GDC | March 21-25, 2022
San Francisco, CA

Real-Time Ray Tracing in HITMAN 3

Alessandro Dal Corso,
Senior Render Programmer,
IO Interactive

#GDC22

HITMAN 3

- Released in 2021
- Built since 2014
- 20+ Levels (3 games in one)

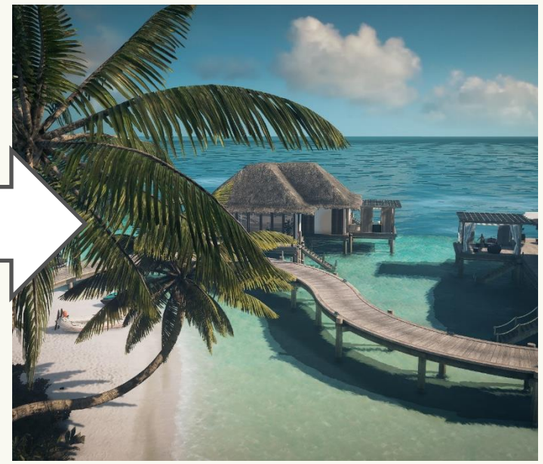
Glacier Engine

- In house engine
  - Tiled deferred lighting
  - Forward step for emissive and transparent

G-buffer

Lighting result

+Transparent



Final image

Design principles

- Reuse existing data
  - Materials, geometry
- Limited content optimization
- Reuse existing systems
  - Culling
  - Reflections
  - etc.

Ray Dispatch anatomy

Ray tracing shaders

```
#ifdef _RAY_TRACING_SHADER
[shader("anyhit")]
void Raytracing_AH(inout Materia
{
    v2fRaytracing_Packed IN_PACK
    UnpackV2FStandardShader(IN_P
```

Any hit shaders

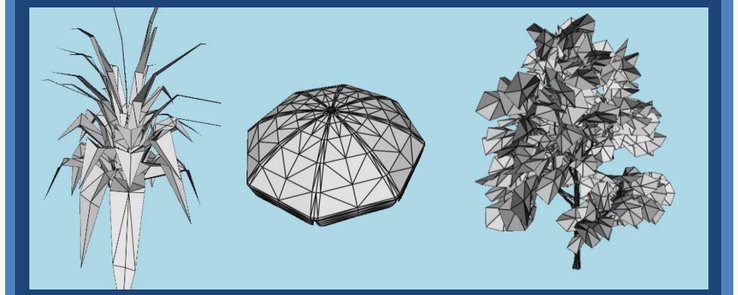
```
#ifdef _RAY_TRACING_SHADER
[shader("closesthit")]
void Raytracing_CH(inout Materia
{
    v2fRaytracing_Packed IN_PACK
    UnpackV2FStandardShader(IN_P
```

Closest hit shaders (optional)

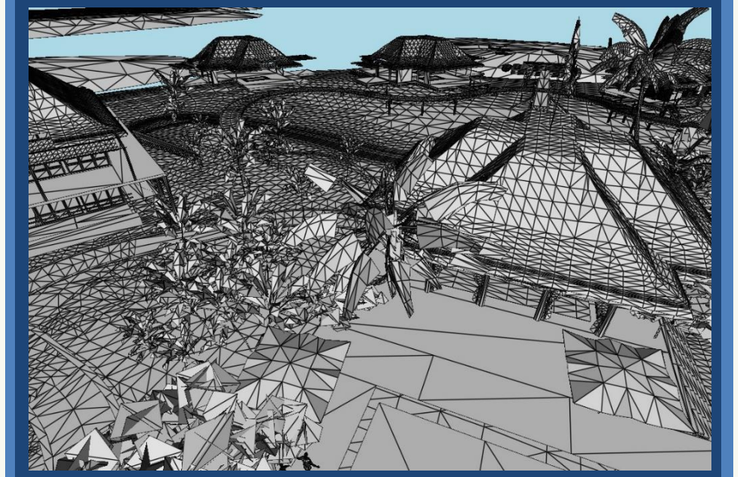
Ray generation shader

Miss shaders

Acceleration structures



Bottom level accelerations (BLAS)

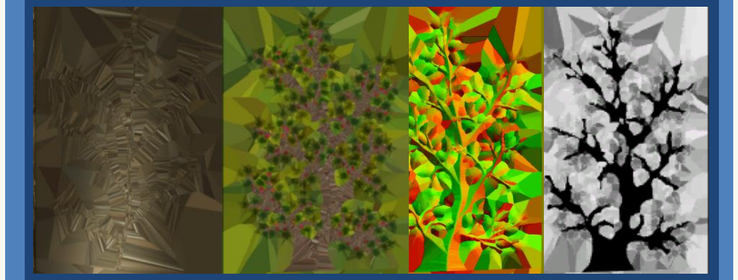


Top level acceleration (TLAS)

Per object data

```
// ------------------------------------
struct SGlobalConstants
{
    float1 ConstantVector1D_03_Value;
    float1 ConstantVector1D_04_Value;
    float1 ShaderLOD_Gloss;
    float1 Specular_Multiplier_Value;
```

Constants

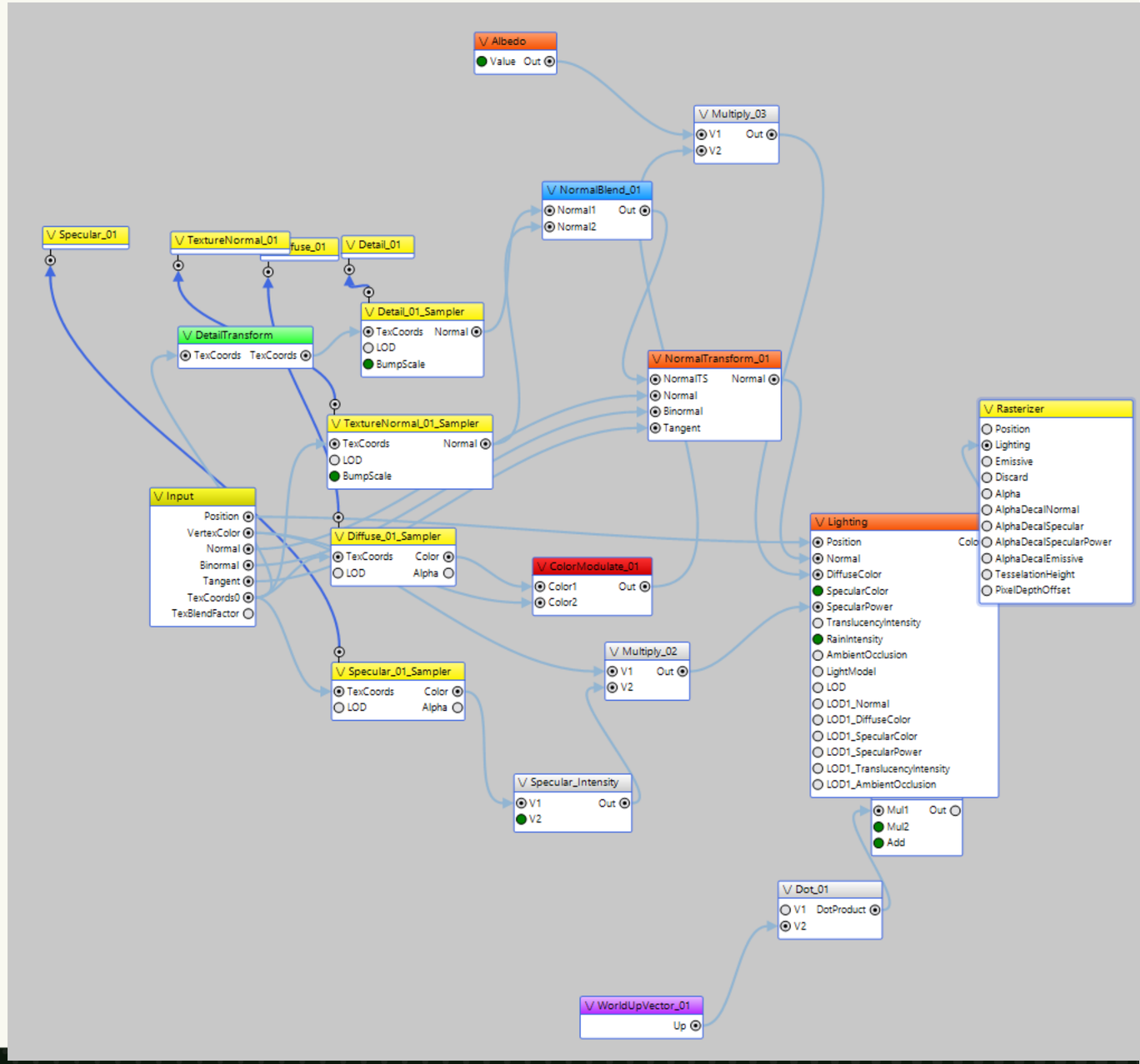


Textures

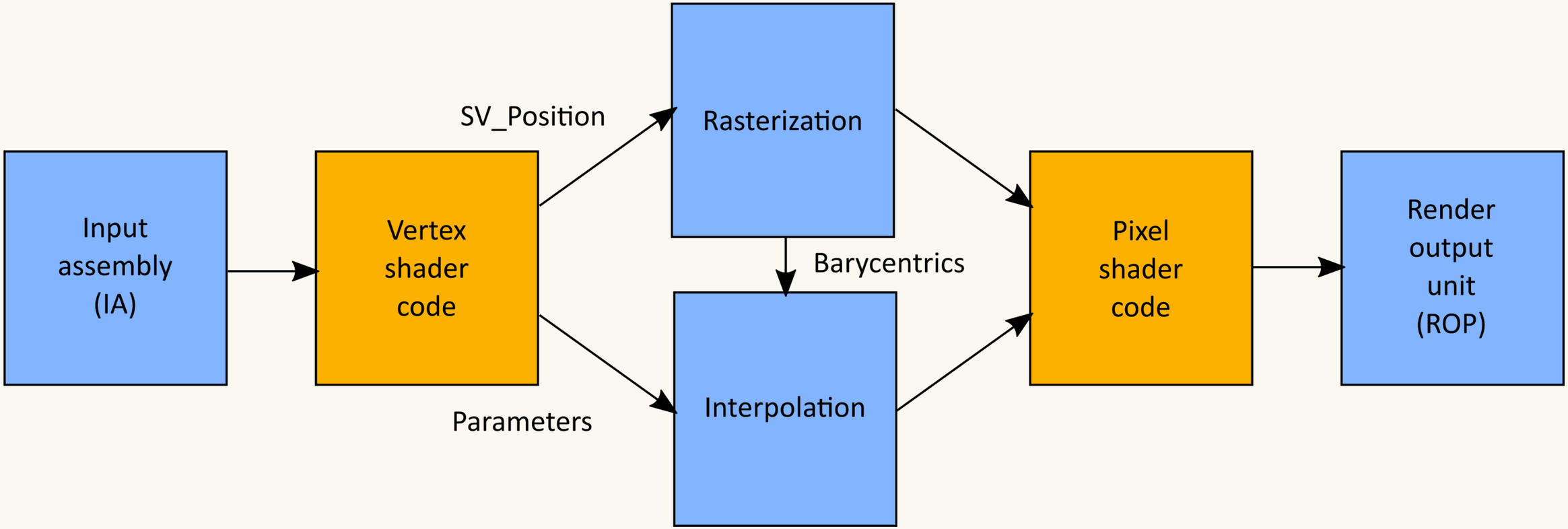
Vertex attributes

Material shaders

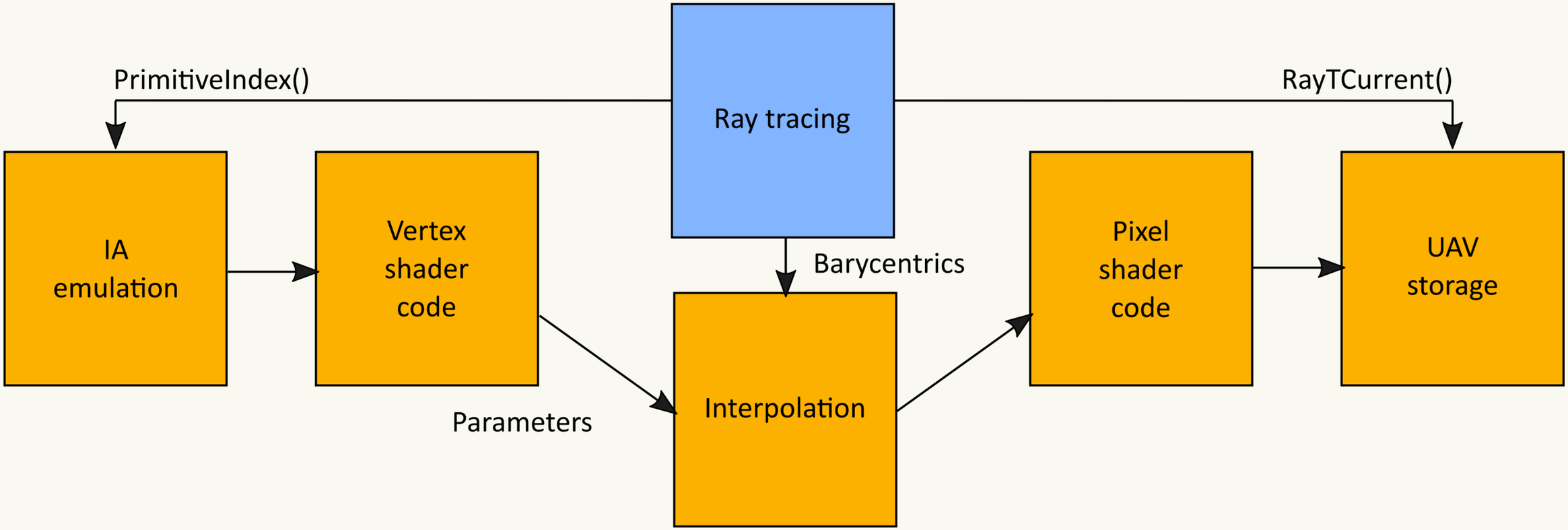
- Materials defined via graph
- Mandatory “pixel” shader and optional “vertex” shader



Converting material shaders

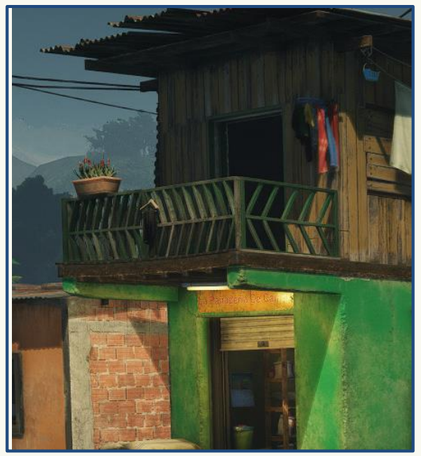


Converting material shaders



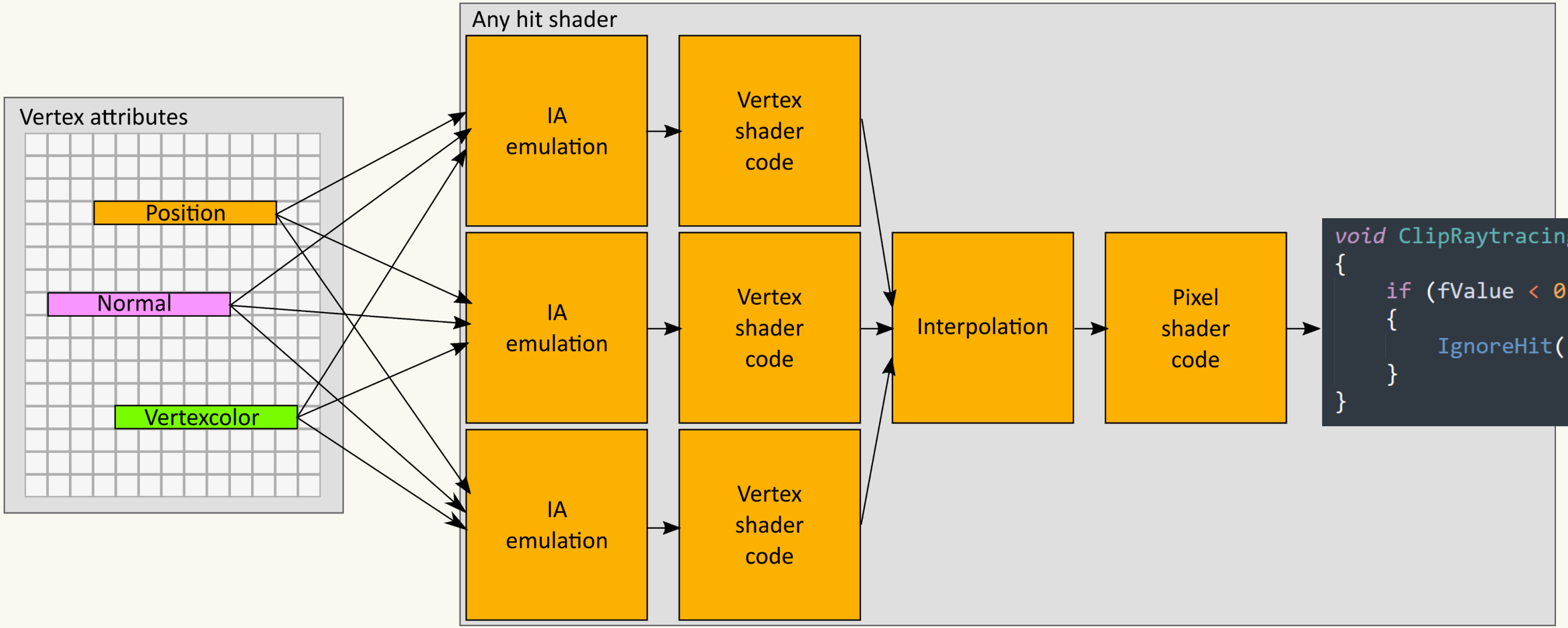
Geometry types

- **Simple (available vertices)**
  - Geometry does not change
  - Can still be translated/rotated/scaled

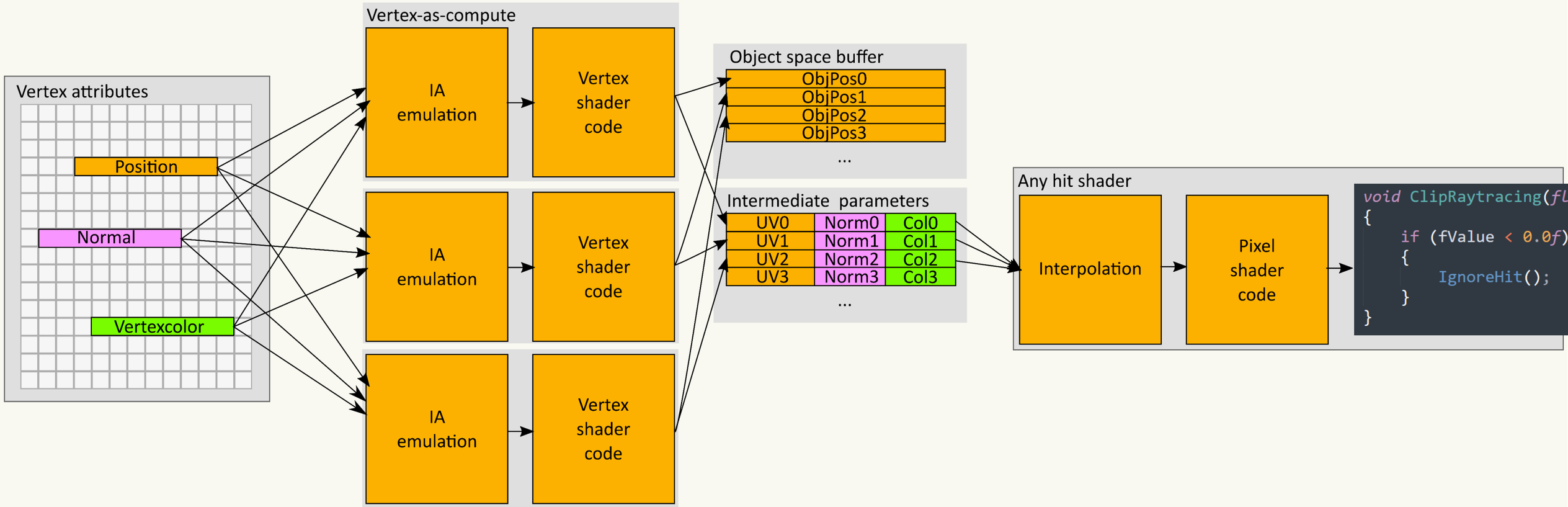


- **Deformable (unavailable vertices)**
  - Custom vertex shaders
  - Bone animation
  - Non-standard vertex layouts
  - Cloth

Shader generation – simple geometry

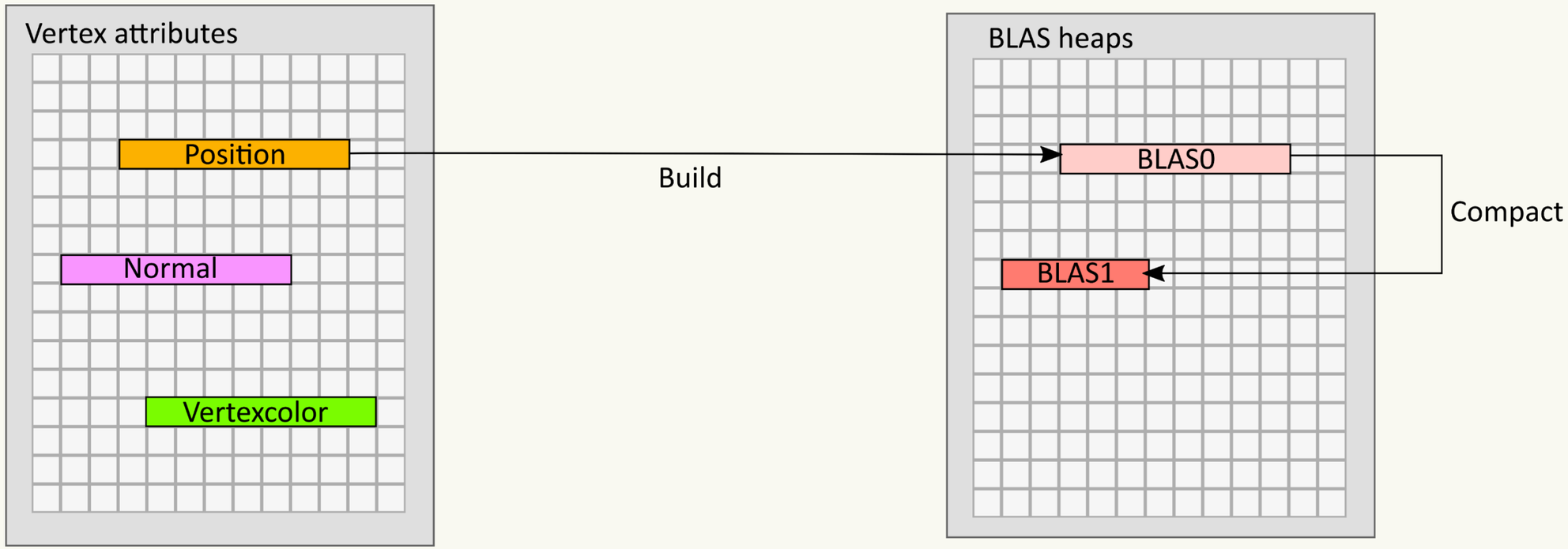


Shader generation – deformable geometry

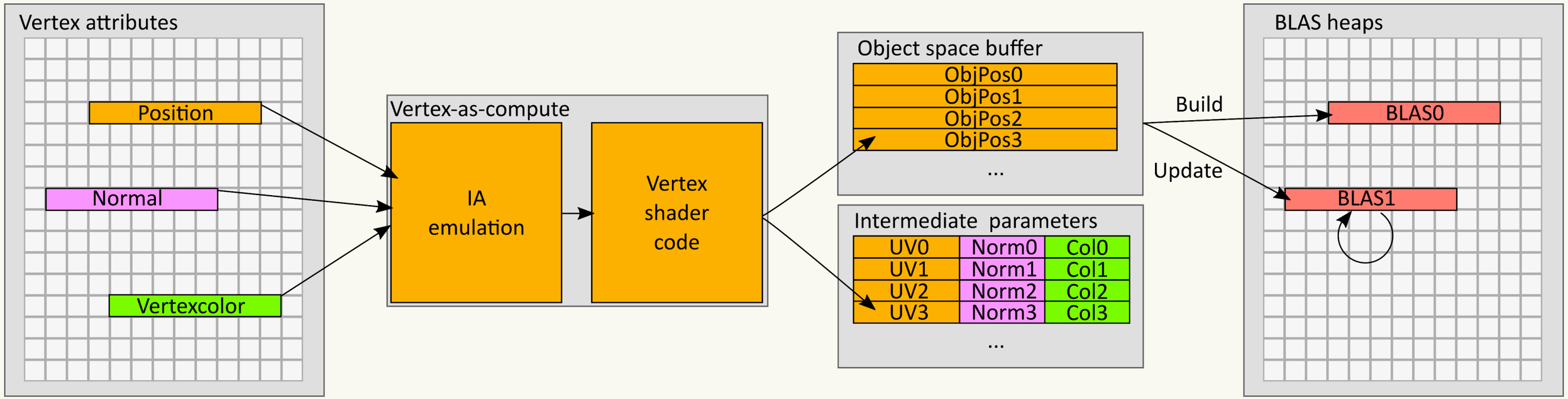


BLAS build – simple geometry

- Built once at the start of a level
  - Compact right after



BLAS build – deformable geometry



Local root signature and shader tables

Material CB + textures

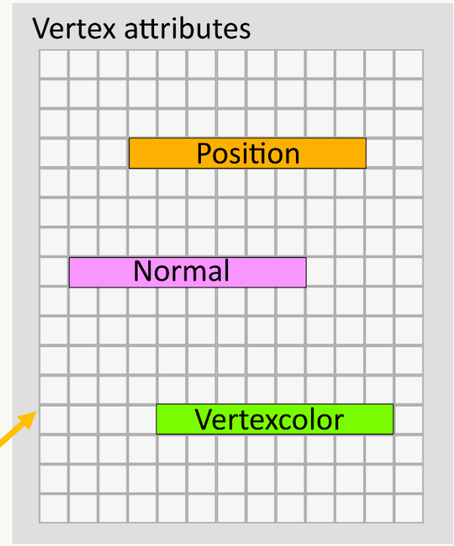
Hitgroup identifier (32 bytes)
AH + CH pair

`register(b0, space1)` Material const buffer (8 bytes)

`register(b1, space1)` Per-object const buffer (8 bytes)

`register(b2, space1)` Geometry const buffer (8 bytes)

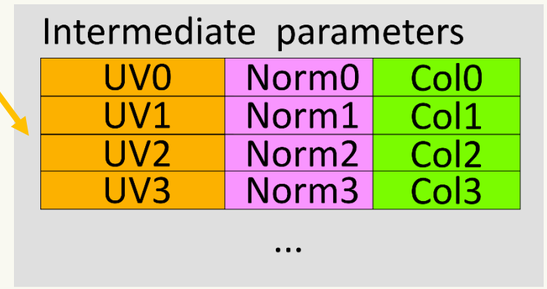
`register(t0, space1)` Parameter buffer (8 bytes)



Geometry CB + attributes

```
struct S_cbPerObject
{
    float4x4        go_mObjectToView;
    float4x4        go_mObjectToViewPrev;
    float4          go_vPosScale;
    float4          go_vPosBias;
    float4          go_vTexScaleBias;
    float4          go_vObjectId;
```

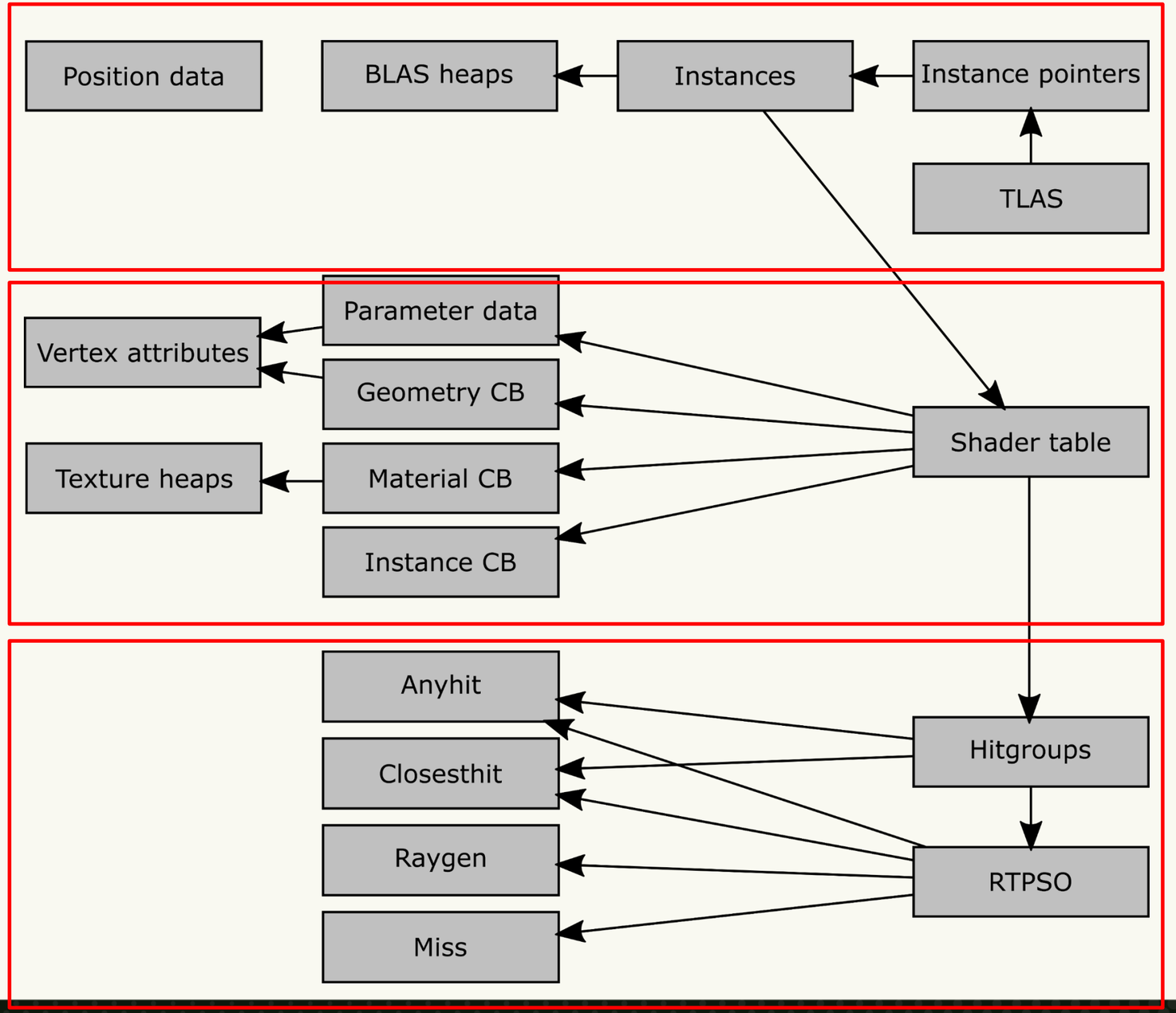
Instance CB



Parameter SRV

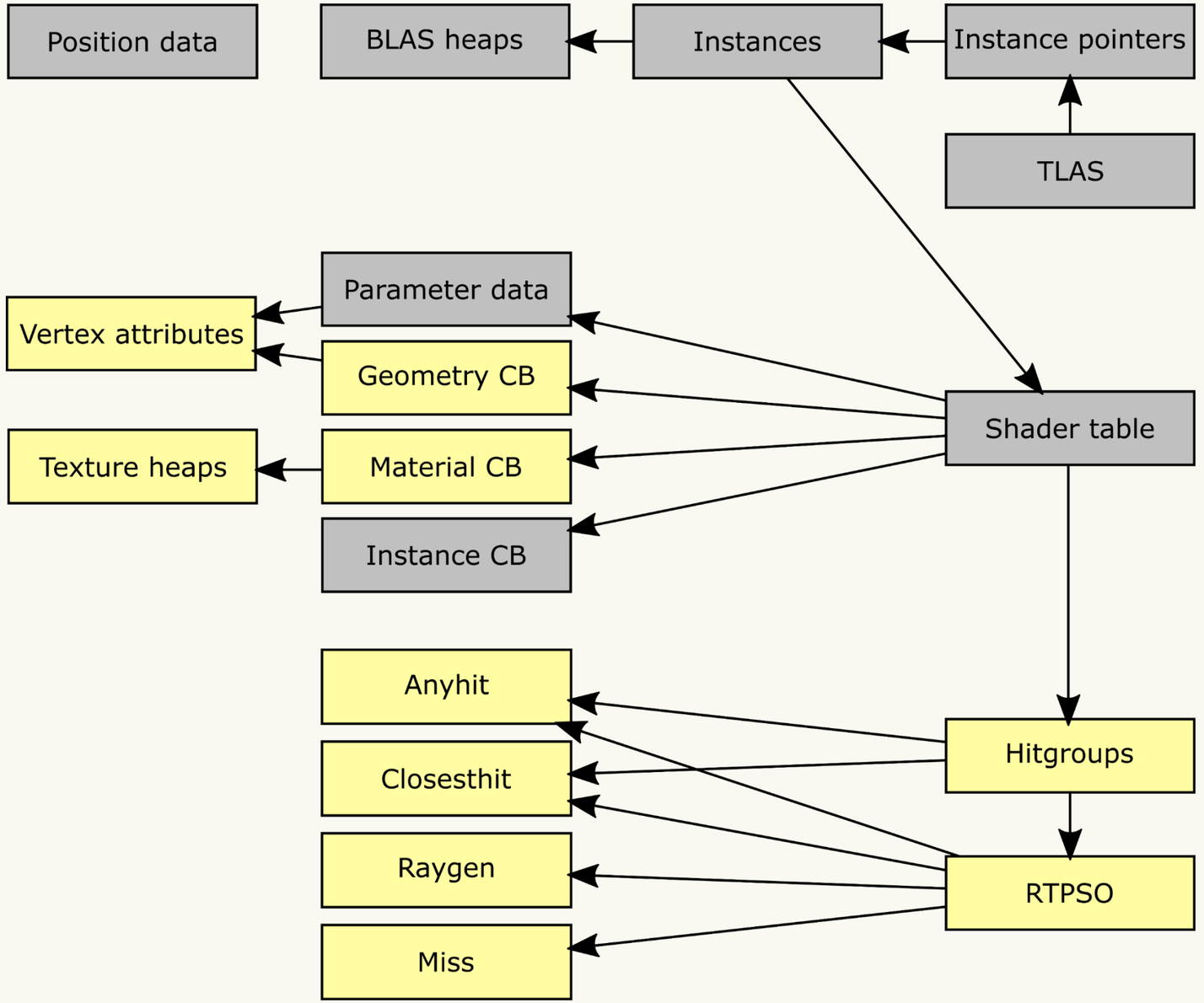
Update tasks

- Diagram represents GPU data
  - Arrows are data dependencies
- Update runs in async compute next to gbuffer generation



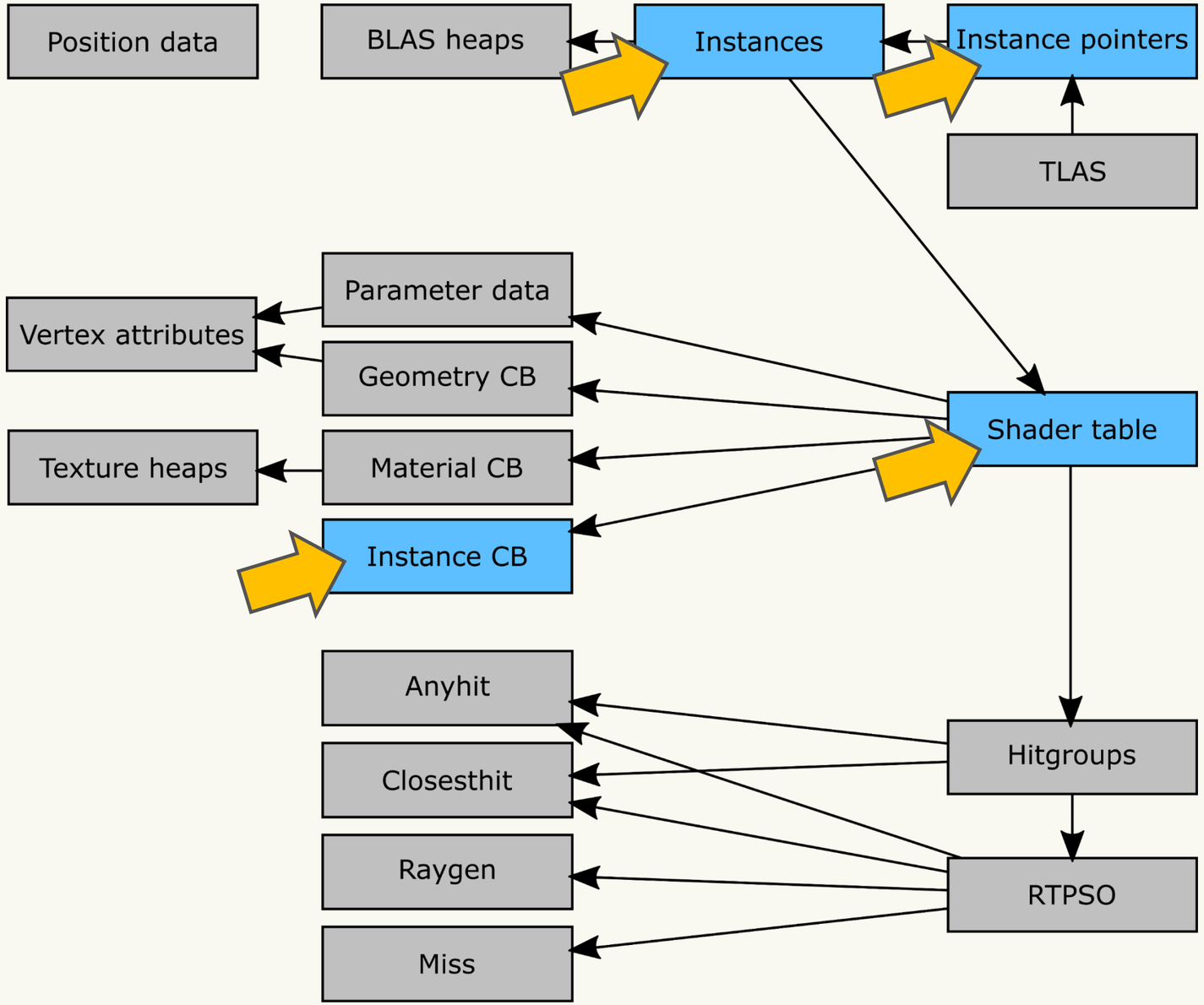
Level load

- Vertex attributes and textures are available for rasterization
- Geometry and material CB are generated at the same time
- RTPSO is built as shaders stream in



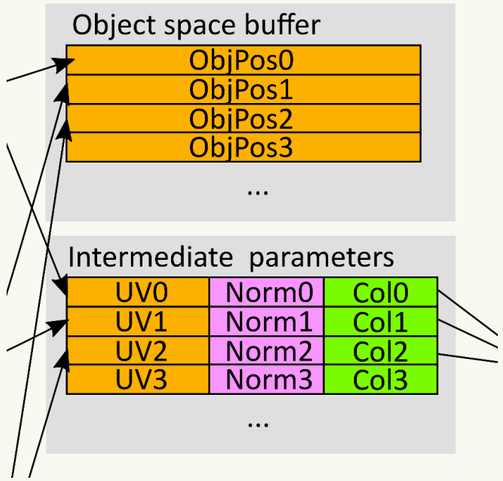
Bookkeeping

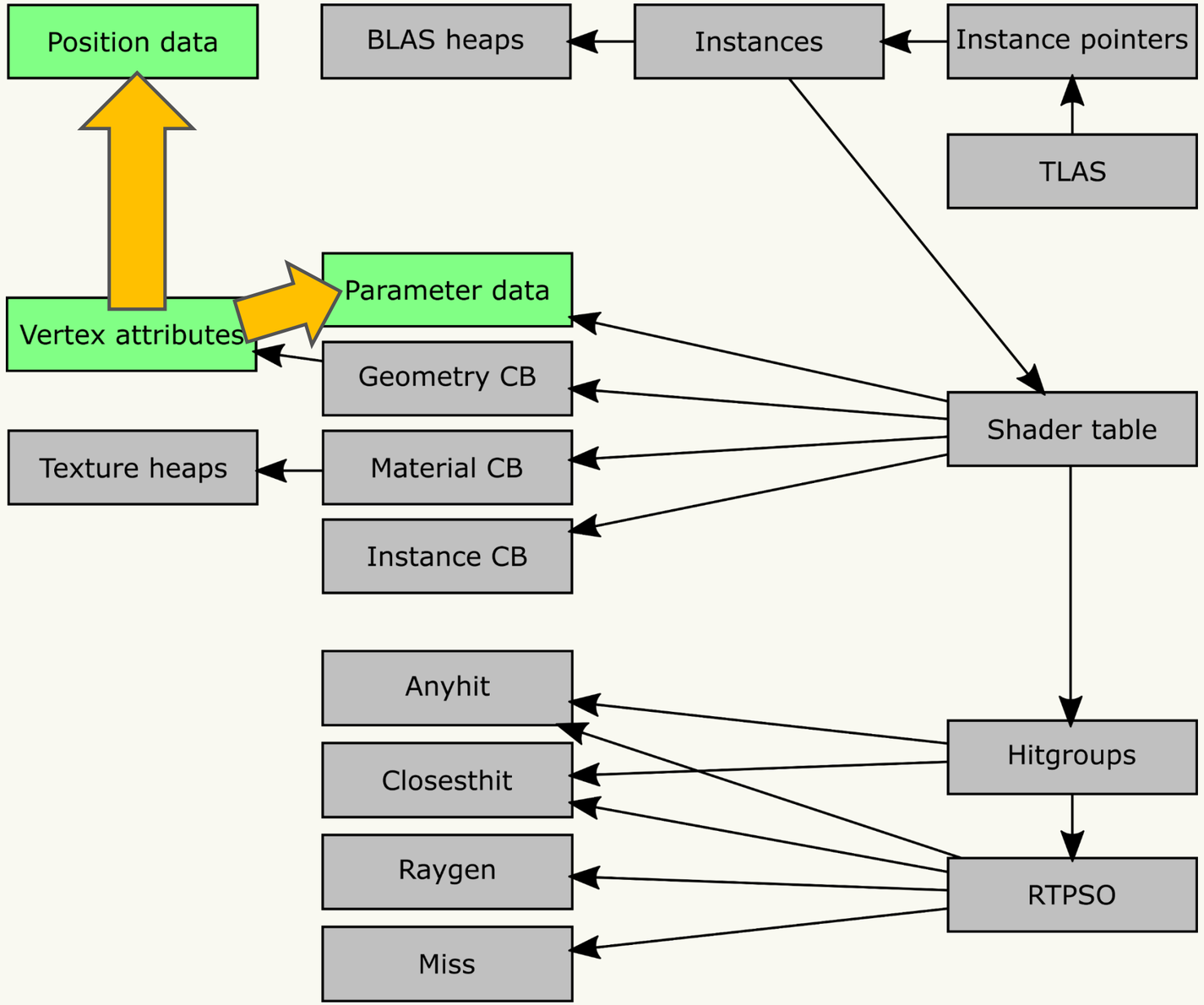
- Various GPU updates
  - Per object data
  - Static object update
  - TLAS instances and pointers
  - Shader table entries



Vertex-as-compute

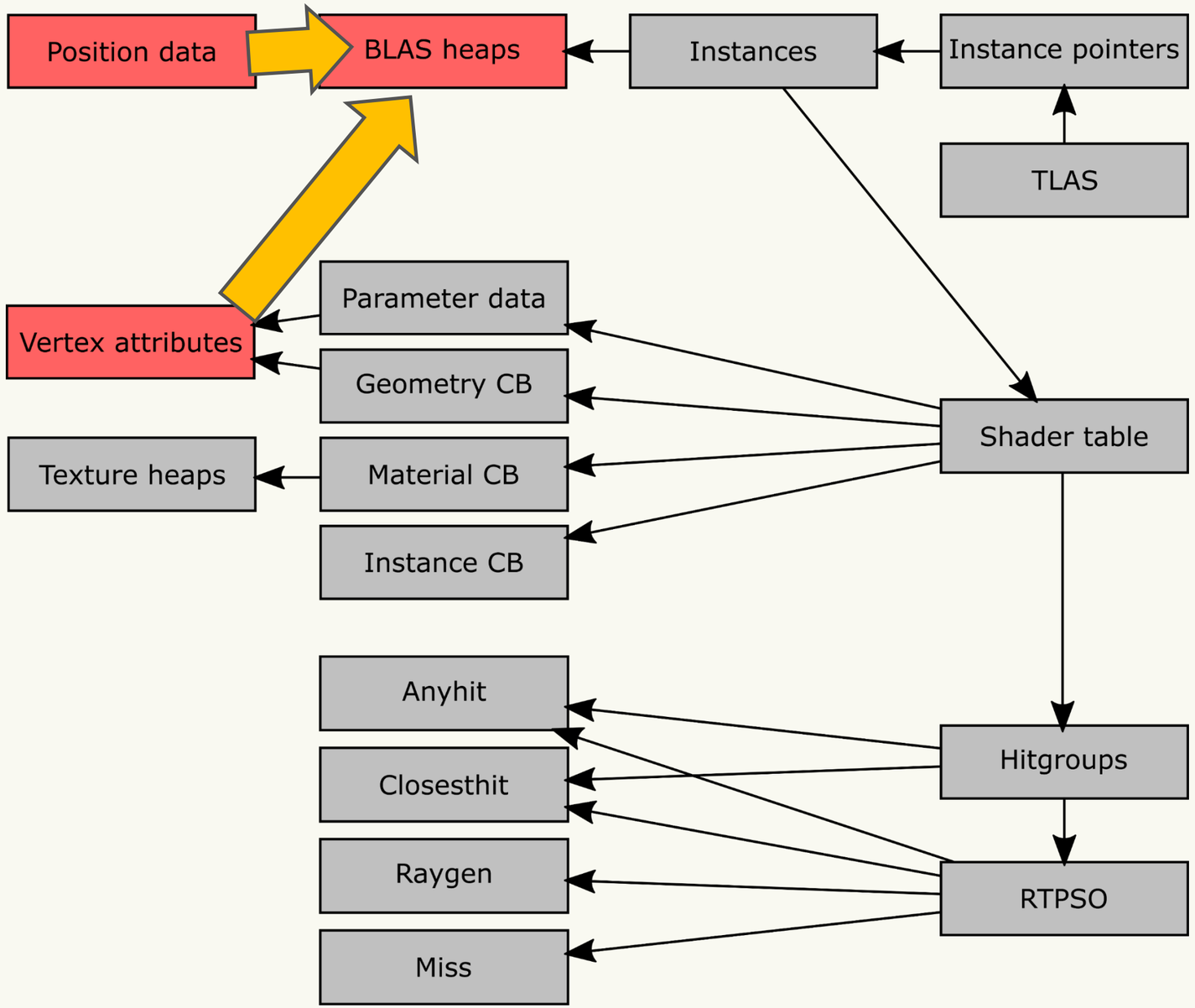
- Multithreaded
  - Modified draw loop
- Position/parameter linearly allocated per frame





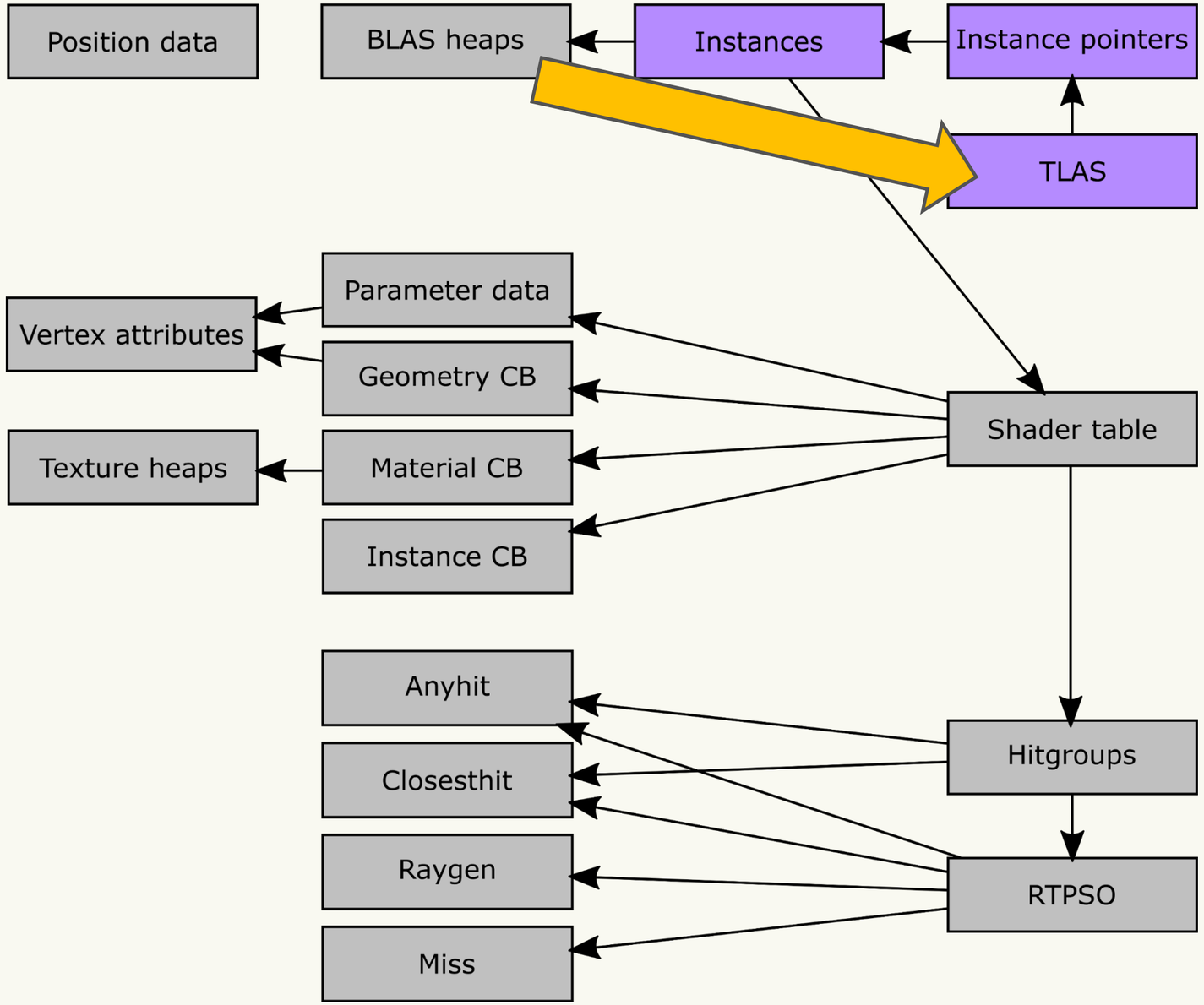
BLAS build

- Batched
- Single command list
- Build/Update ratio based on LOD and frames



TLAS build

- Using pointer buffer
- Instances don’t need to be updated unless they move




Screen space reflections


Planar reflections


Specular probes


Ray traced reflections

Reflections

- Enhance existing screen space reflections (SSR)
- Hybrid system, adding ray traced reflections (RTR)

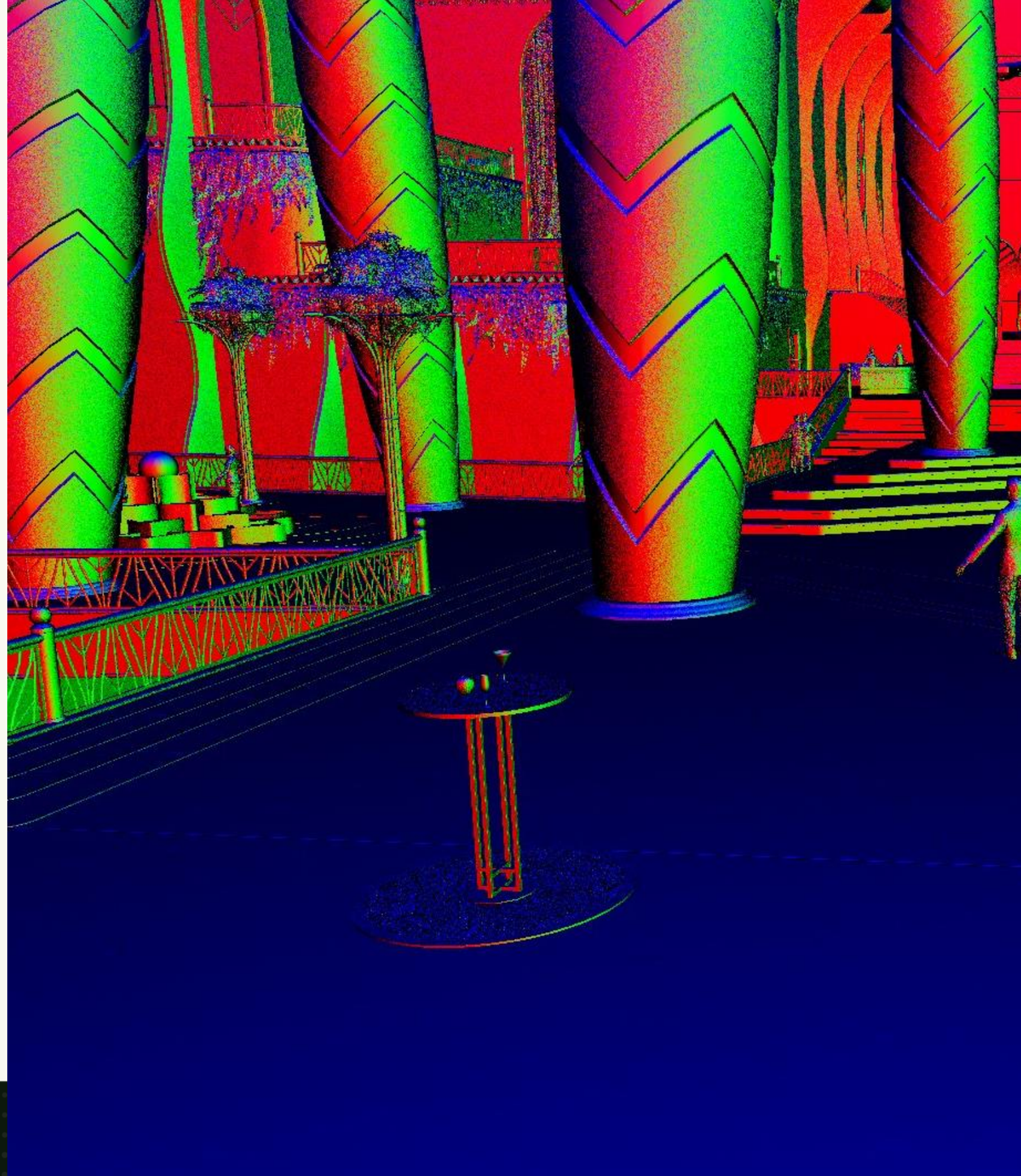
Screen space reflections + specular probes

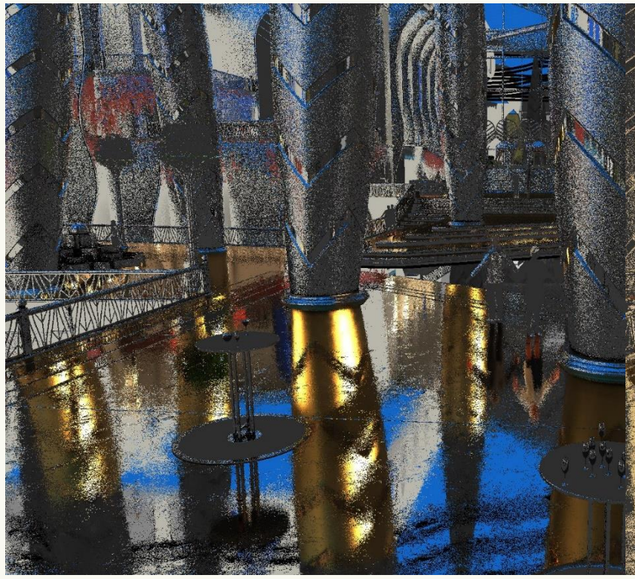
Reflections

- Enhance existing screen space reflections (SSR)
- Hybrid system, adding ray traced reflections (RTR)

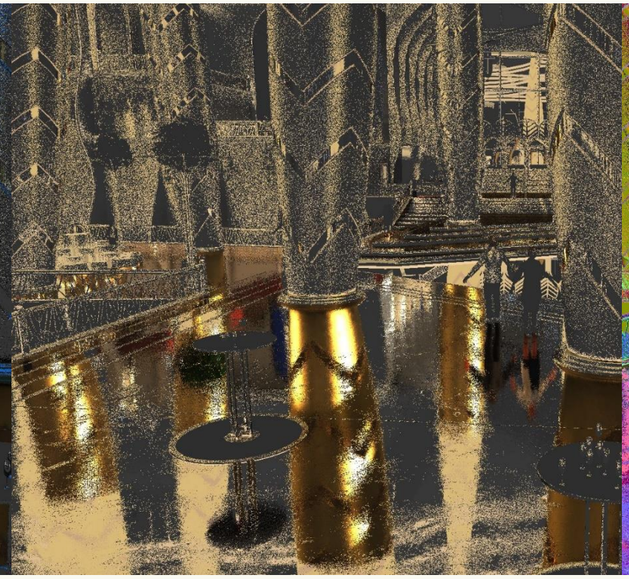
Scheduling rays

- Use SSR result when available
- Do not trace some rays based on reflectance
- Reflectance bump to keep rays aligned

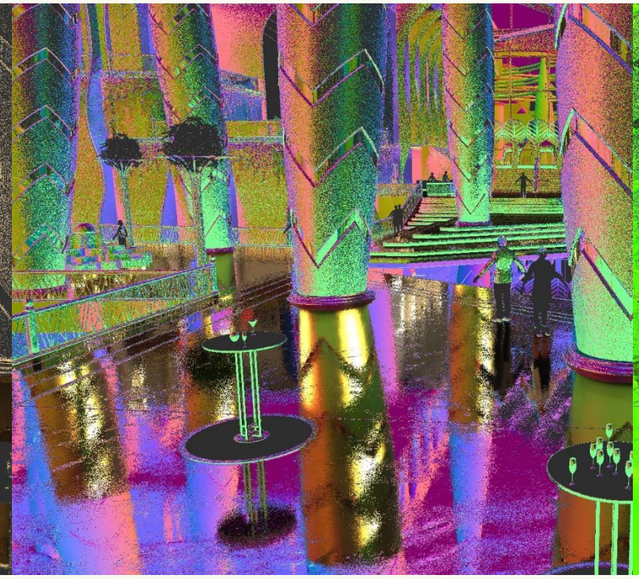




Diffuse



Specular



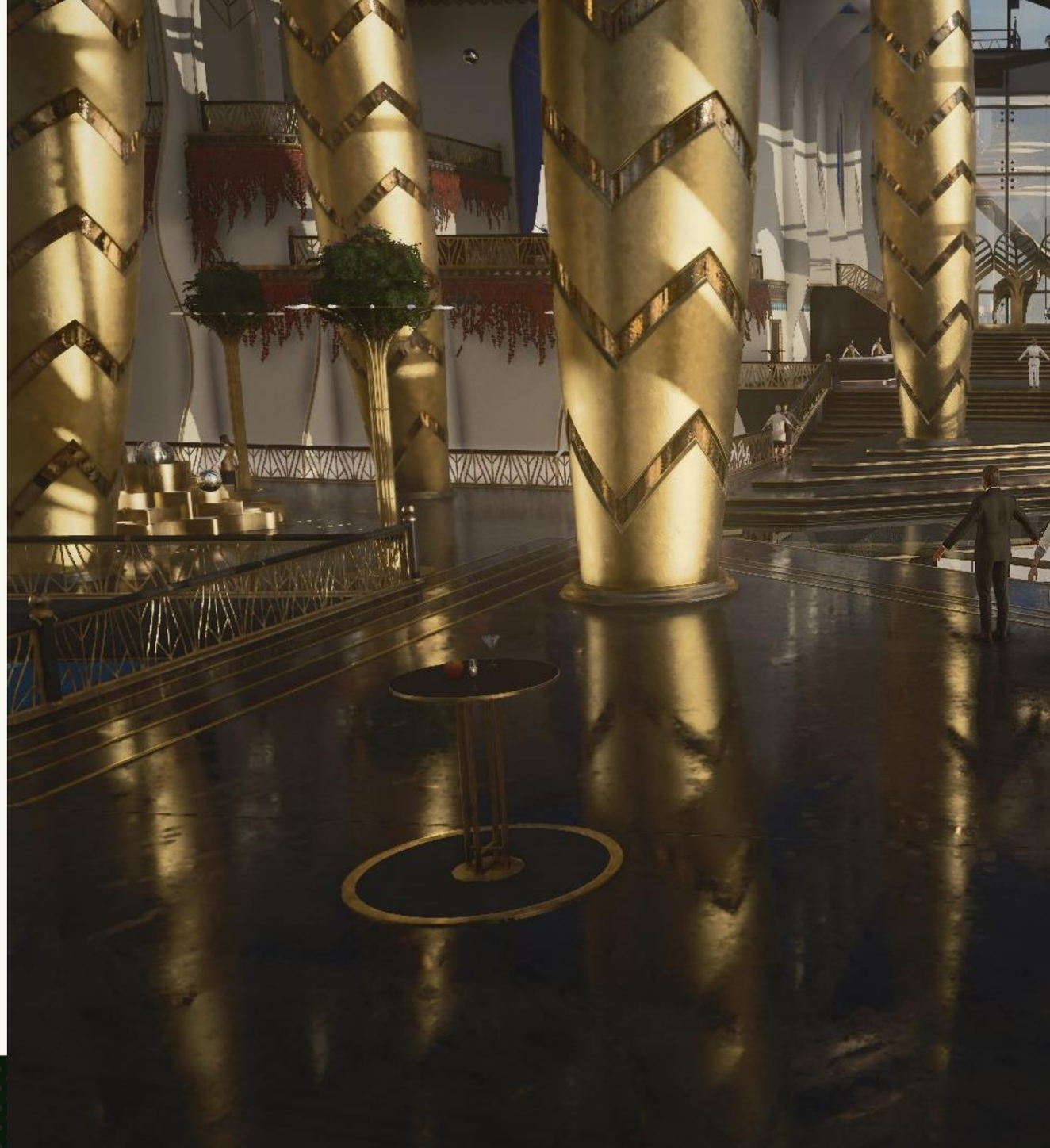
Normal

Specular power + AO

Final lit result

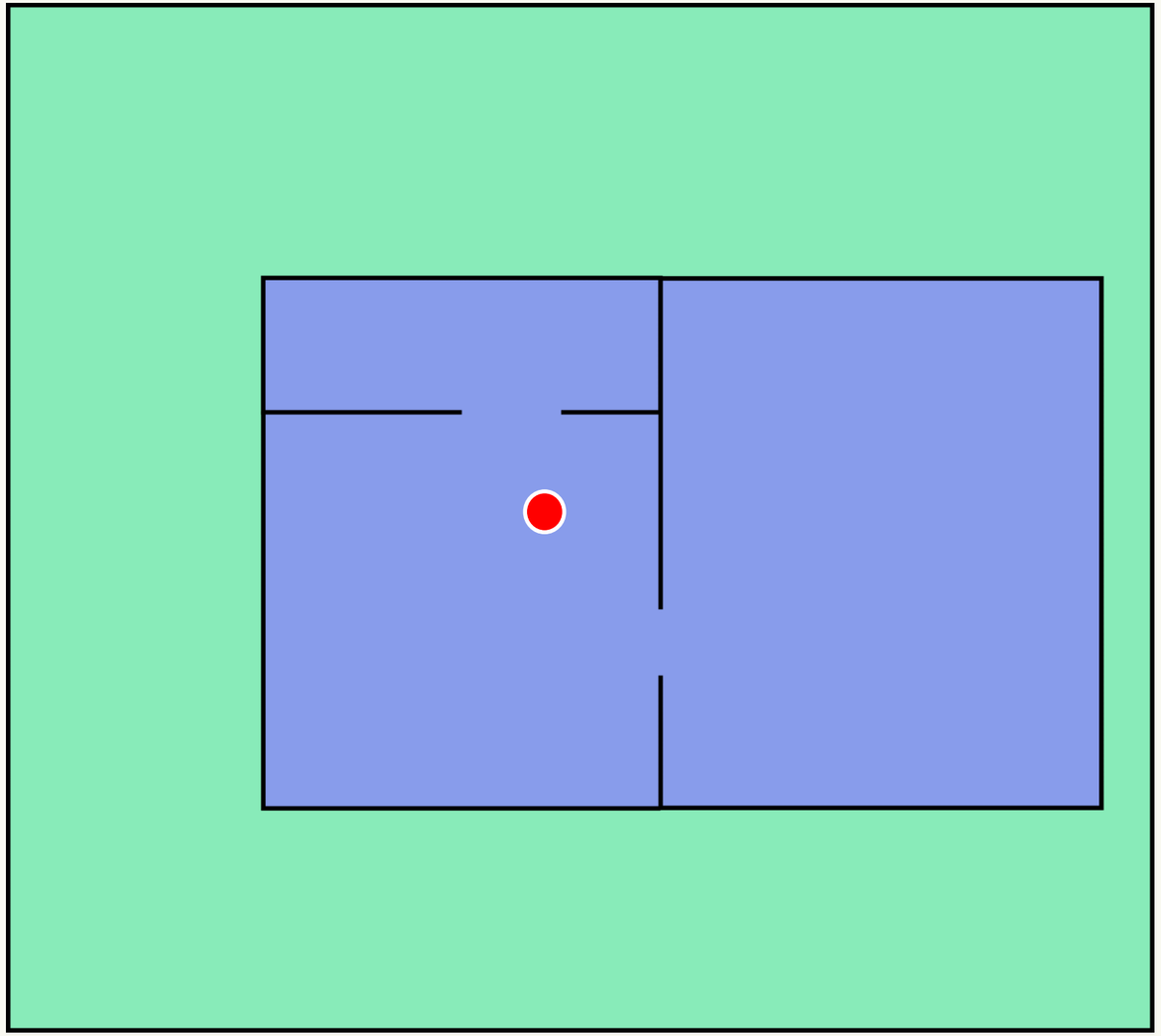
Shading

- Voxel grid data structure
- Grid-based
- Similar in spirit to tile deferred



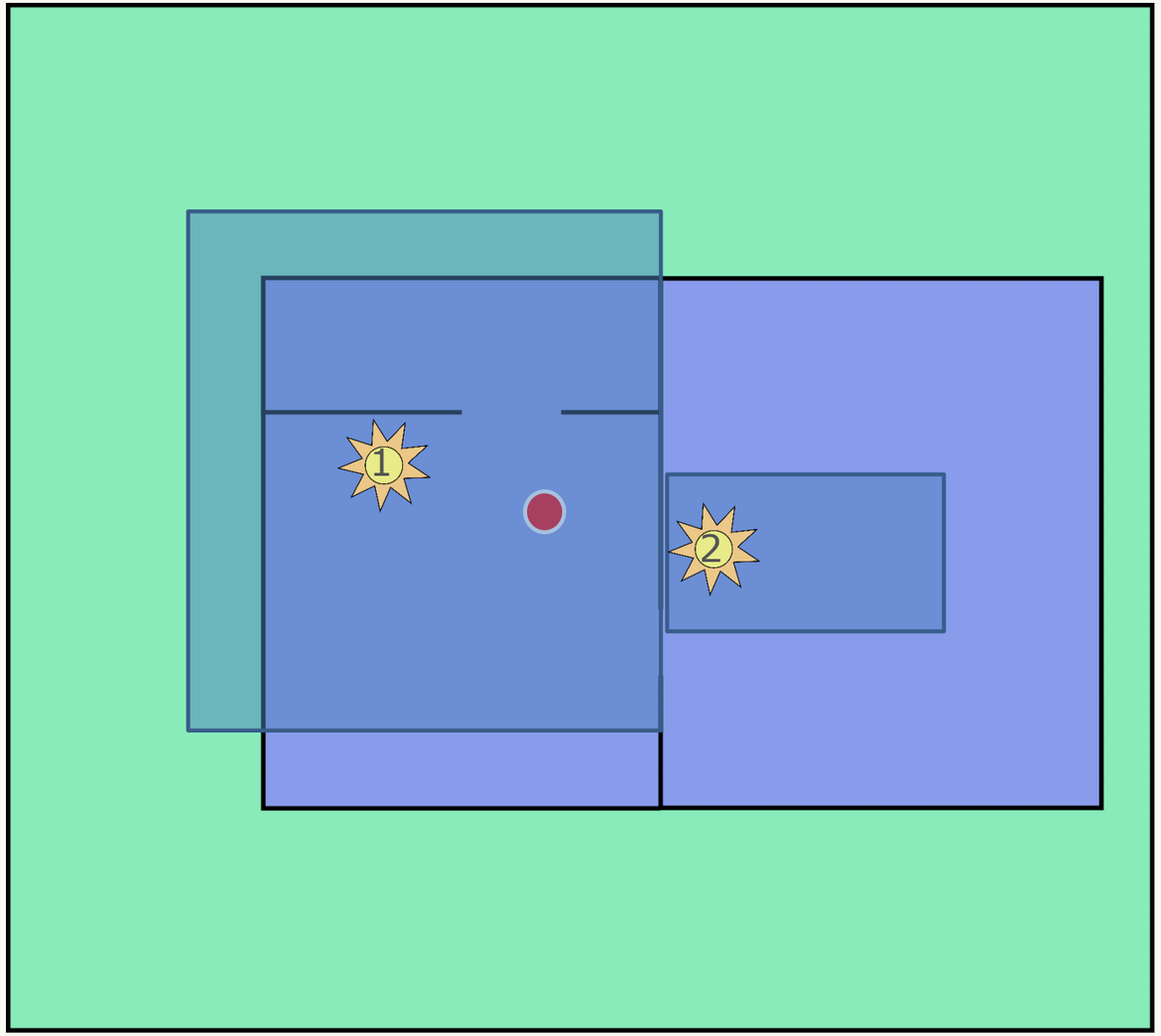
Shading

- Room system
- Caching light area of influence



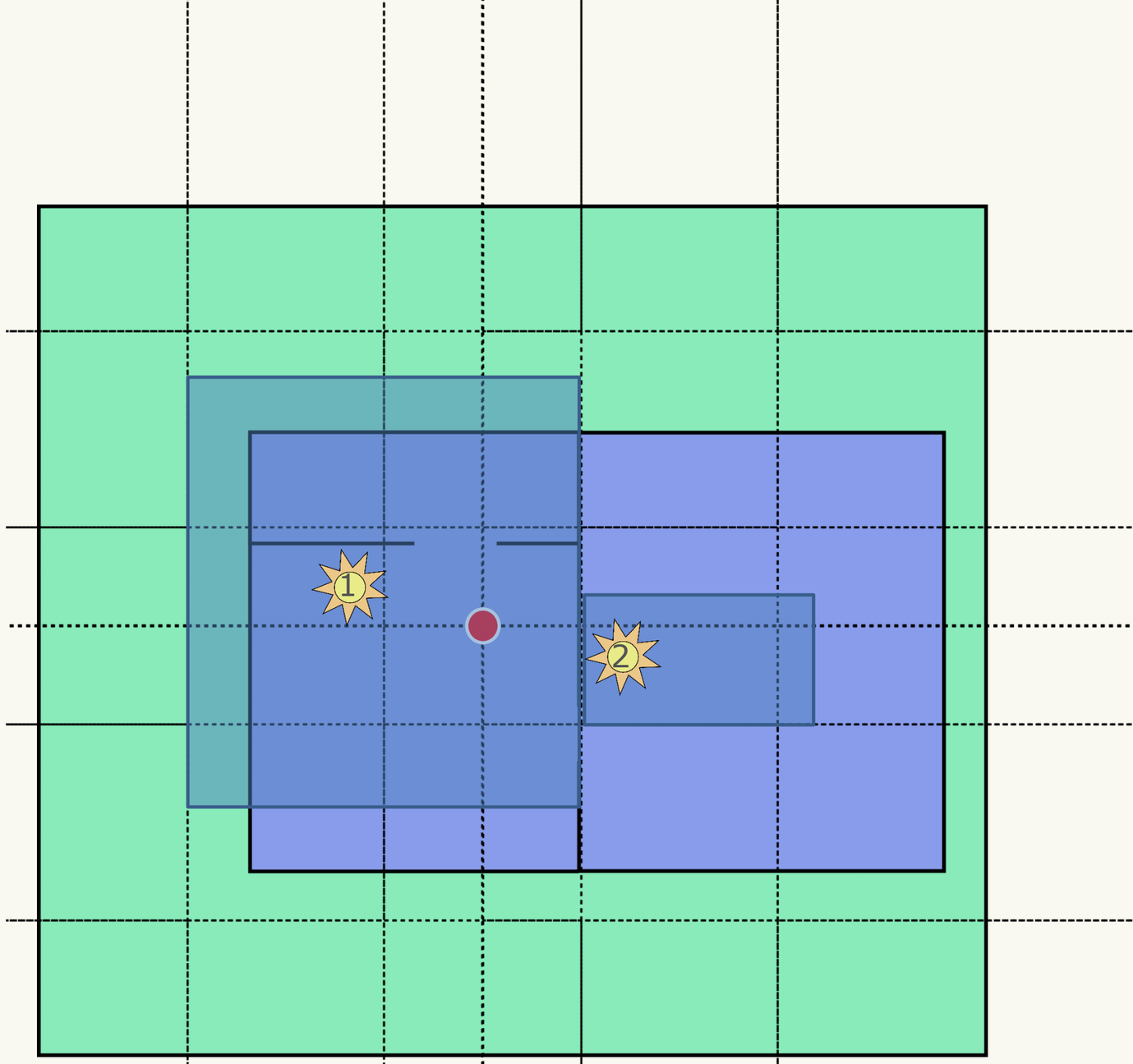
Shading

- Room system
- Caching light area of influence



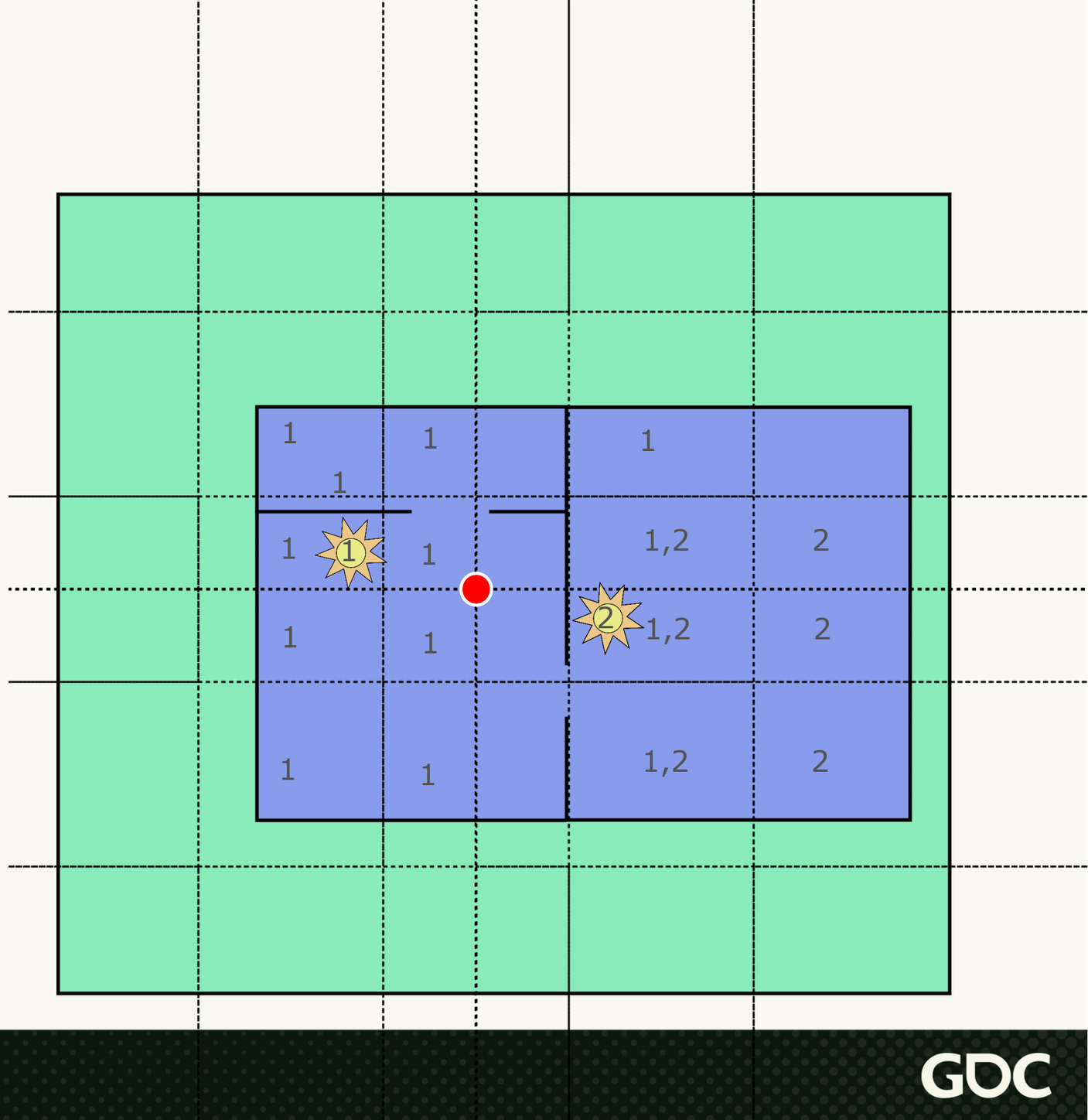
Shading

- Room system
- Caching light area of influence



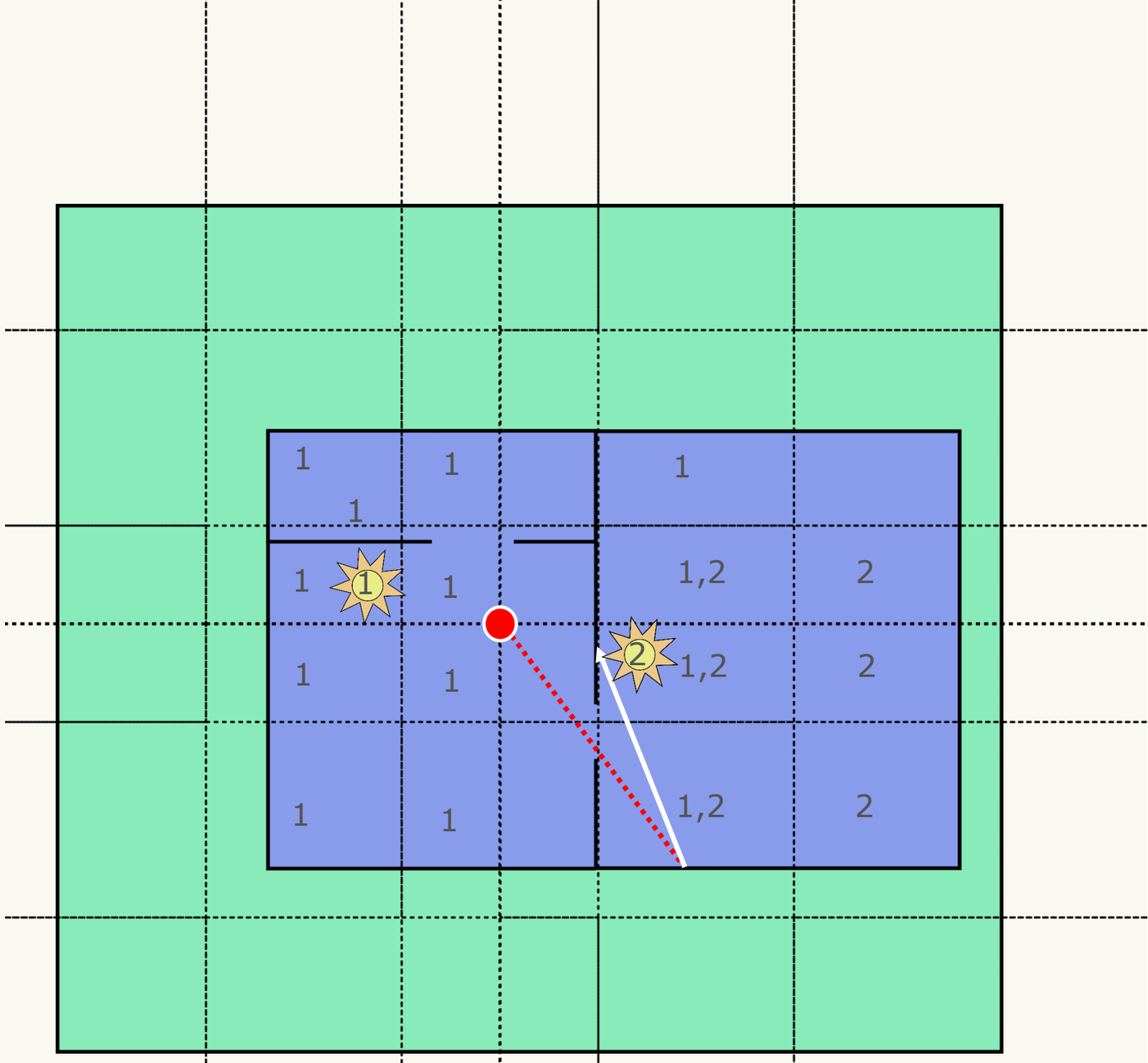
Shading

- Shading done based on rooms
- Avoids light leaks



Shading

- Shading done based on rooms
- Avoids light leaks



Denoising

- NVIDIA Real-Time Denoisers (NRD)
- ReLAX/ReBLUR depending on quality

Final image, noisy

Final image + NRD

Final image, noisy

Denoising

- NVIDIA Real-Time Denoisers (NRD)
- ReLAX/ReBLUR depending on quality

Final image, noisy

Final image + NRD

Final image + NRD

Transparent objects

- Depth and normal
- Assumed perfectly reflective to avoid denoising

SSR + specular probes

Ray traced reflections

SSR + specular probes

Transparent objects

- Depth and normal
- Assumed perfectly reflective to avoid denoising

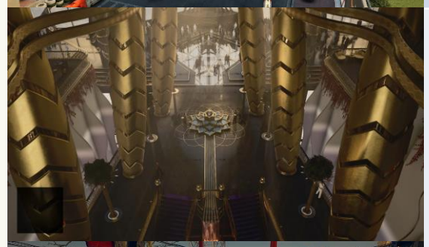
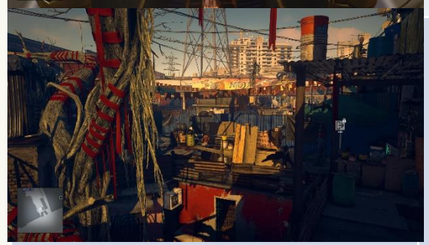
SSR + specular probes

Ray traced reflections

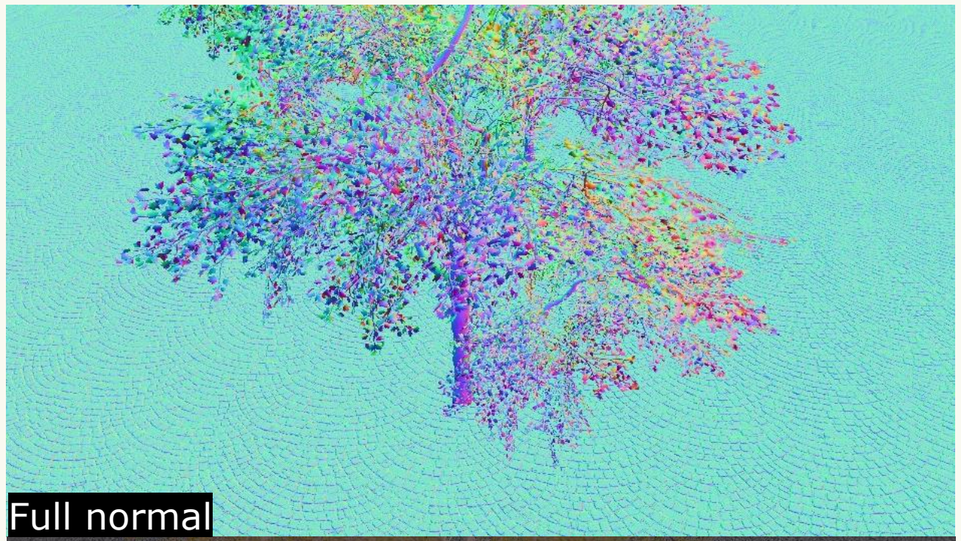
Ray traced reflections

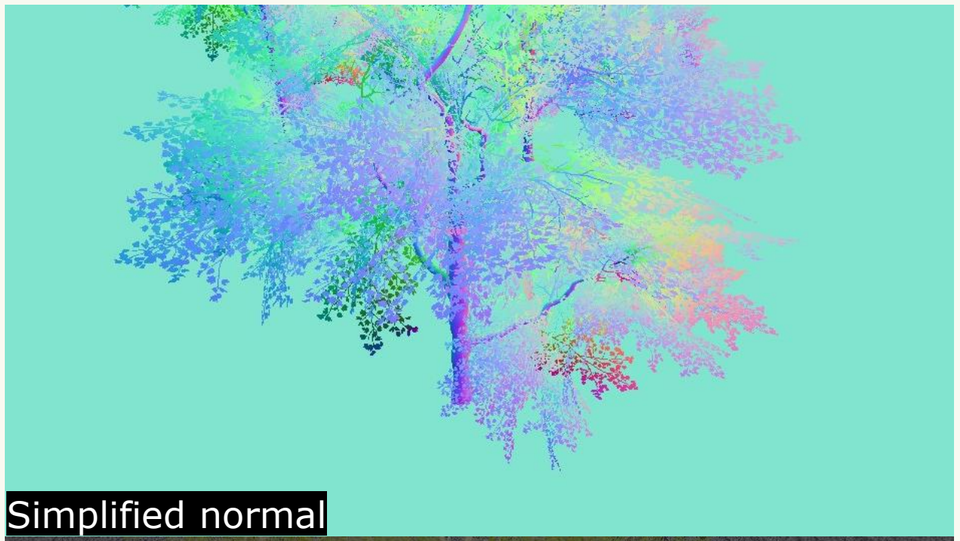
Performance worst cases - Reflections

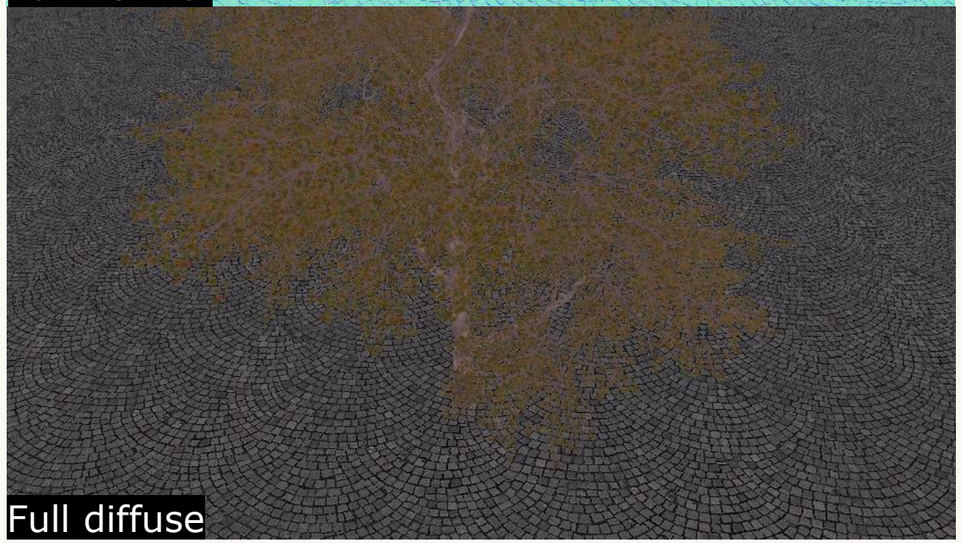
- Captured on NVIDIA RTX 3080 @ native 1080p, High quality

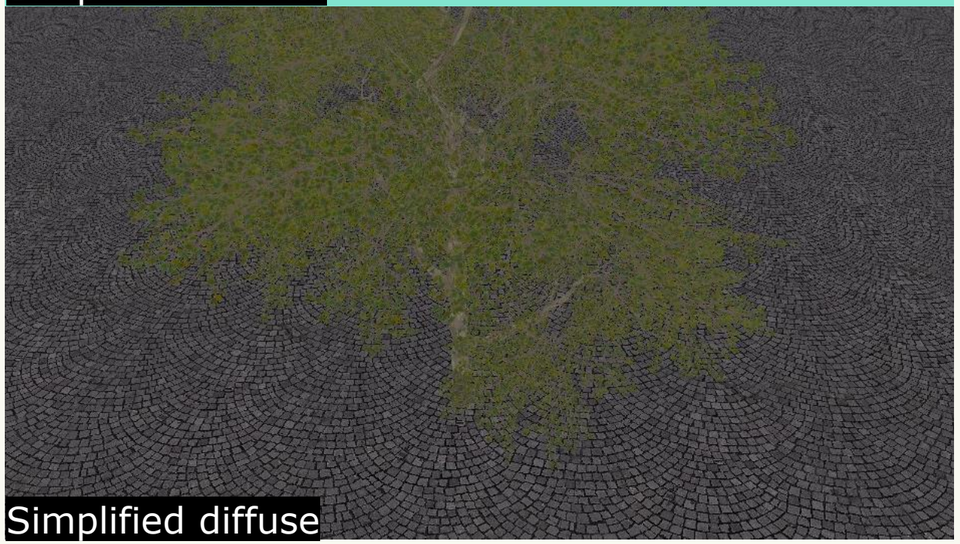
Camera	Frame total	BLAS updates	TLAS build	Tracing	WS Lighting	Denoising
	20.2ms	2.6ms	0.34ms	3.5ms	5.6ms	2.7ms
	20.1ms	2.7ms	0.35ms	3.9ms	3.5ms	2.8ms
	27.8ms	3.0ms	0.37ms	7.2ms	4.6ms	2.8ms

Optimizations – Simplified lighting


Full normal


Simplified normal


Full diffuse

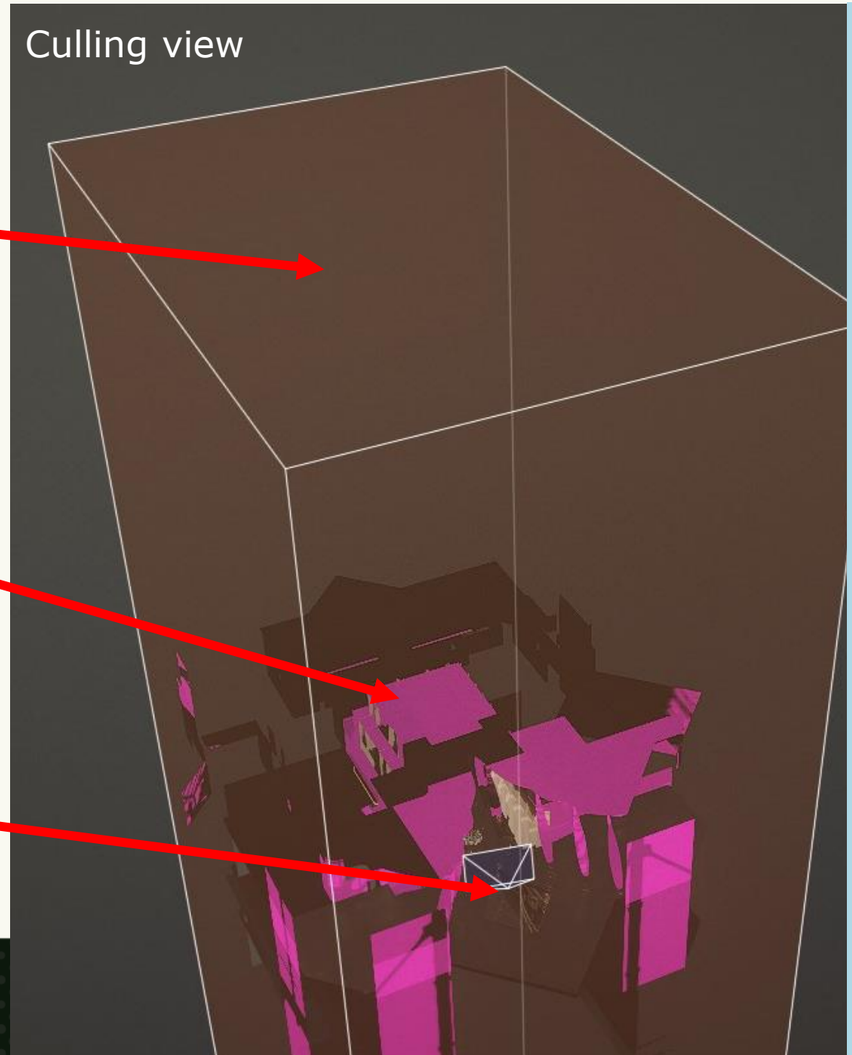

Simplified diffuse

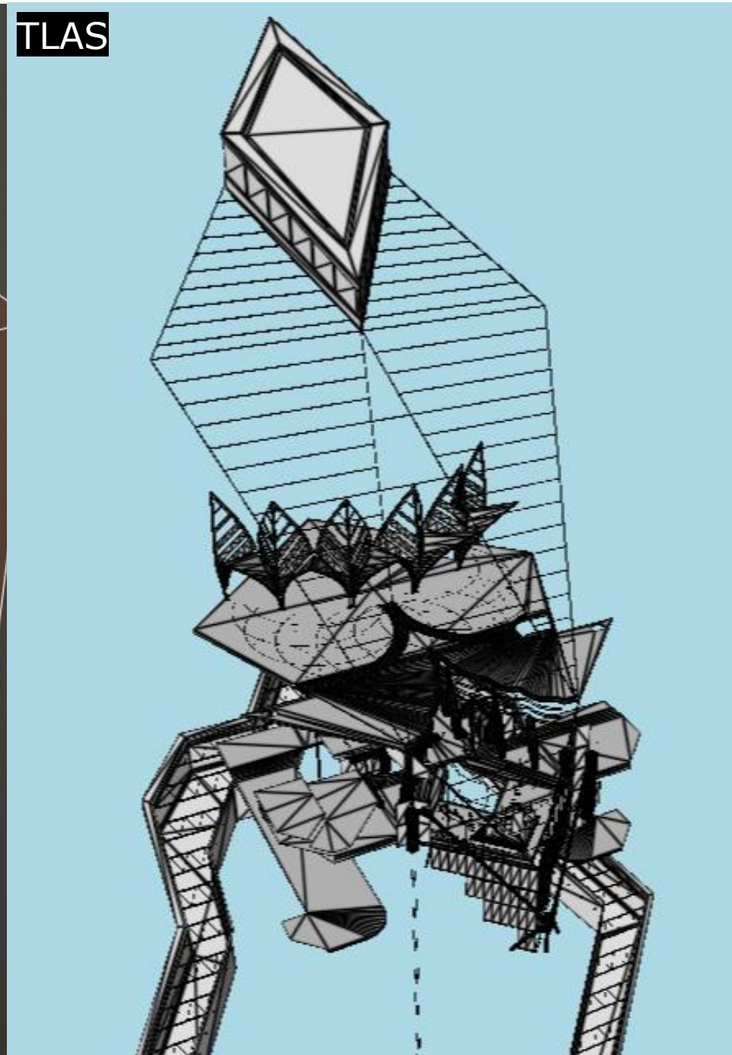
Optimizations – Culling

Reflections culling

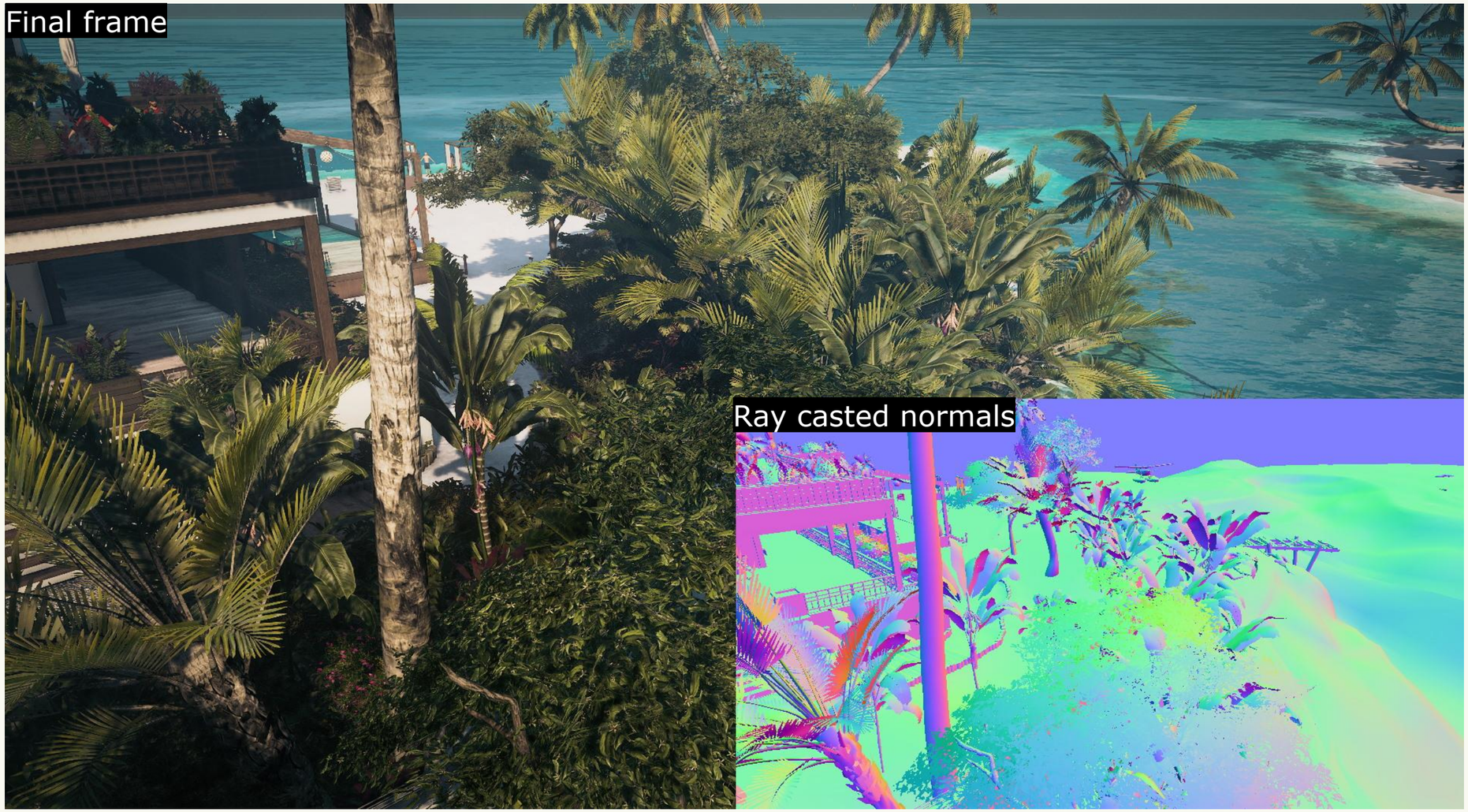
Regular culling

Camera





Optimizations - Content



Future work

- Improve blending between screen space and ray traced reflections
- Transparent support in the reflections
- Emissive support


Blending

Ray traced shadows

- Single light, only for sun/moon
- Similar implementation
  - Accepts first Hit
  - No Closest hit
- Refines some of the existing shadows
- Recycles reflection work if possible


Cascaded shadow map


Ray traced shadows

Thank you:

- The render team at IOI
  - Anders Wang Kristensen and Nikolai Petrov for help on the implementation/bugfixing
  - Brian Engqvist Johansen for QA