Q and A for Undergraduate Computer Graphics

Sung-Eui Yoon
Dept. of Computer Science
KAIST
sungeui@gmail.com
http://sglab.kaist.ac.kr

March 6, 2014

Abstract

Over the last five years I have received numerous questions from students who took my undergraduate computer graphics course, CS380. Some of them are asked repeatedly and I felt that providing my version of answers to those questions should be useful for students to get higher understanding on computer graphics. Note that these answers may not be correct and are not even designed to be correct answers. I wish that these answers serve as an initial starting point to those questions and help students to make more thought-provoking questions and embark deeper intellectual explorations on computer graphics.

1 Introduction Lecture

Do we need an excellent artistic sense to study computer graphics or to become an technical expert in this field?

Not really. Of course, it is always better to have a good artistic sense to work on visual data processing. However, if some jobs require such a high standard of artistic senses, those jobs may be for artistic designers, not for engineers. In my opinion, it is more important to have better engineering backgrounds (e.g., mathematical backgrounds and algorithm developments) and problem-solving skills. For example, I don't have any sense of art, but I work on computer graphics!

I have found that something like tea pot and bunny models are widely used in many papers and technical videos. Why?

You made a good observation. Some of models including the Utah teapot and Stanford bunny have been created earlier as research results or research benchmarks. Then, these models are distributed to other researchers for their follow-on research. That's why these models are widely used in many papers.

Are you taking students for UPR or independent research course?

Yes. If you are interested on some of my research area and have time to work on those area, I will seriously consider to work with you. For example, one student approached me to conduct a research after finishing CS380. We started to work on a research area, since I felt that the student is highly motivated and is willing to spend enough time to make progresses for a chosen research topic. After about one year, we submitted one paper (`http://sglab.kaist.ac.kr/ICAZ/`), which is accepted at Graphics Interface, a second-tier conference. Since I realized that the student is working really, I even asked one of my junior graduate students and another professor to work with us. Because of these team efforts, we were able to publish a paper in a high-quality conference, in a short

amount of time.

Are they any other courses related to computer graphics offered at KAIST?
Yes. Prof. YuWing Tai regularly offers a course on computer vision and image processing. Also, Prof. Jinah Park offers a course on computer animation. If you are interested, you can take those courses after or even before taking CS380.

Because I have prior classes right before CS380, I may be late a few minutes for CS380. I am worried whether I got lots of penalty because of the class attendance rule of CS380.
If you have a legitimate reasons explaining why you are late or absent, we won't give any penalty for you. However, you should notify earlier to me or TAs, before you will be late or absent because of avoidable events.

2 Different Spaces

Can glBegin () with GL_POLYGON can support concave polygons? According to its API description, GL_POLYGON works only with convex polygons. But, what may happen with concave polygons? Since it is not part of the specification of OpenGL, each vendor can have their own handling method for that kind of unspecified cases. If you are interested, you can try it out and let us know.

In the case of rendering circles, shown as an example in the lecture note, we render them by using lines. Is there a direct primitive that supports the circle? OpenGL has a limited functionality that supports continuous mathematical representations including circles, since a few model representations (e.g., triangles) have been widely used and it is hard to support all the possible representations. However, OpenGL keep changing and it may support many continuous functions in a near future. At this point of time, we need to discretize continuous functions with triangles or other simple primitive and render them.

We use the NDC between the world space and the screen space. Isn't it efficient? Also, don't we lose some precision during this process? There is certainly some overhead by introducing the NDC. However, it is very minor compared to its benefits in terms of simplifying various algorithms employed throughout the rendering process. Yes. We can lose more precision during the conversion process. However, it may be very small and may not cause significant problems for rendering purposes.

OpenGL is designed for cross-platform. But, I think that it means that we cannot use assembly programming for higher optimizations. Yes. You're right. We cannot use assembly languages for such optimizations. However, programmers for graphics drivers for each graphics vendor definitely use an assembly language and attempt to achieve the best performance. High-level programmers like us rely on such drivers and optimize programs with OpenGL API available to us.

Multi-threading with OpenGL: Since OpenGL has been designed very long time ago and has many different threads, it requires some cares to use multiple threads for OpenGL. There are many articles in internet about how to

use multiple threads with OpenGL. I recommend you to go over them, if you are interested in this topic.

Why do we use a viewport? The viewport space doesn't need to be the whole window space. Given a window space, we can decompose it into multiple sub-spaces and use sub-spaces for different purposes. An example of using multiple viewports is shown in Fig. 1.

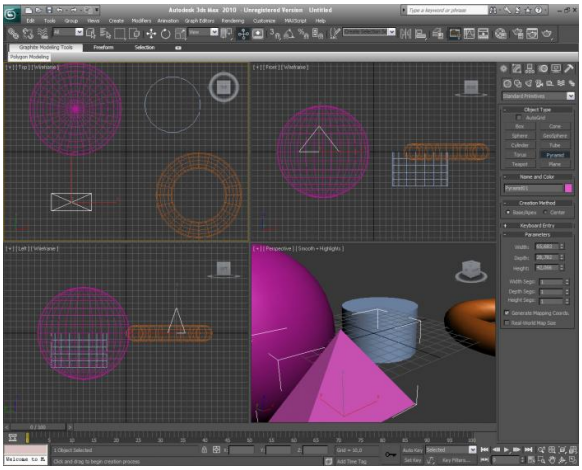

Figure 1: This figure shows multiple viewports, each of which shows an arbitrary 3D view in addition to top, front, and side views. The image is excerpt from screenshots.en.sftcdn.ne.

3 Basic OpenGL Structure

What if we have new input devices (e.g., joystick, or multiple input devices used in PlayStation or XBox)? How can we handle those devices in OpenGL programs?

OpenGL does not have any functionality to support those various input devices. GLUT library supports some of basic input devices such as keyboard and mouses. For other devices, you need to use other external libraries that support those devices.

In what cases, is OpenGL used rather than DirectX?

OpenGL is cross-platform graphics API, while DirectX is proprietary library for Windows. Because of the openness of OpenGL, it, more specifically, OpenGL ES, is widely used for many embedded systems including mobile phones.

In what portions of my OpenGL program are executed in CPU and GPU?

In a typical OpenGL program, rendering parts (e.g., portions started with glBegin and ended with glEnd) are performed in GPU, graphics hardware, if your computer is equipped with such GPU. All the control parts, e.g., calling OpenGL functions and handling events, are performed in CPU. In other words, various functionality inside OpenGL APIs are commonly performed in GPU, while all the other parts are performed in CPU.

4 2D Transformations

Is there any benefit of using column-major ordering for the matrix over row-major ordering?

Not much. Some people prefers to use column-major, while others like to use row-major. Somehow, people who designed OpenGL may prefer column-major ordering.

5 Viewing Transformation

Can we support other projections than orthographic and perspective projections? For example, a projection simulating the image observed from bug's eyes? What if this projection is not represented as a simple matrix?

Yes, we can support many other projections that are represented as some mathematical equations. Also, current GPU can support arbitrary projections although the projection is not represented as a simple matrix.

I felt that there are something missed in the image generated by using perspective projection. Then, I realized that those images do not have effects like out-of-focusing and in-focusing. How can we support these effects?

To correctly simulate these kinds of effects, we need to simulate a lens that we are using in camera. This can be supported by using ray tracing, but may take long computation time. Instead, we can mimic similar effects by considering depth values of rasterized objects. For example, the depth values of the rasterized objects are far away from the user-defined focal depth, we blur the image of the object. This is not a correct solution, but a hacky solution that can run very fast in the rasterzation rendering mode.

6 Clipping and Culling

Even though some objects are outside the view frustum, they can be seen though transparent objects or reflected from mirrors.

Exactly. The rasterization algorithm is a drastically simplified rendering algorithm over the real interactions between lights and materials. The direct illumination, seen thought primary rays, are well captured by rasterization, while other indirect illuminations are not captured well in the rasterization. To address this problem, many techniques have been proposed in the field of rasterization. However, the most natural way of handling them is to use ray tracing based rendering algorithms.

7 Illumination

We have learned that we compute an illumination value for each vertex. For smooth objects (like the teapot model shown in the slide), it should be okay. But, when we draw a box, then the box may look smooth, not showing different and discontinuous colors between neighboring faces of the box.

If we use a normal for each vertex of the box, we may get such smooth rendering result, which is not correct one. Instead, we use multiple vertices for each point of the box. For example, we use different vertex data for each point in

different faces of the box. Note that these vertices should have the same positional value, but they can have different normals, which can generate discontinuous colors between different faces. Refer to the slide of ”Decoupling Vertex and Face Attributes via Indirection” in the lecture slide of ”Interacting with a 3D world”.

Is there any techniques that can show better quality than the Phong Illumination and can be used in interactive games? I want to know techniques that can show near physically-based illumination that can be used in games?

Ambient occlusion has been proposed as an approximation for physically-based global illumination. It can be pre-computed and used quite quickly at runtime, leading to be suitable for interactive games. Moreover, in some CG movies, this technique has been used.