CS 4620 Final Exam (Prof. James)

Thursday 9 December 2010—2.5 hours

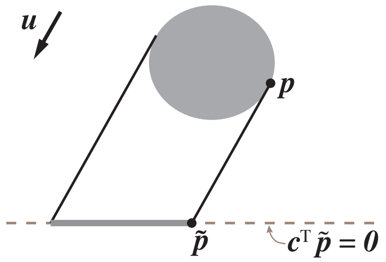
Explain your reasoning for full credit.
You are permitted a double-sided sheet of notes.
Calculators are allowed but unnecessary.

Problem 1: Shading (10 pts)

Given a triangle mesh, briefly explain the key differences in how *triangle normals* are used to evaluate (i) flat shading, (ii) Gouraud shading, and (iii) Phong shading.

Problem 2: Planar Shadows (15 pts)

Consider a directional light source (unit direction, $\boldsymbol{u}$), and the planar shadow created by literally projecting each mesh vertex position, $\boldsymbol{p}$, to its shadow point $\tilde{\boldsymbol{p}}$ on the 3D plane specified (in homogeneous coordinates) by $\boldsymbol{c}^T\tilde{\boldsymbol{p}} = 0$ where $\boldsymbol{c} \in \mathbb{R}^4$. Derive a formula for the 4x4 projection matrix, $\mathbf{A}$, that maps a homogeneous object point, $\boldsymbol{p} = (x, y, z, 1)^T$, to its shadow point, $\tilde{\boldsymbol{p}} = \mathbf{A}\boldsymbol{p}$. *(Hint: Consider the ray $\mathbf{p} + t\boldsymbol{u}$.)*



Problem 3: Splines (30 pts)

Consider the cubic spline segment, $\mathbf{p}(t) = \mathbf{a}\,t^3 + \mathbf{b}\,t^2 + \mathbf{c}\,t + \mathbf{d}$, on the unit interval $t \in [0, 1]$. Recall that Hermite splines use $\mathbf{p}$ and $\mathbf{p}'$ controls at each end point, i.e., $\mathbf{p}_0 = \mathbf{p}(0)$, $\mathbf{p}_0' = \mathbf{p}'(0)$, $\mathbf{p}_1 = \mathbf{p}(1)$, $\mathbf{p}_1' = \mathbf{p}'(1)$. In this question you will derive a spline which instead uses the following controls: at $t = 0$, use position ($\mathbf{p}_0$), velocity ($\mathbf{p}_0'$) and acceleration ($\mathbf{p}_0''$); at $t = 1$, use position ($\mathbf{p}_1$).

(a) What is this spline's 4x4 geometry matrix $\mathbf{G}$ such that $\mathbf{p}(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \mathbf{G} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_0' \\ \mathbf{p}_0'' \\ \mathbf{p}_1 \end{bmatrix}$?

(b) Given a cubic spline contructed from these segments, is the resulting curve (i) C^0 continuous? (ii) C^1 continuous? (iii) G^0 continuous? (iv) G^1 continuous? Briefly explain your reasoning in each case.

Problem 4: Rotations (15 pts)

In this question, you will use quaternion multiplication to show that two rotations don't necessarily commute, i.e., the order in which they are performed matters. In particular, let the quaternions $\boldsymbol{q}_x$ and $\boldsymbol{q}_y$ represent rotations about the x and y axes, by angles θ_x and θ_y, respectively. Use quaternion multiplication to show that $\boldsymbol{q}_x\boldsymbol{q}_y \neq \boldsymbol{q}_y\boldsymbol{q}_x$.

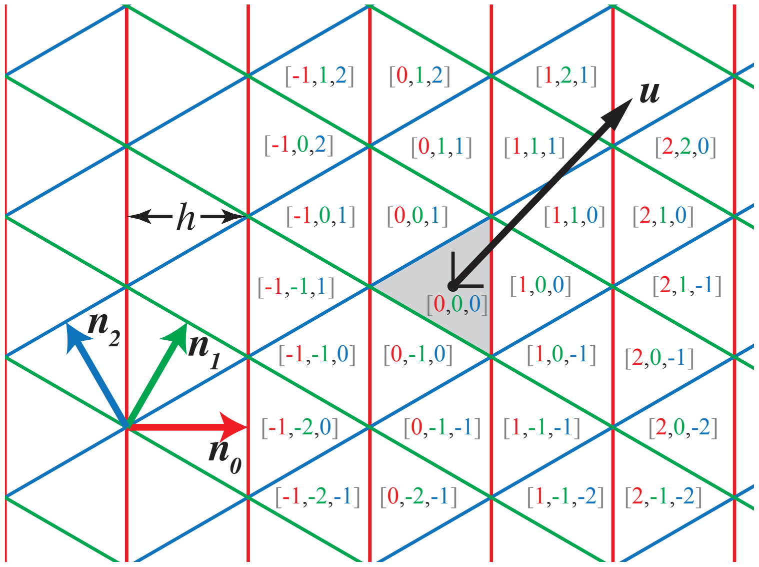
Problem 5: Meshes (10 pts)

Recall the half-edge data structure. What does the following pseudocode accumulate in the set `S`? *(Hint: Drawing a picture may help.)*

```
ProcessFace(f, S) {
  h = f.h;
  do {
    h = h.next;
    S.add(h.pair.next.v);
  } while (h != f.h);
}
```

Problem 6: Ray Tracing (20 pts)

Uniform subdivisions are commonly used for ray tracing, wherein a ray is intersected with cell boundaries to "walk" through the subdivision's cells. In this problem, you will generate pseudocode to walk through a uniform triangular subdivision (see figure). It can be viewed as the union of three 1D uniform subdivisions (color-coded for convenience), each associated with a normalized direction vector ($\boldsymbol{n}_0$, $\boldsymbol{n}_1$ or $\boldsymbol{n}_2$). Each 1D cell is of width, h. Denote each triangular cell by a 3-index label $[i_0, i_1, i_2]$ (RGB color coded) based on the cell's location in each of the 1D subdivisions. Assume that our ray $\boldsymbol{r}(t)$ originates from the origin, $\mathbf{0}$, so that $\boldsymbol{r}(t) = \mathbf{0} + t\boldsymbol{u}$, $t > 0$. Assume that the origin lies at the *centroid* of the triangular cell indexed by $[0, 0, 0]$.



Write pseudocode to efficiently generate the sequence of traversed cells when given the ray direction, $\boldsymbol{u}$ and these particular cell direction vectors (assume normalized). For example, given the $\boldsymbol{u}$ vector shown in the figure, your code would output the infinite sequence:

$$[0, 0, 0], [1, 0, 0], [1, 1, 0], [1, 1, 1], [1, 2, 1], \ldots$$

Output a traversed cell with the function "`output` $[i_0, i_1, i_2]$."

(Hint: Consider each direction's next t intersection value.)