Level 2: Game Design / Iteration and Rapid Prototyping

Originally posted July 2, 2009

Last time we asked the question: what is a game? Today, we look into a related question: what, exactly, is game design? Last time, we made a simple game. This time, we will look into the process of how games are made in general. While it is possible to make a race-to-the-end board game in 15 minutes, you will need to take a little longer if you are looking to make the next ***Settlers of Catan*** or ***World of Warcraft***.

Game Design

We will use the word "design" a lot in this course, and unfortunately it is a term that is a bit overused, so I will clarify what I mean here. As it says in *Challenges*, game design is the creation of the rules and content of a game. It **does not** involve programming, art or animation, or marketing, or any of the other myriad tasks required to make a game. All of these tasks collectively can be called "game development" and game design is one part of development.

Unfortunately, I have seen the term "design" used (particularly in some college curricula) to refer to all aspects of development. When used in the video game industry (or the board game industry), "game design" has a very specific meaning, and that is the meaning that we will use for this course.

Multiple Types of Game Design

As mentioned in *Challenges*, there are many tasks associated with game design: system design, level design, content design, user interface design, world building, and story writing. You could fill several 10-week courses with any one of these, so this summer course will not be a full treatment of the entire range of game design. We will touch lightly on UI, story writing and content when relevant, but the majority of this course focuses on **system design** (also sometimes called "systems design" or "core systems design").

System design is about defining the basic rules of the game. What are the pieces? What can you control? What actions can you take on your turn (if there are "turns" at all)? What happens when you take each action, and how does it affect the game state? In general, system design is the creation of three things:

» Rules for setup. How does the game begin?

» Rules for progression of play. Once the game begins, what can the players do, and what happens when they do things?

» Rules for resolution. What, if anything, causes the game to end? If the game has an outcome (such as winning or losing), how is that outcome determined?

If you look back at ***Three-to-Fifteen*** from Level 1, you'll notice those very simple rules contain all of these parts. The creation of those rules is system design, and that is what we will be spending most of our time with over this summer.

What is a Game Designer?

As you may have noticed, game design is an incredibly broad field. Those of us who are professional designers sometimes have trouble explaining what we do to our families and friends. Part of the reason for this is that we do so many things. Here are some analogies I've seen when trying to explain what it is like to be a game designer:

» Game designers are **artists**. The term "art" is just as difficult to define as the word "game"… but if games can be a form of art (as we saw in Costikyan's definition, at least), then designers would be artists.

» Game designers are **architects**. Architects do not build physical structures; they create blueprints. Video game designers also create "blueprints" which are referred to as "design docs." Board game designers create "blueprints" as well — in the form of prototypes — which are then mass-produced by publishers.

» Game designers are **party hosts**. As designers, we invite players into our space and try our best to show them a good time.

» Game designers are **research scientists**. As I will touch on later today, we create games in a manner that is very close to the scientific method.

» Game designers are **gods**. We create worlds, and we create the physical rules that govern those worlds.

» Game designers are **lawyers**. We create a set of rules that others must follow.

» Game designers are **educators**. As we will see later when we start reading *Theory of Fun*, entertainment and education are strongly linked, and games are (at least sometimes) fun because they involve learning new skills.

If game design is all these things, where would it fit in a college curriculum? It could be justified in the school of education, or art, or architecture, or theology, or recreation management, or law, or engineering, or applied sciences, or half a dozen other things.

Is a game designer all of these things? None of them? It is open for discussion, but I think that game design has *elements* of many other fields, but it is still its own field. And you can see just how broad the field is! As the field of game design advances, we may see a day where game designers are so specialized that "game design" will be like the field of "science" — students will need to pick a specialty (Chemistry, Biology, Physics, etc.) rather than just "majoring in Science."

Speaking of Science…

How is a game designed? There are many methods.

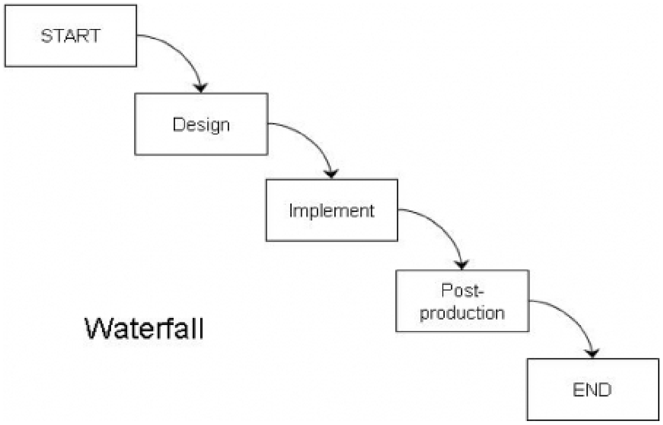

Figure 1 - The Waterfall Method

Historically, the first design methodology was known as the **waterfall** method: first you design the entire game on paper, then you implement it (using programming in a video game, or creating the board and pieces for a non-digital game), then you test it to make sure the rules work properly, add some graphical polish to make it look nice, and then you ship it.

Waterfall is so named because, like water in a waterfall, you can only move in one direction. If you're busy making the final art for the game and it occurs to you that one of the rules needs to change, too bad — the methodology does not include a way to go back to the design step once you are done.

At some point, someone figured out that it might be a good idea to at least have the *option* of going back and fixing things in earlier steps, and created what is sometimes known as the **iterative** approach.

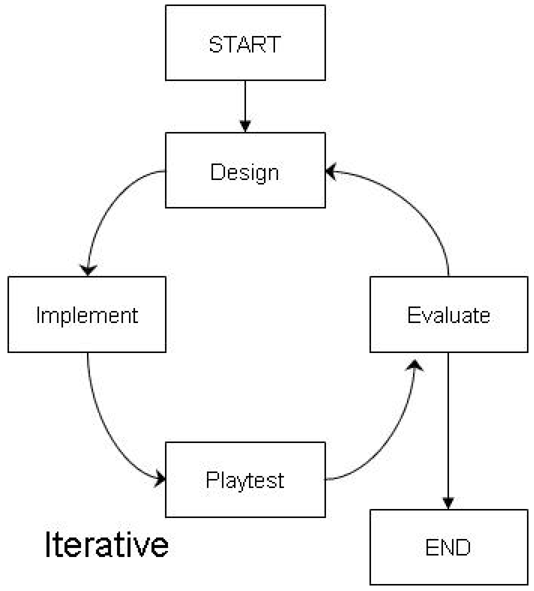

Figure 2 - The Iterative Approach

As with waterfall, you first design the game, then implement it, and then make sure it works. But after this you add an extra step of evaluating the game. Play it, decide what is good and what needs to change. And then, make a decision: are you done, or should you go back to the design step and make some changes? If you decide the game is good enough, then that is that. But if you identify some changes, you now go back to the design step, find ways to address the identified problems, implement those changes, and then evaluate again. Continue doing this until the game is ready.

If this sounds familiar, it is because this is more or less the Scientific Method:

1. Make an observation. ("My experience in playing/making games has shown me that certain types of mechanics are fun.")
2. Make a hypothesis. ("I think that this particular set of rules I am writing will make a fun game.")
3. Create an experiment to prove or disprove the hypothesis. ("Let's organize a playtest of this game and see if it is fun or not.")
4. Perform the experiment. ("Let's play!")
5. Interpret the results of the experiment, forming a new set of observations. Go back to the first step.

With non-digital (card and board) games, this process works fine, because it can be done quickly. With video games, there is still one problem: implementation (i.e. programming and debugging) is expensive and takes a long time. If it takes 18 months to code the game the first time and you only have two years, you will not get a lot of time to playtest and modify the game.

In general, **the more times you iterate, the better your final game will be**.

Therefore, any game design process should involve iterating (that is, going through an entire cycle of designing, implementing and evaluating) as much as possible, and anything you can do that lets you iterate faster will usually lead to a better game in the end. Because of this, video game designers will often prototype on paper first, and then only get the programmers involved when they are confident that the core rules are fun. We call this **rapid prototyping**.

Iteration and Risk

Games have many kinds of risk associated with them. There is **design risk**, the risk that the game will not be fun and people won't like it. There is **implementation risk**, the possibility that the development team will not be able to build the game at all, even if the rules are solid. There is **market risk**, the chance that the game will be wonderful and no one will buy it anyway. And so on.

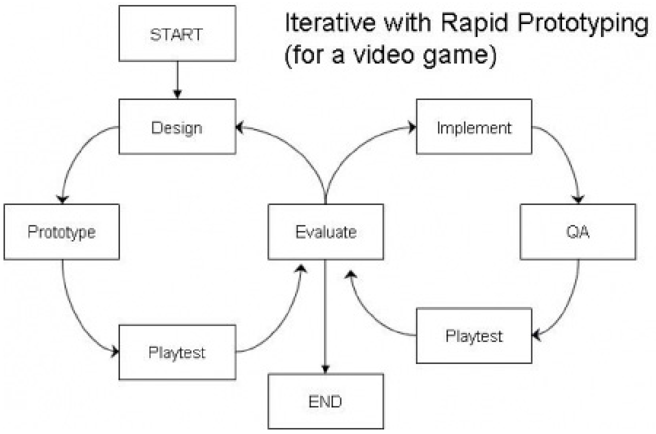

Figure 3 - The Iterative Approach with Rapid Prototyping

The purpose of iteration is to lower design risk. The more times you iterate, the more you can be certain that the rules of your game are effective.

This all comes down to one important point: the greater the design risk of your game (that is, if your rules are untested and unproven), the more you need iteration. An iterative method is not as critical for games where the mechanics are largely lifted from another successful game; sequels and expansion sets to popular games are examples of situations where a Waterfall approach may work fine.

That said, most game designers have aspirations of making games that are new, creative, and innovative.

Why This Course is Non-Digital…

1Some of you would rather make board games anyway, so you don't care how video games are made. But for those of you who would love to make video games, you may have wondered why we will be spending so much time making board and card games in this course. Now you know: it is because iteration is faster and cheaper with cardboard. Remember from Monday: you can make a board game in 15 minutes. Coding that game will take significantly longer. When possible, prototype on paper first, because a 15-minute paper prototype and an hour-long playtest session can save you months of programming work.

Later in this course, we will discuss in detail methods of paper prototyping, both for traditional board games and also for various types of video games.

There is another reason why we will concentrate primarily on non-digital games this summer, particularly board and card games. This is a course in **systems design**, that is, creating the rules of the game. In board games, the rules are laid bare. There may be some physical components, sure, but the play experience is almost entirely determined by the rules and the player interactions. If the rules are not compelling, the game will not be fun, so working in this medium makes a clear connection between the rules and the player experience.

This is not as true in video games. Many video games have impressive technology (such as realistic physics engines) and graphics and sound, which can obscure the fact that the gameplay is stale. Video games also take much longer to make (due to programming and art/audio asset creation), making them an impractical choice for a ten-week course.

The connection between rules and player experience is also muddied in tabletop role-playing games. I realize that many of you have expressed an interest primarily in RPG design, so this may seem strange to you. However, keep in mind that an RPG is essentially a collaborative story-telling exercise (with a rules system in place to set boundaries for what can and can't happen). As such, a wonderful system can be ruined by players who have poor story-telling and improv skills, and a weak system can be salvaged by skillful players. As such, we will stay away from these game genres, at least in the early stages.

Trying it out

Take that 15-minute game you made last time, and play it, if you haven't already. As you are playing, ask yourself: is this more fun or less fun than playing your favorite published games? Why? What could you change about your game to make it better? You do not have to play the game to completion, but only for as long as it takes you to get the overall feeling of what it is like to play.

Then, after playing once, make at least one change. Maybe you'll change the rules for movement, or add a new way for players to interact. Maybe you'll change some of the spaces on the board. Whatever you do, for whatever reason, make a change and then play again. Note the differences. Has the change made the game better, or worse? Has this one change made you think of additional changes you could make? If the game got worse, would you just change the rule back, or would you change it again in a different way?

We will be looking at the playtest process in detail later in this course. For now, I just want everyone to get over that fear: "what if I play my game and it sucks?" With the game you designed in Level 1, the odds are very high that your game *does* suck (seriously, did you expect to make the next ***Gears of War*** in 15 minutes?). This does not make you a "bad designer" by any means — but it should make it clear that the more time you put into a game and the more iterations you make, the better it gets.

Lessons Learned

The one big takeaway from today is that the more you iterate on a game, the better it becomes. Great designers do not design great games. They usually design *really bad* games, and then they iterate on them until the games *become* great.

This has two corollaries:

» You want to have a *playable* prototype of your game as early in development as possible. The faster you can playtest your ideas, the more time you have to make changes.

» Given equal amounts of time, a shorter, simpler game will give a better experience than a longer, complicated game. A game that takes ten hours to play to completion will give you fewer iterations than a game that can be played in five minutes. When we start on the Design Project later in this course, keep this in mind.

Level 2 - Homeplay

Before next Monday, read the following. I will be referencing these in Monday's content when we talk about the formal elements of games:

» *Challenges for Game Designers*, Chapter 2 (Atoms). This will act as a bridge between last Monday when we talked about a critical vocabulary, and next Monday when we will start breaking down the concept of a "game" into its component parts.

» *Formal Abstract Design Tools* (available at `http://www.gamasutra.com/features/19990716/design_tools_01.htm`), by Doug Church. This article builds on Costikyan's *I Have No Words*, offering some additional tools by which we can analyze and design games. While he does use many examples from video games, think about how the core concepts in the article can apply to other kinds of games as well.