Reflection Rendering

1. Preprocess: render billboard (i.e. texture mapped rectangle), save it to file
2. Run time:
   1. For all triangles t of reflector
      1. For all pixels p covered by projection of t
         1. n = normal at p // fragment shader input
         2. xyz = 3D point at p // fragment shader input
         3. ev = eye vector at p // eye – xyz; BLUE highlights fragment shader code
         4. rv = reflected ray at p
         5. for all billboards b
            1. ip = b intersected with rv

intersect rv with plane of b // set pixels to red

check if intersection is inside rectangle // set to green

check if intersection occurs at occupied (i.e. full, opaque) texel of b // set to blue

* + - * 1. if ip and ip is closer along rv

closest intersection = ip

* + - 1. If no intersection with billboards
         1. Lookup rv in environment map // set to yellow

Soft Shadow Rendering

Input: Light rectangle L, light rectangle sampling rate nxn (16x16), boxes B, ground plane G, out. view V

Output: Scene (B + G) rendered from V with soft shadows cast from L

Algorithm:

For all triangles t in B and G (i.e. in the scene)

For all pixels p inside projection of t

p.shadow = 0; // blue code is what the fragment shader does

For i = 1 to n // two for loops that sample the light rectangle

For j = 1 to n

Lr = light ray defined by Lij and p.xyz // Lij is current light sample;

// Lr is curr. Light ray

intFound = false

For all faces q in B

If Lr intersects q

p.shadow++

break

p.outColor = p.inColor\*ka + (1-ka)\*(1-p.shadow/(n\*n))