CHAPTER 20

MULTIPLE IMPORTANCE SAMPLING 101

Anders Lindqvist
NVIDIA

ABSTRACT

We present the basics of *multiple importance sampling* (MIS), a well-known algorithm that can lower the amount of noise in a path tracer significantly by combining multiple sampling strategies. We introduce MIS in the context of direct light sampling and show how it works in a path tracer. We assume that the reader has already written a simple path tracer and wants to improve it.

20.1 DIRECT LIGHT ESTIMATION

We will be focusing on the problem of direct lighting in a path tracer. The content is heavily inspired by Veach's doctoral thesis [4]. Our problem statement is that for a given surface point $\mathbf{P}$ with a normal $\mathbf{n}$ and a material, we want to estimate the direct light being reflected in a direction $\omega_\mathbf{o}$. We will use code to illustrate all concepts, but the direct light integral we are estimating is

$$L_r(\mathbf{P}, \omega_\mathbf{o}) = \int_\Omega \underbrace{L_e(r(\mathbf{P}, \omega_\mathbf{i}), -\omega_\mathbf{i})}_{\text{light}} \underbrace{f(\mathbf{P}, \omega_\mathbf{o}, \omega_\mathbf{i})(\mathbf{n} \cdot \omega_\mathbf{i})}_{\text{BRDF and cosine}} d\omega_\mathbf{i}, \tag{20.1}$$

where r is a ray tracing function that shoots a ray into the scene and picks the first visible surface (if any), L_e is the emission color at a surface point, and f is the *bidirectional reflectance distribution* (BRDF) that describes how light is reflected at the surface.[1] The integral is over all directions $\omega_\mathbf{i}$ from which lights come over the hemisphere Ω around the normal. The word *estimate* here is used in a Monte Carlo sense in that we want to be able to average several such estimates to get a better estimate.

A first path tracer will most likely employ one of two techniques to handle direct lighting: material sampling or light sampling. We will describe them

[1] Here, we assume that if r does not hit a surface, it causes L_e to become black.

A. Marrs, P. Shirley, I. Wald (eds.), *Ray Tracing Gems II*, https://doi.org/10.1007/978-1-4842-7185-8_20

Listing 20-1. *Direct light using cosine hemisphere sampling.*

```
vec3 direct_cos(vec3 P, vec3 n, vec3 wo, Material m) {
  vec3 wi = random_cosine_hemisphere(n);
  float pdf = dot(wi, n)/PI;
  Intersect i = intersect(P, wi);
  if (!i.hit) return vec3(0.0);
  vec3 brdf = evaluate_material(m, n, wo, wi);
  vec3 Le = evaluate_emissive(i, wi);
  return brdf*dot(wi, n)*Le/pdf;
}
```

both and discuss their pros and cons. Though we only talk about emissive surfaces in our examples, the same methods can be used with a sky dome or skylight as well.

20.1.1 COSINE HEMISPHERE SAMPLING

We start with an even more basic sampling variant. In cosine hemisphere sampling we use no knowledge of the material or the light sources when choosing samples. Instead, we only sample according to the $\mathbf{n} \cdot \omega_\mathbf{i}$ term in Equation 20.1. Choosing directions in a way that is not uniform over the hemisphere is called *importance sampling* and is directly supported by the Monte Carlo integration framework.

The code in Listing 20-1 shoots one ray into the scene and gives out an estimate. Because this is the first Monte Carlo integrator, we show a full evaluation using `N` samples for a given surface point `P`, normal `n`, and material `mat`.

```
vec3 estimate = vec3(0.0);
for (int i=0; i<N; i++) {
  estimate += direct_cos(P, n, mat);
}
estimate /= N;
```

When we go to full path tracing, we will use a different surface point for each sample in order to get antialiasing, but the principles are the same when estimating the integral. Note that the two dot products in Listing 20-1 cancel out and, as we only generate directions in the hemisphere around the normal, there is no need to clamp the dot products.

20.1.2 MATERIAL SAMPLING

In material sampling we select directions with a probability that is proportional to $f(\mathbf{P}, \omega_\mathbf{o}, \omega_\mathbf{i})(\mathbf{n} \cdot \omega_\mathbf{i})$, as shown in Listing 20-2. For a fully diffuse

Listing 20-2. *Direct light using material sampling.*

```
vec3 direct_mat(vec3 P, vec3 n, vec3 wo, Material m) {
  vec3 wi;
  float pdf;
  if (!sample_material(m, n, wo, &wi, &pdf)) {
    return vec3(0.0);
  }
  Intersect i = intersect(P, wi);
  if (!i.hit) return vec3(0.0);
  vec3 brdf = evaluate_material(m, n, wo, wi);
  vec3 Le = evaluate_emissive(i, wi);
  return brdf*dot(wi, n)*Le/pdf;
}
```

Listing 20-3. *Direct light using light sampling.*

```
float geom_fact_sa(vec3 P, vec3 P_surf, vec3 n_surf) {
  vec3 dir = normalize(P_surf - P);
  float dist2 = distance_squared(P, P_surf);
  return abs(-dot(n_surf, dir)) / dist2;
}

vec3 direct_light(vec3 P, vec3 n, vec3 wo, Material m) {
  float pdf;
  vec3 l_pos, l_nor, Le;
  if (!sample_lights(P, n, &l_pos, &l_nor, &Le, &pdf)) {
    return vec3(0.0);
  }
  float G = geom_fact_sa(P, l_pos, l_nor);
  vec3 wi = normalize(l_pos - P);
  vec3 brdf = evaluate_material(m, n, wo, wi);
  return brdf*G*clamped_dot(n, wi)*Le/pdf;
}
```

material, this is exactly what we do with the cosine hemisphere sampling technique. For a more glossy material, this could mean that we more often select directions close to the reflection direction.

20.1.3 LIGHT SAMPLING

In light sampling we pick positions on the light sources themselves. This is different from cosine hemisphere sampling or material sampling where we do not use any scene information (except the surface normal and surface material in material sampling).

In Listing 20-3 the sampling of light sources chooses points on surfaces in the scene. In order to be compatible with Equation 20.1, we must apply a

geometric factor G [4, Equation 8.3], which may look familiar to anyone who has implemented light sources in real-time graphics. It accounts for the fact that if we choose a point on a surface, we must take attenuation into account; points far away should be darker. If we look at sampling using directions—like we do for cosine hemisphere and material sampling—there is nothing in there. Instead, it is implicit in the fact that we hit things far away with a lower probability. In Listing 20-3 we see the factor `G` at the end. If there is something blocking the emissive surface (like another surface, emissive or not), `sample_light` will return false and we will not get any contribution. Note that unlike Veach we have chosen to not include `dot(n, wi)` in G. This is reflected in our function name `geom_fact_sa`.

A first implementation of `sample_lights` in a scene where all the light sources are spheres or quads could look something like Listing 20-4.

Listing 20-4. *Sample light sources.*

```
bool sample_lights(vec3 P, vec3 n, vec3 *P_out, vec3 *n_out,
  vec3 *Le_out, float *pdf_out)
{
  int chosen_light = floor(uniform()*NUM_LIGHTS);
  vec3 l_pos, l_nor;
  float p = 1.0 / NUM_LIGHTS;
  Object l = objects[chosen_light];

  if (l.type == GeometryType::Sphere) {
    float r = l.size.x;
    // Choose a normal on the side of the sphere visible to P.
    l_nor = random_hemisphere(P-l.pos);
    l_pos = l.pos + l_nor * r;
    float area_half_sphere = 2.0*PI*r*r;
    p /= area_half_sphere;
  } else if (l.type == GeometryType::Quad) {
    l_pos = l.pos + random_quad(l.normal, l.size);
    l_nor = l.normal;
    float area = l.size.x*l.size.y;
    p /= area;
  }

  bool vis = dot(P-l_pos, l_nor) > 0.0; // Light front side
  vis &= dot(P-l_pos, n) < 0.0; // Behind the surface at P
  // Shadow ray
  vis &= intersect_visibility(safe(P, n), safe(l_pos, l_nor));

  *P_out = l_pos;
  *n_out = l_nor;
  *pdf_out = p;
  *Le_out = vis ? l.material.emissive : vec3(0.0);
  return vis;
}
```

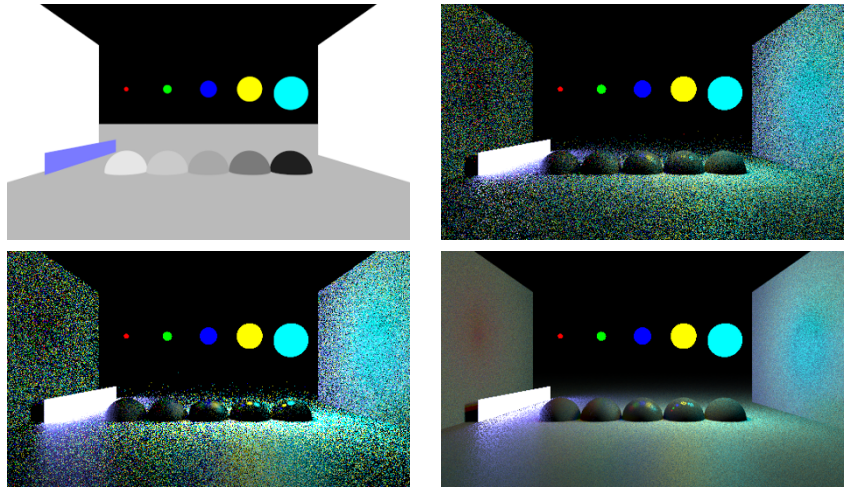

Figure 20-1. *Scene description (upper left), cosine hemisphere sampling (upper right), material sampling (lower left), and light sampling (lower right). Roughness is shown in the first image with black being low roughness and white being high roughness. The color of the emissive light sources are shown as well. We can see that material sampling best captures the sharp reflections of the emissive spheres in the rightmost half-sphere. Light sampling has far less noise but struggles near the emissive quad on the left. All images are rendered using the same amount of samples.*

The function `safe` offsets the position by the normal such that it is moved out of self-intersection.

20.1.4 CHOOSING A TECHNIQUE

Light sampling works very well for scenes with diffuse surfaces but less so for scenes with glossy materials. If the BRDF only includes incoming light in a small region on the hemisphere, it is unlikely that light sampling will pick directions in that region. Light sampling also struggles where emissive surfaces are very near the position from which we sample. This can be seen at the base of the emissive quad in Figure 20-1 where there is noise. Material sampling works best when we have highly reflective surfaces and large light sources that are easy to hit. In Figure 20-1 we can see that only material sampling finds the emissive surfaces on the reflective half-spheres with low roughness.

When we fail to pick the best strategy, we are punished with noise. This happens because we find the light sources with a direction that is relevant for the BRDF only with a very low probability. Once we get there, we *boost* the contribution a lot by dividing by the really low probability. We could try to pick

a technique based on the roughness value of the surface, but it is very error prone because the optimal choice also depends on the size and proximity of the emissive surfaces. Per-scene tweaks of thresholds are never fun, and here we might even have to tweak on an even finer level.

20.1.5 MULTIPLE IMPORTANCE SAMPLING

Let us start by considering what would happen if we would render the image once using each technique and blend them together evenly. What would happen for pixels where one technique was good but the other ones were really bad? Unfortunately, there is nothing from the noise-free images that "masks" the noise from the bad images. The noise would still be there, only slightly dimmed by blending in the good image. Like we discussed in the previous section, there is no optimal strategy even for a single surface point; the choice will be different on different parts of the hemisphere!

A better way was shown in Chapter 9 of Veach's thesis [4]. It shows how multiple importance sampling can be used to combine multiple techniques. The idea is that whenever we want to estimate the direct light, we use *all* techniques! We will assume that all of our techniques generate a surface point and give us a probability of choosing that surface point. With MIS we let each technique generate a surface point. Then we compare the probability of generating that surface point with the probability that the other techniques would generate that same point. If the other techniques can generate that surface point with a higher probability, we let them handle most of it, maybe all of it.

To determine the per-sample-per-technique weights, we need some sort of heuristic. Veach introduced a heuristic called the *balance heuristic* and proved that is close to optimal. See Listing 20-5.

Listing 20-5. *Balance heuristics for two techniques.*

```
1 float balance_heuristic(float pdf, float pdf_other) {
2   return pdf / (pdf + pdf_other);
3 }
```

By plugging in the probability from the material sampling technique and the light sampling technique into the balance heuristic, we can determine the weights to use. Almost. First, we need to adapt the material sampling to also generate surface points. Material sampling generates a direction. By shooting the ray into the scene, we can find the surface point and normal

representing that sampled direction. The probability of choosing that surface point then can be obtained by combining the probability of choosing the direction and the geometric factor *G* like we did for light sampling earlier, as shown in Listing 20-6.

Listing 20-6. *Direct light using MIS.*

```
vec3 direct_mis(vec3 P, vec3 n, vec3 wo, Material m) {
  vec3 result = vec3(0.0);
  vec3 Le, m_wi, l_pos, l_nor;
  float l_pdf, m_pdf;
  // Light sampling
  if (sample_lights(P, n, &l_pos, &l_nor, &Le, &l_pdf)) {
    vec3 l_wi = normalize(l_pos - P);
    float G=geom_fact_sa(P, l_pos, l_nor);
    float m_pdf=sample_material_pdf(m, n, wo, l_wi);
    float mis_weight=balance_heuristic(l_pdf, m_pdf*G);
    vec3 brdf=evaluate_material(m, n, wo, l_wi);
    result+=brdf*mis_weight*G*clamped_dot(n, l_wi)*Le/l_pdf;
  }
  // Material sampling
  if (sample_material(m, n, wo, &m_wi, &m_pdf)) {
    Intersect i = intersect(P, m_wi);
    if (i.hit && i.mat.is_emissive) {
      float G=geom_fact_sa(P, i.pos, i.nor);
      float light_pdf=sample_lights_pdf(P, n, i);
      float mis_weight=balance_heuristic(m_pdf*G, light_pdf);
      vec3 brdf=evaluate_material(m, n, wo, m_wi);
      vec3 Le=evaluate_emissive(i, m_wi);
      result+=brdf*dot(m_wi, n)*mis_weight*Le/m_pdf;
    }
  }
  return result;
}
```

The function `sample_lights_pdf` in Listing 20-7 says at what probability the light sampling would choose this emissive surface point, given that it was queried with the position and normal at which we are currently evaluating.

Listing 20-7. *Sample lights pdf.*

```
float sample_lights_pdf(vec3 P, vec3 n, Intersect emissive_surface) {
  Object object = objects[emissive_surface.object_index];
  if (!object.material.is_emissive) return 0.0;
  float p = 1.0/NUM_LIGHTS;
  if (object.type == GeometryType::Sphere) {
    float r = object.size.x;
    float area_half_sphere = 2.0*PI*r*r;
    p /= area_half_sphere;
  } else if (object.type == GeometryType::Quad) {
    float area_quad = object.size.x * object.size.y;
    p /= area_quad;
  }
  return p;
}
```

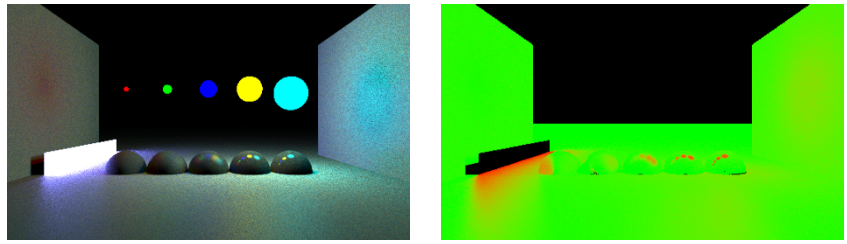

Figure 20-2. *MIS (left) and MIS weights (right). Red means that material sampling had larger weights; green means that light sampling had larger weights.*

It should give out the same probability for a surface as `sample_lights`. If `sample_lights` has an advanced scheme to select lights and adjust `p_choose_light` accordingly, then `sample_lights_pdf` must replicate that so it can give out the same probability. The same goes for `sample_materials_pdf`.

We can see the result in Figure 20-2 for our example with two techniques. There, we can also see the average MIS weight being used for the two techniques with green for light sampling and red for material sampling. If we compare Figure 20-1 with Figure 20-2, we see that we get less noise in the MIS render close to emissive objects. We also manage to find the emissive objects in the highly reflective spheres, something that light sampling failed to do. This is where material sampling is better than light sampling. Unlike material sampling we still have the low noise result that we did with light sampling when we move away from the emissive objects. It should be noted that while MIS give us a very practical way to combine multiple techniques, it would be far better if we could use a technique that choose samples according to the full product of both the material and the light sources—and maybe even taking the visibility into account!

We can think of MIS as doing two separate renders: one image using material sampling with the material MIS weight applied and one image using light sampling with the light sampling MIS weight applied. These two images are independent in the sense that they can choose completely different directions or surface points during sampling. They do not even have to use the same number of samples! The idea is just that they each do what they do best and leave the rest to the other render. Two such renders using each technique with MIS weight applied can be seen in Figure 20-3.

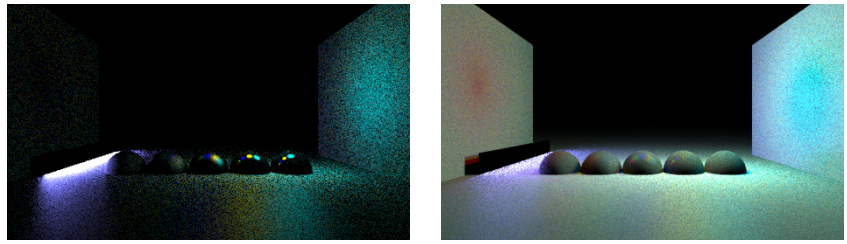

Figure 20-3. *Material sampling contribution (left) and light sampling contribution (right) (scaled by a factor of two).*

The choice to do MIS with a measure over surfaces instead of over solid angles was done to simplify this chapter. It is possible to do both, but the alternative is more complicated to describe. Our choice also makes it easy to use it in our path tracer in the next section. If we had instead chosen to do MIS using a measure for probabilities over solid angles, then we could still use the balance heuristic; it is usable as long as the probabilities use the same measure. Veach has a more general form of the balance heuristic where it can be used with any number of techniques and also with a different amount of samples from each technique. There are also alternatives to the balance heuristic [4, Section 9.2.3], but these are outside of the scope of this chapter.

20.2 A PATH TRACER WITH MIS

Now it is time to put it all together to get a full path tracer! We assume that you have written a path tracer before, so we will mostly add in the MIS of direct light in Listing 20-8.

The throughput—`tp`—says how future contributions should be weighted. If a path encounters a dark material, `tp` will be lower after bouncing the light.

The result can be seen in Figure 20-4 where we compare light sampling and MIS. Because we can reuse scene intersection from the material sampling ray to also do our bounce lighting, the only major extra work compared to a path tracer with only light sampling is the call to `sample_lights_pdf`, which is only invoked when our bounce light ends up at an emissive surface. So if you are lucky, your path tracer could get MIS more or less for free!

20.3 CLOSING WORDS AND FURTHER READING

In this chapter we discussed using MIS to handle direct lighting, but it can also be used to combine multiple ways to do bounce lighting or to balance

Listing 20-8. *Path tracing using MIS.*

```
vec3 pathtrace_mis(vec3 P, vec3 n, vec3 wo, Material m) {
  vec3 result = vec3(0), tp = vec3(1);
  while (true) {
    vec3 Le, m_wi, l_pos, l_nor;
    float l_pdf, m_pdf;
    // Light sampling
    if (sample_lights(P, n, &l_pos, &l_nor, &Le, &l_pdf)) {
      vec3 l_wi = normalize(l_pos - P);
      float G=geom_fact_sa(P, l_pos, l_nor);
      float m_pdf=sample_material_pdf(m, n, wo, l_wi);
      float mis_weight=balance_heuristic(l_pdf, m_pdf*G);
      vec3 brdf=evaluate_material(m, n, wo, l_wi);
      result+=tp*brdf*G*clamped_dot(n, l_wi)*mis_weight*Le/l_pdf;
    }
    // For material sampling and bounce
    if (!sample_material(m, n, wo, &m_wi, &m_pdf)) {
      break;
    }
    Intersect i = intersect(safe(P, n), m_wi);
    if (!i.hit) {
      break; // Missed scene
    }
    tp*=evaluate_material(m, n, wo, m_wi)*dot(m_wi, n)/m_pdf;
    if (i.mat.is_emissive) {
      float G = geom_fact_sa(P, i.pos, i.nor);
      float light_pdf=sample_lights_pdf(P, n, i);
      float mis_weight=balance_heuristic(m_pdf*G, light_pdf);
      vec3 Le = evaluate_emissive(i, m_wi);
      result+=tp*mis_weight*Le;
      break; // Our emissive surface doesn't bounce.
    }

    if (russian_roulette(&tp)) break;

    // Update state for next bounce; tp captures material and pdf.
    P = i.pos;
    n = i.nor;
    wo = -m_wi;
    m = i.mat;
  }
  return result;
}
```

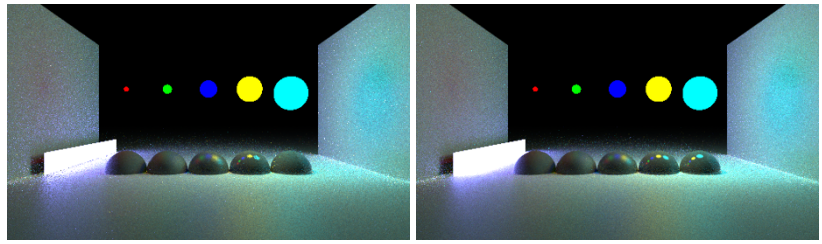

Figure 20-4. *Path tracing using light sampling (left) and MIS (right).*

different techniques to do volumetric lighting. It is a tool that can be applied whenever you have multiple techniques that each have pros and cons but that could work well together!

The concepts shown here are explained very well in Veach's doctoral thesis [4]. Kondapaneni et al. [1] proved that we can improve on the balance heuristic if we also allow negative MIS weights. Shirley et al. [3] show how to make better light sampling strategies for many types of individual light sources, and Moreau and Clarberg [2] present a system supporting scenes with many lights sources.

ACKNOWLEDGMENTS

Thanks to my editor Thomas Müller for extensive comments and mathematics help as well as source code for the reflective material model. Thanks to Alex Evans for reading and commenting on drafts, and lastly thanks to Petrik Clarberg for discussions on how to best present this topic.

REFERENCES

[1] Kondapaneni, I., Vévoda, P., Grittmann, P., Skřivan, T., Slusallek, P., and Křivánek, J. Optimal multiple importance sampling. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2019)*, 38(4):37:1–37:14, July 2019. DOI: 10.1145/3306346.3323009.

[2] Moreau, P. and Clarberg, P. Importance sampling of many lights on the GPU. In E. Haines and T. Akenine-Möller, editors, *Ray Tracing Gems*, chapter 18. Apress, 2019. http://raytracinggems.com.

[3] Shirley, P., Wang, C., and Zimmerman, K. Monte Carlo techniques for direct lighting calculations. *ACM Transactions on Graphics*, 15(1):1–36, Jan. 1996. DOI: 10.1145/226150.226151.

[4] Veach, E. *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford University, 1997.