



SIGGRAPH2015
Xroads of Discovery

Real-time
Man-made
Artificial



SIGGRAPH2015
Xroads of Discovery

The 42nd International Conference and Exhibition
on Computer Graphics and Interactive Techniques



CHALMERS
UNIVERSITY OF TECHNOLOGY

Part 3: Efficient Shadows from Many Lights 40 min

Ola Olsson
Chalmers University of Technology

Outline

- Problem definition
- Dynamic scenes
 - Shadow maps
- Static scene elements
 - Other methods (too)



Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

What's the problem?

- “Shadows for Many Lights”
 - Can mean many different things!
 - Each with many solutions.



Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- Shadows for Many Lights sounds like it might mean something, but
- In fact it can mean very different things, that require very different solutions.

What's the problem?

- Not our problem:
 - Very small and many lights
 - Don't need shadows.

Example:

- Photon Splatting
[Stürzlinger97, Dachsbacher06]
- Occlusion already computed

Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- For example, very small and numerous lights may not require shadow calculations at all.
- An example of this is photon splatting, where occlusion resolved in the photon tracing.

What's the problem?

- Not our problem:
 - Very small and many lights
 - Don't need visibility.
 - Large lights with much overlap
 - Can approximate visibility.

Example:

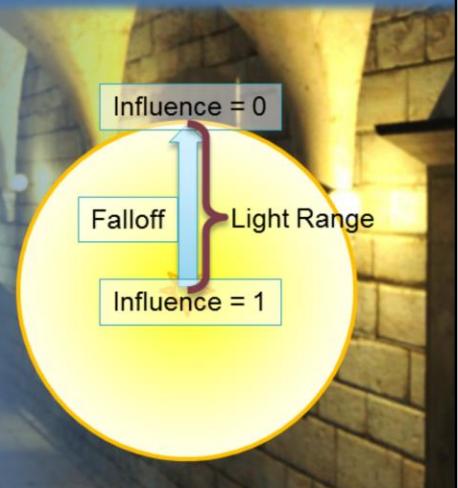
- Imperfect Shadow Maps [Ritschel08]
- ManyLoDs [Holländer11]

Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- On the other hand, with many very large lights we can use approximate visibility
- As errors average out, as it is called.
- An example of where this is done is Virtual point lights.

What's the problem?

- Our problem: In between
 - Current / Future games.
 - Dynamic lights & scenes.
 - 100s of omnidirectional lights
 - < Few 10s of lights per view sample.
 - High-quality shadows
 - No “averaging out” of errors.
 - Real time
 - At least “research real time”



Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- We aim for something like the numbers of lights and degree of overlap we might expect in modern games.
- We also target fully dynamic environments, so nothing is precomputed.
- The resulting numbers of lights per pixel implies that we must calculate high-quality shadows.
- As there will not be enough overlap to hide errors.

UKD Necropolis

- ~300-400 Lights
 - Varying sizes
- 2.6M triangles
- Min \approx 30 FPS
- Geforce Titan

- To illustrate what we are aiming for, here are some results from our paper
- This scene contains almost 400 shadow casting, omnidirectional, point lights,
- Some very large, most smaller.
- All lights and geometry is treated as dynamic.
- Even with almost 20 lights per pixel on average, we achieve...
- ...a minimum of around 30 fps on a Titan GPU.

Shadow Quality



Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- Here is another screen shot from a scene in our paper with an average of 13 lights per pixel.
- Despite this fairly high number each shadow is quite distinct, and so we need a non-approximate solution (which isn't to say a non-aliased).

FULLY DYNAMIC SCENES

Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

Why Shadow Maps? (Oh why?)

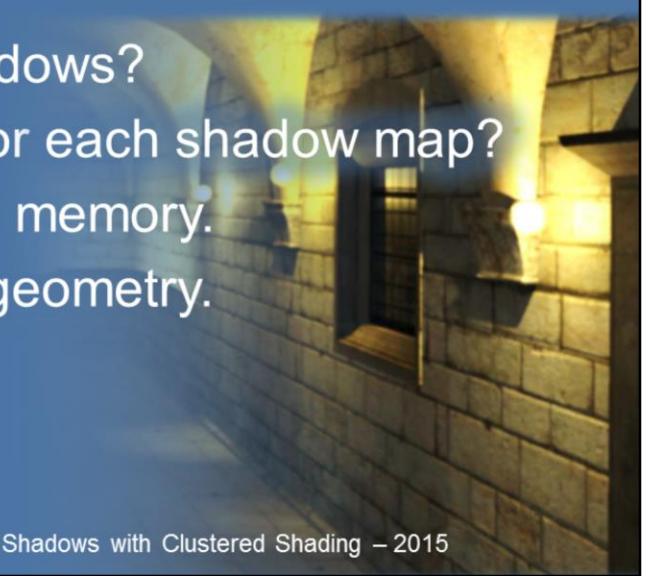
- De-facto standard
 - Despite everything...
- Shadows have high coherency.
- Increased flexibility in HW/APIs.
- Scaling no worse than other techniques.
- Cube shadow maps available
- Ray tracing / Shadow volumes / Etc?
 - Not dynamic.
 - Too slow.
 - Unfilterable.

Real-Time Many-Light Management and Shadows with Clustered Shading – 2015



- Aiming for real time pretty much necessitates the use of shadow maps for the bulk of the work.
- We will consider mostly omnidirectional lights and thus cube shadow maps.
- Other light types are typically simpler and can be viewed as a special case.
- As other solutions fail to deliver this level of performance.

Solution breakdown

- 
1. Which lights cast shadows?
 2. Required resolution for each shadow map?
 3. Manage shadow map memory.
 4. Cull shadow-casting geometry.
 5. Draw shadow maps.
 6. Shade scene.

Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- These are the steps we need to perform each frame, using the current camera view, to calculate a shaded scene.

Shadows for many lights

- Paper from I3D 2014 + TVCG 2015 [Olsson15]
 - “(More) Efficient Virtual Shadow Maps for Many Lights”.
- Builds on Clustered Shading
- Adds support for shadows
 - Hundreds in, real-time
 - Well, research-real-time at least.
 - High & Consistent quality
 - Efficient culling
 - Virtual/Sparse texture use case



Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- We will now talk about the techniques presented in my paper at I3D in 2014, with an extended version in TVCG - out now -
- ...titled More Efficient Virtual Shadow Maps for Many Lights.
- This paper in turn builds on clustered shading,
- and adds support for shadows from hundreds of lights in real time.
- Since this paper provides efficient solutions to all the steps, it is a good starting point for this talk.

Solution breakdown

- Which lights cast shadows?
- Required resolution for each shadow map?
- Manage shadow map memory.
- Cull shadow-casting geometry.
- Draw shadow maps.
- Shade scene.

Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- The first step is the same as determining what lights are needed for shading
- And we have already seen how this can be achieved using clustered shading and other methods.

Clustered Shading Key Features

- Clusters

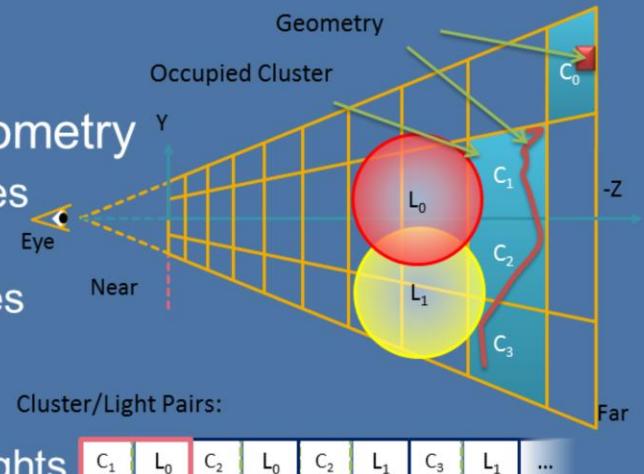
- Proxy for visible geometry

- Bounds view samples
 - 3D boxes

- Orders of magnitudes fewer

- Cluster/Light Pairs

- Links clusters and lights

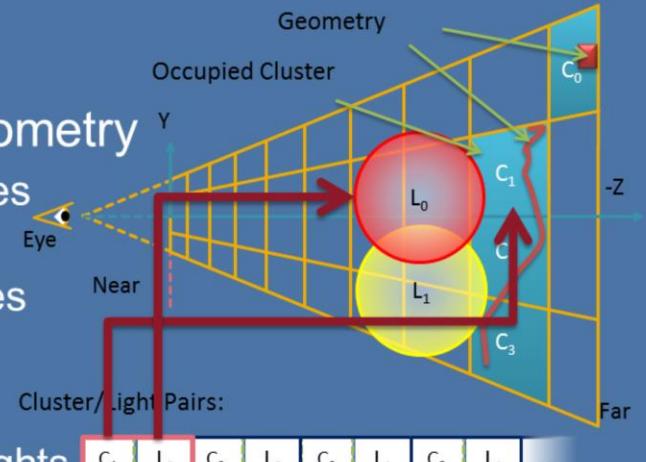


Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- Note that the clustered shading method we build upon here is the one described in the paper, not Emils eminently practical method.
- The reason for this is that we need some information about the shadow receiving geometry to be able to get good shadow performance.
- Therefore, we cannot just use the up-front full grid approach.
- Two things about clusters are worth repeating in this context:
- First, the visible geometry is approximated reasonably well by the clusters, but there are very few in comparison. Usually around a thousand times more pixels than clusters, though this is of course tuneable with cluster size.
- Secondly, clustered shading provides an association between clusters and lights. For shading it allows us to know which lights affect each cluster, but also the reverse mapping, which we will make heavy use of to compute shadows.

Clustered Shading Key Features

- Clusters
 - Proxy for visible geometry
 - Bounds view samples
 - 3D boxes
 - Orders of magnitudes fewer
 - Cluster/Light Pairs
 - Links clusters and lights



Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- The key data is the cluster/light pairs that enable parallel operations on these pairs.
- We use this for a lot of the processing later on.
- In the illustration, the pair of integers link the L_0 light and the C_1 cluster, the cluster is shown here as containing geometry, and overlapping a light, and it is these that we will process.

Solution breakdown

- **Which lights cast shadows?**
- Required resolution for each shadow map?
- Manage shadow map memory.
- Cull shadow-casting geometry.
- Draw shadow maps.
- Shade scene.

Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- So, the clustered shading algorithm provides the lights affecting each cluster, and a light that does not affect any cluster need not be processed further.

Solution breakdown

- Which lights cast shadows?
- Required resolution for each shadow map?
- Manage shadow map memory.
- Cull shadow-casting geometry.
- Draw shadow maps.
- Shade scene.

Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- Next we must work out the resolution of each shadow map

Shadow Map Resolution

- Pick constant resolution
 - Under and Oversampling
 - Poor quality

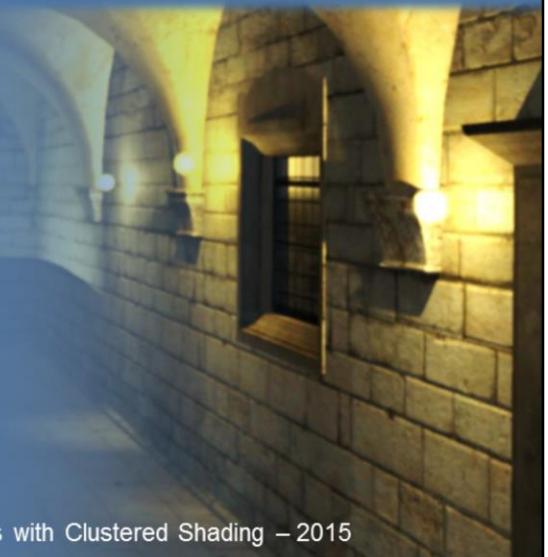


Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- We could of course just pick a constant resolution...
- but then we'd not be talking about high quality shadows any more, with over and undersampling both being the norm.
- Using oversampled shadow maps is not just wasteful in terms of memory, it is also slow and may yield aliasing.

Shadow Map Resolution

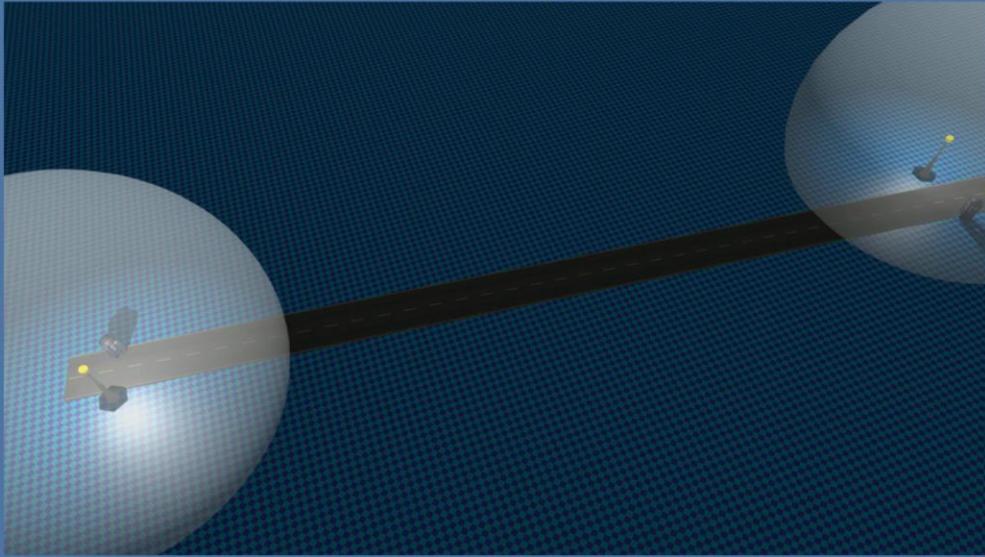
- Pick constant resolution
 - Under and Oversampling
 - Poor quality
- Match sample density
 - Screen space
 - Shadow-map space



Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

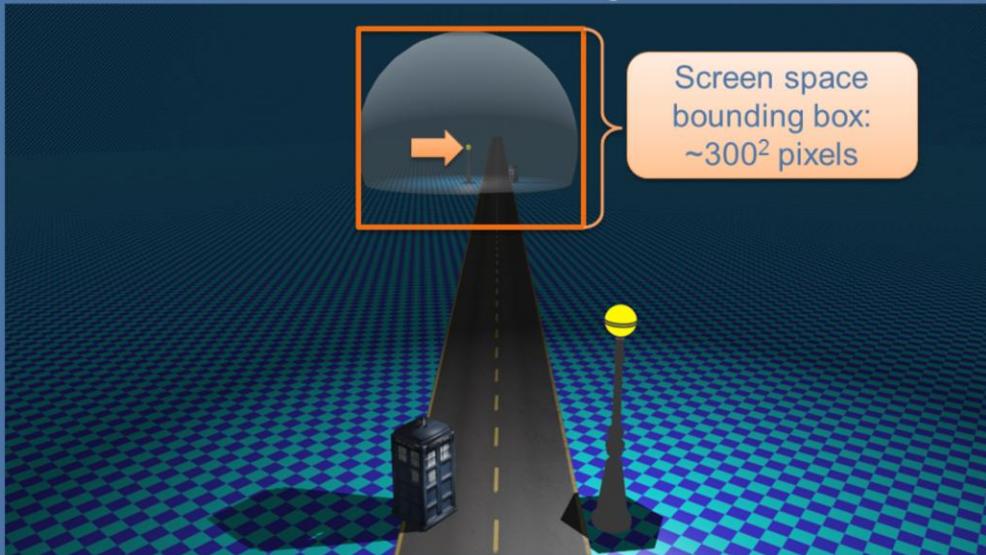
- So what is a good resolution for a shadow map?
- Well, the easiest way to think about it is that to avoid sampling artifacts (as far as possible)
- Is to have one sample in the shadow map for each sample on screen...

2. Shadow map resolution



Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

Shadow map res



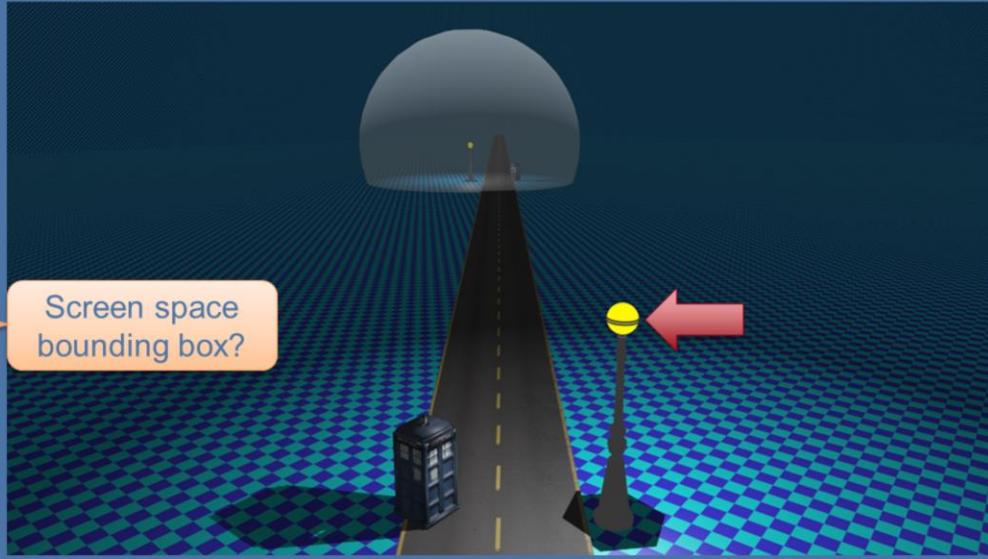
Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- One common approach (e.g., [King04]) is to simply estimate the required resolution of the shadow map from the projection of the light bounding sphere.
- This works well enough when the light is far away, and thus small on-screen.
<click>

Shadow map res



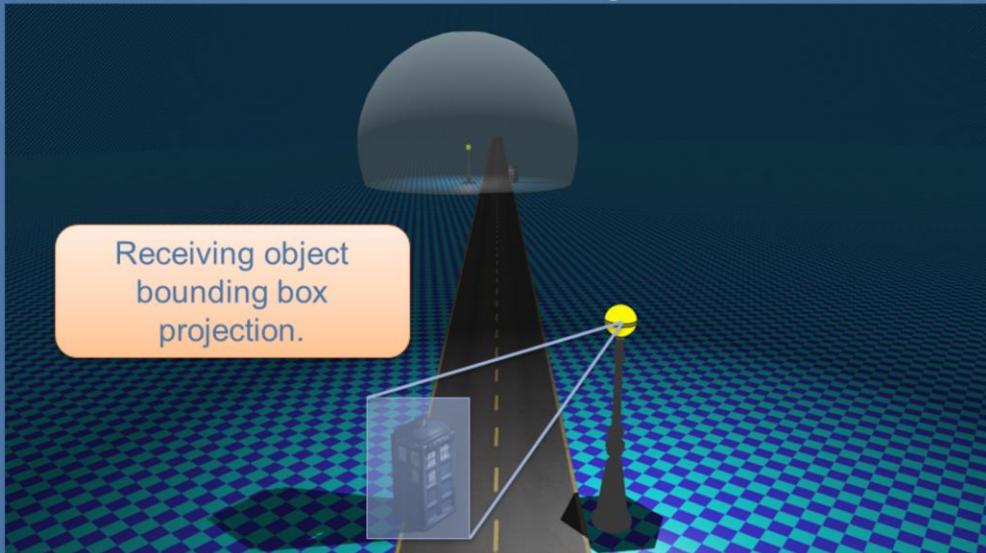
Screen space
bounding box?



Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- But produces fairly useless results when the camera is near or within the light sphere.
- Other approaches have been proposed that use shadow receiver locations to work out better estimates [Forsyth04,Lefohn07]
- From clustered shading, we have a nice and coarse representation of the shadow receivers...

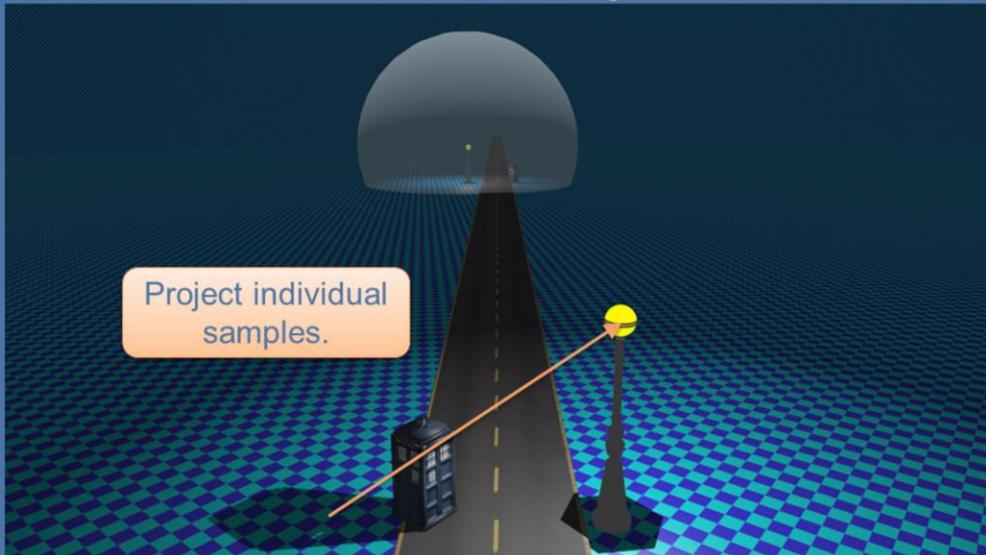
Shadow map res



Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

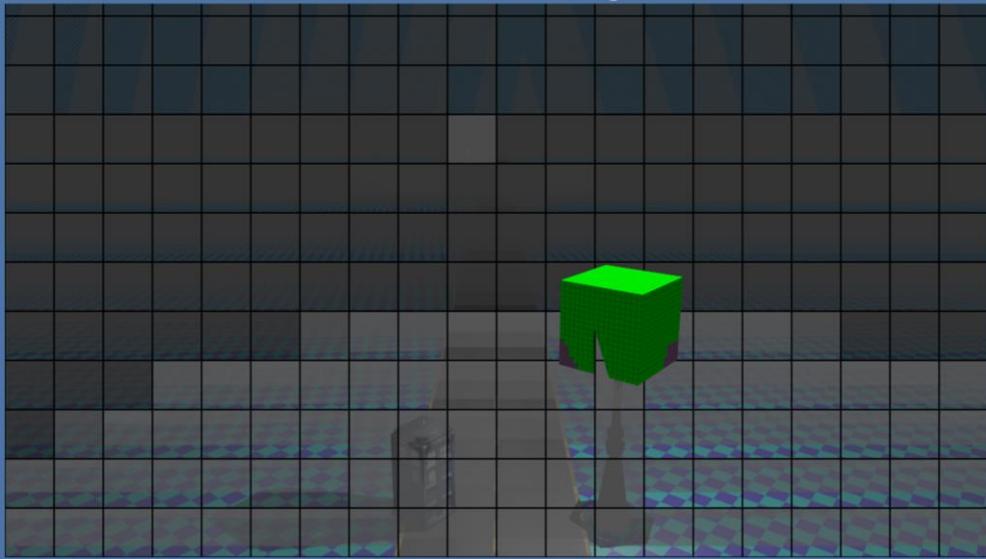
- Other approaches have been proposed that use shadow receiver locations to work out better estimates
- For example projecting shadow receiver bounding boxes onto the light. [Forsyth04]

Shadow map res



- Or individual samples [Lefohn07]
- This produces good results, but is expensive as there are many samples.
- We therefore turn to...

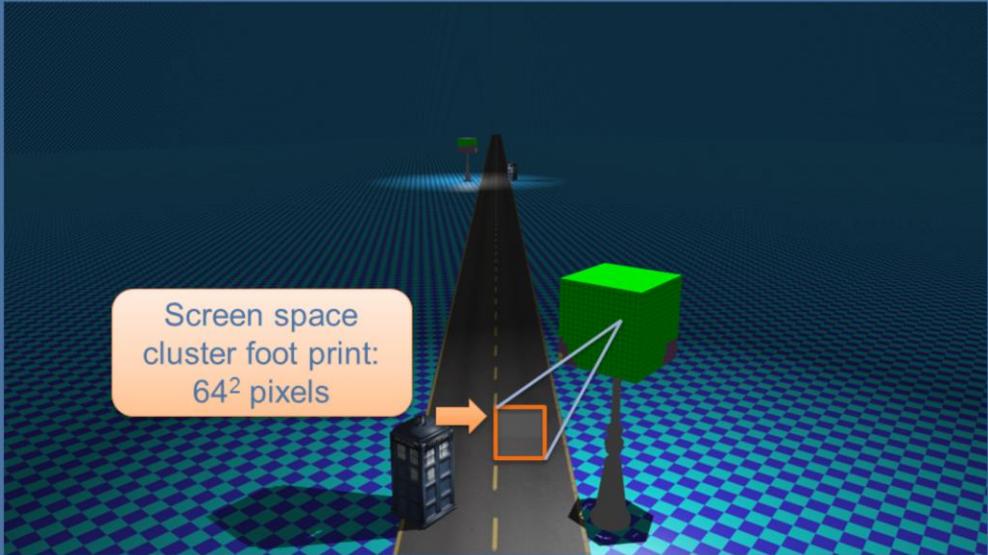
Shadow map res



Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- You said it: the clusters

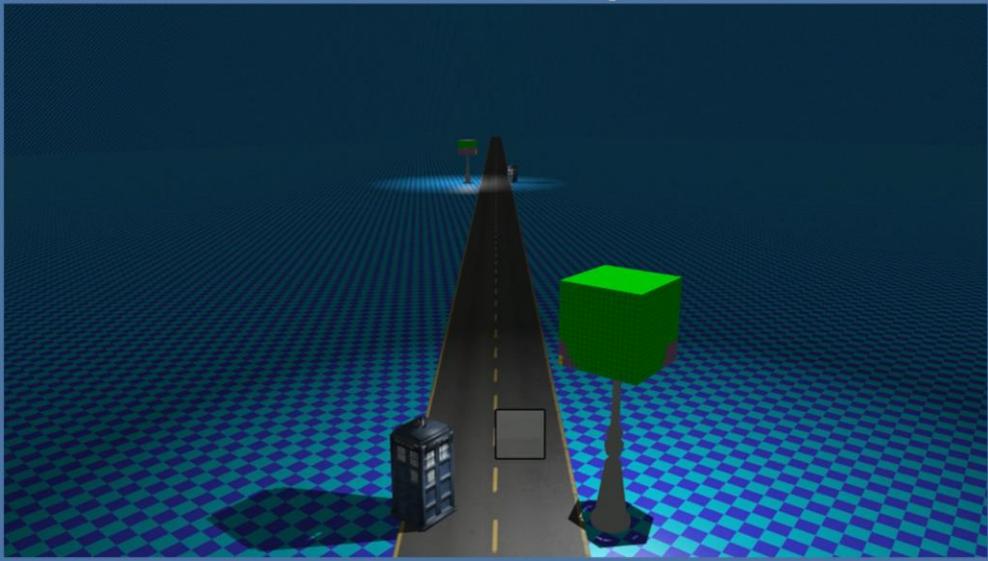
Shadow map res



Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- Each cluster has a fixed screen space footprint <click>

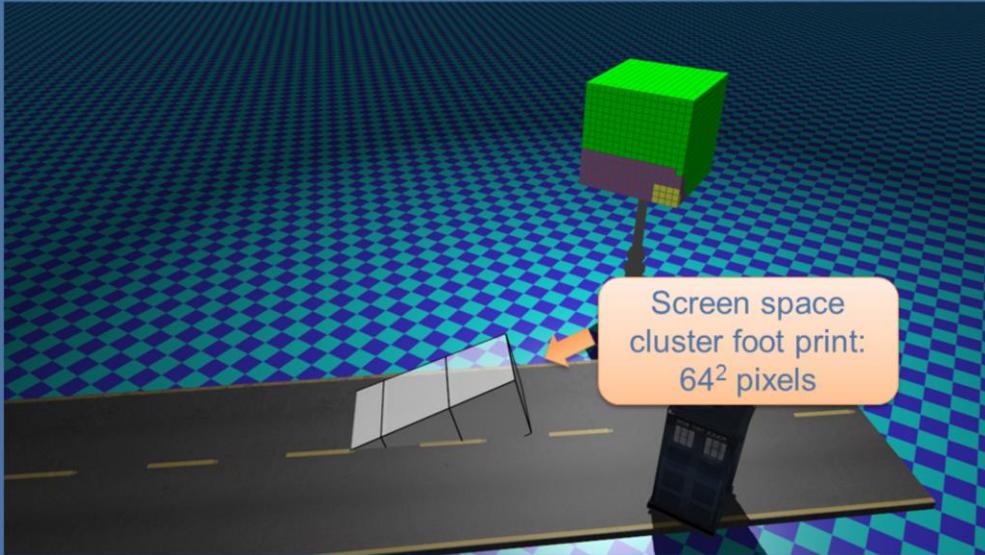
Shadow map res



Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- Seen from the side...

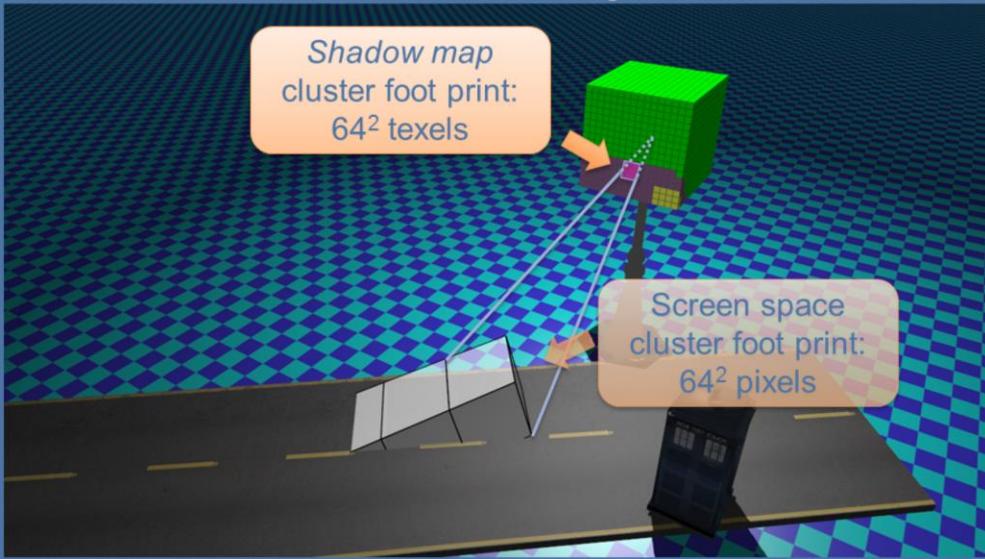
Shadow map res



Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- We see that the cluster, which by design is cubish in shape

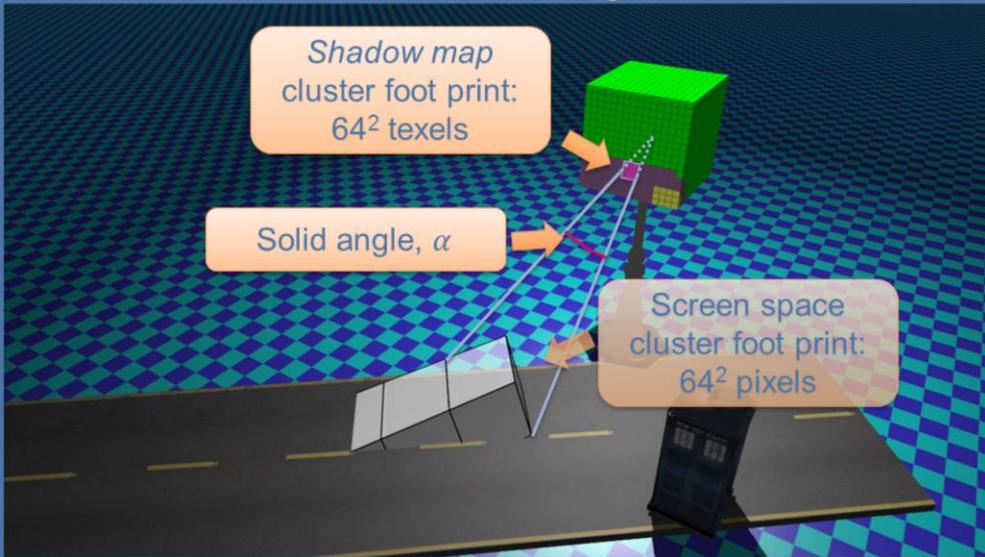
Shadow map res



Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- Has a footprint on the shadow map

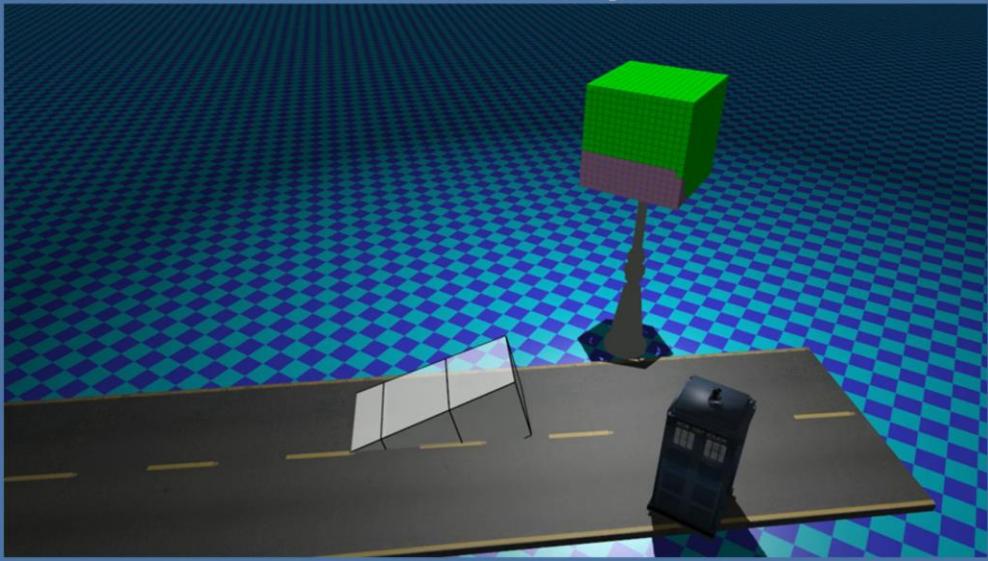
Shadow map res



Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- ...which allows us to compute the required resolution as proportional to the solid angle
- As usual the devil is in the paper, I mean the details!

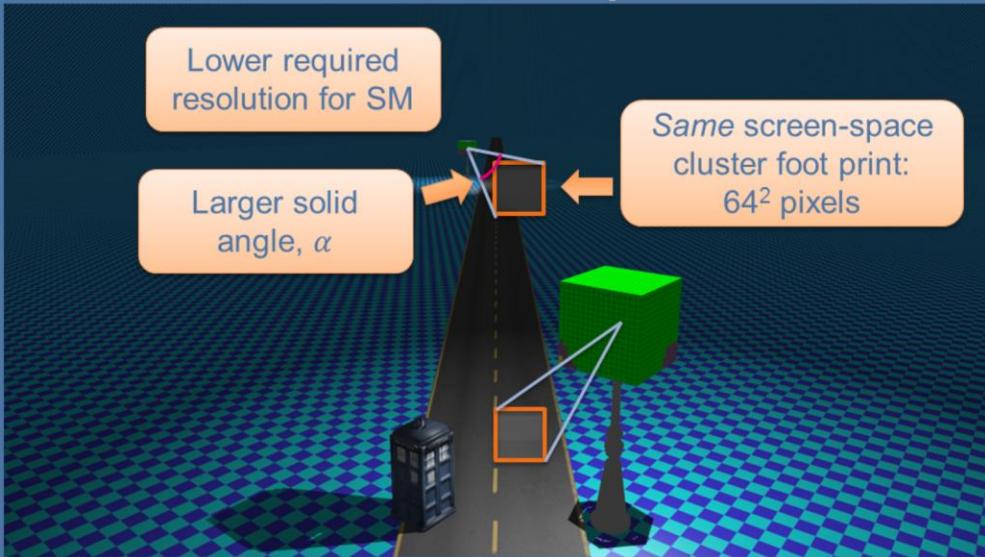
Shadow map res



Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- Going back...

Shadow map res

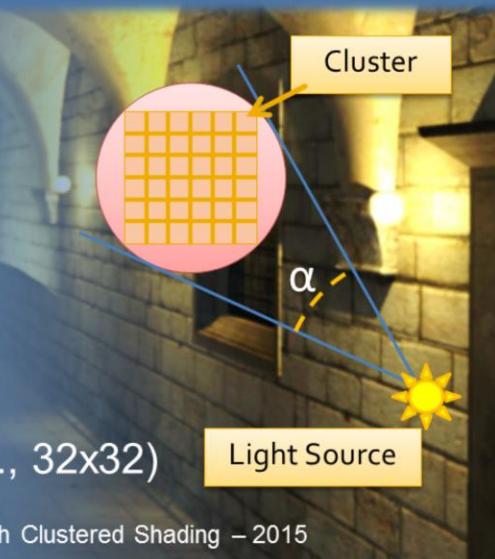


Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- We see that for a more distant shadow map, (or clusters closer to the SM)
- The clusters subtend a larger solid angle, and thus the shadow map resolution becomes smaller.

Res calc.

- Solid angle over light sphere
- All cluster/light pairs in parallel
 - Keep highest for each light
 - atomicMax
- $Res = \sqrt{\frac{S/(\alpha/4\pi)}{6}}$
 - α = solid angle
 - S = Screen space foot print (e.g., 32x32)



Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

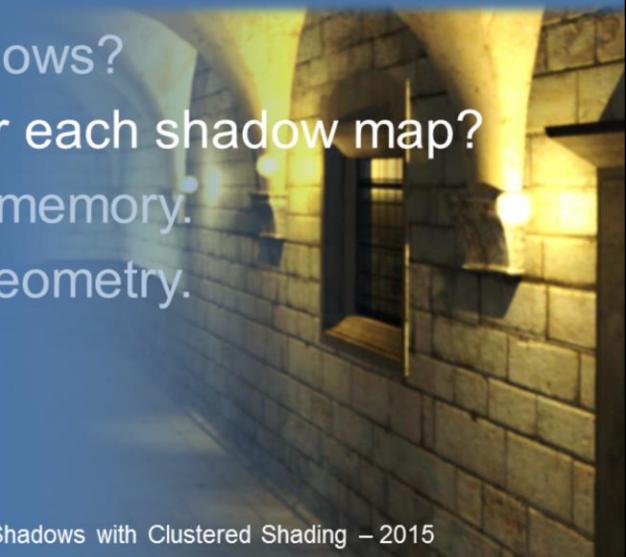
- We perform this calculation in parallel for all cluster light pairs.
- And reduce the final result for each light using atomic operations.

Solution breakdown

- Which lights cast shadows?
- Required resolution for each shadow map?
- Manage shadow map memory.
- Cull shadow-casting geometry.
- Draw shadow maps.
- Shade scene.

Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- This results in a list with a resolution for each light, or shadow map.
- Some will have zero, if they are not referenced by any cluster/light pairs, and need not be considered further.



Solution breakdown

- Which lights cast shadows?
- Required resolution for each shadow map?
- Manage shadow map memory.
- Cull shadow-casting geometry.
- Draw shadow maps.
- Shade scene.

Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- Now, we'll need to provide some shadow maps to match said resolutions.

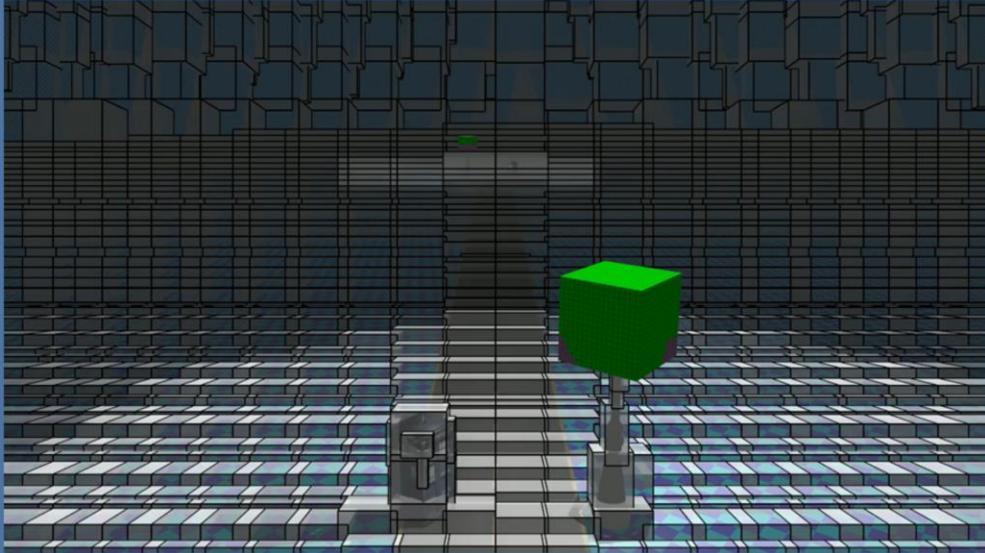
Shadow Map Storage

- Shadow maps required up front.
 - Efficient storage paramount.
 - Ideally: Store *no* unused samples!
 - Somewhat tricky in practice.
 - (Shadow Volumes , Ray tracing, Alias-Free Shadow Maps)

Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- Recall that clustered shading, like other modern real-time shading algorithms,
- have the inner loop, in the fragment shader, iterates over a list of lights.
- This means all shadow maps must exist up-front before the shading pass.
- Consequently, efficiently managing shadow map storage has become a very important problem.
- And this does not just mean parceling out shadow maps as needed,...
- but ideally we should be able to store only those samples that matter.
- Several shadow algorithms do just that, but fail to achieve consistent real-time performance,
- For different reasons.

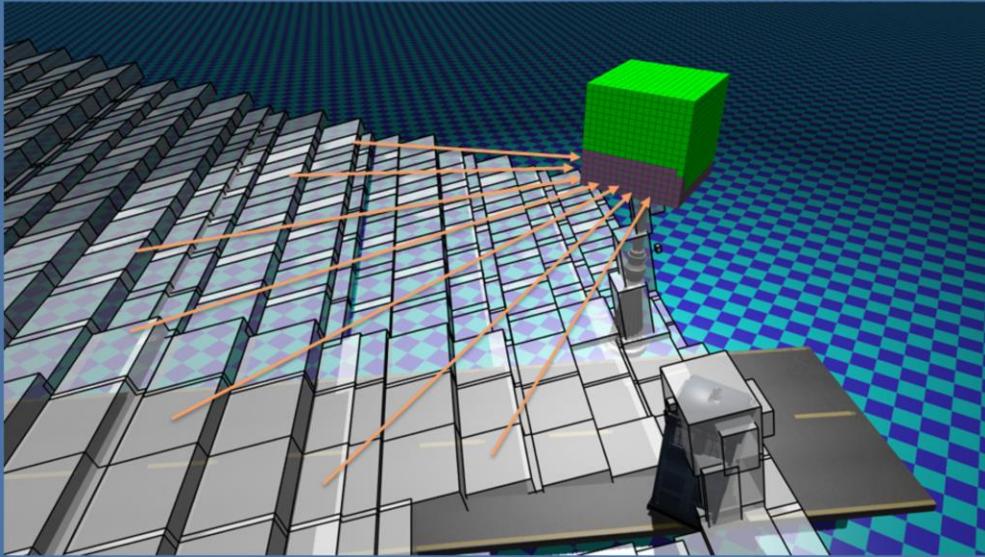
Shadow map storage



Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- Lets have a look again at the clusters...

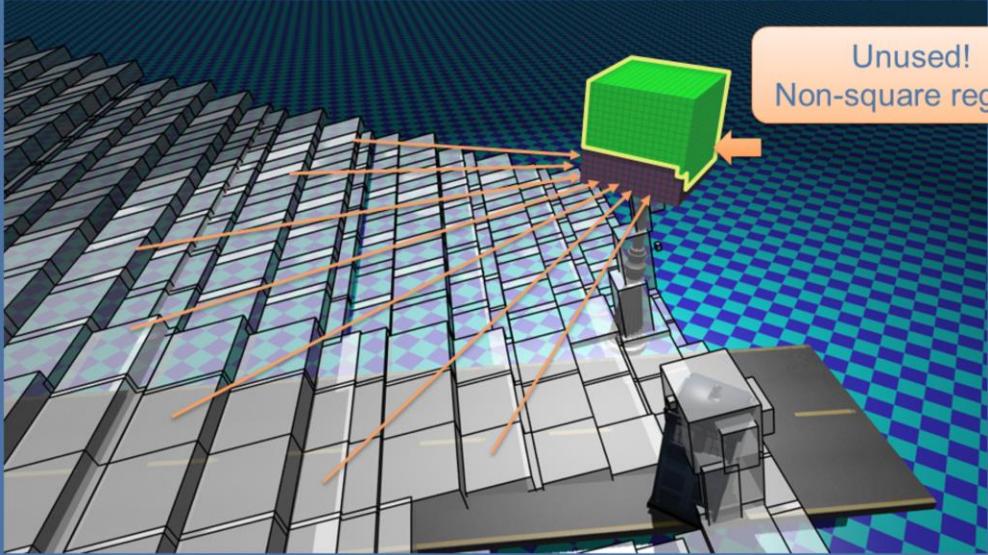
Shadow map storage



Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- showing their projection on the shadow map.

Shadow map storage



Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- Note how a large, and irregular area is unused.
- Anything drawn into this part of the shadow map will not produce any visible shadow.
- Clearly we should avoid allocating memory for this region, but how?

Hardware Virtual Textures*

- Allocate Virtual Texture Storage
 - Just like virtual memory.
 - Low overhead.
- Physical backing in Tiles (Pages)
 - E.g., 256 x 256 Texels.
- Recent APIs.
 - OpenGL 4.4 extension [OpenGL14]
 - ARB_sparse_texture
 - DirectX 11.2
 - Tiled Resources [Coombes14]
 - Vulkan? DX12? Consoles? (GCN)

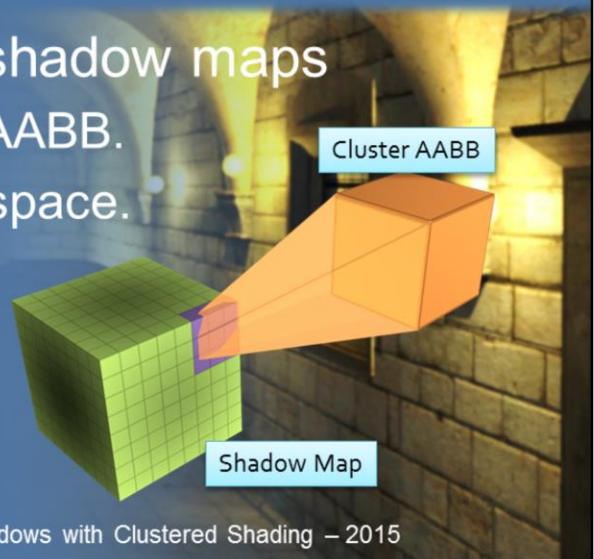


* Or: Sparse Textures, Partially Resident Textures
Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- Our answer is hardware supported virtual, or sparse, textures...
- These have become available in mainstream graphics APIs recently,
- and makes it possible to allocate large virtual textures,
- ...but commit physical storage in tiles only where needed.
- This fits our ideal pretty well, especially given that shadow map samples tend to clump together because of screen-space coherency [Lefohn07].

Determining Used Pages

- Project Clusters onto shadow maps
 - Cluster's world space AABB.
 - Shadow map in world space.
 - Simple calculation.



Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- This requires projecting bounding boxes onto the cube maps, which is not trivial
- By transforming the cluster AABB to world space,
- and also keeping cube shadow maps in world space.
- Calculating the projection can become quite simple and fast.

Cube Face +X

```

float rdMin = 1.0f / max(Epsilon, aabb.min.x);
float rdMax = 1.0f / max(Epsilon, aabb.max.x);

float sMin = min(-aabb.max.z * rdMin, -aabb.max.z * rdMax);
float sMax = max(-aabb.min.z * rdMin, -aabb.min.z * rdMax);

float tMin = min(-aabb.max.y * rdMin, -aabb.max.y * rdMax);
float tMax = max(-aabb.min.y * rdMin, -aabb.min.y * rdMax);

```



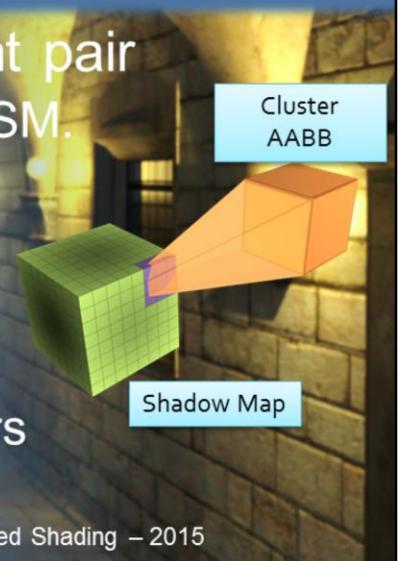
- Bounding rectangle on cube face.
- Repeat 6 times, slightly different.

Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- This code is what we use to compute the projection on one cube face.
- And as you can see, it is not extremely complex.
- We repeat this, slightly differently for each face and get a rectangle in texture coordinates for each.
- This is then converted to a bit mask, with a single bit for each tile or page in the virtual cube map.
- This very quick bit-grid rasterization is something we build most of the efficiency improvements in our algorithm on.

Process

- In parallel, for each cluster/light pair
 - Project cluster AABB onto light SM.
 - Update bit masks for each light.
 - One bit/physical page.
 - $32 \times 32 \times 6$ bits = 768 byte / light.
 - Atomic reduction atomicOr
 - <0.25ms / 180k cluster/light pairs

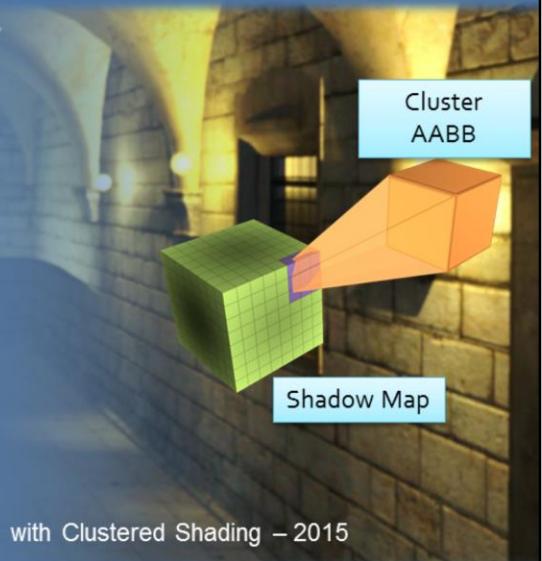


Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- We compute this mask, by running a cuda thread for each cluster/light pair.
- In the kernel we calculate the projection just shown
- As the mask is stored for each light and thus referenced by many cluster light pairs,
- It is updated using atomicOr.
- This is a pretty quick pass, <0.25ms, 180k light/cluster pairs.

Process

- In parallel, for each cluster/light pair
 - Project cluster AABB onto light SM.
 - Update bit masks for each light.
 - One bit/physical page.
 - $32 \times 32 \times 6$ bits = 768 byte / light.
 - Atomic reduction atomicOr
 - <0.25ms / 180k cluster/light pairs
- Copy masks back to host.
- Commit physical pages (host)
 - glTexPageCommitmentARB



Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- The masks are copied back to the host, and used to commit pages for the shadow map textures.
- And here we really could use an indirect API to avoid nasty stalls!

Virtual SMs – Results



Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- So we might ask, is it worthwhile doing virtual shadow maps?
- Well, for the necropolis scene, there is a factor 26 improvement, compared to physical shadow maps of the same quality
- And <click>

Virtual SMs – Results



Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- ...in other terms, this means the difference between impossible on a current console
- And something we might consider.

Quality vs. Memory Usage

- Quite uniform quality.
- Simple to reduce
 - Undersampling parameter.
- Control memory use.
- Could be done dynamically.
 - Guarantee peak memory use?

Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- Our method achieves quite uniform shadow quality,
- This means we can control quality and thus memory usage with a global parameter.
- This allows more flexibility in memory use while maintaining uniform quality.
- Rather than say, some lights getting very poor shadows and others good.
- One idea is to do this dynamically, to ensure a certain memory budget.
- Should be possible as it is very quick to work out memory usage from the used pages and resolutions.

Quality vs. Memory Usage

2x Undersampling ~140Mb 4x Undersampling ~80Mb



Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- Here is a shot showing this, with 2x or 4x global reduction in shadow map resolution.
- Shown are the corresponding peaks in shadow map memory usage.
- Recall that the peak is 322MB without reducing quality.

Quality vs. Memory Usage

2x Undersampling ~140Mb 4x Undersampling ~80Mb



Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- Zooming in, we see that the shadow aliasing is about the expected level when compared to edge aliasing in the image.
- This indicates that our simple scheme for calculating the required resolution yields reasonable results.
- With filtering, this can be acceptable, especially as it allows *uniform* quality rather than unevenly allocating shadow maps.
- Memory usage is, reduced significantly.

Solution breakdown

- Which lights cast shadows?
- Required resolution for each shadow map?
- Manage shadow map memory.
- Cull shadow-casting geometry.
- Draw shadow maps.
- Shade scene.

Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- So that takes care of memory management, and we can now allocate shadow maps to draw into.

Solution breakdown

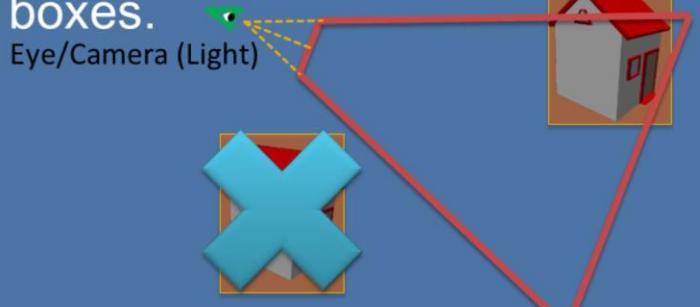
- Which lights cast shadows?
- Required resolution for each shadow map?
- Manage shadow map memory.
- Cull shadow-casting geometry.
- Draw shadow maps.
- Shade scene.

Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- Next must rasterize triangles into the shadow maps.
- To do this efficiently requires culling, as with any rasterization, only more so.

Basic Culling

- Just like View Frustum Culling
 - Cull chunks of geometry.
 - Using bounding boxes.

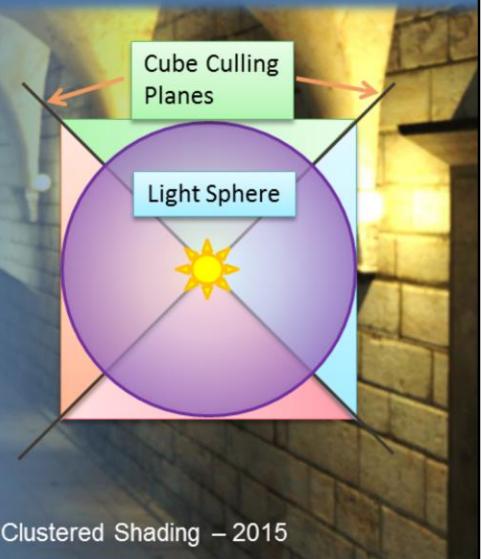


Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- This process is just like good old view frustum culling
- In that we are trying to get rid of geometry that is not visible, or within the view-volume.
- And that we do this by testing bounding volumes representing batches of triangles.

Basic Culling

- Not like VFC
 - Hundreds of lights.
 - Mostly short view volumes.
 - Limited light range.
 - Omni-directional lights.
 - 6 *adjacent* frustra.
 - Sharing planes.



Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- What is not like view frustum culling is that we need to perform hundreds of these tests.
- The view volumes are quite small, or short, given the limited range of the lights
- There are 6 *adjacent* frustra sharing planes.
- And this adjacency is an opportunity to share calculations.

Basic Culling

```

int getCubeFaceMask(float3 cubeMapPos, Aabb aabb)
{
    float3 planeNormals[] = { {-1,1,0}, {1,1,0}, {1,0,1}, {1,0,-1}, {0,1,1}, {0,-1,1} };
    float3 absPlaneNormals[] = { {1,1,0}, {1,1,0}, {1,0,1}, {1,0,-1}, {0,1,1}, {0,1,1} };

    float3 center = aabb.getCentre() - cubeMapPos;
    float3 extents = aabb.getHalfSize();

    bool rp[6];
    bool rn[6];

    for (int i = 0; i < 6; ++i)
    {
        float dist = dot3(center, planeNormals[i]);
        float radius = dot3(extents, absPlaneNormals[i]);
        rp[i] = dist > -radius;
        rn[i] = dist < radius;
    }

    int fpix = rn[0] && rp[1] && rp[2] && rp[3] && aabb.max.x > cubeMapPos.x;
    //...same for other faces...
    return fpix | (fnx << 1) | (fpy << 2) | (fnz << 3) | (fpz << 4) | (fnz << 5);
}
  
```

Literal constants, all ones or zeroes.

If unrolled, very few operations left!

Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- Here is the somewhat condensed code we use to calculate the culling mask, with a bit for each cube face.
- This is a very efficient, testing only 6 planes for six frustums. ([click](#))
- Especially as the plane equations are all ones and zeroes, ([click](#))
- Which means that if the loop is unrolled, most of this code just goes away!
- I think this is a rather big advantage with cube maps, over using separate frustums (As done in [Forsyth04]).
- This efficiency is especially important given that we will be culling a lot more objects than normal culling!
- How so, I hear you say?
- Glad you asked!

Basic Culling

- Fine Granularity (Batches)
 - Must be small.
 - Match size of light.



Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- You see, to enable effective culling, as in getting rid of a large proportion of the triangles, batches must be small,
- Intuitively, for any culling operation, the optimal size of batches correlates to the size of the frustums.

Basic Culling

- Fine Granularity (Batches)
 - Must be small.
 - Match size of light.

1000 Tris x 3 Cube
faces
= 3k



Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- If batches are too large, the triangles get replicated into most cube faces replicated.

Basic Culling

- Fine Granularity (Batches)
 - Must be small.
 - Match size of light.

1000 Tris x 3 Cube
faces
= 3k



Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- Smaller batches, enables

Basic Culling

- Fine Granularity (Batches)
 - Must be small.
 - Match size of light.

1000 Tris x 3 Cube
faces
= 3k

$(250 \text{ Tris} \times 2) \times 2 = 1\text{k}$



Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- More precise culling, and thus fewer triangles drawn.
- So we are trading increased culling work for fewer triangles drawn.
- Triangle drawing is the biggest performance bottleneck so this is important to be able to tune.

Batches

- Batch ~ 100 triangles
 - AABB.
 - Index to transform, triangles.
- Constructed offline
 - Group Coherent Triangles.
 - Same transform(s).
 - Flat array.

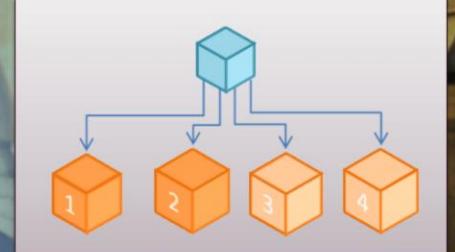
```
struct Batch
{
    Aabb aabb;
    int tris;
    int offset;
    int transformInd;
};
```

Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- We used batches around 100 triangles,
- A batch is represented by an AABB and a list of triangles and they are constructed in a pre-process.
- The batches are stored in a flat array that is loaded into the runtime.

Batch BVH

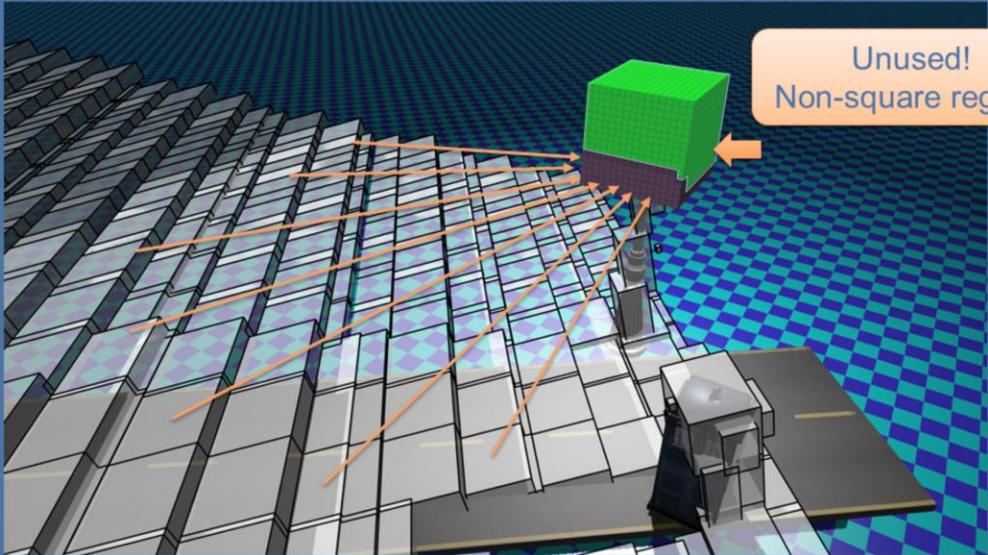
- Constructed online
 - Each frame.
 - Batch AABBs updated from vertices.
 - Simple 32-way full BVH.
- Traverse in parallel for each light



Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- At run-time the batch bounds are updated and a simple BVH is constructed on the GPU
- This is then traversed for each light to build a list of batches for each shadow map.

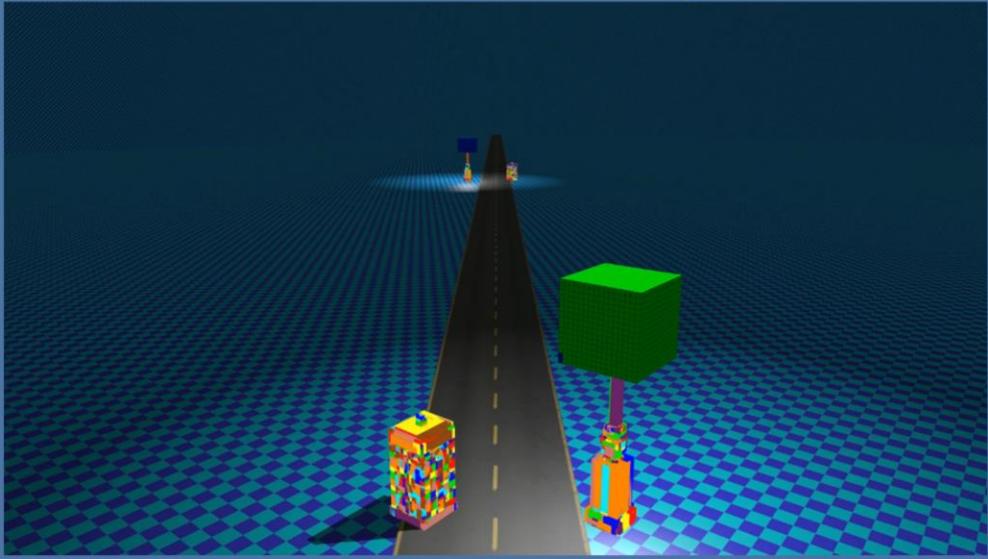
Better Culling



Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

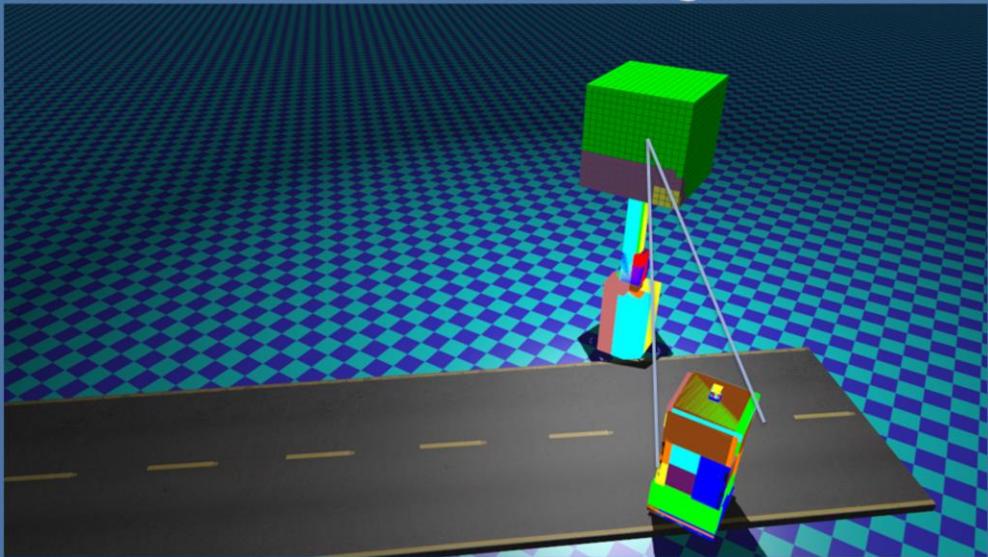
- Recall the projection of the clusters on the shadow map.
- And how a large, and irregular area is unused.
- Apart from storing this region, it is also a bit silly to draw anything into it!

Better Culling



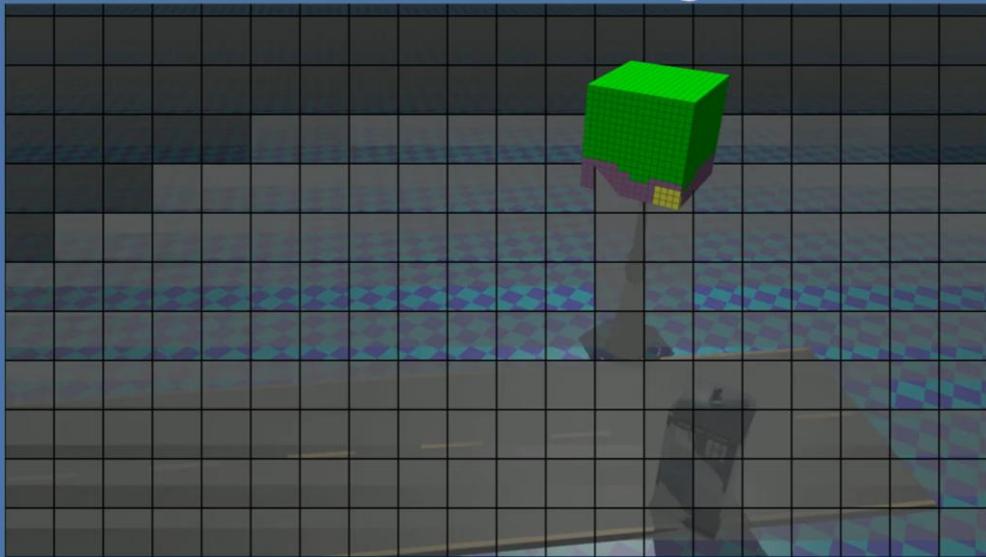
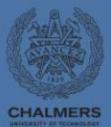
Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

Better Culling



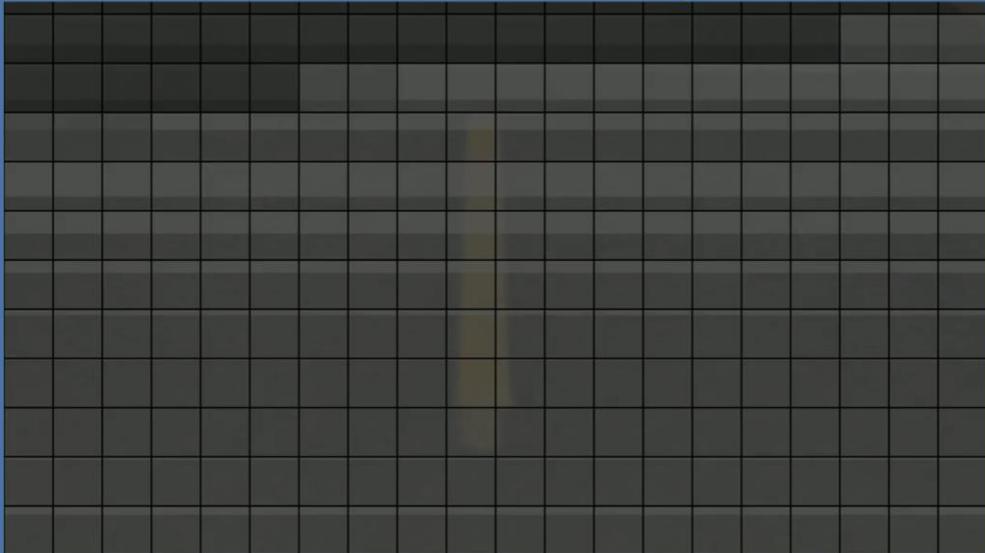
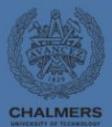
Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

Better Culling



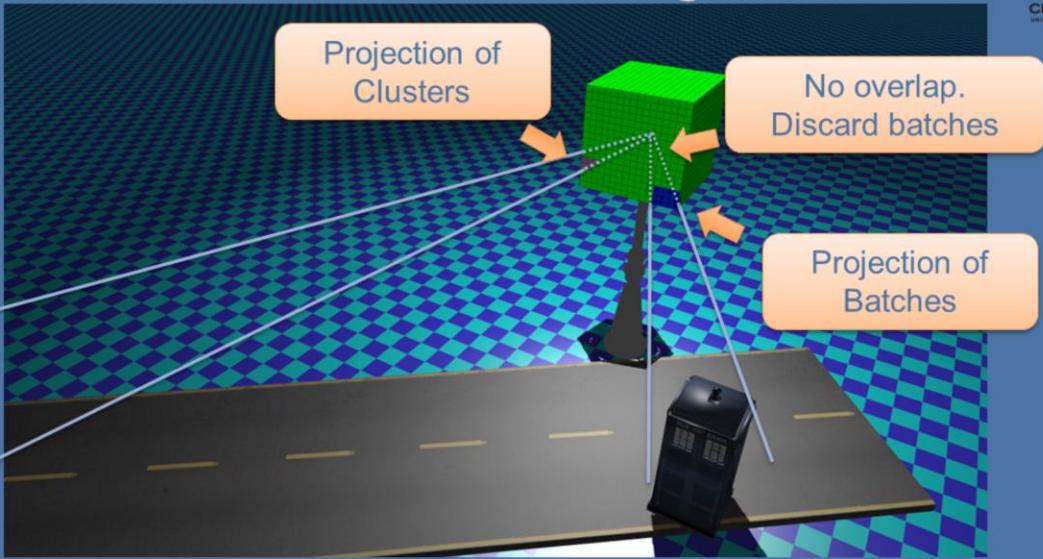
Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

Better Culling

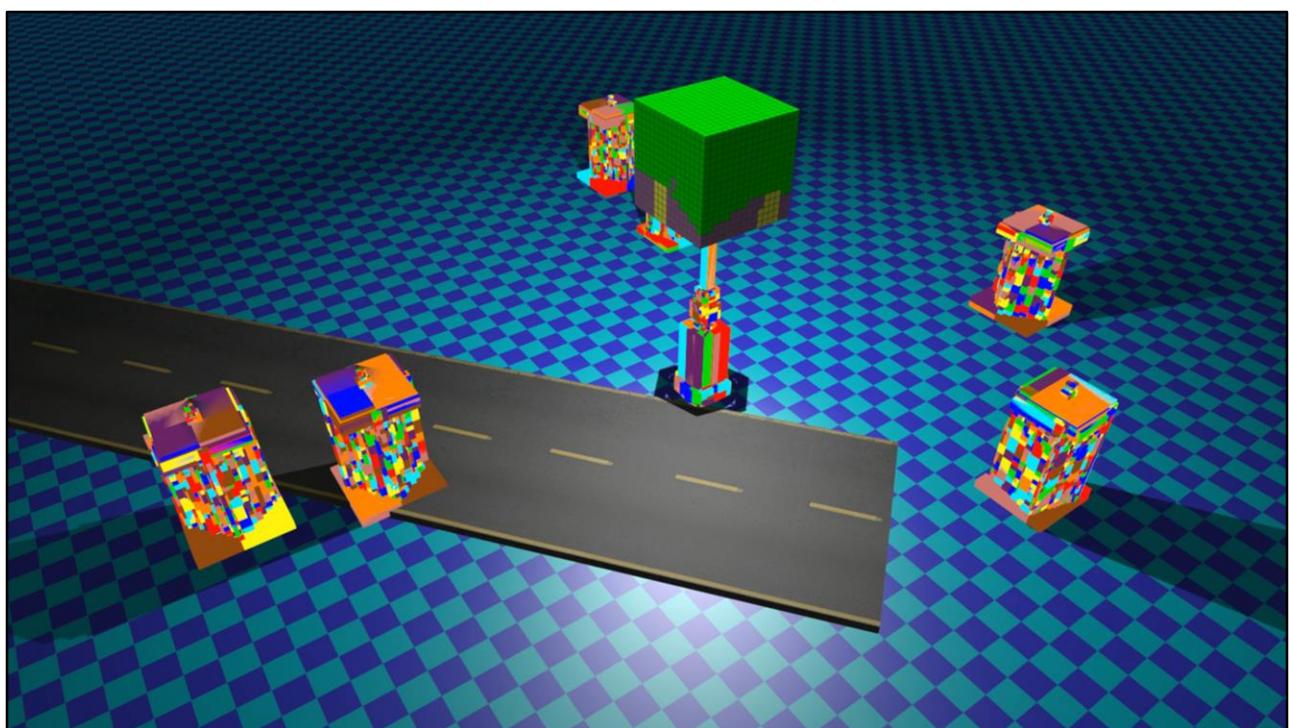


Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

Better Culling



Real-Time Many-Light Management and Shadows with Clustered Shading – 2015



Prune Batches

- Calculate projection maps for lights.
- Loop over result of basic culling
 - Compute batch mask.
 - Compare to projection map.
 - Bitwise AND
 - Discard if no overlap

Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- We implement this with a pass over all the results from the previous step.
- The kernel just loads the light and cluster, and computes the projection for the cluster
- For any face where there is no overlap, the cube face mask bit is cleared, meaning it will not be drawn.
- Finally the updated CFMs are stored back, now usually with fewer bits set, and sometimes zero.

Solution breakdown

- Which lights cast shadows?
- Required resolution for each shadow map?
- Manage shadow map memory.
- Cull shadow-casting geometry.
- Draw shadow maps.
- Shade scene.

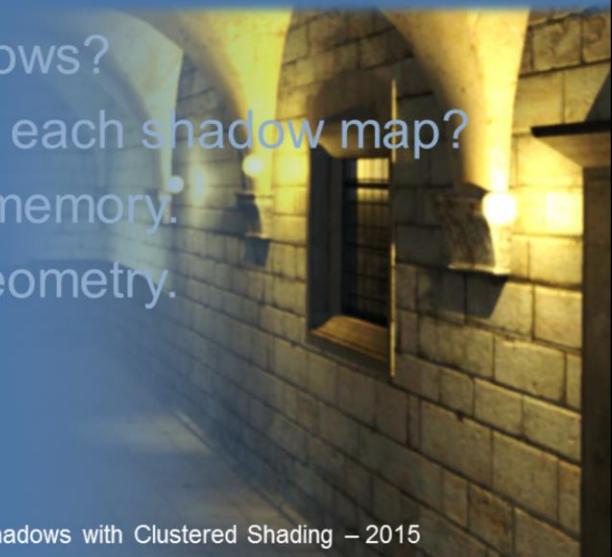
Real-Time Many-Light Management and Shadows with Clustered Shading – 2015



Solution breakdown

- Which lights cast shadows?
- Required resolution for each shadow map?
- Manage shadow map memory.
- Cull shadow-casting geometry.
- Draw shadow maps.
- Shade scene.

Real-Time Many-Light Management and Shadows with Clustered Shading – 2015



Drawing

- Culling on GPU.
- Builds draw commands.
 - GPU buffer.
- One draw call / light
 - `glMultiDrawElementsIndirect(...);`



Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- As all the culling is performed on the GPU, using CUDA, make use of modern OpenGL and build draw commands also on the GPU
- Then we just call multi draw indirect once for each cube map, though it could be done once for all in principle.

Results

- Peak ~250k batches.

Necropolis, Peak Triangles/Frame

Naïve (x6)	CFM	Projection Map	Difference
126 M	20 M	13 M	~10x

Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- At the peak, we draw around 250 thousand batches into the shadow maps, with instancing on top.
- Using the culling techniques just outlined, this results in some 13 million triangles
- which is almost 10 times better than the naïve approach of replicating the triangles to all cube faces.
- And some 65% of performing culling just using the cube face mask.
- The cube face mask path actually already detects completely empty cube faces, and so is already quite effective.

Results

- Peak ~250k batches.

Necropolis, Peak Triangles/Frame

Naïve (x6)	CFM	Projection Map	Difference
126 M	20 M	13 M	~10x
~150 ms?	25 ms	16 ms	Doable!

Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- Again, put differently, this is the difference between infeasible and something we can do in real time.

Results

- Peak ~250k batches.

Optimization Note:
 Achieved triangle
 rate: ~800M Tris/s
 GeForce Titan
 peak:
 ~4.2G Tris/s
 Difference: ~5x

Necropolis, Peak Triangles/Frame

Naïve (x6)	CFM	Projection Map	Difference
126 M	20 M	13 M	~10x
~150 ms?	25 ms	16 ms	Doable!

Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- As a side note, our implementation achieved a fifth of the theoretical peak triangle rate
- So there could be some pretty good improvements for the handy optimizer out there.

Solution breakdown

- Which lights cast shadows?
- Required resolution for each shadow map?
- Manage shadow map memory.
- Cull shadow-casting geometry.
- Draw shadow maps.
- Shade scene.

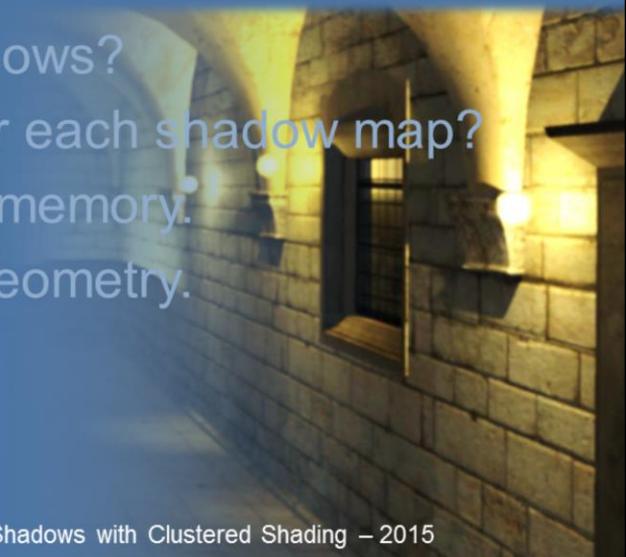
Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- We now have all the components needed to shade the scene.

Solution breakdown

- Which lights cast shadows?
- Required resolution for each shadow map?
- Manage shadow map memory.
- Cull shadow-casting geometry.
- Draw shadow maps.
- Shade scene.

Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

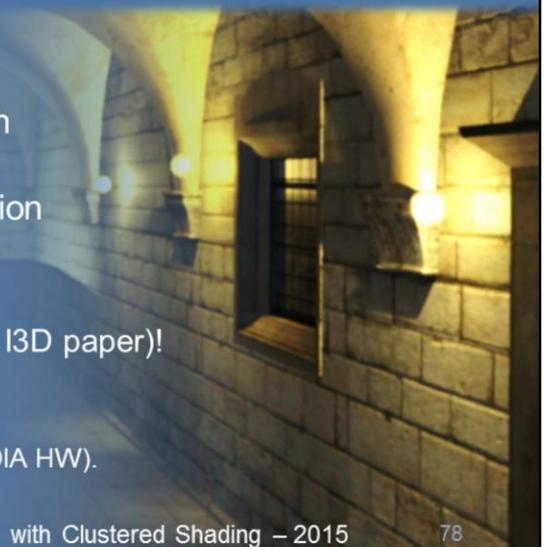


Shading (2015 edition)

- Loop over lights
 - Light index \Leftrightarrow shadow map index
- Calculate cube face index from direction
 - GCN: cubeFaceIndexAMD [OpenGL14].
- Calculate cube face coords. from direction
- Sample 2D layer in shadow map
- Note:
 - Not using bindless textures (contrary to I3D paper)!
 - Reason:
 - Conformant to ARB Spec.
 - 2X performance increase in shading (NVIDIA HW).

Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

78

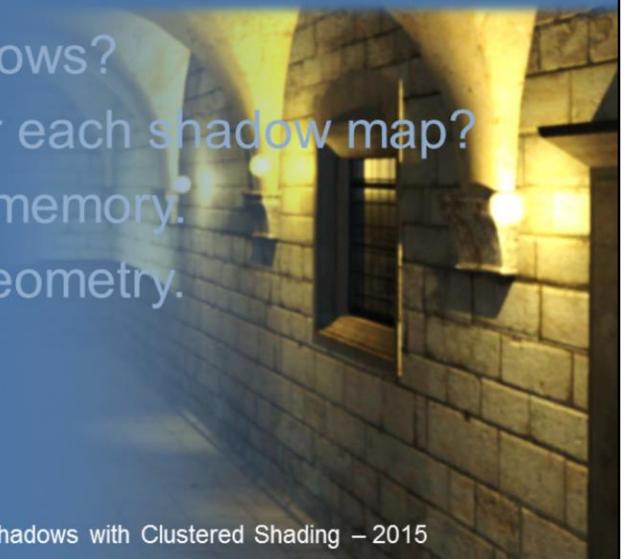


- So shading the image is simple
- The light index is just used to look up the shadow map
- Which is stored in an array texture, and so on.
- The reason we now use an array texture and not bindless textures (like last year) is that it is not actually conformant to the ARB spec, and is only supported on NVIDIA hardware.
- But even there it is very slow.

Solution breakdown

- Which lights cast shadows?
- Required resolution for each shadow map?
- Manage shadow map memory.
- Cull shadow-casting geometry.
- Draw shadow maps.
- Shade scene.

Real-Time Many-Light Management and Shadows with Clustered Shading – 2015



Solution breakdown

- Which lights cast shadows?
- Required resolution for each shadow map?
- Manage shadow map memory.
- Cull shadow-casting geometry.
- Draw shadow maps.
- Shade scene.
- Done!

Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

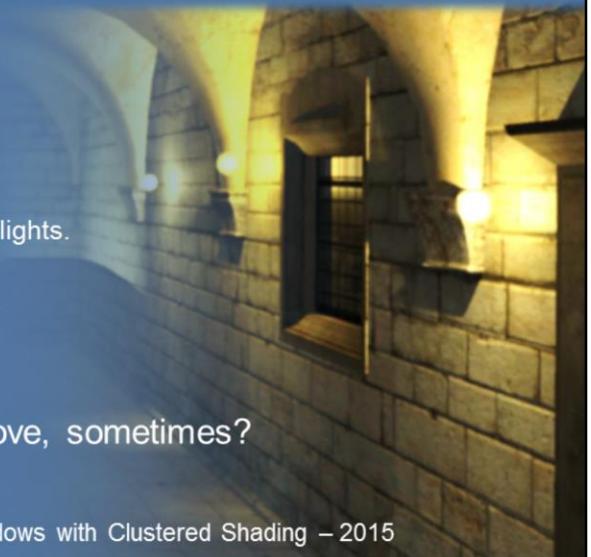
- So we're done... well almost.

EXPLOTING STATIC SCENE ELEMENTS

Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

Static scene elements

- Static lights & shadow casters
 - Precompute shadows [Kämpe15]
 - Voxelized shadows.
 - High-quality filtering.
 - Shadow Proxies [Valient14]
 - Pre-compute static geometry for static lights.
 - Render proxies + dynamic at run time.
- Static shadow casters
 - Ray tracing [Olsson15, Harada13]
 - Signed distance fields?
- Problem: What if shadow casters move, sometimes?



Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

Extending our algorithm

- Keep shadow maps
 - Easy!
 - All SMs allocated up front
- Don't re-draw unchanged portions
 - Not quite as easy...
 - But lets try!

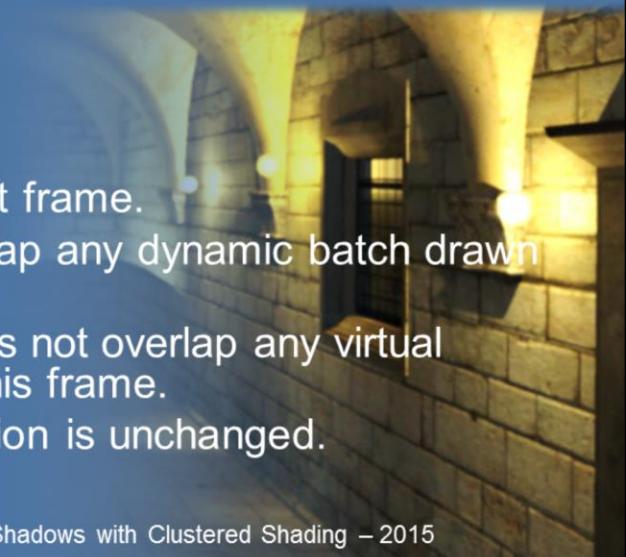


Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

Yet more efficient culling

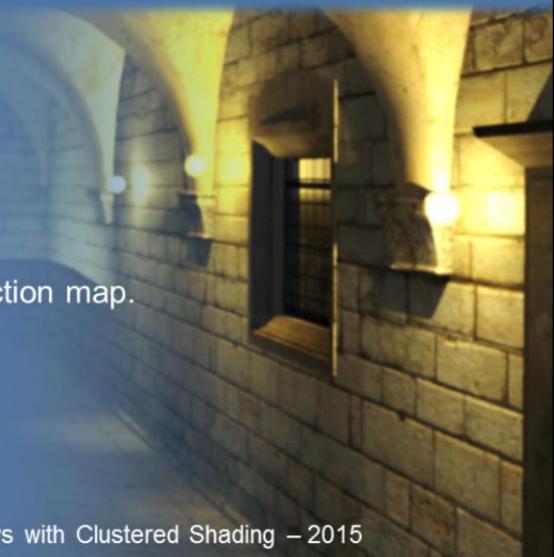
- Discard batch if:
 - The light is static.
 - The batch is static.
 - The batch was drawn last frame.
 - The batch does not overlap any dynamic batch drawn the previous frame.
 - The batch projection does not overlap any virtual page that is committed this frame.
 - The shadow-map resolution is unchanged.

Real-Time Many-Light Management and Shadows with Clustered Shading – 2015



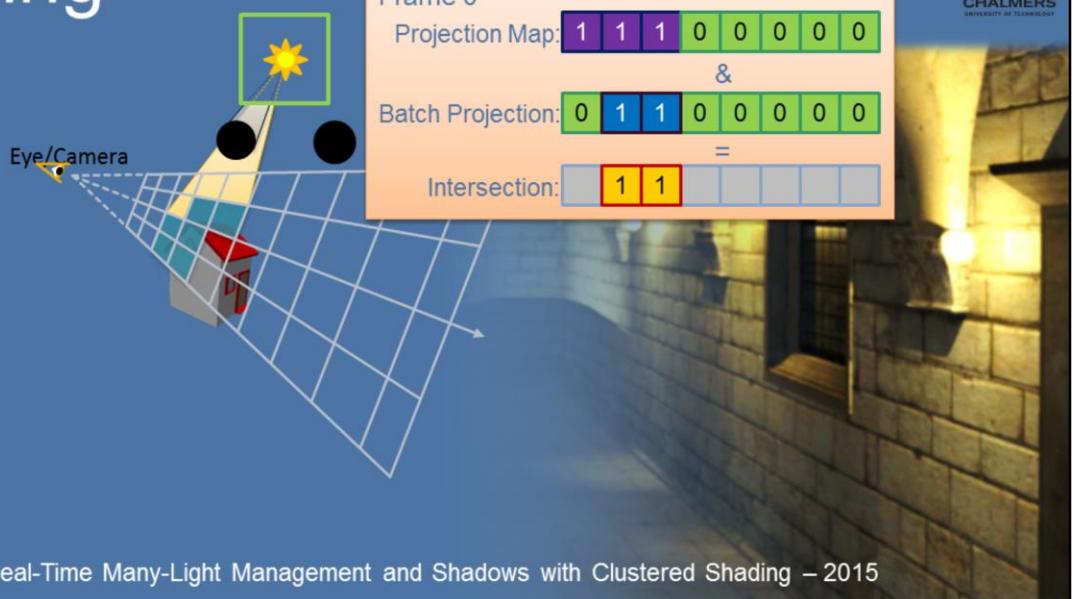
New problems

- Light/Batch static?
 - Just flag them based on animation etc.
- Batch drawn last frame?
 - Flag? - Too many...
 - Test previous frame's projection map!
- Overlaps dynamic?
 - Record dynamic batches in new projection map.
- Newly committed?
 - Test against virtual page mask.
- Clear 'trace' after dynamic batches
 - Draw point sprites to clear.



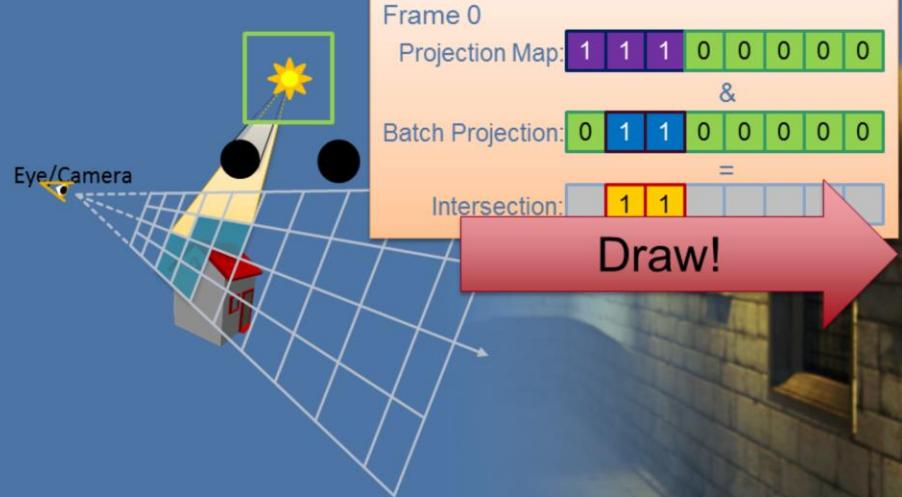
Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

Culling



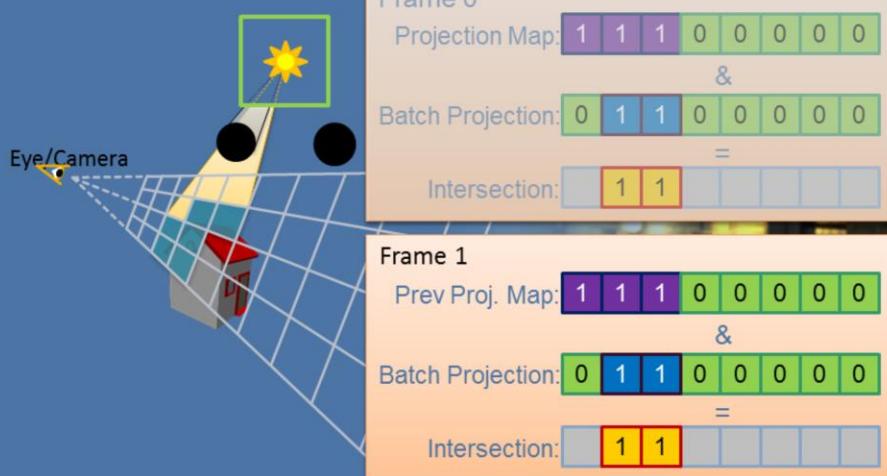
Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- To recap the logic for our improved culling, this time in 1D
- We have the projection map of the light,
- Then build the map for the batch and test against it



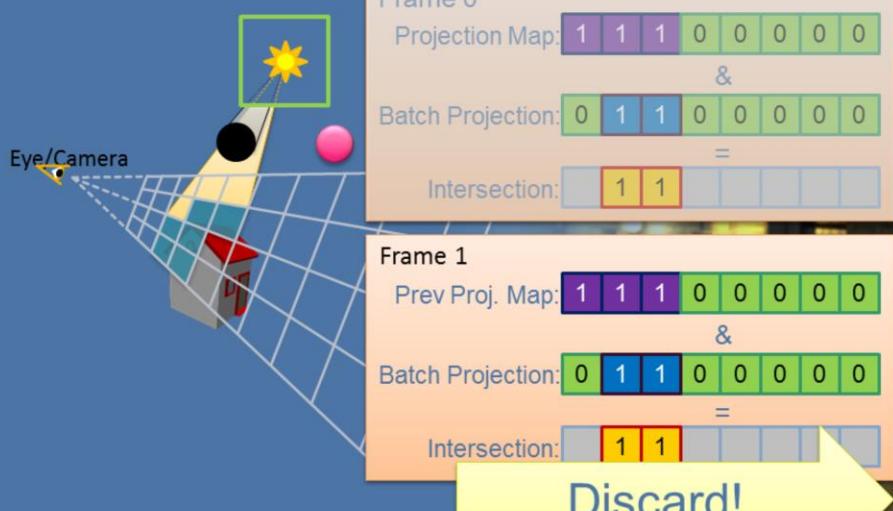
Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- When there is any intersection, we draw the batch.



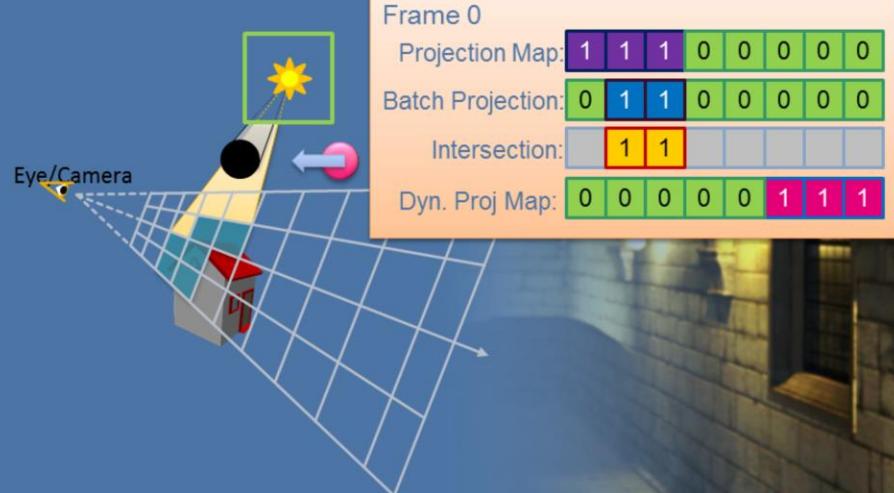
Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- Now, the following frame, imagine nothing has changed.
- We now make use of the projection map of the light from the *previous* frame.
- This is tested against the batch projection, same as before!



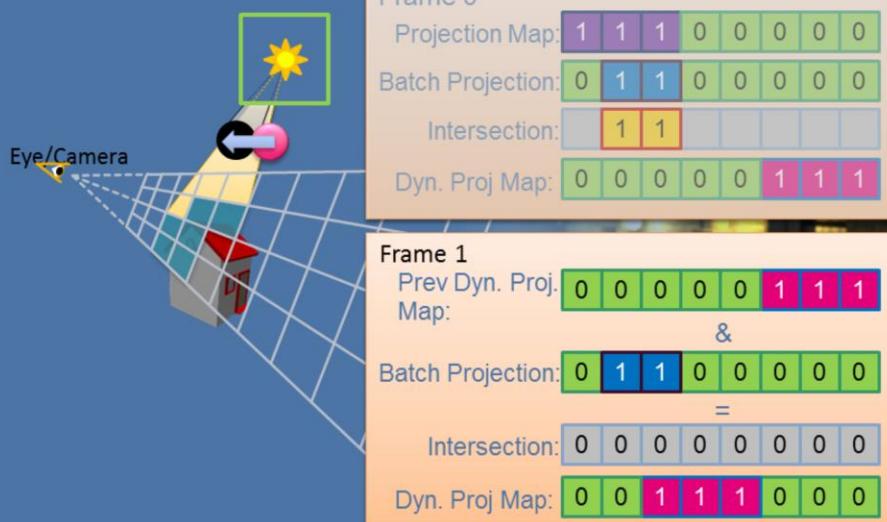
Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- If there is an overlap, we should discard the batch, because it must have been drawn last frame.
- Recall both light and batch are static so neither could have moved.



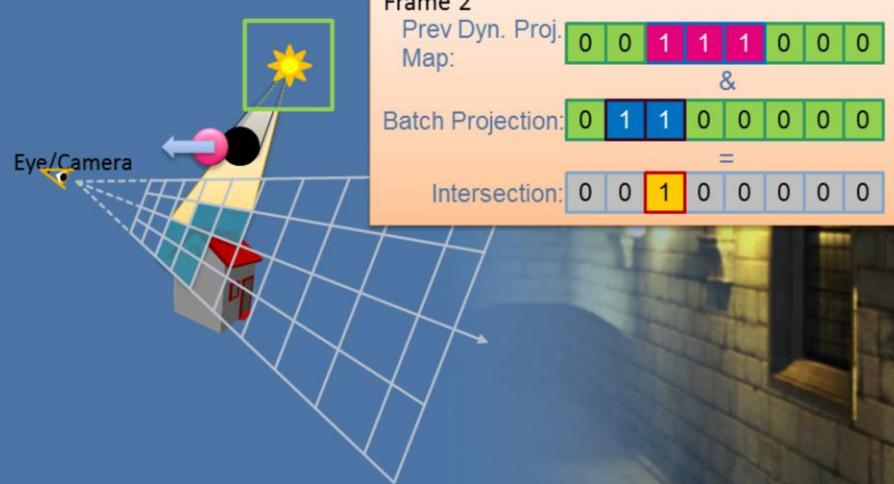
Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- Now let's make the second batch dynamic and moving left.
- We build a dynamic batch projection map while culling, that records any dynamic batches drawn.



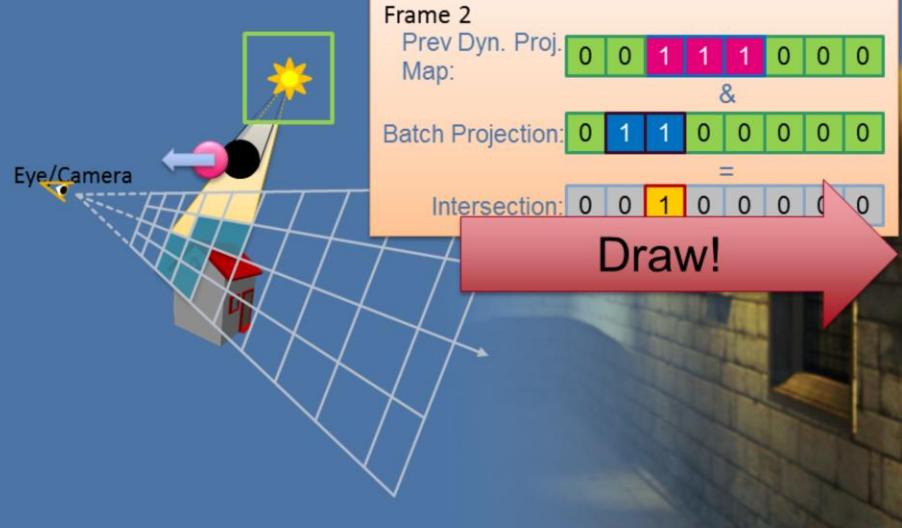
Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- The following frame we intersect this map with the batch projection map
- There is no overlap so we can still discard the static batch.
- But note how the dynamic batch actually now overlaps the static batch...
- Which means that next frame, it must be re-drawn.



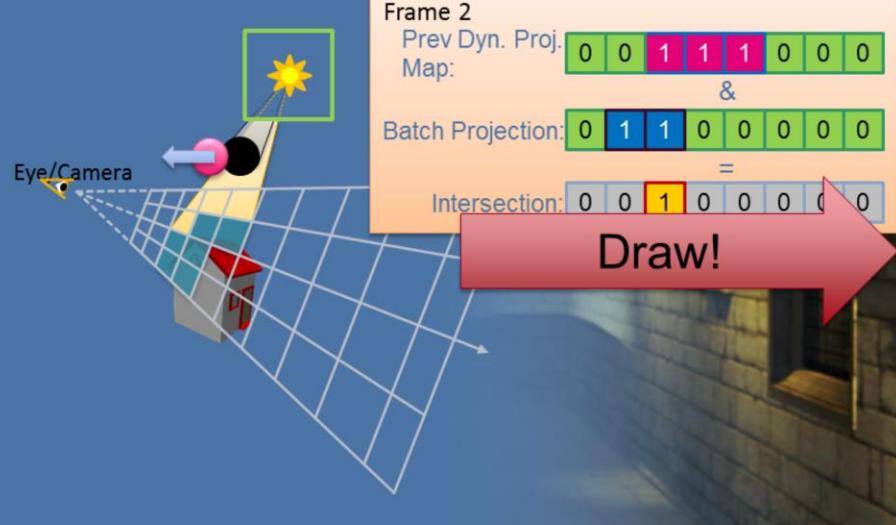
Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- Which we discover by comparing the previous frame's dynamic projection map.



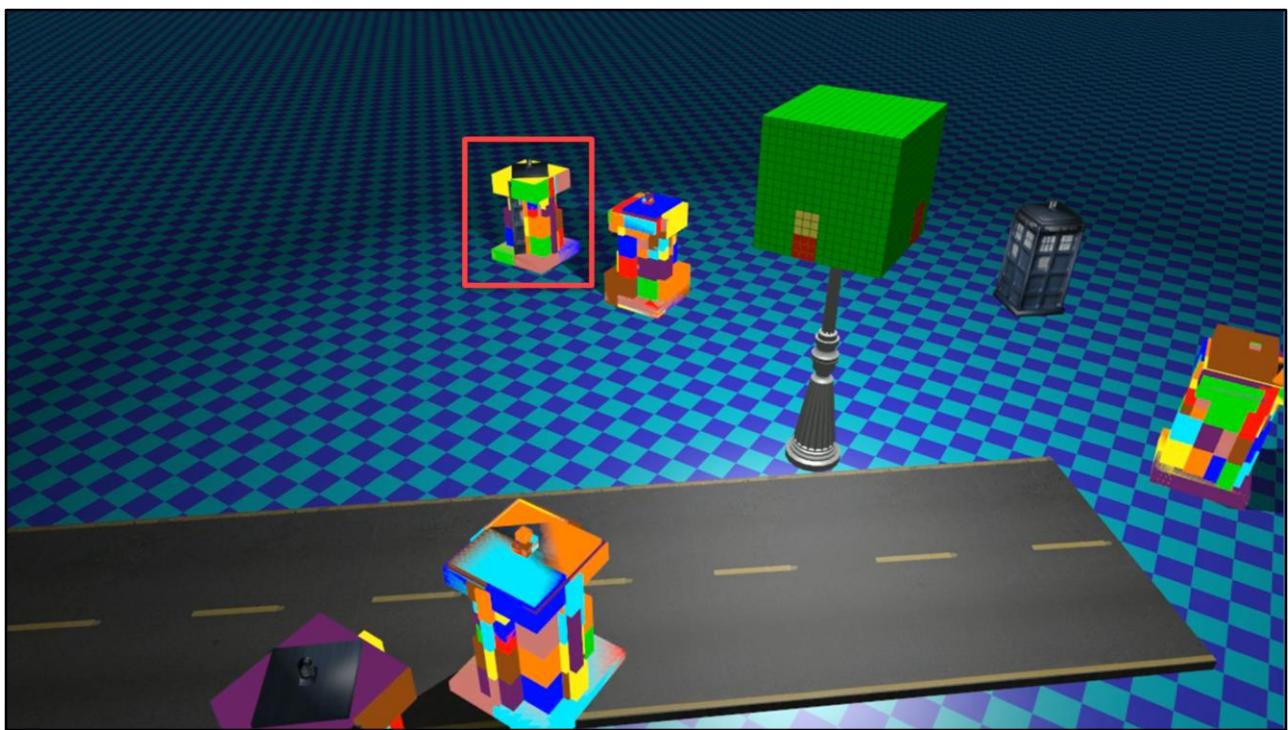
Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- Which we discover by comparing the previous frame's dynamic projection map.

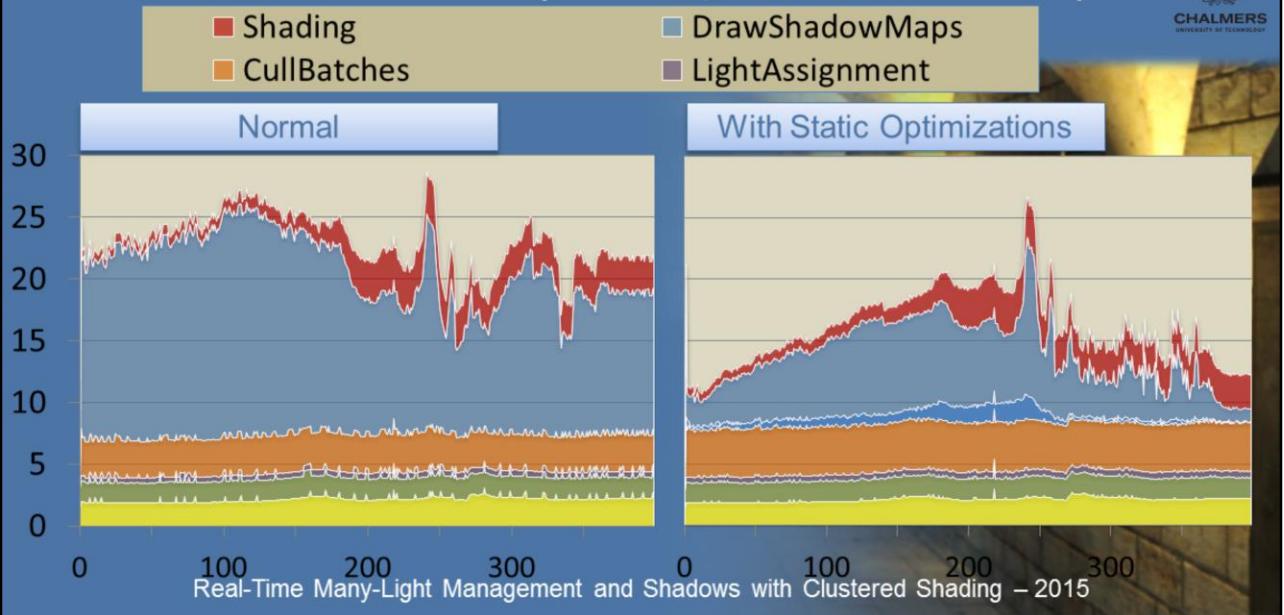


Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- The dynamic projection map tells us what parts to clear and re-draw.
- Page commits have the same effect, essentially, and are treated similarly.
- This means that we can make minimal changes to the shadow maps with very little extra work and overhead.
- So at one extreme, if nothing moves, nothing is drawn.



Performance (Necropolis animation)



- This plot shows how this works out for the heaviest scene in the extended paper, necropolis.
- As you can see, the work is substantially reduced for most of the animation, although the peak is only moderately improved.
- This is a scene where most of the lights in the central area are moving, so this is perhaps not surprising.

Future work

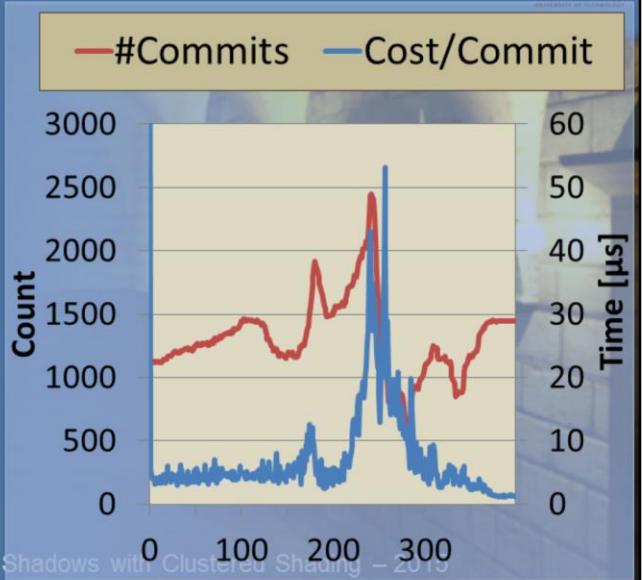
- Our design too forgetful!
 - Drops pages as soon as possible.
 - Changes in resolution causes re-draw.
- Instead
 - Keep pages as long as possible
 - Fixed size memory pool
 - LRU policy for re-use?
 - Keep resolution longer (if static)?
 - Resample higher-res?

Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

- Our design currently drops pages as soon as they are not used.
- This means that we might drop a page only to need to re-render it very soon again.
- If this memory is not needed elsewhere, we caused an unnecessary redraw.
- This was not a problem before we started retaining static information,
- but now a more conservative scheme, for example based on a LRU policy is probably better.
- Or some other policy that avoids discarding a rendered page unless needed.
- For static lights some quality might also be traded for keeping shadow maps around longer.
- When a shadow map becomes lower res, we could even try resampling the higher res shadow map, to save further rendering.

Problem #1 – Slow commits

- Page commits
 - Up to ~2500 commits / frame
 - Average: ~7 μ s / commit.
 - \rightarrow 17ms / frame
 - Peak: ~40 μ s / commit
 - \rightarrow ~100ms / frame
 - High variability.
- Getting better
- Need batch API
 - Like DX11



Real-Time Many-Light Management and Shadows with Clustered Shading – 2010

- I have to bring a couple of issues to your attention.
- The first, that page commits are slow, has gotten a lot better.
- It used to be a couple of milliseconds per commit.
- However, while it can be usable for real-time performance today, there is still a high variability in the cost.

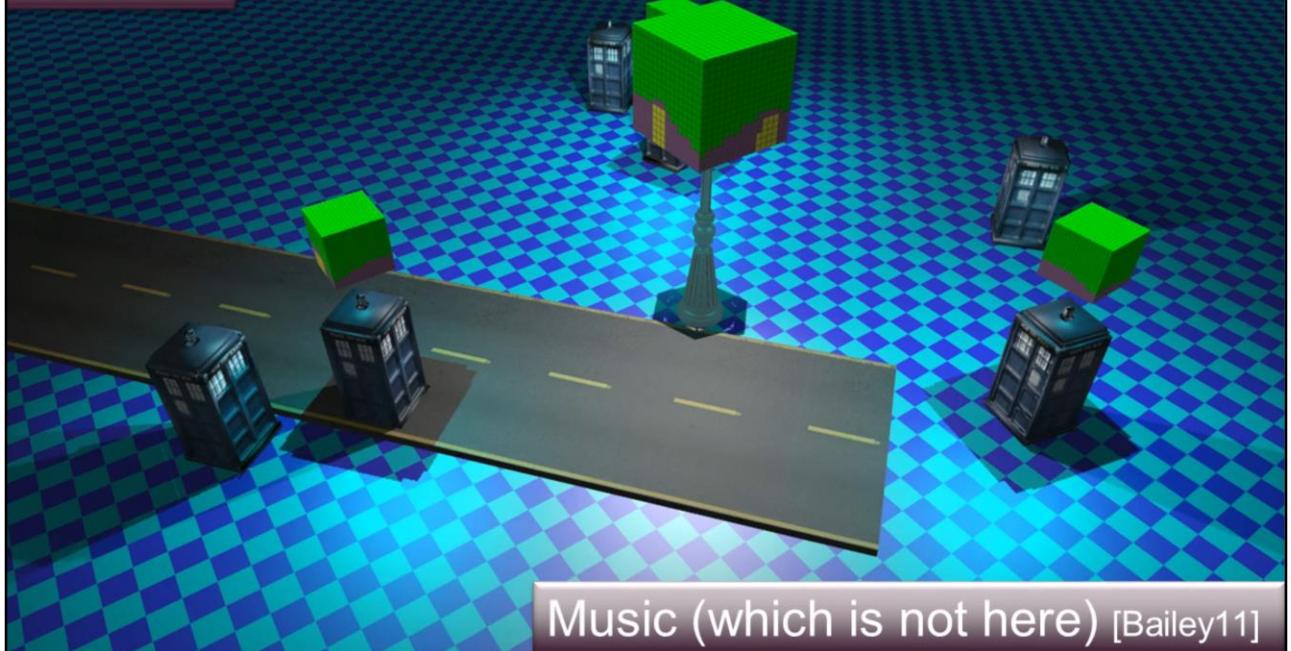
Problem #2 – Clearing large array textures

- Virtual storage
 - $2048 * 8K2 = 128G$ texels
- Clear affects all layers
- Workaround #1
 - Texture views, clear 6 layers
 - Still, very slow (NVIDIA HW)
 - ~90ms / frame
- Peak Physical: 161M texels
- Workaround:
 - Draw point sprite for each committed page.
 - ~1ms / frame



Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

The End!



Music (which is not here) [Bailey11]

References

- [Persson13] Practical Clustered Deferred and Forward Shading, Persson & Olsson 2013, <http://advances.realtimerendering.com/s2013>
- [Olsson15] Olsson, O.; Billeter, M.; Sintorn, E.; Kampe, V.; Assarsson, More Efficient Virtual Shadow Maps for Many Lights, TVCG 2015
- [Valient14] Reflections and volumetrics of Killzone Shadow Fall , SIGGRAPH14, <http://advances.realtimerendering.com/s2014>
- [Andersson14] Rendering Battlefield 4 with Mantle, GDC 2014
- [Shulz14] Moving to the Next Generation - The Rendering Technology of Ryse, GDC 2014.
- [Harada12] A 2.5D culling for forward+. In SIGGRAPH Asia 2012 Technical Briefs, ACM, 18:1–18.4.
- [Lauritzen12] Andrew Lauritzen, Intersecting Lights with Pixels, In SIGGRAPH 2012 Courses, Beyond Programmable Shading,
- [Ferrier11] Alex Ferrier and Christina Coffin, Deferred shading techniques using frostbite in "Battlefield 3" and "Need for Speed the Run", SIGGRAPH 2011 Talks
- [Sintorn14] Erik Sintorn , Viktor Kämpe , Ola Olsson and Ulf Assarsson, Per-Triangle Shadow Volumes Using a View-Sample Cluster Hierarchy, I3D 2014
- [Trebilco07] Damian Trebilco, 2007 (Shader X7 2009), GitHub: <https://github.com/dtrebilco/lightindexed-deferredrender>
- [OpenGL14] OpenGL extension registry, ARB_sparse_texture, ARB_bindless_texture, AMD_gcn_shader
- [Coombes14] Coombes, GDC 2014, Taking Advantage of DirectX11.2 Tiled Resources
- [Forsyth04] Forsyth, GDC 2004, Multiple Shadow frustra, http://home.comcast.net/~tom_forsyth/papers/shadowbuffer_pseudocode.html
- [Lefohn07] LEFOHN, A. E., SENGUPTA, S., AND OWENS, J. D. 2007. Resolution-matched shadow maps. ACM Trans. Graph. 26, 4 (Oct.).
- [King04] G. King and W. Newhall, "Efficient omnidirectional shadow maps," in ShaderX3: Advanced Rendering with DirectX and OpenGL, 2004.
- [Holländer11] HOLLÄNDER, M., RITSCHEL, T., EISEMANN, E., AND BOUBEKEUR, T. 2011. ManyLoDs: parallel many-view levelof- detail selection for real-time global illumination. Computer Graphics Forum 30, 4, 1233–1240.
- [Stürzlinger97] Wolfgang Stürzlinger and Rui Bastos. Interactive rendering of globally illuminated glossy scenes. In Proc. of the Eurographics Workshop on Rendering Techniques '97, pages 93–102. Springer-Verlag, 1997.
- [Ritschel08] T. Ritschel, T. Grosch, M. H. Kim, H.-P. Seidel, C. Dachsbacher, and J. Kautz, Imperfect shadow maps for efficient computation of indirect illumination. ACM Trans. Graph., 2008

Real-Time Many-Light Management and Shadows with Clustered Shading – 2015

References

- [Harada13] T Harada, J McKee, JC Yang, Forward+: A step toward film-style shading in real time, GPU Pro 4, 2013
- [Dachsbaecher06] Carsten Dachsbaecher and Marc Stamminger. Splatting indirect illumination. In Proc. I3D '06, page 93–100. ACM, 2006.
- [Kämpe15] KÄMPE, V., SINTORN, E., AND ASSARSSON, U. 2015. Fast, memory-efficient construction of voxelized shadows. In Proc. I3D '15.
- [Bailey11] Dr Qui, <http://www.youtube.com/watch?v=68wJlQbCtI>

Papers / Slides / Demo (when it is finished!) implementation with source: <http://www.cse.chalmers.se/~glaolss>

Real-Time Many-Light Management and Shadows with Clustered Shading – 2015