Mestrado Integrado em Engenharia Informática

Image Based Lighting

Visualização e Iluminação II

Luís Paulo Peixoto dos Santos

Demo

Rendering with Natural Light; Paul Debevec; SIGGRAPH 19998 Electronic Theater
http://www.pauldebevec.com/RNL/

Image Based Lighting

- Processo de iluminar cenas e / ou objectos com imagens (mapas) de luz do mundo real

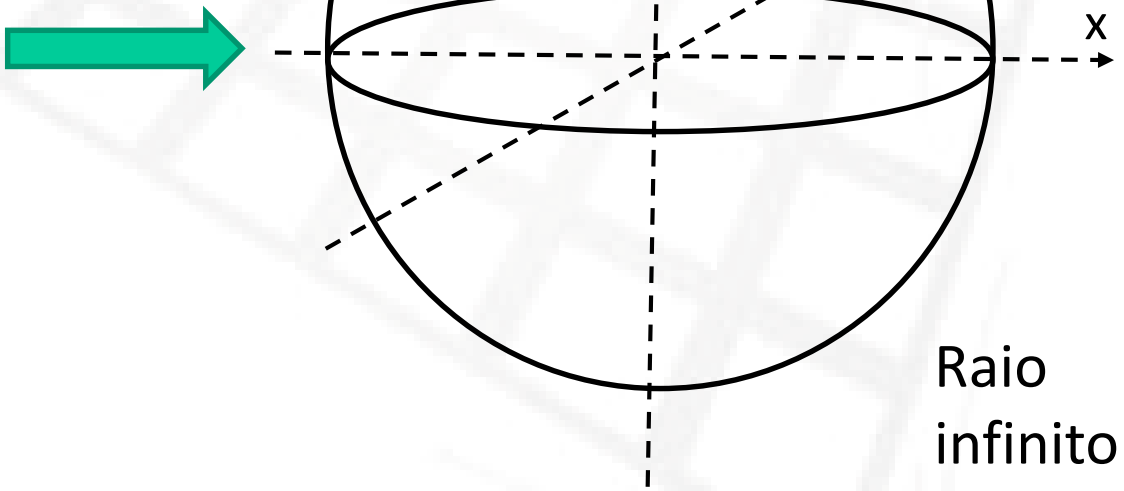
- Permite:
  - Sintetizar imagens com uma aparência realista;
  - Integrar objectos virtuais em cenas reais

Visão Geral

1. Mapa HDR omni-direccional da iluminação

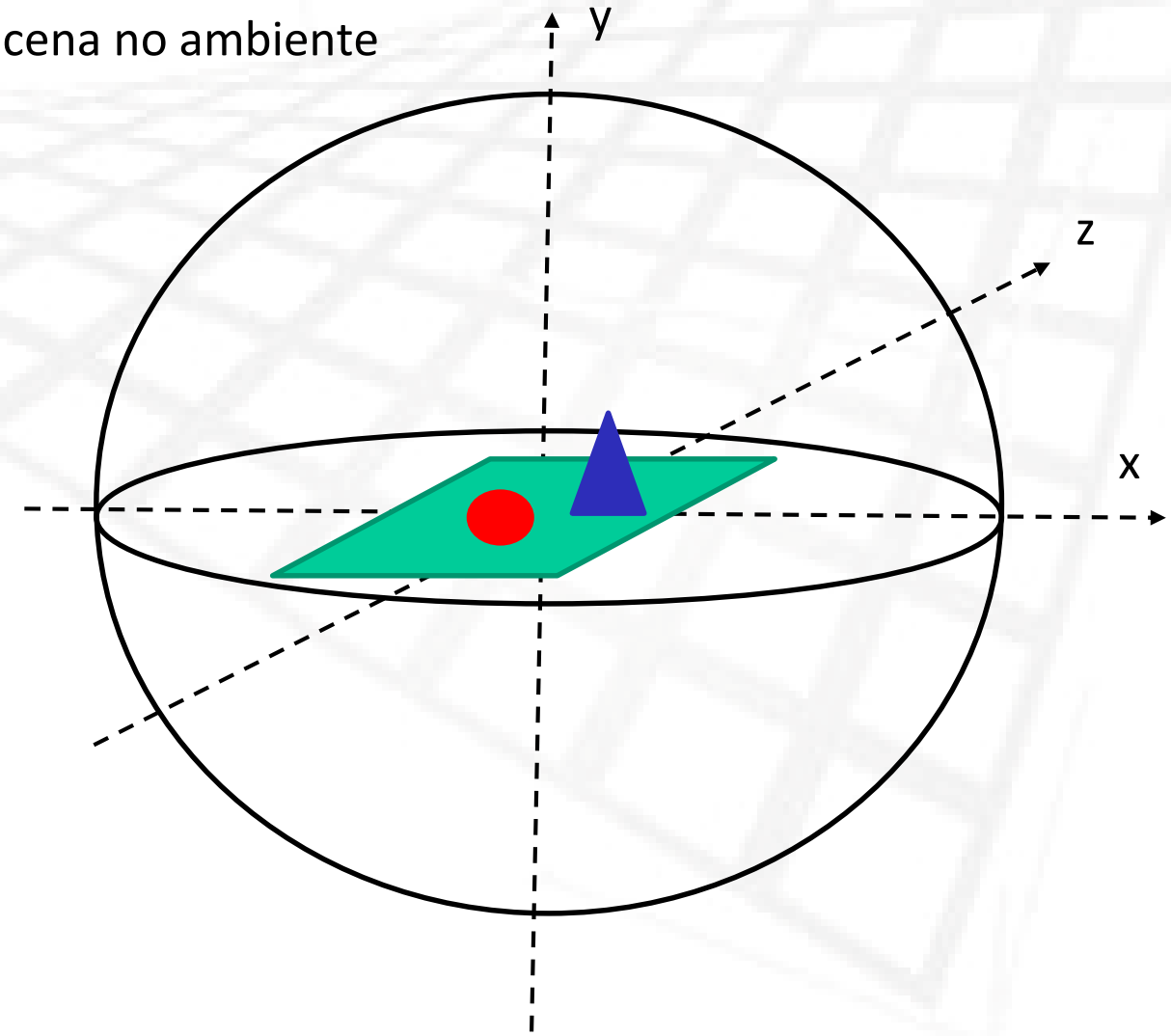
Visão Geral

2. Mapear a iluminação numa representação do ambiente



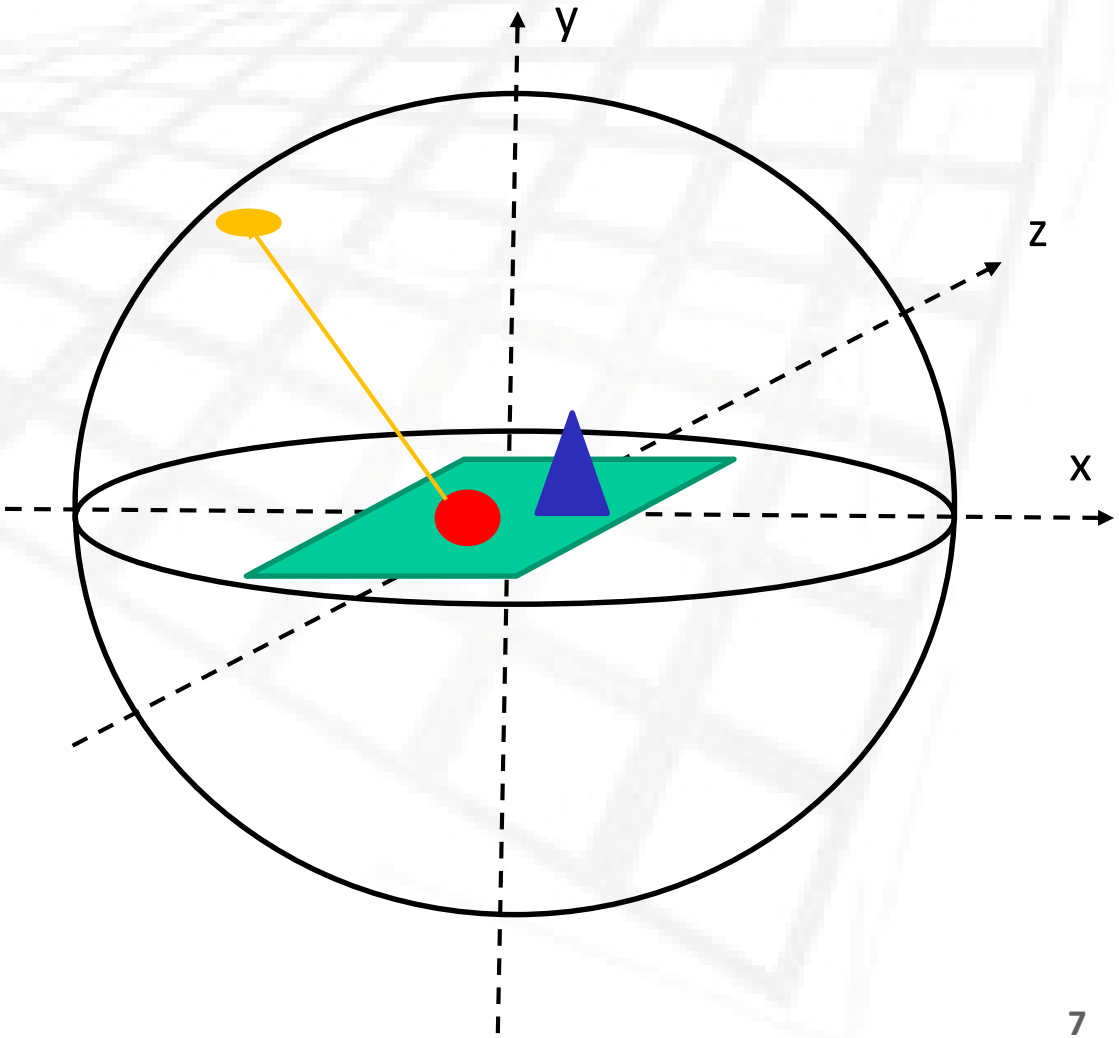
Visão geral

3. Posicionamento da cena no ambiente



Visão geral

4. Transporte de luz



Captura do mapa de Iluminação

1. Sintetizado usando um renderer (exemplo: câmara virtual esférica)

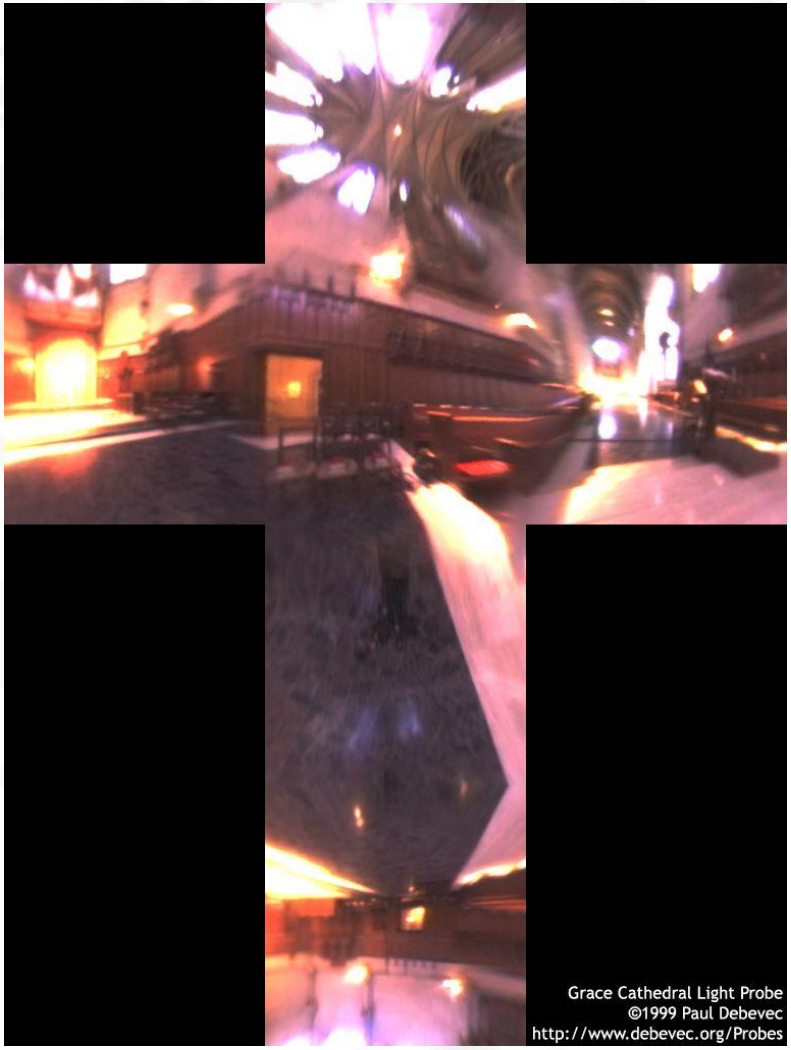
2. Sequência de fotos de uma esfera reflectora especular
   1. Cada foto da sequência capturada com uma exposição diferente
   2. A esfera reflecte os 4 PI estéreo-radianos (e não apenas 2 PI !)
   3. Ver probes em LuminanceHDR

3. Utilização de câmaras HDR e fish eye lenses

4. Galerias de *light probes*
   http://www.pauldebevec.com/Probes/

Mapear a iluminação numa representação do ambiente

- Existem diversos mapeamentos



Simulação de transporte de luz

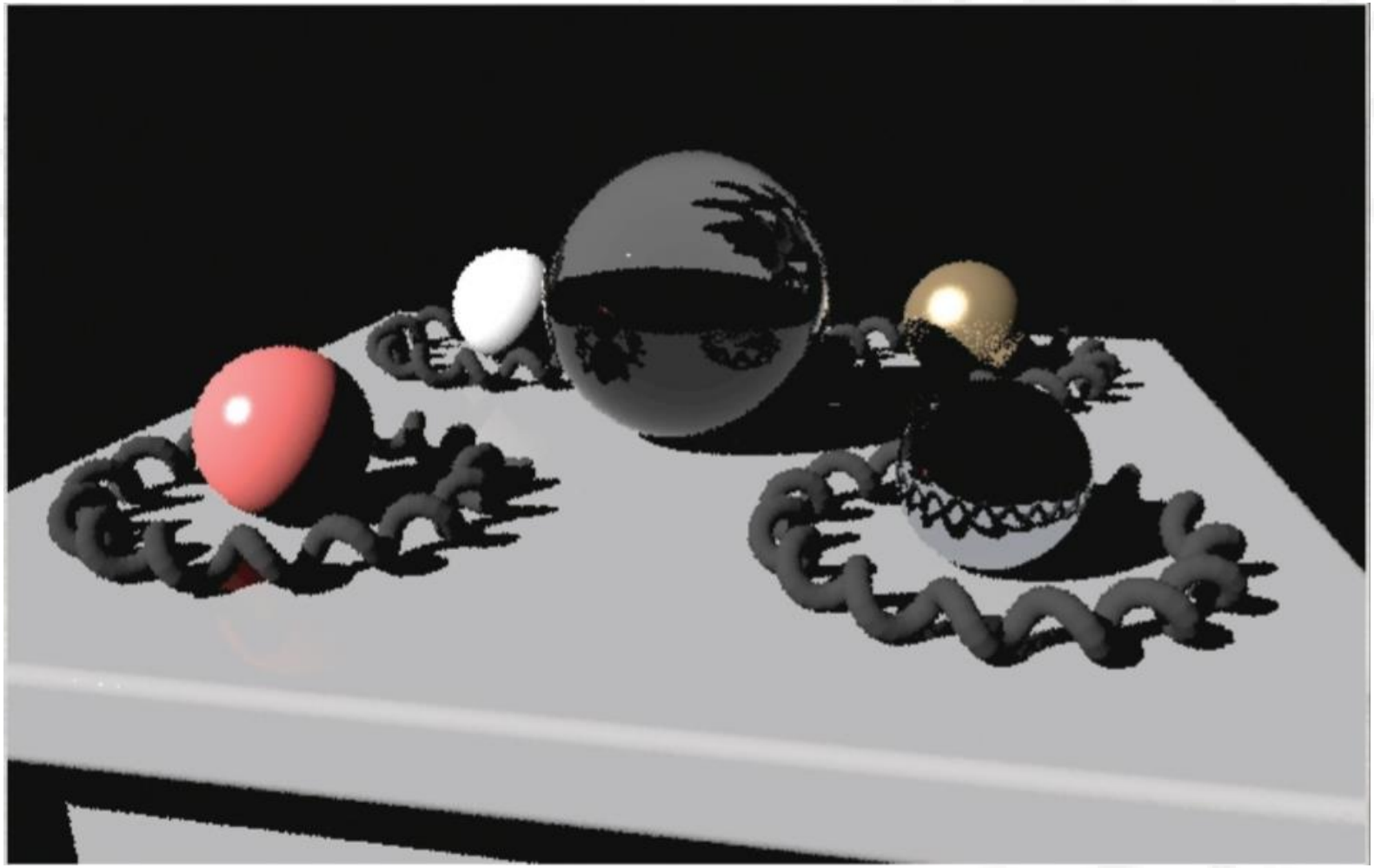
- Distribuir os *shadow rays* pela esfera (semi-esfera ???) definida pelo *environment map*
  - Uniforme, *cosine weighted, importance based*….

- Os raios primários e secundários que não intersectam qualquer geometria devem ter a sua contribuição lida do *environment map*
  - No caso dos raios primários isto corresponde à visualização directa do mapa

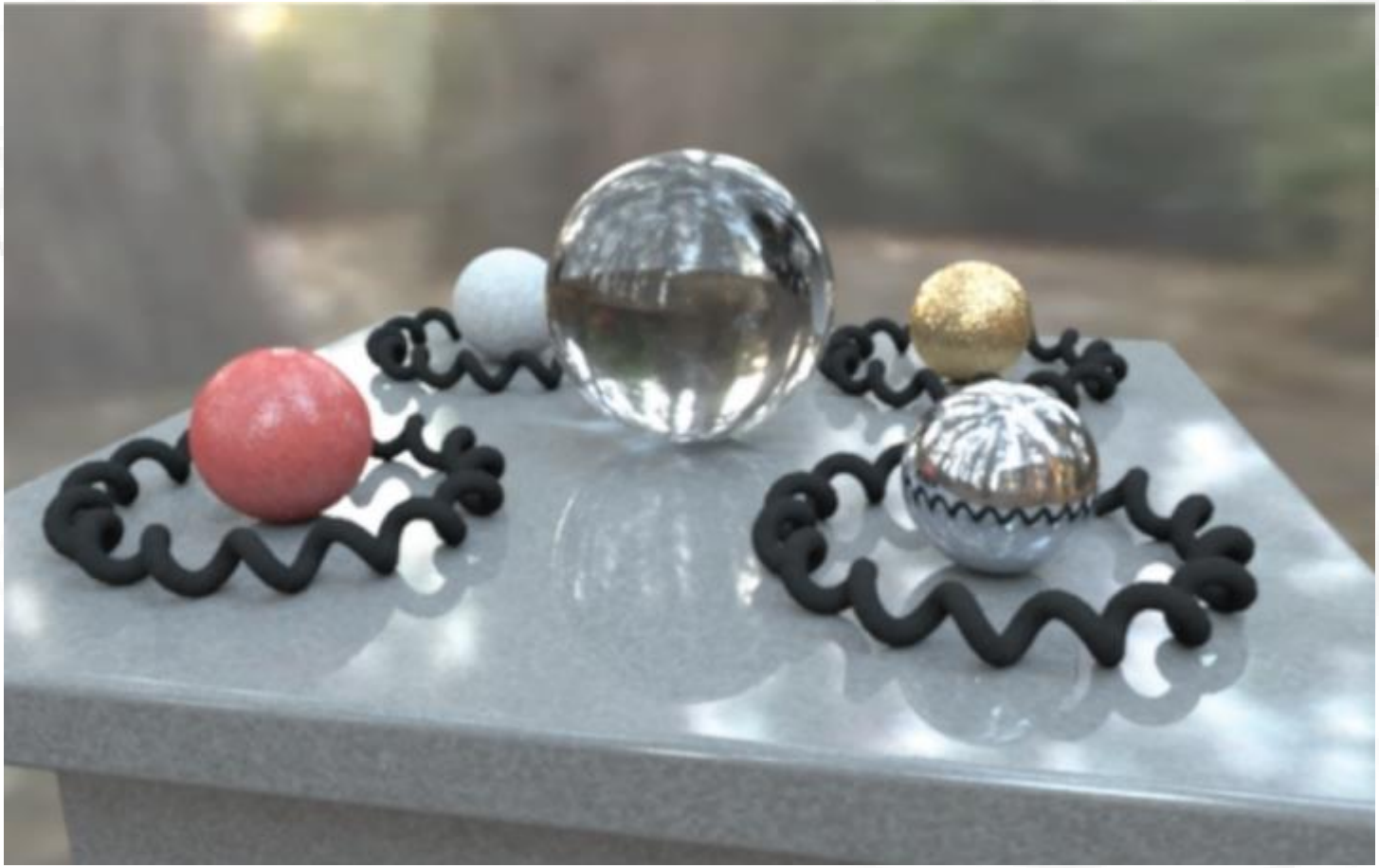
Simulação de transporte de luz

- A fonte de luz subentende um grande ângulo sólido (4*PI ou 2*PI)
  - Quanto maior for a frequência espacial do mapa, maior o ruído
  - Existem técnicas avançadas de amostragem para melhorar a amostragem

Resultados: Iluminação convencional



Resultados: IBL



Resultados: IBL

Implementação

```
Light_SampleRes InfiniteLight_sample(const Vec2f& s)
{
  Light_SampleRes res;    Vec2f map_coord;    Vec3fa radiance;

// uniform sample the sphere to get the direction
  const float phi = float(two_pi) * s.x;
  const float cosTheta = 1.f - 2.f * s.y;
  const float sinTheta = sqrt(1.0f - cosTheta*cosTheta);
  res.dir = cartesian(phi, sinTheta, cosTheta);
  map_coord = dir2map_coord(res.dir);
  radiance = map_lookup(self, map_coord);

  res.dist = inf;
  res.pdf = self->spherePdf;
  res.weight = radiance / res.pdf;

  return res;
}
```

Implementação

```
static Vec2f dir2map_coord(Vec3fa dir) {
  Vec2f res;
  float d, r;

  d = sqrtf(dir.x*dir.x + dir.y*dir.y);

  if (fabsf(d) < 1.e-4f) { r = 0.f; }
  else { r = 0.159154943f * acosf(dir.z); }

  res.x = .5f + dir.x * r;
  res.y = .5f + dir.y * r;

  return res;
}
```

Implementação

```
static Vec3fa map_lookup(const InfiniteLight* self, Vec2f map_coord) {
  int x, y, ndx;
  float *pix_addr;

  x = (int)(map_coord.x * self->hdr_map.width);
  y = (int)(map_coord.y * self->hdr_map.height);  // Radiance maps are -Y
  ndx = (y * self->hdr_map.width + x ) * 3;
  pix_addr = (float *)&self->hdr_map.cols[ndx];
  return Vec3fa(*pix_addr, *(pix_addr + 1), *(pix_addr + 2));
}
```