Rendering Decals and Many Lights with Ray Tracing Acceleration Structures

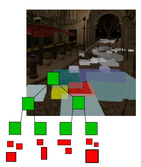
Sidney Hansen sidney.hansen@student.kit.edu

Christoph Peters christoph.peters@kit.edu

Karlsruhe Institute of Technology

Decal rendering methods

Ours:



- Intersect 0-length rays with bounding volumes
- Enumerate decals per shading point

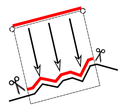
- Independent of shading technique
- Blending decals works
- Handles decals / lights outside view frustrum
- Ray tracing required

Deferred decals[1]:

- Rasterize bounding volume
- Apply decal to g-buffer inside fragment shader

- Performance
- Blending decals works
- Works only in deferred shading
- Limited to view frustrum

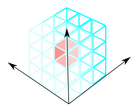
Mesh decals:



- Cut out and duplicate surface geometry
- Apply decal to copy

- Independent of shading technique
- Blending difficult
- Cutting out & duplicating geometry required
- Z-fighting

Decals in clustered shading[2]:

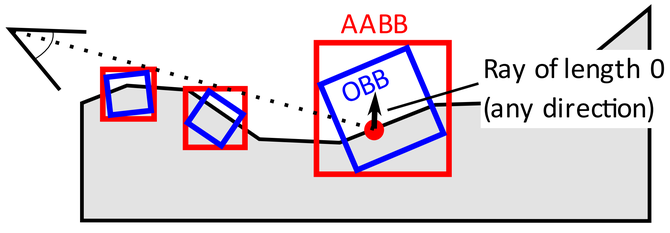


- Intersect bounding volume with grid
- List of decals per cell
- Apply decals in fragment shader

- Works well alongside lights
- Blending decals works
- Fixed grid size
- Limited to view frustrum

Our method

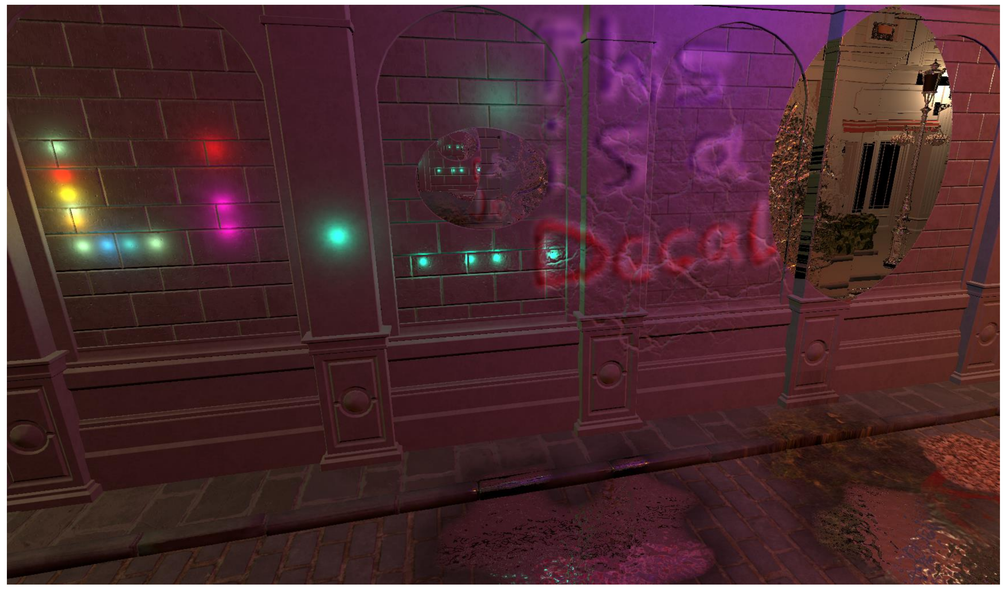
Similar to clustered shading our method works by enumerating decals (and light sources) inside a loop in the fragment shader.



- We use ray queries to trace zero-length rays from the fragment position
- Decal AABBs are stored in a ray tracing acceleration structure
- Any hit means a decal overlaps the shading point
- The same method works for light volumes
- Textures and other data are stored in buffers and accesed via index
- Sampling decal textures works the same way as with other techniques

- **Problem**: The order of the enumeration is arbitrary
  - For blending decals, we use a variation of weighted-blended order independent transparency (OIT)[3] using user defined **layers** and **weights**
    - **Weights:** used to determine the influence of the decal, when blending with other decals of the same layer
    - **Layers:** used to accumulate the weighted attributes of decals belonging to the same layer and are then blended in a fixed order using alpha blending
  - Light accumulation is order-independent anyway

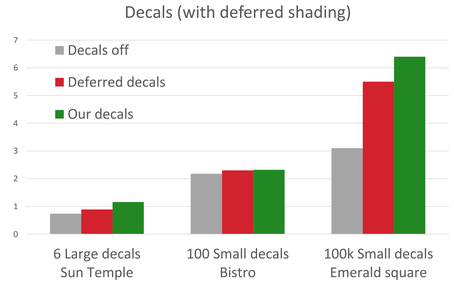
Results



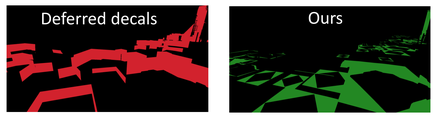
Bistro scene, rendered with decals and local point lights (ours). Shading technique: visibilitiy buffer[4] with ray traced reflections

Comparision to deferred decals

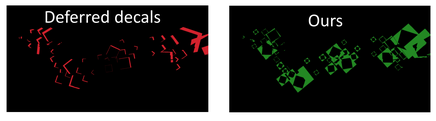
Frametime (in ms) captured on an RTX 2070 SUPER at a resolution of 1920x1017 pixels:



False positives with shallow angle:



False positives with steep angle:

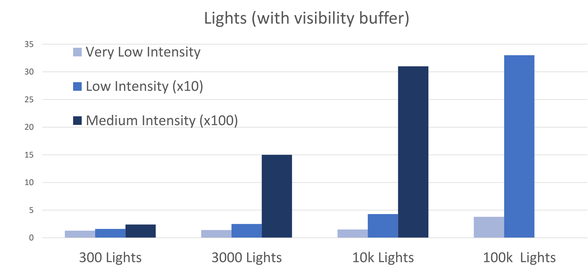


Blending

Floor of sun temple scene, rendered with decals and local point lights (ours). Shading technique: forward with depth prepass

Many lights

Bistro scene, rendered with 10k low intensity local point lights (ours). Shading technique: visibilitiy buffer



Conclusion

Performance:

- Overall lower perfomance than deferred decals
- Overhead increases for large decals / lights
- Scales well with high numbers of small decals / lights

Flexibility:

- Independent of shading technique
- Works with ray traced reflections
- Supports multi decal blending
- Ray tracing required

Future Work

- Evaluation with ray tracing (not only reflections)
- Comparision with clustered shading
- Implementation in dynamic setting

Source code: https://github.com/scratlantis/vulkan-renderer

References

1. Kim, Pope. "Screen space decals in warhammer 40,000: Space marine." *ACM SIGGRAPH 2012 Talks*. 2012. 1-1. https://doi.org/10.1145/2343045.2343053
2. Olsson, Ola, Markus Billeter, and Ulf Assarsson. "Clustered deferred and forward shading." *Proceedings of the Fourth ACM SIGGRAPH/Eurographics conference on High-Performance Graphics*. 2012.https://dl.acm.org/doi/10.5555/2383795.2383809
3. McGuire, Morgan, and Louis Bavoil. "Weighted blended order-independent transparency." *Journal of Computer Graphics Techniques* 2.4 (2013). http://jcgt.org/published/0002/02/09/
4. Burns, Christopher A., and Warren A. Hunt. "The visibility buffer: a cache-friendly approach to deferred shading." *Journal of Computer Graphics Techniques (JCGT)* 2.2 (2013): 55-69.http://jcgt.org/published/0002/02/04/