Clustered Forward Rendering and Anti-Aliasing in ‘Detroit: Become Human’

Ronan Marchalot
Lead Engine Programmer

GAME DEVELOPERS CONFERENCE® | MARCH 19–23, 2018 | EXPO: MARCH 21–23, 2018 #GDC18

UBM

Introduction

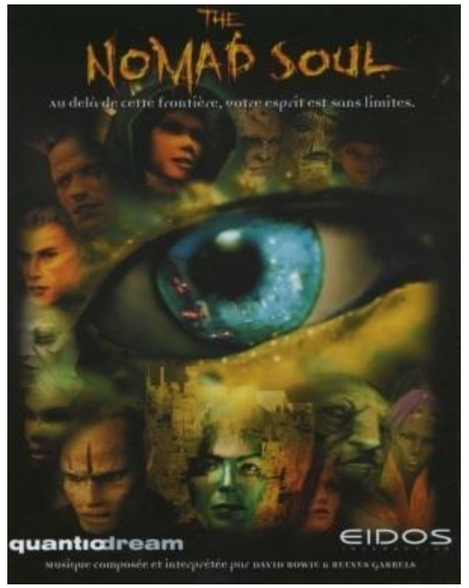
- Quantic Dream
- History of Quantic Dream 3D engine
- Building a new technology for “Detroit: Become Human”
- **Clustered forward rendering**
- **Temporal anti-aliasing**

Quantic Dream

- Independent French studio based in Paris
- Founded in 1997 by David Cage
- Work exclusively with Sony since *Heavy Rain*
- Specialized in “interactive dramas”
- Develop bespoke technology
- 200 employees

Quantic Dream

- Released titles
  - Nomad Soul (1999)
  - Fahrenheit (2005)
  - Heavy Rain (2010)
  - Beyond: Two Souls (2013)
  - Detroit: Become Human (2018)



Quantic Dream

- Technical demos
  - The Casting (2006)
  - Kara (2012)
  - The Dark Sorcerer (2013)

History of QD 3D Engine

- Proprietary engine
  - Optimized for Playstation hardware
  - PC OpenGL version for tools
  - Engine integrated in Maya for assets edition

History of QD 3D Engine

- Heavy Rain (2010)
  - Playstation 3
  - Forward rendering
  - Per-pixel lighting with normal maps
  - One shader per light
  - Shader tree (Authored in Maya)
  - MSAA 2X

History of QD 3D Engine

- Beyond: Two souls (2013)
  - Playstation 3
  - Deferred shading
  - Gamma correct
  - Physically Based Rendering
  - Morphological Anti-aliasing

History of QD 3D Engine

- The Dark Sorcerer (2013)
  - Playstation 4 tech demo
  - First port of our tech on PS4 with early SDKs
  - Deferred shading (5 render targets)
  - Improved materials (Cook-Torrance with specular color)

Building a new technology

- Detroit: Become Human
  - Interactive drama
  - Performance capture
  - Image quality

Building a new technology

- Detroit: Become Human
  - Takes place in a city
  - Lots of night scenes
  - Lots of interior scenes
  - Rain and snow


CYBERLIFE™


TAXI
BUSINESS CLASS

Building a new technology

- Detroit: Become Human
  - 30 FPS / 1080p
    - Not an action game!
    - Better graphics instead of better FPS

Building a new technology

- Detroit: Become Human
  - 30 FPS / 1080P
    - Not an action game!
    - Better graphics instead of better FPS
  - Loadings
    - Avoid loading screens

Building a new technology

- First list of features
- Most of them requires some space in the G-Buffer

Building a new technology

- First list of features
- Most of them requires some space in the G-Buffer
  - Normal-based bias for shadows

Building a new technology

- First list of features
- Most of them requires some space in the G-Buffer
  - Normal-based bias for shadows
  - Multi-layered materials (skin, rain, etc.)

Building a new technology

- First list of features
- Most of them requires some space in the G-Buffer
  - Normal-based bias for shadows
  - Multi-layered materials (skin, rain, etc.)
  - Self occlusion stored per vertex

Building a new technology

- First list of features
- Most of them requires some space in the G-Buffer
  - Normal-based bias for shadows
  - Multi-layered materials (skin, rain, etc.)
  - Self occlusion stored per vertex
  - Eye shader

Building a new technology

- If we want to pack everything in a G-Buffer, we could go beyond 8 render targets
- Different kind of materials clashes with deferred shading
- Deferred shading is fast, but we must keep things simple to obtains good performance
- We decided to go back to forward shading

Building a new technology

- Pillars of Detroit 3D engine

Building a new technology

- Pillars of Detroit 3D engine
  - Clustered forward rendering

Building a new technology

- Pillars of Detroit 3D engine
  - Clustered forward rendering
  - Temporal anti-aliasing

Building a new technology

- Pillars of Detroit 3D engine
  - Clustered forward rendering
  - Temporal anti-aliasing
  - Physically based rendering

Building a new technology

- Pillars of Detroit 3D engine
  - Clustered forward rendering
  - Temporal anti-aliasing
  - Physically based rendering
  - Character rendering

Building a new technology

- Pillars of Detroit 3D engine
  - Clustered forward rendering
  - Temporal anti-aliasing
  - Physically based rendering
  - Character rendering
  - FX

Building a new technology

- Pillars of Detroit 3D engine
  - Clustered forward rendering
  - Temporal anti-aliasing
  - Physically based rendering
  - Character rendering
  - FX
  - Loadings

Building a new technology

- Pillars of Detroit 3D engine
  - **Clustered forward rendering**
  - **Temporal anti-aliasing**
  - Physically based rendering
  - Character rendering
  - FX
  - Loadings

Clustered forward rendering

Clustered forward rendering

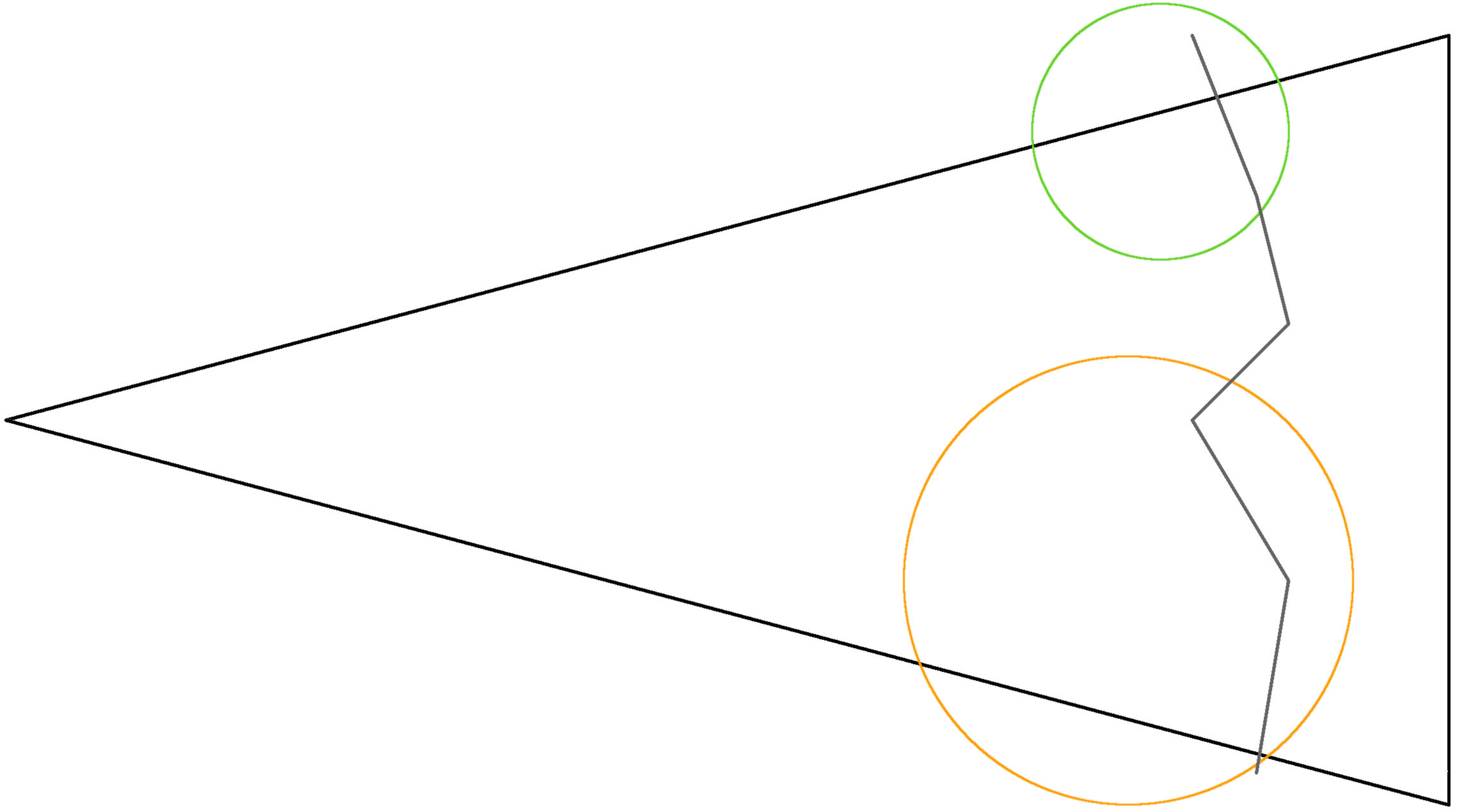
- GPUs are more flexible and efficient

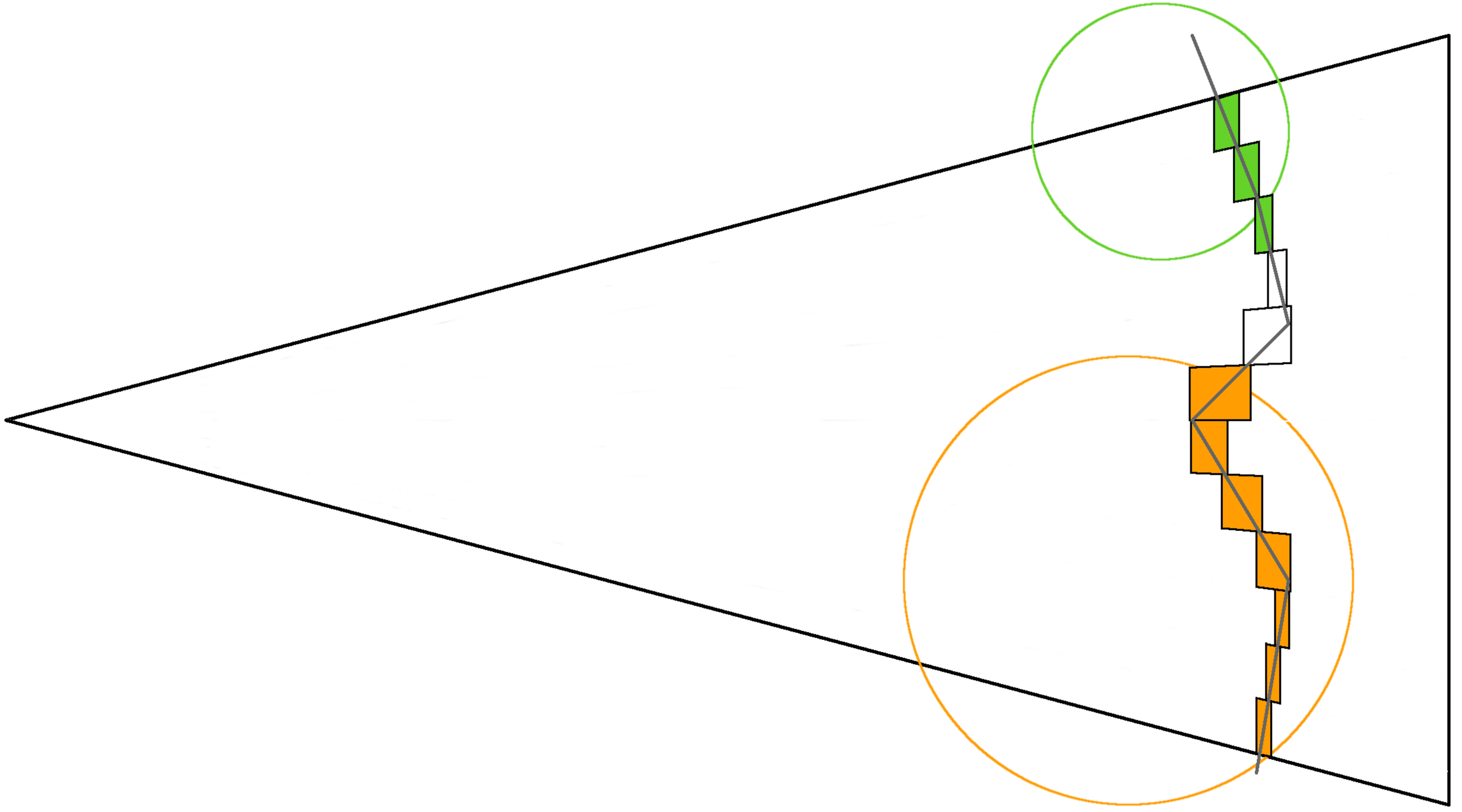
Clustered forward rendering

- GPUs are more flexible and efficient
- New lighting algorithms
  - Tiled rendering
  - Forward + rendering
  - Clustered forward rendering

Clustered forward rendering

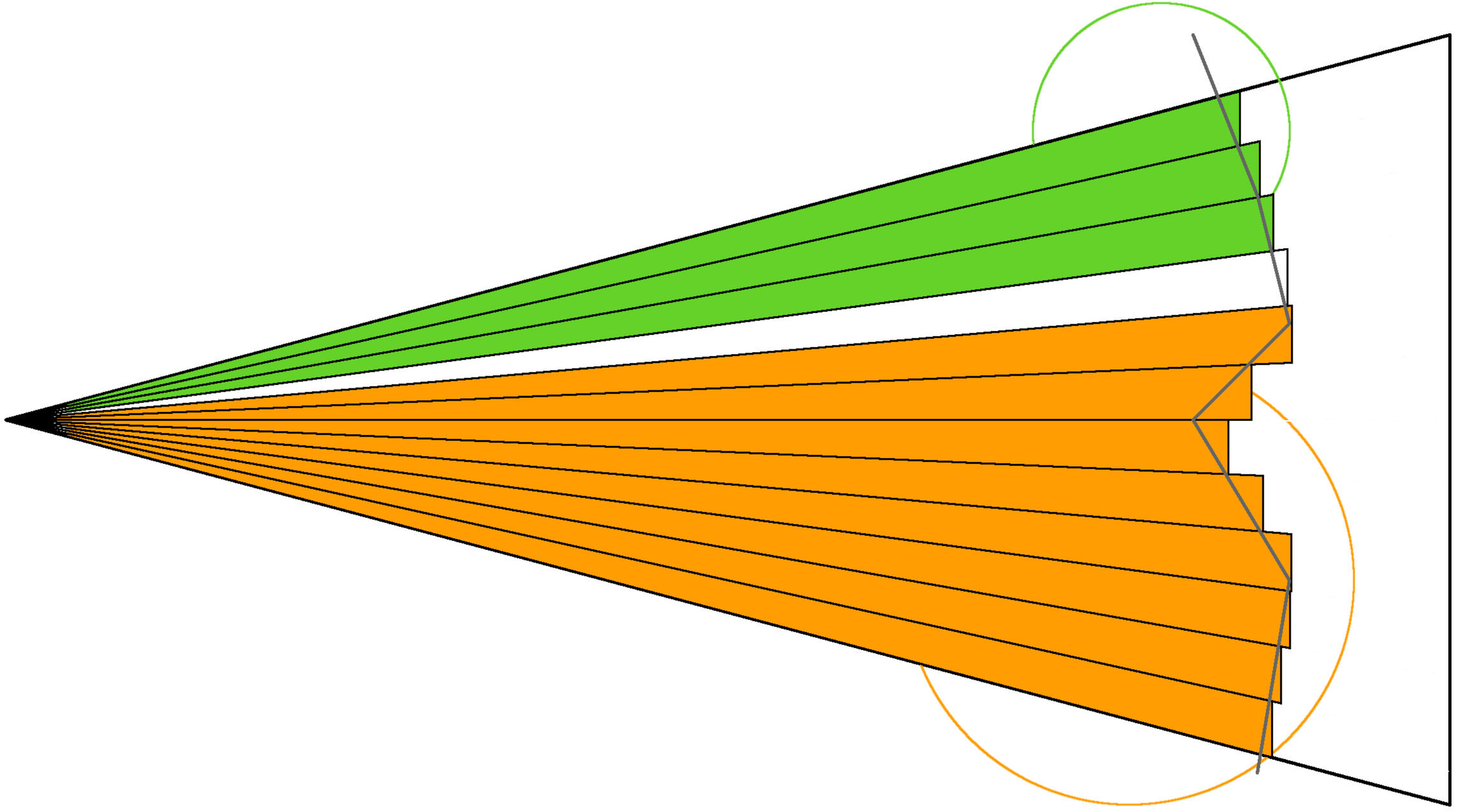
- Tiled rendering
  - The screen is cut into tiles
  - Fill a list of lights for each tiles
  - Perform lighting for each tiles
  - Saves bandwidth as we read G-Buffer once for many lights
  - Doesn’t support transparency





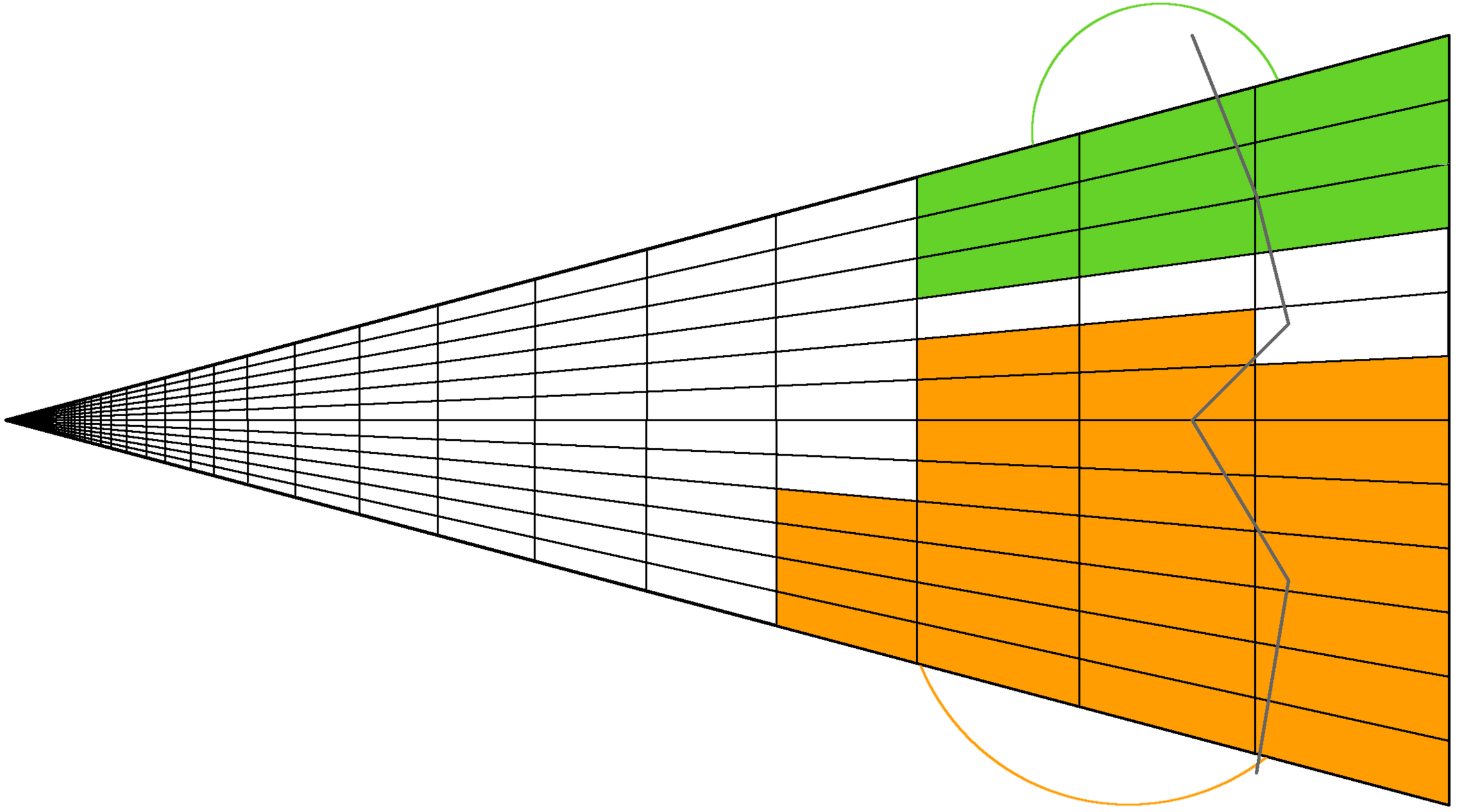
Clustered forward rendering

- Forward + rendering
  - The tiles are extended in depth
  - The list of lights contains all the lights between the Z far of the tile and the Z near of the camera
  - Support transparency



Clustered forward rendering

- Clustered forward rendering
  - The tiles are replaced by clusters in 3D
  - Depth distribution is not linear
  - Fewer lights per cluster than in forward+ rendering
  - But the number of clusters is > to the number of tiles



Clustered forward rendering

- First implementations
  - "Clustered Deferred and Forward Shading" by *Ola Olson et al.*, HPG 2012
  - Just Cause 3 (Avalanche)
    - "Practical Clustered Shading" by *Emil Persson*
  - Doom (Id software)
    - "The devil is in the details" by *Tiago Sousa* and *Jean Geffroy*


[GAME PAUSED]
GET YOURS TODAY!
SIDEWALK CLOSED AHEAD

Clustered forward rendering

- Data Structures
  - One buffer contains cluster data
    - 3D array
    - Width: 36, height: 20, depth: 64
    - First light index + light count

Clustered forward rendering

- Data Structures
  - One buffer contains cluster data
  - One buffer contains light data
    - 1D array
    - Light type, position, color, attenuation, etc.
    - Size = maximum light count

Clustered forward rendering

- Data Structures
  - One buffer contains cluster data
  - One buffer contains light data
  - One buffer contains light indices data
    - 16 bits indices
    - Size depends on maximum light density

Clustered forward rendering

- Fill clusters
  - Filled by asynchronous compute shaders
    - During the depth and shadow pass

Clustered forward rendering

- Fill clusters
  - Filled by asynchronous compute shaders
  - Each cluster is tested with all the light
    - Spot/Frustum, Point/Frustum, Box/Frustum
    - “Practical Clustered Shading” by Emil Persson
    - “Cull that cone!” by Bart Wronski

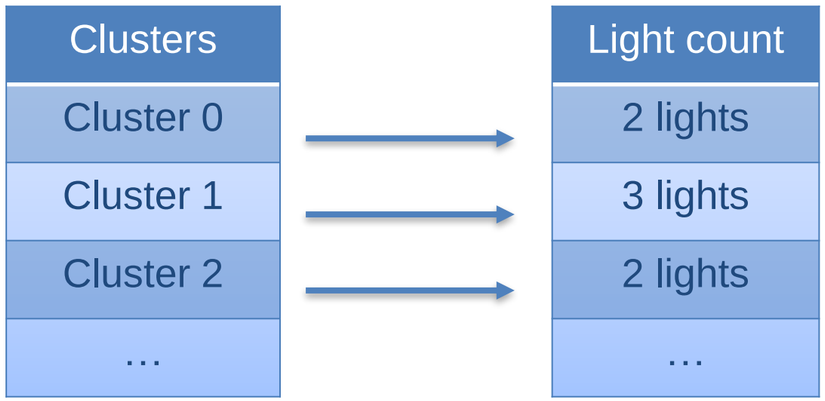
Clustered forward rendering

- Fill clusters
  - Filled by asynchronous compute shaders
  - Each cluster is tested with all the lights
  - 3 passes

Clustered forward rendering

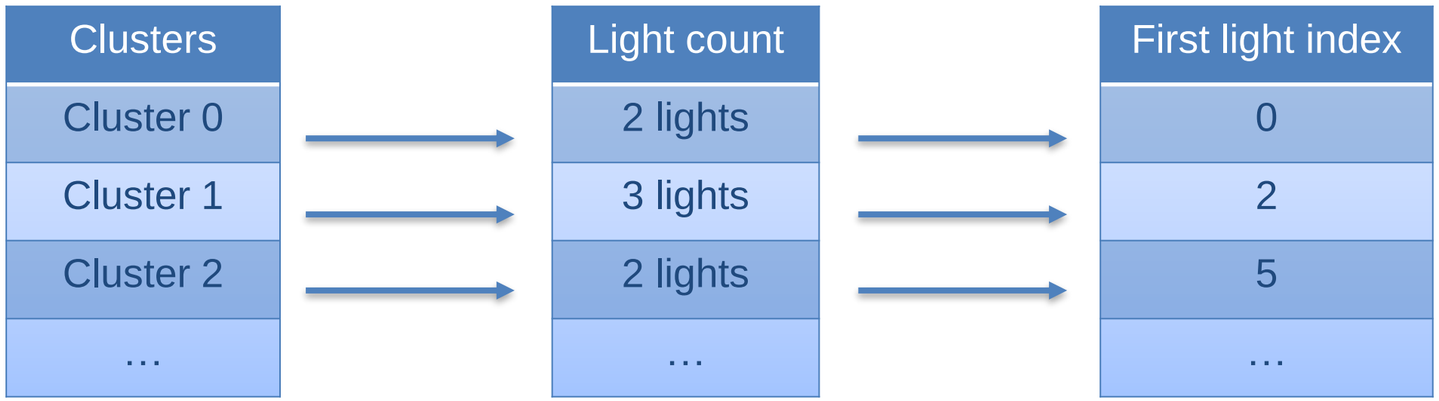
Clusters
Cluster 0
Cluster 1
Cluster 2
…

Clustered forward rendering



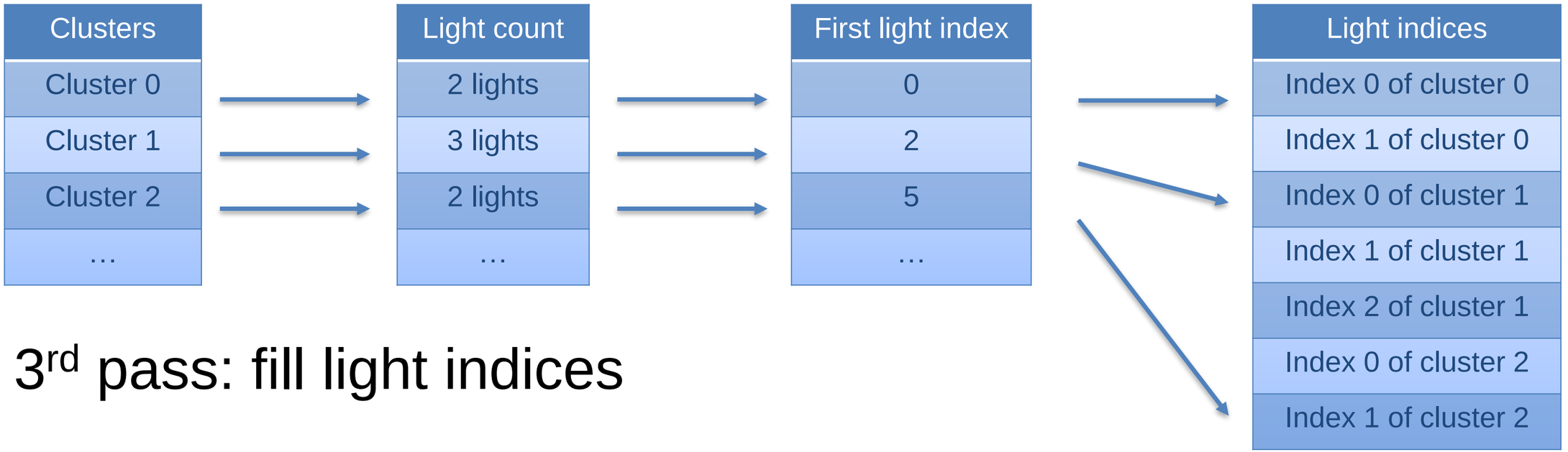
1st pass: compute light count

Clustered forward rendering



2nd pass: compute first light index

Clustered forward rendering



3rd pass: fill light indices

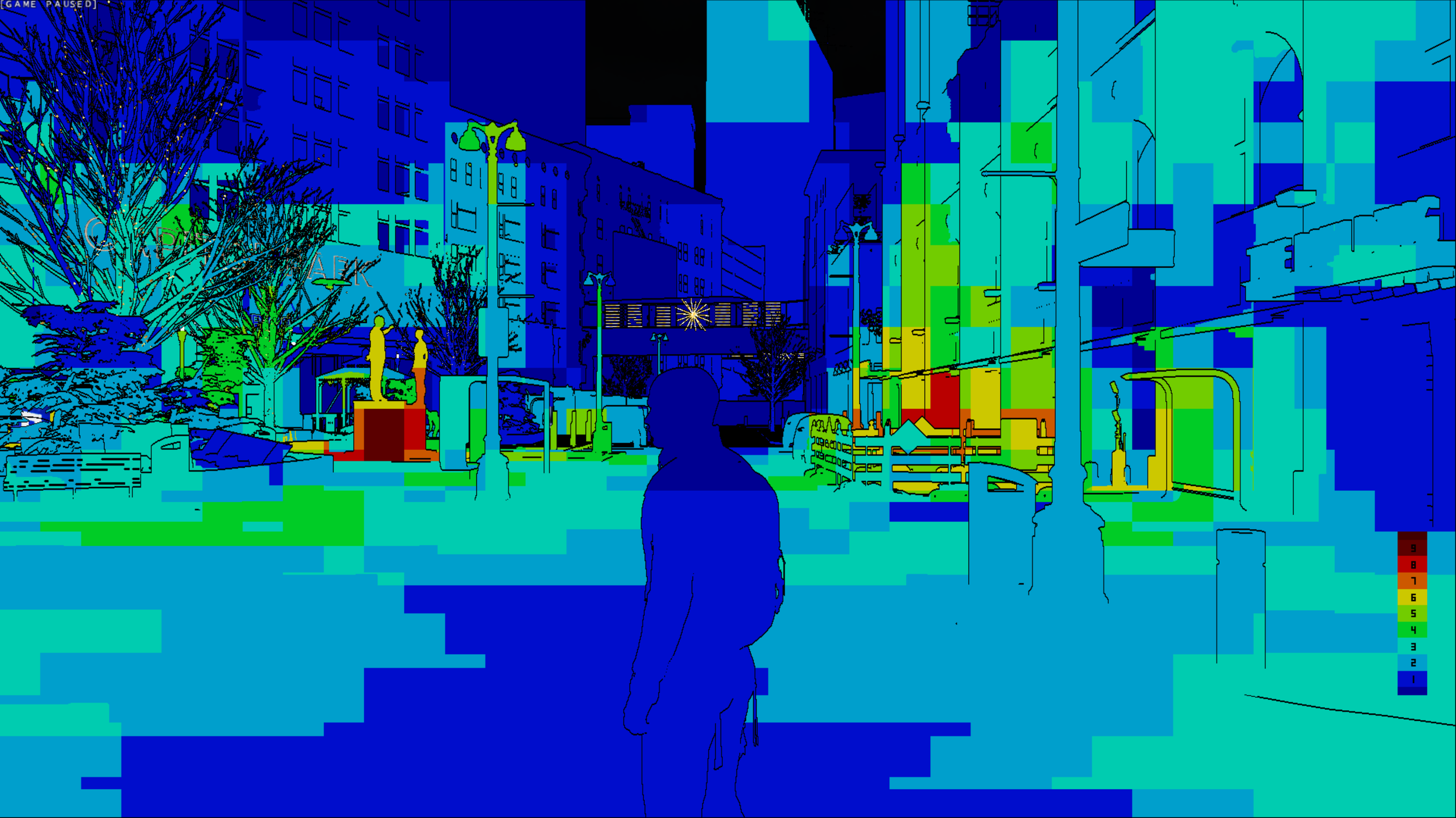
Clustered forward rendering

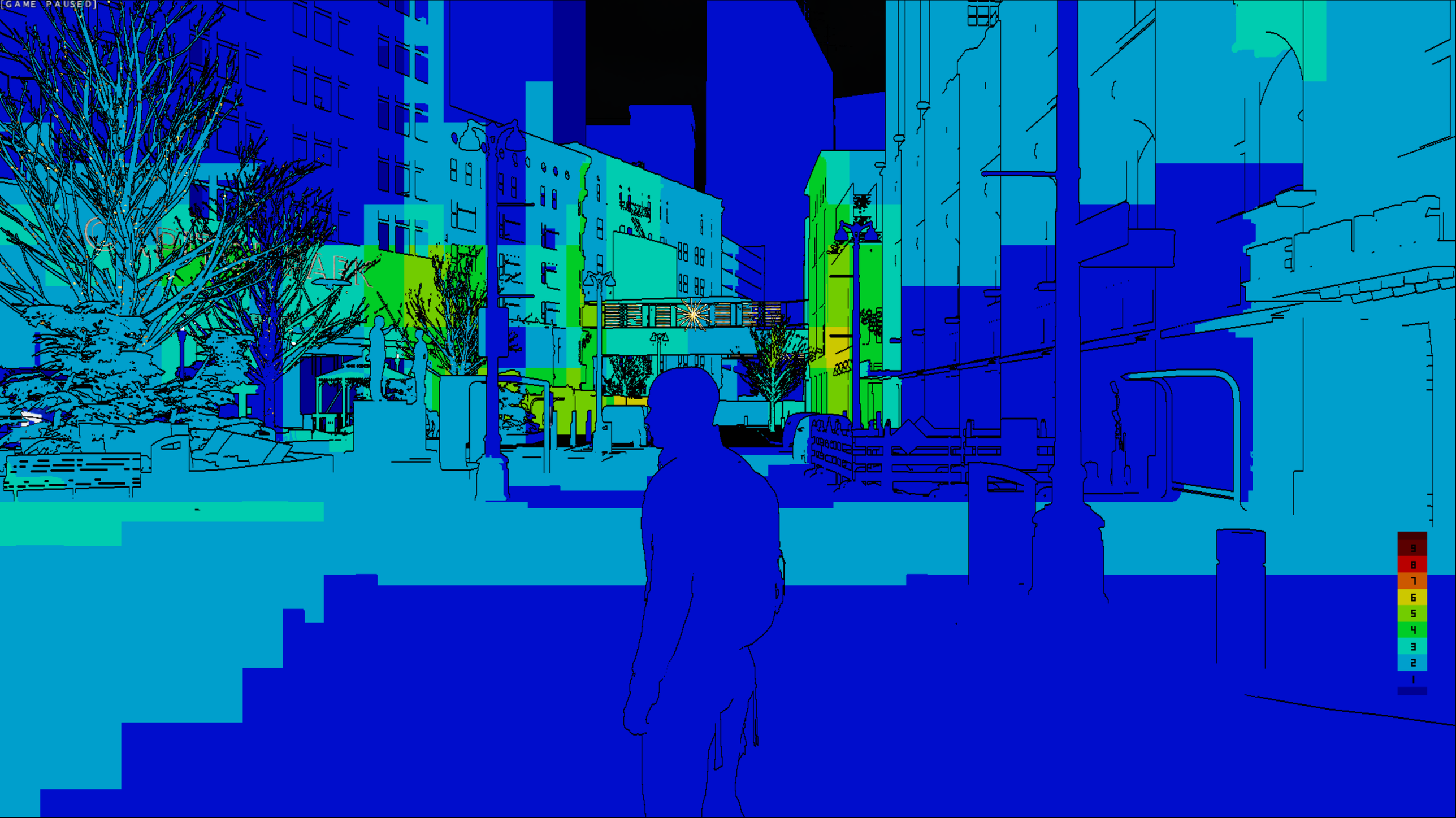
- Fill clusters
  - Filled by asynchronous compute shaders
  - Each cluster is tested with all the lights
  - 3 passes
  - Two hierarchical levels
    - 18x10x32
    - 36x20x64

Clustered forward rendering

- Fill clusters performance
  - 124 lights and 32 Image based lights


[GAME PAUSED]
GET YOURS TODAY!
SIDEWALK CLOSED AHEAD


[GAME PAUSED]
9
8
7
6
5
4
3
2
1


[GAME PAUSED]
9
8
7
6
5
4
3
2
1


QA Tools
GameLogic
Renderer
Kernel
FX
Lighting
Performance
Script
Root/Performance/Render Budgets
Visible Lights Count [200] 156
Directional 0
Projector 48
Spot 52
Point 24
Image Based 32
Visible Decals Count [200] 99
Shadow Map Count Updated [25] 11
Static Shadow Map Count Total 102
Static Shadow Map Count Updated 0
Shadow Atlas Filling % [100] 13
Static Shadow Atlas Filling % [100] 10
Skipped Shadow Map Count [0] 0
Visible Skeletons Count 60
Primitive Count [7K] 4501
Polygones Count [6M] 5293683
Skinning Vertices 482540
Visible Materials 813
GI Probes 17.84Mb
Image Based Lights 123.05Mb
Spawned Rain Particles Count [5K] 0
Spawned Particles Count [5K] 0
Spawned Secondary Count [5K] 0
Integrated Particles Count [100K] 0
Displayed Particles Count [50K] 102
Moved Entities Count [2K] 57
Visibility Manager
Rejected Primitives [0] 0
Missing VRAM [0] 0b
PS4
GET YOURS TODAY!
FPS = 30.0
Frame Time = 33.3
Average = 33.4
CPU = 23.6
Worst = 24.6
Average = 22.7
GPU = 31.7
Worst = 31.9
Average = 31.6

Clustered forward rendering

- Fill clusters results
  - “Night of the long knives” level
  - 124 lights and 32 Image based lights
  - 1.23 milliseconds for clusters filling

Clustered forward rendering

- Lighting
  - Use depth and pixel position to find the cluster for the current pixel
  - Parse the list of lights

Clustered forward rendering

- First results
  - Not very impressive
  - Fat shaders using a lot of registers

Clustered forward rendering

- Optimization
  - Force light loop to use scalar registers instead of vector registers and sort lights
    - “The devil is in the details” by Tiago Sousa and Jean Geffroy

Clustered forward rendering

- Optimization
  - Force light loop to use scalar registers instead of vector registers and sort lights
    - "The devil is in the details" by Tiago Sousa and Jean Geffroy
  - Ensure that everything use the same space as much as possible (view space)

Clustered forward rendering

- Optimization
  - Use less shadow texture samples with TAA (only 8)
  - Force the compiler to use a loop with 2x4 texture shadow samples
  - At some distance, we use a baked shadow texture with only 1 texture shadow sample

Clustered forward rendering

- Optimization
  - Depth pass is necessary

Clustered forward rendering

- Optimization
  - Depth pass is necessary
  - The cluster can be used for per-pixel lighting… and per-vertex lighting!

Clustered forward rendering

- Optimization
  - Depth pass is necessary
  - The cluster can be used for per-pixel lighting… and per-vertex lighting!
  - Image based lighting transferred to a deferred pass when possible

Clustered forward rendering

- Light loop optimization
  - We have 4 types of lights (point, spot, directional and projector)
  - Shadows and projected textures
  - First version use 4 loops (one for each light type)
  - We switched to 1 loop handling all types of lights

Clustered forward rendering

- Light loop optimization
  - For each light
    - Compute light attenuation
    - Compute shadow
    - Compute projected texture
    - Compute final lighting color with material BRDF

Clustered forward rendering

- Light loop optimization
  - For each light
    - Compute light attenuation
    - Compute shadow → Higher register usage for sun shadow
    - Compute projected texture
    - Compute final lighting color with material BRDF

Clustered forward rendering

- Light loop optimization
  - Compute sun shadow → Lower register usage
  - For each light
    - Compute light attenuation
    - Compute shadow
    - Compute projected texture
    - Compute final lighting color with material BRDF

Clustered forward rendering

- Visibility light rejection
  - Our visibility is portal/zone based
  - Visibility information can be used to reject a light as soon as possible
  - Visibility information is stored in a bit field (one bit per zone)
  - We can reject a light if uiObjectVisibility & uiLightVisibility != 0

Clustered forward rendering

- Light loop optimization
  - Compute sun shadow → Lower register usage
  - For each light
    - Visibility test bit field → Early exit
    - Compute light attenuation
    - Compute shadow
    - Compute projected texture
    - Compute final lighting color with material BRDF

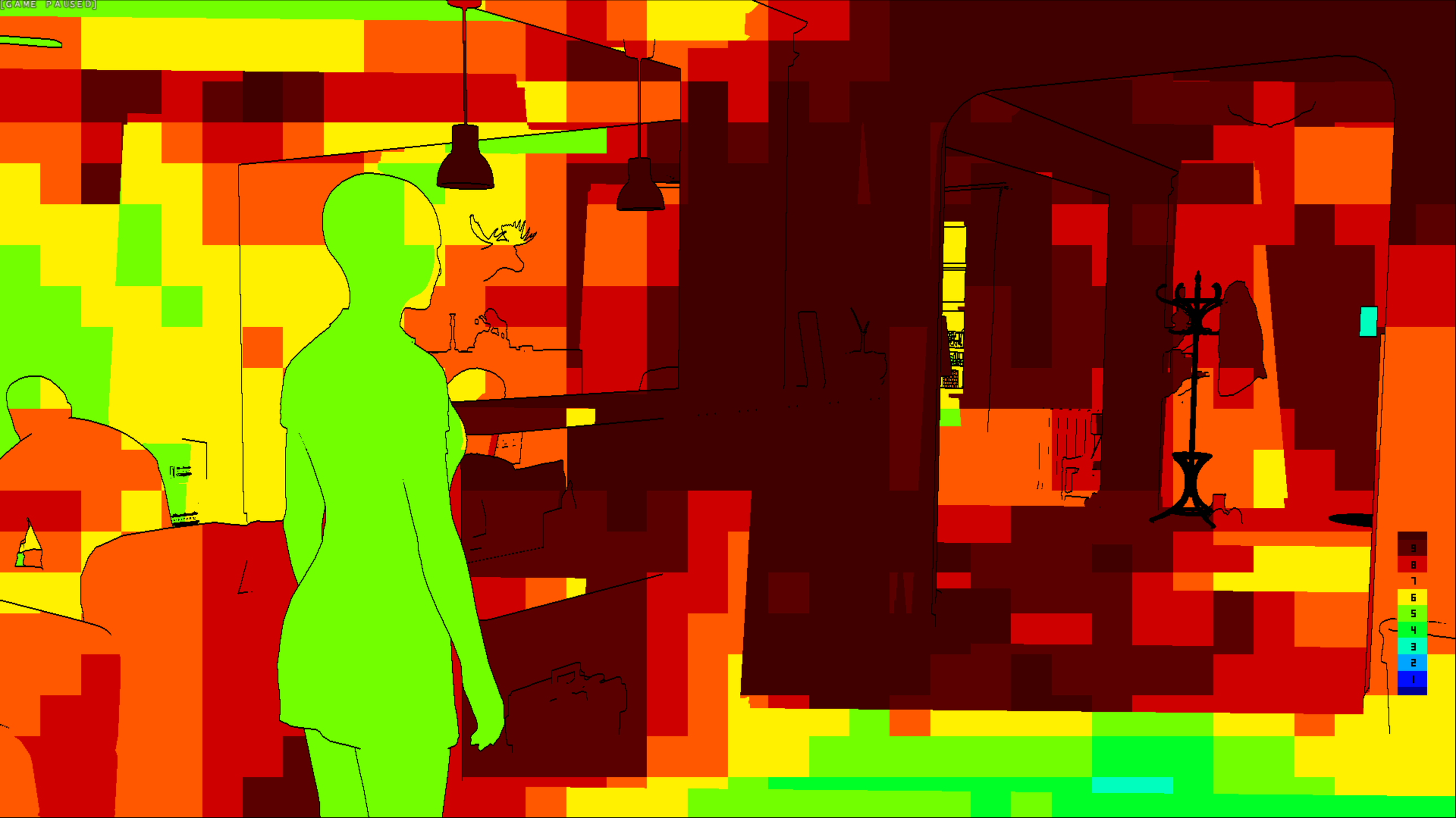
Clustered forward rendering

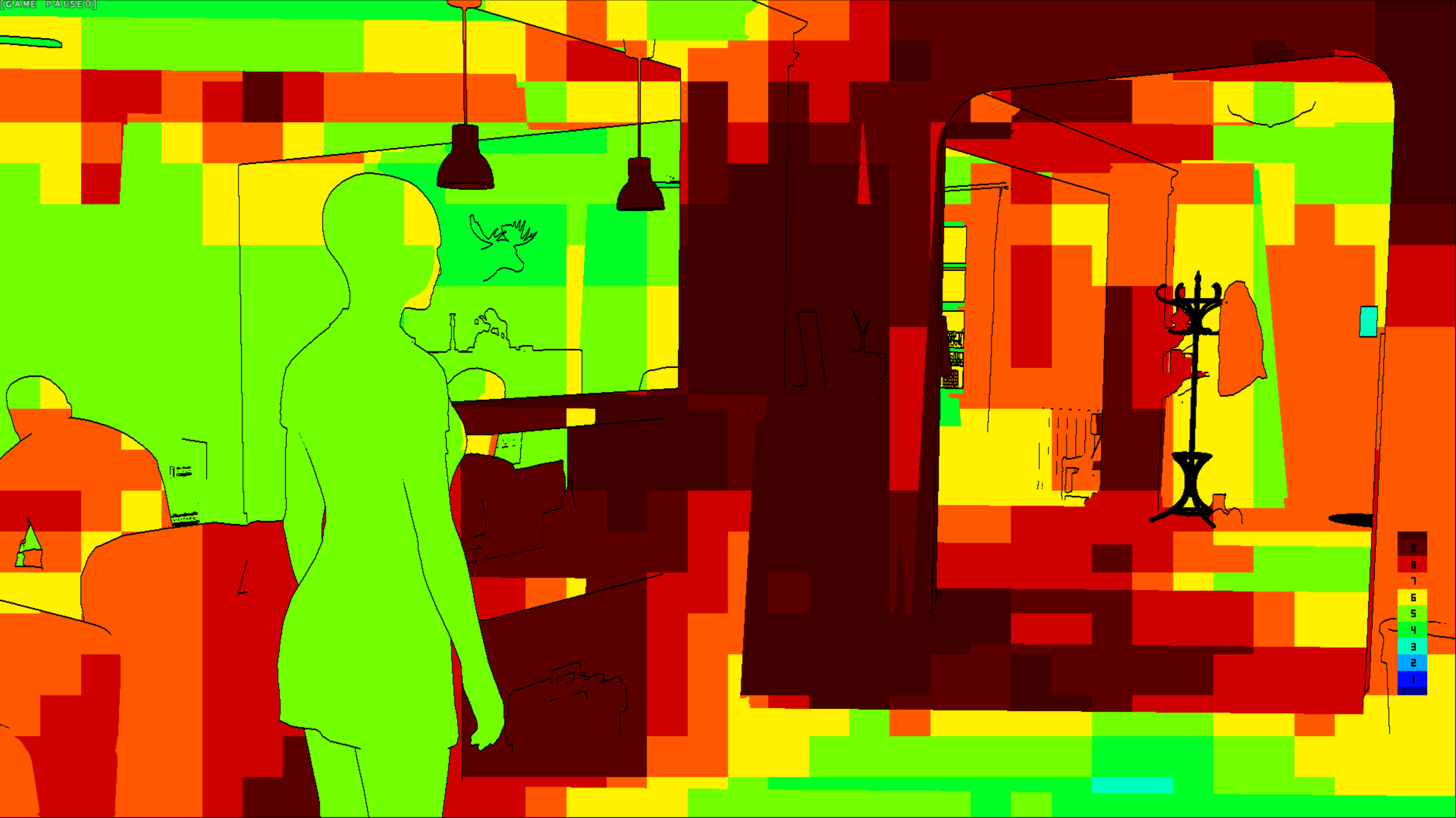
- Other possible early exits
  - N dot L
  - Light attenuation result
  - Shadow result

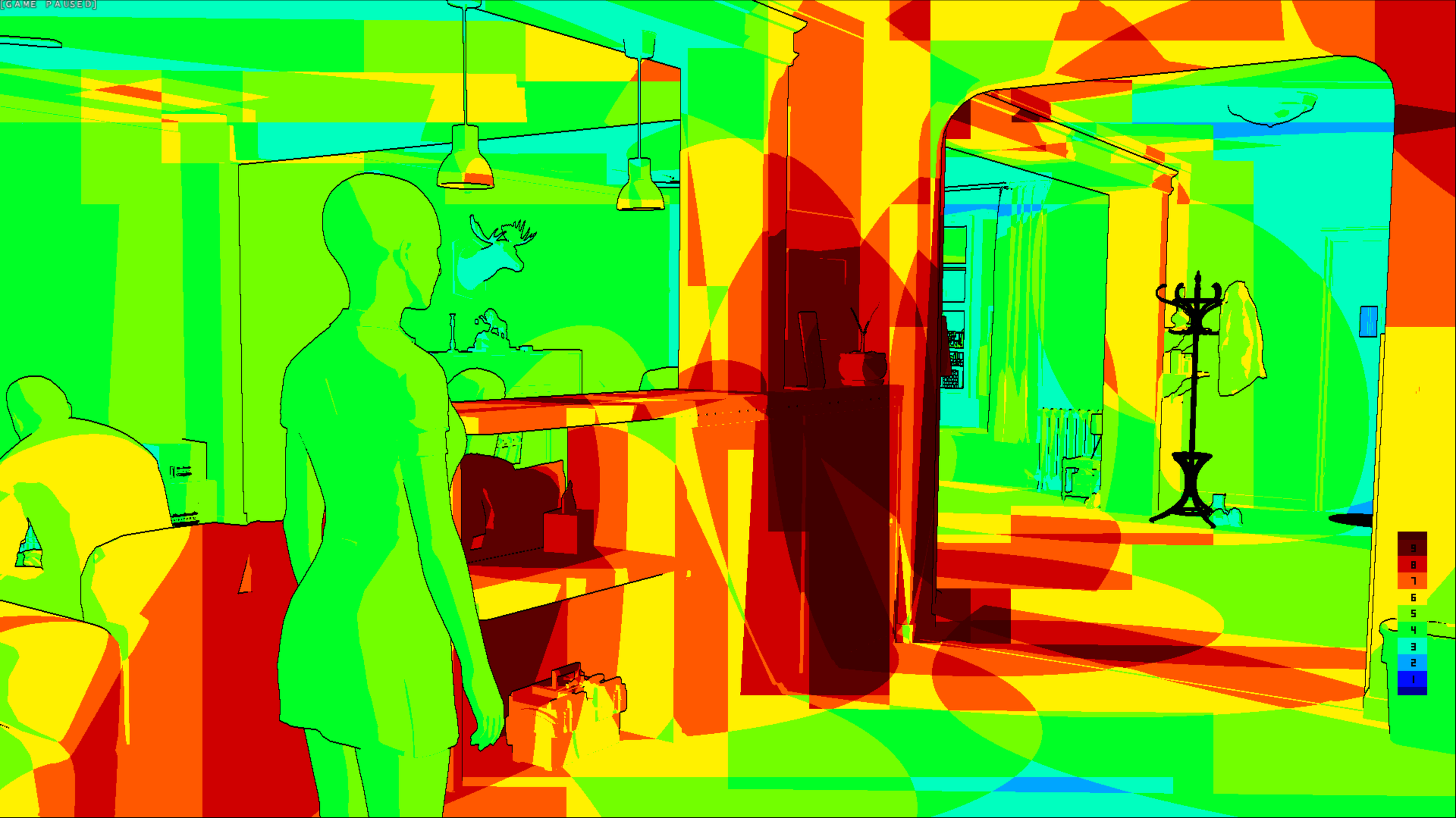
Clustered forward rendering

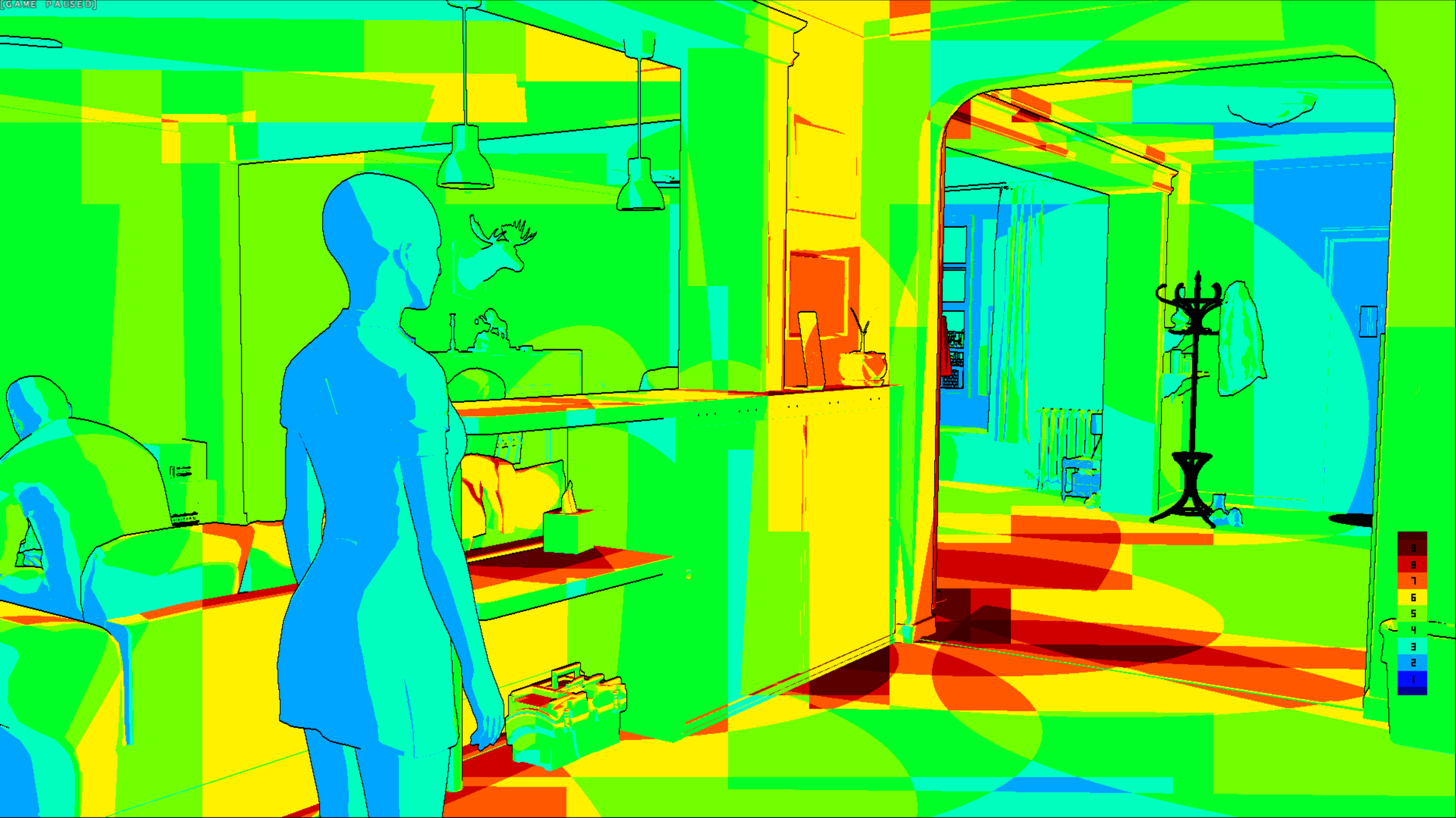
- Light loop optimization
  - Compute sun shadow → Lower register usage
  - For each light
    - Visibility Test bit field → Early exit
    - Compute light attenuation → Early exit
    - Test N dot L → Early exit
    - Compute shadow → Early exit
    - Compute projected texture
    - Compute final lighting color with material BRDF

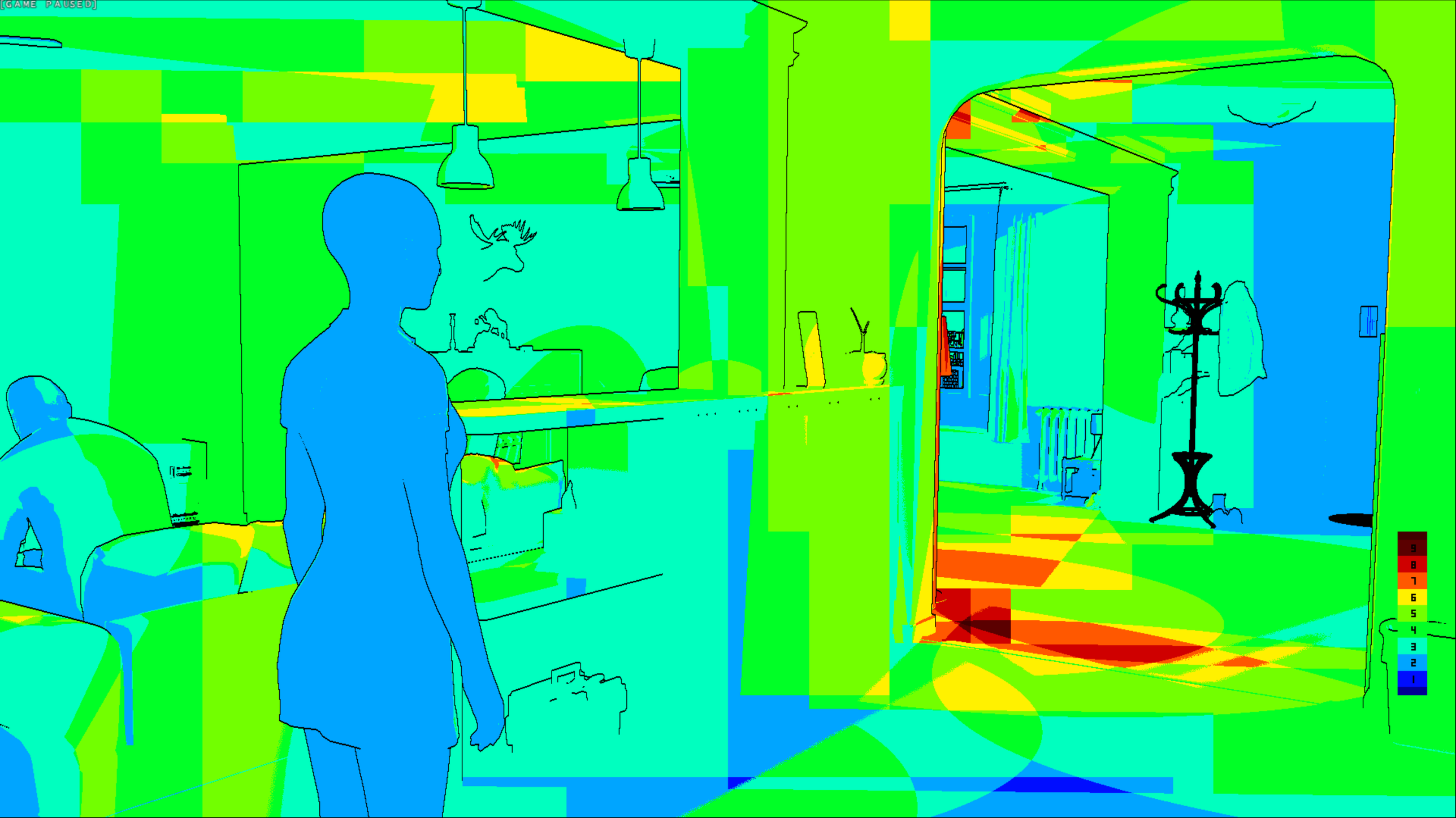

[GAME PAUSED]
KARA
ANDROID


[GAME PAUSED]
9
8
7
6
5
4
3
2
1


[GAME PAUSED]
9
8
7
6
5
4
3
2
1


[GAME PAUSED]
9
8
7
6
5
4
3
2
1


[GAME PAUSED]
9
8
7
6
5
4
3
2
1


[GAME PAUSED]
9
8
7
6
5
4
3
2
1

Clustered forward rendering

- Transparency optimization
  - Transparency can be a performance killer

Clustered forward rendering

- Transparency optimization
  - Transparency can be a performance killer
  - Glass
    - Image based lighting only

Clustered forward rendering

- Transparency optimization
  - Transparency can be a performance killer
  - Glass
    - Image based lighting only
  - Particles:
    - Per centroid
    - Spherical Harmonics
    - Half resolution

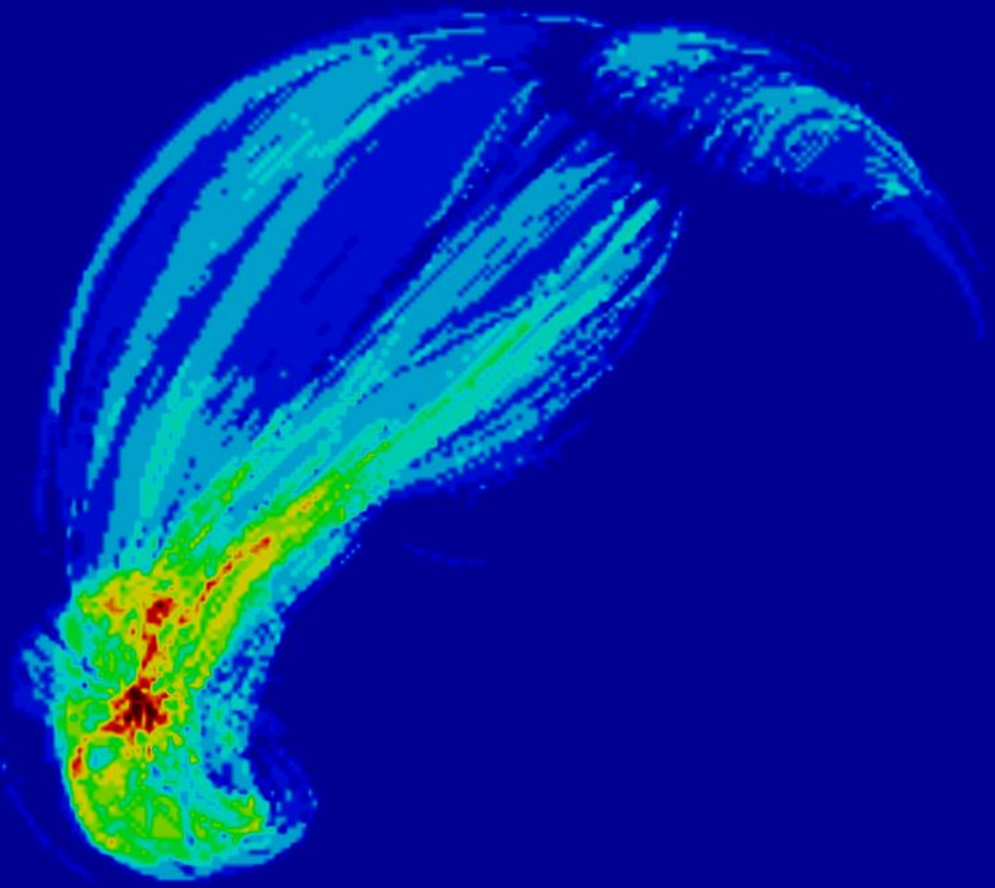
Clustered forward rendering

- Hairs
  - Performance issues with fully transparent hairs

Clustered forward rendering

- Hairs
  - Transparency accumulation pass
    - Additive blending
    - Output tweaked alpha transparency
    - 1/16 resolution

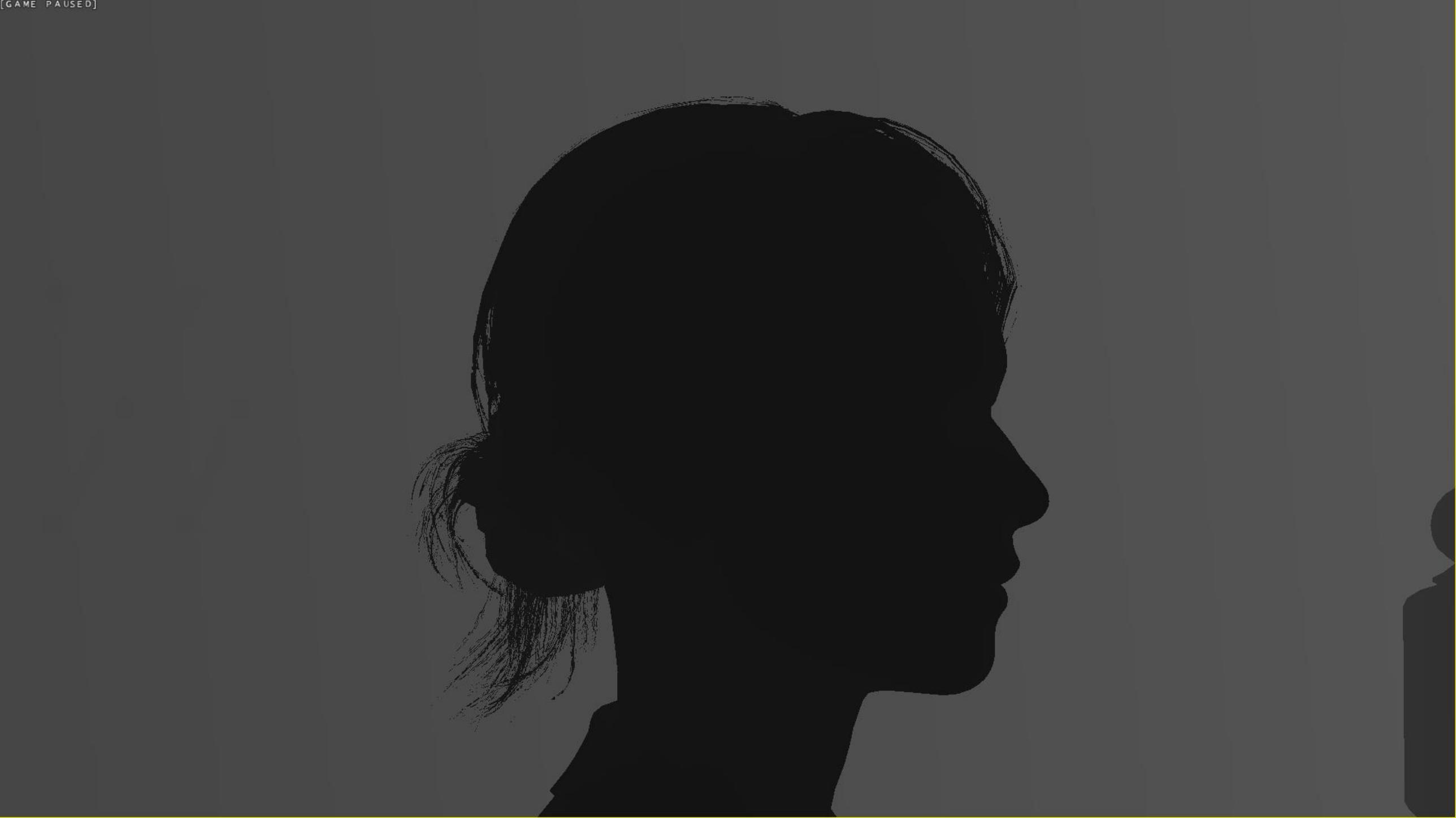
[GAME PAUSED]



Clustered forward rendering

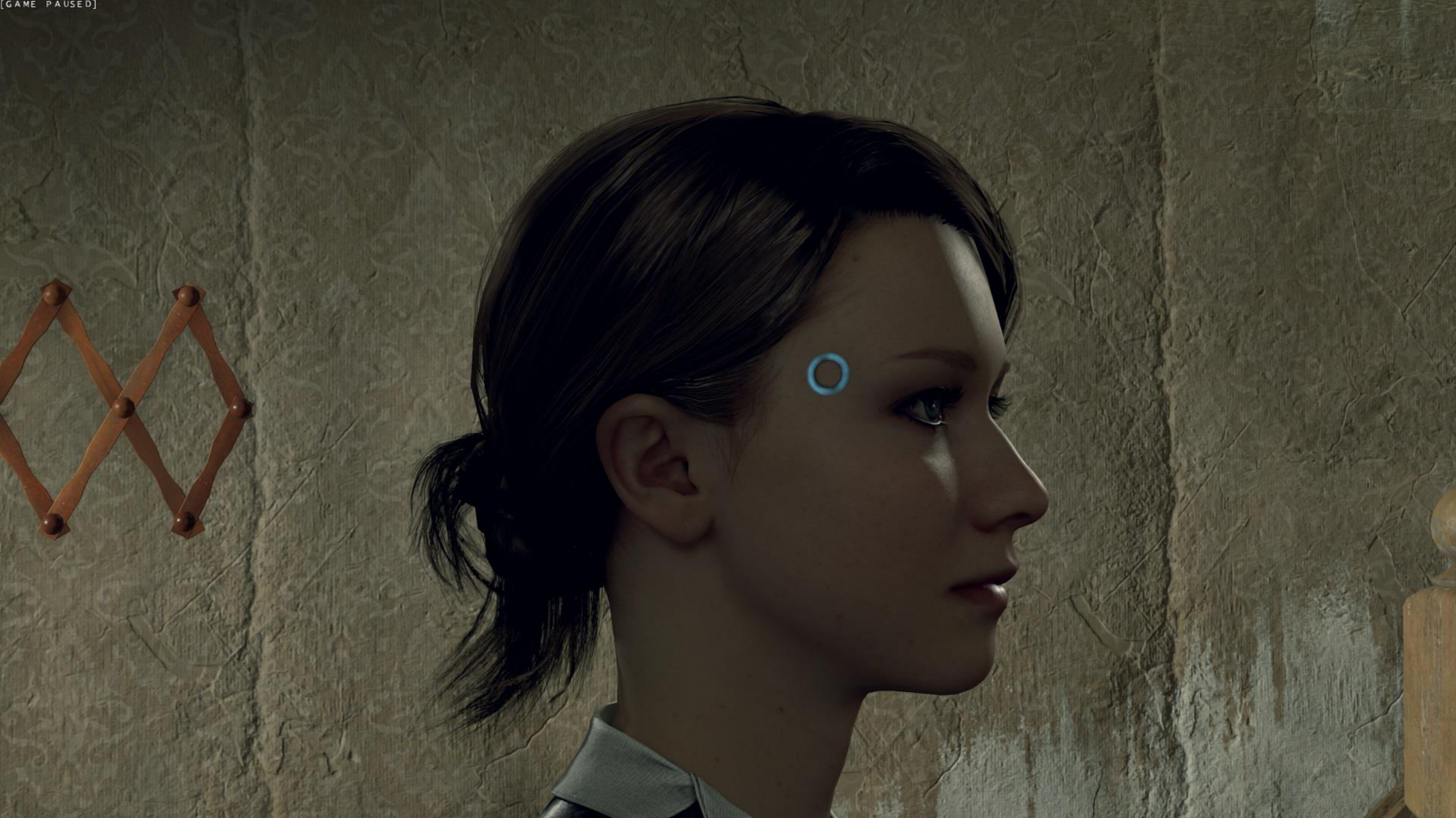
- Hairs
  - Transparency accumulation pass
  - Depth pass
    - Alpha test computed from transparency accumulation pass

[GAME PAUSED]



Clustered forward rendering

- Hairs
  - Transparency accumulation pass
  - Depth pass
  - Back triangles pass
  - Forward triangle pass
  - Motion vector pass


[GAME PAUSED]

Clustered forward rendering

- Things still deferred in our engine
  - Screen Space Reflection
  - Image based lighting
  - Both need normal and roughness

Clustered forward rendering

- Debug
  - With deferred shading, the G-Buffer is very useful for debugging
    - Normal, roughness, albedo, etc.
  - Debug shader
    - Output anything: Normal, Tangent, Binormal, uvs, etc.
    - More powerful than G-Buffer render targets
    - Stored in debug memory. Not used in retail version of the game.

Clustered forward rendering

- Mirrors
  - Fill the clusters once for each visible mirror

Clustered forward rendering

- Other advantages
  - Volumetric lighting

Clustered forward rendering

- Other advantages
  - Volumetric lighting
  - Decals

Clustered forward rendering

- Lighting performance


[GAME PAUSED]
GET YOURS TODAY!
THE BUILDING
SIDEWALK CLOSED AHEAD

Clustered forward rendering

- Lighting performance
  - 124 lights and 32 Image based lights
  - 1.23 ms for cluster filling
  - 1.92 ms for depth pass
  - 8.79 ms for opaque pass
  - 3.48 ms for transparent pass

Clustered forward rendering

- Future
  - Reduce number of passes for clusters filling

Clustered forward rendering

- Future
  - Reduce number of passes for clusters filling
  - Better depth distribution

Clustered forward rendering

- Future
  - Reduce number of passes for clusters filling
  - Better depth distribution
  - Remove some lights during the cluster filling
    - With shadow maps
    - With visibility information

Clustered forward rendering

- Future
  - Reduce number of passes for clusters filling
  - Better depth distribution
  - Remove some lights during the cluster filling
  - VR: fill the clusters once for both eyes

Temporal anti-aliasing

Temporal anti-aliasing

- Aliasing
  - Can’t only rely on better resolution
  - HDR and PBR increase aliasing

Temporal anti-aliasing

- Anti-aliasing
  - Multi-sampling
  - Post-processing
  - Shading

Temporal anti-aliasing

- Multi-sampling
  - Hardware
  - Increase size of render target (2X, 4X, etc.)
  - Shading is performed more at polygons edges
  - Decrease performance
  - Doesn’t improve specular aliasing

Temporal anti-aliasing

- Post-processing
  - Morphological Anti-Aliasing (MLAA)
  - Fast Approximate Anti-Aliasing (FXAA)

Temporal anti-aliasing

- Post-processing
  - Morphological Anti-Aliasing (MLAA)
  - Fast Approximate Anti-Aliasing (FXAA)
  - Temporal Anti-Aliasing (TAA)

Temporal anti-aliasing

- Post-processing
  - Morphological Anti-Aliasing (MLAA)
  - Fast Approximate Anti-Aliasing (FXAA)
  - Temporal Anti-Aliasing (TAA)
    - Based on “High Quality Temporal Supersampling” by *Bryan Karis*
    - Unreal infiltrator real-time demo

Temporal anti-aliasing

- TAA
  - Jitter each frame with a different offset

Temporal anti-aliasing

- TAA
  - Jitter each frame with a different offset
  - Accumulate frames

Temporal anti-aliasing

- TAA
  - Jitter each frame with a different offset
  - Accumulate frames
  - Use motion vectors to retrieve previous pixel position

Temporal anti-aliasing

- TAA
  - Jitter each frame with a different offset
  - Accumulate frames
  - Use motion vectors to retrieve previous pixel position
  - Use heuristic to reject previous frame pixels

Temporal anti-aliasing

- Where to apply TAA
  - Final image
    - Doesn’t prevent from Bloom, DOF and motion blur aliasing
  - Before post-processing
    - Best for stability
  - For specific features
    - SSR, Volumetric lighting

Temporal anti-aliasing

- Jittering
  - Fixed 8 taps
    - Like for 8x MSAA

Temporal anti-aliasing

- Motion vectors
  - Can be written in option on transparent objects

Temporal anti-aliasing

- Motion vectors
  - Can be written in option on transparent objects
  - Clothes, skinned characters

Temporal anti-aliasing

- Motion vectors
  - Can be written in option on transparent objects
  - Clothes, skinned characters
  - Vegetation (Speedtree)

Temporal anti-aliasing

- Motion vectors
  - Can be written in option on transparent objects
  - Clothes, skinned characters
  - Vegetation (Speedtree)
  - Vertex animation

Temporal anti-aliasing

- Pixel rejection
  - Neighborhood clamping
    - Inspired by *Bryan Karis* presentation

Temporal anti-aliasing

- Pixel rejection
  - Neighborhood clamping
    - Inspired by *Bryan Karis* presentation
  - Depth disocclusion
  - Velocity similarity

Temporal anti-aliasing

- Skin shader
  - Decrease TAA strength when camera zoom in/zoom out

Temporal anti-aliasing

- Performance
  - 1.14 ms

Temporal anti-aliasing

- Rain and snow
  - Rain and snow can disappear completely with TAA
  - The flag responsive fix missing particles
  - For rain surface effects, decrease TAA strength according to rain normal strength


TAA Off
FreeCam Locked
R1+R2 to toggle


TAA On
FreeCam Locked
Ri+R2 to toggle


TAA decreased on RAIN
FreeCam Locked
R1+B2 to toggle


TAA Strength
FreeCam Locked
R1+R2 to toggle

Temporal anti-aliasing

- Issues with TAA
  - Some post-processes don’t work well with TAA
  - Depth is jittered
  - Contour pixels are anti-aliased
  - Leaking and vibration on DOF and motion blur

Temporal anti-aliasing

- Depth of field
  - Removing vibrations
    - We perform TAA on depth
  - Removing leaking
    - We erode the Circle of confusion to avoid leaking


[GAME PAUSED]


[GAME PAUSED]

Temporal anti-aliasing

- Motion blur
  - Half resolution motion blur (540P)
  - 2 passes with 8 texture fetches each
  - TAA bleeds on pixels near depth discontinuities
  - Reject these pixels during Motion Blur sampling
  - => Avoids unwanted TAA bleed *streaks*


[GAME PAUSED]
ANDROID
RK800


[GAME PAUSED]
ANDROID
RK800
TAA bleed streaks

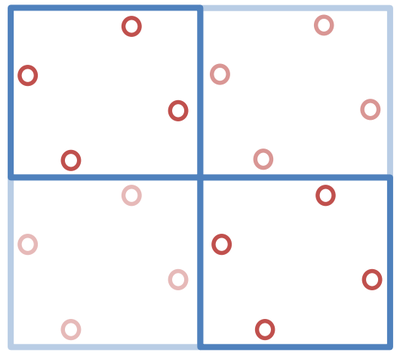

[GAME PAUSED]
ANDROID
RK800

Temporal anti-aliasing

- PS4 Pro considerations
  - Temporal anti-aliasing is compatible with checkerboard

Temporal anti-aliasing

- PS4 Pro considerations
  - Temporal anti-aliasing is compatible with checkerboard
  - Checkerboard should be resolved with temporal AA
    - Jittering split between checkerboard pixels

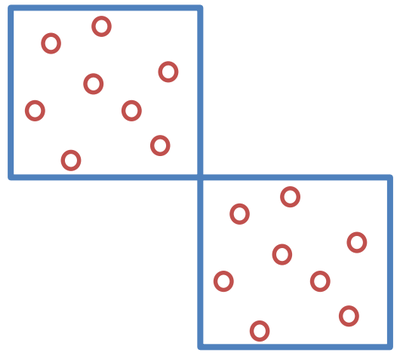


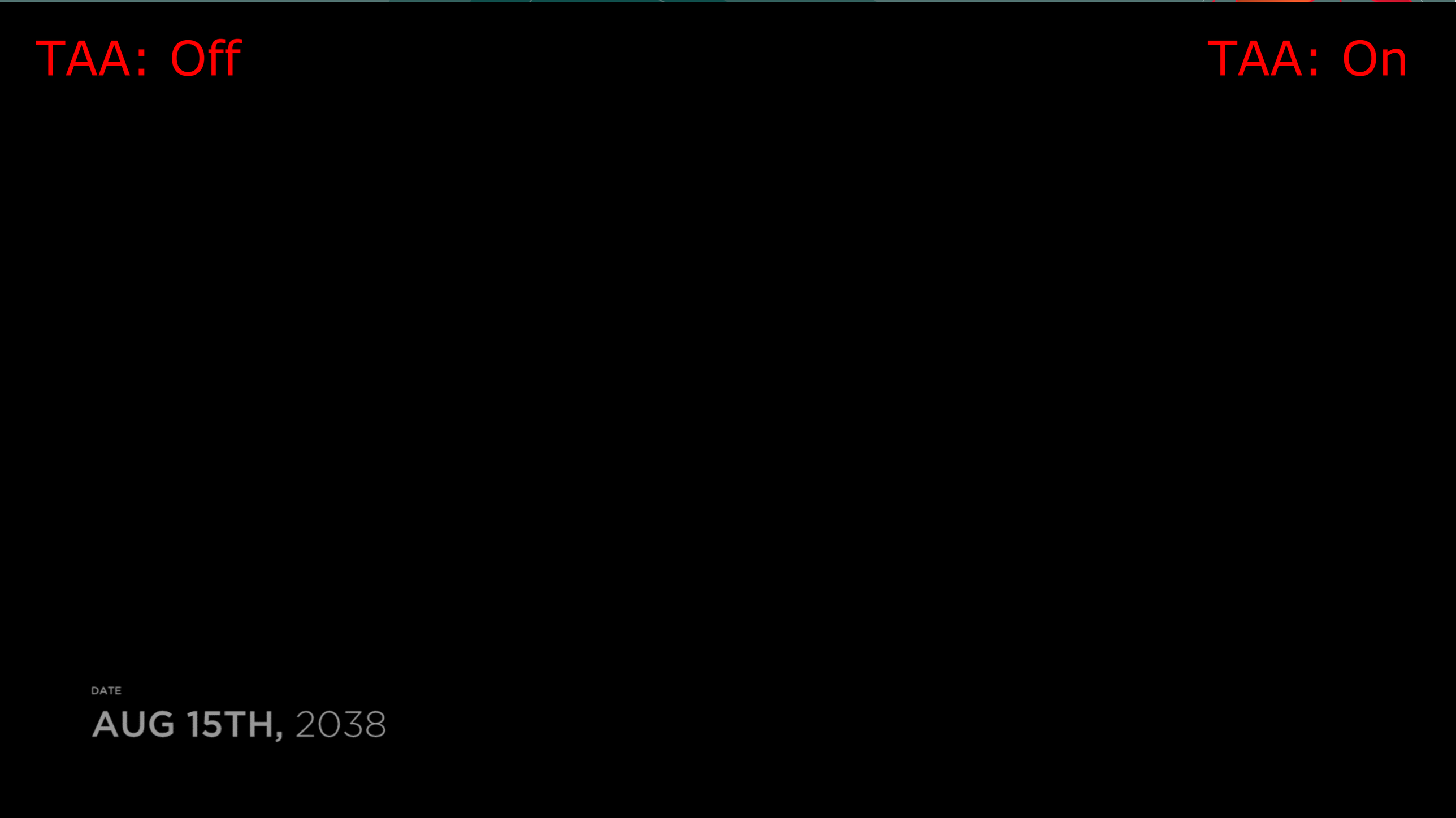
Temporal anti-aliasing

- PS4 Pro considerations
  - Temporal anti-aliasing is compatible with checkerboard
  - Checkerboard should be resolved with temporal AA
  - But 4K ruins post-processing performance

Temporal anti-aliasing

- PS4 Pro considerations
  - Temporal anti-aliasing is compatible with checkerboard
  - Checkerboard should be resolved with temporal AA
  - But 4K ruins post-processing performance
  - We resolve checkerboard after post-processing




TAA: Off
TAA: On
DATE
AUG 15TH, 2038

Temporal anti-aliasing

- TAA is not enough in some situations
  - We are only 8x
  - Very high specular impacts on isolated pixels can still be an issue

Temporal anti-aliasing

- Shading anti-aliasing
  - We can perform shading more than once, using GPU derivatives
  - Good results, but costly

Temporal anti-aliasing

- Shading anti-aliasing
  - Normal Distribution Function (NDF) filtering
    - “Filtering Distributions of Normals for Shading Antialiasing” by A.S. Kaplanyan, S. Hill, A. Patney and A. Lefohn

Temporal anti-aliasing

- Shading anti-aliasing
  - Normal Distribution Function (NDF) filtering
    - Faster version: “Error Reduction and Simplification for Shading Anti-Aliasing” by Yusuke Tokuyoshi

Temporal anti-aliasing

- Shading anti-aliasing
  - Normal Distribution Function (NDF) filtering
    - Works very well with TAA
    - Rain details are more visible


TAA Off
quantic dream


TAA ON
quantic dream


TAA On + NDF Filtering
quantic dream

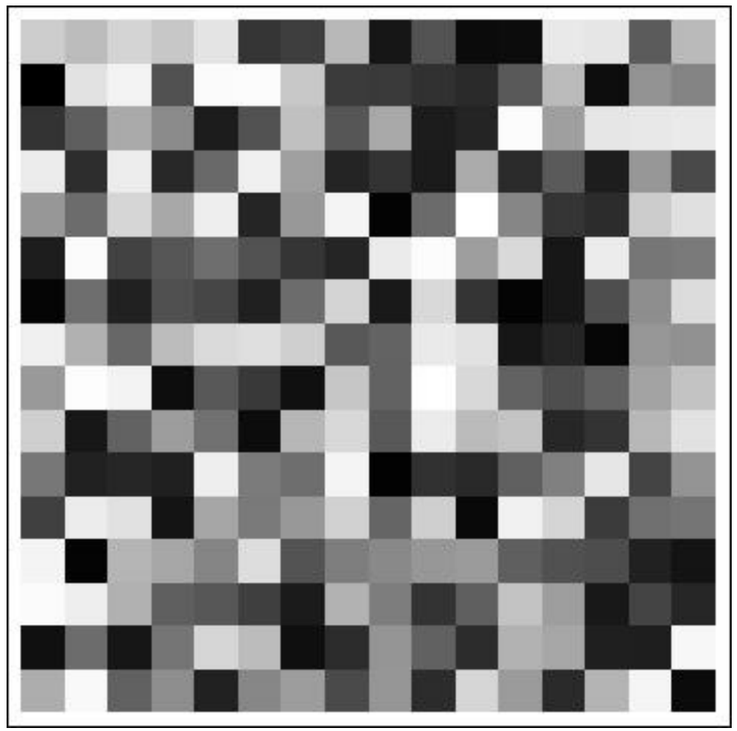
Temporal anti-aliasing

- Other temporal effects
  - Shadows
  - HBAO
  - SSR
  - Skin Screen Space Subsurface Scattering
  - Volumetric lighting

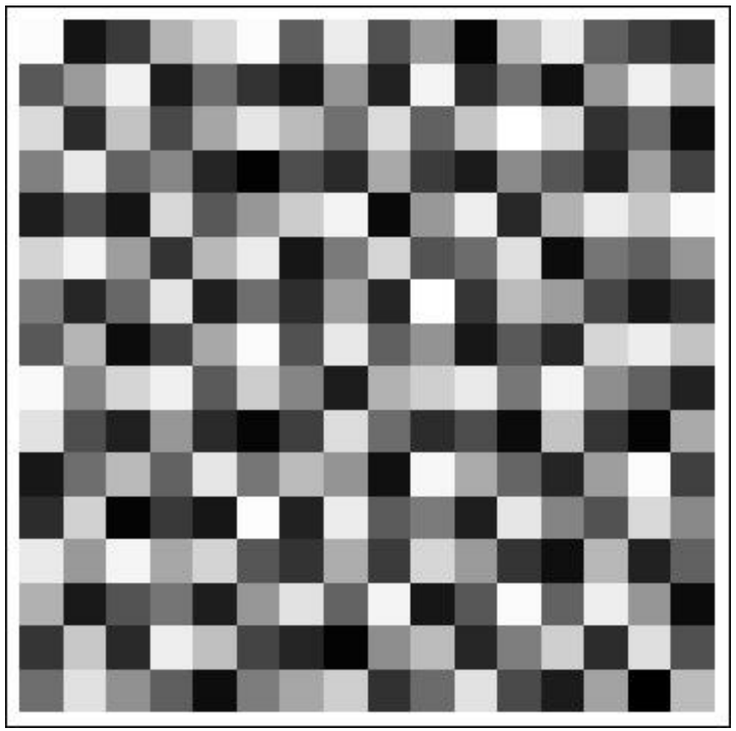
Temporal anti-aliasing

- Blue noise
  - Noise with minimal low frequency components and no concentrated spikes in energy
  - “The rendering of inside” by Mikkel Gjel & Mikkel Svendsen
  - Blue Noise Generator by Bart Wronski

Blue noise

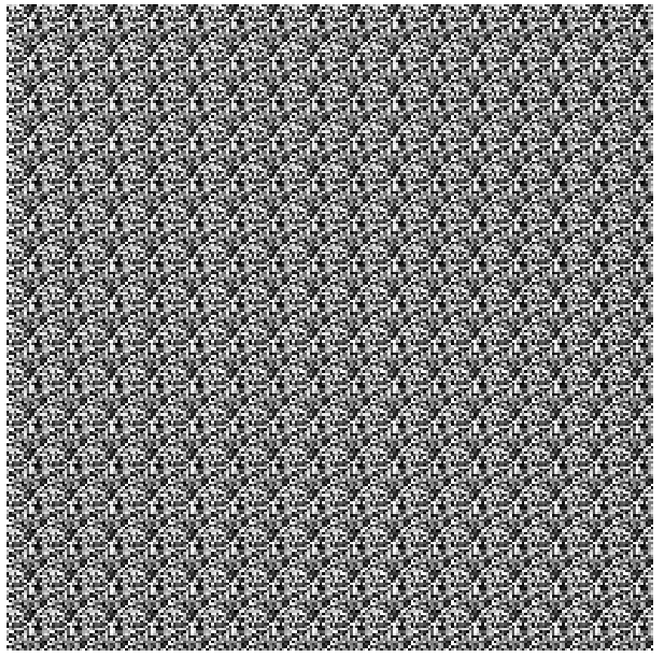


White noise

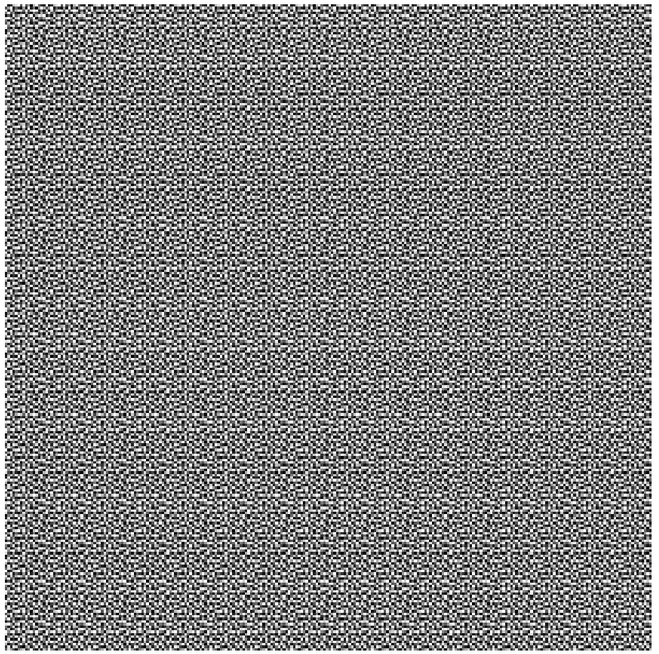


Blue noise

Blue noise (with tiling)



White noise



Blue noise

Temporal anti-aliasing

- Temporal Shadows
  - Use Poisson 8 taps kernel rotated every frame
  - We use a different rotation per pixel with a blue noise 16x16 2D texture
  - The result is smoothed by TAA

Temporal anti-aliasing

```
const float fAARotation = pPassSRT->_fAARotation; // from 1 to 8. Change at each frame.
const float fScale = 1.f/4.f + 1.f/8.f;
float fRand = tex2Dfetch(BlueNoiseTexture, int2(sSurface.fFragCoord.xy) % 16 , 0).x;
float fAngle = 2.0f * PI * (fRand + fAARotation * fScale);
```


[GAME PAUSED]
420


TAA Off


TAA On

Temporal anti-aliasing

- Temporal Screen Space Sub Surface Scattering
  - Cross blur filter in 2 passes (7 taps)
  - Each pass is rotated at each frame
  - Rotation depend on pixel position and frame ID
  - Use 3D blue noise 128 x 128 x 8 texture
  - The result will be smoothed by TAA

Temporal anti-aliasing

```
float fRand = tex3Dfetch(pConstantData->rBlueNoise,
                  int3(screenCoord%128, pConstantData->vFrameID.x),0).x;
float fAngle= fRand * TWO_PI ;
```


[GAME PAUSED]
SSSSS Off


[GAME PAUSED]
SSSSS On


TAA Off


TAA On

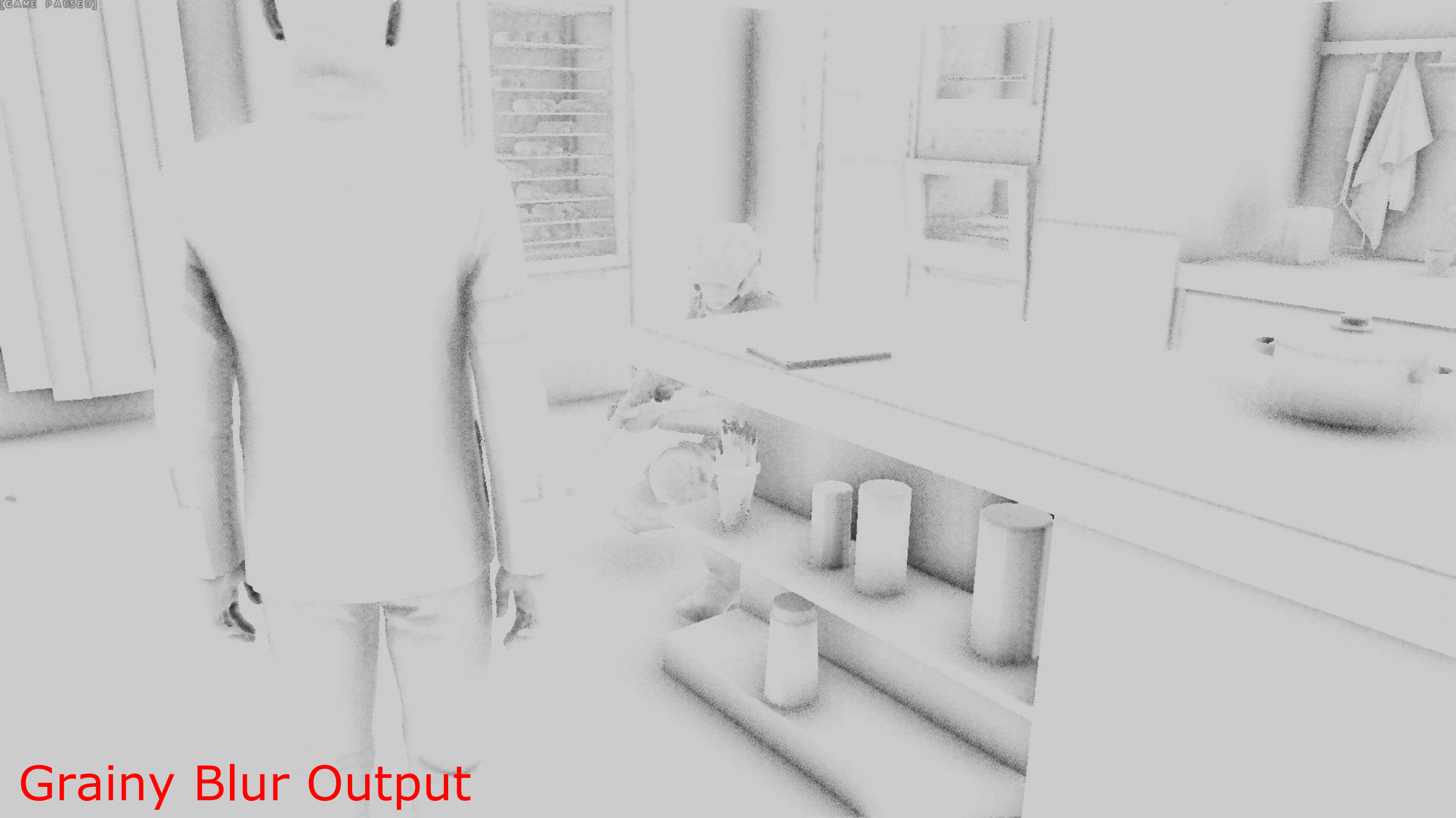
Temporal anti-aliasing

- Temporal SSAO
  - Based on Horizon Based Ambient Occlusion (HBAO)
    - "Image-Space Horizon-Based Ambient Occlusion" by *Louis Bavoil*, *Miguel Sainz* and *Rouslan Dimitrov*
  - Full resolution (1080P)
  - 2 steps and 2 directions
  - The directions are turned each frame

Temporal anti-aliasing

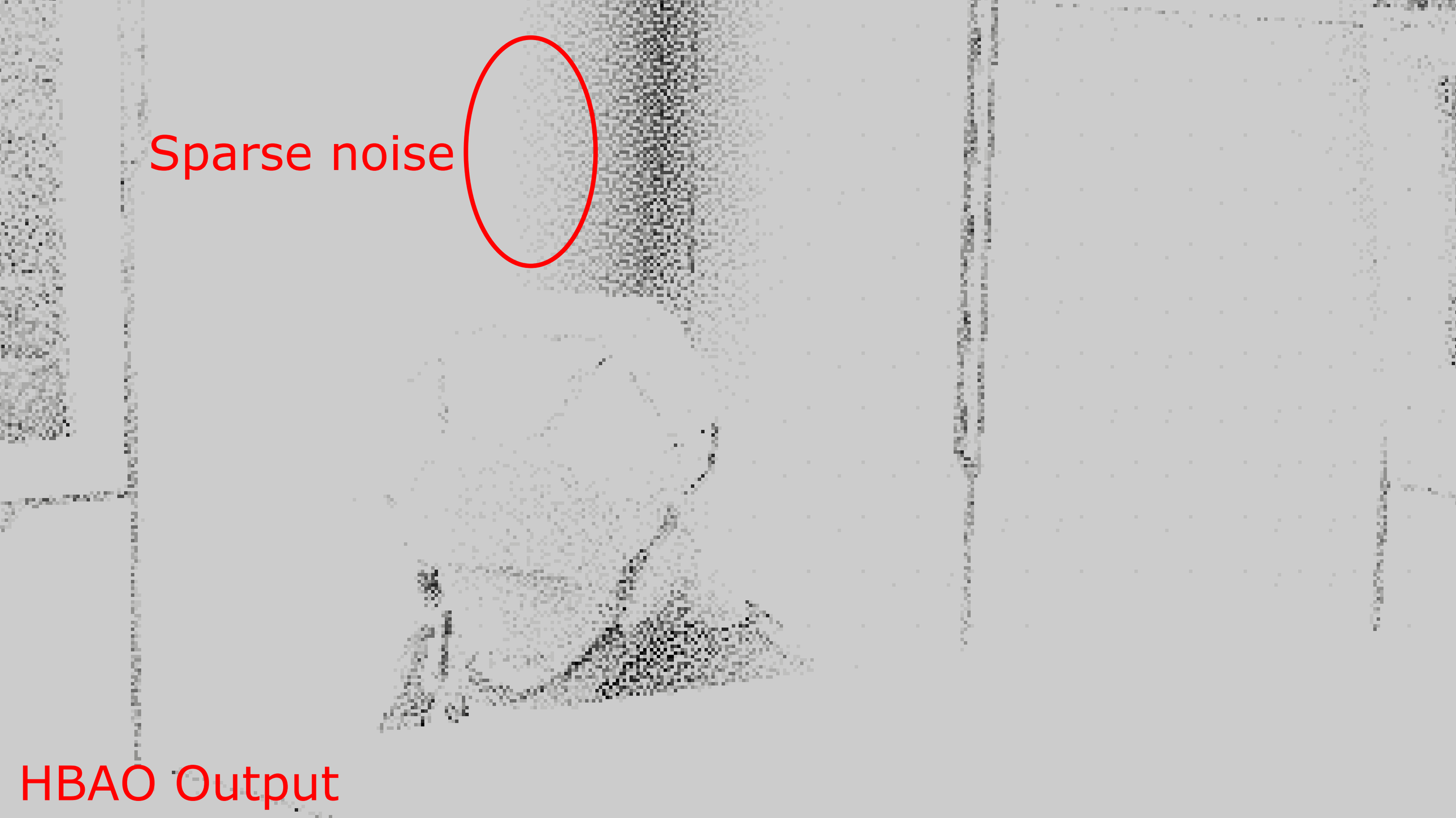
- Temporal SSAO
  - HBAO result can't be smoothed by TAA ("Sparse" noise)
  - "We use a "grainy" blur
  - HBAO: 0.85 ms
  - "Grainy" blur: 0.32 ms


[GAME PAUSED]
HBAO Output



Grainy Blur Output


HBAO Output


Sparse noise
HBAO Output

Grainy Blur Output


[GAME PAUSED]
ANDROID
RK800
SSAO Off


[GAME PAUSED]
ANDROID
RK800
SSAO On

Temporal anti-aliasing

- Temporal Screen Space Reflection
  - "Stochastic Screen-Space Reflections" by *Tomasz Stachowiak* (Frostbite)
  - "Screen Space Reflections in "The Surge"" by *Michele Giacalone*

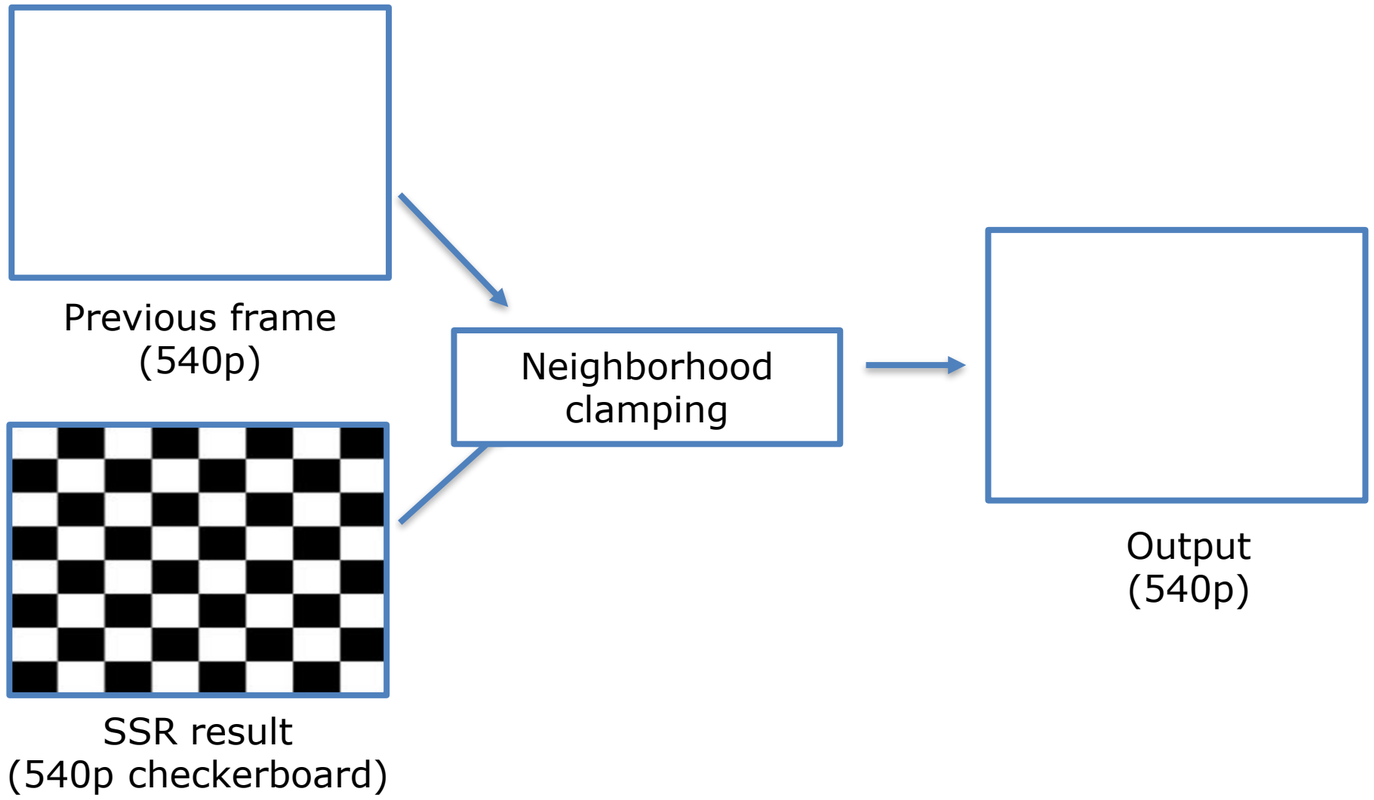
Temporal anti-aliasing

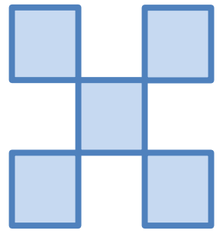
- Temporal Screen Space Reflection
  - Physically based
  - Half resolution with checkerboard
  - To avoid smearing, we use motion vectors at rays intersection points

Temporal anti-aliasing

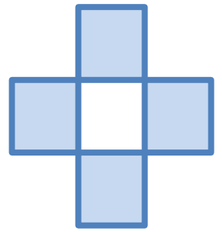
- Temporal Screen Space Reflection
  - Own TAA pass
    - Use neighborhood clamping with checkerboard.

Checkerboard neighbor clamping





Active pixel: clamp previous pixel with 5 current pixels and accumulate for TAA

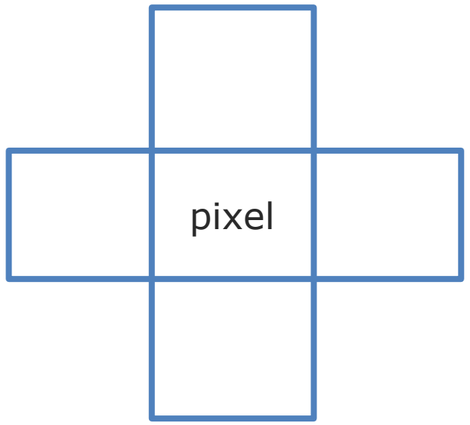


Inactive pixel: clamp previous pixel with 4 current pixels

Temporal anti-aliasing

- Temporal SSR
  - Own TAA pass
    - Use neighbor clamping
  - Upsampling
    - The 2x2 noise of SSR (because of half resolution) will break main TAA
    - We must change the noise from 2x2 pixels to 1x1 pixel
    - We feel it is less blurry than Frostbite version

SSR Upsampling

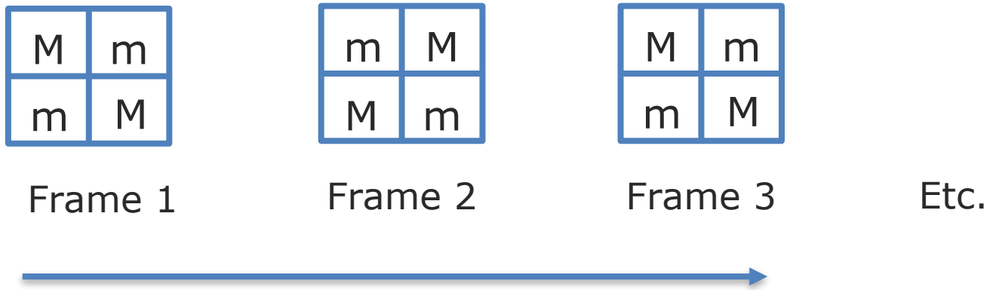


We compute min and max values from 5 half resolution samples

m = lerp(pixel, min, s);
M = lerp(pixel, max, s);
s is a small value.

The average of high resolution pixels will tend to pixel over time.

Neighbor clamping of main TAA will not affect these pixels.



Smoothed over time by main TAA


[GAME PAUSED]
CYBERLIFE
KR200
AP700
HOUSEKEEPING
BABYSITTING
COOKING
CX100
BL100
AP400
BL100
CX100
AJ700
KR200
HARD WORK
FOR IT
FREE TIME
FOR YOU
SSR Off


[GAME PAUSED]
CYBERLIFE
KR200
AP700
HOUSEKEEPING
BABYSITTING
COOKING
CX100
BL100
AP400
BL100
CX100
AJ700
KR200
HARD WORK
FOR IT
FREE TIME
FOR YOU
SSR On

Temporal anti-aliasing

- Temporal volumetric lighting
  - Inspired from “Physically based unified volumetric rendering in Frostbite” by *Sebastien Hillaire.*
  - Fog cluster is a 3D checkerboard
  - Checkboard is disabled on spot borders
  - TAA use neighbor clamping in 3D

Temporal anti-aliasing

- Temporal volumetric lighting
  - Sweeping along a froxel (voxel/frustum) can enter in phase with camera motion. We use a blue noise to avoid this.
  - Resolution: 192 x 108 x 64
  - PS4 Pro Resolution: 235 x 135 x 64
  - Performance: between 2 and 3 ms


8

Temporal anti-aliasing

- Conclusion
  - Long development time
    - Perfect motion vectors
    - TAA itself has a lot of subtlety
    - Noise is very important
    - A lot of implications everywhere

Temporal anti-aliasing

- Conclusion
  - Long development time
  - Improved image quality
    - Image stability
    - Shadows
    - SSAO, SSR, Skin SSSSS
    - Volumetric lighting

Temporal anti-aliasing

- Conclusion
  - Long development time
  - Improved image quality
  - Some drawbacks
    - Ghosting
    - Pixel blinking
    - Leaking and vibrations with DOF, motion blur and GUI

Temporal anti-aliasing

- Conclusion
  - Long development time
  - Improved image quality
  - Some drawbacks
  - Impossible to come back from TAA
    - Too much feature relies on it now

Temporal anti-aliasing

- Conclusion
  - Long development time
  - Improved image quality
  - Some drawbacks
  - Impossible to come back from TAA
  - It clearly worth it

Detroit: Become Human

- PS4
  - 1080p at 30 FPS
  - Volumetric lighting: 192x108x64
  - Support HDR TV
- PS4 Pro
  - 2160p checkerboard at 30 FPS
  - GUI in full 2160p
  - Volumetric lighting: 235x135x64
  - Support HDR TV


18
TM
www.pegi.info

Thanks

- Engine team
  - **Nicolas Vizerie,** Christophe Bonnet, Guillaume Caurant, Bertrand Cavalie, Thibault Lambert, Gregory Lecot, Eric Lescop, Sylvain Meunier,
- Other thanks
  - Everyone at Quantic Dream (lighting, set, character, FX, Maya and others!)
  - Jean-Charles Perrier
  - Christophe Brusseaux
  - Adam Williams
  - Julien Merceron

Questions?

- Contact
  - ronan.marchalot@live.fr