Final project report

Hugo Mateo Trejo 816678
Juan Lorente Guarnieri 816020

1 Human skin

Point light seen through a thin solid cube, the dispersion profiles can be appreciated. Both models are similar, so a more detailed comparison would be required to compare their differences.

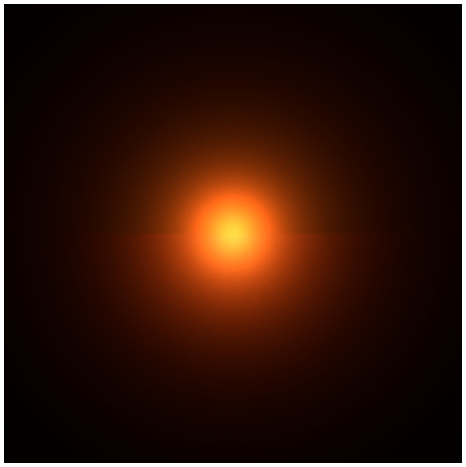

Figure 1: Dipole (up), normalized diffusion (down)

The outgoing point is sampled during pathtracing by importance sampling the model, as opposed to precomputing incoming surface light which decreases convergence speed but allows sample reusage. The outgoing direction is cosine weighted sampling.

The BRDF of the Figure 2 is a layered material that combines a base BSDF with a top translucent BSDF. The top BSDF is integrated for each light sample and the rest of the radiance is scattered into the base.

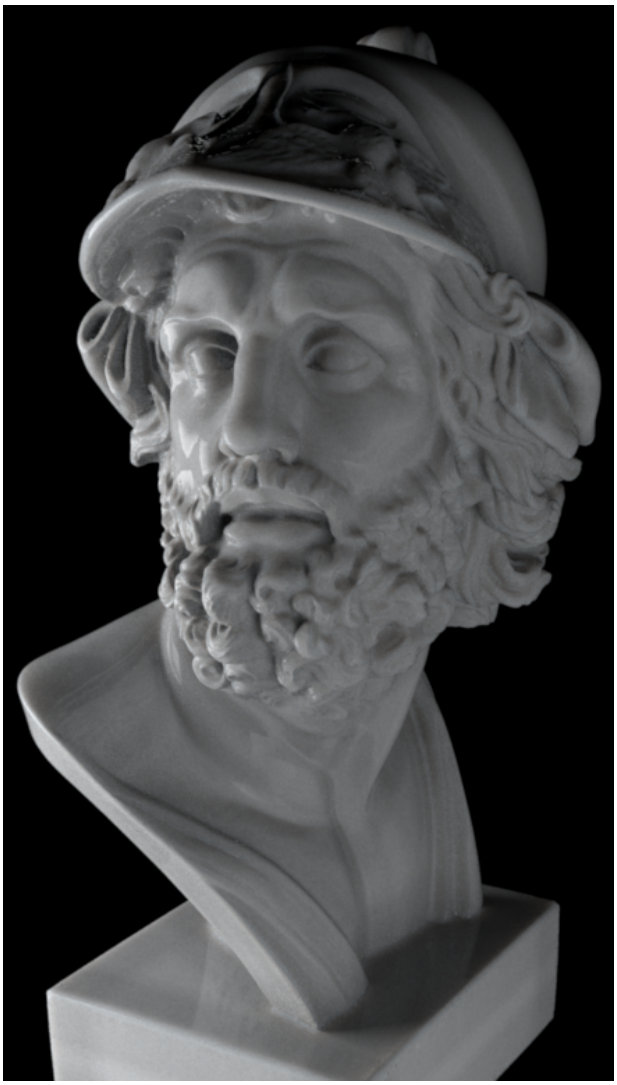

Figure 2: Coated marble
Layered material with glossy beckmann layer and dipole BSSRDF

2 BVH optimizations

The final scene has over 20mill triangles, so some optimizations to the provided BVH are interesting.

2.1 Data reordering

Faces and vertices can be reordered with a proximity heuristic, so that triangles close in 3d space are also close in memory, which improves cache coherency significantly [1][2] and may help the partition heuristic find better partitions. The used heuristic is Morton codes [4], also known as a Z-curve, which projects multi-dimensional coordinates into one dimension preserving spatial locality.

2.2 Priority traversal

The provided BVH traverses the left and right child in that order. However using an heuristic to choose which to traverse first may lead to clipping the ray earlier. Traversing the AABB with the closest intersection first is a weak heuristic, but in practice it yields really good results.

2.3 Performance measurements

To measure non-sorted performance without dependency on the obj exporter, a random sort has been used. The scene has 22 million triangles rendered with path_mis and 64 million samples.

	BVH build time	Render time	Cache misses	Total exec time
Not optimized	9.3s	126s	33.11%	221s
Morton sort	7.2s	102s	27.13%	202s
Priority traversal	9.3s	66s	25.32%	162s
Fully optimized	7s	53s	20.8%	154s

As stated before, changing the data ordering reduces cache misses by 6%. It is notable that data reordering speeds up both the execution and the BVH build time.

Priority traversal helps clipping the ray earlier, and produces the most significant speedup. Since it traverses fewer BVH nodes it also has fewer cache misses, but it is not a relevant measure in this case.

With all the optimizations, the speedup against the original when building+rendering (no loading from disk) is 2.25.

3 Anisotropic beckmann

The basic beckmann formula has been extended into the 2 dimensional alpha/roughness formula.

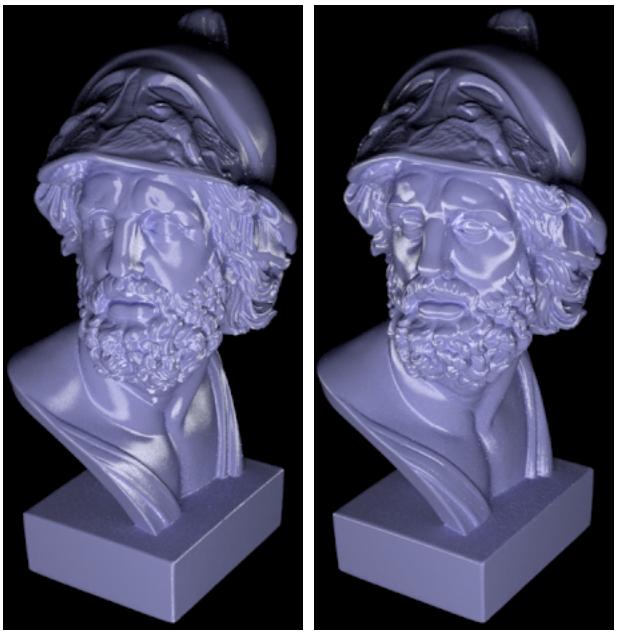

Figure 3: Beckmann with alpha:
X=1, Y=0.05 (left); X=0.05, Y=1 (right)

As seen in Figure 2, smoothness creates noticeable specular highlights on the smooth axis.

4 Final render scheme

The 3D model came in a single mesh without materials. Each mesh was separated automatically with a heuristic based on normal cosines, average mesh roughness and vertex proximity, and each material and texture was hand-painted. The image has been exported with reinhard tonemapping from [3], controlled by luminance scale, light adapt and color adapt.

- **Head:**
  - **Skin:** Multi-layered material
    - Top oily layer: Beckmann microfacet with textured roughness
    - Dermal layers: Normalized diffusion SSS with textured effective albedo
  - **Hair:** Aggregate material (weights two materials)
    - 80%: Anisotropic Beckmann
    - 20%: Dipole SSS model
  - **Eyes:** Aggregate material
    - 87.5%: Lambertian with texture
    - 12.5%: Perfect mirror
- **Body:**
  - **Fabric coat:** Oren-Nayar
  - **Gold and armor:** GGX with metal Fresnel
- **Background:**
  - Area light with blurred sky texture
- **Lights:**
  - Background point lights for light translucency
  - Area lights for general illumination

References

[1] Tero Karras. *Maximizing Parallelism in the Construction of BVHs, Octrees, and k-d Trees*. 2012. URL: https://research.nvidia.com/sites/default/files/pubs/2012-06_Maximizing-Parallelism-in/karras2012hpg_paper.pdf.

[2] Tero Karras. *Thinking Parallel, Part III: Tree Construction on the GPU*. 2012. URL: https://developer.nvidia.com/blog/thinking-parallel-part-iii-tree-construction-gpu/.

[3] Erik Reinhard and Kate Devlin. "Dynamic Range Reduction Inspired by Photoreceptor Physiology". In: *IEEE transactions on visualization and computer graphics* 11 (Jan. 2005), pp. 13–24. DOI: 10.1109/TVCG.2005.9.

[4] *Z-order curve*. Wikipedia. URL: https://en.wikipedia.org/wiki/Z-order_curve.