], [? Salaam ?], Daniel, Kevin, Justin, and Rachel. And Unity, Unity, HaxeFlixel. Ooh, it's going to be kind of tough. We've got three Phaser, two Unity, and one HaxeFlixel, so that's going to be a hard decision to make.

DNA, we've got seven-- Jenny, Lauren, Ava, Tricia, Harry, Eduardo, Derek. HaxeFlixel, HaxeFlixel, HaxeFlixel, Flixel, and one Unity. So probably one of the Flixels.

[GROANING]

You decide. MIT Simulator has Unity, Unity, Unity, Unity, Unity, Unity, Unity. Four Unitys and two Phasers, so probably Unity. And that is Norman, Jordan, James, Peter, Sam, and [? Szeun. ?] We have how many students are not in class today? Five or six students not in class today. They are going to have to-- what's up?

AUDIENCE: Make them a team.

AUDIENCE: Game to learn how not to be late.

PROFESSOR: That's a good question.

PHILLIP: [INAUDIBLE] simultaneously on Wednesday, that might be possible. But if a few of them don't come back on Wednesday, that may not work.

PROFESSOR: So teams will be finalized on Wednesday. These are the teams as they stand right now. You have 40 minutes to meet as a team to figure out if you all match schedules, to talk about the idea you just pitched, and make a really quick lo-fi prototype, or write down that page that you just said out loud on the spur of the moment.

Figure out what is that game that you're making. Work on that vision statement. And feel free to grab your things and put your names on it if you like.