An Overview of
Onion Future,
with additional notes on the development process, pitching, and my own personal thoughts on quickly prototyping at the beginning of the project.

Intro

Onion Future is a pitch project to develop a vertical slice of an RPG for the Playdate. The player is a fox, who receives a letter from the king, and must go on a journey to meet them. The slice only gets to the part where you leave the village though. The Playdate is a pretty small handheld console. It's got a d-pad, a and b buttons, and a crank. The crank is on the side, it's essentially a rotary knob. You can get its position, and its movement. The goal of the project was for the RPG to use the crank throughout its structure, and to be as natural as possible. The Playdate also has an accelerometer.

The Pitch

Our team was made up of 2 artists (Nina and AU), 2 programmers (Collin and Derek), 1 writer (Ean), and 1 producer/sound designer (myself aka Michael). The pitch basically formed during BVW, when myself, Nina, and Derek were sitting around my desk, and said "should we pitch next semester?" and then said "no probably not", and then someone, I can't even remember who, pulled up the Playdate website, and then the three of us got obsessed with making a game for it. Ean overheard us talking, and he got in on it. Then we needed some second years. I already knew Collin from writing a song for Dust to Dawn, so I asked him first, he was thinking of a different pitch so he was a maybe. Then I asked Mia, an artist, who couldn't do it, but introduced us to AU, who agreed. Then Collin agreed, so we had a full team.

Most of our time thinking about the pitch was focused on two things. Trying to make it seem like we weren't overscoping, and justifying a "game" by talking about the uniqueness of the platform. I have no idea how well either of those things worked, but our pitch made it through, so I guess well enough.

The Project

One advantage our project had starting out was that we didn't really have a "planning phase". Or rather, we didn't have any time dedicated to *only* planning. Because we had been thinking of the game for a couple months during BVW, we already had a good idea of what we wanted to do. Derek had already implemented the dialogue system based on something he had done earlier in HTML for his personal website. We also had some fun visual text effects in there too. Anyways, the point is that by the second week, we had already prototyped most of the basic movement and map development systems.

The other great thing was that we had 4 Playdates. We submitted a request to purchase them the moment our pitch went through, which was important because they were on backorder, and even though they were ordered before the pitch even started, they still only arrived in like the third week. But with the actual physical device we got a lot of inspiration. Ideating ideas for using the crank was mostly a matter of going, "Hmmm, everyone just say what you could do with this," then writing down a list. In I think the 4th or 5th week Derek did a kind of game jam where he just implemented 2 of those list items over a weekend: Digging and alchemy. So the process was basically that we implemented those ideas, and then built the rest of the structure around them.

The other important thing is that, while the Playdate has a core library, it doesn't have a visual editor. The core library handled hardware interaction things like displaying things to the screen and getting the crank's position, and then we wrote everything in visual studio code. This had the major benefit that we knew everything going into the game, so we had less conflicts with weird quirks of an engine. It did mean we had to start at the real basics. Like, manually progressing through sprite sheets frames for animation. Luckily, our programming team was especially good at creating systems that were modular and flexible, so we wouldn't get stuck on anything we had implemented later in the semester.

In general, I would say that, as a first time producer, our biggest challenge was organization. Our team was very well organized, don't get me wrong, but I think it was only because everyone was enthusiastic about the project, and also sharing their progress and current status frequently. Like, I would have struggled so much more if that wasn't the case, so I'm really happy about that. Because with that, it felt like we always moved as one unit, which for a project like this where it's all about developing quickly, you really need to. Oh, also I think we should have playtested earlier. That one's on me, I just never had the energy to go to the playtests in Hunt Library on Tuesdays because the way I scheduled our core hours usually meant we

were in from 11am to 6pm, and then Derek always wanted to keep working until 7-8, and because I carpool I couldn't leave without him. But whatever, there are only so many ways to schedule the required hours around everyone's electives. We got a lot of feedback during Playtest day that I think really made our project actually good in the end, so I just wonder if it might have been better earlier. Although who knows, maybe it wasn't ready to receive that feedback earlier.

Lessons to Learn

Our advisor, Chris, made us send him weekly reports that consisted of "what did I do this week, what are my blockers, and what do I plan to do next week". Our weekly advisor meetings also were in the form of a standup, where we all just put screenshots of our work in google slides, and went through them one-by-one. I absolutely would not have done either of those things if he hadn't made us, but they ended up being super helpful. Like for both writing the blog posts and also just keeping organized. We didn't even do standups outside of that. I just used that weekly advisor meeting as our team's weekly standup. It worked out pretty well. As far as I know, Chris might be alone in doing things like that? So I'd just recommend having some kind of weekly system where you just put down the things you've done in some kind of record. In our case, it was just capturing the week's work in the google slides.

The other thing is that it's always better to try things than to spend more than one discussion debating over how something should be done. Just like, do it.

Also also, we spent basically all our time after halves just building out content, so our game ended up feeling pretty "complete". The only reason we were able to do that is because we spent so much time early in the semester building the fundamental systems that our game would run on. Even though they didn't look as impressive, it helped in the end so much to make sure that we weren't bodging things together, and that our systems would *help* us when we went to use them later. I'm especially thinking of our tile maps system. We separated the tilemaps into 4 arrays in a JSON file, so we could have separate walls, base layers, and decorations for the artists to work with. And then that data just worked, no matter what the map wanted to look like. That kind of thing.

Conclusion

Anyways, I'm kind of just rambling, and I have no idea if anyone is going to read this. I also have only ever done this once, and I was on a team of people that I basically hand-picked, and that ended up living up to every expectation I had. So who knows how helpful any of my advice is. I was pretty lucky. But the advice I always give, and which worked out for this project, is to have a playable version of your *main* mechanic in like, a week. It's helpful to be able to say, "our game's main mechanic is this". And when you say this, it shouldn't just be like "oh, organizing a store", or "a bullet hell". You should be able to say exactly what actions the player will perform. Like, "using the dpad to move, they will dodge waves of projectiles being shot at them" or "using a to pick up items presented on the side of the screen, they will use the dpad and b button to move and rotate the items into a shop front, which will be a grid that items take up a specific space in". That way, you can figure out what is the core thing a player will be doing, and build a prototype for *that.* Not only does that let you see if the actual actions the player will perform are fun, it also gives you ideas that you could only get by playing around with what your game *is.*

Also I have no idea what your game might be, but that prototype should ideally be fast. Like, "I don't care about the details I built this in 3 days" kind of fast. Just figure out what it is in your game that you can make in a short time frame, and make it.

Good luck out there,
Michael