Written by Billy D. Spelchan for www.BlazingGames.com

Chapter 2

The Game Development Cycle

Contents

The development cycle for a game varies based on who is developing the game and the particular challenges of the game being designed. For my own games, I break the game development cycle into seven stages. Smaller games may not need all the stages. Some games may have break stages into multiple sub stages in order to keep milestones relatively evenly spaced. As most readers know, a milestone is a set of tasks that are to be completed by a set time.

Initial Concept

At this stage you work out exactly what it is that you want to develop. I write a brief overview of the game which consists of an overview of the story and the game mechanics. For a simple game this description can be a single paragraph. For more complex games this can be a few pages.

The nice thing about this stage is it does not require much time or effort. For that reason you can write concept documents for a large number of games with little cost. As it is a good starting point for a new project, it may be a good idea to write a concept document for any good ideas you come up with, even if you are currently working on another project.

Before proceeding past the initial concept phase it is wise to take a realistic look at your current situation, how the game is going to be released, and time constraints on the development cycle. On my game site (currently at www.BlazingGames.com , though this could change so if you want to visit my games site you are best to use www.Spelchan.com which will redirect you to my current game site [or will at least have a link to my current game site if I decide to do something with the site]). I am currently releasing new content on my site every week.

Having a weekly content requirement does affect game design. The big concern that I have when planning a game is how long the development is going to take versus how many episodes that game is going to generate. Ideally I want to have games that only take an average of three days per episode. Why? To make money I have to do third party work, so the amount of time I have to work on my site varies based on what contracts I have. While currently I have a lot of free time to work on this site, that can (and hopefully will) change. By having three days to put together an episode, in a worst-case situation I can use my weekends to finish a game.

Prototyping

The prototyping stage is an optional stage and can be placed before or after the design specifications stage, and in many cases may be concurrent with the creation of the design specification. The goal behind prototyping is to prove that some critical code or play mechanism will actually work. The prototype then is a piece of code quickly written to test some aspect of the final game. Usually the code for the prototype is very rough code without any thoughts to usability in mind when it is being created. Many companies will scrap the prototype code entirely, while other companies simply go through the prototype code and enhance it. I go for a mixed approach. Writing new code, but cutting and pasting code from the original prototype wherever possible.

The key to prototyping is deciding if a prototype is needed for a project, and if so what needs to be prototyped. My thoughts on prototyping is that it should primarily be used when you are dealing with a new technology, with a new technique for accomplishing a task, or when you have to validate performance.

By new technology, I do not necessarily mean new hardware, though hardware would certainly qualify. For instance, if you have never worked on a Game boy Advance™ (Trademark of Nintendo), and were writing a Game boy Advance game, doing some prototypes to find out if what you wanted to do was possible and what complications and restrictions the particular platform has would be a good idea. At the same time, new software, languages, or libraries may also warrant writing a prototype. For instance, my Dragon Hunt game is actually a prototype to test Flash and how well Flash would be at handling an adventure game. The version released on my site is not the original prototype as the original prototype was testing loading movies as rooms.

New techniques is fairly self explanatory. Any time you are using an algorithm which you have never used before, prototyping may be in order. For instance, if you developing a game that needed path finding, and had never used the A* (A Star) algorithm, writing an A* prototype might not be a bad idea.

Validating performance is trickier thing to explain. The goal here is to make sure some aspect of the game engine will work within some set of requirements. For instance, if you are writing a 3D engine for a game, you may want to make sure that the engine will be able to put out enough polygons while still sustaining a high enough frame rate. Perhaps the frame rate isn't important but you require the image quality be high enough. Some other examples would be to make sure the memory requirements be low enough, to make sure the network bandwidth requirements will be low enough, or to make sure elements of the game play be compelling enough.

Commercial Prototyping

One area where prototyping is vital is when trying to get a third party to fund or distribute a game you are developing. Many companies, in fact, will require that a proposal have some type of prototype. In these cases, the prototype is a much more elaborate prototype and is commonly a playable portion or level of the game. This level of prototyping is almost certain if the developers are unknown to the third party.

The purpose of such an elaborate prototype is usually to prove two things to the third party. First, that the game is possible to create. By having a playable level, the third party can get an idea of what the final game will be like. What elements of the game that should be in the prototype will largely depend on whether the third party is impressed by glitz or if they want game play.

A more important reason a third party would want a prototype would be to see if in fact the developers are able to put together a functioning program. You have to understand that developing a game is a lot more difficult than most people believe.

Design Specifications

One important thing to remember when designing larger games is the need for flexibility in the design. The design specifications should be considered to be a work in progress. Some companies like to have minimalistic design specifications. Others try to cover every aspect of the game. I personally want a design specification that has enough detail to show what the final product should be, yet enough flexibility to change the product if necessary. There is no standard format for design specifications. What follows is the format that I personally use.

I generally start my document off with an overview of the project. This will quite often be a rehash of the initial concept document (if such a document exists). This overview is used to quickly establish what the project is all about and get the readers all on the same page.

Next I go into a game play overview. This describes how the game would be played and covers many of the user interface elements of the game. Rough layouts help explain the interface. My layouts generally consist of boxes with text describing what goes in the box, though I have seen design specifications where game screen shots are mocked up. While this makes the document nicer to look at, unless you are writing the document for a third party I don't think the required time to do fully rendered screen mockups is needed.

At this point that the reader knows how the game plays, so having a description of the games rules is the next order of business. Rules for a game should cover both the rules that the player knows about as well as rules that govern the game. For instance, in a role-playing game you would have rules on aspects that the player sees, such as equipping of items, as well as rules on things the player may not see, such as how hits and damage is calculated in combat or how monster moral is controlled. In addition to the game rules, you may have additional sections on other game specific aspects. For instance, you may have a section on how network play would be handled, or a section containing biographies of characters in the game. These sections, of course, are game specific.

Next I describe the content of the game. This would include a story overview, and could also contain a level by level breakdown. How much of the games content would be described in the design specifications is largely dependent on the type of game that is being designed.

I then conclude with a breakdown of the assets used in the game. An asset is simply an image, music file, sound file, text file, or other data file that has to be created for the game. Knowing all the assets that will need to be created can really help scheduling the game. Having a checklist of assets that need to be created is useful by itself. Some design specifications may have scheduling, financial, marketing and other sections that managers like to see.

Design Debate

Far too often new game developers don't bother with any type of design stage and just jump right into a project. By knowing what exactly you are going to do and having a rough idea of what tasks must be done you can better estimate how long it will take to complete a project. The design specifications also help the development process as a roadmap of what has to be done when. This allows for the developers to stay focused on what has to be done. It also allows the developers to streamline the amount of content that has to be created.

Fewer problems programming the game is another benefit of a plan. By knowing exactly what a game is required to do, and just as importantly what a game does not require, it is easier to plan out the programming. Some companies will use the design specifications to write a programming specifications document which breaks down the program and outlines data structures and other programming guidelines. Unless the project is sufficiently large and will require more than a few programmers, I do not feel that this is a necessary step.

While small projects may not need too much planning (some would argue that it doesn't need any at all), I have found that it is good to do as it keeps you in the habit of planning projects. It also has the benefit of helping you better determine how long different tasks will take as you will be able to compare your estimated time with your actual time. As smaller games obviously can be completed quicker, you will gain experience with scheduling much quicker.

Milestones

While alpha, beta, gold are common ways of determining the state of a project, large projects tend to be broken into a large number of milestones. A milestone is whatever the publisher and the developer decide they are. Milestones tend to be feature lists which contain a list of features and the level at which each feature has to be implemented.

Pre-Alpha

Once you start work writing code for the game, not including any prototype code, I consider you to be in a pre-alpha phase. Pre-Alpha continues until you reach an alpha stage. Some companies will have other phases before a pre-alpha stage and consider pre-alpha to have a set level of functionality. Depending on your programming methods and on how the game is going to work, pre-alpha may be a short phase or may be a huge part of the development process.

Alpha

A game is in Alpha once a game has reached the point where it is possible to play the game even though all the features of a game may not be functional. Some companies may define an alpha stage after a set amount of functionality has been reached. Some developers consider alpha to be when internal testing of a fully functional game starts. Other companies may consider alpha to be a set amount of code, while others consider alpha to be what I call Pre-Alpha.

Beta

There is a Big variation on meaning of what Beta is. I consider beta to be code complete. What this means is that all the programming for a game, with the possible exception of scripting language scripts, has been completed and the game is fully playable. This does not mean that the programming work is finished. At this stage bugs may still exist, and code code will have to be changed to fix the bugs. Likewise, during testing, new features may be desired so code will have to be written to support those features. Likewise, by fully playable, I mean all the functionality is written, not all the assets and levels.

Some companies consider beta to mean what I consider to be Alpha. Other companies feel that all assets should be in a functional state (no placeholder graphics in use and all levels to a playable state) before they consider the game to be in beta. Yet other developers consider a game that is being tested by third party testers to be beta.

Gold

A game going gold does not mean that it has sold a large number of copies. Instead gold means that the game is considered ready to be shipped. The term is used because before a product ships or is released, the publisher still has final say over whether a product is ready to be released. Sometimes the publisher may feel that the product is not ready for release so the gold version may be rejected. The opposite (the developer feeling that a product isn't ready to be shipped but the publisher wants to ship anyway) also happens. Because the publisher is usually the one who is spending the money, the publisher almost always gets the final say.

The term gold -- as well as gold master -- comes from the early days of recordable CDs. Back when CDs were just starting to be used as a distribution medium for games, CD burners were very expensive (thousands of dollars). More importantly blank CDs were also expensive. Due to this expense, many developers used a hard drive image of the cd and never actually burned a CD until near the end of development. The blanks in use back then were gold. Hence, when the CD was being burned for final validation, developers started referring to this as going gold. Some companies use a different term for this, such as gamma, or final master, or even white label, but gold still seems to be the common term.

Is Done done?

Now, just because something has been published that does not mean that the program is done. Not all bugs end up being caught, and there still are some publishers who push for release of a product before it is ready. With computers, patching a game is not that big of a deal. Consoles don't have this luxury, so console games tend to be a bit more extensively tested. More importantly, consoles are a locked design, so every unit is the same as every other one. This is not true of computers.