Full Name: _____

Andrew Id: _____

15-462/662, Fall 2016

Midterm Exam

Oct XX, 2016

Instructions:

- This exam is CLOSED BOOK, CLOSED NOTES (with the exception of your one post-it note).
- You MUST answer Problem 1, and can choose any 5 out of the remaining 6 questions. Please circle the questions you choose on the front page.
- The exam has a maximum score of **100** points. Answering the extra (non-circled) question can earn you up to 5 points of extra credit.
- If your work gets messy, please clearly indicate your final answer.

Problem	Your Score	Possible Points
1		20
2		16
3		16
4		16
5		16
6		16
7		16
Total		100

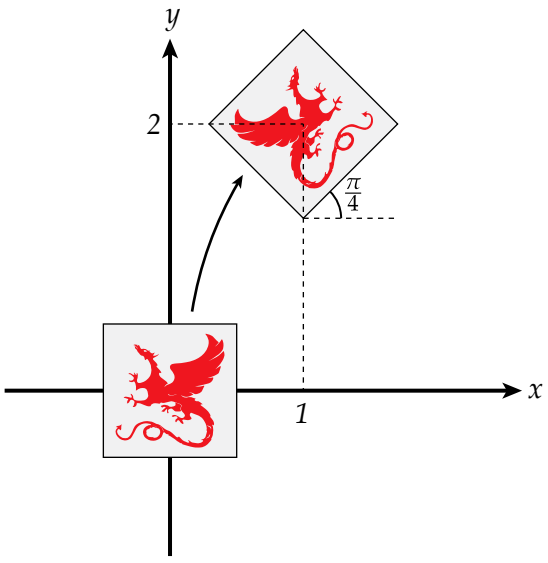
Getting the Party Started: Miscellaneous Short Problems

Problem 1. (20 points):

A. (4 pts) You graduate from CMU and start making serious bank at your graphics startup that dominates the niche, but surprisingly lucrative, mesh simplification market. To reward yourself you buy a limo and take it to the shop to receive a window tint. Tint is applied by fixing 10% opaque ($\alpha = 0.1$) layers of tint to your windows (more layers results in more light attenuation). If you want to have your windows reduce the amount of light entering the limo *by at least 20%*, what is the minimum number of layers of tint required?

Solution: *Need 3 layers (not 2), since first layer transmits 90% and next layer transmits 90% of that (81% of total, 19% attenuated). Need a third layer to get above the 20% tint requirement (27.1% is what you get).*

B. (4 pts) Describe in words any sequence of operations whose composition describes the overall transformation indicated by the figure below. (You do **not** need to write the matrices!)



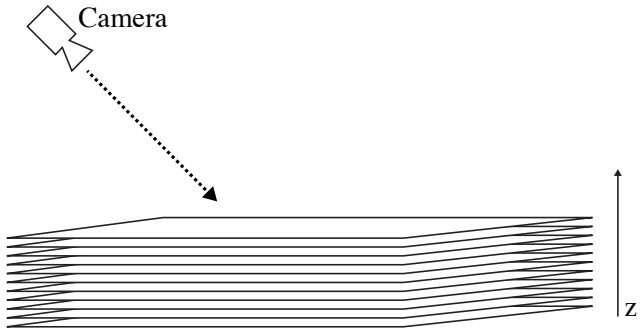
Solution:

- *Reflect across the vertical axis.*
- *Rotate by 45 degrees counter-clockwise.*
- *Translate by (1,2).*

C. (4 pts) In class we discussed the "wagon wheel" effect, where rapidly spinning tires on a moving car appear to spin backwards when viewed on TV or in a 30 fps video. Prof. Kayvon (correctly) claims that you've never witnessed this effect with your own eyes when looking at a moving car in broad daylight. (If you've seen the wagon wheel effect in the real-world, it would only have been outside at night under street lights, or indoors under cheap fluorescent bulbs). What is the cause of the wagon wheel effect, and why can it not be observed by the naked eye in daylight? (Optional: for kicks, why *might* it be observed by the naked eye if the scene is illuminated at night?)

Solution: *The image in the video is being sampled at a very low rate (24-30Hz) thus the wagonwheel effect is due to aliasing. In daylight our eyes integrate light continuously so there's no single moment of exposure. At light you might have seen the effect because of light sources oscillating due to their electric source.*

D. (4 pts) You've just landed a job implementing the renderer for the latest *World Series of Poker* video game, and you're trying to determine whether to use a rasterizer or a ray tracer. Consider the scene illustrated below with a camera looking down at a 45 degree angle on a deck of 5000 cards stacked in the +Z direction.



Consider the difference between rendering this scene using a rasterizer and a ray tracer: specifically think about the cost of determining what card is visible at each screen sample (or along each camera ray in the ray tracer). Do you think determining occlusion using the Z-buffer algorithm or via ray tracing is a better solution in this scenario? (Your answer can assume that the ray tracer has prebuilt a BVH or uniform grid to organize scene geometry.) Why?

E. (4 pts) You place an object **that is a perfect mirror** reflector in a scene and try to render the object using the starter code for Assignment 3. For those that haven't started (grrrr....) the starter code evaluates the reflectance equation as follows:

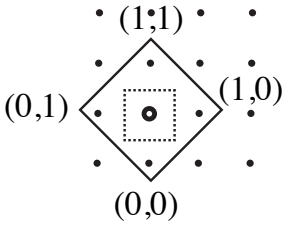
```
// to compute reflectance in the direction Wo
Spectrum Lo = 0;
for all reflection estimate samples:
    wi = generate random direction about surface normal
    Li = determine incoming light from direction wi
    Lo += brdf(w0, wi) * Li * cos(theta) / pdf;
return TWO_PI * Lo / num_samples;
```

Assume that the number of illuminance samples is four and the scene is lit with a uniform hemispherical light source so that incident radiance from any direction `wi` is the constant L. You render a picture and the reflective object is black in nearly all pixels. Why?

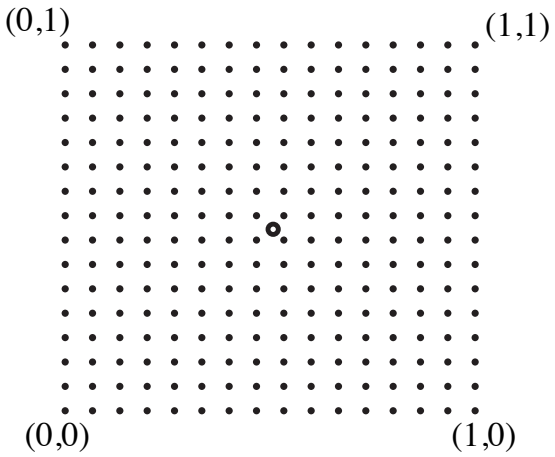
Solution: *The problem here is that by randomly sampling the hemisphere, you're almost always going to pick a direction where the BRDF is 0. You should have sampled incident light in the direction where the BRDF is non-zero (only one such direction), which would be the direction of perfect reflection about the surface's normal.*

Problem 2. (16 points):

Consider rendering the quadrilateral shown at left in the figure below. Per-vertex texture coordinates are given, and the dots indicate the position of screen sample points. Now consider the computation to compute the color of the quadrilateral at the highlighted screen sample point, which requires a texture lookup into the 16×16 texture shown at right. The location in texture space of the desired sample from the texture function is also shown in the figure.



Image



16 x 16 texture (color samples stored at dots)

A. (5 pts) Imagine you implement the texture lookup using a bilinear resampling ("bilinear filtering") of the texture map. Assuming the texture image contains details such as many sharp edges, what might be the problem with this approach?

Solution: *Aliasing! Samples are very spread out in texture space since this is a case of severe texture minification.*

B. (6 pts) On the figure on the previous page, draw the bounds of the region in texture space corresponding to the dotted rectangle surrounding the shaded screen sample point (at left). If you wanted to properly filter (avoid aliasing) the texture sampling result, briefly describe an accurate, but slow algorithm for computing the resampled value of the texture at this specified point?

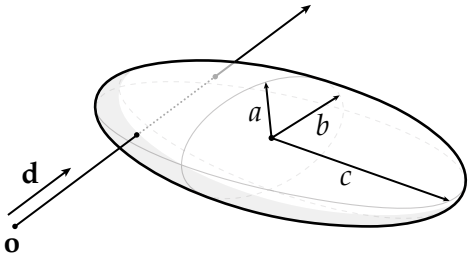
Solution: *It's a diamond. Just average all the texels in the diamond.*

C. (5 pts) Consider using a mip-mip to accelerate the algorithm you proposed in the previous subproblem. What is one quality problem that might arise from using this more efficient solution?

Solution: *Since the texture support region is a diamond, no axis-aligned square will fit the region well. It's going to overblur. Interesting the amount of blurring is a function of the rotation of the object. Ick.*

Rendering an Egg

Problem 3. (16 points):



Consider an ellipsoid with radii $a, b, c > 0$, which can be expressed explicitly as

$$(\theta, \varphi) \mapsto (a\cos(\theta)\cos(\varphi), b\cos(\theta)\sin(\varphi), c\sin(\theta))$$

for $-\frac{\pi}{2} \leq \theta \leq \frac{\pi}{2}$ and $-\pi \leq \varphi \leq \pi$. It can also be expressed implicitly as

$$\mathbf{x}^\mathsf{T} D \mathbf{x} = 1,$$

where $\mathbf{x} = (x, y, z)$ and

$$D = \begin{bmatrix} 1/a^2 & 0 & 0 \\ 0 & 1/b^2 & 0 \\ 0 & 0 & 1/c^2 \end{bmatrix}.$$

A. (4 pts) Suppose you wanted to visualize the ellipsoid either by tracing rays or by rasterizing points. Which representation would you use in each case? Why?

Solution: *For ray tracing I would use the implicit representation, because it lets me easily test whether a given point is contained in the ellipsoid (in particular, points along the ray). For rasterization I would use the explicit representation because I could easily plot points on the ellipsoid.*

B. (7 pts) Now consider a ray $\mathbf{r}(t) := \mathbf{o} + t\mathbf{d}$. At what times t will this ray intersect the ellipsoid? Give an explicit formula in terms of $\mathbf{o}$, $\mathbf{d}$, and D. When is this formula valid? *(Hint: quadratic formula!)*

Solution: *If we plug the ray equation into the implicit equation for the ellipsoid, we get*

$$0 = \mathbf{r}(t)^\mathsf{T} D \mathbf{r}(t) - 1 = (\mathbf{o} + t\mathbf{d})^\mathsf{T} D(\mathbf{o} + t\mathbf{d}^\mathsf{T}) - 1 = \underbrace{\mathbf{d}^\mathsf{T} D \mathbf{d}}_{=:u} t^2 + \underbrace{2\mathbf{o}^\mathsf{T} D \mathbf{d}}_{=:v} t + \underbrace{\mathbf{o}^\mathsf{T} D \mathbf{o} - 1}_{=:w}.$$

This equation is a quadratic polynomial in t, which we can solve using the quadratic formula:

$$t = \frac{-v \pm \sqrt{v^2 - 4uw}}{2u} = \frac{-\mathbf{o}^\mathsf{T} D \mathbf{d} \pm \sqrt{(\mathbf{o}^\mathsf{T} D \mathbf{d})^2 - (\mathbf{d}^\mathsf{T} D \mathbf{d})(\mathbf{o}^\mathsf{T} D \mathbf{o} - 1)}}{\mathbf{d}^\mathsf{T} D \mathbf{d}}.$$

When the discriminant is strictly positive, the ray pierces the ellipsoid at two distinct points. When it is zero, the ray merely grazes the ellipsoid tangentially. When it is negative, there is no intersection.

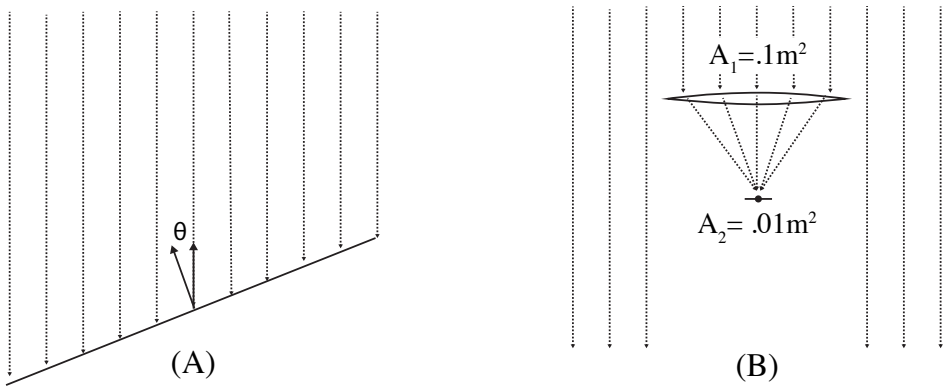
C. (5 pts) Suppose now that the ellipsoid is rotated via a matrix U. What's a **simple** way you can find the intersection with the rotated ellipsoid, *without* redoing all of your work from part (B)?

Solution: *We can simply apply the inverse rotation U^{-1} (or U^T) to the ray origin* $\mathbf{o}$ *and direction* $\mathbf{d}$ *before computing the intersection.*

Magnifying Glass

Problem 4. (16 points):

Consider unidirectional sunlight falling on a tilted plane as shown in part (A) at left in the figure below (due to the large distance to the sun, all incident light is from the same direction). The plane is oriented with angle θ relative to the incoming beam, and a measurement of **irradiance** on the plane yields the value 500 Watts/meter^2.



A. (8 pts) If you tilt the plane so that it is normal to the incoming beam, give an expression for the new irradiance on the surface (in terms of θ)? Does surface irradiance increase or decrease?

Solution: We have $dA_{\text{tilted}} \cos\theta = dA$, so $dI/dA = dI/(dA_t \cos\theta) = (dI/dA_t)/\cos\theta$. *The irradiance on the tilted plane is equal to* $500/\cos\theta$

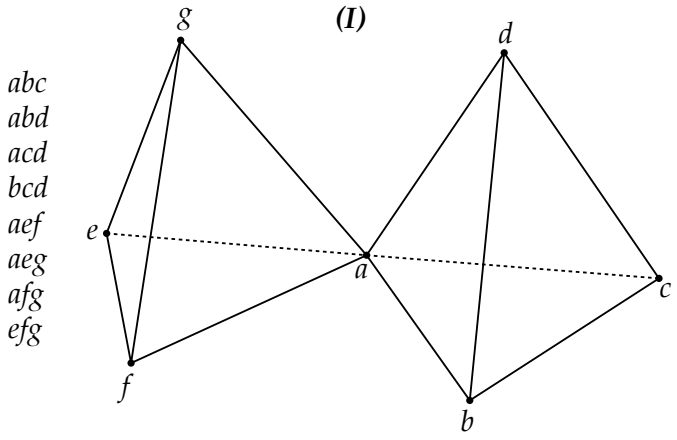
B. (8 pts) Now consider the setup in part B of the figure where a magnifying glass with area $A_1 = 0.1\text{m}^2$ has been placed about a small surface of area $A_2 = 0.01\text{m}^2$. Assume the surface of the magnifying glass and the surface of the receiving surface are flat planes that oriented directly toward the beam, and assume that the receiving surface is sufficiently small that it receives energy uniformly about its surface. Give an expression for the irradiance at points on the small receiving surface. (Hint: don't overthink this one, just consider the definition of irradiance! No integrals are required in this computation.)

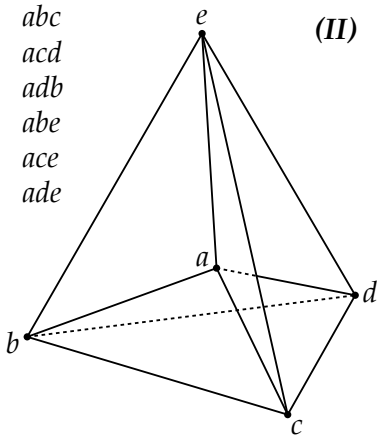
Solution: *irradiance is energy per unit area. But due to the magnifying glass all the energy hitting the magnifying glass is not focused on the little area that's ten times smaller. Therefore the energy density is 10x larger, so it's 10 times the answer in part A.*

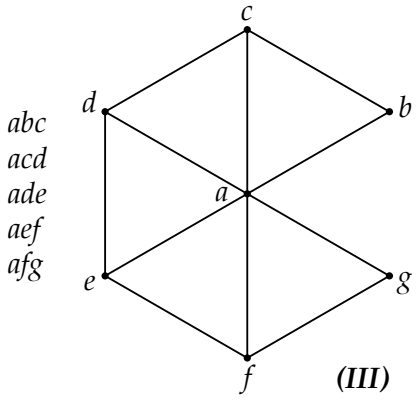
Meshes and Manifolds

Problem 5. (16 points):

A. (5 pts) A triangle mesh has *boundary* if at least one of its edges is contained in only one triangle. A triangle mesh is a *manifold* if (i) every edge is contained in one or two triangles, and (ii) every vertex is contained in a single loop or fan of triangles. Using these definitions, indicate whether each of the examples below are manifold and/or have boundary. If a mesh is nonmanifold, specify one of its nonmanifold vertices or edges. If a mesh has boundary, specify one of its boundary edges.







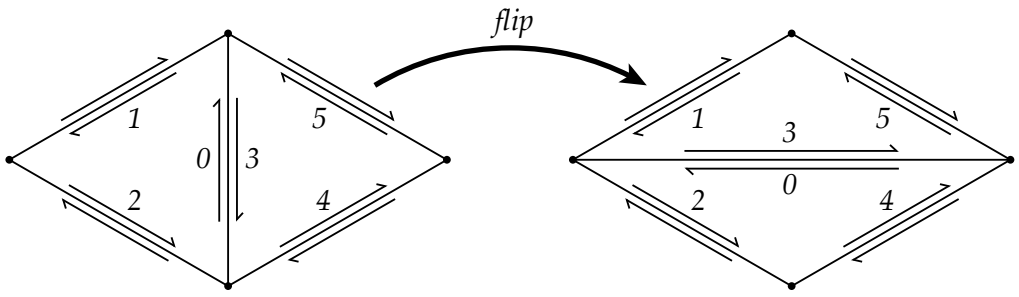
Solution:

(a) *Nonmanifold, no boundary. Vertex a is nonmanifold.*

(b) *Nonmanifold, with boundary. Vertex a is nonmanifold, edge bc is on the boundary.*

(c) *Manifold, with boundary. Edge ab is on the boundary.*

B. (5 pts) Consider a simplified halfedge data structure that does not explicitly encode vertices, edges, or faces—each halfedge keeps track of only its "next" and "twin" halfedges. This data is stored in two fixed-length arrays `N` and `T` of size $2E$, where E is the number of edges in the mesh. In particular, suppose we give each halfedge a unique index in the range 0 to $2E - 1$. Then for any halfedge `k`, `N[k]` is the index of its next halfedge, and `T[k]` is the index of its twin. Update these arrays to reflect the edge flip below, using as few assignments as possible.



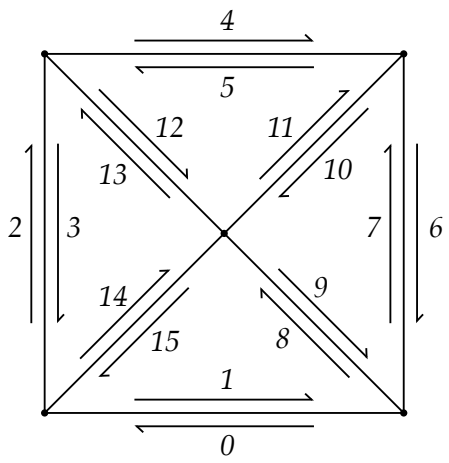
Solution: *The twins all remain the same, so we need only make these assignments:*

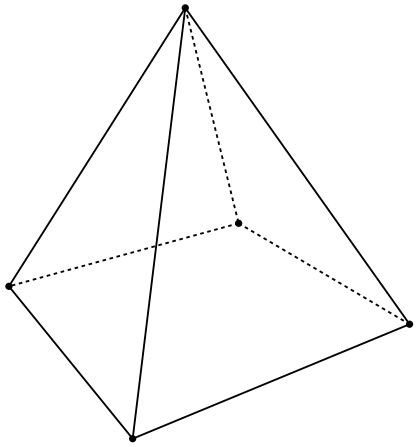
```
N[0] = 2;
N[1] = 3;
N[2] = 4;
N[3] = 5;
N[4] = 0;
N[5] = 1;
```

C. (6 pts) Consider the same simplified halfedge data structure as in the previous problem. We can make this data structure even simpler by adopting the convention that halfedges $2p$ and $2p + 1$ are always twins, for $p = 1, \ldots, E - 1$. This way, we don't even have to store the array `T`—just the array `N` of "next" pointers. In this way, any permutation of the integers $0, \ldots, 2E - 1$ specifies a valid manifold polygon mesh. Verify this claim by drawing a picture of the mesh defined by the `N` array below. *(Hint: draw the individual faces first, then think about how they connect up.*)

k	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
N[k]	2	8	4	14	6	12	0	10	15	7	9	5	11	3	13	1

Solution:





BVH Refit

Problem 6. (16 points):

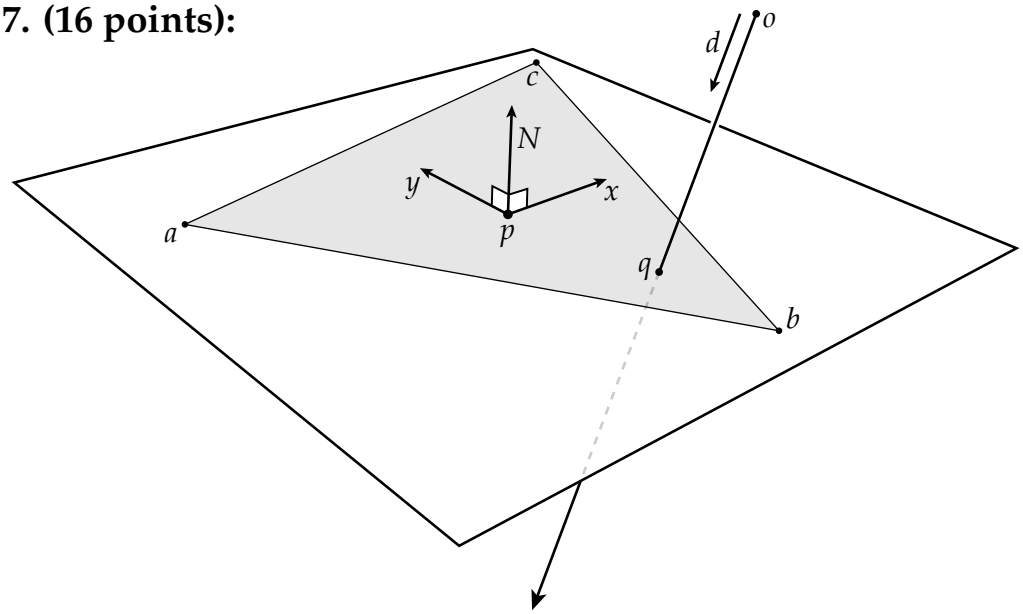
One of the nice properties of a BVH is that it can be efficiently updated ("refit") when a primitive in the bounding volume hierarchy is moved. This refit modifies the BVH so that the BVH property is maintained: the bbox of a node is the bound of the primitives contains in all child nodes. The refit **does not modify the structure of the BVH – only the bounding boxes of the nodes change.**

In the pseudocode below you're given a definition of a `BVHNode` and interfaces for getting the bounding box of a primitive. Please implement the function `refitBVH(node)` below which should update bounding boxes in the BVH tree so that the BVH property holds **assuming that only scene primitives in the node `node` has moved**. To keep things simple, we've given you a pointer to the parent node `parent` in the BVH. You can assume that the `left` and `right` pointers of an interior are always non-NULL. A full-credit answer yields a BVH with the tightest bounds possible for the current tree.

```
struct BBox {
   void clear();               // resets bbox to empty
   void union(const BBox& b);  // enlarges bbox to include volume in b
};

struct Primitive {
  BBox getBBox() const;    // returns primitive's world-space axis-aligned bbox
};

struct BVHNode {
   BVHNode* left, *right;
   BVHNode* parent;   // parent node in the BVH: NULL if root

   BBox bbox;         // nodes bbox, you need to update this!

   int numPrims;      // 0 if node is interior node, non-zero otherwise
   Primitive* prims;  // prims[i]->getBBox() returns bbox of i'th prim in node
};

void refitBVH(BVHNode* node) {
  node->bbox.clear();
  if (numChildren > 0)
     for (int i=0; i<num_children; i++)
       node->bbox.union(prims[i]->getBBox());
  else {
    node->bbox.clear();
    node->bbox.union(node->left->bbox);
    node->bbox.union(node->right->bbox);
  }
  if (node->parent)
    refitBVH(node->parent);
}
```

Intersection

Problem 7. (16 points):



A. (4 pts) Suppose you have a plane passing through the point $\mathbf{p}$ with unit normal $\mathbf{N}$ and two orthogonal unit tangent vectors $\mathbf{x}$ and $\mathbf{y}$. Write an equation for the plane in both implicit and explicit form.
Solution: *The implicit form is* $\mathbf{N}^\mathsf{T}(\mathbf{x} - \mathbf{p}) = 0$. *The explicit form is* $(u, v) \mapsto \mathbf{p} + u\mathbf{x} + v\mathbf{y}$.

B. (4 pts) Suppose you use the implicit plane equation to intersect a ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ with the plane containing the triangle $(\mathbf{a}, \mathbf{b}, \mathbf{c})$. If $\mathbf{q}$ is the point of intersection, how would you test whether it is inside the triangle? Try to be as detailed as possible.

Solution: *There are several closely related solutions. For instance, you could compute the barycentric coordinates as the ratio of signed sub-triangle areas over the total triangle area. If they are positive and sum to one, then the point is inside the triangle. Explicitly, the three sub-triangle areas are* $A_1 := N \cdot ((b-a) \times (q-a))$, $A_2 := N \cdot ((c-a) \times (q-b))$, *and* $A_3 := N \cdot ((a-c) \times (q-c))$, *and the total triangle area is* $N \cdot ((b-a) \times (c-a))$. *These same barycentric coordinates can be computed by picking two of the triangle edges as a basis; alternatively, one can check to see that the point q is contained in the three half-spaces determined by the three edges (which entails calculations not very different from computing barycentric coordinates).*

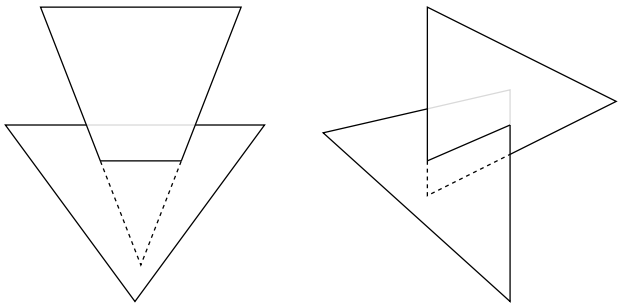
C. (4 pts) Now suppose you use the *explicit* plane equation to compute the ray-plane intersection. What equation(s) do we have to solve, for which variables? What are some pros and cons of computing the intersection this way?

Solution: *We have to solve the equation*

$$\mathbf{p} + u\mathbf{a} + v\mathbf{b} = \mathbf{o} + t\mathbf{d}$$

for the parameter values t, u, and v; this can be viewed as a system of three simultaneous linear conditions on three variables. The benefit of computing the intersection this way is that we immediately get the barycentric coordinates u and v, in addition to the ray parameter t.

D. (4 pts) Briefly describe how you might use your ray-triangle intersection routine to perform a triangle-triangle intersection test. In this setting, what is the significance of the value of t? To keep the answer short, you can assume that triangles are not parallel.



Solution: *When the triangles are not parallel, one can simply check whether any of the edges of one triangle intersect the other triangle (six tests total). To perform an edge-triangle intersection test, one can shoot a ray from one endpoint of the edge to another, checking whether the intersecting t value is between 0 and the length of the edge.*