CS380 Introduction to Computer Graphics
Homework #2

20160042 Inyong Koo

Describe the following terms with respect to computer graphics.

1. Vertex

   A **vertex** is a position in space. We use vertices to specify the atomic geometric primitives that are recognized by our graphics system. It is slightly different from *point*, which is the simplest geometric primitive, usually specified by a single vertex.

2. Clip coordinates

   Clip coordinates system, is one that our vertex shader uses to send information to the rasterizer. Objects outside this cube will be clipped and cannot appear on the display.

3. World coordinates

   The advent of device-independent graphics freed users from worrying about the details of input and output devices and work in any coordinate system that they desires. The user's coordinate system became known as the **world coordinate** system.

4. Object coordinates

   Object coordinate system, or *application* is another name for world coordinates system. we may use any numbers that fit our application in object coordinate system.

5. Window coordinates

   Window coordinates or *screen coordinates* are device coordinates for raster devices, such as most CRT and flat panel displays. Window coordinates are always expressed in some integer type.

6. Primitives

   Primitives are the low-level objects or atomic entities that our system can display. Depending on the API, the primitives can include points, line segments, polygons, pixels, text, and various types of curves and surfaces.

7. Attributes

   Attributes govern the way that a primitive appears on the display. If primitives are the *what* of an API, then attributes are the *how*. Available attributes depend on the type of object.

8. Simple polygon

   Polygon is an object that has a border that can be described by a line loop, and has a well-defined interior. In two dimensions, we have a **simple polygon** if no two edges of the polygon cross each other.

9. Strips and Fans

These objects are based on groups of triangles that share vertices and edges. In the triangle **strip**, each additional vertex is combined with the previous two vertices to define a new triangle. A triangle **fan** is based on one fixed point. The next two points determine the first triangle, and subsequent triangles are formed from one new point, the previous point, and the first (fixed) point.

10. Additive / subtractive color

With **additive color** system, primaries (usually red, green, and blue) add light to an initially black display, yielding the desired color.
Subtractive color model is more appropriate for processes such as commercial printing and painting. Color pigment remove color components from light that is striking the surface. The primaries of subtractive systems are usally the complementary colors: cyan, magenta, and yellow.

11. Indexed-color

Indexed-color model was used in early graphics systems due to its lower memory requirements and limited colors available on displays Colors can be selected by interpreting limited-depth pixels as indices into a table of colors rather than as color values.

12. Alpha (as in 4^{th} color)

In four-color (RGBA) system, **Alpha** have various uses, such as combining images. In an OpenGL program, alpha value is treated as either an opacity or transparency value.

13. Orthographic view

Orthographic view is OpenGL's default view. Mathematically, the orthographic projection is what we would get if the camera in our synthetic-camera model had an infinitely long telephoto lens and we e could then place the camera infinitely far from our objects.

14. Viewing volume

In computer graphics, our synthetic-camera can see only those objects in the volume specified by **Viewing volume**.

15. Vertex-array object

Vertex-array object allow us to bundle data associated with a vertex array.
We use `glGenVertexArray` to find an unused name for the buffer. The first time `glBindVertexArray` is executed for a given name, the object is created.

16. Buffer object

We create **buffer object** on the GPU and place our data in that object.
We use `glGenBuffers` to give an unused identifier for our buffer object. `glBindBuffer` creates the buffer with the identifier from `glGenBuffer`. `glBufferData` allocates sufficinet memory on the GPU for our data and provide a pointer to the array holding the data.

17. Program object

Program object is a container that has an integer identifier we can use to refer to it in the application. We can attatch the shaders to the program object.

18. Window event

Window events are window associated events, such as resizing a window, iconifying a window, and exposing a window that was covered by another window.

19. Idle callback

The **idle callback** is invokd when there are no other events. Its default is the null function pointer. We may use the idle callback to continue to generate graphical primitives through a display function while nothing else is happening, or to produce an animated display.

20. Double buffering

An application program operates asynchronously and can cause changes to the frame buffer at any time. Hence, a redisplay of the frame buffer can occur when its contents are still being altered by the application and the viewer will see only a partially drawn display.
In order to solve this problem, we use **double buffering**. We define two frame buffers; the front buffer, is one that is displayed. The back buffer, is then available for constructing what we would like to display. Once the drawwing is complete, we swap the front and back buffers. Then we clear the new back buffer and can start drawing into it.