Homework 1

Handed out Tue, Apr 12. Due at the start of class Thu, Apr 21. Late submissions will not be accepted, so turn in whatever you have finished. If you will not be in class, it is your responsibility to see that it is delivered to me before the due date. You may use external public sources you like. You may discuss problems with your classmates, but your solution and write-ups must be done individually.

Problem 1: A given mesh can be drawn using either a collection of triangles using GL_TRIANGLES or as a triangle strip using GL_TRIANGLE_STRIP.

(a) Give a reason why a user might prefer to use GL_TRIANGLE_STRIP.

(b) Give a reason why a user might prefer to use GL_TRIANGLES. (Hint: The reason does not involve efficiency.)

Problem 2: For the purposes of collision detection in 2-dimensional space, consider the following methods:

(a) A single axis-parallel bounding rectangle

(b) The union of a small number (say 1–5) axis-parallel bounding rectangles

(c) A rectangular that may be rotated

(d) A circle

Briefly explain the advantages and disadvantages of each methods in terms of the accuracy of approximation and processing time.

Problem 3: In class we presented the blending functions for the Bezier curves of degrees 0, 1, 2 and 3:

$$\begin{array}{llllllllll} b_{00}(u) & = & 1 & & & & & & & \\ b_{01}(u) & = & (1-u) & b_{11}(u) & = & u & & & & \\ b_{02}(u) & = & (1-u)^2 & b_{12}(u) & = & 2(1-u)u & b_{22}(u) & = & u^2 & \\ b_{03}(u) & = & (1-u)^3 & b_{13}(u) & = & 3(1-u)^2u & b_{23}(u) & = & 3(1-u)u^2 & b_{33}(u) = u^3. \end{array}$$

We can generalize this by defining $b_{ij}(u) = 0$ if $i < 0$ or $i > j$. An interesting mathematical fact is that up to a constant multiplicative factor, the derivative of any Bezier blending function can be expressed as the difference of two blending functions of lower degree. To illustrate this, compute the derivatives of $b_{03}(u)$, $b_{13}(u)$, $b_{23}(u)$, and $b_{33}(u)$ with respect to u and, in each case, show that (up to a constant factor) the result can be expressed as the difference of two blending functions of lower degree.

Problem 4: Consider the following code, which we presented in class for a Phong lighting vertex shader:

```
varying vec3 normal;
varying vec3 toLight;

void main() {
  gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;
  normal = gl_NormalMatrix * gl_Normal;
  vec4 viewVertex = gl_ModelViewMatrix * gl_Vertex;
  toLight = vec3(gl_LightSource[0].position  viewVertex);
}
```

and the associated fragment shader:

```
varying vec3 normal;
varying vec3 toLight;

void main() {
  const vec4 AmbientColor = vec4(0.1, 0.0, 0.0, 1.0);
  const vec4 DiffuseColor = vec4(1.0, 0.0, 0.0, 1.0);
  vec3 uNormal = normalize(normal);
  vec3 uToLight = normalize(toLight);
  float diffuseTerm = clamp(dot(uNormal, uToLight), 0.0, 1.0);
  gl_FragColor = AmbientColor + DiffuseColor * DiffuseTerm;
}
```

Answer the following question about this code.

(a) What does the classifier "varying" mean in the declarations of the variables normal and toLight?

(b) We computed the normalized vectors uNormal and uToLight in the fragment shader. Why didn't we just save time and do this in the vertex shader?

(c) We computed the variables gl_Position and gl_FragColor, but we never used either. Why did we do this?

Problem 5: Recall that in our discussion of flocking and boids, we introduced the following constraints on the motions of boids: separation, alignment, cohesion, avoidance. What would the resulting motion look like under each of the following modifications:

(a) Suppose that all constraints were implemented except separation.

(b) Suppose that all constraints were implemented except alignment.

(c) Suppose that all constraints were implemented, but the cohesive component of the force was set way too low.

(d) Suppose that all constraints were implemented, but the avoidance component of the force was set way too high.