



So You Want To Be A Game Programmer?

Marc Mencher

April 7, 2006

Game artists create wondrous landscapes and bizarre characters; game designers write the documents that plot how the game will play. But without programming code, all there'd be in a game would be pretty pictures and a bunch of words. Game programmers -- also known as software engineers -- create the code that makes the game work.

Of all the programming disciplines, coding for games may be one of the most difficult -- and challenging. That's because of the many different programming disciplines and skills that a game programmer must have under his or her belt -- graphics, animation, collision detection, networking, physics, database, GUI (Graphical User Interface), audio, input, and more. But, as difficult as game programming may be, it's also the programming discipline that's the most rewarding. Imagine what it must feel like to be able to go down to GameStop and see your game up there on the shelf!

At its most basic level, a programmer's job is to translate a solution to a problem into a language that a computer can understand. This ability, of course, is not enough. To have a successful career in the game industry, you must be able to write or "cut" code that is efficient, fast, and reusable. That code must be flexible ... and so must the programmer. Frequently you will be expected to complete a task or create new technology in an area that is unfamiliar to you. This will also take place in a production environment, which means that not only will you be learning, but you will also be learning under a deadline. An additional positive aspect of the job is that you will rarely find yourself bored. Every new project promises exploration into new and uncharted territories. You can expect to find the same programmer working on a game engine in one project, building tools in another, and developing a game's core logic routines in yet another. And, above all else, game programmers must be absolutely passionate about making games and solving problems no one has seen or even thought of before.

Sounds Great! Where Do I Sign Up?

Whoaaa there! Are you sure you have what it takes to be a game programmer? While people are often at different skill levels, there are certain core skills that all game programmers need. Let's go through three sample cases of different skill sets and discuss how one might move forward from those points. Remember that most game companies don't have formal training or internship programs. They use advanced technologies on the bleeding edge and, to do that, they seek people who can jump in and immediately contribute to a development project. If you can do all of the things listed in any of these three categories, you are on the right track:

New To Programming: You've just started working with a new programming language. You could be a high school student or someone casually interested in programming. If that sounds like you, do the following:

- Select one language to learn at a time. A good one to start with is C++ or Java.
- If you have a textbook associated with a class you're taking, read it! There is valuable information and code samples to review in books. Don't ignore them.
- Type in all the sample programs in your textbook and experiment with them.
- Experiment with the code listings to learn how different operators work.
- Once you start to learn conditional statement like "*while*" loops, see if you can make a "Guess The Number" game. Then try making a game of "Hangman."



- If using C, C++ or any language that uses pointers, make sure that you are comfortable with the concept and usage of pointers fully and are comfortable with using them in a variety of ways.
- Learn as many aspects of the language as you can. Once you've made a "Guess The Number" or "Hangman" game, see if you can change or re-implement the workings of your program to use new techniques that you are learning. This is one of the best ways to learn a language because you know the expected outcome of your game. Modifying it will make the game better and more robust. At the same time, you'll also be improving your programming abilities.

While you're at it, here are some tips for new programmers:

- Get ready to be frustrated. This is an integral part of programming and it will stay with you long into the future.
- Don't give up when something gets hard. This might sound like an old cliché but all programmers go through the early stage problems and have to power through them. Giving up too easily may be a sign that game programming is not the profession for you.
- Learn how to use the debugger.
- Comment your code! It's really hard to help someone when you have to take 40 minutes to figure out what that person is doing. Commenting is key to becoming a professional programmer and working in a development team.
- Read everything you can on the language. Watch and learn from the mistakes you make. You will always keep learning from a language even if you have been using it for a long time.
- Look at other people's code and how they organized it. How easy is the layout to read through? How easy is your code to read through for another programmer?

Some Experience Programming: You have completed a course or been studying a language for a little while, probably more than nine months. You can write programs that will solve simple problems and, perhaps, you have written a few small text-based adventure games. At this point, the next step is to start working with some game tools. Look into gaming libraries and SDKs like [Allegro](#) and [CDX](#). They are free or shareware and can be downloaded from the Internet. If you are at this stage, the following is your "to do" list for improvement:

- Read through the documentation on the game library of your choice. Look through sample source code and get some ideas of how things work in the organization of the library. Reading documentation is an important skill to have. If you come across something you need but don't know about, you will need the ability to research and learn as you go. This ability to adapt and overcome makes you a valuable component in a development team.
- Work with the game library to solve some basic problems. You should write test or "driver" programs to solve problems like: Drawing a single sprite to the screen; moving the sprite across the screen; testing for collision with the objects on the screen; playing some sounds; testing the input mechanism for the keyboard, mouse, and joystick; finding an existing tile engine and making some basic maps.
- Then, working with those tools and programs you have now built, write a basic game. A game similar to the original *Asteroids* would be perfect. That should take you about two weeks. Don't sweat it if it takes longer; just make sure that you finish.

Programming For Some Time: Perhaps you started programming in high school and have progressed into college. At this point, you might be a junior or senior in a computer science or computer engineering degree program. You could also be an enterprise developer looking to make a transition from database programming to the game industry. Your knowledge of syntax



and data types will help you write better programs and games. In any case, you should be able to complete all of the tasks that have been listed in the previous two sections and now you need to tackle the bigger problems. A background in computer science or a game training program will be very beneficial. My suggestions for you are these:

- Write a bitmap and wave file loader.
- Write a tile map engine.
- Create a state-based AI engine.
- Try to simulate some basic Newtonian physics models in 2D games.
- Write an RPG in the vein of the classic NES games.
- Learn Windows MFC and basic Win32 programming methods.
- Learn basic assembly code for Pentium processors.
- Learn DirectX.
- Learn basic concepts of 3D.
- Work with an existing 3D engine and try to get some basic concepts down.
- Start a larger project with some friends. Focus on laying out a plan to follow during the course of production and stick to it as best as you can. Make sure that you note where your plans went bad and try to figure out ways to anticipate prevention of similar mistakes in future projects.

And here are some tips for experienced programmers:

- Try to emphasize the robustness of your design. Make things as elegant as you can without causing issues in performance. Don't hesitate to use Object Oriented Programming methods, unless the platform prohibits it.
- Make sure that you have strong group skills. This is one of the most critical skills to have in the game industry. Often studios won't hire you if they think they could not effectively live with you for three months at a time, even if you have an impressive skill set.
- Read as much as you can on the games industry and the techniques it uses.
- Write clean code. Ask someone to review your code to see if it is easy to understand.

Do You Have 'The Right Stuff'?

When an employer begins a search for a particular programmer candidate, they will likely have many different expectations about the type of person they would ideally like to find. You will almost always find that there are some desired qualities that are specific to that particular company (such as the ability to speak French), but there are some general qualities that will always make you an extremely valued employee at any company.

Team Attitude: Probably the single most important quality in a production environment, team attitude implies a type of selflessness and dedication to the team and project over your own personal interests. If people on your project are more concerned with what title they will get on a project, petulantly establishing who is "right," or spending their time pointing fingers at others, the chances of your project making it out the door and onto the shelves are not all that bright.

Self-starters: These are the programmers who do not need constant supervision. Once given a task, they will supervise themselves. They will ask another programmer on the team for information when and if they need it. They will go to the designer and ask questions relating to the implementation of their task. They will go to the artists and track whatever assets are needed for their task. This ability to self-start allows a manager to assign a task and walk away knowing that they do not have to oversee all the different aspects of accomplishing that particular task. Self-starters are worth their weight in gold.



Follow-through: The ability to see tasks through to the end, on both a micro and macro level, is a highly valued quality. There is a thrill with the beginning of every new task you take on. It is something new and exciting and you generally don't have any difficulty getting started on it. Sometimes, however, the last 10% of a task can take 90% of the time. When you are working on that last 10%, it can be a bit painful. There will be times when you will be getting feedback from designers and artists that forces you to retread old ground, revisiting the same code/functionality over and over again. It can become tedious, but rarely do you nail something such as AI character behavior the first time around the block. Perfection will come with time.

Communication: Communicating -- and communicating well -- with others in a team environment is extremely important. One of the biggest tendencies that programmers have when communicating is to slip into "programmer speak" or jargon. This can glaze over the eyes of nearly any artist, designer, or producer. It is a rare gift indeed for a programmer to be able to communicate to others in a language that they understand. This isn't easy; it can take time to perfect, but it is well worth the effort.

Responsibility: Taking responsibility for what you have done or what you are supposed to do goes hand in hand with all the other qualities. Those that embrace responsibility and, in fact, seek more are those who rise to lead programmer status and become one of the cornerstones of any team.

Last But Certainly Not Least...

When all is said and done ... when you've acquired the right skills, determined you have the right qualities for the job, are convinced game programming is the career path you want ... you should consider whether the wages are the sort you're after.

Salaries, as in any job, depend on where you sit on the food chain. If you're at the lower end (an associate or junior programmer), somewhere in the middle (a senior or mid-level programmer), or at the top (a lead programmer or a director/VP of engineering), your paycheck will differ accordingly.

Regardless, you should be aware that there does seem to be a lower average salary for programmers in the game industry when compared with business programmers from equal positions of experience and education. If that is the case, why would anyone want to work for less money doing the same job?

The truth of the matter is that, while the skill of programming may be the same or similar, the jobs are completely different. On a game project, you have a good time working on a piece of entertainment. You also have more liberal hours and you don't have to wear a suit to work. The bottom line is to imagine whether you would want to work on a project writing drivers for printers or on a project making the next *Doom*.

Most people who love making games make the sacrifice of money for a job they love to work at every day. Give that some thought before you charge full-speed-ahead into that next game company employment office.