Advice to Future Students of CS 679

Planning Things Out

Our group spent a lot of time planning. We really just sat in a room for 4–5 hours a number of times. This sounds excessive at first, but when you're coming up with an idea for a game that will have this much time spent developing it, it's important to emphasize that details matter! Everyone has a different interpretation of how the game will look, feel, and play and it's so important to stress that everyone needs to be on the same page. It's not fun to have people go off and implement their pieces only to come together and realize you had very different ideas about how the game would actually work.

Polish Your Pitch

Spend a lot of time on your pitch. It might seem like an easy assignment to spend an hour on and get out of the way, but this is going to be a project you'll be working on for the next month of your life. Don't expect other people to come up with the great idea, because if you don't like any of the pitches, you're a little out of luck. Hash out what the player will see, experience, and do. Be specific. It's easy to be vague. You need to actually understand your game to be specific.

Dividing Things Up

Take the time to do planning documents about architecture and division of labor. It's easy to just fly by the seat of your pants and see what happens. One person does some work, the next mucks with it, etc. It's much more efficient (Not to mention good practice) to sit down and chart out data flow through the program. What happens when they hit the landing page, what parameters do each component of your game need? Where does this information come from? How do things like levels, turns, different players, etc. work? These are things that are easy to skip over and can end up biting you. Potentially one of the best parts of this is everyone gets to see where the code lies. Each person can look at the diagram to figure out what code is where to fix a bug or add a feature.

Features!

Become feature complete as soon as possible. The sooner you can get your game running with all the mechanics you want (Read: not technical/shiny things, just gameplay) the sooner you can realize what's wrong with your design. If you're able to iterate 2–3 times on your game, you'll be able to iron out those obvious issues that someone playing will see that isn't immediately clear to you while implementing and designing. The sooner you can sit and say here's what works and here's what doesn't, the better.

Consider All Your Technology Options

Don't write off web technologies just because you prefer a language like C# or C++. It's easy to underestimate how difficult things like user interface can be in lower level languages, while JavaScript has tons of libraries around like jQuery that make nice looking UI extremely easy to come by. A full screen game in C++ with effective UI (Especially if it requires mouse input onto buttons or menus) is extremely time consuming when compared to web technologies.