

Beginning Programming

Typing convention:

Courier New - Code extract

If you're reading this, you're either curious or starting out with programming. Programming is no cake walk. It is hard, sometimes frustrating, work. But it will pay off in the end when you see, hear and play what you've spent God knows how many hours throwing code into your text editor. This is not a tutorial on how to program, it is a guide to what you need to know to get started on programming. I am considering writing some programming tutorials with C++, though. Before continuing reading, think long and hard about whether you really want to do programming, be it as a hobby or a profession. Would you rather be happier to design games, create the audio or write them? There are many different avenues to take in the game industry, the ones that are most often talked about are the design, programming and art aspects. Without the detail it's hard to see all the different jobs there are. For instance, there are AI programmers, tool programmers, physics programmers, graphics programmers, network programmers, etc. They all break down into different areas, as you can see. However, if you're set on programming, I won't waste any more time on this introduction.

Programming Languages

The first thing you'll need to decide on before you leap into the shallow end of programming is a programming language. This is where you will really need to think about how far you're going to take this programming lark. Putting the time in to learning a programming language is just like learning another human language such as German, Japanese or French. Only, it's just learning the written form, if you will. You may be confused as to what I'm even writing about when I say "language", after all, it's not as if you and the computer talk to each other, is it? Well, it is kind of like that. If I were to write a program like a calculator, I'd be inputting numbers in some way for the program to process and that would show some output (if no syntax errors were present in my calculations). While this is not an example of programming directly, it is an example of interaction between the user and the machine. When programming, you type in commands into a text editor that break down into the computer's native language when the program gets compiled (more on that, later). These commands have a syntax to follow, again, like a human language does.

There are four generations of programming language:

First Generation (Machine Language)

This is the lowest possible level of code, it is essentially this:

```
1000 1000 1111 0000
```

Assuming this is a 16-bit machine we're looking at, of course. Each 1 or 0 is called a bit, **binary digit**. Binary just means two values, true or false, yes or no, 1 or 0. It is also called base 2, humans work in base 10, usually. Luckily, you won't have to deal with this in your programming adventures.

Second Generation (Assembly Languages)

Assembly language is a much easier to read language. It consists of code like this:

```
MOV ESI, THIS
```

This is some simple X86 assembly. Most CPUs in desktop computers are X86_X64, now. Anyway, the usual syntax for assembly-level code is like this, where you have the command to be taken, the register to operate on and the value to use. Others switch the order of the value and register. A register, in case you were wondering, is a kind of store for values, these values are usually read/write.

Third Generation (Imperative)

These languages are laid out like this:

```
int c = a + b;
```

Credit to Hendore for the example, I was worried that I'd get hung up on explaining everything this line has to offer that isn't required in this introduction. This particular line could be for C, C++, C#, Java or another language that uses the keyword int for integer. The letters after int are what are known as variables (like in mathematics, a variable can have its value changed), in this case, the entire line simply states "the integer, c, is a plus b". Simple algebra. The semi-colon is there to signify the end of the line, some imperative languages don't require it. This will be the level you will most likely be starting out at. Though, learning some assembly doesn't hurt, it can help in most cases. You'll have more respect for your compiler, that's for sure. Which leads me on to the next topic.

Compilers, Linkers and Assemblers

All of that lovely code you write has to be compiled down into an assembly listing, which is then stored in an object file (or multiple object files, as the case may be). Finally, everything is linked together, taking all those object files and producing the executable at the end of it all, if you're building an executable file, that is. The linker will resolve any external references and assigns the absolute final addresses to the functions and variables in the process. That's pretty much all there is to that step. I'm skimming on the gory details because there's not much more beyond that you need to know.

There are many compilers out there, such as GCC (GNU Compiler Collection), Borland's C++ Compiler and Microsoft's Visual C++ Compiler (not the IDE, I'll get to that soon). These will take care of compiling, assembling and linking your source code for you. Assuming you're programming with C++.

Languages such as Java and C# are built into bytecode, this is code that's ran into a virtual machine. This tends to be slower than a native compile, but it is also good for cross-platform projects that aren't going to be heavy on the processing.

IDEs (Integrated Development Environments) are programs that act as an all-in-one package for your compiling, editing and debugging of your code. So you don't have to switch between the text editor, debugger and open a terminal to compile your executable file. Microsoft's Visual Studio is pretty much the IDE of choice for Windows development. It can also cross-compile for the Xbox if you have access to the Xbox SDK (Software Development Kit) and other platforms, too. Code::Blocks is also popular as is Eclipse for Linux development. There are a lot of other IDEs, such as NetBeans and Borland's IDEs. Depending on your development environment and programming language, you may spend quite a while searching for your ideal IDE.

APIs

How do all those shiny little pixels switch colours and that beefy gunfire emit from your 7.1 surround sound system? The answer is an API (Application Programming Interface) or two, or more. APIs help by lifting a lot of the weight from the programmer's shoulders. Back in the day, you had to choose what graphics accelerator you were going to support, what audio card, Roland? Sound Blaster? Yes, there are specific APIs still out there, but it was much more restrictive not too long ago (really!) when you had to consciously make that decision. Now we have DirectX with it's Direct3D, Xinput and other API branches as well as OpenGL. No longer do we have to choose nVIDIA, Ati or Matrox, as there's an API that they all implement.

I've talked about the merits of APIs for standards, but what does an API do and how does it do it? Put simply, you could create your own API for just about anything. However, you probably don't want to for common things, unless you're working for one of these silicon chip manufacturers or if you want to gain more understanding of what happens under the hood. You may wish to create your own rendering engine, however, and by doing that, you've created your own API.

Resources

This short and (hopefully) sweet introduction to programming has either scared you from ever wanting to delve into programming or has pulled you a little further into the rabbit hole. If you're still being pulled in, here are some links to tutorial websites, reference websites as well as titles of books for tutorial and reference on various programming languages.

Websites

C

<http://www.cprogramming.com>

Language Reference and Tutorial

<http://www.cplusplus.com>

Language Reference

<http://www.programmersheaven.com>

Language Tutorial

<http://www.programmingtutorials.com>

Language Tutorial

C++

<http://www.cprogramming.com>

Language Reference and Tutorial

<http://www.cplusplus.com>

Language Reference

<http://www.programmersheaven.com>

Language Tutorial

<http://www.programmingtutorials.com>

Language Tutorial

C#

<http://www.cprogramming.com>

Language Reference and Tutorial

<http://www.programmersheaven.com> Language Tutorial

General Game Development

<http://www.gamedev.net>

<http://creators.xna.com>

<http://developer.nvidia.com>

Artificial Intelligence

<http://www.aiguru.com>

<http://www.aigamedev.com>

<http://www.ai-junkie.com>

Graphics

<http://www.msdn.com/directx>

<http://www.opengl.org>

<http://www.khronos.org>

<http://nehe.gamedev.net>

Books