Real-time rendering engine for implicit surfaces using sparse structures

Author
Víctor Orrios Barón

Directors
Alfonso López Ruiz
Néstor Monzón González

Image

Image

↑

Rendering

Image

↑

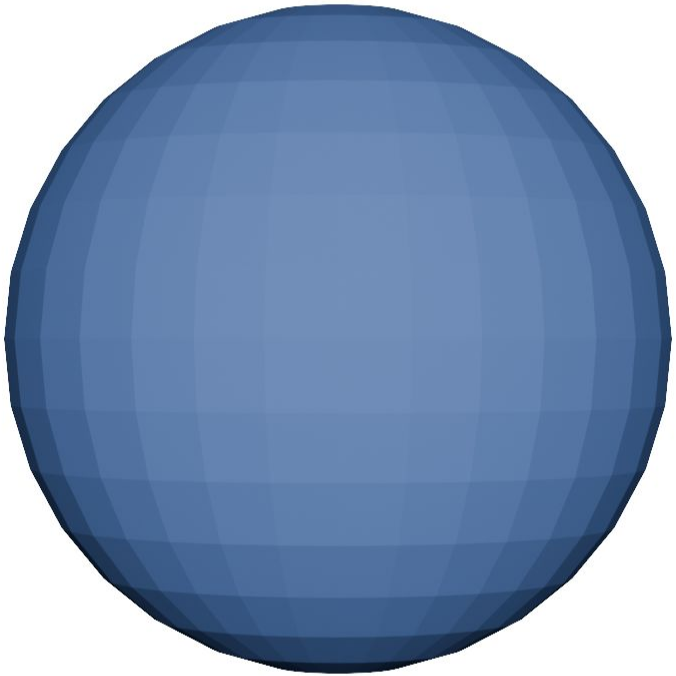
Rendering

↑

Model

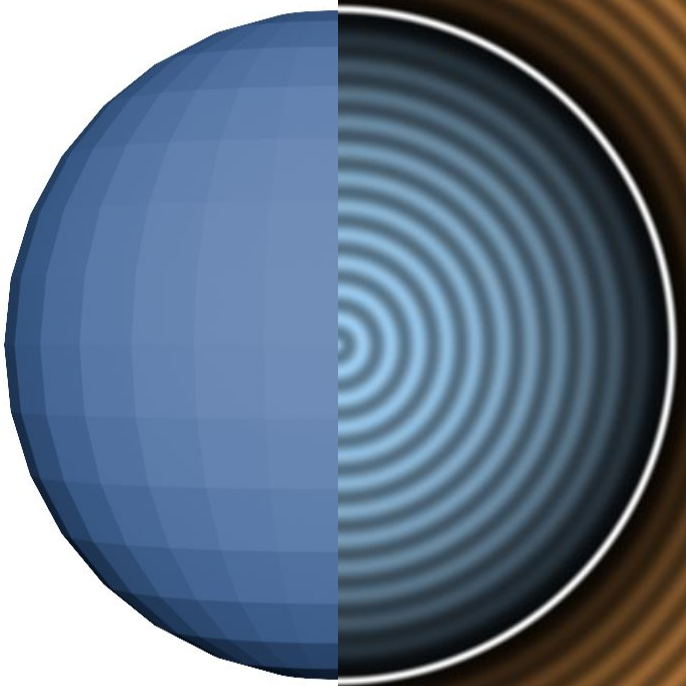
Model representation

Explicit
Polygon mesh
960 tris

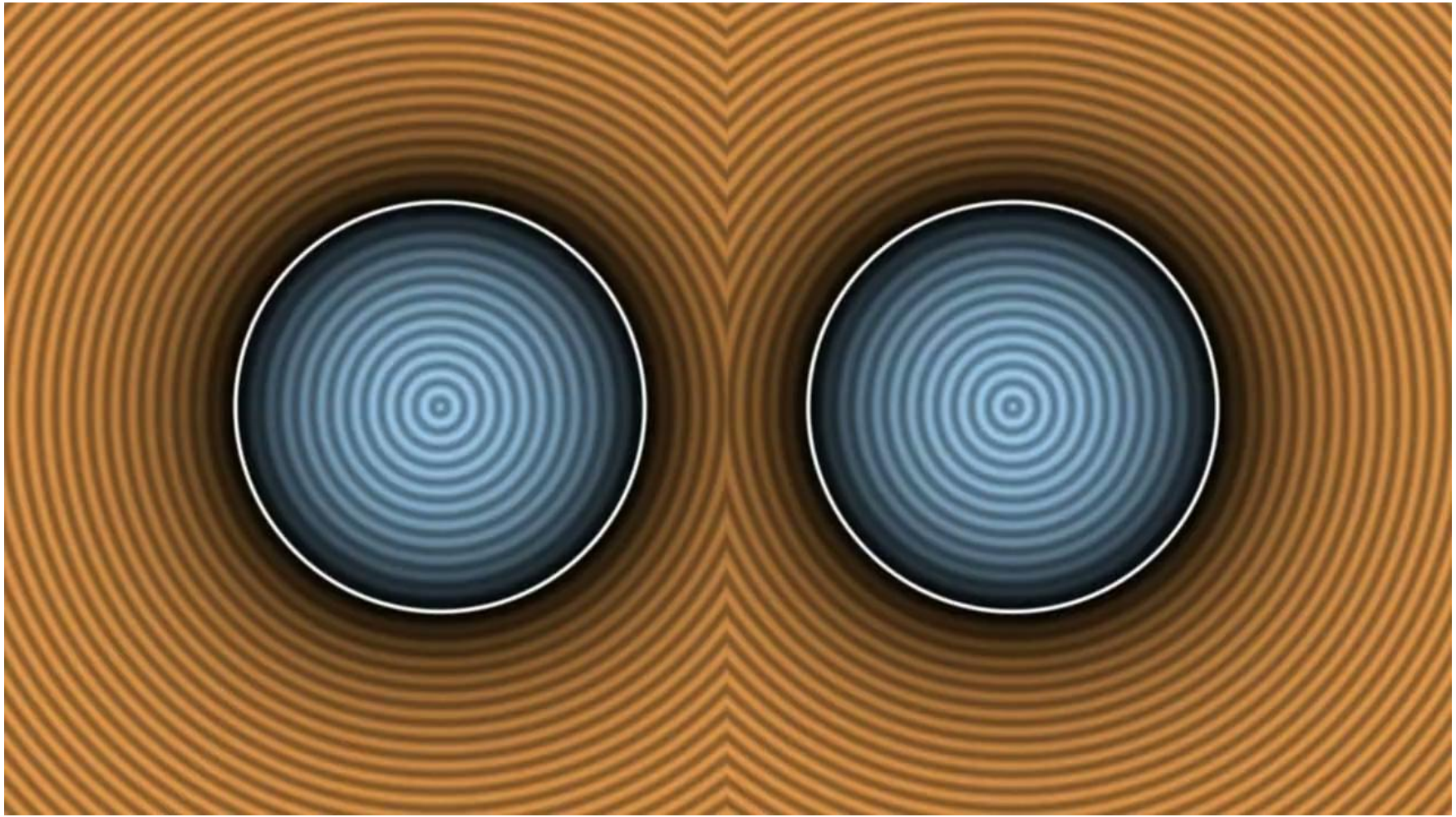


Model representation

Explicit
Polygon mesh
960 tris



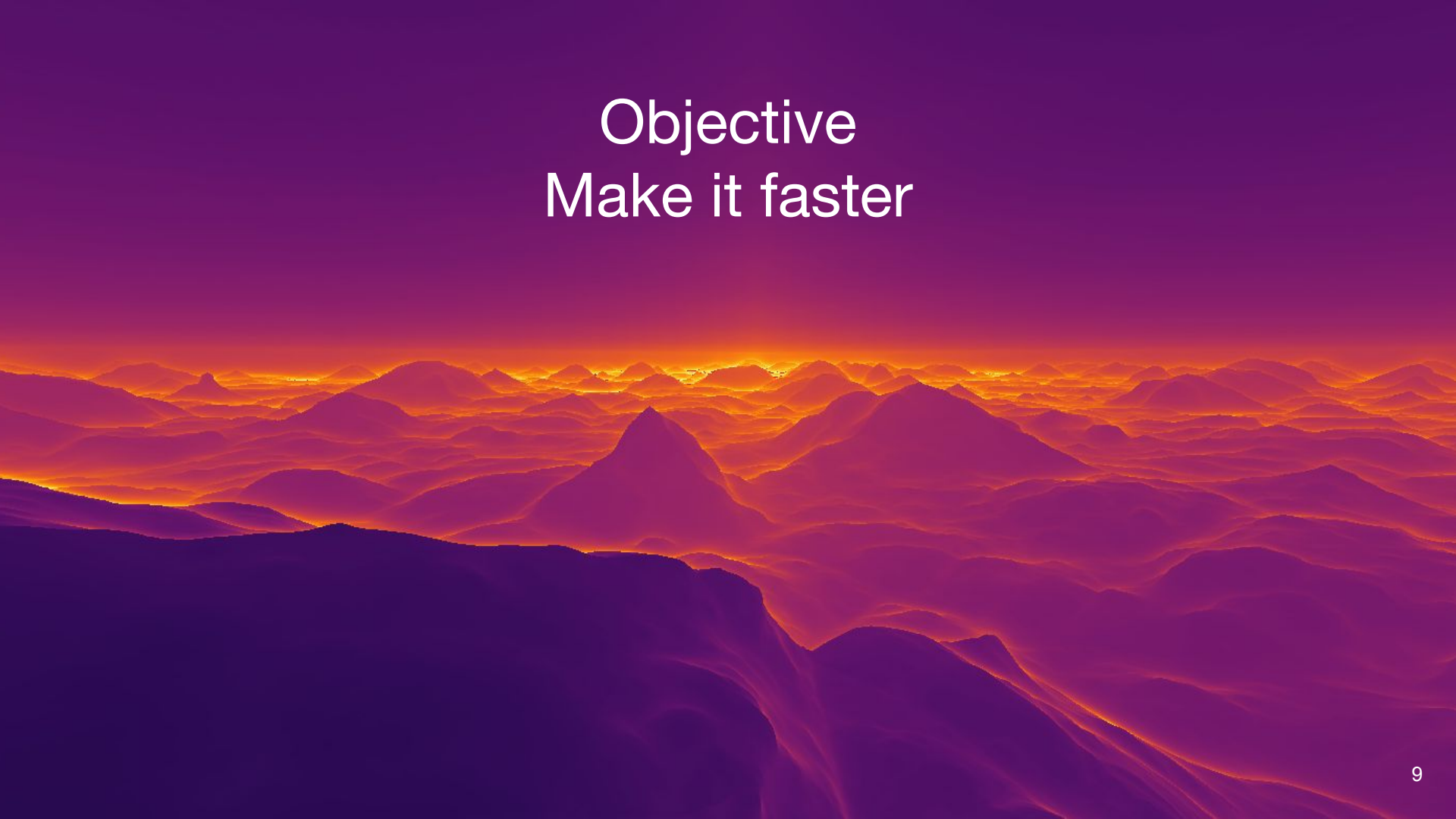
Implicit
SDF
$\| p - c \| - r = 0$



But …

- Evaluation of complex scenes is **very expensive**
- The scene is evaluated multiple times per pixel
- Polygon meshes are faster to render

Performance scales with scene complexity (function)


Objective
Make it faster

SDF

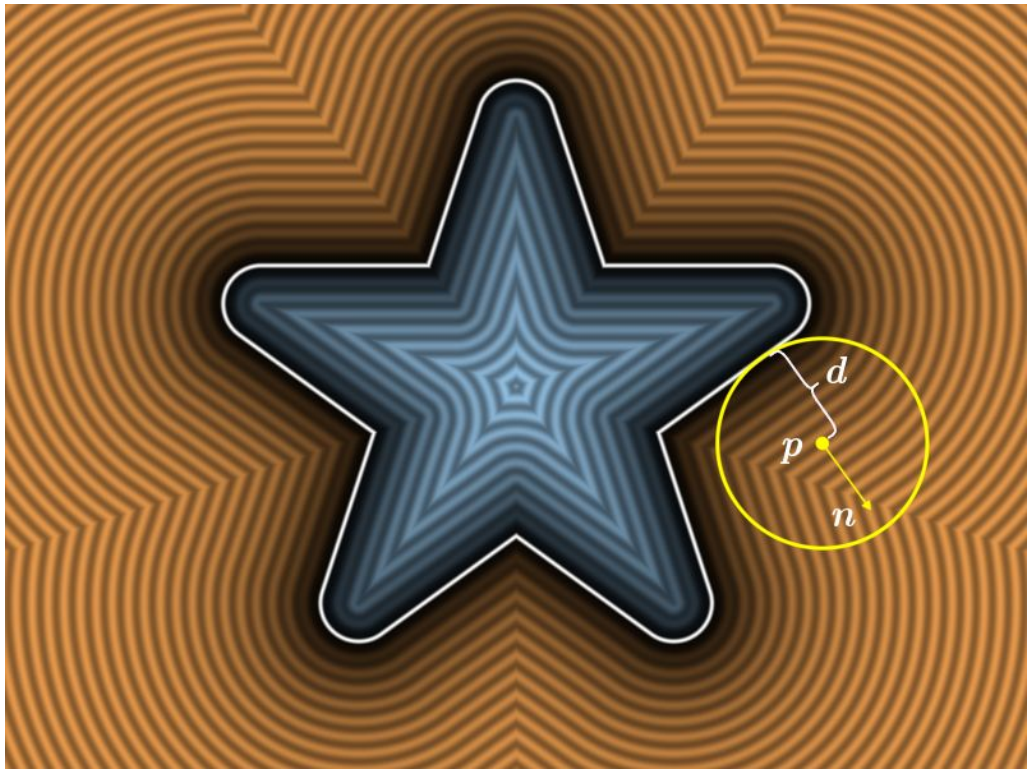
Signed Distance Function

Basic concept:

$$f(p) = d$$

The set of results forms the gradient of the function

The direction of the gradient is the normal direction (***n***)

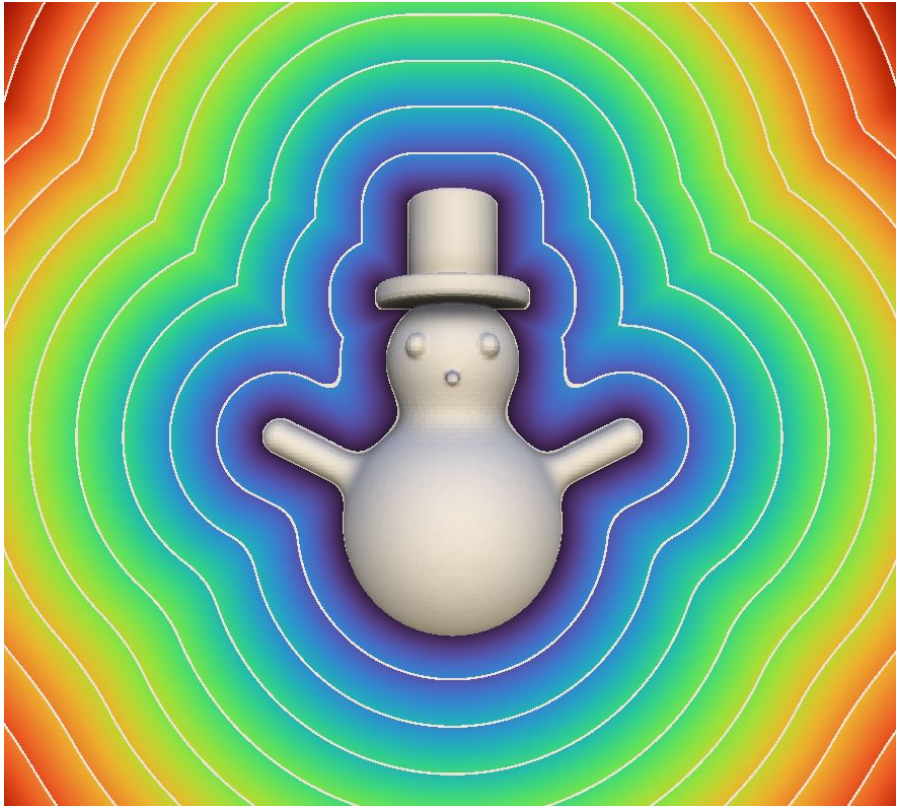


d>0 Exterior d=0 Surface d<0 Interior

SDF

Signed Distance Function

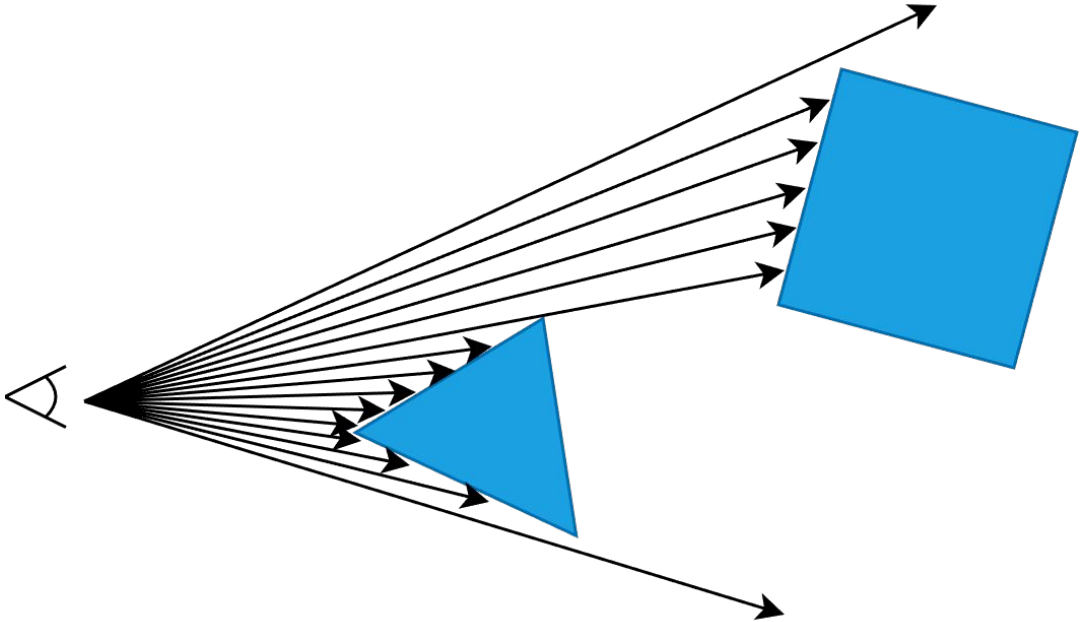
Works the same for 3D SDFs



Depth map

Rendering requires the depth of each pixel

Depth is obtained via *ray tracing*



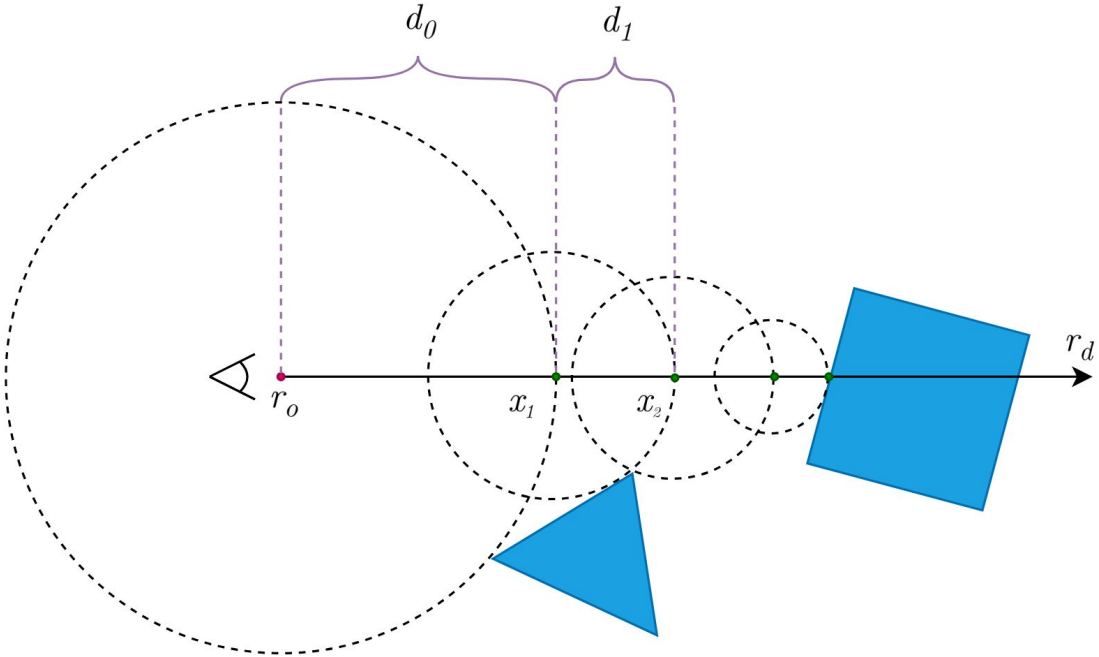
Sphere tracing

How are SDFs visualized?

Iterative ray-marching method:

$$x_i = x_{i-1} + r_d \cdot d_{i-1}$$

The ray advances until it encounters a surface

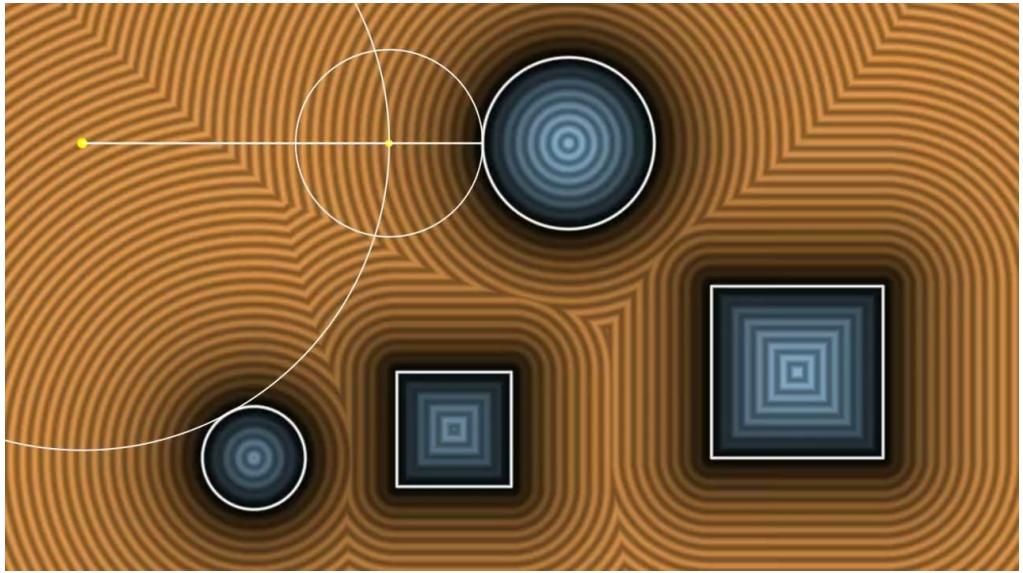


Sphere tracing

Problems:

- Many iterations
- Costly evaluations

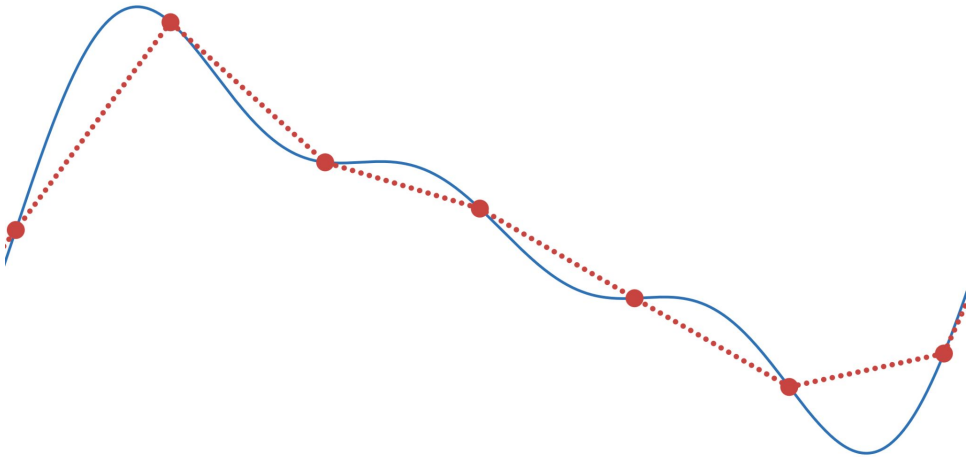
Our optimization goals



Interpolation

A function can be interpolated using samples of its outputs

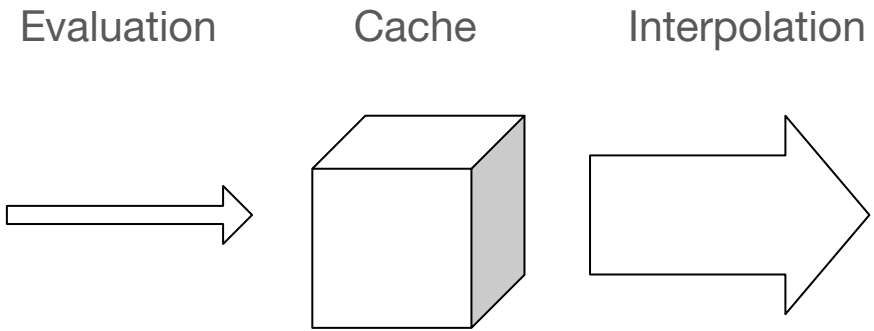
Interpolating a complex SDF is faster than evaluating it



Dense Cache

- Values returned by the scene function are stored
- These values are interpolated to obtain an approximate value

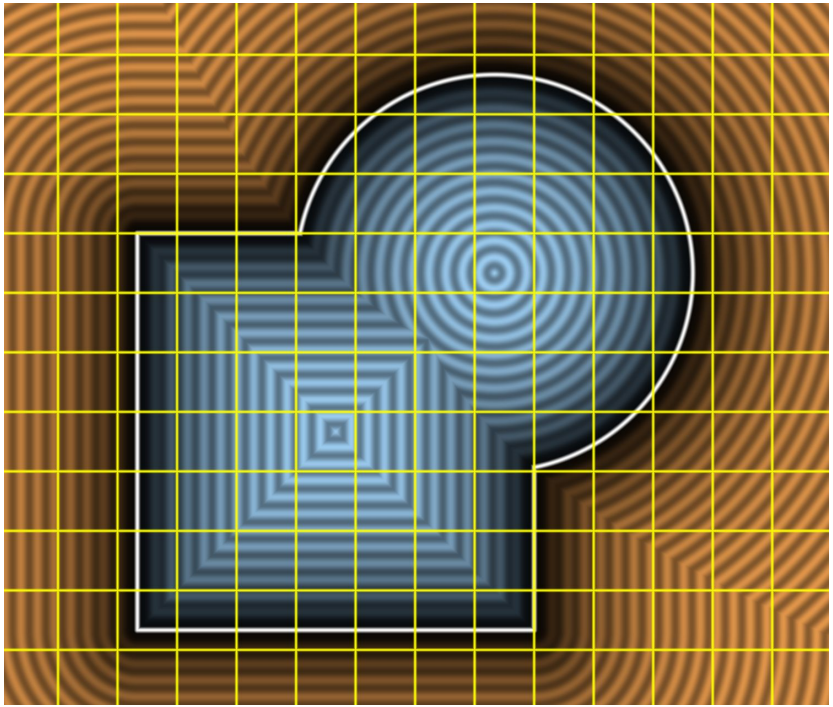
Constant interpolation cost



Dense Cache

How is the cache created?

Points distributed uniformly across the three dimensions.



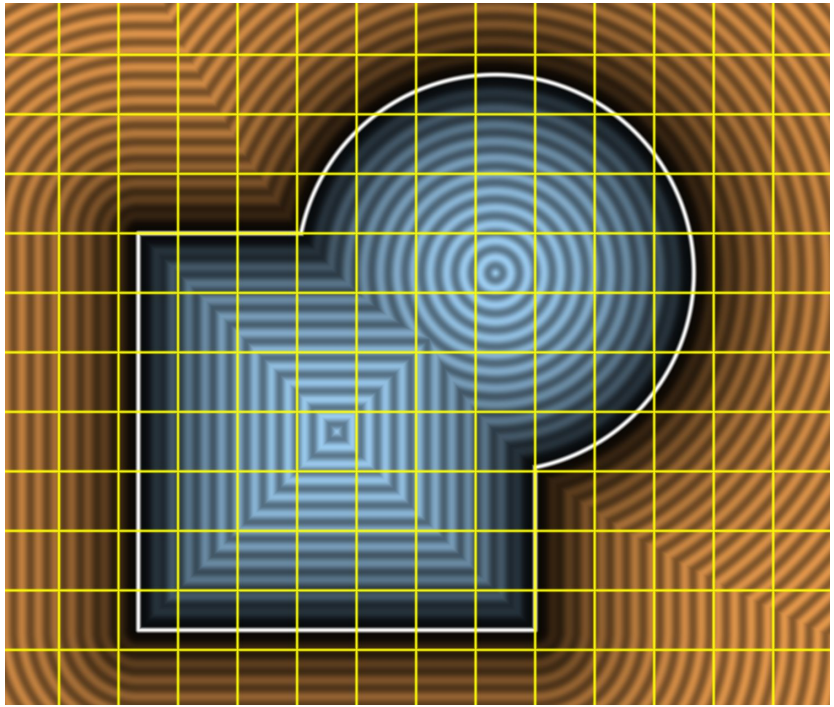
Dense Cache

How is the cache created?

Points distributed uniformly across the three dimensions.

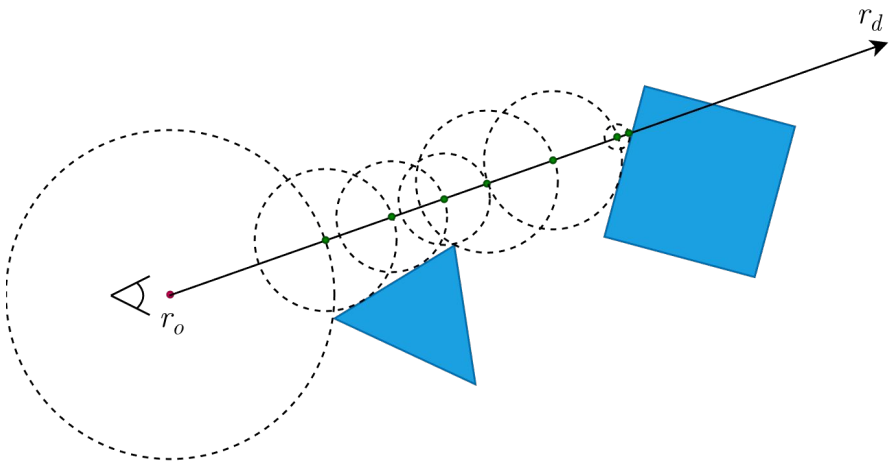
Problem: Cubic memory cost

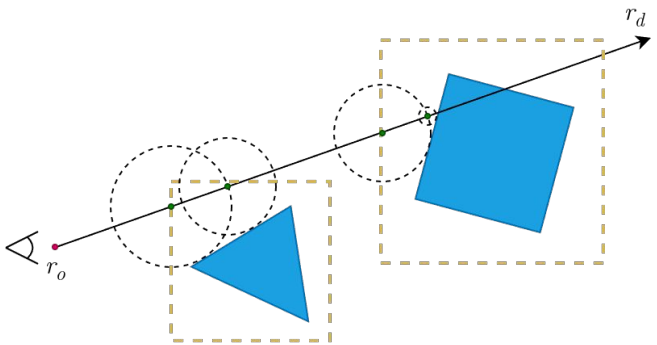
1 m^3	100 m^3	100 km^3
130 MB	$2.4 \cdot 10^{12}$ TB	$2.9 \cdot 10^{39}$ TB

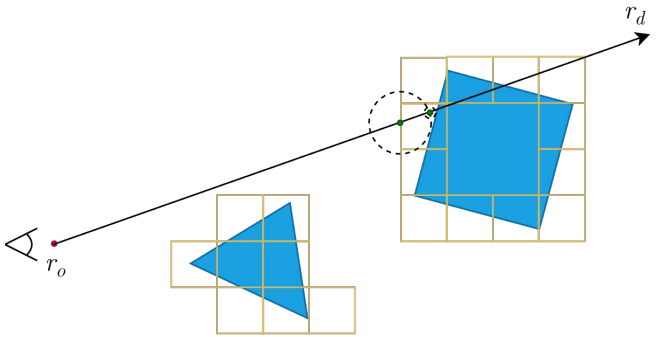


Optimize Sphere Tracing

Reduce the distance traveled by the ray

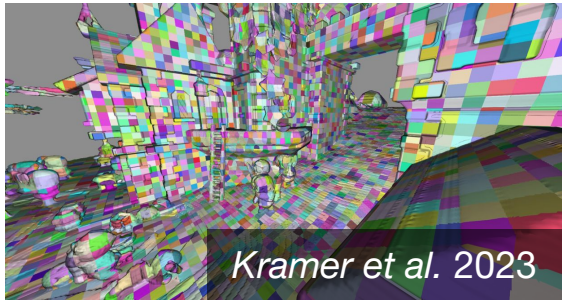




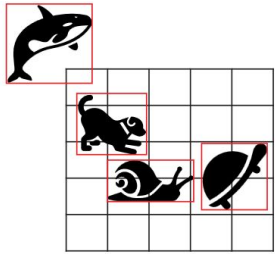


Related works

AMD Brixelizer

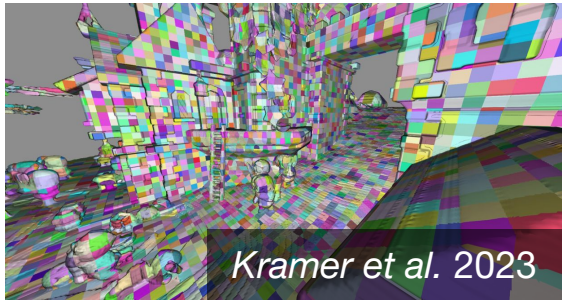


Real Time Sparse Distance Fields For Games
Bricks SDF Visualization Tool

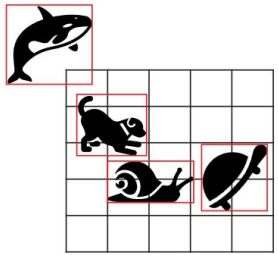


Related works

AMD Brixelizer

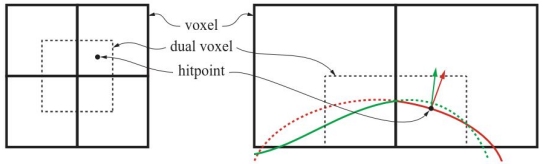


Real Time Sparse Distance Fields For Games
Bricks SDF Visualization Tool



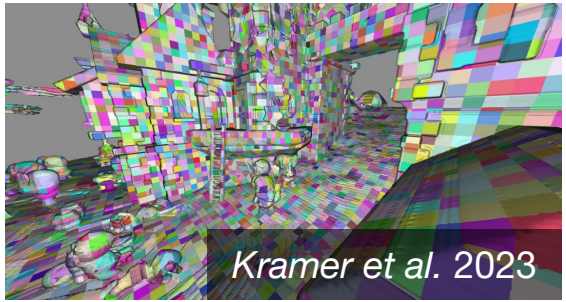
NVIDIA Comparison

Ray Tracing of Signed Distance Function Grids
Comparison of ray-tracing techniques on SDFs

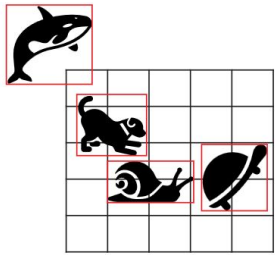


Related works

AMD Brixelizer

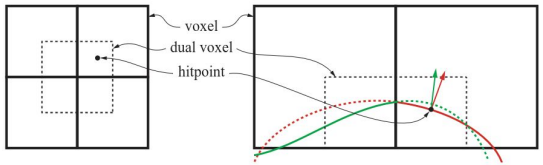


Real Time Sparse Distance Fields For Games
Bricks SDF Visualization Tool

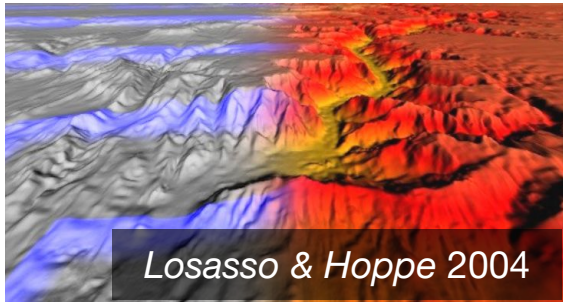


NVIDIA Comparison

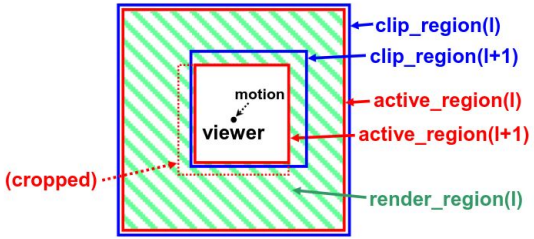
Ray Tracing of Signed Distance Function Grids
Comparison of ray-tracing techniques on SDFs



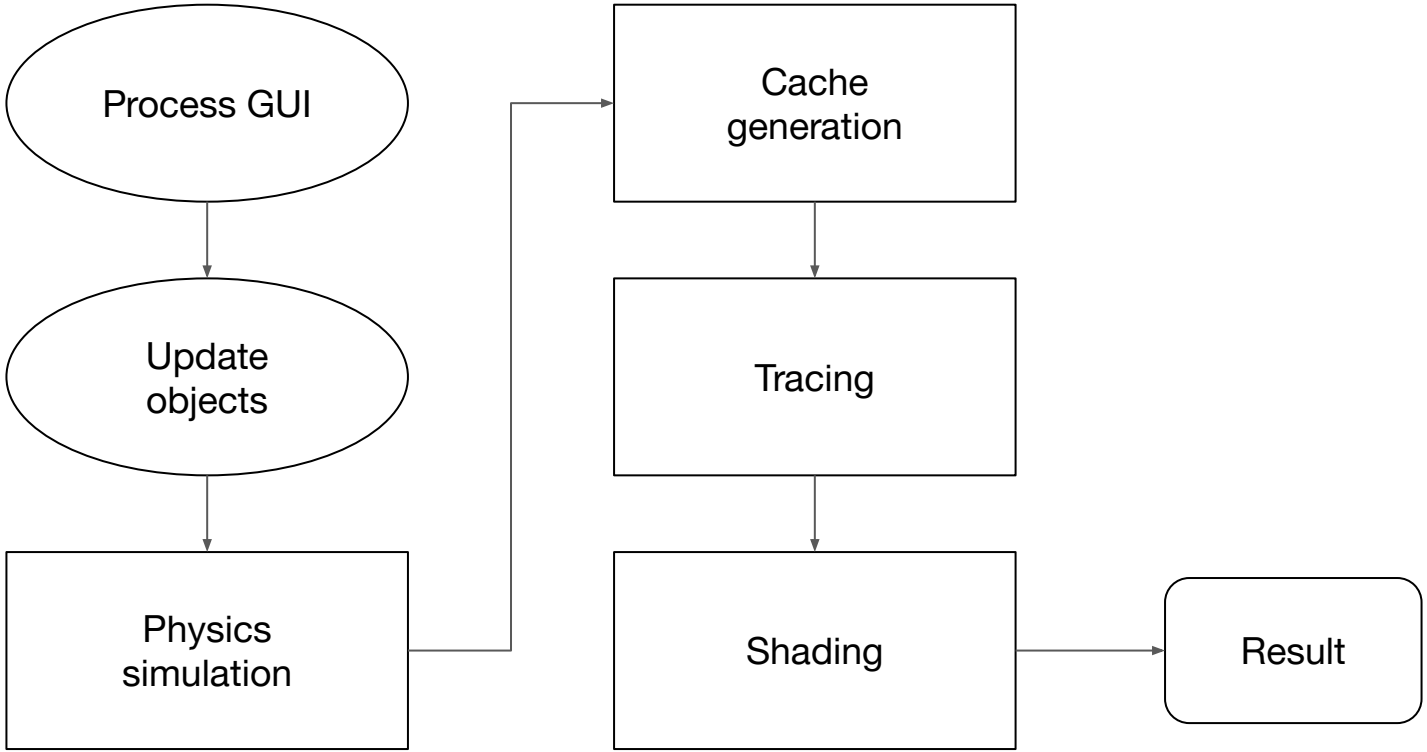
Terrain with LOD



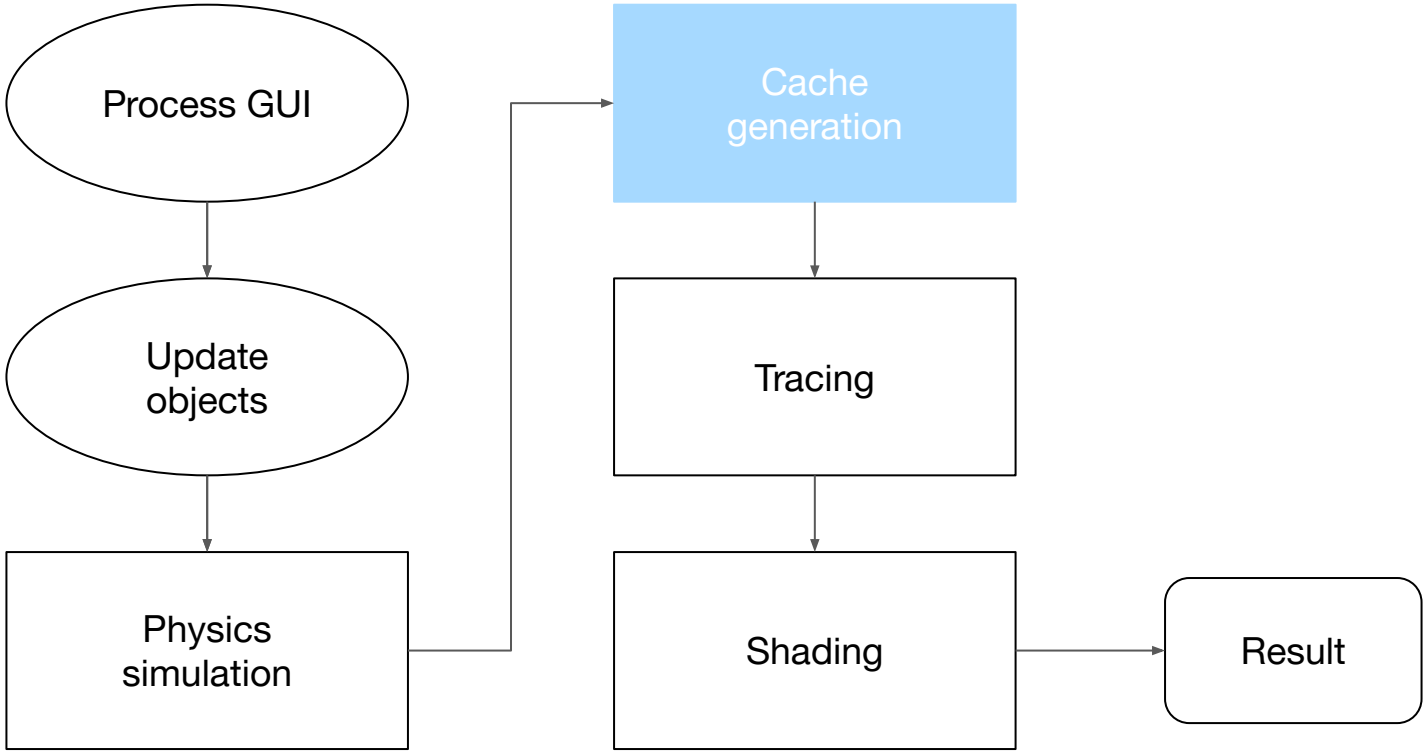
Geometry clipmaps: terrain rendering using nested regular grids



Pipeline


Process GUI
Update objects
Physics simulation
Cache generation
Tracing
Shading
Result

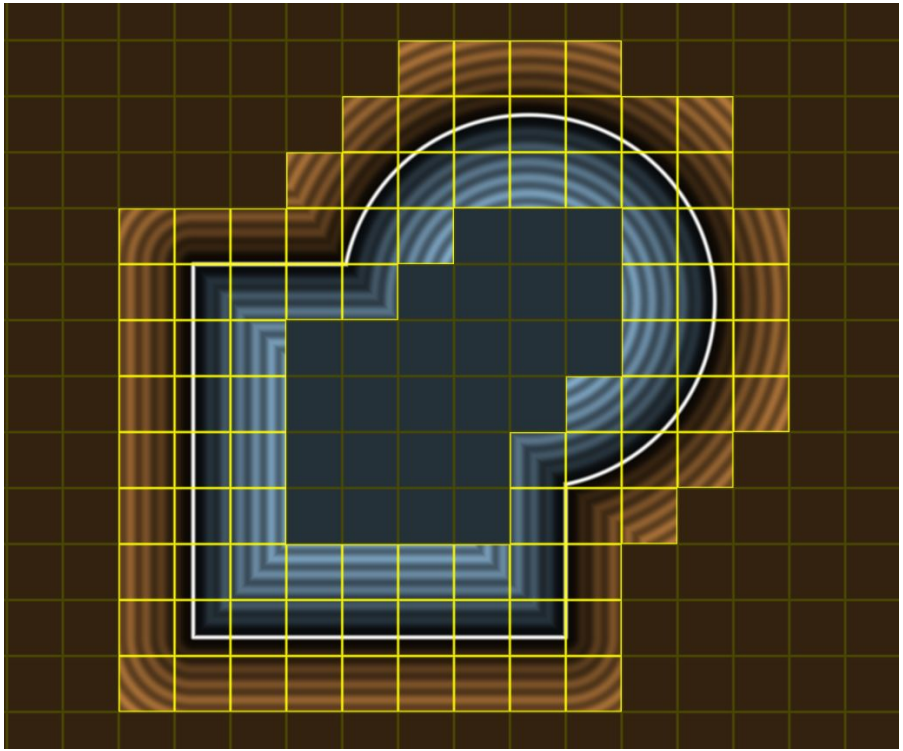
Pipeline


Process GUI
Update objects
Physics simulation
Cache generation
Tracing
Shading
Result

Sparse cache

Only points near the surface are saved

It does not change the final result

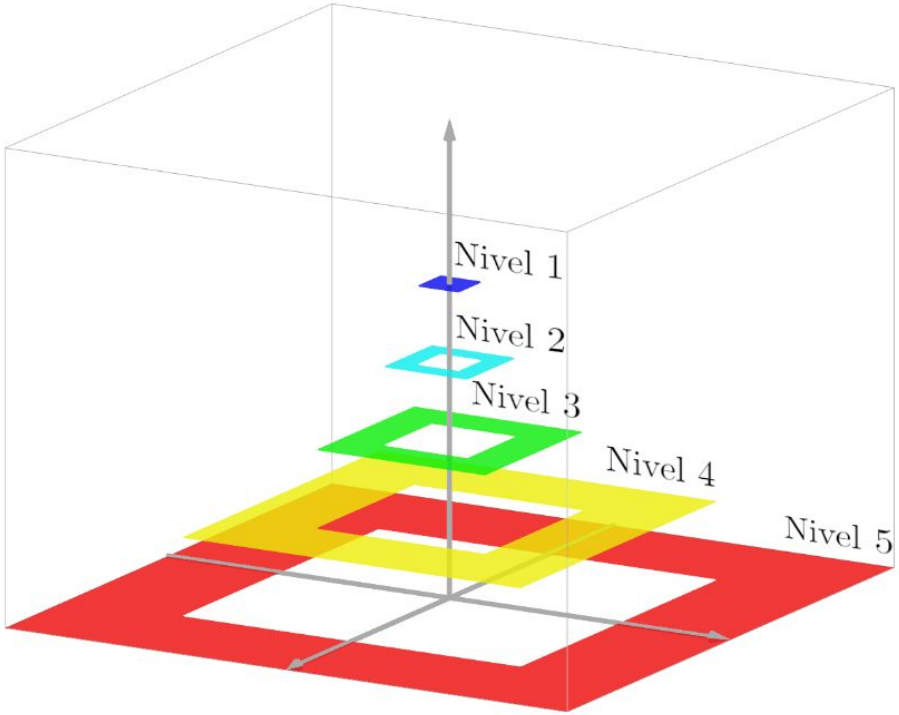


Cache with levels of detail

Level of detail (LOD)

- Camera-centric caching
- The farther away an object is, the less detail it requires

Reduce resolution based on distance from the camera



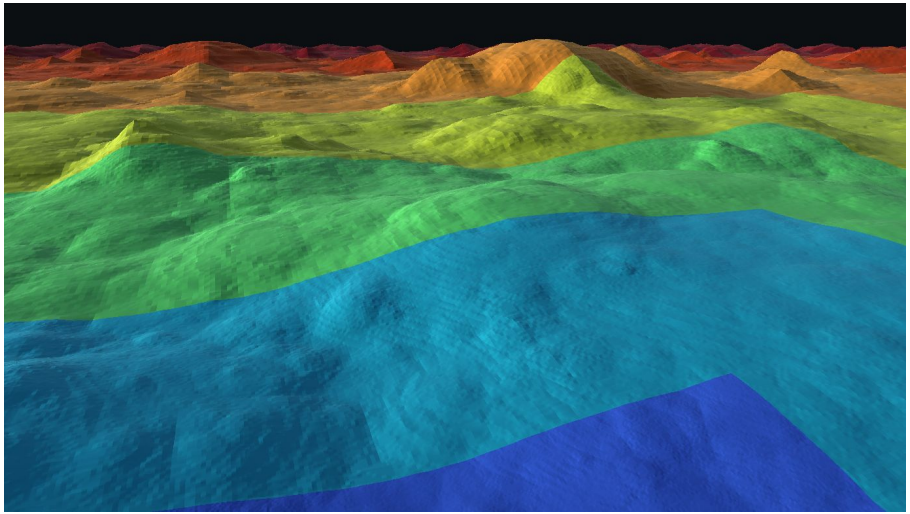
Cache with levels of detail

Level of detail (LOD)

Memory reduction:

$$O(n^3) \Rightarrow O(\log(n))$$

1 m^3	100 m^3	100 km^3
130 MB	$2.4 \cdot 10^{12}$ TB	$2.9 \cdot 10^{39}$ TB
21 MB	**141 MB**	**345 MB**



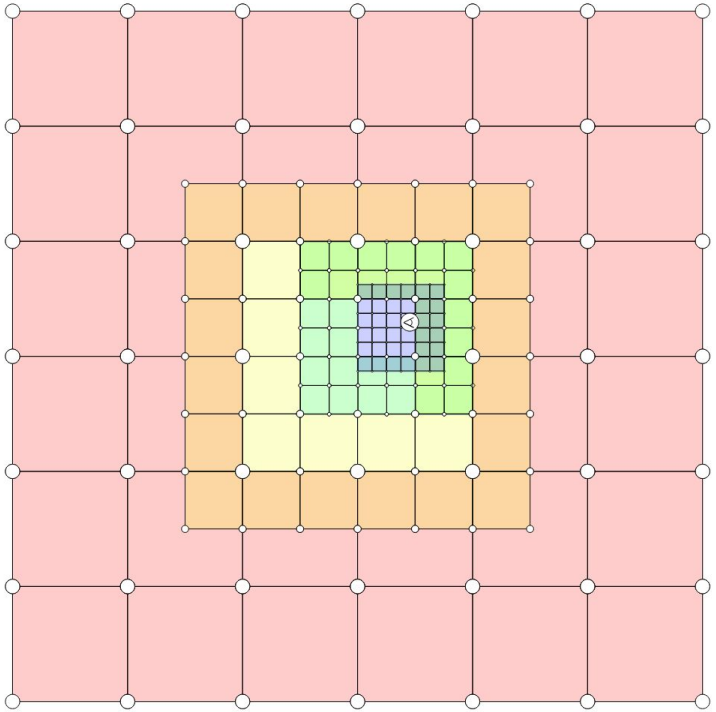
Cache update

Updating the cache is expensive

What triggers a cache update?

- An object in the scene changes
- The camera moves

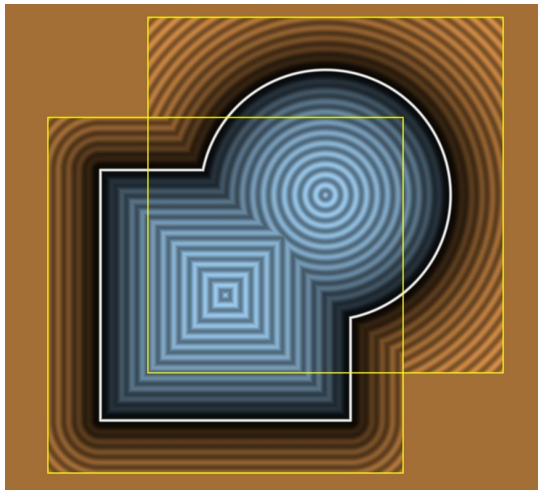
Update only the affected areas



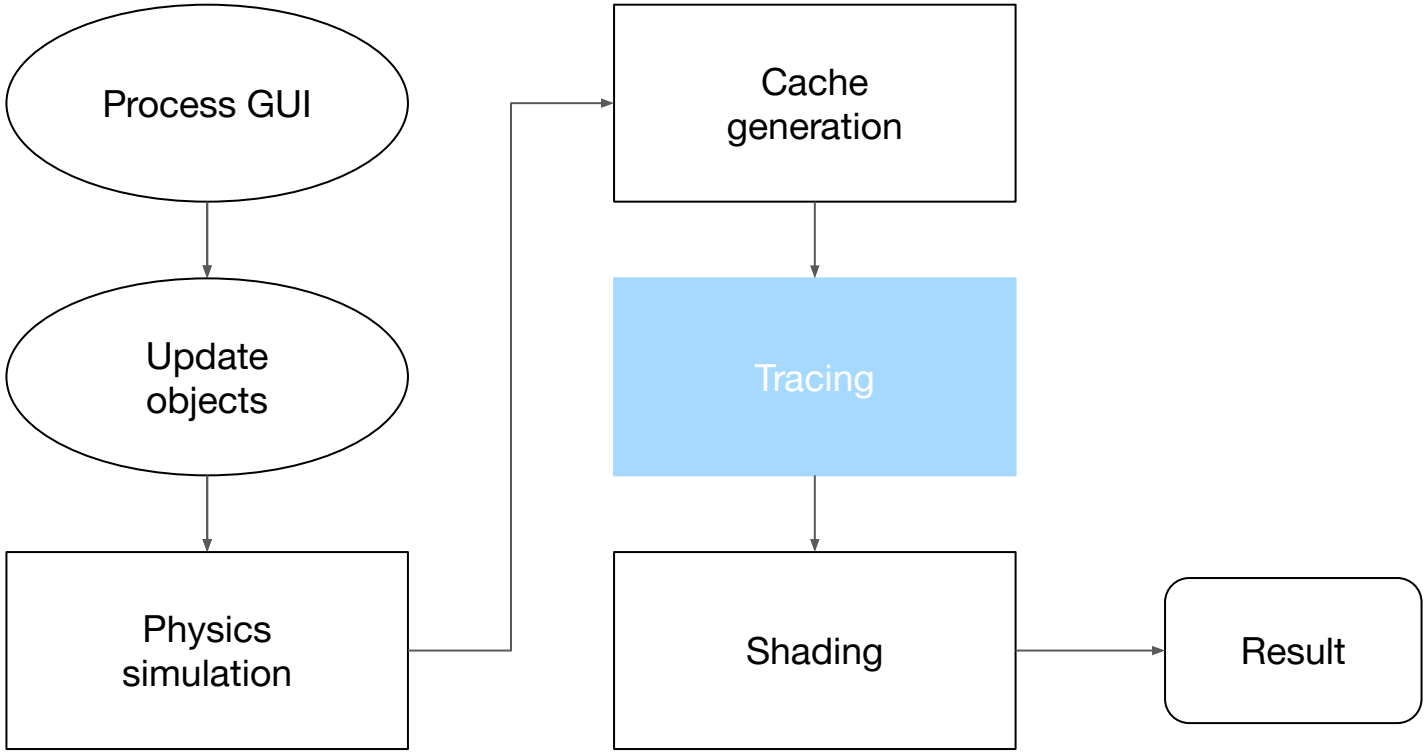
Cache update

The update is performed based on spatial regions:

- Objects are enclosed by an AABB (Axis-Aligned Bounding Box)
- Cache regions change resolution

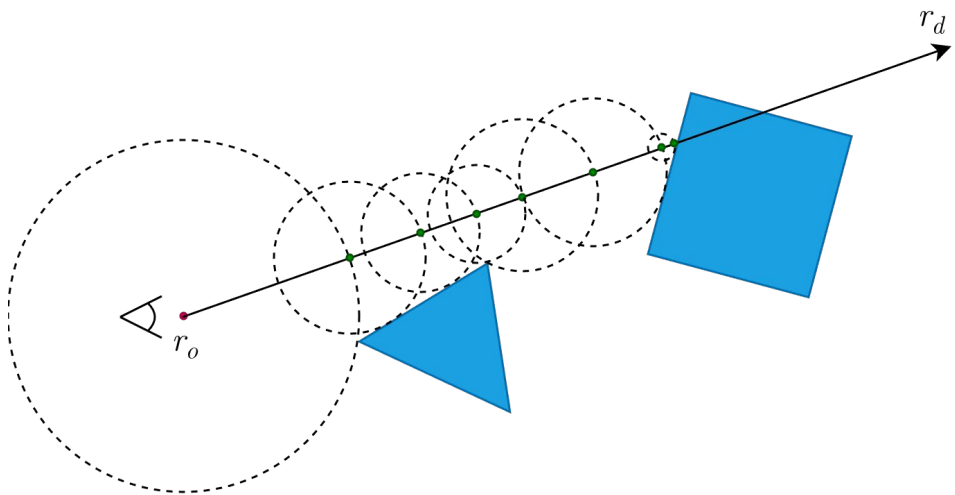


Pipeline


Process GUI
Update objects
Physics simulation
Cache generation
Tracing
Shading
Result

Sphere tracing (Base method)

To reduce the number of steps, we need
to reduce the distance traveled

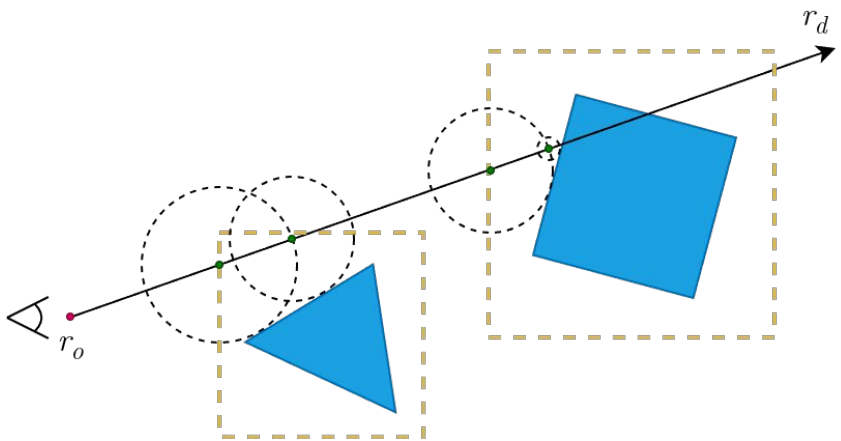


Compute tracing

To reduce the number of steps, we need to reduce the distance traveled

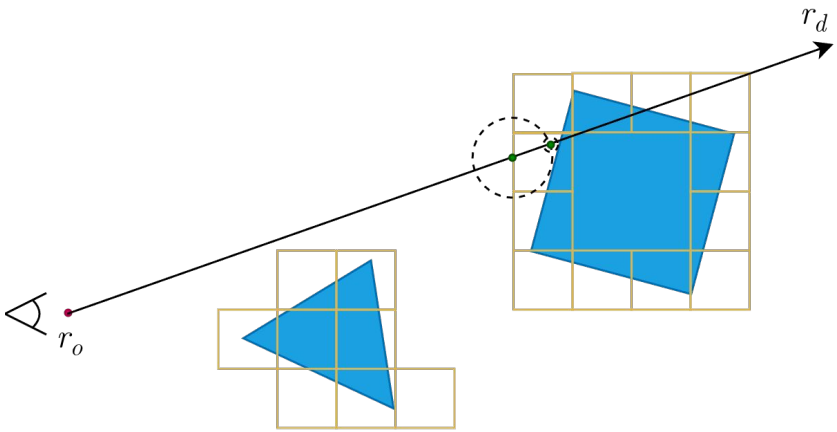
It only executes within the objects' AABBs

The ray starts closer to the surface



Hardware tracing

The regions stored in the cache are used



Hardware tracing

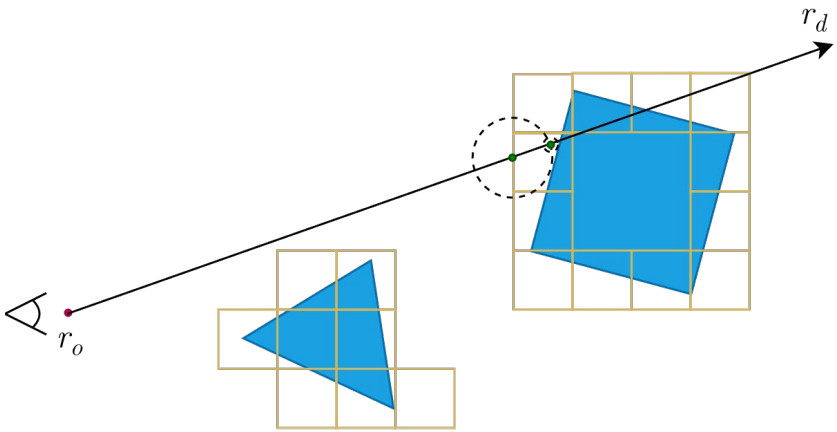
The regions stored in the cache are used

Problem: ***O(n)***

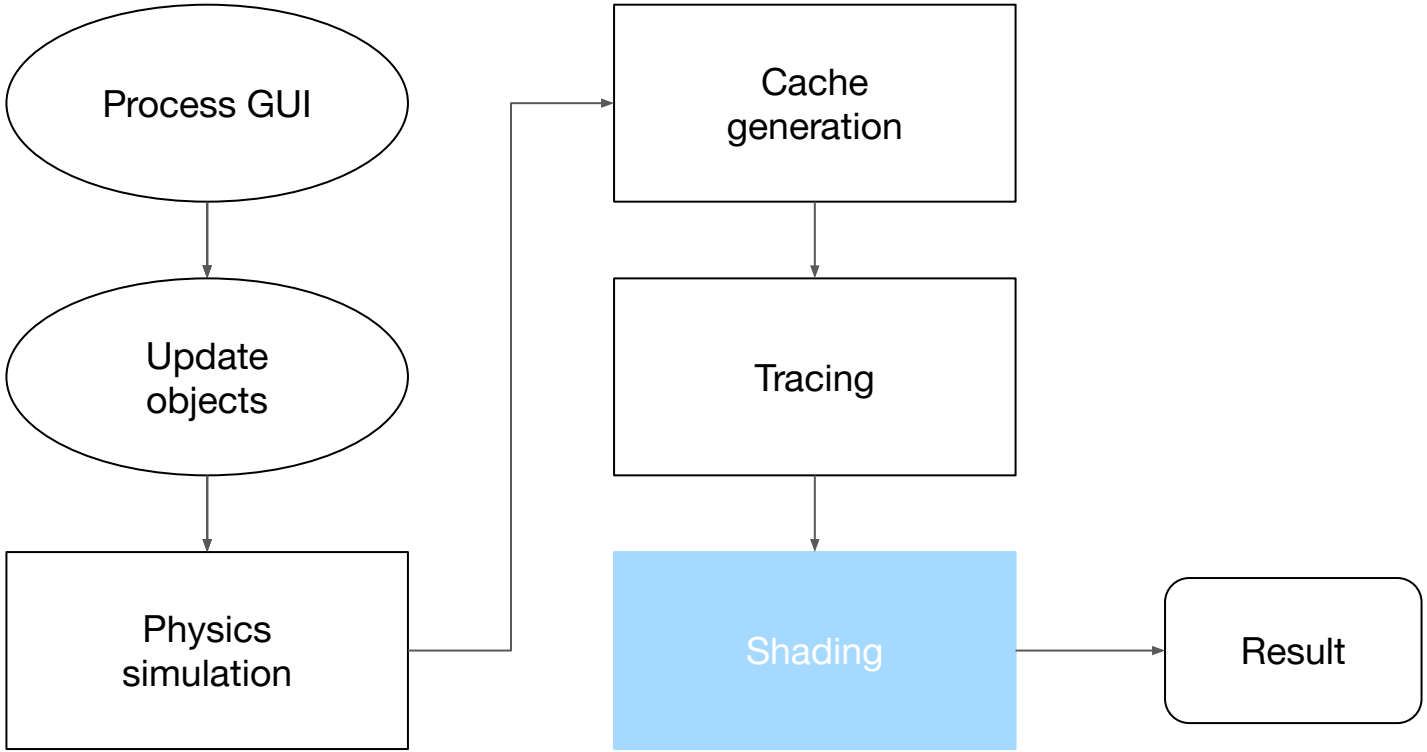
n = Size of cache

Use a hardware acceleration structure:

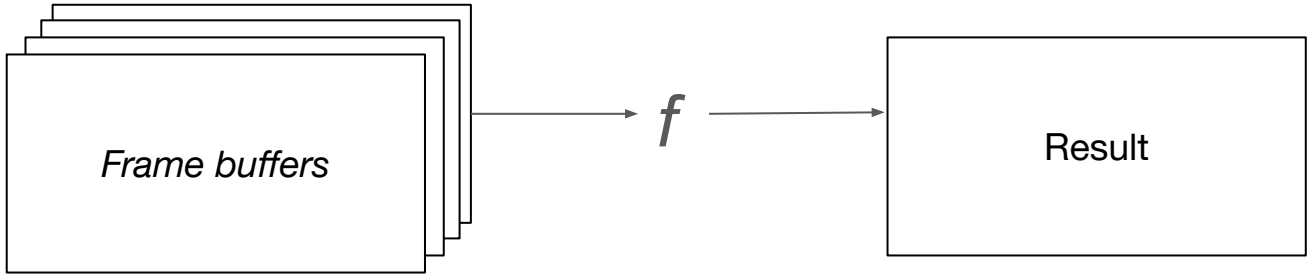
O(log(n))



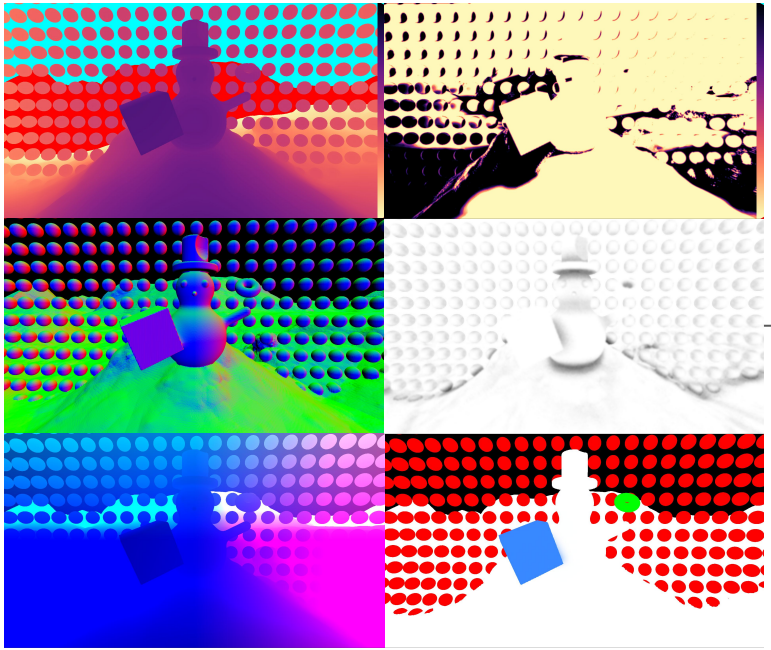
Pipeline


Process GUI
Update objects
Physics simulation
Cache generation
Tracing
Shading
Result

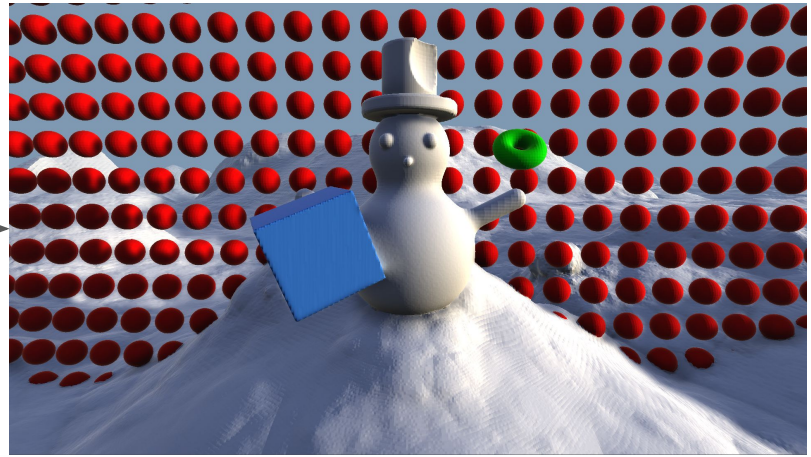
Shading



Shading

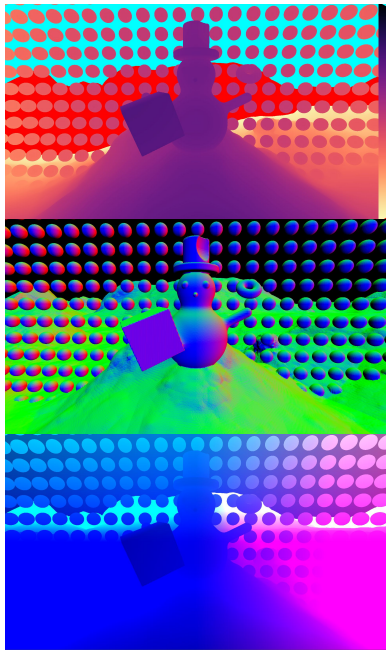


f



Shading

We have already obtained these frame buffers during the **tracing phase**



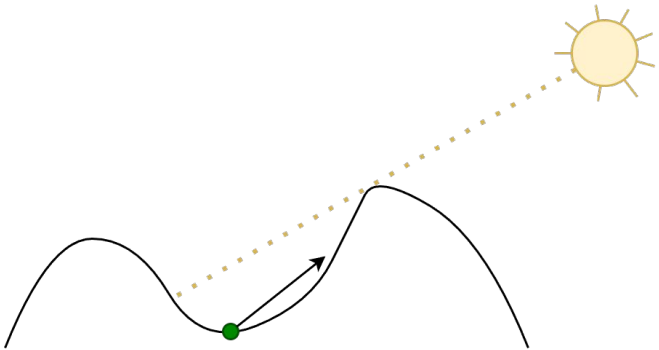
Depth

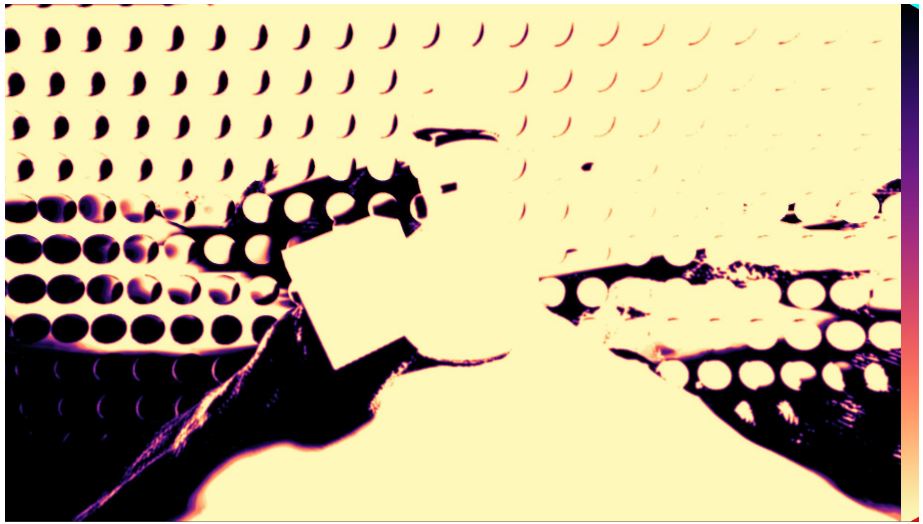
Normal

Relative position

Direct shadows

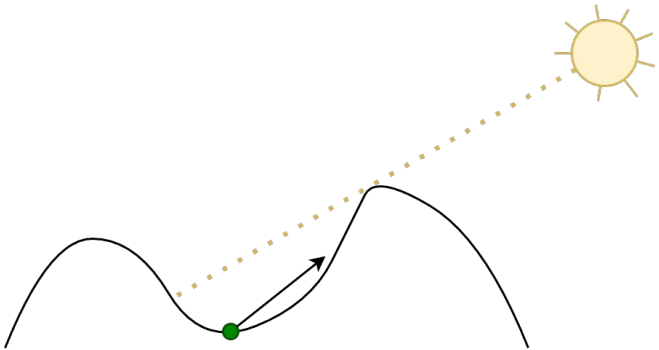
Directional lights create direct shadows



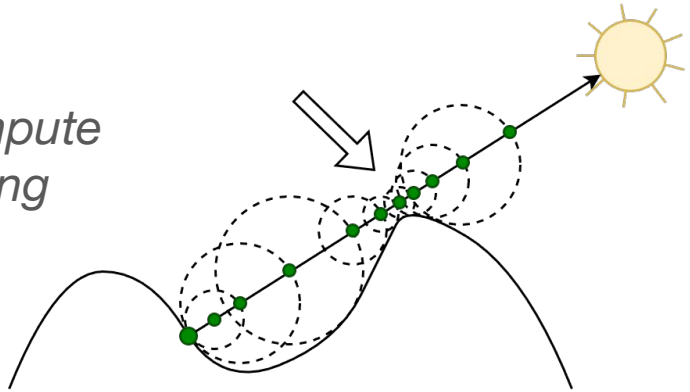


Direct shadows

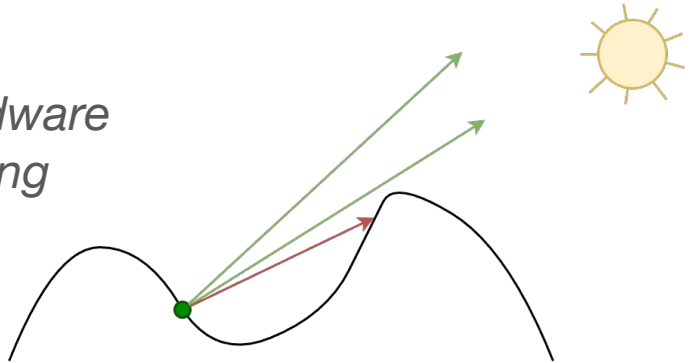
Directional lights create direct shadows



Compute tracing

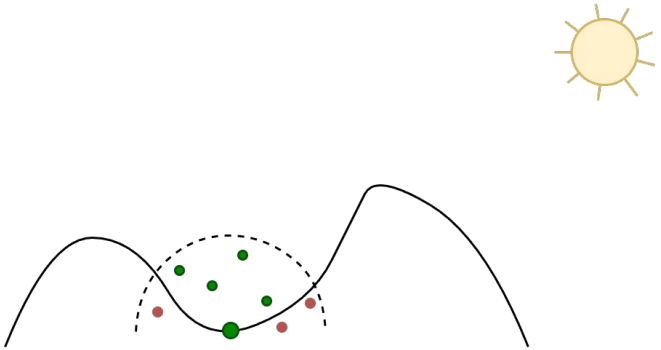


Hardware tracing



Ambient occlusion

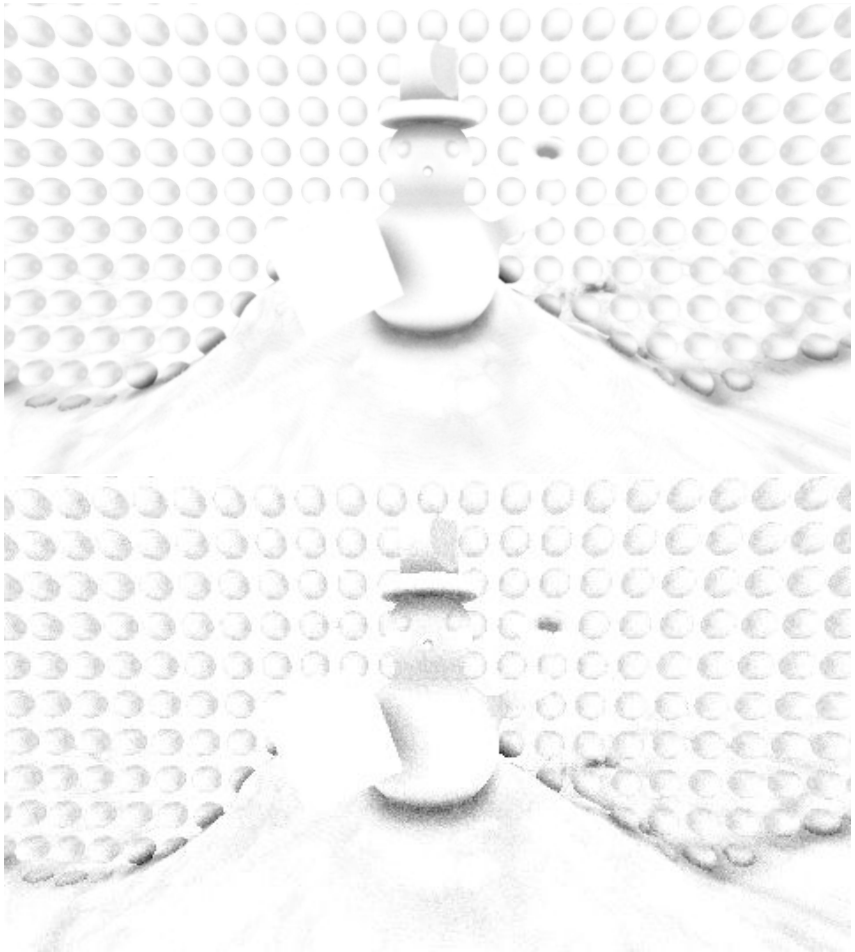
Nearby surfaces create indirect shadows



Downsampling

Reduce shadow resolution

Proportional cost reduction

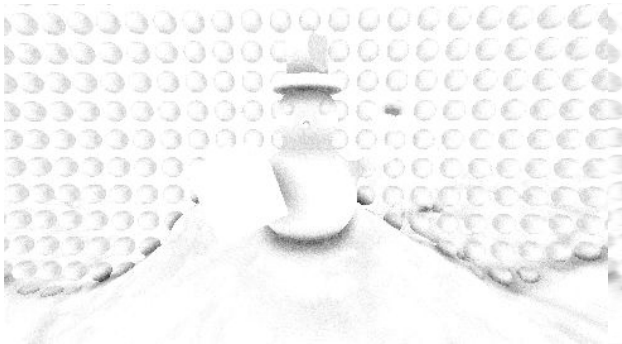


Downsampling + Blur

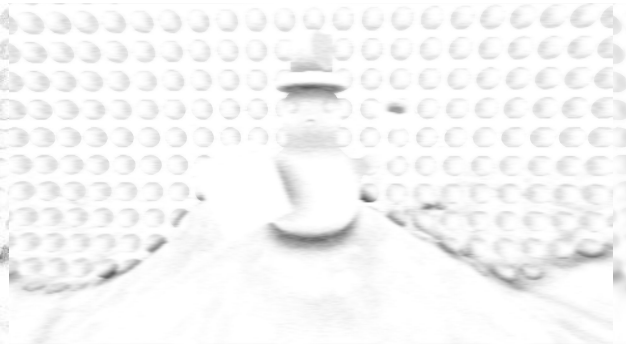
Solution: Blur pixels to simulate high resolution

The *Bilateral Blur* smoothing filter is used

Original



Horizontal pass



Vertical pass (final)

Downsampling + Blur

High definition

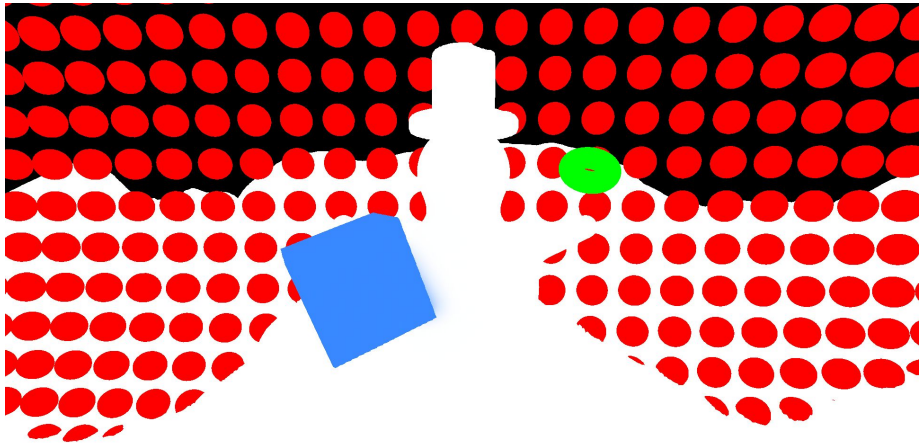
It also applies to direct shadows

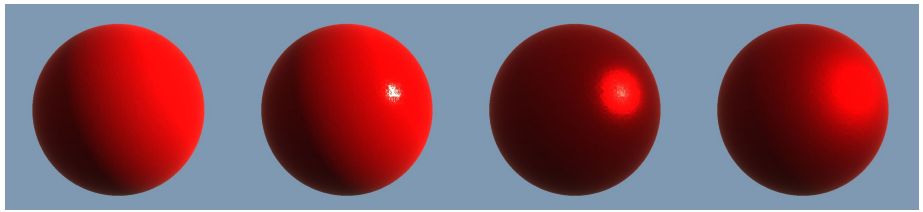
Downsampling + Blur

Materials

A material has 3 parameters:

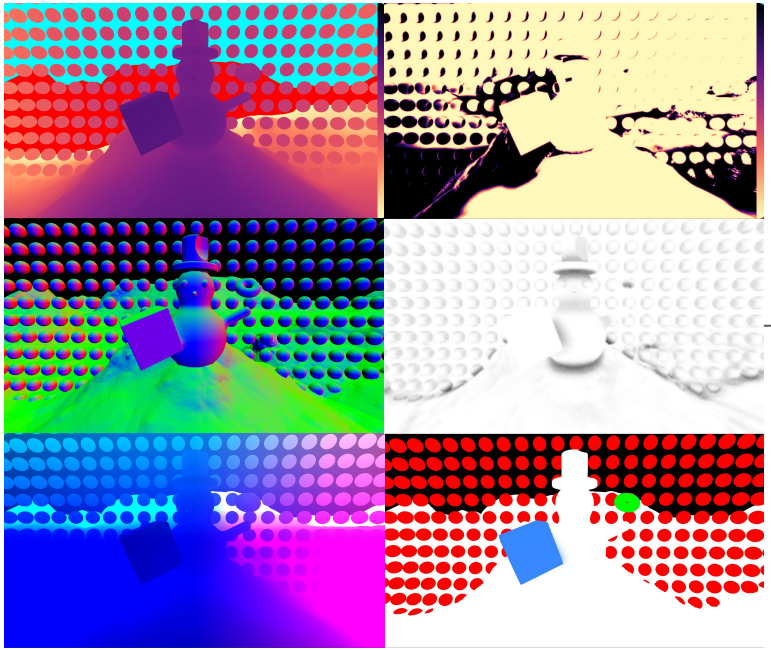
- *Albedo*
- *Roughness*
- *Metalness*



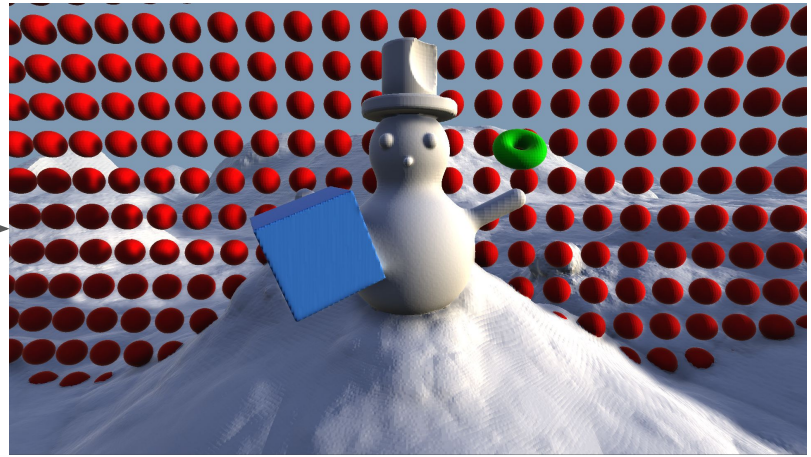


Roughness	80%	10%	20%	80%
Metalness	0%	0%	100%	100%

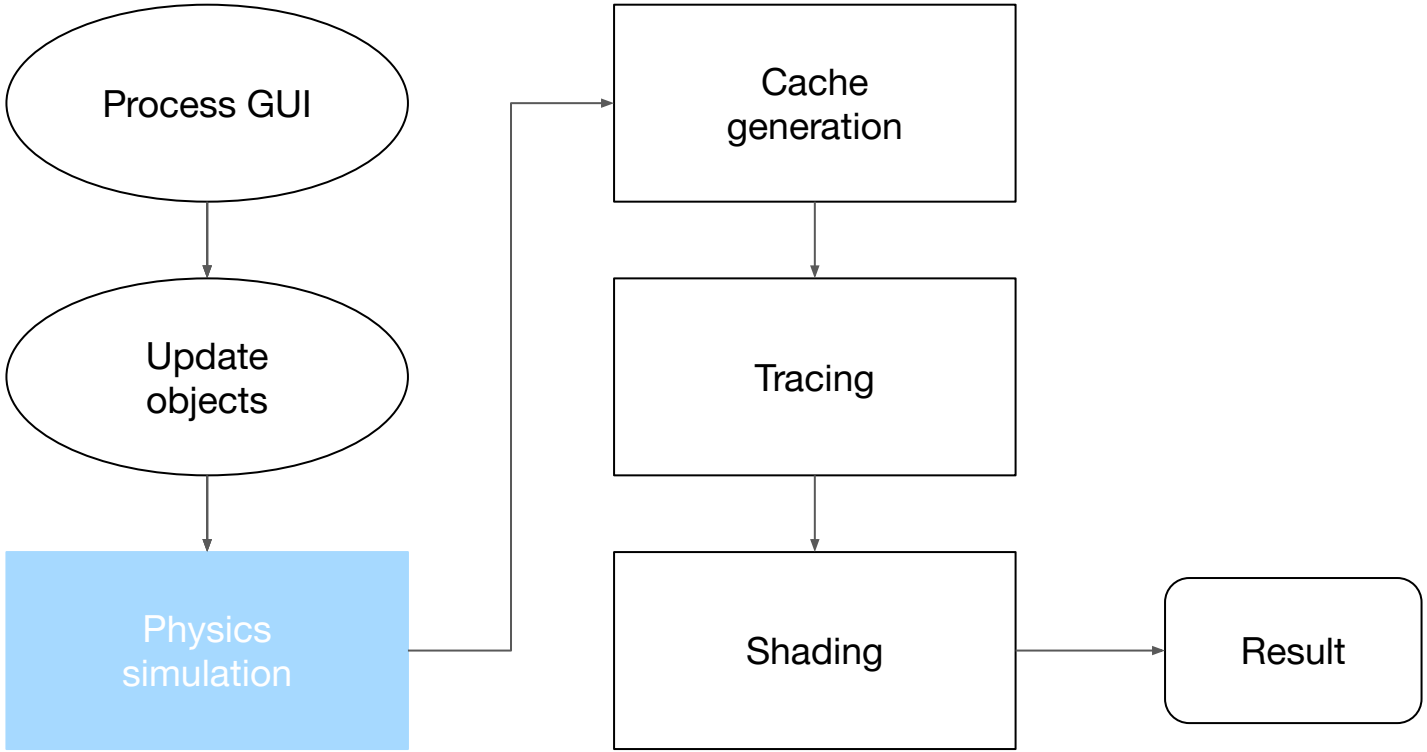
Shading



f

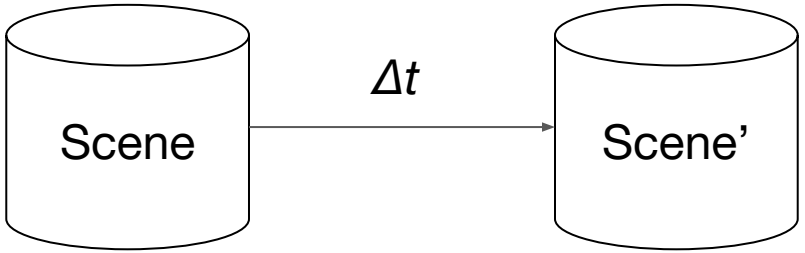


Pipeline


Process GUI
Update objects
Physics simulation
Cache generation
Tracing
Shading
Result

Physics simulation

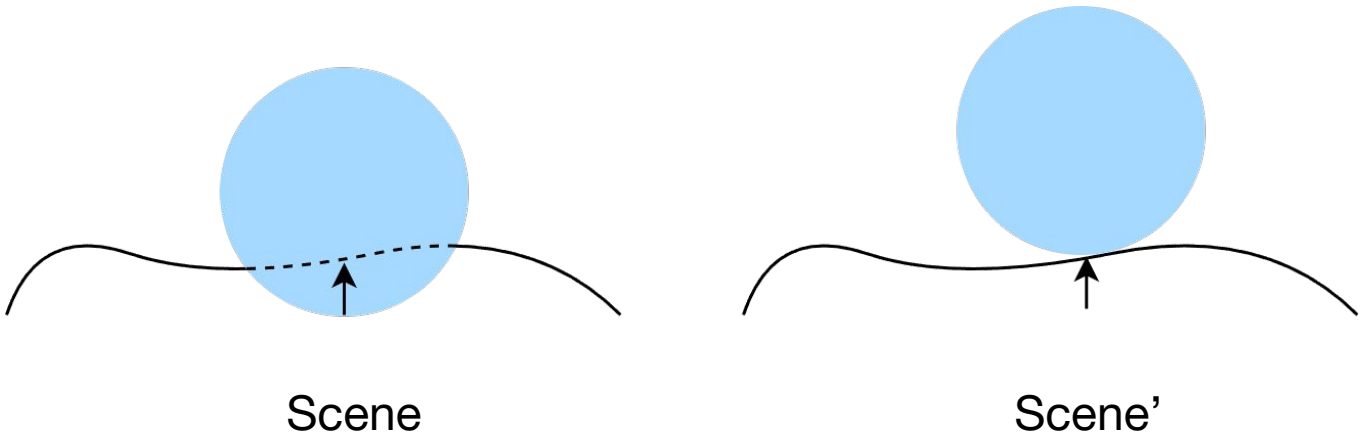
Simulate object dynamics as realistically as possible, and modify their position and rotation



XPBD

Mathematical model based on position correction

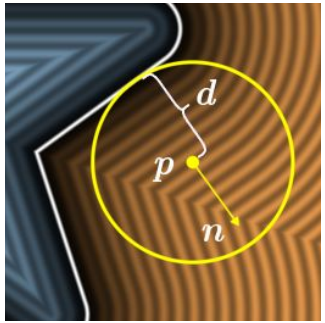
Constraints are satisfied through position corrections

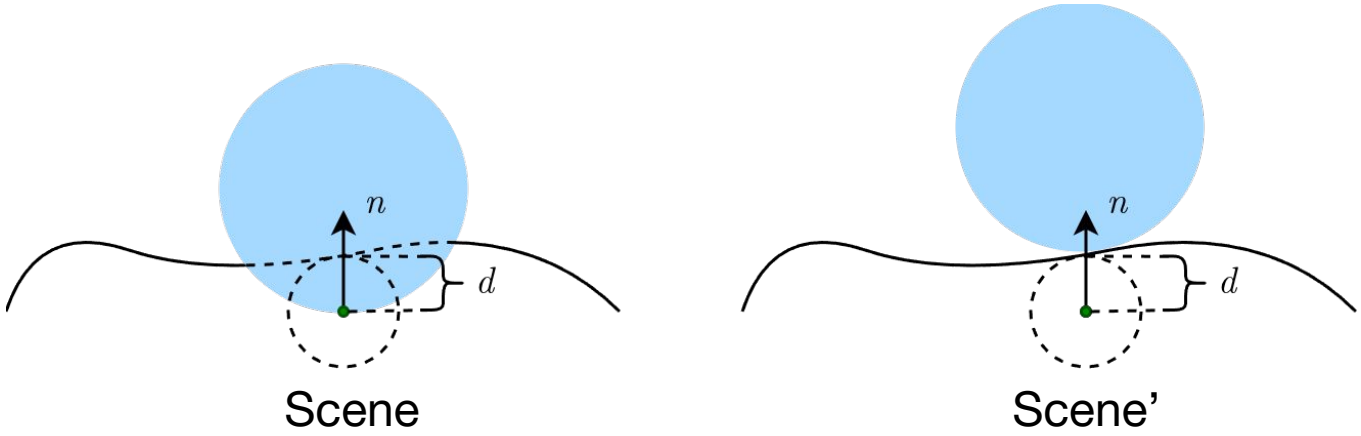


XPBD

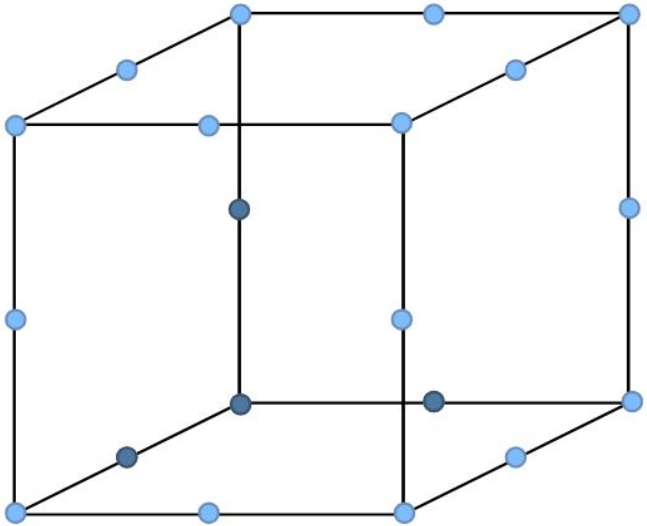
Collision correction vector = SDF evaluation · normal vector

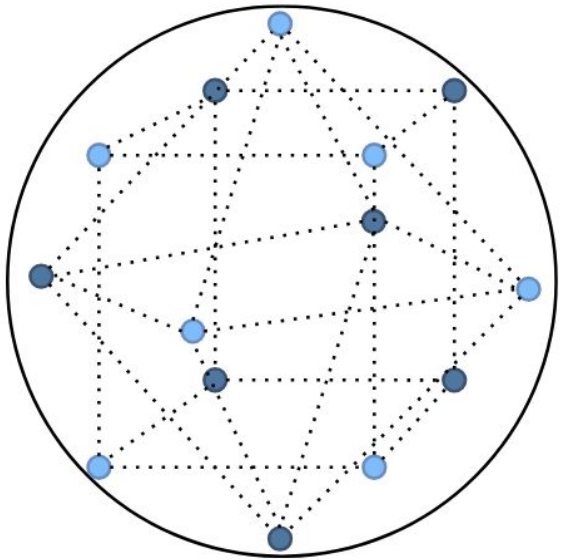
Much faster than a triangle mesh

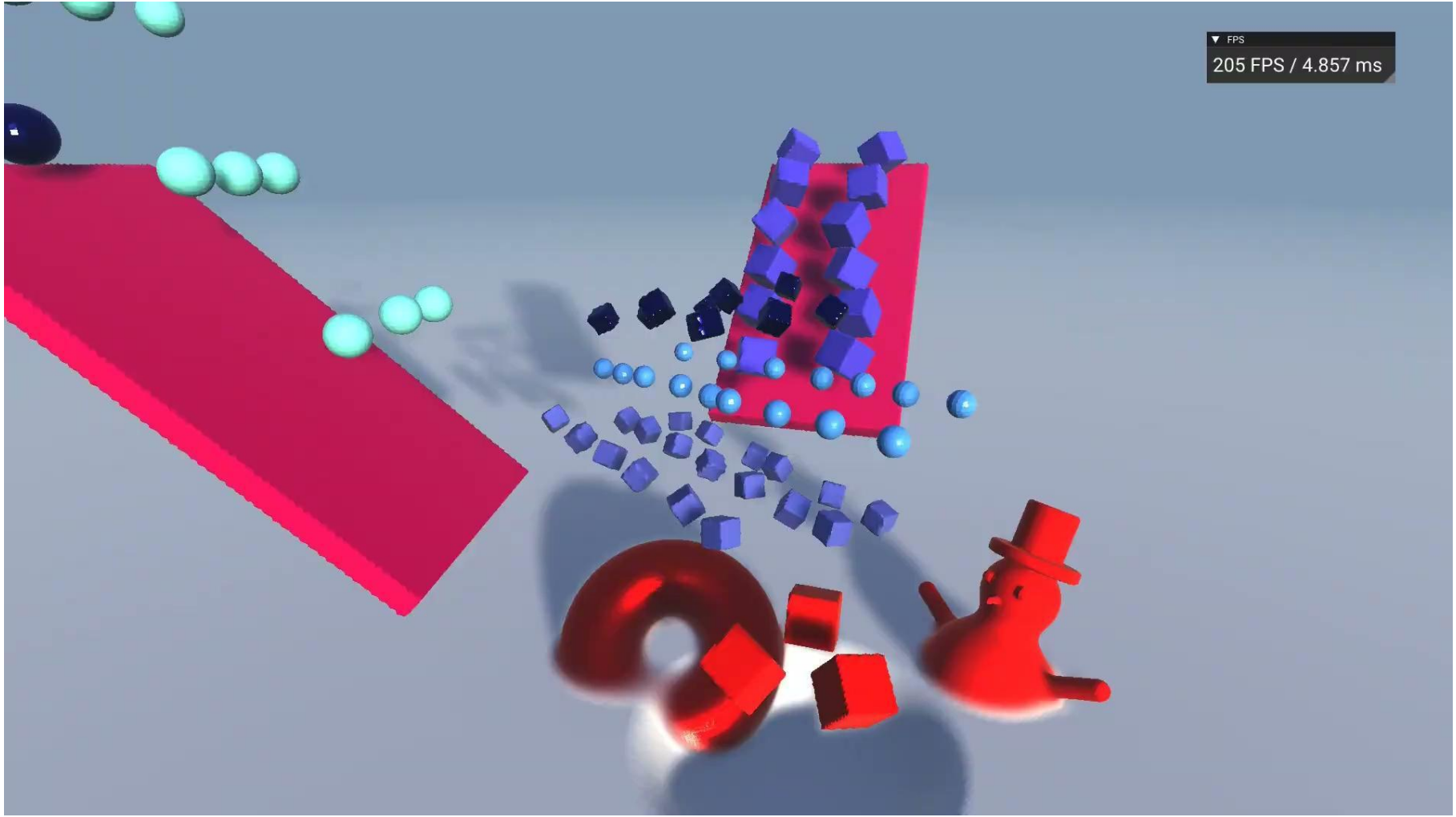




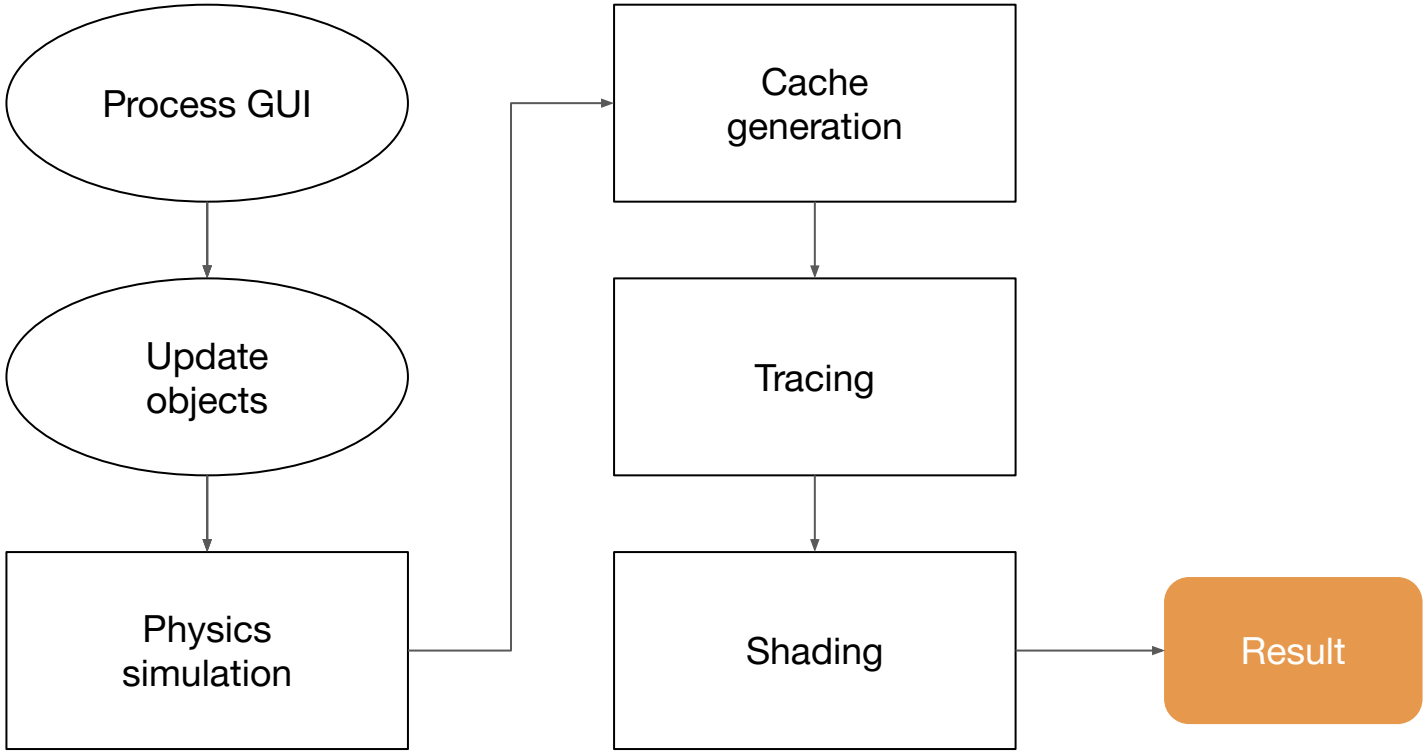
Sampling points





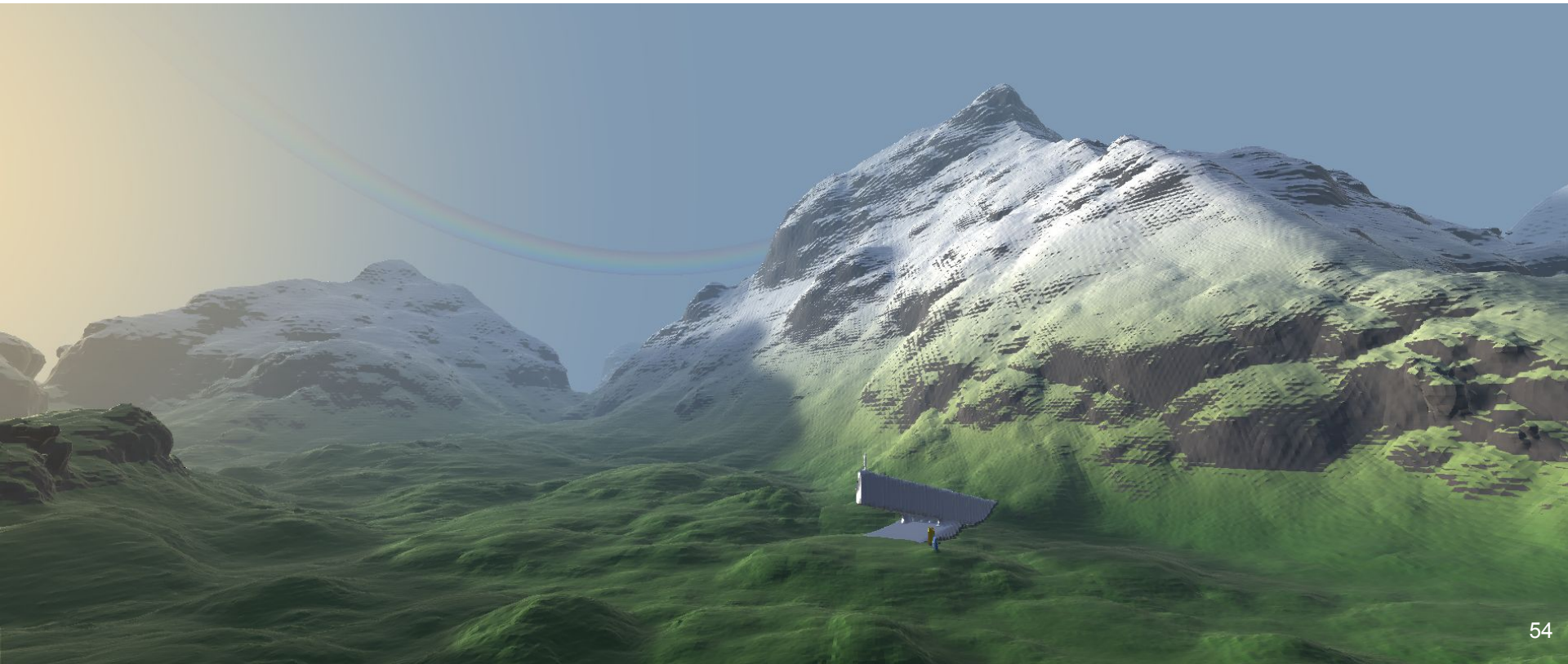

FPS
205 FPS / 4.857 ms

Pipeline


Process GUI
Update objects
Physics simulation
Cache generation
Tracing
Shading
Result

Visual demo

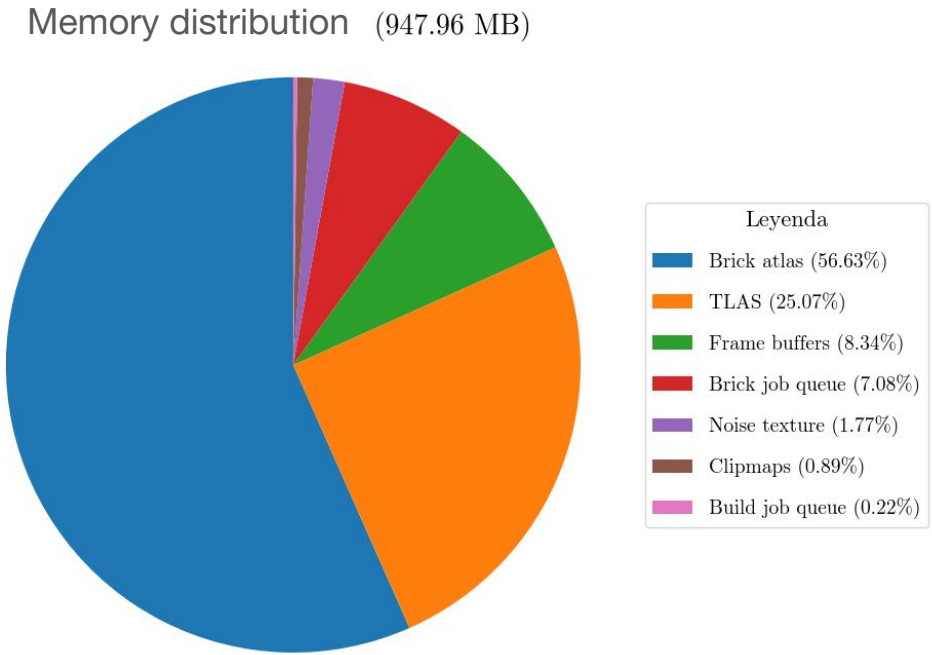
36 times faster | 6 FPS -> 242 FPS
0.17 Perceptual error FLIP



Memory

Less than 1GB

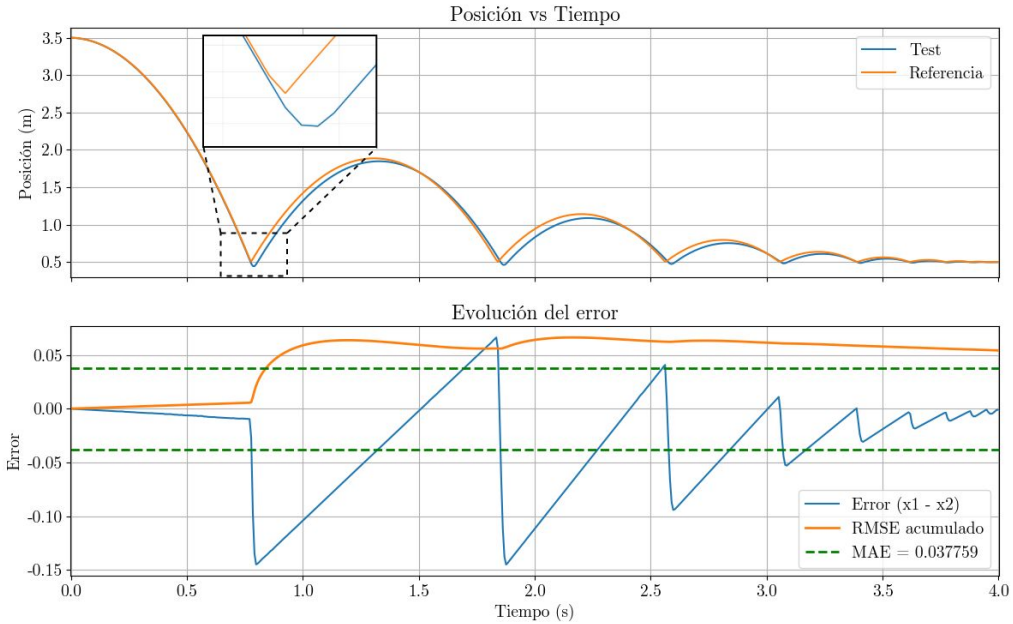
Dense Cache	$6.33 \cdot 10^{17}$ TB
Sparse Cache	$3.16 \cdot 10^{17}$ TB
Sparse + LOD Cache	546 MB



Physics

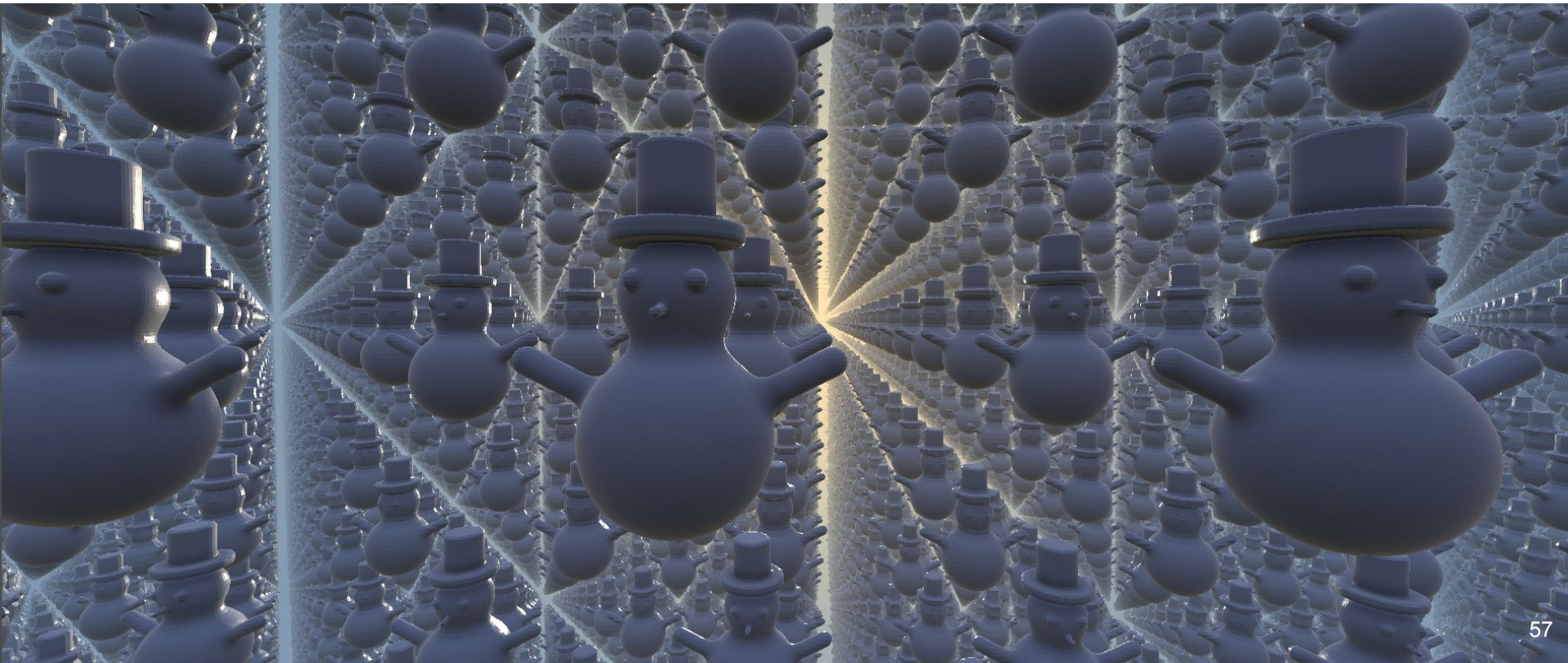
Precise and efficient physics

66 active objects @ +100 FPS



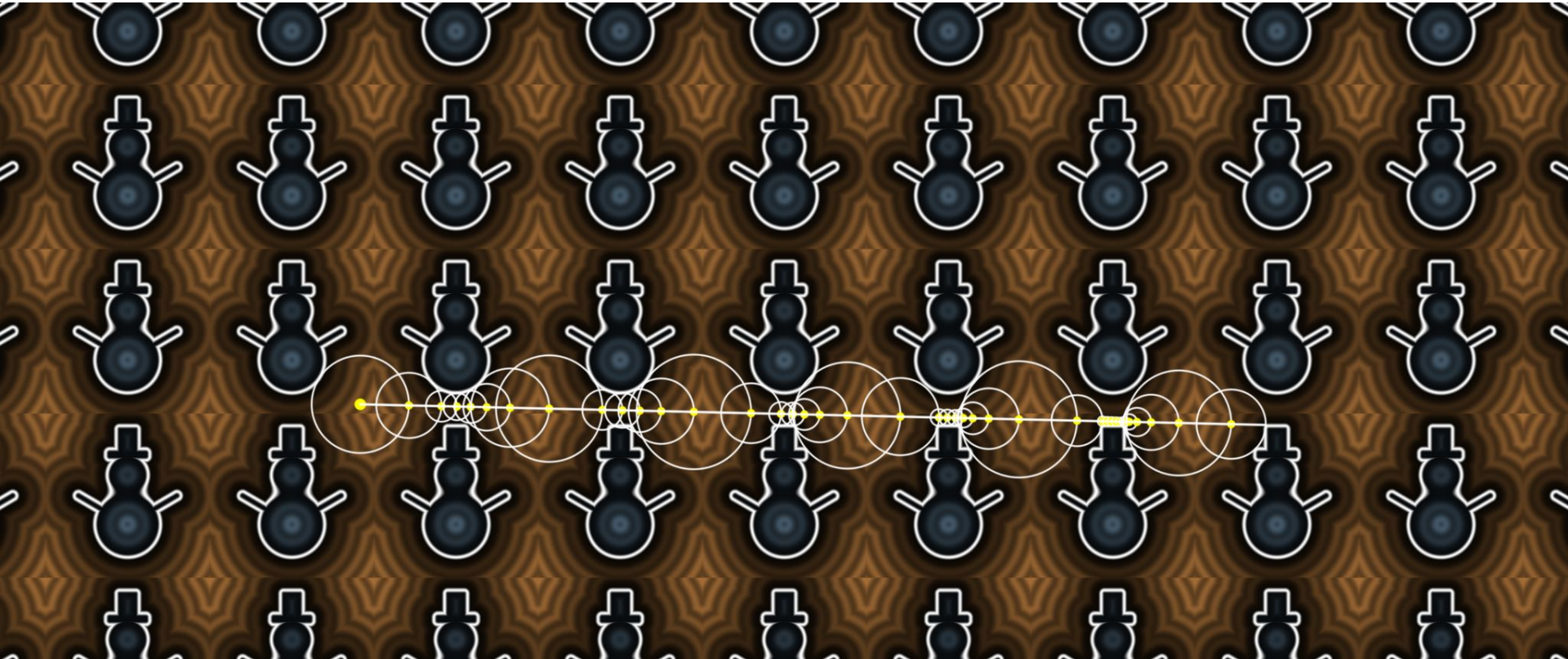
Worst case

137 FPS -> 128 FPS



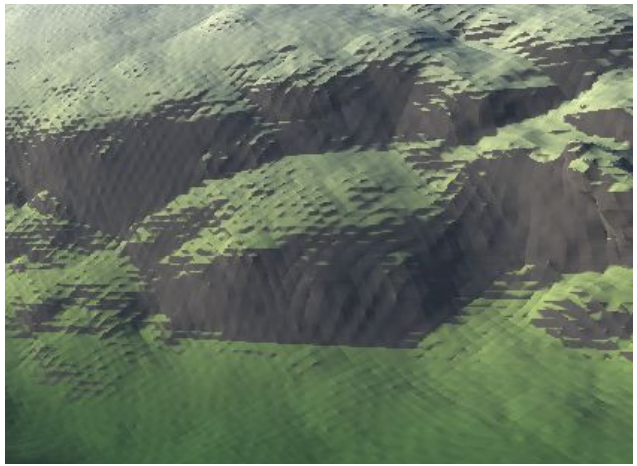
Worst case

137 FPS -> 128 FPS

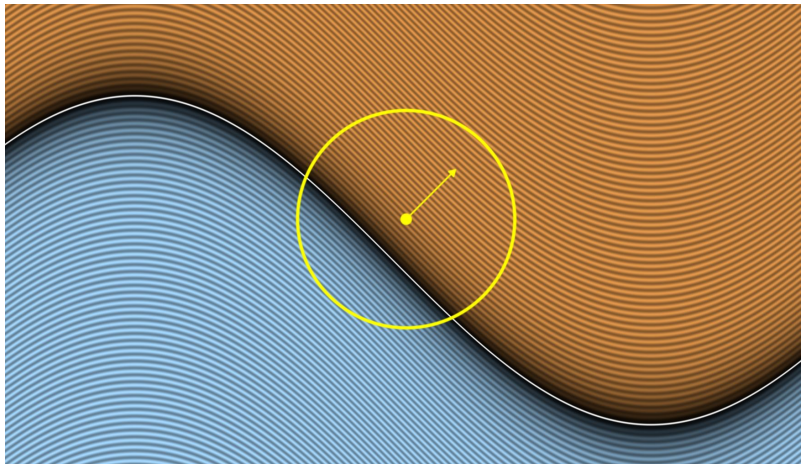


Limitations

- Abrupt normals (breaks the illusion)
- Poor results with imperfect SDFs



Normal faceting



Inexact SDFs

Conclusion

- x36 speedup
- High fidelity
- SDF Physics

Sunset function: *robobo1221*

Extra - GUI


File View Tools
Scene
Objects Materials
Add Delete
Plane
Box
Box
Box
Box
Sphere
Box
Box
Snowman
Box
Object
Blue
Material
Translate Rotate Scale
8.565 0.874 15.302
Position
0.005 0.945 0.013
Rotation
0.200
Scale
Physics active
1.000
Density
0.000
Roundness
Union
Combination operation
0.000
Smoothness
Box
Morphing primitive
0.000
Morphing
None
Deformation operation
None
Repetition operation
0
Terrain octaves
Log
Options Clear Copy All Stats Debug Info Warnings Errors Filter
-> 2.972 ms
| createAccelerationStructures
AS size: 384 bytes
| | createTopLevelAS
AS size: 59419520 bytes
-> 3.037 ms
| |-> 12.698 ms
| createDescriptorSetLayout -> 0.066 ms
| compileShaders -> 2044.419 ms
| createPipelines
| | createShaderBindingTable -> 0.026 ms
| |-> 0.925 ms
|-> 2158.124 ms
Adding DefaultWindowTitle
Adding Logger UI
Running application
Loading from strand.json
Profiler
Table BarChart LineChart PieChart
CPU total 1
Levels
Graphics
Buffer updates
Simulation
Generation
Tracing
Lighting
Post process
0.03
0.82
Settings
75 FPS / 13.261ms
Viewport Size: 1900 x 743
Camera
#1 #2 #3
Speed 3.00
Projection
Type Perspective Orthographic
FOV 60.0°
Z-Clip 0.001000 100000.000000
Position
Other
Tonemapper
Tracing
Compute tracing
Tracing Mode
Simulation
Stop Resume
0.000
Time dialtion
3
Sub steps
0.000 -9.800 0.000
Gravity
User action
Lighting data
Debug colors
Atlas%
Active
Debug color
Mode
Magma
Palette
Camera id0: 299,79,223
Reset TLas

Extra - Graphical errors