Evolution of Game Development

Computer games have evolved along with the evolution of programming languages and computer hardware. Computer games are a relatively new medium, they’ve gone from being simple tests to show off and benchmark hardware and programming languages to having Hollywood blockbuster level budgets that push the limits of today’s programmers and engineers. To understand how video games evolved into their modern state we must understand the methods and the history of how they are created and how they’ve evolved.

One of the earliest computer games is a game called OXO, also referred to as Noughts and Crosses, it’s a version of tic-tac-toe written for the EDSAC (Electronic delay storage automatic calculator) programmed in an ALGOL like language [12]. The program was written by Alexander S. Douglas as part of his PhD thesis at the University of Cambridge. The EDSAC is a vacuum tube computer that was invented in 1949. Douglas programmed his OXO program in 1952. His program displayed graphics onto a 6” VCR97cathode ray tube display. Douglas’s program is one of the first graphical computer games, if not the first. In OXO the player competes against the computer, this demonstrated one of the first examples of simple artificial intelligence. Since the EDSAC was a unique computer, emulators exist online along with the source code for EDSAC programs that allow OXO to be run today.

ALGOL (Algorithmic language) is an imperative programming language designed specifically for scientific computations. ALGOL was first introduced in the late 1950’s. The first formal version of the language was ALGOL 58. ALGOL went through many iterations including ALGOL 60 and ALGOL 68 [1]. The more recent language, Pascal, is a decedent of ALGOL. Imperative programming requires the developer to write code that describes the method of tasks the computer must complete in order to accomplish a goal [2]. Imperative programming is also referred to as algorithmic programming.

Another early computer game is a game called Tennis for Two. Tennis for Two was written for analog computers in 1958. The game was developed by William Higinbotham for an interactive display at the Brookhaven National Laboratory that was open to the public. Higinbotham used a model 30 analog computer along with an oscilloscope in order to display graphics. Unlike XOX in which a user played against a computer that used simple AI, Tennis for Two, as the title implies, was a two player game. Unfortunately, the game had flaws. One of which was that the user could keep the ball in the air by mashing the button on the controller making it impossible to lose [3]. The game also lacked features such as scoring and basic error detection in some circumstances. These oversites and small bugs were no big deal since just seeing a video game was something incredible at the time.

Three years later in 1961, a game called ‘Spacewar!’ was developed by an MIT student named Steve Russell for the PDP1 (Programmed Data Processor-1) computer [13]. The PDP1 could be programmed using assembly language. On a side note, the first text editor, and word processor, along with one of the first examples of computerized music were created on the PDP1. Spacewar is a two player game in which each player fires missiles at the other in order to destroy their opponent’s ship. Steve Russel, the creator of Spacewar, transferred from MIT to Stanford University where he showed his Spacewar game to a student named Nolan Bushnell. Bushnell later went on the form the Atari video game company. Like the EDSAC the PDP1 can be emulated on a modern computer and the source code for Russel’s Spacewar is available online.

Assembly Language is a low level programming language in which each assembly instruction translates to roughly one machine code instruction. Since different computers have different processors they require different assembly languages. Most of the assembly languages are similar though, MIPS and Intel’s x86 architecture are examples of this. They both have their own similar assembly languages that vary in slightly different ways. Game developers, even today, use assembly language for various reasons. One reason why assembly language is sometimes used in game development is because it’s quick to execute on the processor. Developers wanting to speed up their programs will profile their games to find the slowest parts and then program those parts in Assembly Language in order to get a speed increase [4]. Since modern games need to run multiple tasks quickly and efficiently assembly language is a great way to optimize such tasks. Assembly language also allows the programmer more low level control over how their program will run. It gives the programmer direct access to system resources such as registers on the CPU, memory, and the stack.

The late 1960’s saw the emergence of a new type of computer, one built specifically for gaming. The first game console was invented by Ralph Baer in 1966. “Specifically, he’s responsible for the “Brown Box,” a prototype he built in 1966 that would eventually become the Magnavox Odyssey, the first game console.”(Matt 129) Since early computers were expensive, game consoles began to gain traction in the marketplace because of their more affordable price point. It wasn’t until the mid-1970’s that game consoles saw mainstream success. In 1983 the video game industry specifically home consoles and arcades saw a major lag in sales. Ironically while video game console companies experienced the effects of the crash, Computer Company’s such as Apple and Commodore International were able to get personal computer prices down to more reasonable prices. This meant more households could afford to buy a computer. While the console market was in decline the PC market was just beginning to take off, so while the game crash was happening future game designers were learning how to code and use PC’s [5].

One of the most well know computer games, Pong, created in 1972 by Atari Inc., was a derivation of other ball and paddle games that already existed. Pong saw success because it was released on the Atari home console in 1976. Pong was popular in arcades before its release on the Atari home console, which may have helped with its success on the home console. Pong wasn’t coded in a programming language, but built using hardware and transistor-transistor logic.

The popularity of game consoles has continued to grow through the decades since the Magnavox Odyssey. Each decade consoles continue to add new features which require different programming languages. The Nintendo Entertainment System released in the United States in 1985 for example ran games written in assembly language. The Gameboy Advanced, a portable console released by Nintendo in 2001 ran programs written in C or ARM assembly language. One of the most popular games of the past decade is Minecraft. Minecraft was developed using the object oriented language Java. Minecraft was purchased by Microsoft in 2014 for 2.5 billion dollars. Of course Microsoft wanted to make Minecraft available to as many potential players as possible so they had to port the existing Java code to C++ code so that it could run on their game consoles the Xbox 360 and the Xbox One. Since 59% of Americans play video games, a number that continues to increase, it’s safe to say that the demand for developers and ways in which developers can create games will continue to increase [6].

Java is an object oriented programming language developed in 1991 by a group of engineers called “Green Team” at Sun Microsystems. The “Green Team” was led by James Gosling. Java gained traction as a programming language due to the growth of the internet during the mid-1990s. Java is widely used today in the development of Android games and applications. One of the major advantages that Java offers is its portability. Java bytecode will run on any hardware that has a compliant JVM (Java virtual machine) [7]. Most modern AAA video game titles aren’t written in Java. One reason for this is that frameworks, game engines, and legacy code exists for game development in other languages. Using such frameworks and game engines would save programmers time and money developing their game. Another reason why a game developer may choose to use languages such as C++ or C over Java is because Java is not supported on any major game console. If a developer were to port their computer game to a console they would have to rewrite their code in a language supported by the game console. C++ has historically been the predominate language for teaching object oriented programming. Since game developers may have learned to program using C++ they would rather create their games in a familiar language than trying to learn Java, a whole new language. Java is a popular language for web based computer games though because of its easy implementation with web browsers.

Most games today are developed using object oriented programming languages such as C++, C#, and Java. Object oriented languages are great for game design for multiple reasons. They allow for easy code reuse. For example, the programmer could create code for a monster object. The programmer could then declare new individual monsters using the code they created in there monster class instead of writing code for each instance of a monster in the game. Simply, it’s more efficient and less work for the programmer. Object oriented programming allows for features like inheritance and polymorphism. Inheritance allows for an object to inherit the properties of another object. This would make it simple for a programmer to create a boss monster from a simple monster class they had already created. Properties that dealt with health would be the same for both the boss and the regular monster for example but the boss monster may need new methods and properties that the regular monster would not. Polymorphism is how the behavior of a piece of code changes depending on the data types used. Polymorphism is used in implementing Inheritance [8]. As far as AAA games are concerned, C++ is the most heavily used programming language.

C# is a programming language created by a team of computer scientists led by Anders Hejlsberg at Microsoft released in June of 2000. The purpose of creating C# was to build upon existing languages and add new features. C# is an object oriented programming language and it borrowed ideas from Java, so much so that the creators of java said that C# is java without security and portability [11]. C# is was built with C and C++ in mind, but distinct features differentiate it from C and C++. C# applications run on the .NET Framework which was also created by Microsoft. If the developer chooses to use the .NET Framework, then, when a user installs the application on their computer they will also need the .NET Framework installed on their computer. Put simply, a framework is code a programmer can use without having to explicitly write it out [9].

With the advent of the iPhone in 2007 cellphones have become ubiquitous. Cellphones have transformed the way in which video games are played and created. Most mobile games are created for the two big platforms, iOS and Android. Since iOS requires applications to be written in the Swift and Objective-C programming languages and Android requires applications to be written in Java, developers who want to port their applications to either platform must write a version of their application in both languages. This can be an extensive process, but it’s worth the effort for developer’s to get their applications working flawlessly on both platforms. According to Artyom Dogtiev, mobile application revenue was projected at 45.37 Billion in 2015 [10].

Computer games will continue to advance, and as they advance new programming languages, game engines, and frameworks will be created to allow programmers to build games and applications that take advantage of the advancements in hardware. Virtual reality games are an example of new hardware that will require programmers to adapt to new styles of programming. Just as Java saw an increase in popularity because of the World Wide Web new programming languages will come along because of advancements in technology.

**References**

In text brackets used to denote where sources used. The bracket corresponds with the source below it. Sources divided into the section they were used in.

**General History**

[13]

Overmars, Mark. "A Brief History of Computer Games." (2012): 1-35. *History of Games*. 30 Jan. 2012. Web. 20 Apr. 2016. <www.cs.uu.nl/docs/vakken/.../history\_of\_games.pdf>.

Laid, John E. "History of Computer Games." *History*. 7 Sept. 2005. Web. <http://www.emunix.emich.edu/~evett/GameProgramming/History.pdf>.

[12]

"Pong-Story : A.S.Douglas' 1952 Noughts and Crosses Game." *Pong-Story : A.S.Douglas' 1952 Noughts and Crosses Game*. Web. 23 Apr. 2016. <http://www.pong-story.com/1952.htm>.

**Tennis for Two**

[3]

"Tennis For Two." *Simulator*. Web. 22 Apr. 2016. <http://www.gamersquarter.com/tennisfortwo/>.

"The First Video Game?" *BNL*. Web. 22 Apr. 2016. <https://www.bnl.gov/about/history/firstvideo.php>.

**Spacewar!**

"The History of the Computer Game SpaceWar." *About.com Inventors*. About.com. Web. 22 Apr. 2016. <http://inventors.about.com/od/sstartinventions/a/Spacewar.htm>.

**Assembly Language**

"Assembly Language." *Encyclopedia Britannica Online*. Encyclopedia Britannica. Web. 22 Apr. 2016. <http://www.britannica.com/technology/assembly-language>.

[4]

Buuck, Philip. "Assembly Language in Game Programming." *Philip Buuck*. Web. 22 Apr. 2016. <http://philipbuuck.com/assembly-language-in-game-programming>.

**Home Consoles**

[12]

Barton, Matt. *Honoring the Code: Conversations with Great Game Designers*. CRC. Print.

[5]

"Ten Facts about the Great Video Game Crash of '83 - IGN." *IGN*. Web. 22 Apr. 2016. <http://www.ign.com/articles/2011/09/21/ten-facts-about-the-great-video-game-crash-of-83>.

[6]

"Essential Facts About the Computer and Video Game Industry." *2014 Sales, Demographic, and Usage Data*. Web. Mar.-Apr. 2016. <http://www.theesa.com/wp-content/uploads/2014/10/ESA\_EF\_2014.pdf>.

**GBA Console Programming**

"Gameboy Advance Programming for Beginners - Loirak Development." *Gameboy Advance Programming for Beginners - Loirak Development*. Web. 22 Apr. 2016. <http://www.loirak.com/gameboy/gbatutor.php>.

**Pong**

"Pongtronics." *Pongtronics*. Web. 22 Apr. 2016. <http://www.cs.ubc.ca/~hilpert/e/pongtronics/>.

**Java**

[7]

"What Is the Meaning of "Java Is Portable"?" *Portability*. Web. 22 Apr. 2016. <http://stackoverflow.com/questions/13068510/what-is-the-meaning-of-java-is-portable>.

"Digital Foundry vs. Minecraft Xbox 360 Edition." *Eurogamer.net*. Web. 22 Apr. 2016. <http://www.eurogamer.net/articles/digitalfoundry-vs-minecraft-xbox-360-edition>.

**C#**

"What Is C#?" *- Developer.com*. Web. 22 Apr. 2016. <http://www.developer.com/net/asp/article.php/922211/What-is-C.htm>.

[11]

"C Sharp (programming Language)." *Wikipedia*. Wikimedia Foundation. Web. 22 Apr. 2016. <https://en.wikipedia.org/wiki/C\_Sharp\_(programming\_language)>.

[9]

Person, and Kevin Purdy. "What Is the .NET Framework, and Why Do I Need It?" *Lifehacker*. 2011. Web. 22 Apr. 2016. <http://lifehacker.com/5791578/what-is-the-net-framework-and-why-do-i-need-it>.

**Object Oriented Programming Languages**

[8]

"Advantages and Disadvantages of OOP." *Advantages and Disadvantages of OOP*. Web. 22 Apr. 2016. <http://wiki.tcl.tk/13398>.

**Mobile Applications**

[10]

"App Revenue Statistics 2015 - App Industry Insights." *Business of Apps*. 2015. Web. 22 Apr. 2016. <http://www.businessofapps.com/app-revenue-statistics/>.

**ALGOL**

[1]

"The ALGOL Programming Language." *The ALGOL Programming Language*. Web. 22 Apr. 2016. <http://groups.engin.umd.umich.edu/CIS/course.des/cis400/algol/algol.html>.

[2]

"Functional Programming vs. Imperative Programming." *Functional Programming vs. Imperative Programming*. Web. 22 Apr. 2016. <https://msdn.microsoft.com/en-us/library/bb669144.aspx>.