Solutions to Homework 1

Solution 1: (a) A triangle strip uses fewer calls to glVertex and hence is more efficient since fewer vertices need to be processed. (b) The advantage of using GL_TRIANGLES is that a vertex shared between two triangles results in (at least) two calls to glVertex. This means that it is possible to provide two different normals and/or two different texture coordinates for this vertex. This is useful for shading/coloring effects where we want to have a clear discontinuity between two triangles. Because you have control over the ordering of vertices, it is also possible to alter the orientations of individual triangles with GL_TRIANGLES. This might be useful, for example, if you use different front and back face colors.

Solution 2:

(a) A single axis-parallel bounding rectangle is the simplest of all of these to process. The reason is that it involves only comparisons between x- and y-coordinates to determine overlap. It does not provide a good approximation, for objects that have long protrusions that are not axis aligned (e.g., a thin spear tilted at 45 degrees).

(b) A union of a small number (say 1–5) axis-parallel bounding rectangles involves more processing time than a single rectangle (proportional to the number of rectangles), but it helps remedy some of the shortcomings of a single axis-aligned box if the object in question can be modeled in this manner. For example, a sprite that consists of a standing person who can lunge horizontally with a sword can be modeled as two boxes, one for the body and one for the sword. As in (a), if the object consists of a tilted features, then nothing based on axis-aligned boxes provides a very tight approximation.

(c) A rotated rectangle involves more complexity of processing than an axis-aligned rectangle. Determining the intersection of two such rectangles involves considerably more computational effort. (In fact, there is little advantage over this representation than any representation based on a polygonal enclosure where the polygon has a small number of sides.) This representation provides a significantly better approximation for tilted objects, especially if the object is convex.

(d) A circle

Solution 3: Let's let $b'_{ij}(u)$ denote the derivative of $b_{ij}(u)$ with respect to u. Recall that

$$\begin{array}{llllllllll} b_{0,2}(u) & = & (1-u)^2 & b_{1,2}(u) & = & 2(1-u)u & b_{2,2}(u) & = & u^2 & \\ b_{0,3}(u) & = & (1-u)^3 & b_{1,3}(u) & = & 3(1-u)^2u & b_{2,3}(u) & = & 3(1-u)u^2 & b_{3,3}(u) & = & u^3, \end{array}$$

and $b_{i,j}(u) = 0$ if $i < 0$ or $i > j$. We will show that, for $0 \le i \le 3$, $b'_{i,3}(u) = 3(b_{i-1,2}(u) - b_{i,2}(u))$.

First, let's consider $b'_{0,3}(u)$. Since $b_{0,3}(u) = (1-u)^3 = 1 - 3u + 3u^2 - u^3$, we have

$$\begin{aligned} b'_{0,3}(u) &= -3 + 6u - 3u^2 = -3(1 - 2u + u^2) = -3(1-u)^2 \\ &= 3(0 - (1-u)^2) = 3(b_{-1,2}(u) - b_{0,2}(u)). \end{aligned}$$

Next, consider $b'_{1,3}(u)$. Since $b_{1,3}(u) = 3(1-u)^2u = 3u - 6u^2 + 3u^3$, we have

$$\begin{aligned} b'_{1,3}(u) &= 3 - 12u + 9u^2 = 3(1 - 4u + 3u^2) = 3((1 - 2u + u^2) - (2u - 2u^2)) \\ &= 3((1-u)^2 - 2(1-u)u) = 3(b_{0,2}(u) - b_{1,2}(u)). \end{aligned}$$

Next, consider $b'_{2,3}(u)$. Since $b_{2,3}(u) = 3(1-u)u^2 = 3u^2 - 3u^3$, we have

$$\begin{aligned} b'_{2,3}(u) &= 6u - 9u^2 = 3(2u - 3u^2) = 3((2u - 2u^2) - u^2) \\ &= 3(2(1-u)u - u^2) = 3(b_{1,2}(u) - b_{2,2}(u)). \end{aligned}$$

Finally, consider $b'_{3,3}(u)$. Since $b_{3,3}(u) = u^3$, we have

$$b'_{3,3}(u) \;=\; 3u^2 \;=\; 3(u^2 - 0) \;=\; 3(b_{2,2}(u) - b_{3,2}(u)).$$

Solution 4:

(a) The classifier "varying" means that the quantities computed at each vertex are interpolated at each fragment (pixel) in the final rendered object.

(b) Normalizing in the vertex shader is not sufficient. The reason is that you would need to perform a spherical linear interpolation (SLERP), but GLSL does not know what you intend to use the normal vectors for and just performs a simple linear interpolation. If the angle between the vertex normals is very large (close to 180°, then computing a linear interpolation may produce a much shorter vector than a spherical interpolation. By normalizing the length of the vector in the fragment shader, we can correct for this effect.

(c) The variables gl_Position and gl_FragColor are not used by our program, but they are required by the system. The first gives the position of the vertex relative to the camera coordinates, and the second gives the fragment's color.

Solution 5:

(a) If separation is not implemented, the cohesion force will eventually cause the boids to converge about a single point.

(b) If alignment is not implemented, the boids will tend to swarm about a common location (due to cohesion) but they will not maintain a common direction. Thus, they will tend to look more like bugs flying randomly about a light than a flock of birds.

(c) If cohesion is too low, then after any incident that cause the boids to disperse (such as avoidance to a predator) the boids may take an extremely long to regroup into a flock.

(d) If avoidance is set too high, then any obstacle or predator can generate a ridiculously large force, which will cause the boids affected by the avoidance force to be ejected a large distance from the flock.