Light Culling Methods for Real-time Rendering Applications

Jared Watrous

Light Culling Methods

1. **Background & Goals**
2. Methods
3. Results

Real-time Rendering: Light Culling

For many pixels, some light sources will be negligible

Light is attenuated using the inverse square law:

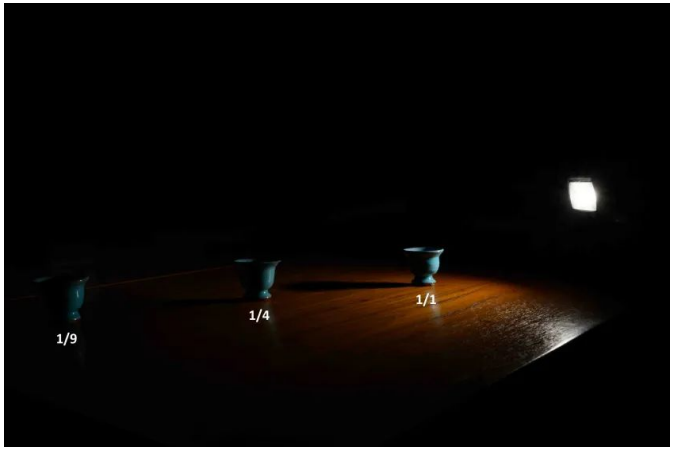
$$\text{Attenuation} = \frac{1}{\text{distance}^2}$$

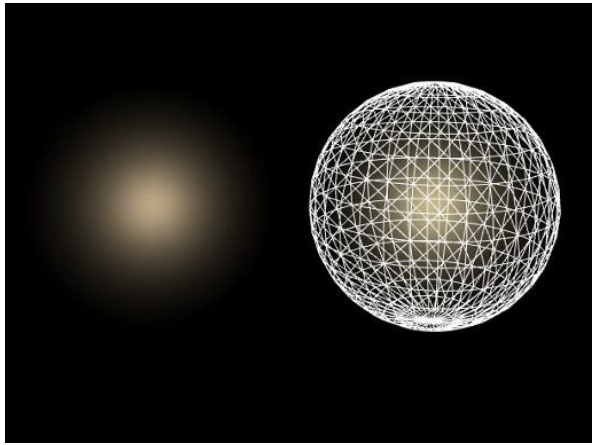
Derive a bounding sphere of influence (“light volume”):

$$\text{Radius} = \sqrt{\frac{\text{Power}}{\text{Threshold}}}$$

Choose a “threshold” such that any pixel outside “radius” is negligibly affected

“Light Culling”: in the pixel shader, determine whether the pixel should compute a light source





Principles & Constraints

An ideal light culling method satisfies the following constraints:

- At least as fast as no light culling
- Unbounded number of lights (no hard constraints)
- Scene-agnostic
- Transparent to the programmer/artist
- Minimal visual artifacts

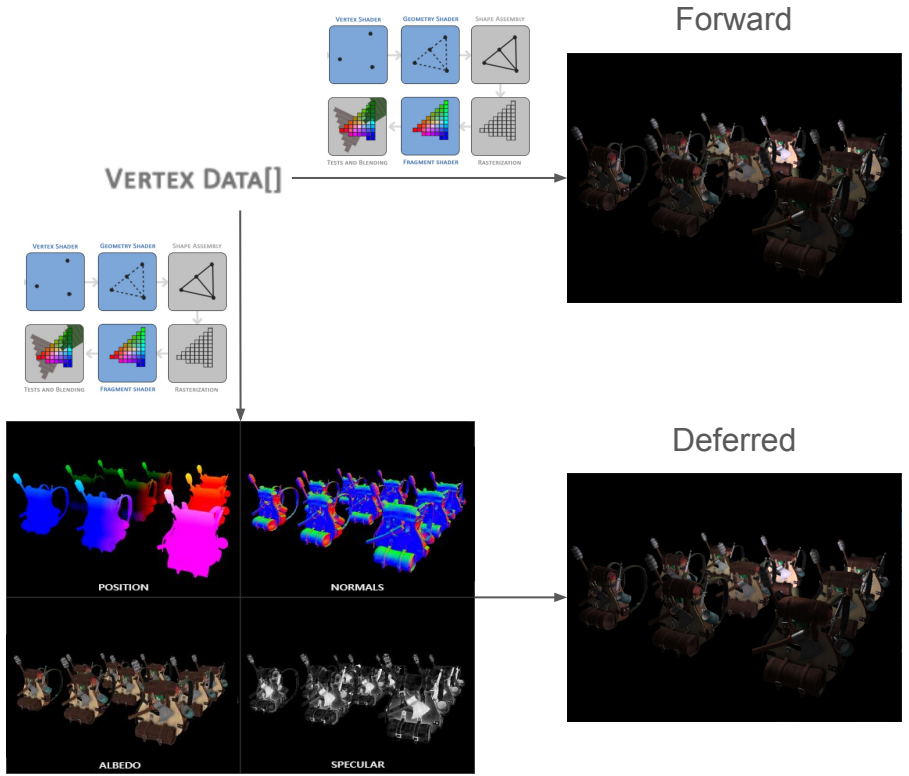
Forward & Deferred Rendering

Forward rendering:

- Rasterize directly to the screen
- Simple, low memory overhead
- Optional Z-prepass to reduce overdraw
  - (renders geometry twice)

Deferred rendering:

- First render material attribs to “G-buffer”
- Compute light sources in separate pass
- No overdraw in lighting pass
- More complex, high memory overhead
- Some features (transparency, AA) hard



Light Culling Methods

1. Background & Goals
2. **Methods**
3. Results

No Light Culling

Just iterate over all light sources

```
for light in LightsList:
    color += computeLight(light)
```

Works for:

- Forward
- Deferred

Bounding Sphere

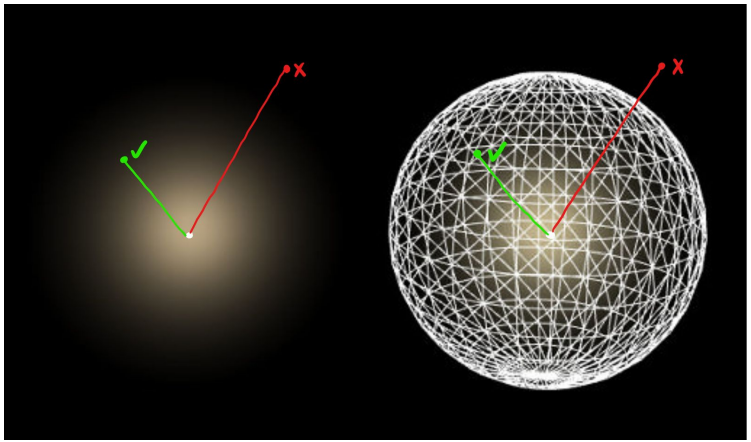
Before computing the light, check if the pixel is inside its bounding sphere

Still iterate over all the lights

```
for light in LightsList:
    if light.volume.contains(position):
        color += computeLight(light)
```

Works for:

- Forward
- Deferred



Raster Sphere

Render each light volume as a 3D sphere, use G-buffer to compute light

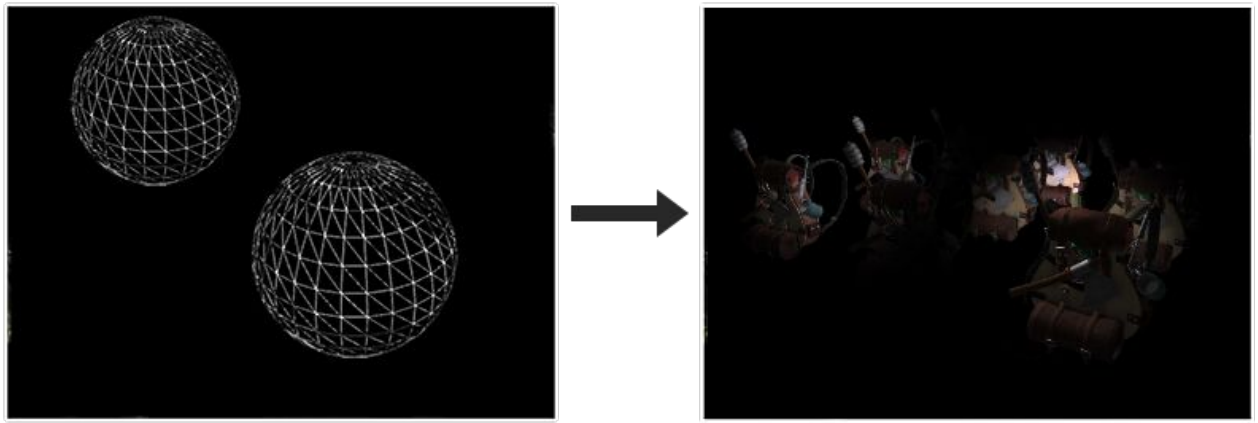
Only one light per shader evaluation

Renders more geometry (sequentially)

```
# For each sphere:
light = LightsList[current_index]
color += computeLight(light)
```

Works for:

- Deferred only



Tiled Rendering

*(Not implemented for this project)

Break the image into screen-space “tiles”

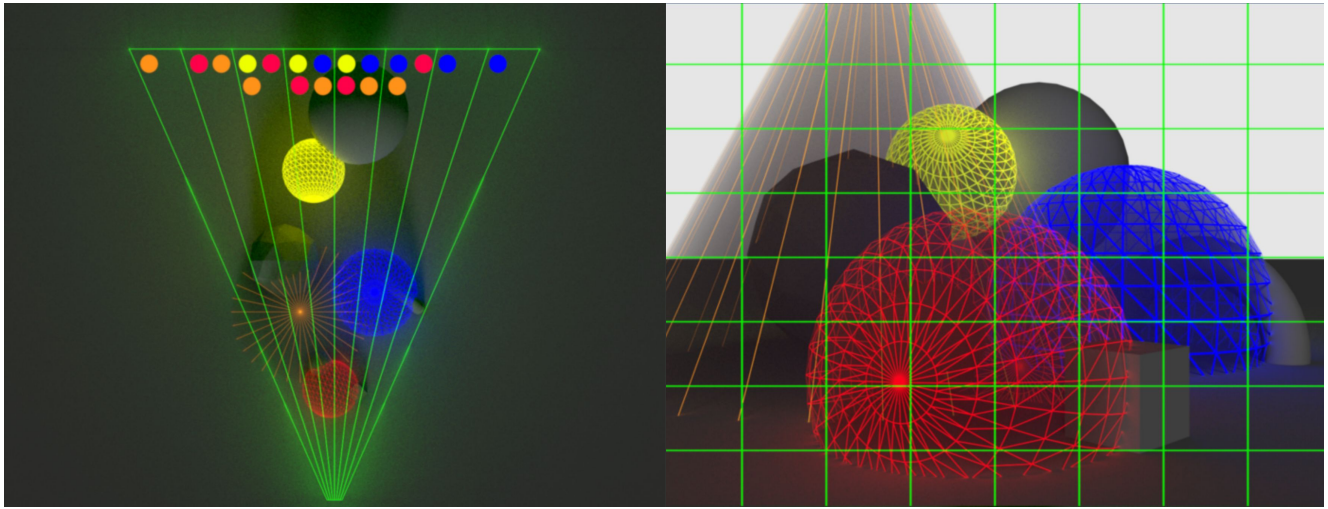
Find tile-light intersections first

Only compute lights in the tile

```
tile_index = getTileIndex(position)
for light in tiles[tile_index]:
    color += computeLight(light)
```

Works for:

- Forward
- Deferred



Clustered Rendering

Break the scene into 3D “clusters”

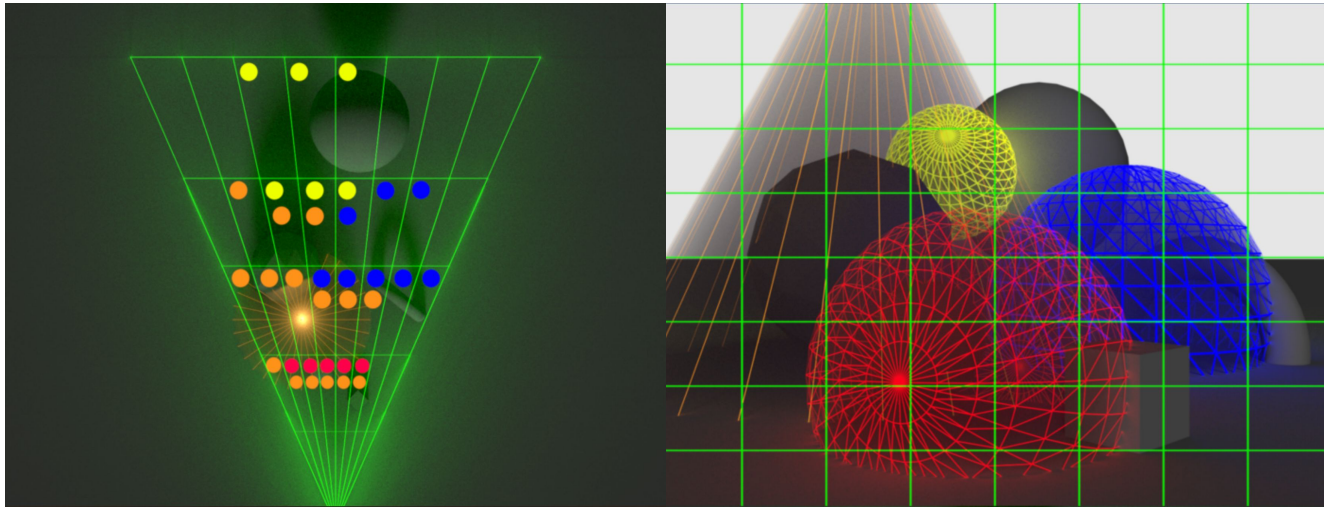
Find cluster-light intersections first

Only compute lights in the cluster

```
cluster_index = getClusterIndex(position)
for light in clusters[cluster_index]:
    color += computeLight(light)
```

Works for:

- Forward
- Deferred



Light Culling Methods

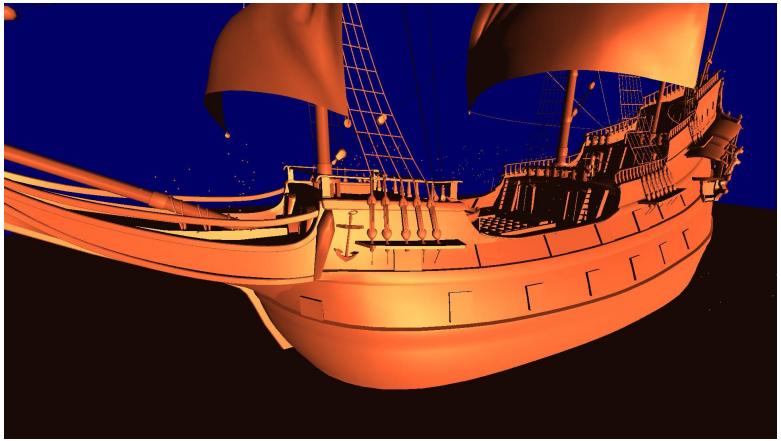
1. Background & Goals
2. Methods
3. **Results**

Implementation

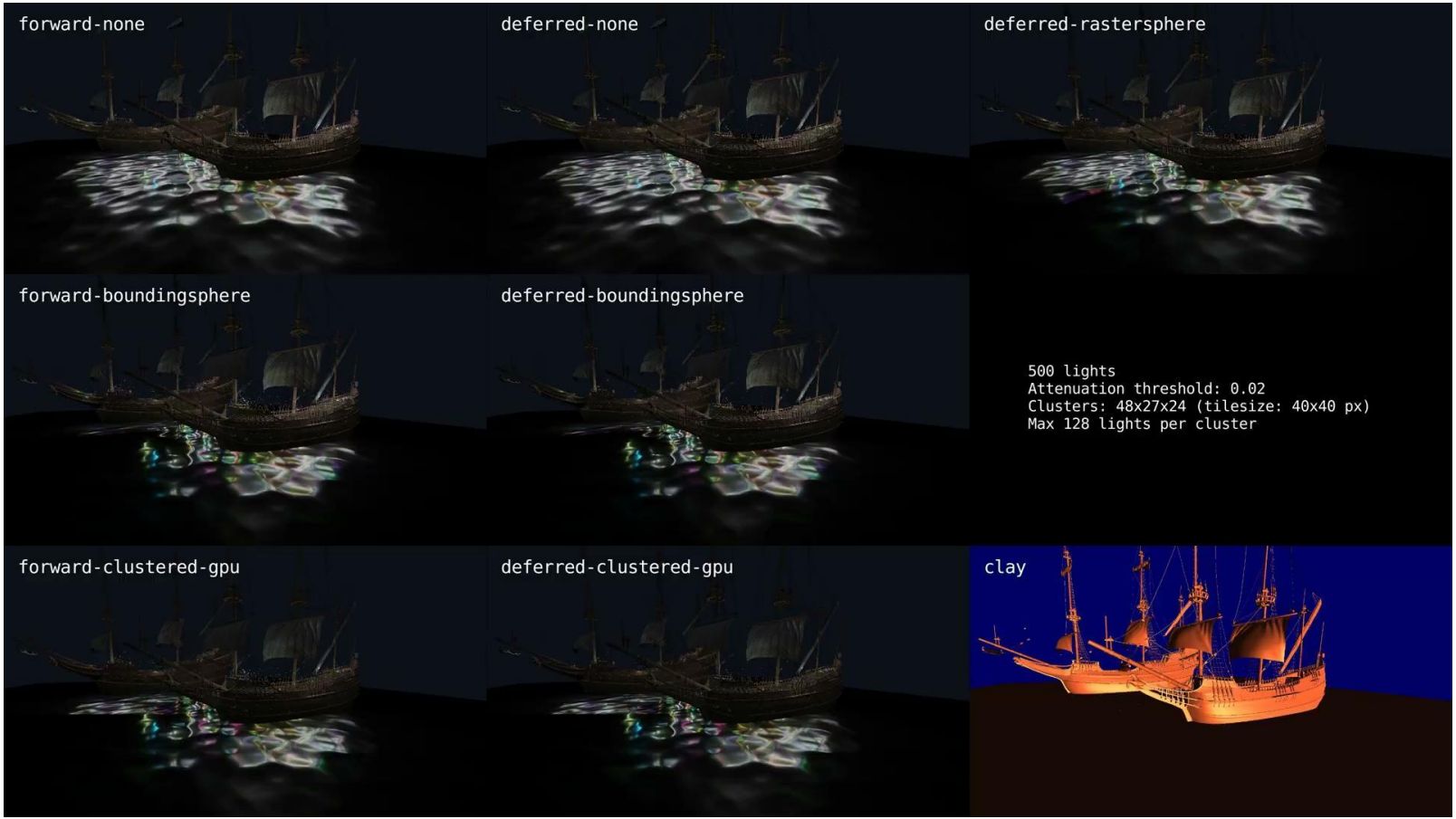
- Written in C++17 with OpenGL
- Test scene uses CC0 assets
- Variable number of light sources

Control: “Clay” renderer

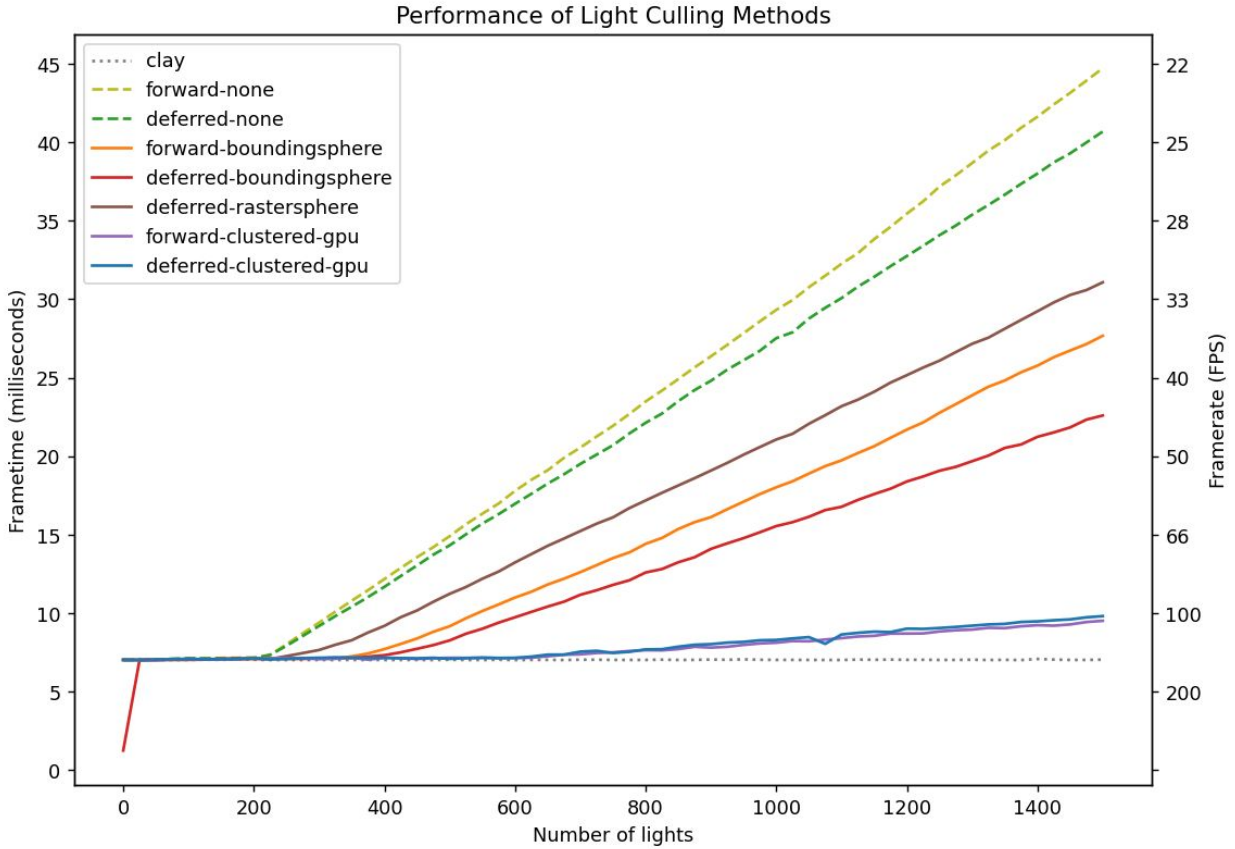
- Draws geometry *without computing any lights*



Results


forward-none
deferred-none
deferred-rastersphere
forward-boundingsphere
deferred-boundingsphere
500 lights
Attenuation threshold: 0.02
Clusters: 48x27x24 (tilesize: 40x40 px)
Max 128 lights per cluster
forward-clustered-gpu
deferred-clustered-gpu
clay

Performance

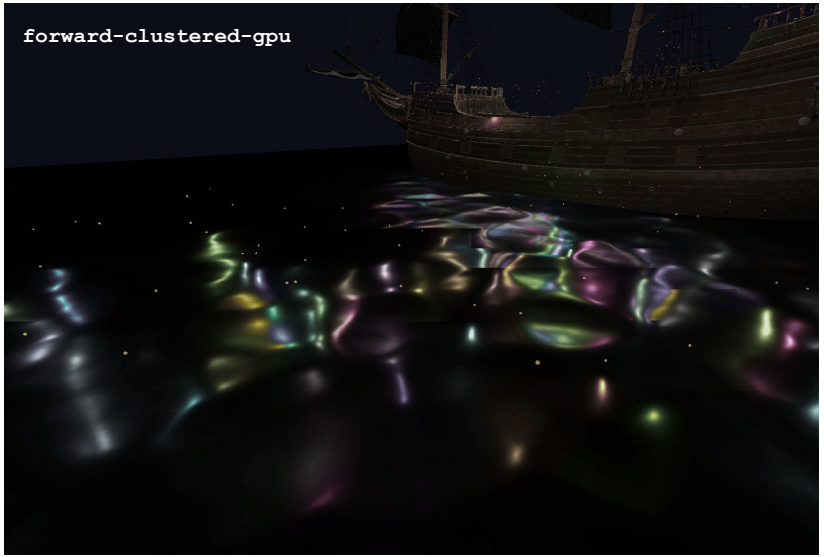


Artifacts: Reflective Surfaces

Reflective surfaces can arbitrarily extend the range of influence of lights

This can cause artifacts when reflective surfaces enter/exit a light volume

Open problem for future study



In Practice (Engine Defaults)

Unreal Engine 4/5: Deferred w/ clustered

Unity: Forward w/ clustered

Source Engine: Forward w/ clustered

Godot: Forward w/ clustered

Most mobile applications: Forward

Most VR applications: Forward

Figures & References

https://learnopengl.com/Getting-started/Hello-Triangle
https://learnopengl.com/Advanced-Lighting/Deferred-Shading
https://frederikboving.com/what-is-light-falloff-in-photography/
https://developer.unigine.com/en/docs/latest/objects/effects/volumetrics/volume_sphere
https://www.aortiz.me/2018/12/21/CG.html
https://medium.com/@lordned/unreal-engine-4-rendering-overview-part-1-c47f2da65346
https://www.gdcvault.com/play/1021771/Advanced-VR

Specs

Evaluations were performed with a Ryzen 5900X (12-core) and NVIDIA RTX 3080