CS 4620 Final Exam

Wednesday 9, December 2009—2$\frac{1}{2}$ hours

Explain your reasoning for full credit.
You are permitted a double-sided sheet of notes.
Calculators are allowed but unnecessary.

Problem 1: Continuity (8 pts)

You have learned about parametric and geometric continuity. For each 2D curve, answer the continuity query as correctly as possible, and provide a *brief* explanation:

(a) Is a circle C^0 continuous?

(b) Is a circle G^0 continuous?

(c) Is a circle C^∞ continuous?

(d) Is a circle G^∞ continuous?

(e) Is a square C^0 continuous?

(f) Is a square G^0 continuous?

(g) Is a square C^1 continuous?

(h) Is a square G^1 continuous?

Problem 2: View Frustum Culling (10 pts)

"View frustum culling" is a technique to avoid drawing (or cull) geometry which is outside the view frustum. To assist with culling, assume that *each object has a bounding sphere* with object-frame center position, $\mathbf{c}_o = (c_x, c_y, c_z, 1)^T$, and radius R_o. Imagine that you know you have an $[l, r] \times [b, t] \times [f, n]$ orthographic viewing volume, and you know each of the matrices ($\mathbf{M}_{vp}$, $\mathbf{M}_{orth}$, $\mathbf{M}_{cam}$, $\mathbf{M}_m$) used to construct the orthographic view transformation which maps points from *object space* to *screen space*:

$$\mathbf{p}_s = \begin{pmatrix} x_s \\ y_s \\ z_c \\ 1 \end{pmatrix} = \mathbf{M}_{vp}\,\mathbf{M}_{orth}\,\mathbf{M}_{cam}\,\mathbf{M}_m\,\mathbf{p}_o = \mathbf{M}\begin{pmatrix} x_o \\ y_o \\ z_o \\ 1 \end{pmatrix}.$$

Derive a simple mathematical test to determine if an object is safely "off screen."

Problem 3: Rasterizing Curves (15 pts)

In this question you will extend Bresenham's midpoint algorithm for line rasterization to build a DDA-based rasterizer for a *quadratic Bézier curve*. For simplicity you may assume that the curve is parameterized in the form

$$y(x) = y_0 B_0(x) + y_1 B_1(x) + y_2 B_2(x),$$

where

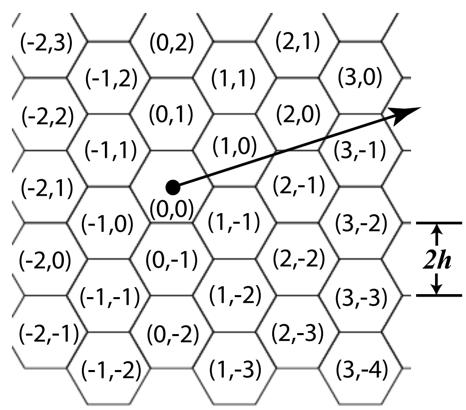
$$B_i(x) = \binom{2}{i} x^i (1-x)^{2-i}$$

are the quadratic Bernstein polynomials. You may even assume that the slope of the curve satisfies $0 \leq y'(x) \leq 1$.

(a) First, derive the equations needed to use **forward differencing** to evaluate the Bézier curve at unit $\Delta x = 1$ spacings without unnecessary multiplication. (Hint: First convert $y(x)$ to monomial form.)

(b) Second, provide pseudocode for a simple DDA rasterizer from $x = x_0$ to $x = x_1$. You need not consider shading, attribute interpolation, or antialiasing—you only need to "turn on" pixels using appropriate calls to `output(x,y)`.

Problem 4: Tracing rays through hexagonal subdivisions (12 pts)



You have seen how to trace a ray through a square grid in 2D, and even a voxel grid in 3D. In this question you will consider 2D hexagonal grids. Analogous to rectangular grids, assume that the hexagonal cells have an (i, j) indexing as shown in the figure. Assume that each hexagon's parallel edges are $2h$ apart (see figure).

Propose an efficient pseudocode implementation to trace the ray through an infinite hexagonal subdivision, making calls to `output(i,j)` indices of hexagons traversed. For simplicity, assume that the ray $\boldsymbol{r}(t) = \boldsymbol{e} + t\boldsymbol{v}$, $t \geq 0$, starts at the *center* of cell $(i, j) = (0, 0)$ as shown in the figure. Ignore boundaries.

Problem 5: Phong Tesselation (20 pts)

Recall that *Phong Shading* interpolates vertex normals across a triangle for smooth shading on low-resolution meshes, i.e., the unnormalized surface normal at barycentric coordinate (u, v, w) (where $w = 1 - u - v$) is approximated by barycentrically interpolated vertex normals,

$$\boldsymbol{n}'(u, v) = u\boldsymbol{n}_i + v\boldsymbol{n}_j + w\boldsymbol{n}_k,$$

where the unit vertex normals are $\boldsymbol{n}_i$, $\boldsymbol{n}_j$ and $\boldsymbol{n}_k$. Of course, since each triangle is still planar,

$$\boldsymbol{p}(u, v) = u\boldsymbol{p}_i + v\boldsymbol{p}_j + w\boldsymbol{p}_k,$$

the piecewise planar shape is still apparent at silhouettes.

Recently, Boubekeur and Alexa [SIGGRAPH Asia 2008] introduced *Phong Tesselation* as a simple way to use vertex normals to deform a triangle mesh to have smoother silhouettes (see Figures 1 and 2). In the following, you will derive their formula for a curved triangle patch, $\boldsymbol{p}^*(u, v)$, and analyze surface continuity.

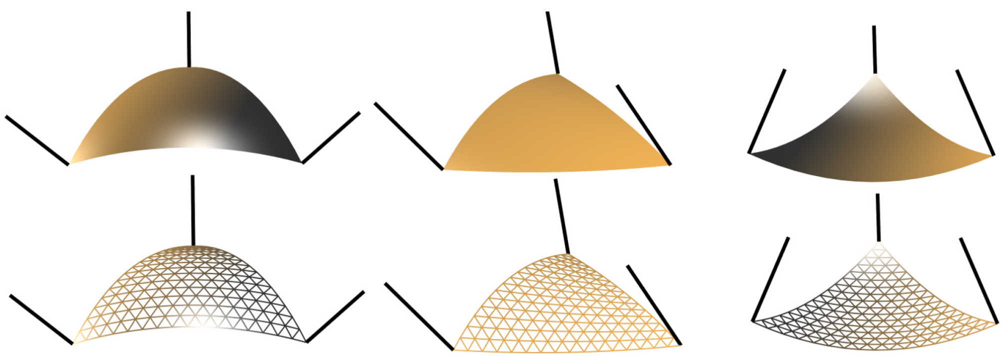


Figure 1: Phong Tesselation Examples: A triangle deformed with different vertex normals.

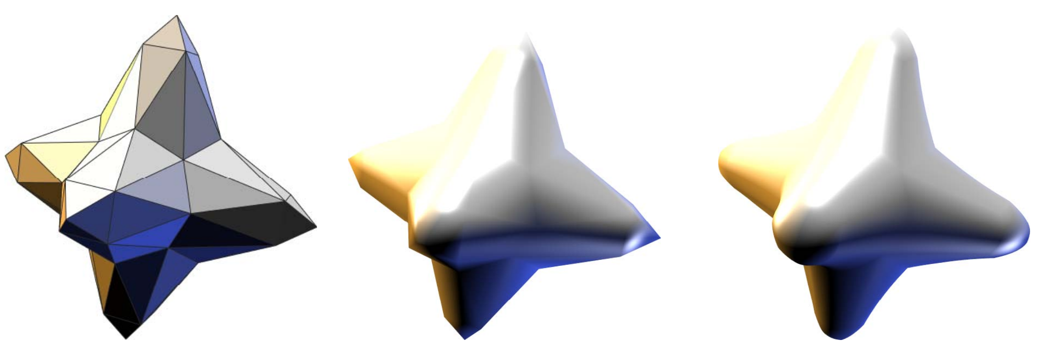
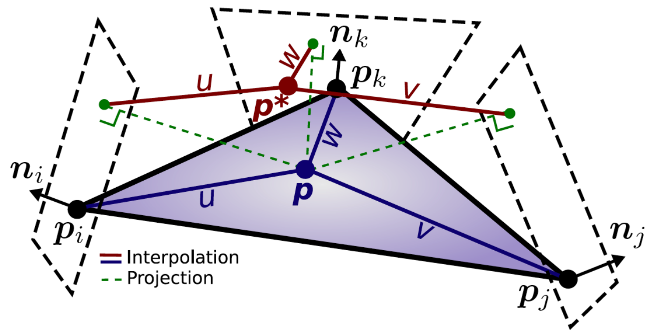


Figure 2: Phong Tesselation

Answer the following four questions:

(a) Consider the plane passing though vertex i's position, $\boldsymbol{p}_i$, and sharing the same normal, $\boldsymbol{n}_i$. Give an expression for the orthogonal projection of a point $\boldsymbol{p}$ onto vertex i's plane, hereafter denoted by $\boldsymbol{\pi}_i(\boldsymbol{p})$.

(b) The deformed position $\boldsymbol{p}^*(u, v)$ is simply the barycentrically interpolated projections of the undeformed point $\boldsymbol{p}(u, v)$ onto the three vertex planes, i.e., the barycentric interpolation of $\boldsymbol{\pi}_i(\boldsymbol{p}(u, v))$, $\boldsymbol{\pi}_j(\boldsymbol{p}(u, v))$, and $\boldsymbol{\pi}_k(\boldsymbol{p}(u, v))$. Derive a polynomial expression for $\boldsymbol{p}^*(u, v)$ in terms of u, v and w—you can also write it only in terms of u and v but it is messier. (Hint: Express your answer in terms of projected-vertex positions, such as $\boldsymbol{\pi}_i(\boldsymbol{p}_j)$.)



(c) What degree is this triangular bivariate polynomial patch, $\boldsymbol{p}^*(u, v)$?

(d) Given a triangle mesh that is converted to these polynomial patches, consider the parametric continuity of the resulting spline surface:

(i) Show that the surface is G^0 continuous.

(ii) Show that the surface is not G^1 continuous.