MIT EECS 6.837 Computer Graphics, F09

Final Exam, December 17, 1:30pm-3:30pm

Name: **Total: [/ 42]**

1 Rendering Basics [/ 8]

1.1 Ray Casting vs. Rasterization [/ 4]

Give pseudocode for rendering an image using a ray caster and a rasterizer.

Ray Casting

```
for each pixel
  generate ray through pixel
  for each object
     test if ray intersects object
     (keep closest intersection)
(+2)
```

Rasterization

```
for each primitive
  (project primitive onto screen)
  for each pixel
    test if pixel inside primitive
    (keep closest intersection)
(+2)
```

1.2 Visibility [/ 2]

How is correct visibility ordering between primitives achieved in ray casting? What is the idea in z-buffering?

```
ray casting: as we test objects along ray, we only update intersection
             if it's closer than the current closest intersection (+1)
z-buffer: each pixel keeps a "closest depth" value, pixel is only written to
if current primitive is closer. (+1)
```

1.3 Working Set [/ 2]

What are the main differences between the working sets in rasterization and ray casting? I.e., what needs to be kept in memory during execution in each case?

```
ray caster must have entire scene in memory (+1)
  (can render image in tiles)
rasterizer must have entire image (and z-buffer) in memory (+1)
  can stream over primitives
```

2 Ray Casting/Tracing [/ 12]

2.1 Implicit/explicit surface representations [/ 1]

What is the difference between implit and explicit surface representations?

```
Implicit representation only allows testing if a given point is on the surface.
Explicit representation lets you generate points on the surface.
```

2.2 Ray Representation [/ 1]

What is the explicit representation of a ray? Give a formula.

```
P(t) = O + tD,
where O is the ray origin, D is ray direction, and t>=0 is a scalar.
(no points off if no definition of terms)
```

2.3 Ray-Plane Intersection [/ 4]

An infinite plane may be represented by the formula $\mathbf{p} \cdot \mathbf{n} + d = 0$ where $\mathbf{n}$ is the plane normal, d is a real constant, and $\mathbf{p} = (x, y, z)$ is a (variable) 3D point.
a) Is this an implicit or an explicit representation? [/ **1**]

```
Implicit (only allows testing, not generation)
```

b) Derive the formula for intersecting a ray and a plane. You can assume that the ray direction is not tangent to the plane. [/ **3**]

```
    P(t) is on plane
<=> P(t).n + d = 0          (+1)
<=> (O + tD).n + d = 0      (+1)
<=> t(D.n) = -O.n - d
<=> t = -(O.n + d)/D.n      (+1)
```

2.4 Barycentric triangle representation [/ 2]

Give the barycentric representation of a triangle with vertices $\{\mathbf{a}, \mathbf{b}, \mathbf{c}\}$ in terms of $\{\alpha, \beta, \gamma\}$, including possible equality and inequality constraints.

```
P(alpha,beta,gamma) = alpha*a + beta*b + gamma*c,       (+1)
with alpha+beta+gamma = 1 and alpha,beta,gamma >= 0.    (+1)
OR P(beta,gamma) = a + beta*(b-a) + gamma*(c-a),
with beta+gamma <= 1 and beta,gamma >= 0.
```

2.5 BVH Traversal [/ 4]

A bounding volume hierarchy (BVH) based on bounding spheres may be represented using a following structure (in pseudocode).

```
struct Node
{
    Sphere            boundingSphere;
    bool              isLeaf;
    Node*             children[2];
    list< Triangle* > primitives;  // contains stuff in case of a leaf
};
```

The following pseudocode traverses a BVH. Fill in the two missing pieces. A rough outline will suffice, see the child case for an example.

```
bool rayIntersects( Ray* ray, Hit* rayHit, Node* node )
{
    // a) test if the node intersects the ray at all and act accordingly (      / 1)
    if node->boundingSphere does not intersect ray
       return false

    // leaf node? test triangles, update hit if necessary, and return
    if node->isLeaf
        test all triangles in node->primitives
        when a triangle is hit
            compute t and update rayHit if t closer than rayHit
        return true if any triangle was hit, false otherwise
    endif

    // b) recurse into the children in the right order, paying attention
    // to handling the case of overlapping nodes correctly (       / 3 )

    compute t's for both child nodes' bounding spheres
    recurse into closer node first, based on t's                (+1)
    if there was a hit in the closer node                       (+1)
      check if hit point is inside the farther node too
      if yes
        recurse into farther node as well
    if no hit in the closer node                                (+1)
      recurse into farther node
    return true if anything was hit in either child node, false otherwise

}
```

3 Rasterization [/ 7]

3.1 Edge Functions [/ 2]

Edge functions $e_i(x, y) = ax+by+c$ are 2D line equations that are computed from the three edges ($i = 1, 2, 3$) of a projected triangle. What is the mathematical condition that holds when a pixel/sample at (x, y) is inside the triangle?

```
e_i(x,y) >= 0 for all i=1,2,3
```

3.2 Rasterization Using Edge Functions [/ 5]

Give pseudocode for rasterizing a triangle using edge functions, starting before projection. Your code should use screen bounding boxes for avoiding testing all pixels on the screen and include z-buffering for visibility. You can assume the triangle has a constant color and that clipping has been performed already.

```
project vertices onto screen
compute edge functions e_i from projected vertices
compute bounding box B from projected vertices        (+1 until here)
(set up interpolation matrix)
for all pixels (x,y) inside B                         (+1)
   evaluate e_i(x,y) for i=1,2,3                      (evaluate+test, +1)
   if all positive
      interpolate z from vertices
      if zbuffer[x,y] > z                             (z test, +1)
         framebuffer[x,y] = color                     (z+color update, +1)
         zbuffer[x,y] = z
      endif
   endif
endfor
-1 if no z update
```

4 Shading, Sampling and Textures [/ 15]

4.1 Irradiance

How does the irradiance incident on a surface vary with the angle between the surface normal **n** and incident light direction **l**? [/ **1**]

With the cosine (+1). No light from below, i.e., when cosine is negative.
(No points off for just cosine.)

4.2 The BRDF [/ 1]

The BRDF stands for "Bidirectional Reflectance Distribution Function". It is often denoted by $f_r(\mathbf{l},\mathbf{v})$, where **l** is incident (light) direction and **v** is the outgoing (viewing) direction. What does the value $f_r(\mathbf{l},\mathbf{v})$ tell you?

f_r(l,v) is the fraction of light that reflects from direction l to direction v.

4.3 Diffuse Reflectance [/ 1]

How does the BRDF of an ideally diffuse surface vary with **l** and **v**?

It is a constant, no variation. (+1)
(What constant? Albedo/pi.)

4.4 Types of Aliasing [/ 2]

What is meant by pre-aliasing and post-aliasing?

Pre-aliasing happens when sampling rate is not high enough. This leads to
the spectral replicas to overlap, and makes it impossible to reconstruct the
original signal from the samples (+1)
Post-aliasing means that we perform poor reconstruction based on sampled values (+1)
The end result is again that the reconstruction does not match the original.

4.5 Avoiding Pre-Aliasing [/ **4**]

a) Give the two main ways of preventing or alleviating the effects of pre-aliasing. [/ **2**]

1: Sample at a higher rate (+1), this pushes replicas further apart and we can
represent higher frequencies.
2: Prefilter the signal, i.e., blur before sampling to remove the high
frequencies that cannot be represented using the chosen sample rate. (+1)

b) Which one of the two does MIP-mapping approximate? What is the general idea? [/ **2**]

MIP-mapping approximates prefiltering (+1) by precomputing a set of different
low-pass filtered versions of the texture (+1). (The actual prefilter is
then approximated as a combination of the prefiltered results.)

4.6 Supersampling, Multisampling [/ **6**]

a) Imagine you are rendering an image with a single sample per pixel. First you sample the image. This creates replicas in the frequency domain. Reconstructing the samples by a low-pass filter that corresponds to the sampling frequency recreates a continuous image. Any original image frequencies above the pixel pitch get aliased in the reconstruction. Describe supersampling in similar terms. Which frequencies are aliased in the final output? How does this relate to the two methods for avoiding pre-aliasing? [/ **4**]

In supersampling, we first sample the signal at a higher rate than the output. (+1)
This pushes frequency replicas further apart.
We then low-pass filter the supersampled signal by a low-pass filter that
corresponds to _the final output sampling rate_ and sample the low-passed result
at the sample locations that correspond to the output sampling rate (+1).
Aliasing only happens for frequencies above the supersampling rate (+1)
This approximates a prefilter using a higher sampling rate -- we remove
frequencies above the output rate (as we should in prefiltering), but only up
to the limit given by the supersampling rate (+1)

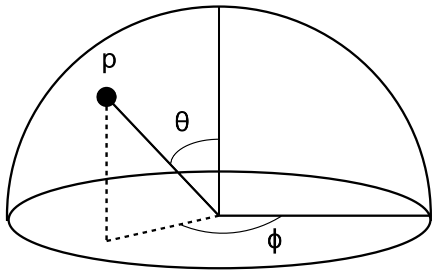
b) How does multisampling differ from supersampling? Why is it useful? [/ **2**]

In multisampling, we compute shading results (colors) only once per pixel
but supersample visibility (+1) (and share shading results for subpixel samples
that fall within the same primitive).
It is useful because shading is expensive compared to visibility.

5 Extra Credit

No partial credit for extra credit questions.

5.1 Cosine Importance Sampling [/ 6]



Given two uniformly distributed random numbers $x_1 \in [0, 1]$ and $x_2 \in [0, 1]$, give the polar coordinates for a point $p = (\theta, \phi)$ on the hemisphere as functions of x_1, x_2, such that the distribution of ps on the hemisphere is proportional to $\cos\theta$. (This is helpful, for instance, when rendering indirect diffuse illumination.)

```
Answer: ( acos(sqrt(x_1)), 2*pi*x_2 )
(Of course, x_1 <=> x_2.)
```

You can use this page for sketching.

You can use this page for sketching.

You can use this page for sketching.