Prototyping Tip #6: It Doesn't Have to be Digital

Your goal is to loop as usefully and as frequently as possible. So, if you can manage it, why not just get the software out of the way? If you are clever, you can prototype your fancy videogame idea as a simple board game, or what we sometimes call a **paper prototype**. Why do this? Because you can make board games *fast*, and often capture the same gameplay. This lets you spot problems sooner — much of the process of prototyping is about looking for problems, and figuring out how to fix them, so paper prototyping can be a real time saver. If your game is turn-based to start with, this becomes easy. The turn-based combat system for Toontown Online was prototyped through a simple board game, which let us carefully balance the many types of attacks and combos. We would keep track of hit points on paper or on a whiteboard, and play again and again, adding and subtracting rules until the game seemed balanced enough to try coding up.

Even real-time games can be played as paper prototypes. Sometimes they can be converted to a turn-based mode that still manages to capture the gameplay. Other times, you can just play them in real-time, or nearly. The best way to do it is to have other people help you. We'll consider two examples.

Tetris: A Paper Prototype

Let's say you wanted to make a paper prototype of *Tetris*. You could cut out little cardboard pieces, and put them in a pile. Get someone else to draw them at random, and start sliding them down the "board" (a sketch you've drawn on a piece of paper), while you grab them, and try to rotate them into place. To complete a line, you have to just use your imagination, or pause the game while you cut the pieces with an X-acto knife. This would not be the perfect Tetris experience, but it might be close enough for you to start to see if you had the right kinds of shapes, and also enough to give you some sense of how fast the pieces should drop. And you could get the whole thing going in about 15 minutes.

Doom: A Paper Prototype

Would it be possible to make a paper prototype of a first person shooter? Sure! You need different people to play the different AI characters as well as different players. Draw out the map on a big piece of graph paper, and get little game pieces to represent the different players and monsters. You need one person to control each

of the players and one for each of the monsters. You could then either make some turn-based rules about how to move and shoot, or get yourself a metronome! It is easy to find free metronome software online. Configure your metronome to tick once every five seconds, and make a rule that you can move one square of graph paper with every tick. When there is a line of sight, you can take a shot at another player or monster, but only one shot per tick. This will give the feeling of playing the whole thing in slow motion, but that can be a good thing, because it gives you time to think about what is working and not working while you are playing the game. You can get a great sense of how big your map should be, the shapes of hallways and rooms that make for an interesting game, the properties your weapons should have, and many other things — and you can do it all lightning fast!

Prototyping Tip #7: Pick a "Fast Loop" Game Engine

The traditional method of software development is kind of like baking bread:

1. Write code
2. Compile and link
3. Run your game
4. Navigate through your game to the part you want to test
5. Test it out
6. Go back to step 1

If you don't like the bread (your test results), there is no choice but to start the whole process over again. It takes way too long, especially for a large game. By choosing an engine with the right kind of scripting system, you can make changes to your code while the game is running. This makes things more like working with clay — you can change them continuously:

1. Run your game
2. Navigate through your game to the part you want to test
3. Test it out
4. Write code
5. Go back to step 3

By recoding your system while it is running, you can get in more loops per day, and the quality of your game goes up commensurately. I have used Scheme, Smalltalk, and Python for this in the past (I'm a big fan of Panda3D: www.panda3d.com), but any late-binding language will do the job. If you are afraid that these kinds of languages run too slowly, remember that it is okay to write your games with more

than one kind of code: write the low-level stuff that doesn't need to change much in something fast but static (Assembly, C++, etc.), and write the high-level stuff in something slower but dynamic. This may take some technical work to pull off, but it is worth it because it lets you take advantage of the Rule of the Loop.

Prototyping Tip #8: Build the Toy First

Back in Chapter 3, we distinguished between toys and games. Toys are fun to play with for their own sake. In contrast, games have goals and are a much richer experience based around problem solving. We should never forget, though, that many games are built on top of toys. A ball is a toy, but baseball is a game. A little avatar that runs and jumps is a toy, but Donkey Kong is a game. You should make sure that your toy is fun to play with before you design a game around it. You might find that once you actually build your toy, you are surprised by what makes it fun, and whole new ideas for games might become apparent to you.

Game designer David Jones says that when designing the game *Lemmings*, his team followed exactly this method. They thought it would be fun to make a little world with lots of little creatures walking around doing different things. They weren't sure what the game would be, but the world sounded fun, so they built it. Once they could actually play with the "toy," they started talking seriously about what kinds of games could be built around it. Jones tells a similar story about the development of *Grand Theft Auto*: "Grand Theft Auto was not designed as Grand Theft Auto. It was designed as a medium. It was designed to be a living, breathing city that was fun to play." Once the "medium" was developed, and the team could see that it was a fun toy, they had to decide what game to build with it. They realized the city was like a maze, so they borrowed maze game mechanics from something they knew was good. Jones explains: "GTA came from Pac-Man. The dots are the little people. There's me in my little, yellow car. And the ghosts are policemen."

By building the toy first, and then coming up with the game, you can radically increase the quality of your game, because it will be fun on two levels. Further, if the gameplay you create is based on the parts of the toy that are the most fun, the two levels will be supporting each other in the strongest way possible. Game designers often forget to consider the toy perspective. To help us remember, we'll make it Lens #15.