Game Development Methodology for Small Development Teams

By Lorenzo Phillips

Introduction

If you have ever developed software for a company of adequate size, then you are probably familiar with the terminology used within the Software Development discipline. If you are a game developer, then chances are good that you have found it difficult to apply traditional Software Methodologies to your project(s). Even worse, if you are a solo developer or a small band of game developers (i.e., a pack of lone wolves), then you most likely disagree with traditional Software Methodologies (even those used in larger game development shops) and have formulated your own development style. This article was written for the lone wolf developer and the lone wolf packs of the game development community. It is not meant to bash the traditional development methodologies, but merely to provide a working alternative for game developers that do not have the time or the money to implement such complex structures in their projects. In life, there is no one solution that solves every problem. So, simply use the methodology documented in this article as a guide and modify it and cater it to fit your particular situation.

Software Development

Before we get into the actual methodology, let's take a look at why traditional software development techniques do not satisfy the needs of the game community. The traditional style of software development is geared towards business applications. Business applications generally take a minimum of a year to release and work with large budgets. In the gaming community, excess time is not an option. Publishers attempt to release entertainment software on a regular basis throughout the year. Why? In the real world, businesses are not designed to change rapidly. It costs too much money and there is too much overhead involved (i.e., upgrading all of the company's computers, testing the upgrade to ensure it fits into the current environment and can exist with all of the other applications, and so on).

In the real world of the gaming community, there are times when a product can generate high volumes of sales, and therefore, more money. Releasing entertainment software can be thought of like releasing a movie, where timing is a major factor in the success or downfall of the product. Holidays such as Christmas or seasonal times like spring break or summer break when many young adults will be at home and will be more inclined to play games.

Another strategic option that must be given serious consideration is what the competition is releasing simultaneously. For example, if you release a product that uses an older 3D rendering technology while the competition is using a new and improved 3D rendering game engine, then the public may lean towards the game that provides the better graphics. Another major reason why it is so difficult to use the traditional methods is because games tend to stay current with the technological advances, and thus, leaves many unanswered questions. For example, how can you document the development effort against a new graphics card if very few people have ever used it? You may think it will take a couple of weeks, but there is always a learning curve and new bugs to discover that cannot be accounted for in the project plan.

In larger gaming companies there are large development teams consisting of developers, artists, and musicians dedicated to completing the project(s). As a lone wolf developer or a small development team, it is not an option to use complex methods to get the product(s) finished. However, game developers can still leverage some of the techniques and apply them to their specific situations and receive the benefits of the proven and more complex methodologies. So, if you are a lone wolf game developer or a part of a lone wolf pack continue reading and discover how to successfully implement a development methodology that you can use in your game development efforts.

Business Applications vs. Game Applications

I always like to ask the reader this question, "Why do you develop games, instead of business applications?" I'll be honest, I love programming period, but I love to develop games because I feel they are superior to business applications in every way. If you have been doing game development long enough then you will agree that most people in the business community feel that game development is for kids and they have no problem displaying their attitudes. In my opinion, this is a false statement made by people that do not know the first thing about game development. Understand that I'm not disagreeing with the statement because of my passion for game development. I disagree with it because I have programmed in both disciplines and I know it to be an absolute fact that game development is far more advanced than most business application development efforts. Review the following two lists I have compiled for games and business applications and form your own conclusions (if you have not done so already):

GAMES:

- Almost always use Artificial Intelligence (AI) so that the game will think and learn, and thus, become more challenging in later stages
- contain constant real-time interaction that must be blazing fast to keep the interest of the game player
- are optimized to be as fast as possible on any machine configuration (of course this has been aided by the development of the DirectX API)
- use a wide variety of algorithms to achieve eye-popping special effects
- network usage links many game players around the world into one game environment to compete against each other
- graphics are almost always superior to those used in the business community (except in some special cases like military simulations or other disciplines that command high graphical resolutions)
- use a wide range of colors to aid with the realism of the game experience

- make use of 2D/3D sound effects not just the basic bells and whistles
- make use of music (i.e., MIDI and CD) to enhance the product (multimedia or web applications may use music, but the musical scores in games enhance the product dramatically)
- allow artistic creativity (including, but not limited to, algorithms, game engines, 2D/3D graphics, 2D/3D sounds, music and so on)
- can be made to simulate real-life events or fantastic non-existent worlds

BUSINESS APPS:

- are slow in comparison
- little effort goes into optimizing the code
- rarely involve thinking and learning (unless it is a specific app for science, military, or some other related field)
- use colors in the form of themes and blending, which do not attempt to enhance the application experience
- use standard sounds (beeps and whistles)
- rarely make use of music (unless it is a multimedia app or web related app)
- may link people around the world, but an instance of the shared app is created rather than all parties residing in the same location like a Death Match environment
- almost never make use of the new technology that is available (some apps can still run on a 386 machine configuration)
- are just plain boring (when is the last time you saw someone get excited over a new business application that was out on the market?)

To date, I have not used many of the skills that I have learned in school when I have developed business applications. I have yet to find a use for mathematics, graphics, sound, music and AI when performing a database query. Another issue I have is the fact that I have not been allowed to utilize any portion of my creativity. This is mainly because business applications are, for the most part, redundant (and should be in most cases). Most, if not all, of the development tools are plug-and-play or drag-and-drop, which puts heavy constraints on the creativity of the developers. Also, the end user community has come to expect business applications to be standardized in some way. It may drastically effect the company's product sales if they were to stray to far away from the norm.

While in the gaming community, the expectation is to be different and creative every time out. We may re-use some components (i.e., game engines, bitmaps, sounds, etc.), but each game must be unique in it's own way or the developer risks spending some time in the legal system.

So having exposure to both worlds, I pose this question to those that feel that only children get involved with game development. When was the last time you created a business application (including multimedia and web based) that learned how it's user was using it in an effort to be better, stronger, and more manipulative the next time the player fires it up or as the player advances deeper into the application?

Lone Wolf and Lone Wolf Pack Development Techniques

Now it's time to get to the good stuff. The following paragraphs document techniques that can be used to fit your needs as a single or small game development shop. Again, this is not a means to all ends, but it should give you a solid starting point as you create the structure that best suites you.

I think everyone in the game development community would agree that you first need an idea. Now most would say to be original in your thoughts, but I beg to differ. Since we were born into this world, we have been exposed to all types of things to stimulate our senses. We listen to sounds and music on a daily basis, we watch television and movies frequently, and we wake up everyday to a world that provides a rich bed of creativity. So it would be tough to come up with an idea that is completely original. However, I do believe that it is possible to use these things as a source of inspiration. It is highly probable that your idea will stem from a combination of previously established sources, but it's how you use them that will set you apart from another developers idea.

Second, you should flush out your idea(s) and write the high-level details down on paper. One tool that some game developers use, in addition to writing the details down, are storyboards. This will keep the creative juices flowing in the right direction and help to solidify the idea. Some developers may also produce a prototype, but I only use this mechanism if I'm trying to get the attention of a large-scale publisher or get some funding for the project.

The next step in the process is to create a design document. I know, I know---most developers hate documentation. However, this is a crucial step in the process. For one, this is the document that you must provide to a publisher if that is the distribution method you select or to the person or committee for funding. Two, if you were to bring in additional help (developers, artists, musicians, or sound effect technicians) they would expect something like this document to gain a better understanding of the idea and concepts of the game. Three, this document can be used to see if new ideas fit into the scheme of the product or to determine that new ideas should be saved for another game (i.e., impact analysis).

Here are a few tips on creating a successful and complete design document. The document should be fluid from start to finish. The introduction needs to catch the reader's eye, just like the introduction in the game itself. The body of the document should be the game itself, but be careful not to let the introduction overshadow the body of the document. How many times have we, as game players, been sucked in by the cool introduction only to be let down once we actually played the game? This document should flow in the same manner. At the end of this document, the reader should feel a sense of closure. The game delivered what it was supposed to so at the end of it all, the player feels that they accomplished the main goal or objective of the game. Just like in the movies, if you like the introduction and the body of the movie, but at the end feel that the director left some questions unanswered, you don't feel the movie was complete. Who knows, maybe they ran out of money and couldn't finish the project, but as a consumer you will not care. The bottom line is that you feel you were cheated. As a lone wolf developer or lone wolf pack, this should never happen because you are in complete control of the product, so take your time and give the game player what they want.

With that said, here is a breakdown of a simple, but very complete, design document:

- Game Overview: this is where you draw the reader (this includes your team members, the money holder, and the publisher) into the game itself and get them motivated to be a part of the project. You would also discuss the ultimate goal of the game.
- External Information: you provide the reader with information like genre classification/target audience, the platform the game will be for, and expected game-play time per session.
- Details: the following information should be provided to the reader for each level of the game
    - Story Relevance
    - Environment
    - Characters (guy and bad guys)
    - Obtainable objects
    - Action & Animation
    - Sound Effects
    - Music
    - Scenarios---if there is more than one way to accomplish the main goal of the game.

This structure should allow small shops to get up and running in no time at all and with very little effort. I cannot stress enough that this above structure is not set in stone, so feel free to choose and modify the document template as you deem necessary.

Another mechanism I use during my development is documentation/source code control. I have been a Configuration Manager for the past three years and I have learned the value of having documentation and source code controls in place. Many of you are probably familiar with Microsoft's SourceSafe and it is a viable tool for small development teams like ours. However, because I'm not solely an MS user, I'm inclined to notify my readers that there are other tools available as well. Tools like PVCS Version Manager or MKS Source Integrity are viable tools that are fairly inexpensive. There is also a wealth of free tools available to assist you in this area. Just look out on the Internet for more information or perform a search on Configuration Management.

In case you are not familiar with the concepts behind Configuration Management, here is a very brief introduction. The concept of controlling project items (i.e., usually only documentation and source code) of a project is not new. Larger organizations use this for every project (or at least they are suppose too) to meet the standards of their certification(s). A given company will try to remain compliant with the standards set forth by organizations like the Software Engineering Institute (SEI) as they release their software products. There are other organizations like ISO and IEEE, but that is entirely another discussion in itself, and therefore, will not be covered here.

This concept provides a tracking mechanism of all of the items that were placed into the selected tool. Once the items are in the tool, they can be checked-in and checked-out for modification and will allow you to go back in time to a previous version and continue forward from that point or rebuild a previous version of the product. In short, if you're using this mechanism properly and you modify the code beyond recognition, you can simply throw away the changes and restore a previous working version of any documentation or source code. I guess a simple way to look at it is like a backup mechanism. If a piece of code became unusable you can easily go back to a point in time without having to start over completely from scratch.

Conclusion

This article was created to provide the lone wolf developer and the lone wolf packs of the world with a mechanism or tool to be more consistent and successful with each product that is developed. It also helps the developer to cover their tracks when legal issues arise. Imagine that you have a cool concept for a game and you discuss it with others. One of the people in the discussion steals your idea and is making a profit from it. You know the game is based on your concept and with a solid development methodology you can prove it. You have your design document and you have your source code as evidence. The design document will be a key piece of evidence because I'm not to sure how many judges can read pure source code or even understand the lingo of the game development community.

In closing, I hope this article at least got you thinking about methodologies that will help you become more successful as a game developer. And if you have any tips that you would like to educate me about on this subject then feel free to send me an email or drop by my web site.

-Lorenzo Phillips
RenWare, LLC (CEO/Lead Developer)
www.byeight.com/~renwarellc
pain19@ix.netcom.com