Michal Valient, Lead tech @ Guerrilla.
We're makers of the Killzone franchise where you get to shoot these bad guys.

Today I'm going to talk about techniques that we and a lot of the other game teams use to render shadows in their games.

I'm going to focus on specific examples from shipped titles.

OUR DAILY CHALLENGE

30fps = 33ms

13ms - geometry

10ms - post process

10ms - lights & shadows

GPU from 2005/06

Siggraph 2012 - Efficient Real-Time Shadows - Michal Valient

Let’s first take a look at what challenges a graphics programmer faces.

You have 33ms to render the whole frame. 13ms is taken by geometry and all the fancy shaders and roughly 10ms is taken by the post processing.

That means there’s 10ms left for all lighting and shadowing calculations.
If you spend 5ms on shadow map rendering and 3ms on your physically correct lighting calculations, you’re pretty much left with 2ms for any kind of shadow filtering.

For the people still quoting Frames Per Seconds this actually means that shadow filter demo needs to run at 300 FPS on the PS3/XBox class hardware if it’s targeting games.

And I’m not even talking about the 60FPS games.
People who make these are made of Awesome-sauce and I’m sure they all grew up riding velociraptors and fighting polar bears.

OVERVIEW

Most games use very similar shadow techniques:

Percentage closer filtering, Variance shadow maps, Cascaded shadow maps

We'll focus on few game specific tricks that help to increase shadow performance or quality.

You will notice that most of the games use very similar techniques and a few special game-specific tricks.
The short version for the people who have to catch their flight:
percentage closer filtering and cascaded shadow maps are still pretty much a norm today and
variance shadow maps are increasingly being used more and more...

That's of course unless your company is called Media Molecule and your game is Little Big Planet.
Because these guys just decided to ignore all common sense and they're just happily
voxelizing the scene every frame to compute the irradiance volume and
use raycasting to render the volumetric shadows and ambient occlusion.
And that all on PS3.

Alex Evans presented their algorithms on Siggraph last year,
look it up, the topic is quite complex.

So meanwhile back on earth where the rest of us common game developers have to work...

Our recent game – Killzone 3 – represents the common conservative shadow approach with all the basic tricks in one package.

Pretty much everything you need to shoot the space nazis.

KILLZONE 3

Cascaced shadow maps, PCF, stabilization

10+ shadow casting lights

Single 1408x1408xD16 shadow map

5000+ objects

Shadow map rendering costs: 4-5ms

We use deferred renderer, run at 720p and 30FPS on PS3.

Pretty standard cascaded shadow maps renderer with maximum 4 cascades.
Pixel stabilization to avoid shimmering (you can find it in ShaderX6 article)

We usually have 10 or more shadow casting lights on screen and
we reuse single shadow map which means our
transparent objects have no shadows.

We generate entire display list on SPUs so number of objects is generally not a problem –
5000 objects rendered per frame is reasonable number and
shadow map render alone costs us around 5ms and it is our biggest optimization target.

KILLZONE 3

Simple optimizations, configurable per light

Remove small shadow casters

Distance based shadow map resolution

Agressive shadow fade out

Use depth bounds

We avoided the more complicated optimizations for shadows and we keep it simple.

We determine for each object if it has big enough contribution to the shadow
based on the bounding sphere size in shadow map and on the screen.
If the size is bellow certain threshold we don't render it into shadow map.
This alone removes incredible amount of objects for sunlight without visual glitches.

For local lights we also half the shadow map resolution based on the light distance.
And we fade out the shadows fairly aggressively either
based on the light size or the light distance.

As pretty much most of the PS3 games we use depth bounds check during the light rendering.

KILLZONE 3

- Pre-rendered sun shadows
- Allows for shorter cascade ranges
  - Last cascade at 80 meters
- Used to drive stencil test for real-time sun
  - Improves speed and quality

Perhaps the most interesting bit is our usage of pre-rendered sunlight shadows
stored in one of the lightmap channels for each object.

We mix the pre-rendered shadows with the real-time ones and
It allows us to disable real-time sun shadows very quickly - at around 80 meters.

We also use the pre-render shadows to quickly skip the pixels
that are completely in shade during the sunlight rendering.

So what do we do...

Here's a scene from Killzone which contain mix of real-time and pre-rendered shadows.

This is the g-buffer channel that contains the pre-rendered shadows.

Siggraph 2012 - Efficient Real-Time Shadows - Michal Valient

We run a full-screen pass which reads g-buffer channel and marks
the fully shadowed pixels in the stencil buffer (shown red here)

These are the pixels where we don't run any further computation.

Here’s a scene rendered exclusively with the real-time shadows.

Siggraph 2012 - Efficient Real-Time Shadows - Michal Valient

Here's the scene with real-time shadows mixed with the pre-computed shadows.

As you can see we got much more soft shadows and even ambient occlusion for free.

Here’s the final frame again.

If you’re interested check the Guerrilla GDC 2009 presentations
where we cover our tech in great detail.

Siggraph 2012 - Efficient Real-Time Shadows - Michal Valient

Sat, September 1, 2012

The next game I would like to talk about is inFamous.

inFamous is an open world game franchise from Sucker Punch Production
part of Sony Computer Entertainment.

The games are set in urban environment with a lot of buildings and long view distances.

INFAMOUS 2

Cascades resolved to screen space mask

Pre-lit impostors beyond 150 meters

SPU culling against hand placed occluders

Vertical edges match u-coord in shadow map

Last cascade used for particle shadows

The shadow mapping basics are very similar to what I mentioned just moments ago for Killzone.

The sun can have up to 3 cascades and they resolve and blend them into a screen space buffer to avoid harsh transitions the between cascades.
Real-time lighting ends beyond 150 meters where they use pre-lit impostors.

The world contains large artist placed occluders (for example buildings) which serve two purposes:
Occluders are rendered to shadow map to cheaply prevent light leaking.
All potential shadow casters are culled against these occluders on the SPU to reduce the GPU load.

There are two nice game specific tricks worth mentioning:
Since the world contains a lot of buildings the shadow map projection is chosen specifically so that vertical edges in the world match u-coordinate in the shadow map
This prevents much of the shadow aliasing on buildings.

The last shadow cascade is also preserved and used during the particles rendering.

INFAMOUS 2

Specialized setups for cutscenes

Perspective shadows just for heads

4-light rig for characters

Use stencil to limit the effect

8000-9000 objects, 4-6ms

inFamous uses specialized lighting setups for cutscenes:

Head closeups use perspective shadow map aimed at tight box around the head.

The characters in cutscenes use a special 4 light rig.
One light casts detailed shadows, 3 lights non shadowed.
Whole setup is rendered as fullscreen pass and uses depth bounds and stencil tests to limit the effect on the character.

The interiors use another specialized setup where only the 4 closest spotlights cast shadows.

Siggraph 2012 - Efficient Real-Time Shadows - Michal Valient

This image shows the high precision of the cutscene shadows.
You can see that shadows on the character are much more precise than the ones in the background.

The team behind God Of War PSP games and God Of War collection for the PS3

READY AT DAWN ENGINE

Light Indexed Deferred

Pool of 16 shadow maps

Cascaded EVSM for directional lights

Sample Distribution Shadow maps

They have been experimenting with light indexed deferred where one stores much simpler g-buffer with just indices of lights that hit given pixel.

The geometry is rendered with full materials and the shaders read light indices from the g-buffer and perform lighting.

This also means that all shadow maps used by lights need to be present during the whole frame. Therefore there's a pool of 16 shadow maps.

The engine uses exponential variance shadow maps for smooth filtering and Sample distribution shadow maps for increased precision.

I'm not going to spend more detail on this - just google all the terms and read the papers.

Ready At Dawn Engine

Extensive offline shadow preprocessing

Rasterize Depth + ObjectID buffer

Use stencil for overdraw counting

No overdraw means no visible shadow

Collect ObjectIDs to determine shadow casters

Ready At Dawn aim for a lot of lights per frame with high amount of shadowed lights.

Most of the lights are static and therefore they implemented very efficient offline shadow culling process.

For each light rasterize depth and object id buffers and use stencil buffer to count the overdraw.
Zero overdraw means there's no shadow receiver on that pixel (for example floors) and
the shadow casting object can be rejected.

The ObjectIDs collected from the shadow map represent all shadow casters for the light and
are used to build a stripped down version of the scene that can be rendered quickly.
Memory vs. drawcall cost.

3rd installment of the popular Far Cry franchize by Ubisoft.
Huge open world, focus on minimal precomputation,
real-time GI, check the GDC slides.

FAR CRY 3

Dynamic long range sun shadows

Used beyond regular cascaded SM

World split into 64x64m tiles, 144 loaded

Each sector has 32x32 top down SM

Stores the shadow occlusion height

I'm going to talk about previously unpublished technique and I'm really grateful that Ubisoft and Mickael Gilabert allowed me to talk about it.

Far Cry 3 uses a custom shadow rendering solution that maps well to their large open world.
They use cascaded shadow map for the closer distances and blend to the long range shadows after 40 meters.

The world is split into 64 by 64 meter tiles around the player, 144 tiles are loaded at the same time on the console.

The long range shadows are stored in a single top-down 768 by 768 shadow map texture with 32 by 32 tile associated with each sector and stores the sunlight occlusion height.

FAR CRY 3

Update of one tile:

Frame 1 - Render 256x256 shadow map

Frame 2 - Render top down view of the tile

- **Test shadow at fixed heights**
- **Store the occlusion height**

Frame 3 - Downsample to 32x32

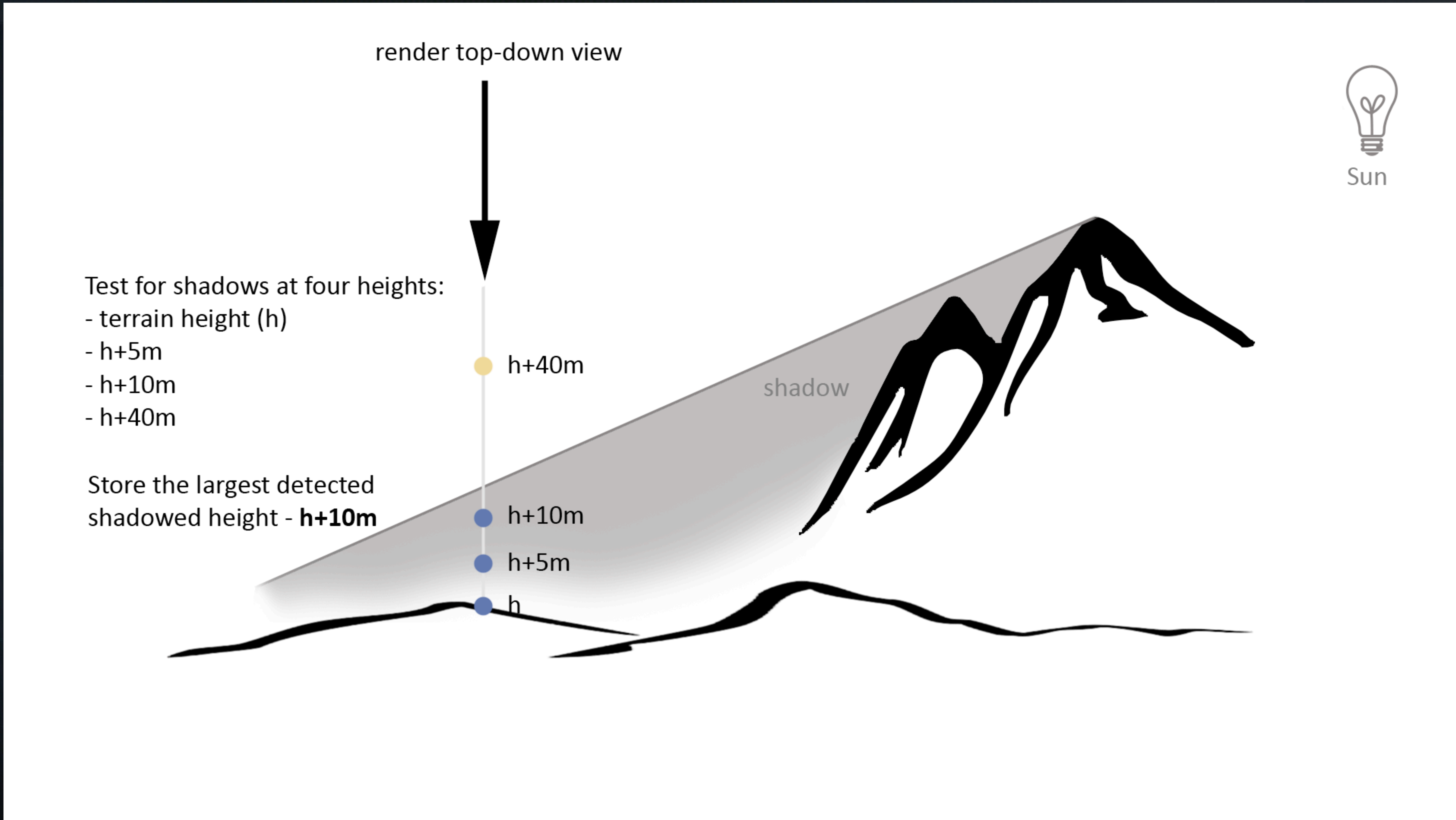
Only one tile is updated at a time, update staggered across multiple frames.

First render the sun shadow map for the given tile into a 256x256 texture.

Then render the top down view of the tile into a 192x192 texture.
The shader performs shadow test for a fixed set of heights above the terrain –
terrain level, 5 meters above, 10 meters above and 40 meters above the terrain.
The shader then outputs the minimum sunlight occlusion height.

As the final step downsample the top down texture to 32x32 and store in the shadow map.

During the frame render just project to the 2D space of the shadow map and test the height.



The image illustrates the occlusion height map rendering process (sideview)

Far Cry 3 uses heuristic to determine the tiles that need to update the shadow map
For example just loaded tiles or tiles where a lot of objects has been unloaded are rendered first.

The advantages of this approach is better quality of the distant shadows
- Stable, behaves well during filtering and interpolation, completely dynamic.

The high-end PCs use 48x48 tile.


FAR CRY 3
FARCRY3

God Of War is one of the biggest Playstation franchises.
It's epic third person action adventure where you take the role of Kratos, the god of war.

It's probably the only game where you get to fight a Greek god size of the building
all that while climbing up the back of a few hundred meters high Titan.

GOD OF WAR 3

Forward rendered with screen-space shadows

Shadow direction is detached from lights

All cascades share single shadow map

Artists can specify the shadow texel per meter

- Tiled cascade shadow map

5ms GPU budget for all shadow processing

The games scale and perspective requires detailed shadows both far away and close by.

God Of War uses forward renderer, but it uses depth-pre-pass to
resolve the shadows into a screen space buffer very similarly to what you know from deferred renderer.

The shadows are actually detached from lights which allows for more artistic freedom.
Shadow casting directions can be associated with characters -
imagine character running through a hallway, nearest light affecting the shadow of the character.

A system of shadow map cascades that allows artists to specify shadow texel density per square meter.
All share single 1408x1408 cascade.
Each cascade can be split into tiles that are rendered separately based on the required resolution.

GOD OF WAR 3

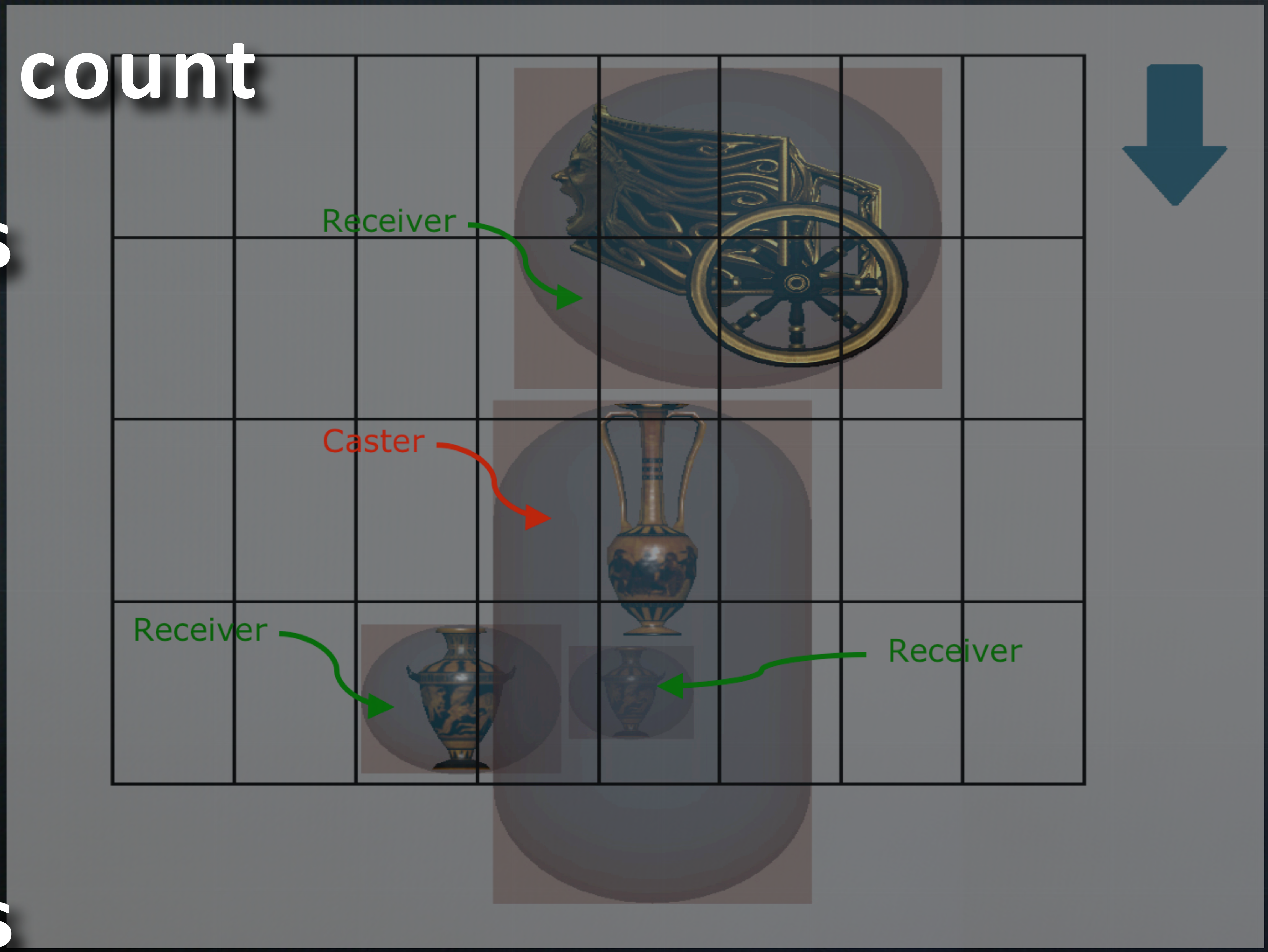
Uses SPU to reduce occluder count

Split cascade into 8x8x4 cells

What cells need shadows?

- Intersecting any casters
- Intersecting any receivers

Uses extruded object bounds



God of war also uses very simple and effective shadow caster culling mechanism.

Split the shadow cascade space into 8x8x4 cells.
2D case is shown in the image with light direction pointing down.

Use SPU job to test all shadow receivers against each cell.
Similarly the job tests all shadow casters against each cell.
This process uses object bounds extruded in the light direction.

Only shadow casters which intersect with cells containing shadow receivers need to be rendered.

GOD OF WAR 3

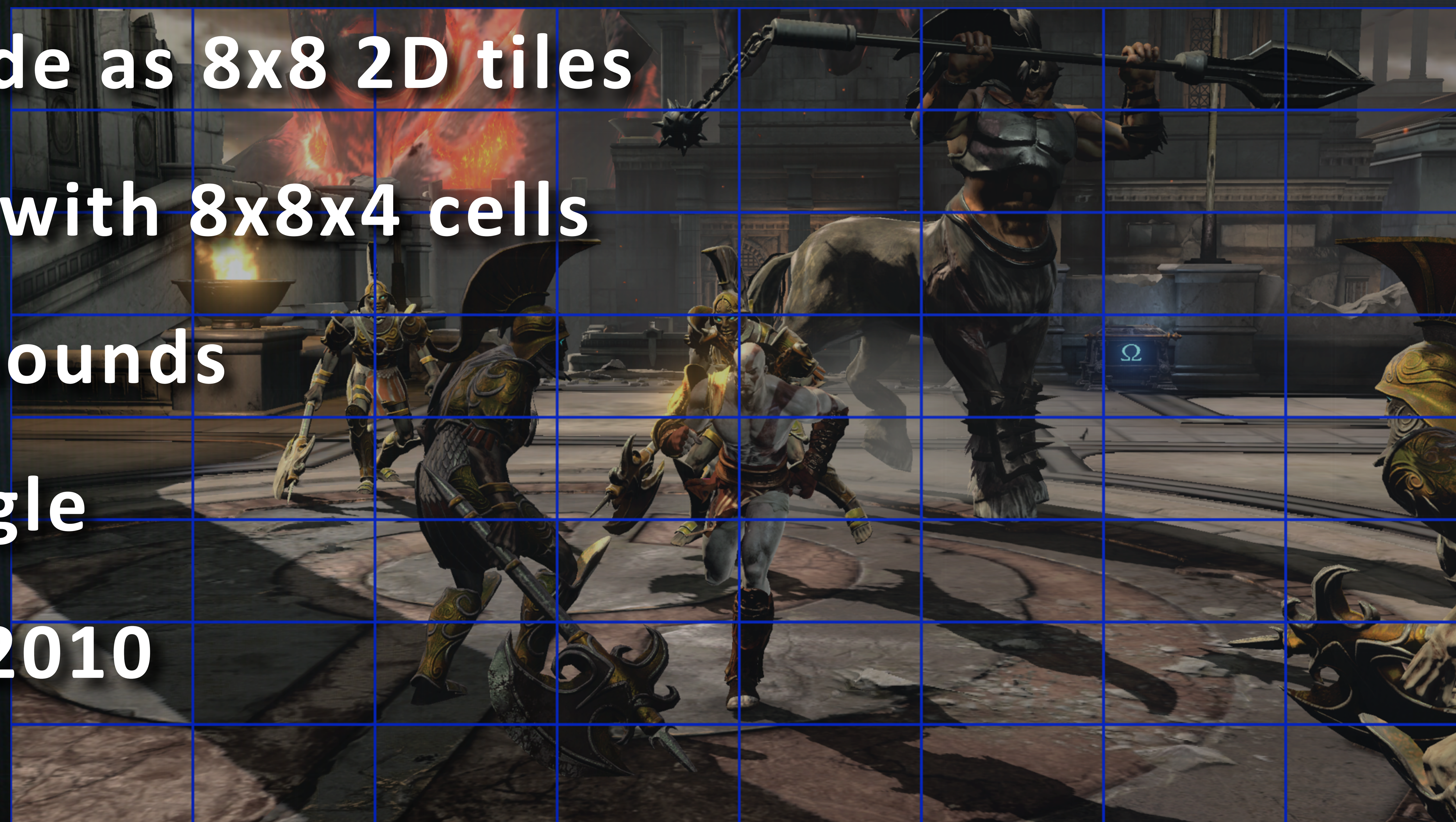
Render each cascade as 8x8 2D tiles

Intersect each tile with 8x8x4 cells

Determine depth bounds

+ scissors rectangle

Presented at GDC 2010



The 3D cell structure can be used to further optimize the rendering of the cascades into the screen buffer.

In God Of War they render each cascade as 64 separate tiles and for each tile first compute intersection with the previously mentioned 3D cell structure. <**flip**>

Since each cell contains list of all shadow casters and receivers you can compute the conservative depth bounds and scissors rectangle required to render all shadow receivers which greatly reduces the cost of cascade rendering.

You can find more details in a GDC 2010 presentation from Ben Diamand.

GOD OF WAR 3

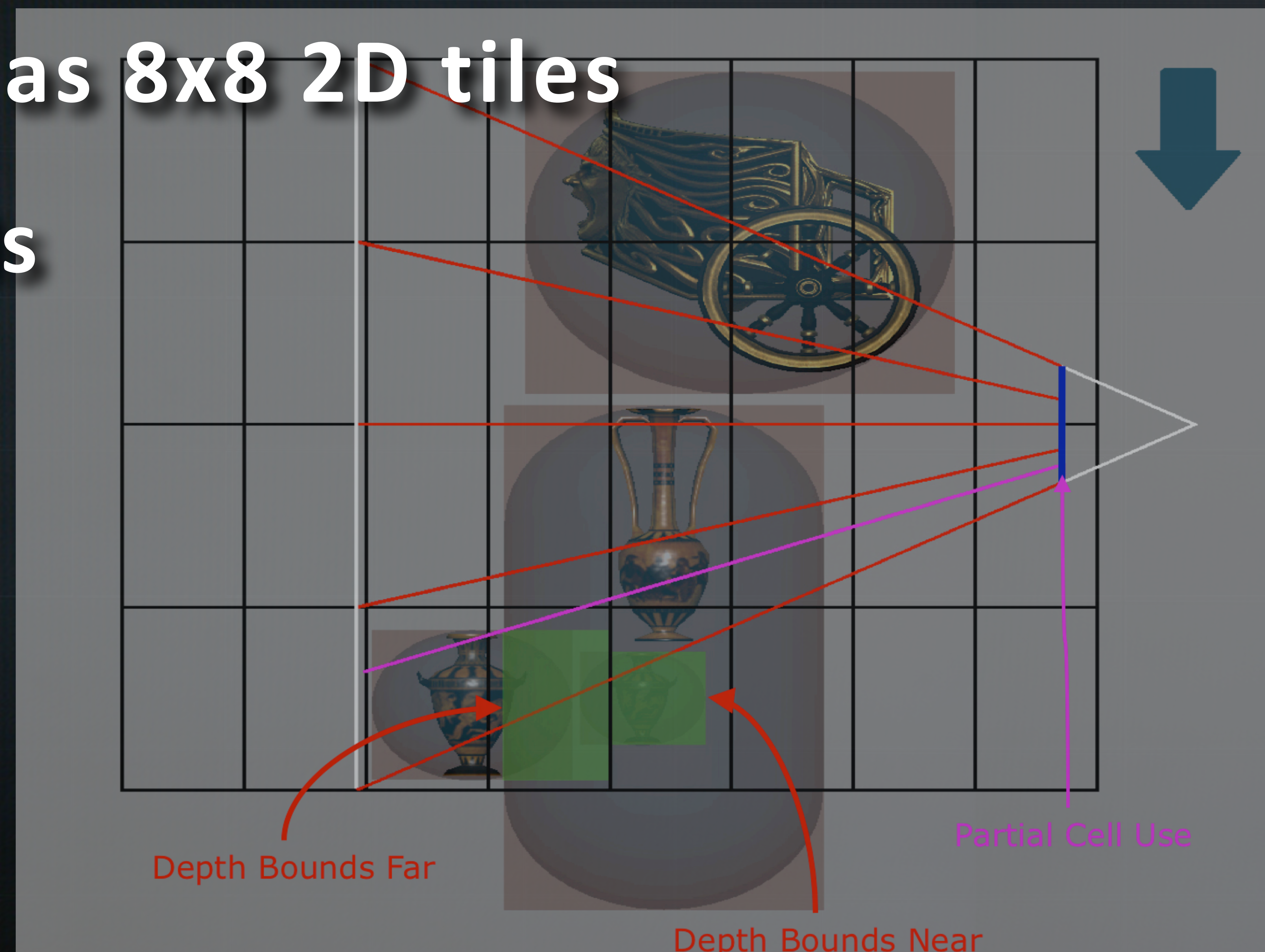
Additionally render cascade as 8x8 2D tiles

Intersect tile with 8x8x4 cells

Determine depth bounds

+ scissors rectangle

Presented at GDC 2010



The 3D cell structure can be used to further optimize the rendering of the cascades into the screen buffer.

In God Of War they render each cascade as 64 separate tiles and for each tile first compute intersection with the previously mentioned 3D cell structure. <**flip**>

Since each cell contains list of all shadow casters and receivers you can compute the conservative depth bounds and scissors rectangle required to render all shadow receivers which greatly reduces the cost of cascade rendering.

You can find more details in a GDC 2010 presentation from Ben Diamand.

Two Worlds 2 is a multiplatform open world role playing game developed by Reality Pump.

TWO WORLDS 2

Cascaded shadow maps

Lower resolution screen space mask

Bilateral upsampling

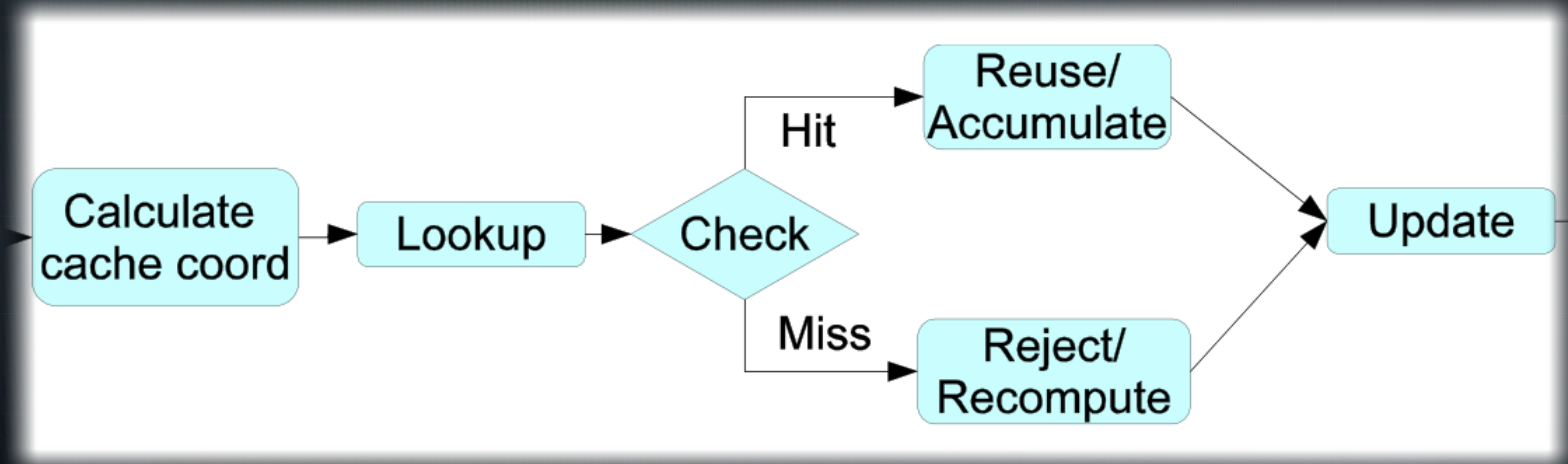
Temporal caching

Combined with SSAO

The games uses cascaded shadow map.
The shadow map filtering is happening in a half resolution screen space buffer
Later bilateral upsampling is used to upsample the shadow mask to the native resolution.

The interesting trick is that the game uses temporal caching to greatly
improve the shadow and SSAO quality with minimal cost.

TWO WORLDS 2



A quick introduction to the reprojection caching.

You have a buffer called reprojection cache where each pixel has a key and a value.
During the rendering you use previous frame camera matrix and/or motion vectors to determine the pixel position in the previous frame.

You lookup the cache at given position and compare the current key with the stored key value.
The key can be pixel color or pixel depth, anything of importance that tells you if the pixels match.
If you have a match you use the value from the cache, otherwise you have to recompute.

In both cases you update the cache with the new key and value data.

Two Worlds 2

Cache contains depth and shadow, SSAO values

Resolve shadow, SSAO to half resolution buffer

Accumulate with the cached values

Jitter / Alternate the samples to improve quality

Keep as new cache

The reprojection cache in Two Worlds contains pixel depth as a key and then accumulated shadow and SSAO values.

Each frame the shadow is resolved into the half resolution offscreen buffer using plain PCF filtering.
For each pixel you also read the value from reprojection cache and combine the two.

The game increases the shadow quality in many ways.
The camera and shadow matrix position is jittered a bit during the shadow resolve and the PCF and SSAO samples are rotated every frame.
This means that the cache is updated with new original data every frame.

TWO WORLDS 2

Siggraph 2012 - Efficient Real-Time Shadows - Michal Valient

Sat, September 1, 2012

Results

The technique has been published in Game Engine Gems 2 where you can find more details.

FUTURE

Volumetrics

Transparencies

Adaptive shadow methods

Voxelization

Still within 2ms :)

The future challenges lie mainly in the area of volumetrics, particle lighting and shadowing and transparencies and some generic framework that works for all these systems.

The precision and shadow map rendering improvement comes from adaptive methods.

I think the solutions based on voxels are also very interesting and have a great future because they allow to decouple the essential scene shadow casting blocks from the ever increasing scene polygon geometry.

Of course we want this all to be within 2ms.

THANKS

Anton Kirczenow
(Little Big Planet 2)

Ben Diamand
(God Of War 3)

Mickael Gilabert
(Far Cry 3)

Michal Drobot
(Two Worlds 2)

Ronan Marchalot
(Quantic Dream)

Stephen Hill
(Ubisoft Montreal)

David Neubelt
(Ready At Dawn)

Matt Pettineo
(Ready At Dawn)

Adrian Bentley
(inFamous 2)

Bill Rockenbeck
(inFamous 2)

Tiago Sousa
(Crytek)

Anton Kaplanyan
(KIT)

A lot of people talked to me about their tricks and gave me feedback, I hope I did not forget anybody.

Thank you all.