Sound field calculation with GPU

Fast under water ray tracing

Olav Haugehåtveit
haugehat@stud.ntnu.no

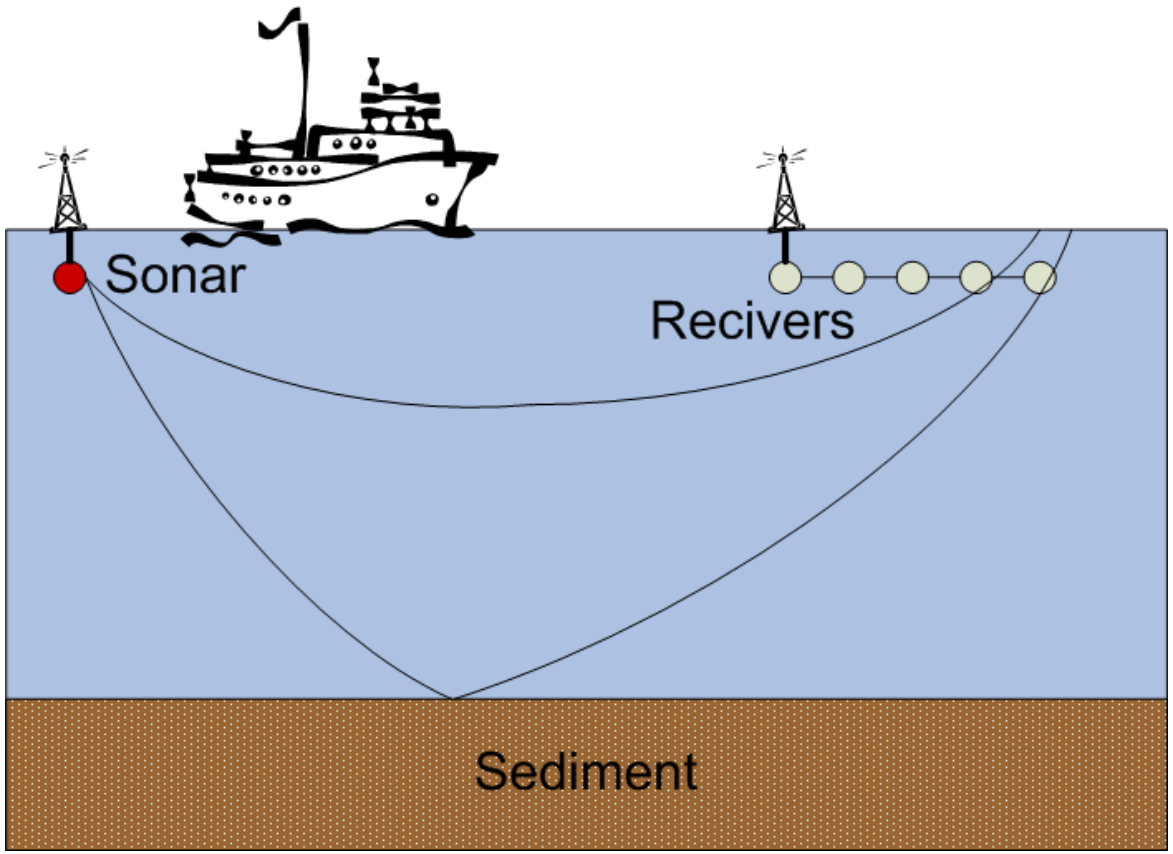
Department of Electronics and Telecommunication
Norwegian University of Science and Technology

2006 June 9.

Outline

Outline

Scenario



Ray propagation

Starting condition

- Current sound speed at source, $c(z)$
- Current depth of source, z
- Number of rays, n
- Starting angles for each ray, θ_{0n}
- Receiver location

Ray propagation

Starting condition

- Current sound speed at source, $c(z)$
- Current depth of source, z
- Number of rays, n
- Starting angles for each ray, θ_{0n}
- Receiver location

Ray propagation

Starting condition

- Current sound speed at source, $c(z)$
- Current depth of source, z
- Number of rays, n
- Starting angles for each ray, θ_{0n}
- Receiver location

Ray propagation

Starting condition

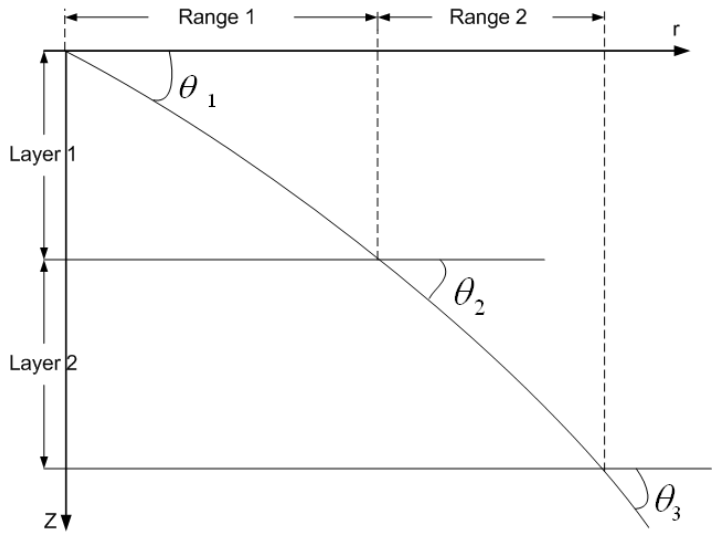
- Current sound speed at source, $c(z)$
- Current depth of source, z
- Number of rays, n
- Starting angles for each ray, θ_{0n}
- Receiver location

Ray propagation

Starting condition

- Current sound speed at source, $c(z)$
- Current depth of source, z
- Number of rays, n
- Starting angles for each ray, θ_{0n}
- Receiver location

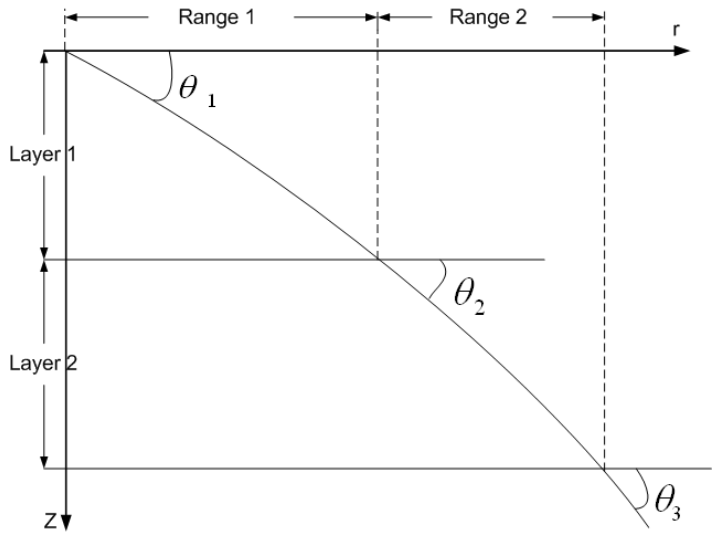
Ray propagation



Calculating a step

- Depth steps with $\pm\Delta$ layer
- Sum of all ranges will give the total range

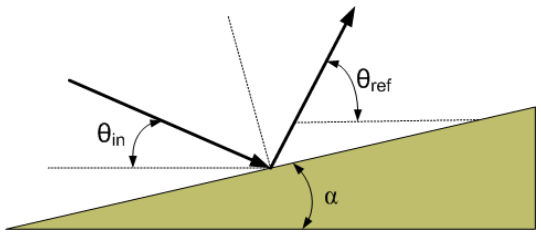
Ray propagation



Calculating a step

- Depth steps with $\pm\Delta$ layer
- Sum of all ranges will give the total range

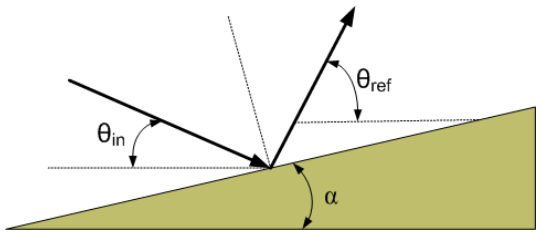
Bottom inclination



Bottom inclination

- $\theta_{ref} = \theta_{in} + 2\alpha$
- The ray will as consequence change direction in accordance with Snell's law.

Bottom inclination



Bottom inclination

- $\theta_{ref} = \theta_{in} + 2\alpha$
- The ray will as consequence change direction in accordance with Snell's law.

Values storage

Values to store

- Bottom and Surface reflection
  - Range
  - Travel time
  - Intersection angle
- Eigen values at receiver
  - Intersection angle
  - Range
  - Travel time
  - Depth
  - Start angle for ray

Values storage

Values to store

- Bottom and Surface reflection
  - Range
  - Travel time
  - Intersection angle
- Eigen values at receiver
  - Intersection angle
  - Range
  - Travel time
  - Depth
  - Start angle for ray

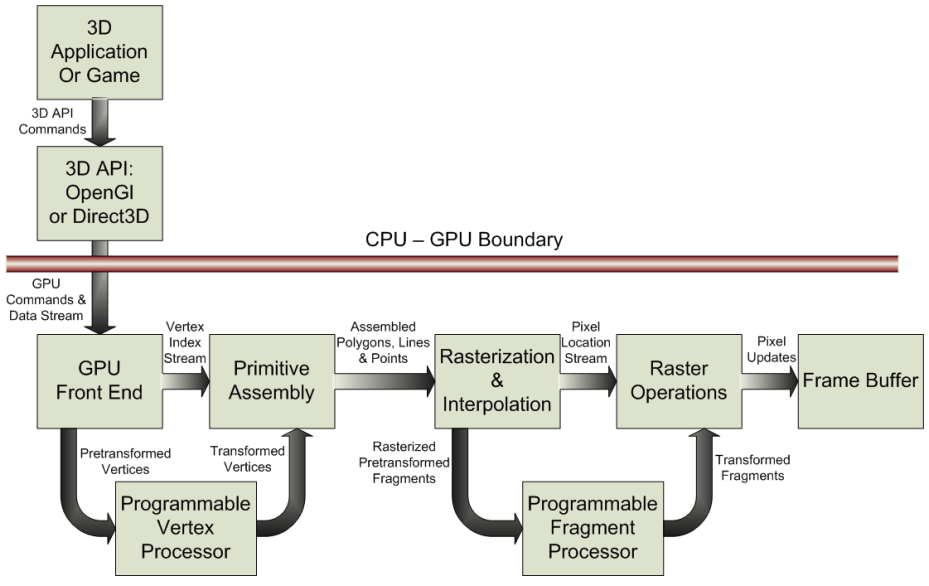
Values storage

Values to store

- Bottom and Surface reflection
  - Range
  - Travel time
  - Intersection angle
- Eigen values at receiver
  - Intersection angle
  - Range
  - Travel time
  - Depth
  - Start angle for ray

Outline

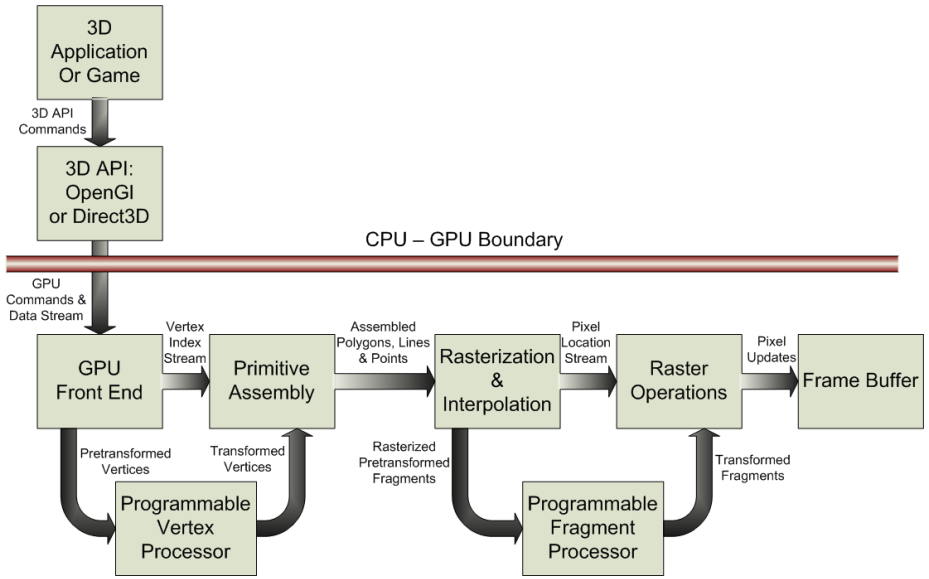
Overview



Processors

- Vertex processor (pass the data trough)
- Fragment processor (executes the math)

Overview



Processors

- Vertex processor (pass the data trough)
- Fragment processor (executes the math)

Vertex and fragment programs

Vertex shader

- Pass the initial data trough
- Sets up the space coordinates

Fragment shader

- A pixel is thought of as one ray
- 8x8 pixels will result in 64 rays
- Different values for every pixel
- All rays are computed in parallel
- Computes the range and travel time for every loop

Vertex and fragment programs

Vertex shader

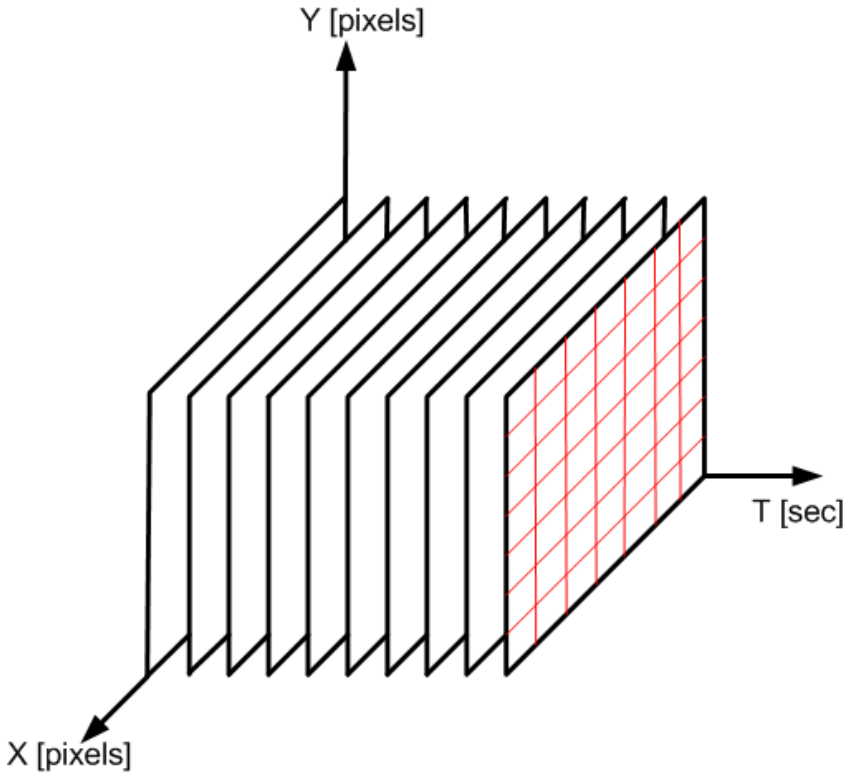
- Pass the initial data trough
- Sets up the space coordinates

Fragment shader

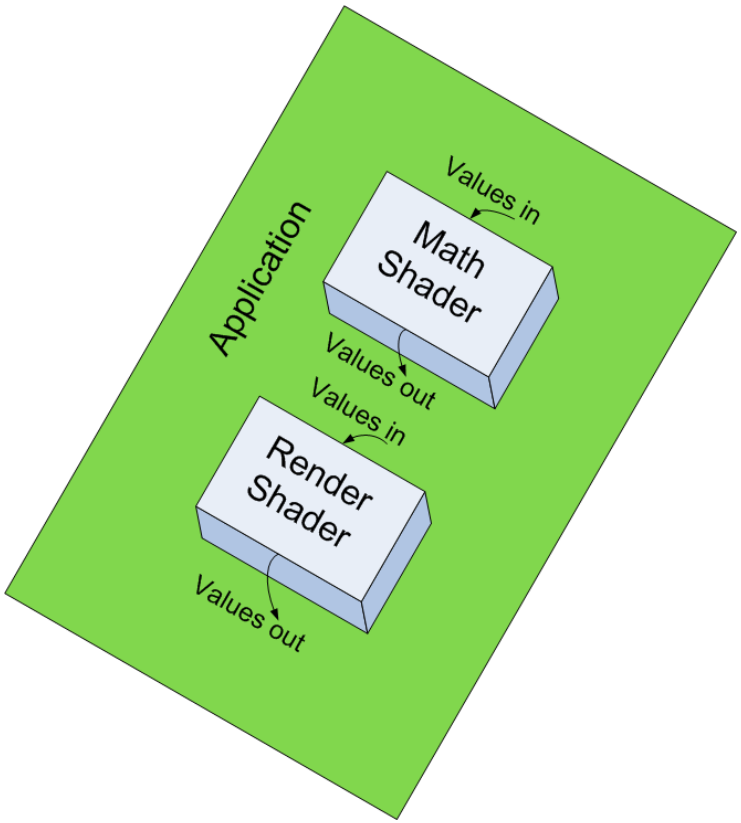
- A pixel is thought of as one ray
- 8x8 pixels will result in 64 rays
- Different values for every pixel
- All rays are computed in parallel
- Computes the range and travel time for every loop

Pixel matrix

Frame generation



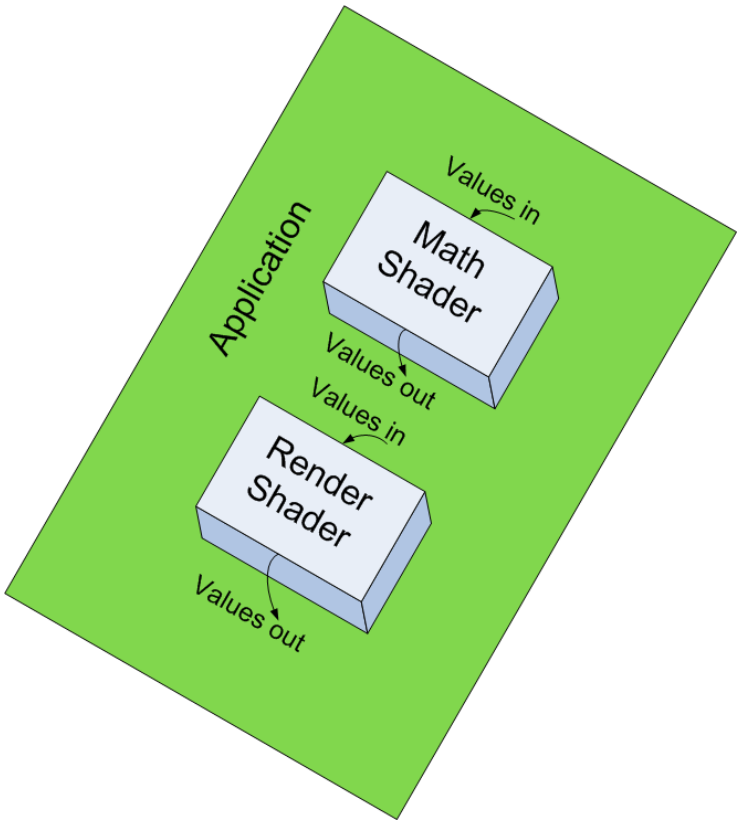
Looping



Looping

- Values read into math shader
- Results from math shader to screen
- Results from math shader set as input to next loop
- Ping-ponging

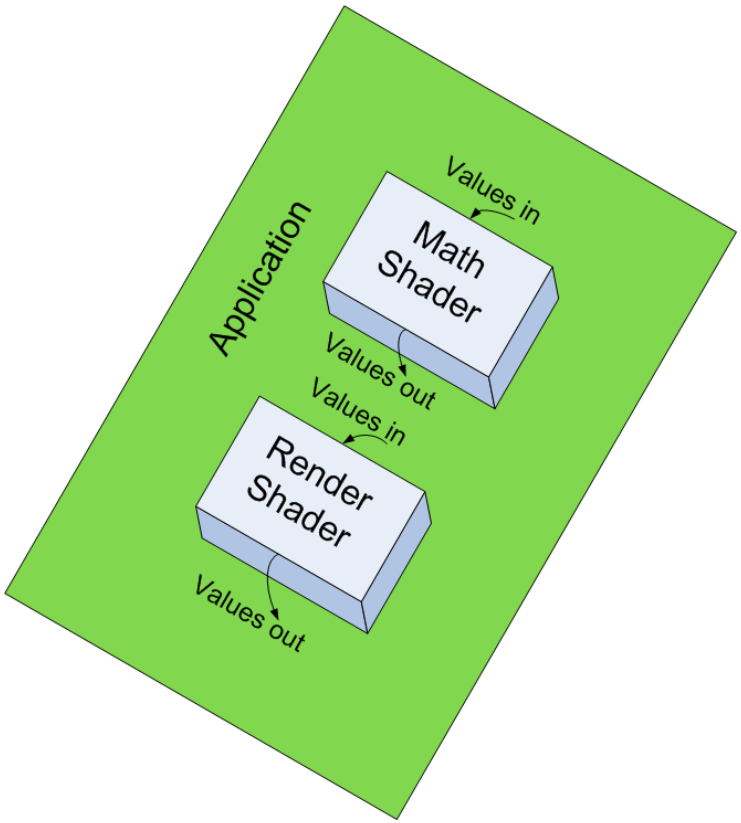
Looping



Looping

- Values read into math shader
- Results from math shader to screen
- Results from math shader set as input to next loop
- Ping-ponging

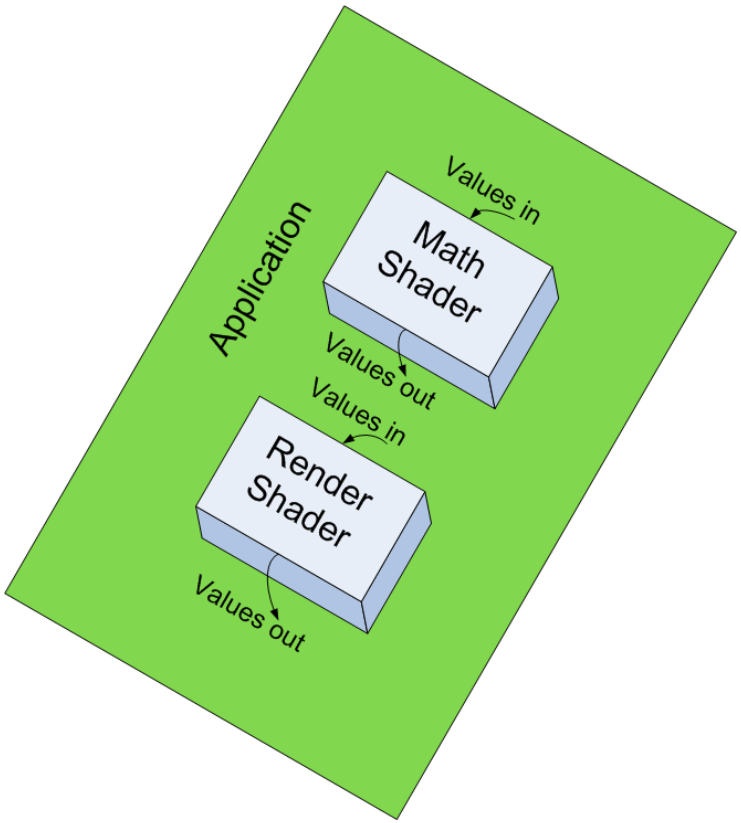
Looping



Looping

- Values read into math shader
- Results from math shader to screen
- Results from math shader set as input to next loop
- Ping-ponging

Looping



Looping

- Values read into math shader
- Results from math shader to screen
- Results from math shader set as input to next loop
- Ping-ponging

Read back

Asynchronous read back

- Read back of values will slow down the overall speed. (GPU-frame buffer-CPU-GPU)
- Asynchronous read back will speed up the process. (GPU-pixel buffer-GPU)

All values are stored in a text file for use i later stages.

Read back

Asynchronous read back

- Read back of values will slow down the overall speed. (GPU-frame buffer-CPU-GPU)
- Asynchronous read back will speed up the process. (GPU-pixel buffer-GPU)

All values are stored in a text file for use i later stages.

Outline

Theorem

Program demonstration

Conclusion

- Matlab is not optimized CPU code.
- GPU program is theoretically 200 times faster.
- GPU program is timed to be 45 times faster with 40000 rays.
- The GPU program is not optimized.

Summary

Summary

- The GPU has increased the speed by 40-50 times.
- Accuracy is the same
- The performance will increase when program is optimized

End

Questions?