Eric Arnebäck Home Projects Articles Resume

Interviewing for your First Job as a Graphics Programmer: a Checklist of Common Interview Questions

Update 18/3-2018: After posting this article on Twitter and receiving some feedback, I added some additional questions that I realized I had forgotten about

I have recently been interviewing at game companies, trying to land a job as a junior graphics programmer. Having done so, I gained knowledge of what skills game companies expect of a newly graduated graphics programmer, and what questions they are likely to ask during interviews. I will in this article compile these findings into a handy checklist. The intention is that other newly graduated programmers can use this list as a way to prepare, by providing them with a list of topics that they should rehearse before going into their first interview. However, as a disclaimer, **I do not recommend just memorizing a bunch of answers to these questions**. The topics in this list are topics that you need to understand and master in order to solve the programming problems that you will encounter as a professional graphics programmer. Always strive to understand, not to memorize.

The questions that will probably be asked during the interview can be organized into the categories of **C++**, **Mathematics**, **Optimization** and **Computer Graphics**. As can be observed, these categories are topics that are central to the day-to-day work of a graphics programmer; C++ is a language that is pretty commonly used in graphics programming jobs, so it is natural that there are many questions about this language during the interview. Further, graphics programming requires a better handle of math than most other types of programming, so math skills are paramount. Finally, in order to achieve 60FPS while still rendering a game with high graphical fidelity, strong optimization skills are a must. Let us go through a number of common questions in each category.

C++ Questions

- When should you use virtual destructors?(interviewers absolutely love this question!)
- What is the difference between allocating memory on the heap versus the stack?
- What C++11 and C++14 features are you using?
- What are templates used for?
- Explain the `inline` keyword
- What is little and big endian?
- Explain what const-correctness is?
- What overhead is associated with calling a virtual function?
- There will probably be some question where you are asked to perform some bit twiddling tricks, using operators such as `&` and `|` and friends
- There will probably be one question where you are asked to do something with a linked list, or some other pointer-based data structured like a tree. For instance, reversing a linked list.
- What is the size of a pointer in C++?(that is, the result of `sizeof` for a pointer)

As a general rule, none of the questions were about advanced language features. I was for instance never asked to do any template metaprogramming, which was a relief.

Mathematics Questions

There was actually not much variety when it came to the math questions:

- What is a dot product?
- What is a cross product?
- Why should you use quaternions over euler angles?
- How do you use matrices to apply transformations to an object? For instance, how do you scale, translate and rotate an object with matrices?
- How do you calculate the intersection between a ray and a plane/sphere/triangle?

- Explain concepts like world space, object space and camera space.

You will probably also be asked to solve some practical, graphics-related problems that will involve using dot products, cross products, and quaternions. But if you are an experienced graphics programmer, this should pose no issues.

Optimization Questions

- How can we use a Bounding Volume Hierarchy(or an octree, or something similar) to speed up a raytracer?
- Explain about Cache Memory(L1 and L2 caches, and so on)
- What is Data Oriented Design?
- Explain how view frustum culling can be optimized using multithreading and SIMD(see e.g. the blog post by Andreas Asplund)
- Do you have experience with using performance profiling tools for the GPU?

Game companies seem to expect you to have experience with profiling and optimizing your code using tools such as NVIDIA Nsight, so do get familiar with them. Also, this is a good book for brushing up on computer architecture topics such as cache memory.

Computer Graphics Questions

- What anti-aliasing techniques do you know about?(some possible techniques are MSAA, MLAA, FXAA and TXAA)
- What are the most common elements of a rendering engine?(common elements will be things like a system for handling culling, rendering of shadows, handling of light sources with something like deferred/forward shading, how materials are handled in the engine, and so on)
- What shadow rendering techniques do you know about?(there are TONS of shadow rendering techniques out there. Some examples are variance shadow mapping and exponential shadow mapping, and the newer moment shadow mapping)
- What are the pros and cons of a deferred renderer?
- Explain to me physically based rendering?
- Can you give an explanation of how the Rendering Equation works?
- What is a BRDF? What does it mean to say that a BRDF is "energy conserving"?
- What are the performance implications of branching in a shader?(hint: read up on the concept of a warp in GPU architecture)
- What advantages does newer API:s like Vulkan and DirectX 12 offer over old API:s like OpenGL and DirectX 11? (hint: primary reason is lower driver-overhead)
- What is the last graphics paper you read, and can you explain it to me?(this was a pretty common question)
- Describe to me the entire graphics pipeline?(your answer will probably be pretty long. You will explain about the vertex shader and the fragment shader, about perspective correct interpolation, about the z-buffer, about double buffering the framebuffer, about alpha blending, about transformation matrices, about homogeneous coordinates, about reflection models in the fragment shader and so on.)

The last question in particular seems to be pretty common in interviews for graphics programming jobs. If you are not entirely confident on all parts of the pipeline, then my general advice is that you write a small software rasterizer. This is an exercise that allows you to gain a deep understanding of the pipeline, because writing a software rasterizer is in many ways similar to implementing the graphics pipeline from scratch.

Many of the above questions should primarily be considered as warm-up questions. There will probably be some more detailed and involved questions after them(but I will not disclose them here, since the companies I interviewed for would probably not appreciate that). However, the above list of questions should give the reader a good understanding of what topics he/she will be expected to be proficient in.

Some readers may still be in school, and are unsure which courses they should be reading, if they are interested in a career in computer graphics. My recommendation is to primarily focus your efforts on courses that involve High Performance Computing, and Applied Mathematics. Regarding programming courses, try to read courses involving topics like multithreading, computer architecture, and GPGPU, and other topics that are related to High Performance Computing. As for math courses, primarily courses with names like "Linear Algebra", "Multivariable Calculus"(in USA, this course is called "Calculus III" I think), "Probability Theory", "Numerical Optimization", "Differential Equations", and "Computational Geometry" will be of interest to you. Primarily focus your attention on very applied mathematics, and not on more abstract math topics such as Topology and Abstract Algebra, as they are not that useful in computer graphics.

In general, you are not expected to know about every single subtopic of computer graphics .The label "computer graphics" covers a vast array of subtopics, such as global illumination, occlusion culling, shadow rendering, pathtracing, fluid simulation, geometry processing, GPGPU, physically based rendering, and so on. Expecting a junior programmer to know all these topics in detail would be unreasonable. However, if you put something on your CV, do be prepared to give detailed answers related to this topic.

So you are not expected to know EVERYTHING there is to know about computer graphics. Instead, what often will happen is that you are asked to explain about major computer graphics projects that you have worked on. Then the interviewer will be asking plenty of follow-up questions, and make you explain your project in detail. So my advice is that, before the interview, you pick out a couple of projects that you are especially proud over, and prepare yourself for giving a detailed explanation of these projects.

In my opinion, the best way to prepare for the interview, is to work on plenty of side projects in your spare time. Showcasing the projects you have worked on is a good way of selling yourself to the interviewer, and it allows you to show that you are truly passionate for this field. Finally, working on these side projects will allow you to sharpen your programming skills, and this in turn makes it much easier to answer all the questions I have listed in this article.