CS380 Introduction to Computer Graphics
Homework #7

20160042 Inyong Koo

Describe the following terms with respect to computer graphics.

1. Buffer

   We can define a (two-dimensional) **buffer** as a block of memory with $n \times m$ k-bit elements. All buffers are inherently discrete: They have limited resolution, both spatially and in depth.

2. Frame buffer

   Frame buffer is the set of buffers that the graphics system uses for rendering, including the front and back color buffers, the depth buffer, and other buffers the hardware may provide.

3. Precision of a buffer

   The numerical accuracy or **precision of a buffer** is determined by its depth. For example, we have 64 bits for the front and back buffers and 32 bits for the depth buffer. Thus, if a frame buffer has 32 bits each for its front and back color buffers, each RGBA color component is stored with a precision of 8 bits.

4. Bit block transfer (bitblt)

   We tend to read and write rectangular blocks of pixels (or bits), known as bit blocks. It is important to have both hardware and and software support a set of operations that work on rectangular blocks of pixels, known as **bit-block transfer operation (bitblt)**, as efficiently as possible. These operations are also known as raster operations (raster-ops).

5. Texture mapping

   Texture mapping uses an image (or texture) to influence the color of a fragment. Textures can be specified using a fixed pattern, such as the regular patterns often used to fill polygons; by a procedural texture-generation method; or through a digitized image.

6. Texture coordinates

   textures, either formed by application programs or scanned from a photography, are eventually brought into processor memory. The elements of these arrays are called *texels*, or texture elements $T(s,t)$, are represented in **texture coordinates** s and t.

7. Sampler variable

   We define a new type of variable called a sampler, which most often appears only in a fragment shader. A **sampler variable** provides access to a texture object, including all its parameters.

8. Magnification problem

   The size of the pixel that we are trying to color on the screen may be smaller or larger than one texel. If the texel is larger than one pixel, it's called **magnification problem**. We can solve it by using the value of the nearest point sampling.

9. Minification problem

   If the texel is smaller than one pixel, it's called **minification problem**. It can also be resolved by using the value of the nearest point sampling. (continued to next item)

10. Mipmapping

    Another way to deal with the minification problem is **mipmapping**. For objects that project to an area of screen space that is small compared with the size of the texel array, we do not need the resolution of the original texel array. OpenGL allows us to create a series of texture arrays (mipmap) at reduced sized; it will then automatically use the appropriate size.

11. Multitexturing

    We can apply multiple texture to an object. recent versions of OpenGL supports **multitexturing** using multiple texture units. Each unit acts as an independent texturing stage starting with the results of the previous stage.

12. Environment map (or reflection map)

    We cannot render an object with highly reflective surfaces correctly without knowing about the rest of the scene. This effect requires global information. However, the ray-tracing calculations are too time-consuming to be practical for real-time applications. Instead, we use variants of texture mapping that can give approximate results that are visually acceptable through **environment maps** or **reflection maps**.

13. Sphere mapping

    In the original version of environment mapping, the surface of the sphere was then converted to a rectangle using lines of longitude and latitude for the mapping. Although conceptually simple, there are problems at the poles where the shape distortion becomes infinite. Computationally, this mapping does not preserve areas very well and requires evaluating a large number of trigonometric functions.

    OpenGL supports a variation of this method called **sphere mapping**. The application program supplies a circular image that is the orthographic projection of the sphere onto which the environment has been mapped. The advantage of this method is that the mapping from the reflection vector to two-dimensional texture coordinates on this circle is simple and can be implemented in either hardware and software. The difficult part is obtaining the required circular image. It can be approximated by taking a perspective projection with a very wide-angle lens or by remapping some other type of projection.

14. Multipass rendering (or multirendering)

    In order to compute a single image, we compute multiple images, each using the rendering pipeline. **Multipass rendering** methods are becoming increasingly more important as the power of graphics card has increased to the point that we can render a scene multiple times from different perspectives in less time than is needed for reasonable refresh rates.

15. Bump mapping

    The technique of **bump mapping** varies the apparent shape of the surface by perturbing the normal vectors as the surface is rendered; the colors that are generated by shading then show a variation in the surface properties. Bump mapping cannot be done in real time without programmable shaders.

16. Normal map

We have a sampled version of displacement function $d(x, y)$ as an array of pixels $\mathbf{D} = [d_{ij}]$ The required partial derivatives to specify the bump map, can be approximated by the difference between adjacent elements. These arrays can be precomputed in the application and stored as a texture called a **normal map**.

17. Alpha bending

OpenGL provides a mechanism, through **alpha bending**, that can, among other effects, create images with translucent objects. The α channel is the fourth color in RGBA color mode. Like the other colors, the application program can control the value of A(α) for each pixel. In RGBA mode, if blending is enabled, the value of α controls how the RGB values are written into the frame buffer. Because fragments from multiple objects can contribute to the color of the same pixel, we say that these objects are blended or composited together.

18. Multisampling

Rather than antialiasing individual lines and polygons, we can antialias the entire scene using a technique called **multisampling**. In this mode, every pixel in the frame buffer contains a number of samples. Each sample is capable of storing a color, depth, and other values.

19. Motion blur

If we jitter an object and render it multiple times, leaving the position of the other objects unchanged, we get dimmer copies of the jittered object in the final image. If the object is moved along a path, rather than randomly jittered, we see the trail of the object. This motion blur effect is similar to the result of taking a photograph of a moving object using a long exposure time. We can adjust the object's α value so as to render the final position of the object with greater opacity or to create the impression of speed differences.

20. Nyquist sampling theorem

We can explain the consequences of sampling, without being overwhelemed by the mathematics, if we accept the fundamental theorem known as the Nyquist sampling theorem. There are two parts to the theorem: The first allow us to discuss sampling errors, where as the second governs reconstruction.

Part 1 : The ideal samples of a continuous function contain all the information in the original function if and only if the continuous function is sampled as a frequency greater than twice the highest frequency in the function.

Part 2 : We can reconstruct a continuous function $f(x)$ from its samples $\{f_i\}$ by the formula

$$f(x) = \sum_{i=-\infty}^{\infty} f_i \text{sinc}(x - x_i)$$

The function sinc(x) is defined as $\text{sinc}(x) = \dfrac{\sin \pi x}{\pi x}$.

The two-dimensional version of the reconstruction formula for a function $f(x, y)$ with ideal samples $\{f_{ij}\}$ is

$$f(x, y) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} f_{ij} \text{sinc}(x - x_i)\text{sinc}(y - y_i)$$