Advanced Uses of Pixel Local Storage

Marius Bjørge
Graphics Research Engineer

ARM Developer Day - London
December 3rd 2015

Agenda

- Pixel Local Storage
- Indirect lighting pipeline

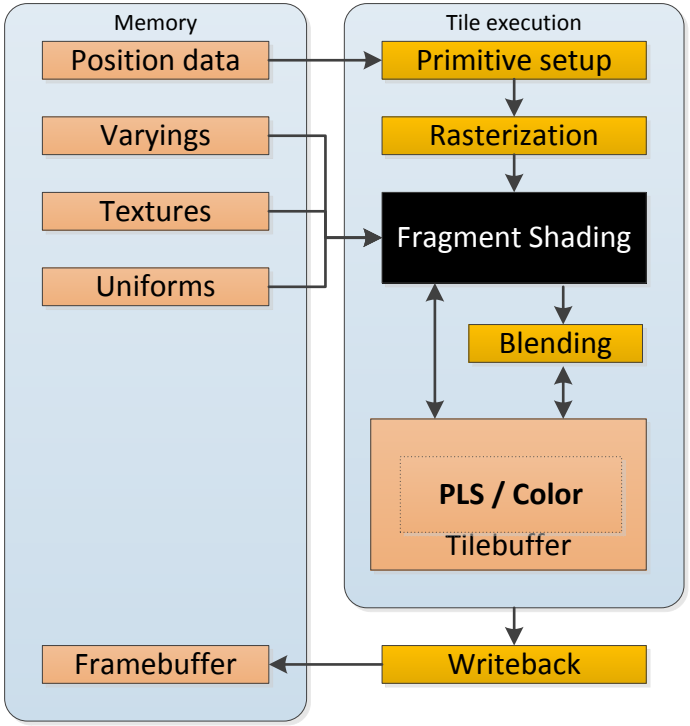
Pixel Local Storage

- Exposed as EXT_shader_pixel_local_storage
- Per-pixel scratch memory available to fragment shaders
  - Automatically discarded once a tile is fully processed
  - No impact on external memory bandwidth

- Shader declares a view of PLS memory
  - Re-interpret PLS between different passes
  - Can have separate input and output views
  - Independent of framebuffer format

Pixel Local Storage

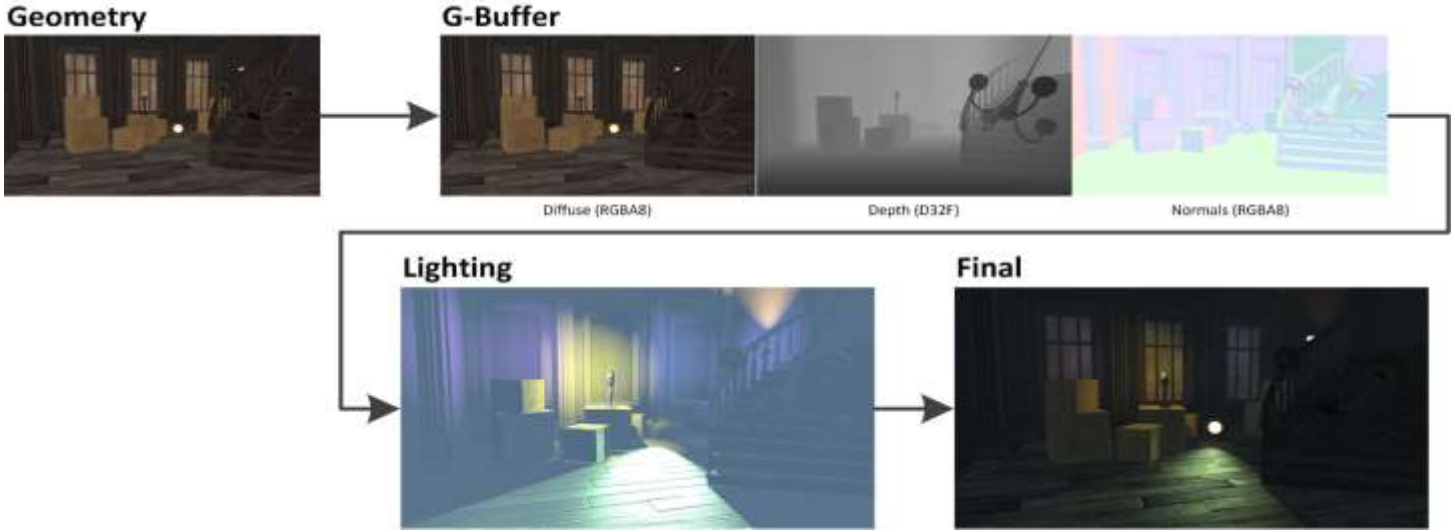
- Rendering pipeline changes slightly when PLS is enabled
  - Writing to PLS bypasses blending
- Note
  - Fragment order
  - Fragment tests still applies
  - PLS and color share the same memory location



ARM

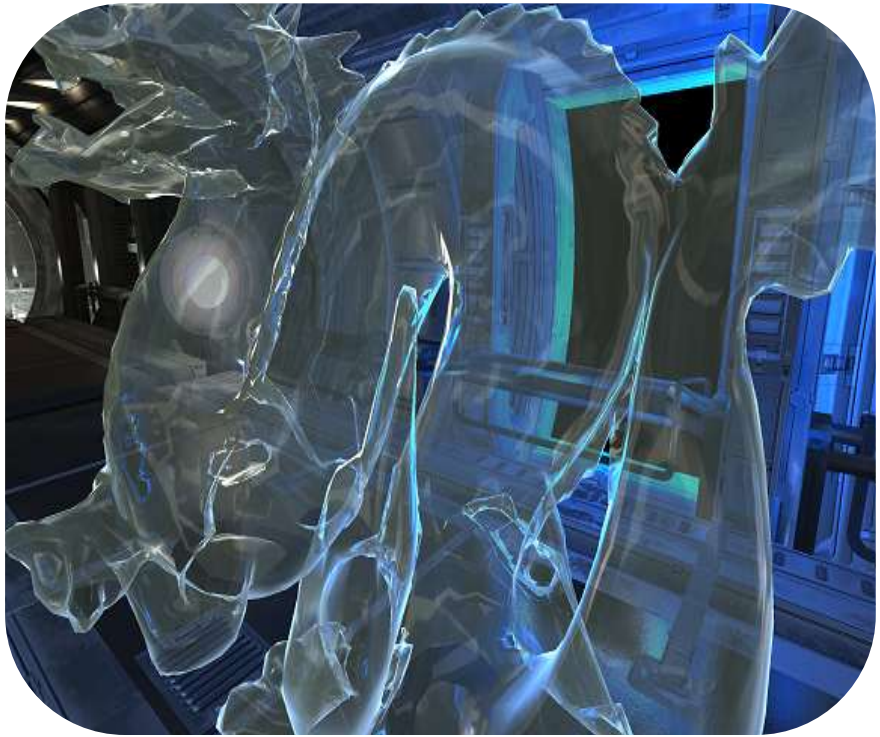
Deferred Shading

- Popular technique on PC and console games
  - Very memory bandwidth intensive
  - Traditionally not a good fit for mobile

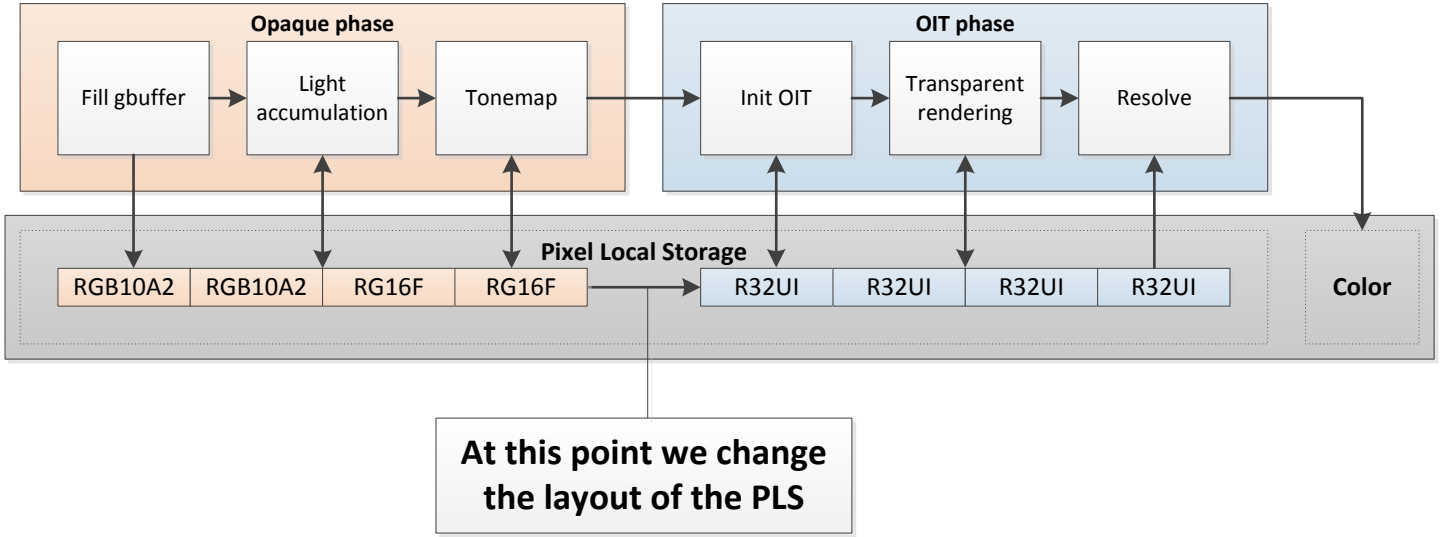


Order Independent Transparency

- Depth peeling
- Approximate approaches
  - Multi-Layer Alpha Blending [Salvi et al, 2014]
  - Adaptive Range



Pixel Local Storage



Sample Code

- http://malideveloper.arm.com/resources/sdks/mali-opengl-es-sdk-for-android/

Indirect Lighting Pipeline

Related Work

- Efficient Rendering With Tile Local Storage [Bjorge14]
- SH Irradiance Volumes [Tatarchuk05]
- Reflective Shadow Maps [Dachsbacher09]
  - Virtual Point Lights
- Light Propagation Volumes [Kaplanyan09]

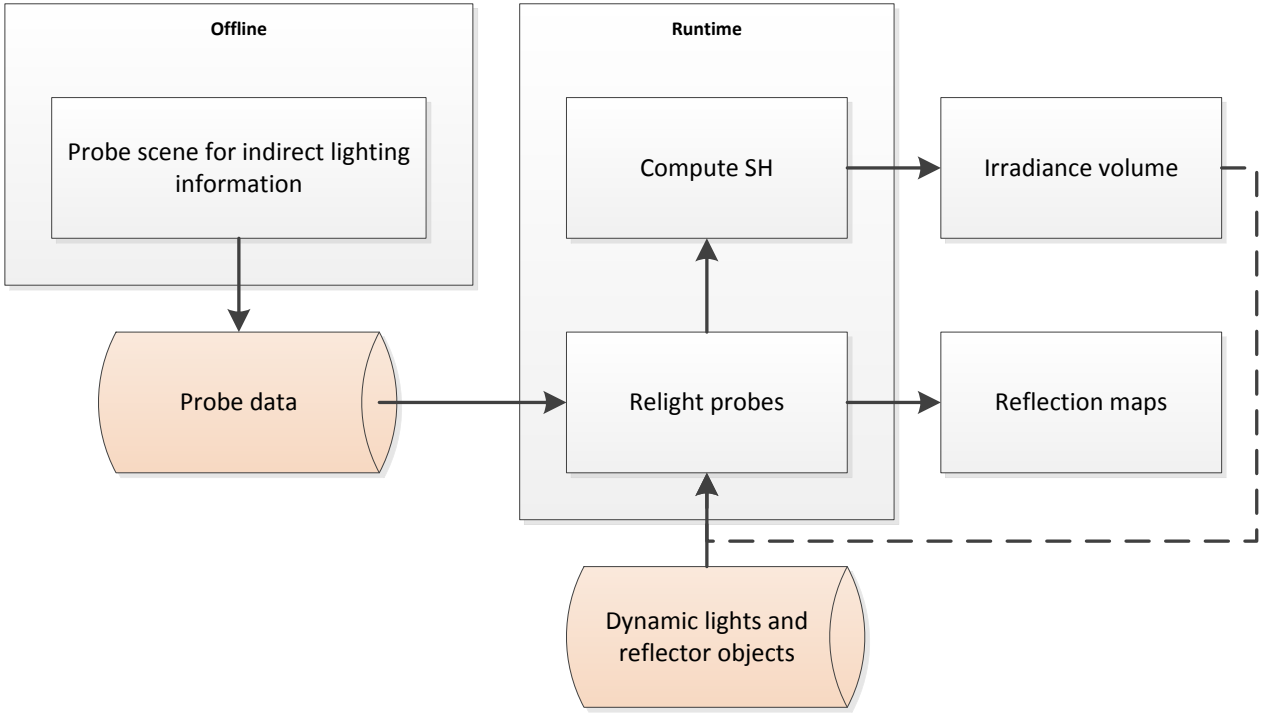
Global illumination with ARM

- Enlighten remains the GI solution of choice for mobile
  - Performant
  - High quality
  - Proven
  - PLS + Enlighten = bandwidth saving! [Bjorge14]
- This R&D project is separate
  - Exploration of PLS capabilities
  - Indirect lighting as first challenging use case
  - Results remain an approximation

The Idea

- A grid of probes are placed throughout the scene
- Albedo, normal and depth information is stored
  - Resolution can be quite low (64x64, 32x32 or lower)
  - Selectively store as either cubemaps/latlon maps or spherical harmonics
- Relighting does simple deferred shading style rendering per probe
  - Output: SH irradiance + reflection map

High-Level Overview



Direct Lighting

Indirect Lighting

Reflections

Final Image

Lighting Only

Offline

Offline – Probe Generation

- For every probe in grid
  - Render scene from every direction storing albedo, normal and depth
  - Save to a texture atlas
- Depth == 1.0
  - Hemisphere

Offline – Probe Storage

- High frequency
  - Cubemaps / latlon maps
- Low frequency
  - Spherical harmonics

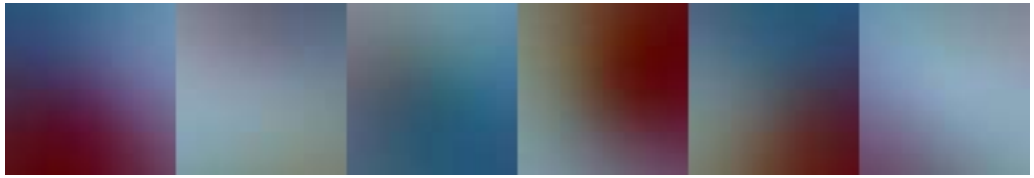
- Both are used in the demo application
  - Cubemaps for probes that are used for environment mapping
  - Spherical harmonics for the rest
  - Gives the best quality/performance trade-off

Offline – Probe Storage

- Cubemaps
  - 6kb

- Latlon
  - 2kb

- Spherical harmonics
  - 40 bytes



Offline – Meta Data

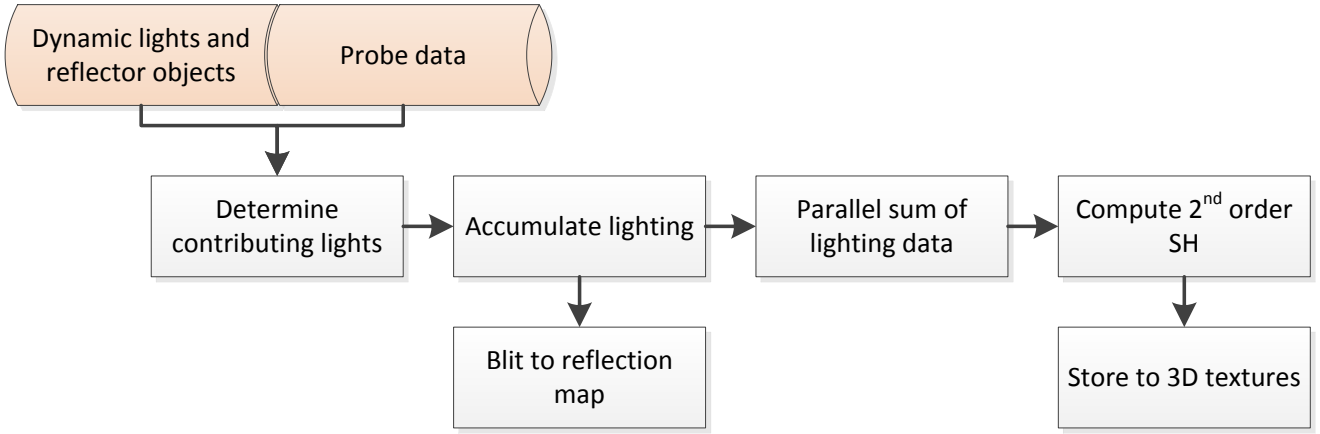
- Neighbour visibility information
  - Reduce amount of work required when updating probes
  - Snap texture coordinates to reduce light leakage
- Hemisphere visibility
  - Avoid updating probes that are never affected by global lights (such as the hemisphere or sun light)
- Local cubemap bounding box

Runtime

Runtime – Relighting

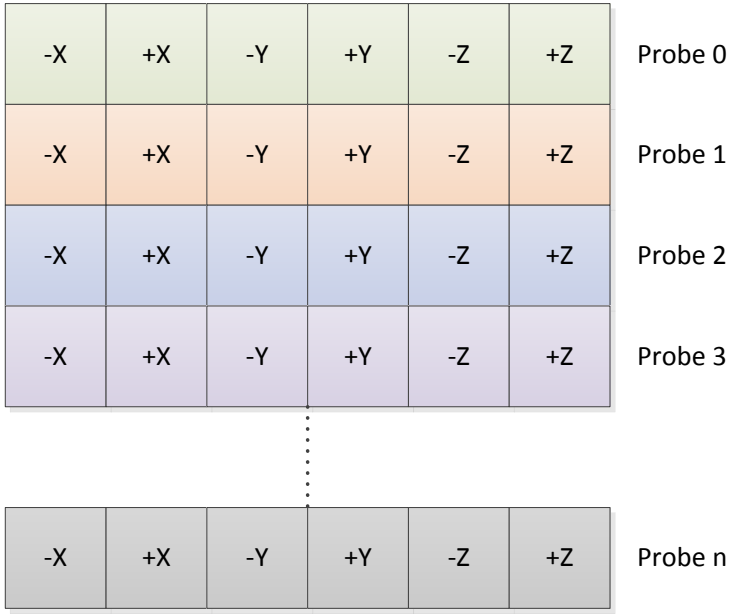
- First step is to determine which probes require updating
  - Probe meta-data is really useful here
- Dirty probes are pushed to queue for processing

Runtime – Per Probe Pipeline



Runtime – Deferred Probe Relighting

- Deferred shading implementation
  - PLS is really useful here
  - Sample from the baked albedo/depth and normal maps
  - Possible to re-use existing deferred shading lighting code path
  - Makes it easy to support custom light types and reflector objects
- Heavily batched



Runtime – Compute SH

- Parallel sum with spherical harmonics coefficients
  - Compute shader
  - … and/or multiple fragment reduction passes
- Output formats
  - Pack down to separate red, green and blue 3D irradiance textures
  - Store in FP16 for best visual quality

Runtime – Irradiance Volume

- 2nd order SH
  - 3x 3D textures for red, green and blue coefficients
- Simple 3D texture lookup using world-space position

Runtime – Reflections

- Output to cubemap matching size of probes
- glGenerateMipmap
  - Cubemap filter too expensive for runtime

Runtime – Multiple Bounces

- Temporal probe relighting
  - Slowly accumulate into irradiance volume

Single Bounce

Multiple Bounces

Future Work

- Move “everything“ to compute
- Octree representation
- Occlusion culling?

For more information visit the Mali Developer Centre:

http://malideveloper.arm.com

- Revisit this talk in PDF and audio format post event
- Download tools and resources

Thank you!

ARM

Questions?

References

1. http://www.ppsloan.org/publications/StupidSH36.pdf
2. http://www.crytek.com/cryengine/cryengine3/presentations/light-propagation-volumes-in-cryengine-3
3. http://developer.amd.com/wordpress/media/2012/10/Tatarchuk_Irradiance_Volumes.pdf
4. http://www.vis.uni-stuttgart.de/~dachsbcn/download/rsm.pdf
5. http://www.geomerics.com/wp-content/uploads/2014/11/Efficient-Rendering-with-Tile-Local-Storage.pdf