GUEST SPEAKER: That says choices for your game project. These are the six choices that I want to make available to you.

The next one is perhaps the most challenging one intellectually. It's about new funding mechanisms for disaster preparedness. I will explain this. We call it forecast-based financing, or FBF, and it's the one that has the most potential to transform the humanitarian world.

All right, moving to the next one. The next one is the most complex. Do you see the drawing of the time bomb with the fuse? OK. So allow me to give a little bit of context. We did a little bit of this conversation when I showed up a few weeks ago, but I imagine you all forgot by now.

The humanitarian sector has two main funding mechanisms. Imagine a pot of money. Even if there's almost no money there, if the pot exists, money can go in to then be taken out to spend. One of the pots of money that exists, one of the funding mechanisms that exists, is for disaster response, meaning hurricane or flood or volcanic eruption. Something has already happened, is happening, and is killing people. People are dying, but if you did something, you could save lives, for example, get on a boat and rescue people who are stranded on a roof. Or people are sick, go give them medicine. This is disaster response, and if you spend money, you can do very good things when the funding mechanism exists.

There's another pot of money which is for a normal day. The Red Cross has vehicles. The vehicles need maintenance. They need to change tires. You need to buy a new computer. You need to pay rent and electricity bills. You need to train staff. So on a normal day, you have a budget for, you know, either maintenance or development of new projects or writing proposals and so on. And that pot of money exists. We call it the annual appeal. Every year we appeal for help, and people or donors or governments give us some money.

What does not exist is what the drawing, the cartoon depicts, which is when there's a signal from science that a disaster-- that something bad is likely to happen soon. We don't have money to fund action before the disaster, after the forecast. So you can click to see the text. The context is that there's that missing money pot. We have my for after a disaster, we have money for the normal day, but we don't have money for before the disaster after the forecast.

The issue is that if you spend money in that time window before the disaster and after the science says a disaster is likely, you can do very good things for very cheap and relatively

simply and much more reliably. It's not only more expensive, but more difficult, to save a life when the storm waters are flashing through a city and people are about to drown. The boat is unsafe. Whereas if you do it the day before, when the water has come down from the sky, but it's still in the river and not in the city, people can take action by just walking away on their own means, with their valuables, and not just depending on someone to show up with a boat.

So we came up with a financial mechanism. We call it forecast-based financing for disaster preparedness, which is [INAUDIBLE] in our particular method. There's a pot of money, so like a bank account. You need to have a threshold of disaster risk that needs to be exceeded. So it can't be two drops of rain. It has to be a lot of rain. It cannot be some breeze. It has to be a hurricane of Category 1 or more.

But once the threshold is established and reached by the forecast, then there's some standard operating procedures, some actions that people have to take, spending money to save lives. And that is something that if you could come up with a game that captures that, this is the one that if we help communicate forecast-based financing-- I said to the public, because it can work for a broad audience digitally. Ideally it would reach those who can contribute either money or decisions, muscle power.

The chances of embrace for this one are not as high as all the previous ones, but if it were to happen, it would have the highest impact. Because right now there is billions of dollars being spent in the two extremes, and if a small fraction were spent-- invested in that precious window of opportunity when you see the fuse getting close, that time of effort can really have a very, very high impact.

There is a link to a journal article that should be very clear for MIT students. We also have some simpler readings on general. And I have been engaged with it. And I'm ready to help. This is, I insist, the one that is intellectually most challenging. But I think if you nail it, it will also be the most rewarding. Questions?

PROFESSOR: So one thing. When we first started designing this class, this was actually the problem that we were most interested in, mainly because the other problems we just hadn't heard about since. The cholera was a recent proposal that was asked for, and Ebola, of course, very recent as well. So if you really enjoyed some of the challenges from projects two and three, and you really took on the planning aspects, the trade-offs aspects of those two games, if you're looking at kind of trying to communicate to another person how these kind of probabilities

works, this is probably a really good project you might be interested in.

And I should say also that working with a few MIT teams, including the Humanitarian Response Lab, including the Environmental Engineering, we have two generations of three students plus on professor go to Uganda to work on making this happen for real. We got German money to do it in Togo and in Uganda. So it's beginning to happen, but it's hard to explain because it's dry. It's like explaining insurance. People get bored before they get it. So maybe a game can help motivate.

STUDENT 1: Our team is working on forecast-based financing. So because we're about planning for disasters, our game is pretty heavy on chance. But we want to teach the players to do is to use planning and forecasting in order to reduce the effects of that chance and sort of gain valuable skills in mitigating that.

PROFESSOR: So basically if you have-- there's one, two, three, four, five testers, plus let's say each team send out two people to test other games. Remember to rotate. We're going to do this for about 20 minutes, as long as it takes, and then see where we are and do it again to get some just quick testing and make sure that the five of us get to play a good number of your games and give you some feedback on that. So remember if you're using digital-- if you're not-- If a computer's not being used for testing, close it, or make it look like it's not being used for testing by typing on it. All right. Let's get started.

[INTERPOSING VOICES]

STUDENT 1: Hi. We're forecast-based funding. The general idea behind our concept is that planning ahead for disasters is much better than trying to react to them. So if you can have operating procedures or ways of planning for them and money allocated for that, you can reduce loss of lives in the event of disasters. We are actually between two prototypes right now that we're testing to try and get at the ideas.

The first one is a sort of higher-level city-based simulation of cities at risk of disaster, which you then have to fortify by training volunteers or preparing for upcoming disasters in order to prevent too much damage from happening to them. One of the problems that we're seeing with the game is that it's kind of abstract and not as interactive for players to connect with, but they are getting a good understanding of the idea of planning ahead.

STUDENT 2: And so to try and address those issues, we have a second prototype right now, which is about trying to actually, like, rescue volunteers in a flooding-- rescue people in a flooding city. And so that hits the other end of the scale, where the player is told ahead of time this disaster was planned for, versus this disaster was not planned for, and the appropriate effects for each. And then they have to rescue people under those two different conditions, and then they get to directly compare what the experience is for, like, acting in one circumstance versus the other.

STUDENT 1: Over the next couple days, we'll be looking at what we learned from both of them and trying to either combine them or pull out the parts that we thought were really useful to make one game.

PROFESSOR: I can tell you've worked hard on this. This one's a hard one.

STUDENT 1: Yeah.

PROFESSOR: Did we assign you a target audience for this one?

STUDENT 1: I think it was stated a couple of times that we should be looking at people like policymakers or donors in the sense of people who would be allocating funds from governments or nonprofits, things like that, basically to make it clear that this type of planning is a good idea.

PROFESSOR: Oh. And what-- have you decided on a technology yet, or are you still pondering it?

STUDENT 1: We're probably--

STUDENT 2: We're using Phaser.

PROFESSOR: OK.

[LAUGHTER]

All right. Thank you.

[APPLAUSE]

STUDENT 1: Our game is called Hello Waves. We don't have the title screen in yet, but the idea is still forecast-based financing. So it's a little washed out on there, but if you can see, there are these five different sand castles, with workers associated with each, and up here in the corner, we have a forecast of what the water level will be like in a couple days. This is the water level

right here, and throughout the game it'll rise and fall, and the idea is not to let your workers get drowned by the rising water.

The other aspect of the game, though, is that you have a certain amount of supplies, and whenever you move someone from their castle in order to prevent them from getting drowned, they will consume supplies from that stock, and if you don't have enough supplies then they'll take damage.

So as you can see right now, everybody's in their own castle. If we click Next Turn, the water level will change, and we'll see that the forecast will update. We have a range of values on this forecast, a high prediction and a low prediction, and if you mouse over any of the castles, you'll see a red line appear on the forecast, just to help you orient what heights are on the forecast, or if you're looking at the forecast, what it is in the real world.

So you can see the water move over these couple days. Right now the forecast is pretty low. If we see that it's kind of getting close and we're worried about someone, we can click and drag them to another castle, and on the next turn they'll move over.

You can also click people to toggle between building and gathering. It's very hard to see the status text right now because it's small. We threw that in quickly for right now. But he is set to building and these three are set to gathering. So we'll see that the each turn these three will gather one supply each. This teddy bear will consume one. So we'll see supplies go up by two and victory progress will go up by one.

What we have planned for the rest of the game is we've made a lot of progress in terms of the intuitiveness of our game once you understand it. But right now, there's a lot in the game that isn't explained from the get go. And so if I give that spiel to anyone, they can play the game fine, and they do pretty well with it, but you just open up the game you're lost. So a lot of our work is going to be on the UI, of making things more self-evident and making sure that people can understand what's going on and what their actions will do without me having to stand there and tell them.

PROFESSOR: Any questions? All right. So I'm going to ask you again, Tom, what is your biggest risk going forward?

STUDENT 1: So--

STUDENT 2: I mean, I think in terms of our technical implementation, we've already got good playtests of our work done, and we have all of our major graphics in, and things like that. So I think our major risk going ahead is not being able to explain the game properly, and so even if we have a finished product, it may still not be playable if we don't write good instructions.

GUEST SPEAKER: All right. So on that front, think of incremental additional features. So the game begins with a super-simple, even maybe too boring choice, but then a new choice arrives. So the learning by playing can be staggered in a way that is intuitive. Good. Good luck to you, and we'll talk more about choices later, but-- I should say that this high versus low prediction is something that I wish scientists did like you're doing, because most people say one value and then it's not that value, it's up here, and then everyone doesn't believe forecasts. Good. Thank you.

STUDENT 1: Thank you.

[APPLAUSE]

PROFESSOR: OK. One down.

STUDENT 1: OK. So it's a little light on there, but our game is Hello Waves. It's a game about forecast-based financing that we're developing for the Red Cross. The idea of forecast-based financing is using the idea of forecasts about the future in order to make decisions that are more effective than just reacting to disasters when they happen.

So I'd like to show you a playthrough of our game. So, as I said, Hello Waves. Here are the instructions. We would have done a tutorial if we had enough time, but this gets the idea across. And designing a good tutorial that actually teaches the player well is kind of hard to do in a logical flow, so we have these instructions that explain how to play, what the point of the game is, and a couple of tips about how the game works.

So if we go into the Play screen, you can see something similar to what we showed you a couple weeks ago, where you have all these different toys at their castles, and as you go through the days you'll see a forecast of what the water level is going to be at over time. And so you can see that right now they're gathering candy. The water level will change. This character is underwater, and so he takes damage. And so what you actually want to do is you want to move toys out of the water so they don't get damaged. However, they can only move one castle over at a time. So because we didn't think ahead well enough, we'll see that this truck will actually take damage on the next turn, because he's underwater. Actually, he ended

up not underwater. We got lucky there. But theoretically he would have.

And so as you go through it, the end goal of the game is to build a castle. And so every toy when they're home can either choose between gathering candy to feed evacuated toys or building the castle to reach your end goal of the game. And then the idea of the forecast is that you want to use the forecast in order to know how much candy you're going to need over the next couple days, and to use it to know when you're going to need to evacuate various toys from their variously-heighted homes.

So back to the presentation. Full screen.

So we had a couple of challenges in the design process. The first-- and the major one-- is really that we were trying to teach about forecast-based financing, which was a bit of an abstract topic. It's a little different than just thinking long term, because you have to use the idea of there's some information we know about the future that we want to use to make optimal decisions, or at least decisions based on some idea of the risk that's out there. But we also wanted to avoid things like just pushing a button to win, where you have all the information you ever need, and there's just one option that you know you're going to pick every time, and the game has no thought whatsoever. And we also needed to think about how we were going to communicate the forecast to the player so that they could then use that to make decisions.

One of the problems that we also ran into related to this was that we focused a lot on the idea of forecast-based financing as the topic and then tried to build a game built on top of that topic instead of building a game that used forecast-based financing. So that held us back a lot in the beginning when we were trying to come up with ideas.

We also had a really difficult initial target audience of policymakers, 50, 60-year-old government officials or people at NGOs who were going to be using forecasts to make decisions of some sort, and it was supposed to teach them about how they can use forecasts to make these decisions. Except this was a really difficult audience, because they don't generally play games and they don't have a lot of time to learn about this kind of thing, and so we couldn't really expect to get them to sit down for a long period of time and play around with our game.

In order to deal with these problems, we came up with a couple solutions. The first thing was

that we had no idea what kind of game would work or would make sense or anything like that. So what we did is we just broke our team up into multiple groups and came up with a bunch of different prototypes. On paper we had two different prototypes, one that focused on the idea of managing a city and its resources and its response to disasters, versus focusing on individual people and how you're going to move them around to keep them safe from the disasters.

And then we went into a whole bunch of different digital prototypes, one that was a text-based game about managing a city. We had one that looked like this, where you had two different cities and you were specifying what workers were going to do or when they would leave the city in order to stay safe. And then we would take the different ideas that we were learning from both of these prototypes, combine them together, take things out, and our final prototype actually uses ideas from all of these.

Another solution that we kind of got lucky with is when Pablo came to play our game, he told us that we actually shouldn't focus on the policymakers, because he wasn't confident that he could actually get them to play the game. And so we switched our target to being grade schoolers, which is why you saw the sort of cutesy art with the beach and the toys. This actually made it a lot easier for us, because we could target people who probably had some experience with games, or at least wanted to do something fun and would be curious to learn about our topic.

STUDENT 2: So another for our development process-- Or I can just speak here, right?

PROFESSOR: Yeah. Just speak right there.

STUDENT 2: So another big issue-- another set of challenges that we ran into was through our development process. And so we had initially issues with communication and facilitation. Our team had a wide variety of experiences and backgrounds. Some were hardcore gamers, some mostly mobile gamers. And so there were initially a lot of disagreements on what level of game we wanted to create and what sort of game, casual versus hardcore, that we wanted to create, and so we needed to overcome challenges of facilitation and communication within our team.

Another major challenge in our development process that we had to face came from our design issues, where for a very long time we had a vague vision of what to do. We didn't know what kind of game we wanted to make, and so we purposely tried to keep our game ideas vague as we built prototypes. But then we ran into issues where we would have solutions but,

like, no consensus on which solution was best, and where we went for long periods of time without having a clear direction of where we wanted our final game to be.

So our solutions for the challenges posed by the development were a team structure. And so we structured our team loosely into three subteams, a production subteam which would take care of production, like the deliverables and making sure that all the game ideas are being communicated properly, a technical team which worked primarily with the code and making sure the game got done, and then a user experience team which handled art, UI, and sound. And so we kept the responsibilities flexible. So as team members got busy over the semester or as changing conditions led to different people contributing, we kept-- responsibilities were able to easily flow between teams and team members.

Additionally, another idea we started with in the beginning was the idea of subteam leaders, which were the two people marked with the Ls. But that was an idea we later abandoned in favor of just having a more flexible team structure or flat team structure.

Another solution for helping our development process was the use of good code practices, and I cannot emphasize enough that this really helped speed up our development, because we didn't run into trouble with code. It was mostly with design. So we used Yeoman, which is a JavaScript module system, and basically it allowed our code to be interoperable. We could write one module separately from another module. So that solved a lot of issues with dependencies or people working in parallel, because it allowed people to work in parallel without overwriting each other's code.

We also used good code practice with state machines and MVC, which is model-view-control, and so we had a very object-oriented code, very modular. And when we did need to change our code, rip it all out and put it back in, it actually didn't turn out to be too painful, because we just had to switch a couple objects around.

And then one final solution that we used for our development processes were Slack and Scrum. So Slack is like a modernized IRC chat room, and it's very feature rich. It has a lot of integrations with GitHub and Google Drive and things like that. And so that sort of real time communication actually made it so that we didn't really have to meet outside of class too often. If someone was working, we would just email out saying I'm on the Slack, and then people could meet on the Slack, and it was full-featured enough-- like we could send attachments and things like that-- that most of our in person communication could be done in class. And in class

we adopted a sort of, like, daily Scrum format, where we simply said what we had done since the previous class and what our goals were until next class.

So in the end, though, we did have to cut some features. These features mainly were the idea of multiple-- again, a tutorial or multiple levels, simply because there would just be too much content that we would need to playtest in order to make sure that it was of consistent quality and got our message across. And also trying to add more individuality to the toys that you saw. They have different graphics, but that's as much as we could do given the time constraints.

So kind of just bring it back. Our three worst decisions were first, we did end up spending a lot of time on code and assets that never got used. We maybe, like, used 10% to 20% of our final work in our final project. Actually, maybe that is a bit overkill. But OK. Maybe 30% of our final code and assets in our final project.

This really was due to this second bad decision, that we kept the game and its direction too vague for too long. We always were holding out for, oh, maybe we'll be able to come up with a better game idea, or maybe we'll find some magic solution to how we can make forecast-based financing into a game. And because of this mindset, we spent probably the first half of the project, like, just staying too vague. And that hurt us in the end, because we spent so much time going in all these different directions.

And really the decision that kind of captures those first two, though, is the fact that we tried to make a game on top of forecast-based financing. So we had forecast-based financing, and we were like, how can we skin this as a game? Whereas once we switched that mindset and thought, let's have a game, and how can we put forecast-based financing into it? I think that was the moment that we then came together as a team and really started making the final game that we wanted.

So our best decisions--

STUDENT 1: So one of our best decisions was that we chose good tools at the beginning, which meant that as we went through all these different digital prototypes, we actually didn't have to completely rewrite our game. We could pull out the way that workers worked in one game. We could pull out the view that we were using in another game, and then we could just combine them together, and that allowed us to move quickly whenever we were changing our prototype.

We also weren't afraid to trust each other, both in terms of what everyone was working on, but also in terms of the decisions that we were making. And so when we said that we had to throw something out, we all understood that it was for the good of the project. And we didn't have a lot of complaining or hurt feelings when something didn't get put in or when we decided to throw out certain assets or code.

And when we got to the end, we had been through enough of this vagueness that we were all kind of frustrated with it, and we realized that we had an idea that we all liked and we really got on board with it and made it happen. Once we started working on our final idea, and we saw that it worked, basically every decision from that point on was how do we make this game better, and we were all on board with that vision.

Thank you. Any questions?

[APPLAUSE]

PROFESSOR: Anyone in the audience have feedback for them first, before--

AUDIENCE: You can't see those sand castles very well.

STUDENT 1: Yeah, we'll have to make all of our slides darker.

PROFESSOR: Yellow and white.

AUDIENCE: Yeah. Like, the yellow and white, on your early screens. I was doing this on your instruction screen.

PROFESSOR: It's also--

AUDIENCE: I realize that's the game, not the presentation, but wow.

PROFESSOR: It's also a little faded in the game itself. It could be the display resolution you're using. So we're all done with everything, try a different resolution. See if it makes a difference.

AUDIENCE: The music in the game level was too high. It was overpowering your voice. So I think you were almost hollering just to be able to be heard. it doesn't need to be that loud. So you can just crank down the volume, either on the computer or on the controls over there.

Something that I had a question for, you don't have to answer it today, but you might want to

put it in your presentation was, how long was the design of what you eventually established on the table before you decided that this was the thing that you were going for? Because I'm assuming it was one of your vague-- it came up during your vague idea phase, and you're implied that you were in that mode for too long. But I don't know how long it was on the table, or was it only something you figured out at the end of the vague idea phase? Because otherwise you wouldn't have been able to switch to this idea if it was not on the table in the first place.

And how did you decide this was it, that the sand castle game was going to be it? I understand-- I think you very clearly explained the benefits of deciding this was it. But how did you come to that conclusion? So that's stuff I wanted to hear more about. But you don't have to answer it now.

AUDIENCE: Yeah. Just a little bit of specificity. You mentioned you spent too long on that design phase. I wonder how long that was.

PROFESSOR: You said Pablo switched your target audience for you. Telling us whether that was because of the game that he saw or because of something else-- if you know that information, throw it in there. If you don't know that information, that's fine. Knowing why you dropped the team lead. You mentioned you dropped it, but you didn't exactly say why. Again, really quick. This is what-- we weren't getting blah out of it, or the flexibility was more, whatever. And then, yeah, to--

Oh. Defining your terms. You were saying things like casual versus hardcore. Give a little bit a definition of what you mean by that. It means different things for everybody. So what is your use of that.

AUDIENCE: Yeah. I think he specifically said hardcore and mobile, but there are hardcore mobile players out there, so this is like--

PROFESSOR: And casual can be considered a version of hardcore, just in a different way. So just be really-- just be a little more clear about-- because I think you were talking about the target audience and the kind of players and the kind of games they might play. So just be a little bit more focused on what exactly you mean by that.

That's it from me.

AUDIENCE: Yeah. So in terms of technical observation, Yeoman is not really a module framework. It's just a system to generate the project and it puts different frameworks. Again, just check that.

AUDIENCE: So that's a terminology issue, then. Clearly it worked out for your team. So we're not saying don't mention Yeoman. It's clearly a good thing. But just check your definition of what it is. Because it'll be more helpful for other people to understand what it is so that they can think about whether they want to use it in the future.

PROFESSOR: Oh, and actually-- So your demo, you spent about three minutes doing it. We are going to have you have a player from the audience play your game live, without getting a lot of help. You can talk over it. You can, after a while, start helping them. We want to see them at least just to start playing on their own. Decide when you're going to put that demo in. You could actually combine them both together if you do it at the beginning or end. But if you do that, give the player a little bit of time before you start talking.

AUDIENCE: That's all I got.

PROFESSOR: Great. Thanks.

[APPLAUSE]

Snap, come on down.

STUDENT 1: Hello. We are Hello Waves. We're a game about forecasting, specifically this idea called forecast-based financing, of using forecasts to make decisions about possible disasters in the future and how to prepare for them.

We'd actually like to start with a playthrough of our game. So if anybody would like to be a volunteer to try it out.

AUDIENCE: The guest from not our class. Would you please come down, Andrew? [LAUGHTER]

PROFESSOR: Thank you for volunteering.

[LAUGHTER]

ANDREW: Hi, I'm Andrew.

PROFESSOR: Oh, do you need a chair? There's one right there. All right.

[? STUDENT 1: I'm not sure how to make it full screen.

But anyways, this is our game. I'd recommend looking at the instructions first and reading through them. And we'll keep the description minimal as you read through, just to show the player's initial reaction to it.

[MUSIC PLAYING]

ANDREW: Music.

STUDENT 1: So yeah. Start the game.

So as you can see, our game, as I said before, you control some toys on a day at the beach. And so by dragging and dropping them between the castles, you'll see that their status is changed, and say that they want to move on their next turn or that they want to be collecting-- or that they're going to be collecting candy or anything like that. And the game is turn-based, so all of the actions will resolve on the next turn.

ANDREW: Am I reading this right?

STUDENT 1: Yeah.

STUDENT 2: You can also access the Help by clicking in the bottom right corner.

STUDENT 1: And so when you go to the next turn, you'll see all the toys move as specified by the status bubble above their heads.

ANDREW: So go ahead to the next turn? [INAUDIBLE] and Next Turn.

STUDENT 1: But unfortunately you'll find that when toys have been evacuated, they're unhappy and need candy to survive, so they'll all take damage.

ANDREW: OK. I don't want to be in that [INAUDIBLE].

STUDENT 2: So you can try returning them to their homes.

ANDREW: Ah.

STUDENT 1: And so you'll see that on this turn instead now they have their statuses set to gathering candy, except for the dump truck, who's still evacuated.

ANDREW: Gotcha. OK. I think he needs to be evacuated.

STUDENT 1: Exactly.

And so the idea is that as the player plays through it, they get better at understanding how to use the forecast to make decisions about the future, both in terms of how much candy that they need to have stocked up in order to weather out the rising tides, and also in terms of when they're going to need to move their workers out-- their toys out of the areas that are in danger.

ANDREW: They're gonna be really affected.

STUDENT 1: I think you have those two guys.

ANDREW: Oh, right.

So is there a reminder of where they started?

STUDENT 1: Yeah. It's on the castle. It's actually blocked there right now by the toy in front of it. But there's a little shadow on there, an imprint. And because he ran out of candy and then-- Actually, sorry. Because he went to a place that was underwater, he took too much damage and then was swept away by the waves.

[LAUGHTER]

STUDENT 1: And so on the forecast, you can see that high water is coming for quite a while, which is going to be a danger for the toys, both in terms of possibly getting swept away and not having enough candy for all the toys that you're going to have to move out of the way.

ANDREW: [INAUDIBLE]

PROFESSOR: Ooh. He may be out of luck.

ANDREW: Yeah.

PROFESSOR: And when you've lost two toys, you lose the game.

ANDREW: Pretty good.

STUDENT 1: Thank you for playing.

[APPLAUSE]

STUDENT 1: So that's Hello Waves.

So in our game we had a few challenges to overcome. The first was that our game was based on forecast-based financing, which is a very abstract topic. It's a pretty understandable idea of using information about the future and ideas of risk in order to decide where to allocate resources. But it's still a bit abstract and building a game around it took a little bit of work. It's useful to note that it's different than long term planning. It's not just thinking about what will happen in the future, you know, building a dam to prevent water or things like that. It's actually about using the information you have to make the best decision for events that may be upcoming in the semi-near future.

We also wanted to avoid making a game that was overly preachy or simplified, where it was clear exactly how you're going to win and you could just basically push the forecast-based financing button to win the game. We wanted players to actually think and understand the concept there, instead of just coming up with the buzzword of forecast-based financing.

And finally, we had the challenge of actually communicating that forecast to players in a way that they would be able to understand and then make use of. You could see in that game that we had water levels, and it would show on the map, and we found that players were pretty good at using that in order to make decisions about what was going to happen in the future and how to allocate their resources.

The other big challenge that we had in the beginning was that our initial target audience was policymakers. Like for Snap, Pablo had come in and pitched us this game idea, and originally he had wanted us to build a game for policymakers that would help them understand the benefits of forecast-based financing and therefore convince them that they should develop policies that would give resources to plans based on forecast-based financing.

So I would like to take you through a couple of our prototypes, just to show you the evolution and comment on our process.

Actually, to preface that, we had a lot of prototypes, because our idea was so abstract and because we weren't sure how to address our audience. So we built a lot of prototypes to start with. We had ideas that ranged across levels of scope of what you controlled, where you controlled entire cities or where you controlled individual workers and moved them around, and then we would pull from all these different kinds of ideas, what worked, what didn't, what

did people understand, what confused them. And from that we got a really good idea of what concepts helped people understand the idea and brought them into this final game that we ended up with.

So the project started on October 15th, and this was our first prototype. It was a terminal-based game where you had some kind of information about a future rainfall, and then you had to type in your commands of how you controlled different cities. This-- Actually, people found it fun, but as you might expect the feedback wasn't very good and people didn't quite understand how to move forward with it. It put a lot of cognitive load on people.

So when we moved forward, we tried to give people more easily understandable actions to use. But the problem with this game was that it was time-based and it updated every second. And so there were so many numbers flying at people that even MIT students who playtested it couldn't understand. So we figured that people like policymakers who didn't have much experience with games really wouldn't be able to understand the game at all.

So instead we went to turn-based. And that helped, but at the same time-- it's tough to see on this projector, but we have a forecast underneath that says how much rainfall is expected. And again, that wasn't understandable to people, because they couldn't understand what three inches of rainfall meant for their city and they couldn't understand how that contributed to a possible disaster.

At this point, Pablo actually came by the class and played the game, and then he told us that he wasn't even sure that he could get policymakers to play the game, because they may not have enough time. Instead he wanted us to switch to grade schoolers, because we could teach them something about forecast-based financing and help them understand as they grew up. And this was great for us, because making a game that was serious, easy for somebody who didn't have experience with games to play, and also fun and engaging was too difficult for us, actually. So moving to grade schoolers was awesome.

He also suggested that we try to make the idea of rainfall or water levels more visceral, and that's when we came upon this idea of the rising waters. Whenever players looked at this, they instantly understood the concept of the game. The feedback might not be there. The beautiful UI might not be there. But the idea of having cities with workers in them and a rising water level coming towards them, everybody understood that, and it made it a lot easier for players to reason about the game.

From there we added things like nicer art, better feedback, which you can't quite see in a static picture, more improvements to how the forecast worked, and eventually we ended up with our final version today.

STUDENT 2: So to talk a little bit about our actual development process. So our team was structured into three main subteams-- production, which was in charge of managerial roles, deliverables, and playtesting, and so there was a shared responsibility there, a technical team, which was in charge of the bulk of the coding work, and a user experience team, which would be in charge of assets and UI design. We also initially envisioned subteam leaders, where we'd be kind of communicating through them. But we found the concept kind of redundant, and so we worked pretty much with, like, a flat structure between the three teams.

So from the beginning, we encouraged good coding practice, and so we used good tools available to us. One of these is Yeoman, which is a JavaScript scaffolding framework. And this helped us out a lot by basically automating a lot of our JavaScript tasks and making our code modular. We also used Phaser's state machine, which is this kind of badly documented new feature in the Phaser game engine, which was the JavaScript game engine that we used. It's a bit badly documented, but it did save us a lot of headaches, and once we figured that out, that proved immensely helpful. And we also used MVC, which is a software engineering term standing for Model-View-Controller. And again, by encouraging these good coding practices, we reduced dependencies and made sure that our team was productive.

In terms of communication, we-- also similar to Snap-- used Slack, which is kind of like a modernized chat room. It's very feature-rich, and so you can share files in channel and things like that. And we also used the idea of the daily Scrum. We implemented it in class, and we would say what we had done that class, what we would be doing, and what we wanted to do until next class.

And so the major challenges that we faced during the development process, though, were that our team members came from very different backgrounds and had very different preferences about games. You know, some of our team members were very hardcore StarCraft players and very good at RTS games, while other members of our team preferred like a more laid-back mobile game, Fruit Ninja kind of approach. And so trying to mediate those two viewpoints and trying to create a game that would engage both types of gamers was a challenge that we had to overcome.

And so another big challenge that we had in our development process, as you saw through our progression through our different prototypes, was that our direction was not very clear until about halfway through the project. And so partially because we had different ideas on what the game should look like, and also partially because we had such an abstract idea of forecast-based financing that even we didn't have that good of a grasp on initially, it took us a while to really get settled on what we wanted to build. And so this really challenged our development process and made us have to build a lot before we got something that we liked.

And so eventually we did end up having to cut some features, like multiple levels, or a guided tutorial for the player. We thought that this would introduce too much new content that would need to be playtested, balanced, and tested to ensure consistency with the rest of our game, which we viewed as taking up too much time. And we also cut the idea of adding more individuality to the workers or to the toys that you saw, other than different graphics for each.

STUDENT 1: So some of the worst things that we did on our team is that we spent a lot of time on work that got thrown out entirely. All the prototypes we did, they were actually pretty useful because of the things we learned about the concept and about how people would play the game. But we did spend a lot of time on things like art or nitty gritty details that didn't really need to be figured out and that we could have put off until later in the project.

We also kept the game direction to vague for too long. As Norman said, we spent a lot of time with that, and it probably ate up too much of our time. Although it helped us learn, we could've moved faster in the beginning to get to a solid idea. Because once we got to a solid idea of the rising waters, our team started to centralize around a lot better because we could actually deal with something concrete. While we were dealing with the abstract ideas, everybody was all over the place and arguing about things that didn't quite line up.

And the worst decision of all that we started with, and that sort of made the problem of going too vague and all these other things happen, is that we were originally thinking how do we skin forecast-based financing as a game? How do we take this idea of using forecasting decisions and then just gamify it? Which we eventually realized wasn't fun, and didn't help us actually come up with any ideas. Instead, when we flipped it and started talking about what game could we create and use forecast-based financing to improve it and teach players how to play the game, and therefore allow them to come out of the game having learned about forecast-based financing, the world sort of opened up. Everything became a lot more interesting and we found that we started to move faster.

Some of the best decisions that we made, on the other hand. As Norman said, we had good tools, which meant that even when we threw out prototypes, although we wasted things like art resources and things like that, we actually didn't end up wasting very much code, because things like the idea of workers or cities could literally be pulled out of the old games we had, put into our new game, and then reworked to build our new structure.

We also weren't afraid to trust each other and throw out the things that didn't work. Once we started moving fast, we had a lot of ideas that would come out, and we would say, OK, this doesn't work. We're actually going to scrap it. Or we think that this isn't the direction we need to go in. And everybody was willing to go along with it. It's not a good feeling to see your things thrown out, but everybody understood that was for the best of the game, and I really appreciate their understanding with everything too.

And part of that all comes down to the fact that we were on board with our final idea. We were all excited about the concept that we had. Part of that might have been the relief of coming to a concrete idea after spending so much time being vague. But once we had that concrete idea, we really moved fast and worked well around it.

So thank you, and any questions?

[APPLAUSE]

AUDIENCE: Who did your sound? It's awesome.

STUDENT 1: That was from our UI team.

AUDIENCE: Oh. OK. It's really cool.

PROFESSOR: Where did you find it?

STUDENT 2: So it's on our credits page. Most of it was online. The credits page in our game.

[LAUGHTER]

STUDENT 1: It's a little small, I guess, up there.

PROFESSOR: Oh, OK.

STUDENT 1: But it looks--

PROFESSOR: [INAUDIBLE].

STUDENT 2: Yeah. "Hold My Hand," AJP, by--

AUDIENCE: Would it be possible maybe to create a short URL for this? For the game?

STUDENT 2: Yeah, a bit.ly link?

STUDENT 1: Yeah, sure. We can make a bit.ly link, and we'll send it out to everyone.

AUDIENCE: Yeah, just so we've go it. Yeah.

STUDENT 1: That's a good call. Yes.

AUDIENCE: Having watched the early crash and burn of the playthrough, how often have people-- I don't know if you've actually had a lot of people to play your current version of your game-- do people usually take a playthrough or two before they start getting the concept?

STUDENT 1: Yes. That's why something like a tutorial would be really nice. Unfortunately, we didn't have the time to put it in--

AUDIENCE: Yeah, no, no. I was just wondering how that--

STUDENT 1: Yeah. Usually what happens is even if after maybe a couple turns of playing through, they start to get the idea. The problem is that as the water starts to rise, they haven't prepared enough, and so all their toys will starve or get carried away by the waves, which is a bit unfortunate and probably makes the players feel bad on the first time. But then they-- It actually teaches them to think ahead about it. So the next playthrough, they're much more careful and understanding.

AUDIENCE: So I was wondering if you were playing any other games or thinking about other games as inspiration or thinking about how to deal with some of-- hitting the right level of strategic thinking in your game.

STUDENT 2: Well, so I guess in terms of early on, because we had a very different idea of what we wanted to do it early on in the process. We were thinking about games like Civilization and how did they communicate all these complex worker movements and managing multiple cities. But once we actually came up with this game concept, I think we had much, much smaller goals,

and so did we have any specific game models, do you think, or?

STUDENT 1: There's none that I specifically think of. There were some things that we were sort of inspired by, standard tricks like when you hover over one of the characters, they got bigger. Some idea of showing off that this is clickable, things like that. But specific games themselves, not really.

AUDIENCE: OK. I was just thinking it ended up being kind of board-game-y. And I feel like there's a lot of board games where they've been thinking a lot about getting that right level of tactics rather than strategy. But yeah. I guess it worked out too.

AUDIENCE: So you mentioned that you were able to change how your workers looked and just keep your models. So that's the holy grail of object-oriented programming, that you have an object that's reusable and you don't have to throw out the code. Do you think there's a reason why in particular you were able to achieve that? Because I think that's not common, necessarily, in object-oriented programming.

STUDENT 2: Partially a little bit of OCD-ness, like, very early on, very strictly saying we're going to write this object-oriented code, and we're not just going to hack things together. I think that helped a lot, because we actually spent the time in the very beginning to think about, like, which objects were responsible for what and what their purpose should be. So basically, I think because we moved more slowly in the start and thought more carefully about how that code should be structured, we ended up with having an easier time later on.

STUDENT 1: There's also the fact that because we had learned these things from the prototype that we were putting into this game, that also meant that the objects that we created-- because we wanted similar functionality to things that we had already seen and we knew worked, it meant that we were comfortable pulling out the functionality into that. So I don't want to say it was designed to fit, but there was the fact that we moved it on purpose, really.

AUDIENCE: I would like to something on that. So we had a very good MVC model. So models were in this tree-like structure, and it was very easy to change models. So models knew about-- Sorry. User knew about models, but models had no idea about [INAUDIBLE]. So basically, it was quite easy.

PROFESSOR: Thank you.

STUDENT 1: Thank you.

[APPLAUSE]