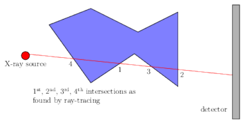
Topic 3 – Implementing the Beer-Lambert Law on GPU using OpenGL

IBSim-4i 2020

Dr Franck P. Vidal

13th Aug 2020

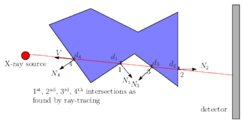
Path Length: Naive Approach



Is finding intersections in the right order important?

1. Detect every intersection between a ray and the objects;
2. Sort intersection (Can be handled by GPUs using depth-peeling, a multi-pass rendering technique for semi-transparent polygonal objects without sorting polygons);
3. Compute path length.

Path Length: L-Buffer



Finding intersections in any order does not matter

Intersection sorting is actually not needed!

- By convention normals are outward;
- A ray penetrates into an object when the dot product between the view vector (V) and the normal (Ni) at the intersection point is positive;
- It leaves an object when the dot product is negative.

L-Buffer Implementation

$Lp=\Sigma i$ - $sng(V \cdot Ni)$ x di

- i refers to ith intersection in an arbitrary order;
- di distance from X-ray source to intersection point;
- $sgn(V \cdot Ni)$ stands for the sign of the dot product between V and Ni;
- In a shader program, compute:
  - $sgn(V \cdot Ni)$;
  - di the distance between the X-ray source and the intersection;
  - Assign $-sng(V \cdot Ni)$ x di as the fragment value.
- For each pixel, compute Lp thanks to high-dynamic range and OpenGL blending function (pixel values may not be between 0 and 1).

See http://dx.doi.org/10.2312/LocalChapterEvents/TPCG/TPCG09/025-032 for more details.

Multipass Rendering Pipeline

pixel = E x Nout

pixel = E x Nin(E) $e(-\Sigma i$ i $Lp(i))$

- Needs 3 FBOs with high-dynamic range capability for off-line rendering:

- For each object of the scene:

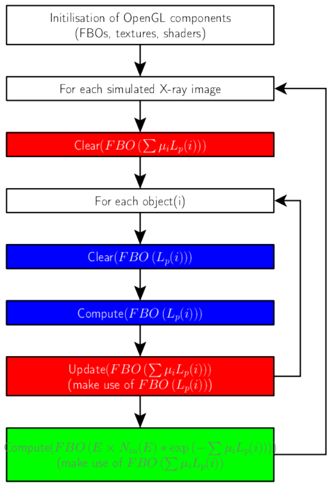
  1. Compute Lp(i);
  2. Update results of Σ i Lp(i).

- For the final image only:

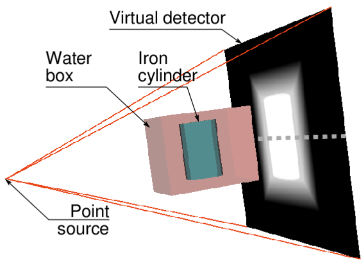
  1. Compute Nout;
  2. (Optional when direct display only is needed).

Adding the Beam Spectrum

- Take into the different energies within the incident beam;
- This is known as *beam hardening*;
- Iterate over several energy channels:
  - pixel = Σj Ej x Nout(Ej)



OpenGL pipeline to compute the Beer-Lambert law (monochromatic case).



Simulation parameters

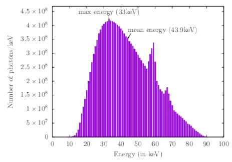
  - pixel = Σj Ej x Nin(Ej) e(-Σi i(Ej,,Z) di)
- Example:

Simulation with Different Source Shapes

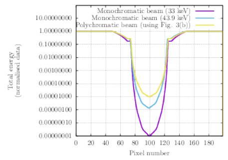
- Take into account the focal spot of the X-ray source;
- Iterate over several point sources within the volume of the focal spot:
  - pixel = Σk Σj Ej x Nin(Ej) e(-Σi i(Ej,,Z) di(k))
  - See blur in the corresponding image below.

Final Simulation Flowchart

- Iterate over several energy channels: Three extra for loops;
- Iterate over several point sources within the volume of the focal spot: One extra for loop.



Polychromatic beam spectrum for 90kV X-ray tube peak voltage



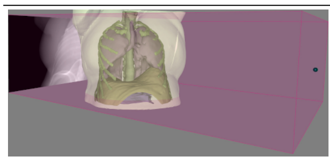
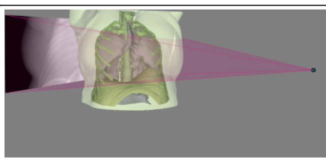
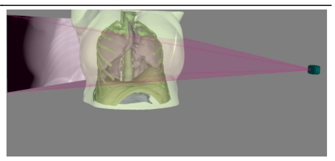
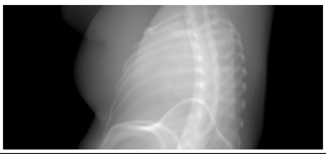
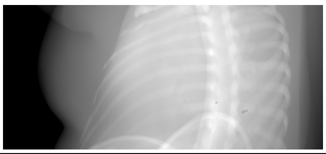
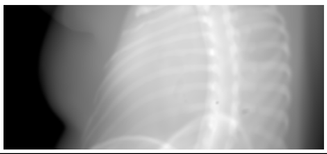
Intensity profiles, see dash line in image above

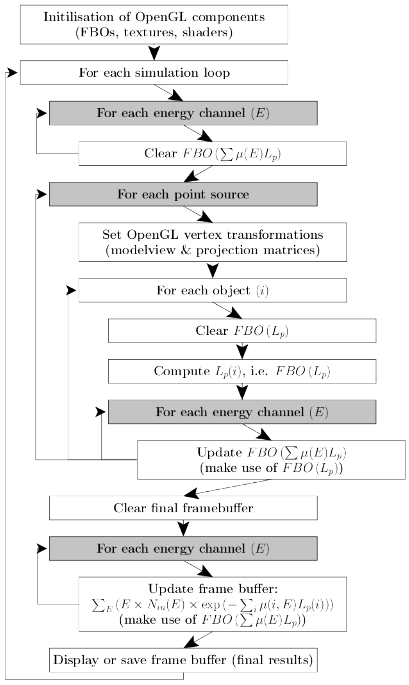
Bibliography (links)

- DOI: 10.2312/LocalChapterEvents/TPCG/TPCG09/025-032
- DOI: 10.1007/s11548-009-0367-1
- DOI: 10.2312/egp.20101026
- DOI: 10.1016/j.compmedimag.2015.12.002

Back to main menu

Click here

Parallel beam	Infinitely small point source	13mm source
		
		



Final OpenGL pipeline