Session 4

Trends in Game Development

Leon Rosenshein
Microsoft Corporation

Abstract

Game development has come a long way in the last thirty years. What was once done by one or two people over a few weeks working at a kitchen table now takes teams of over fifty people working together over a period of two or more years and costing tens of millions of dollars. Along the way the size of the games and the required data has grown at a similar astonishing rate. Recently, with the increase in acceptance of "serious games", the verification, validation, and repeatability requirements have also been raised significantly. To meet this challenge, successful game developers have had to put together processes and procedures to manage and track the development progress. This paper will examine this transition and identify some of the trends that have emerged.

Introduction

PC Game development has gone from someone working at his kitchen table developing games to give to his friends to a multi-billion dollar industry in just over thirty years. Along the way hundreds of small companies sprang up, released one or two games, and then disappeared. Others collapsed without ever offering anything for sale. The hardware platforms have continued to evolve, from the 16 bit CPU, 256 Bytes of RAM Altair in 1975 to today's 64 bit, dual core machines with over 16 gigabytes of RAM. Persistent storage has gone from paper tape to low cost terabyte drives, and display systems have gone from 25 line, 80 character text displays to bitmap displays supporting over 4 billion colors and over 4 million pixels. To support this new hardware, developers, their tools, and the companies they worked for needed to change.

Developers and companies began to specialize, and processes began to form. Developers specialized not only by the kind of code they wrote, but also into entirely different roles, such as content developer, tester, and product or project manager. Companies also specialized, some along game subject lines, some along hardware platform lines. Game publishing companies that did not actually develop any games of their own, but marketed and sold games developed by other companies emerged and flourished. The entire methodology behind the development of games became much more rigorous and often predictable. It has only been through these kinds of changes that today's successful million-selling titles have been able to be produced.

The Birth of PC Gaming

With the announcement of the Altair 8800 in January of 1975, the era of the home computer was born. As originally released the Altair was limited to toggle switches for input and red LEDs for output, but it wasn't long before someone noticed that the RF leakage could not only be detected by nearby radios, but it could also be controlled by varying the op-codes the CPU was executing. This was quickly followed by someone writing a program that allowed users to play "music" using their Altair. This marked the one of the first available pieces of entertainment software on a home computer.

As the hardware advanced and keyboards and CRTs became more common the software changed to take advantage of it. Text-based adventure games and simple puzzle-solving games began to appear. Over the next five to ten years game development was largely a hobbyist endeavor. Little of the work done during this period was done as a paying job, and few even considered charging for what they had done. People would work alone, or occasionally in small groups, freely sharing their best work. The developer(s) would not only write the code itself, but anything else (i.e., text, pictures, documentation) that needed to be created. The size of the code was constrained by the amount of memory available on then current computers, so individual developers were able to keep track of entire projects with little, if any formal documentation. The limited scope of these programs also meant that minimal formal testing was done as well.

The Early Years

The 1980's witnessed the birth of commercial game development. As the number of home computers continued to grow a corresponding market for computer games began to develop. People who were not hobbyists began to purchase computers for new uses, ranging from spreadsheets to word processing, and were beginning to look for other ways to use their investment. These people were not in general as technically savvy as earlier users, and were not able or willing to do their own development, but were willing to buy the products of others. This growing commercial market for software led to the emergence of both game development and game publishing companies. This dual nature of the industry continues to this day.

Game development companies evolved out of the hobbyist development community, and followed many of the same conventions. Entire games were developed by small groups of up to five people, with the developers generating any artwork, documentation, sound, or other non-code content needed. The game and game-play was the most important thing, and everything took a back seat to it. Schedules, documentation, testing—everything else, in short—was considered unimportant. Hardware limitations continued to keep the size of the projects small. Development cycles were measured in months and delivery was via floppy disk, so content size was extremely limited. This meant that everyone on a given project was still able to, and often did, understand everything about the project in detail.

At this time there were many different home computer systems available, with the Apple II, Atari 800, Commodore 64, and the Timex/Sinclair ZX-80 being amongst the most popular. Individual companies such as Broderbund typically focused on only one particular hardware platform, as each such platform was incompatible with the others. By contrast, some other companies focused not on one platform, but on one genre of games. As an example SubLogic's

Flight Simulator™ product line was originally available on both the Apple II and the TRS-80, subsequent versions were also available on the Commodore 64, Atari 800, and Macintosh computers. Meanwhile, Spectrum HoloByte's Falcon series was also available on those same computers.

As small groups of hobbyists got together to develop the games as they saw them, other groups of people got together based on a perceived market. These new groups had a completely different goal, to run a business and make a profit. They became the game publishing companies and took a wholly different approach. Rather than focusing on the game for its own sake, the publishing companies put their focus on schedules, marketing, and sales forecasts. In order to get into what they felt was a lucrative market, companies such as Microsoft, Electronic Arts, and Sierra On-Line purchased or acquired licenses to some of the leading flight simulation products.

Adolescence of an Industry

As the 1990's began the hardware base changed again. Moore's law continued to describe the advance in computer power available to the typical user, and games continued to evolve and expand to make use of that power. Game play continued to be important, but visual content and fidelity began to take a much higher precedence. The mid 90's saw the introduction of the graphics accelerator cards by, initially, 3dfx, followed shortly by NVIDIA, ATI, and Matrox among others. In addition to video, sound hardware became much more common, with Creative Labs' SoundBlaster family leading the way. As the variations in the PC platform continued to grow, the number of platforms competing with it dropped. With the exception of the Macintosh, other home computer platforms either disappeared completely or were relegated to niche markets and stopped being a factor in the industry.

The software development side of the industry was in the midst of a similar set of changes. The major publishing houses, such as Electronic Arts, Hasbro Interactive, Origin, and Sierra began to exert more and more control over exactly which games were made and delivered to the mass market. As with other creative industries, such as the recording industry or the film industry, the publishers were primarily business entities rather than creative entities. As the publishing houses grew they brought in more and more people whose background was in sales and marketing, often from completely unrelated industries. Before agreeing to publish or possibly to fund the development of a new title, the publisher needed to be assured that there was a reasonably good chance of a significant return on investment (ROI). This focus on ROI tended to make publishers less willing to try something new. This resulted in a long line of sequels, "branded" products tied to established intellectual property, and "me-too" games throughout the mid to late 90's and into the new century. Publishers also tried, with varying levels of success, to impose some sort of order on the chaos that was then current game development.

Rather than allowing the development team to work at its own pace and have no defined end point the game companies required a much more formal schedule. Design documents, deliverables, and milestone reviews became the norm in the industry. This level of rigor and detail was a huge change. Roles on a development team became much more clearly defined, and entirely new roles emerged. In prior times a development team would be made up of two or three people who could and did share all the work equally. Now, suddenly, managing game development started to become a role unto itself, as did testing, content development, level

design, and in some cases, licensing, as publishers sought to drive existing brands and properties into games.

At the same time, individual game developers were starting to be known by name, at least within the industry and by the most hard-core among the game players. Based on this "name recognition" and on the idea that the people in charge would never allow them to develop the perfect game, many of these developers split off and started new game development companies. Their goal was to find the creative freedom to develop the game they wanted to make. Unfortunately, in most cases the developers were not equipped to manage the required level of complexity. This led to significant churn in the development community as companies started and folded, often without releasing any games at all. Perhaps the most well know example of this is the story of John Romero and Ion Storm.

Ion Storm was founded in 1996 by John Romero and several other developers from id Software. Following his successes with Doom and Quake, Romero and others set themselves up in expensive, luxurious offices in Dallas and proceeded to announce to the world that they were about the make the perfect game that no one else would allow them to build, and that it would be ready within one year. Four years and millions of dollars later, Ion Storm's flagship product, Daikatana, and several other projects were released to much fanfare. Sadly, they were almost universally panned by critics. By 2001 Ion Storm was effectively gone. While an extreme example, Ion Storm was by no means unique for its time.

Individual Development Systems Mature

Even as the publishers and personalities were changing the way games were made, the requirements of the games themselves were causing additional changes. The size of games continued to grow in order to take advantage of all of the new hardware and power. Rather than being distributed on one or two floppy disks, games began to be delivered on CDROM. One CD was typical, two was common, and three was not unheard of. At the high end of the trend was 1996's Wing Commander 4, which took six CDs to deliver. The ability to deliver this level of content changed the way the industry developed games. Creating and managing this quantity of data opened the doors for entirely new classes of developers.

As with the amount of code being developed, the number of developers also grew. Rather than two or three developers, teams of five to ten became the norm. This increase in the number of developers changed the way the code itself was developed. Because it was impossible for all the developers to understand the entire codebase, it became necessary for developers to focus on one area at a time, if not to actually become expert on one and only one thing. To allow these separate areas of code to be developed in semi-isolation it became necessary to not only document the design after the fact, but to document beforehand and moreover, at least at a high level, to define the interfaces between the different areas or pieces of the game.

The size of the development team also brought with it the need for some kind of source control. In order to allow developers to work without adversely affecting the rest of the team each developer typically was given a sandbox in which to develop, and only when he or she felt the code was ready was it copied into the main codebase. In many cases this was initially done as a purely manual process, with each developer being responsible for maintaining his or her own

version and making sure the changes were correctly shared. As the number of developers on a team continued to increase the system was found to be unwieldy, error prone, and difficult to scale. To solve this problem teams turned to formal source code control systems, which automated much of the work, provided a history of changes made, and allowed teams to go back to previous versions as needed.

The volume of data used in these games also changed the way content was developed. For the first time pure content developers began to outnumber the coders. For the early versions of Flight Simulator all of the artwork, such as it was, was developed by the same people who developed the code itself. By the mid 90's the aircraft modelers and terrain modelers roughly equaled the coders. By the time Flight Simulator 2000 was released in 1999, the developers writing the actual code were outnumbered by the content creators in a ratio of approximately 1:3. Not only did all of this new content need to be created, it needed to be managed.

In the early years of the game industry there was no documented process for getting content into the game, be this content images, sound, level data, or game-play rules. Someone, typically the lead developer, would simply "know" how to do it. As the content continued to accumulate development teams learned that those simple scripts did not scale well and that ad-hoc process began to fall apart. At first processes were documented and someone would be assigned the task of manually running a process periodically. This worked for a while, but again, it was not scalable and was very error prone. The next step was to automate the process. Simple scripts were created to string all of the many steps used into a single pipeline that ideally could be run automatically without human intervention.

At the same time that these processes were being automated the content developers themselves were trying to work around the limitations of the small set of commercially available tools. For certain tasks, such as 3D modeling and image creation, where there were programs available, extensions or plug-ins were developed to make them better suited to the content developers' needs. For other situations, such as world-building or game-play rules generation, entire tools were developed by the teams to make the content builders more efficient. As these toolsets became more involved and complex game companies began to develop entire internal development teams to build and maintain just the tools and pipelines.

The increased amount of code and content also forced a change in the way games were tested. Rather than performing ad-hoc testing on completed systems, the test team was often split on lines similar to the code development, allowing testers to focus on specific areas just as the development team did. This allowed them to better understand what they were testing and to test it more effectively. The test team also became engaged in the process at an earlier and earlier date. This allowed the test team to better understand what was coming and to have some influence in the design so that it was at least possible to test.

The Industry Begins to Mature

By the late 1990's and early 2000's many of the pieces of game development had matured in isolation. The next step in the maturity of the industry was for the development system as a whole to come of age. This maturation is just beginning, with the larger, more successful developers and publishers seeking to take advantage of all of the lessons they have learned and

become even more efficient. These changes are taking place at a systems level, and often affect the entire team, not just one facet of development.

By taking a systems level approach to development and looking outside the games industry at how more traditional software development was done, game companies began to change the way they developed software. Some changes were simple and only affected individual developers. Changing from simple text editors to context-sensitive editors or to entirely integrated development environments enabled individual developers to be more productive. Most changes however were team-wide process changes.

The overarching goal of most of the process changes was to formalize best practices and to keep simple problems and issues from falling through the cracks and becoming bigger and more intractable issues. Some of these changes were as straightforward as standardizing the documentation formats used at the various stages of development, allowing people to quickly and easily find the information they needed.

Other changes, such as design reviews and code reviews, were designed to ensure that information was shared and that there was always more than one person that understood how any given part of the codebase worked. Not only developers, but designers, testers, and project managers might be included in these reviews. This provided insight from many different perspectives, and this allowed errors that might otherwise have been missed until much later in the process to be caught early.

Another area that was found to be ripe for improvement was the development cycle of building, testing, and bug fixing. In the early days of game development the test team might not see a version of the game until it was almost ready to ship, at which point there would only be limited time for testing and bug fixing. As the process matured the frequency of versions of the game being made available to test increased. At first only milestone builds were available, then there were weekly, and finally daily builds. As game content began to be moved into the source control system and the content pipeline became automated, it too was incorporated into these daily builds.

To help prevent regressions, where changes made on any given day caused bugs in already written code, build verification tests (BVTs) were developed and run on every build. Depending on the project, these BVTs could take hours and might run on multiple machines in parallel in order to cover as much of the game as possible. In some cases this build/verify cycle time was reduced even further by building and running developer regression tests (DRTs) as needed every time something changed. While BVTs might take hours, DRTs were typically designed such that the build and test time would be less than thirty minutes to ensure timely feedback.

In some cases there were also meta-system level efficiencies being achieved. Companies realized that it was possible to use the tools and engines that had already been developed for one game as the basis for another. Sequels, modifications, and derivative games became part of the plan for many game companies and publishers.

As the teams that worked on those tools and pipelines continued to expand the capabilities and usability of their tools they also realized that those tools themselves were valuable, saleable assets. By putting a little more effort into the fit, finish, and polish of these tools and pipelines, and then properly documenting the interfaces and usage of these tools, they too could be sold.

Emerging Trends

Increasing sales of these content tools led to the emergence of middle-ware tools and rendering engines that could be used as the core of multiple products. Games are now being built on top of successful engines such as the Half-Life, Quake, and CryTech engines. These engines come not only with runtime engines but with the tools and content pipeline needed to create highly detailed worlds with deep, immersive gameplay. At the other end of the spectrum, products like GarageGames' Torque or Microsoft's XNA allow the independent developer, working alone or in a small group, to develop and share/sell a full featured game with minimal knowledge of the deep intricacies of graphics, sound, or physics. The full effect of this is just starting to be felt in the industry and the end results are far from clear, but the potential to ignite a "garage-band" bonfire of new game development could be enormous

Conclusions

It is only the continuing evolution of the past thirty years that has allowed the game development industry to keep pace with and take advantage of the ever-increasing power available in home computers. Where once a single person could develop a game in a few months working on his own, the ever-increasing sophistication of today's games demands two or more years and a small army of developers, testers, interface designers, level designers, artists, writers, musicians, project managers, and assorted others to develop a bestselling game. Rather than the old free-form development process, the industry has adopted a set of processes and best practices that allow developers to take advantage of hard-won experience while still allowing game developers to maintain the creative freedom needed.

The game development industry has reached a level of maturity that allows for creativity and innovation while simultaneously maintaining the predictability that is needed to run a profitable business. Improvements and changes continue to happen, but the rate of change has been slowing.

When the game industry first began it was fueled by dedicated hobbyists working alone with the home version of the tools used by the then current giants of the software industry. Since then the increasing power of the development platforms and the codification of the experiences learned has allowed the creation of sufficiently powerful toolkits to make game development accessible to many more people. It is once again possible for someone working alone in his or her kitchen to develop a game and share it with friends, bringing the industry full circle. Given the overall rate of advance in the power and capacity of gaming hardware with such exciting new advances as stream programming, the continuing development of computer game technology is likely to lead to even more astonishing and unforeseeable marvels in the near future.

Bibliography

GarageGames: http://www.garagegames.com
Altair 8080: http://en.wikipedia.org/wiki/Altair_8800
Game Development: http://en.wikipedia.org/wiki/Game_development
History of computer and video games: http://en.wikipedia.org/wiki/History_of_computer_games
Levy, Steven, 2001, *Hackers: Heroes of the computer revolution*, Penguin
John Romero: http://en.wikipedia.org/wiki/John_Romero
XNA: http://msdn.microsoft.com/directx/XNA/default.aspx
MobyGames: http://www.mobygames.com

LEON ROSENSHEIN, BS Cornell U. '88, MS Northrop U. '90 is a Development Lead at Microsoft Corporation where he is part of the Virtual Earth team. During his seven years at Microsoft he has worked on various simulation products including Flight Simulator. Prior to that, Leon worked on various simulation projects with the U.S. Air Force.