**CST325 HW 4.1 – Graphics Pipeline**

(weighted to 50pts)

1. What is the difference between geometry represented using polygon meshes versus the implicit surfaces we used in the raytracer? (2pts)

The difference between geometry represented using polygon meshes versus implicit surfaces is that implicit surfaces code the object directly through some mathematical formula based on the shape of that object. Polygon meshes are a series of connected triangles that can form almost any shape given enough of them.

1. Why do we need a “near” plane in rasterization? (3pts)

We need a “near” plane in rasterization because as the computer chooses z values that get infinitely closer to 0, the w values will become very large which can cause problems with perspective correct rasterization. The “near” plane fixes this error by creating a known minimum for the z values.

1. Why do we typically use a “far” plane in rasterization? (3pts)

We use a “far” plane in rasterization to limit the number of objects that need to be rendered. The “far” plane also establishes what z value in camera space will be the maximum value that can be stored in the depth buffer.

1. What are the different spatial reference frames that each transform in the pipeline will take us through and what matrix do we use to get there from the previous space? (4pts)

The Object Space is the start and is transformed to the world space using the world matrix. Further down the pipeline, we apply the view matrix to the world space to get to the view space, then apply the projection matrix to get to the clip space, and finally apply the viewpoint matrix to get to screen space.

1. Where is the origin located in Object Space? (4pts)
   1. The center of an object
   2. Toward the eye point with a unit length of 1
   3. Can be anywhere
   4. At the eye (camera) point
   5. At the center of the viewing volume
2. Where is the origin located at in World Space? (4pts)
   1. The center of an object
   2. Toward the eye point with a unit length of 1
   3. Can be anywhere
   4. At the eye (camera) point
   5. At the center of the viewing volume
3. Where is the origin located at in View Space? (4pts)
   1. The center of an object
   2. Toward the eye point with a unit length of 1
   3. Anywhere
   4. At the eye (camera) point
   5. At the center of the viewing volume
4. Where is the origin located in Clip Space? (4pts)
   1. The center of an object
   2. Toward the eye point with a unit length of 1
   3. Anywhere
   4. At the eye (camera) point
   5. At the center of the viewing volume
5. What is the painter’s algorithm? (2pts)

The painter’s algorithm is where you start with the background elements of a picture and then add in the foreground, basically overlaying things that would naturally be on top of each other.

1. Why doesn’t the painter’s algorithm ensure opaque objects are rendered in the correct order? (2pts)

The painter’s algorithm has a problem where if you must paint the entire object at once with no interruption, at least one object will always be “on top” meaning that it will never be behind anything. Therefore, if all the objects are overlapping each other at some point, this algorithm will render them incorrectly.

1. If depth is only initially computed per vertex, how do we get depth values for every fragment? (2pts)

As we render, we are constantly updating fragment’s depths based upon the z buffer, but only when the current depth is less than the closest rendered fragment at the current pixel.

1. Describe in your own words how a vertex becomes a pixel. In other words, how does the graphics pipeline enable a seemingly abstract collection of points to be converted into an image onscreen? Be as thorough as possible (10 pts)

All pixels were originally born vertexes. It is at the vertex stage that geometric transformations and attributes are setup such as color, normal, relative position, etc.

The first major transformation vertexes undergo is becoming Primitives. Primitives are triangles and begin to form the shapes of the world we are viewing.

The first space primitives inhabit is object space. Then, through the magic of matrix transformation, they are transformed into world space (The shared space of our view).

Once the space is shared, we again transform the world space into the View Space. Relative distances among objects remain, but now our origin becomes the camera from which we will be viewing.

Then comes along the Projection Matrix and again transforms the View Space into Clip Space. Space is distorted into a cube getting us ready for the geometry processing stage. This cube is scaled and biased to fit your screen (lucky you!) Here, the 3rd dimension is only kept to determine what is closer to you, but the rest of our work is carried out in 2 dimensions.

Many primitives will be rejected due to culling. This means to take any primitive that will not be visible to us in the scene and cut them out of the rest of the pipeline. This includes pieces of triangles off screen, as well as surfaces that are facing away from us.

It is at this point the primitive is REBORN through the process of Rasterization. During scanline rendering or perhaps sampling, it becomes a collection of pixels. There is no going back from here. The mathematical perfectness of the primitive is now in a pixilated world.

Again we will be cutting away fragments. If the depth according to the z buffer is further away than the closes already known fragment is discarded.

At the next stage, we will shade the remaining fragments with per fragmentation computation.

At this point, we’re looking really great and ready to enter into our frame buffer and finally, yes finally get outputted to the sweet screen as a truly gorgeous pixel.

Then after 1/60th of a second, our pixel life is over and we are replaced with the next pixel from the pipeline. Oh well.