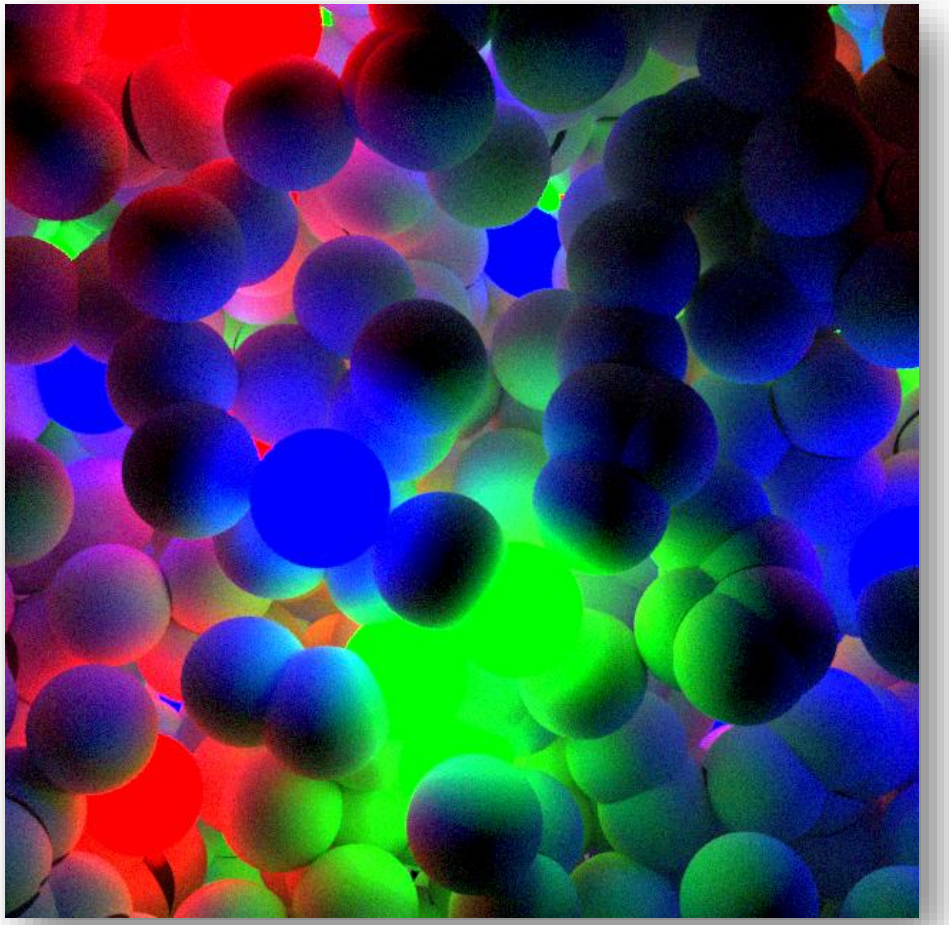
Acceleration Structures, Textures

Computer Graphics
Fall Semester 2025 FHNW

S. Felix

University of Applied Sciences and Arts Northwestern Switzerland
School of Computer Science

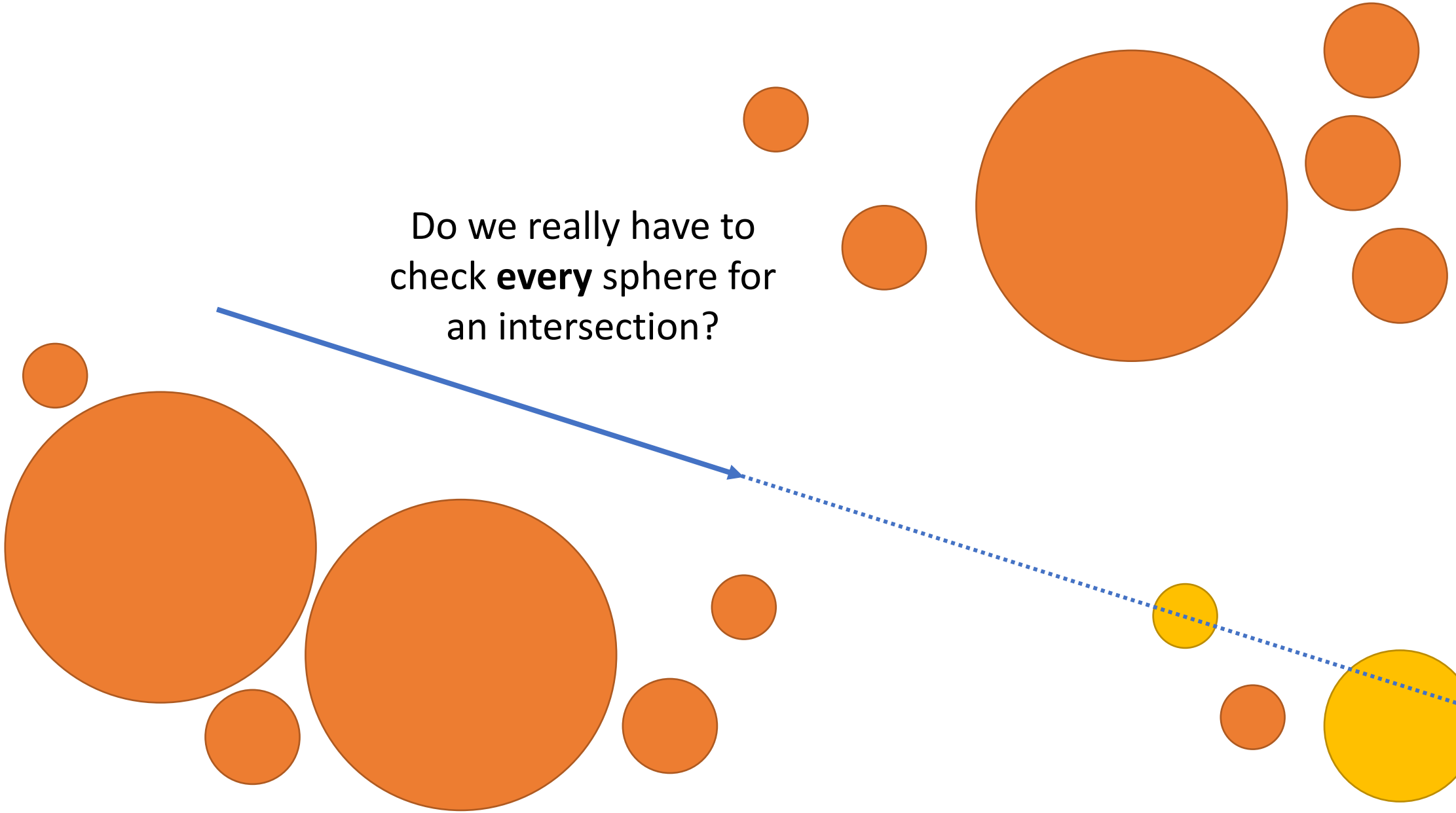


This **640x640 pixel** image of **1200 spheres** was rendered with 1024 samples per pixel.

419’430’400 eye rays

1‘138‘293‘374‘400 intersection tests

Do we really have to check **every** sphere for an intersection?



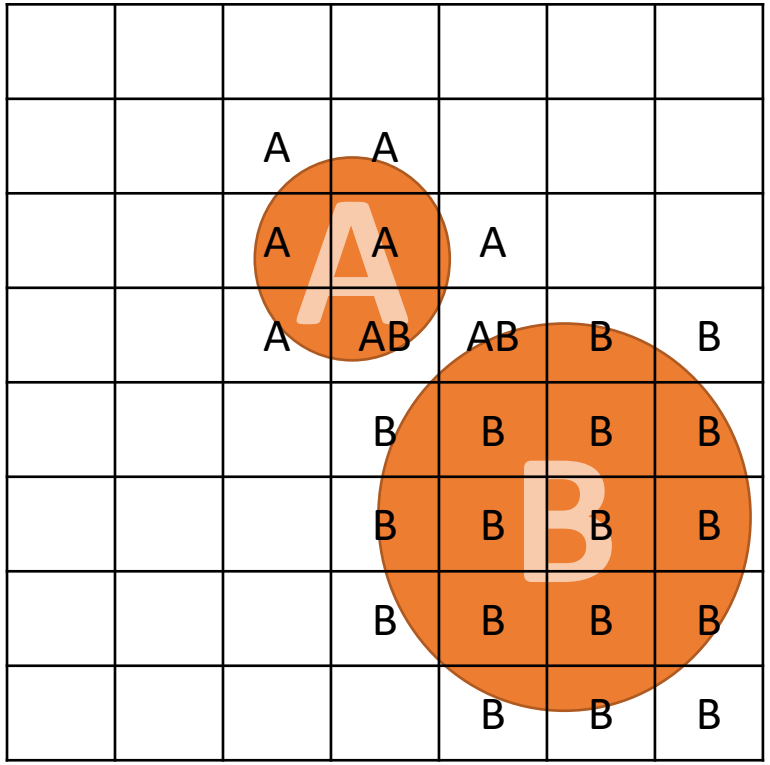
Acceleration Structures

Grids

Trees

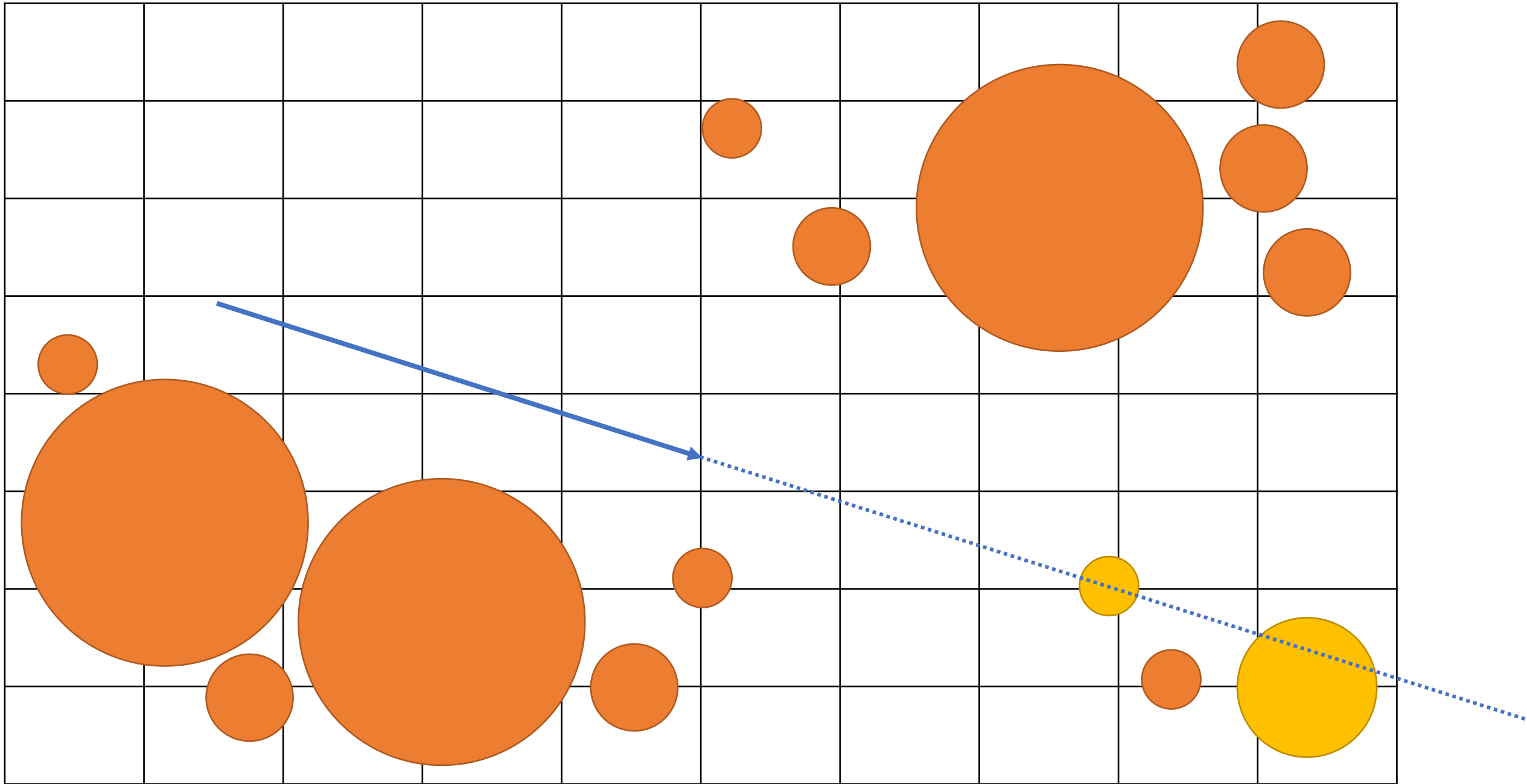
Bounding Volume Hierarchies (BVH)

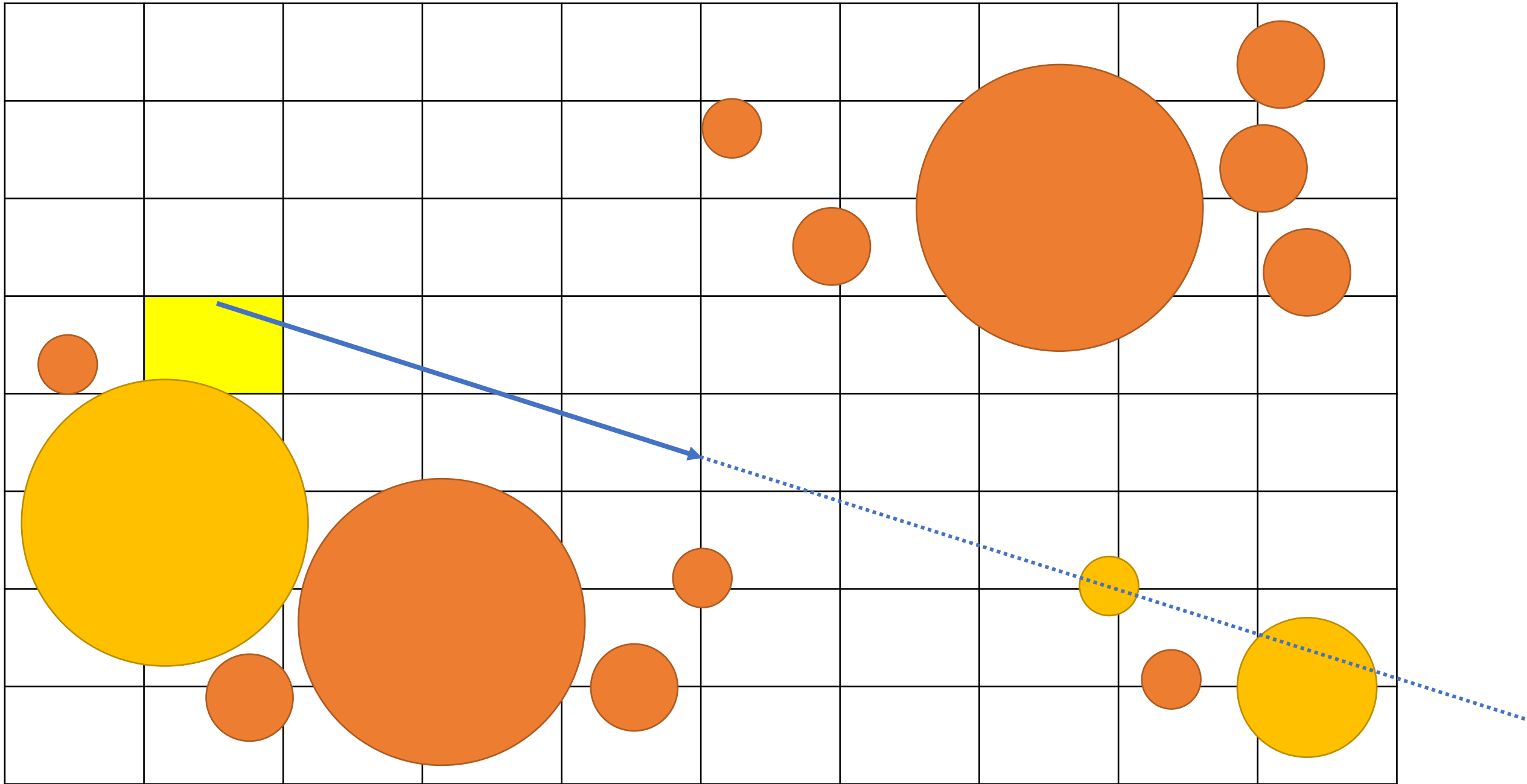
Grid

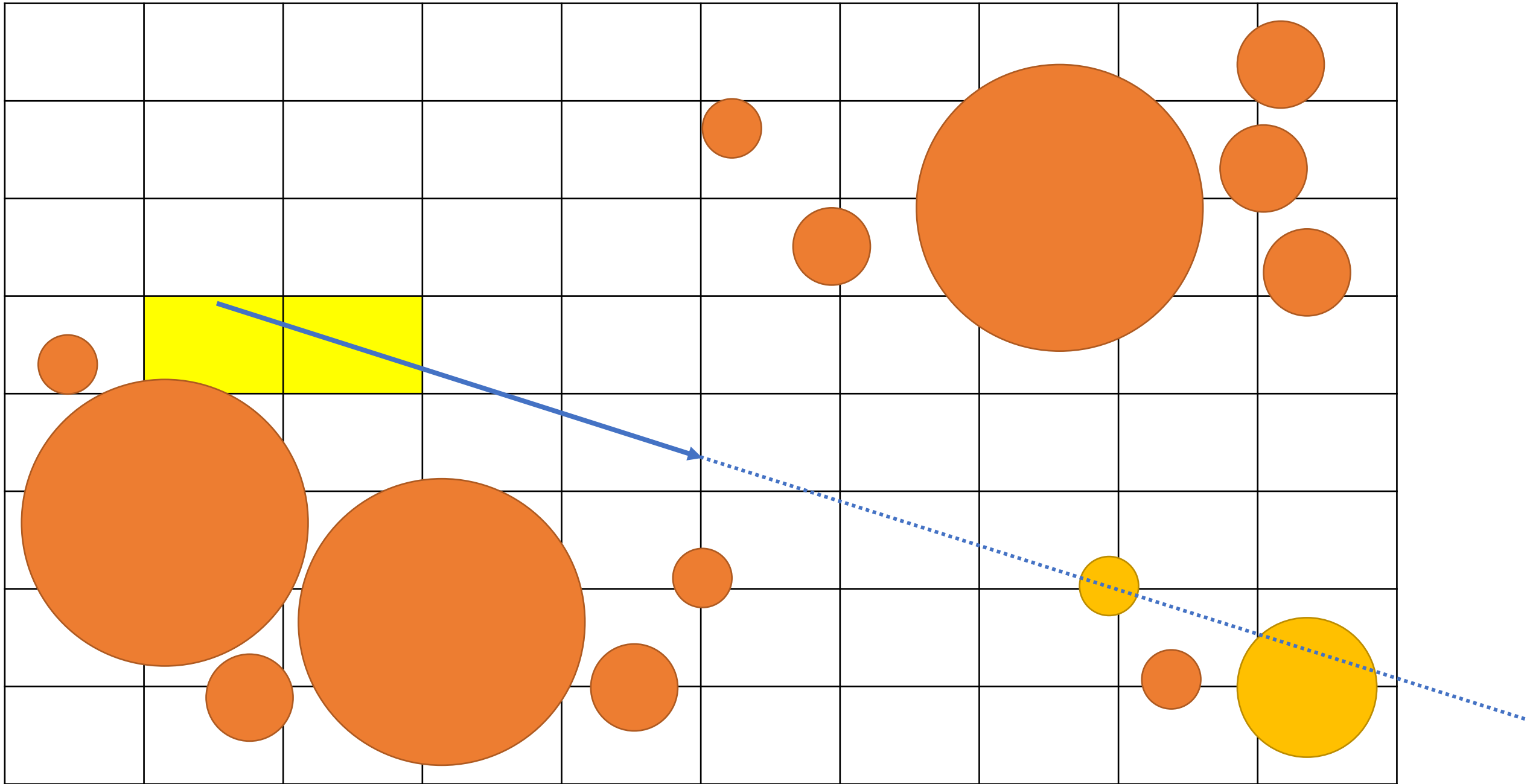


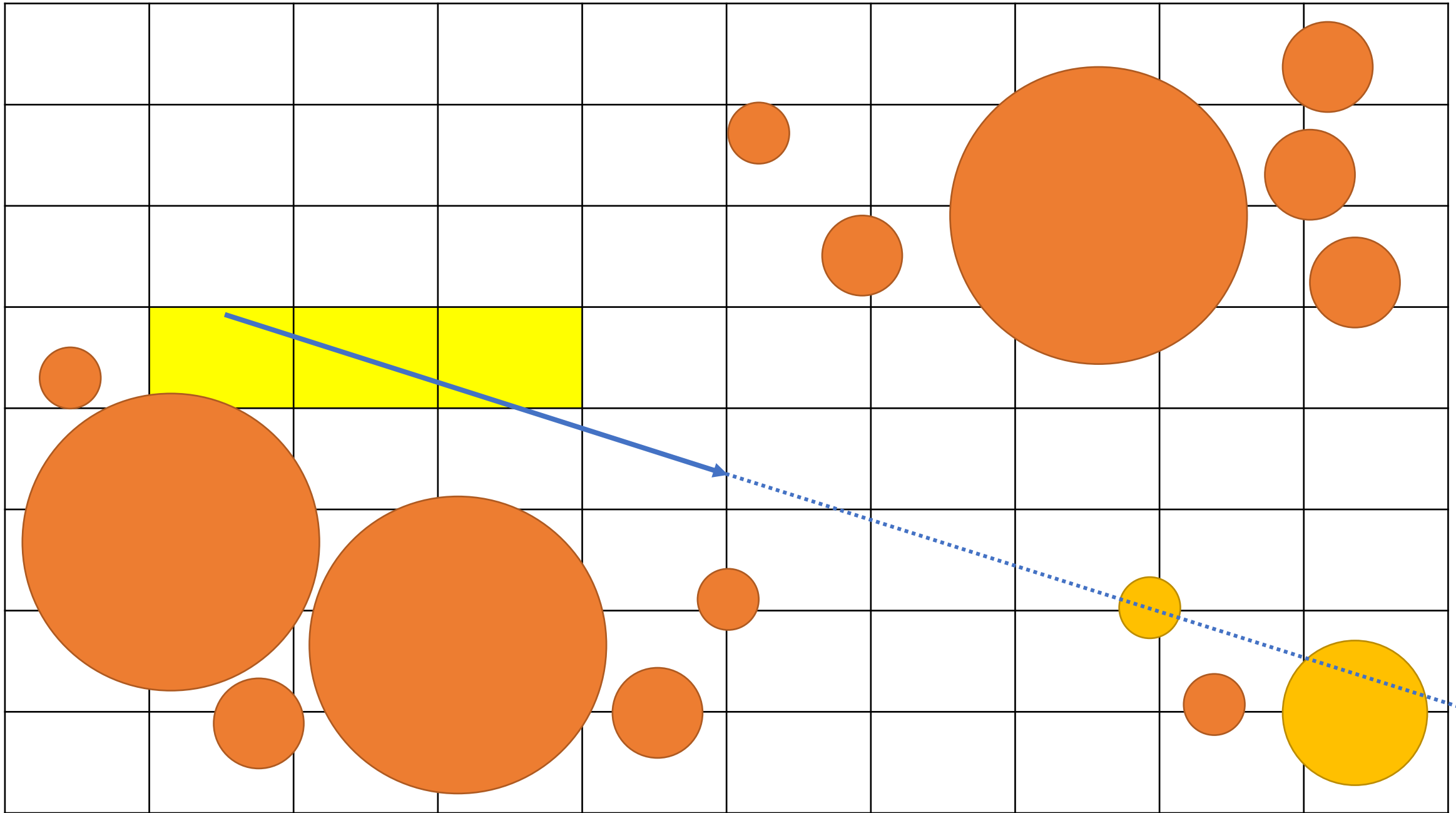
1. Define a grid, covering the whole scene.

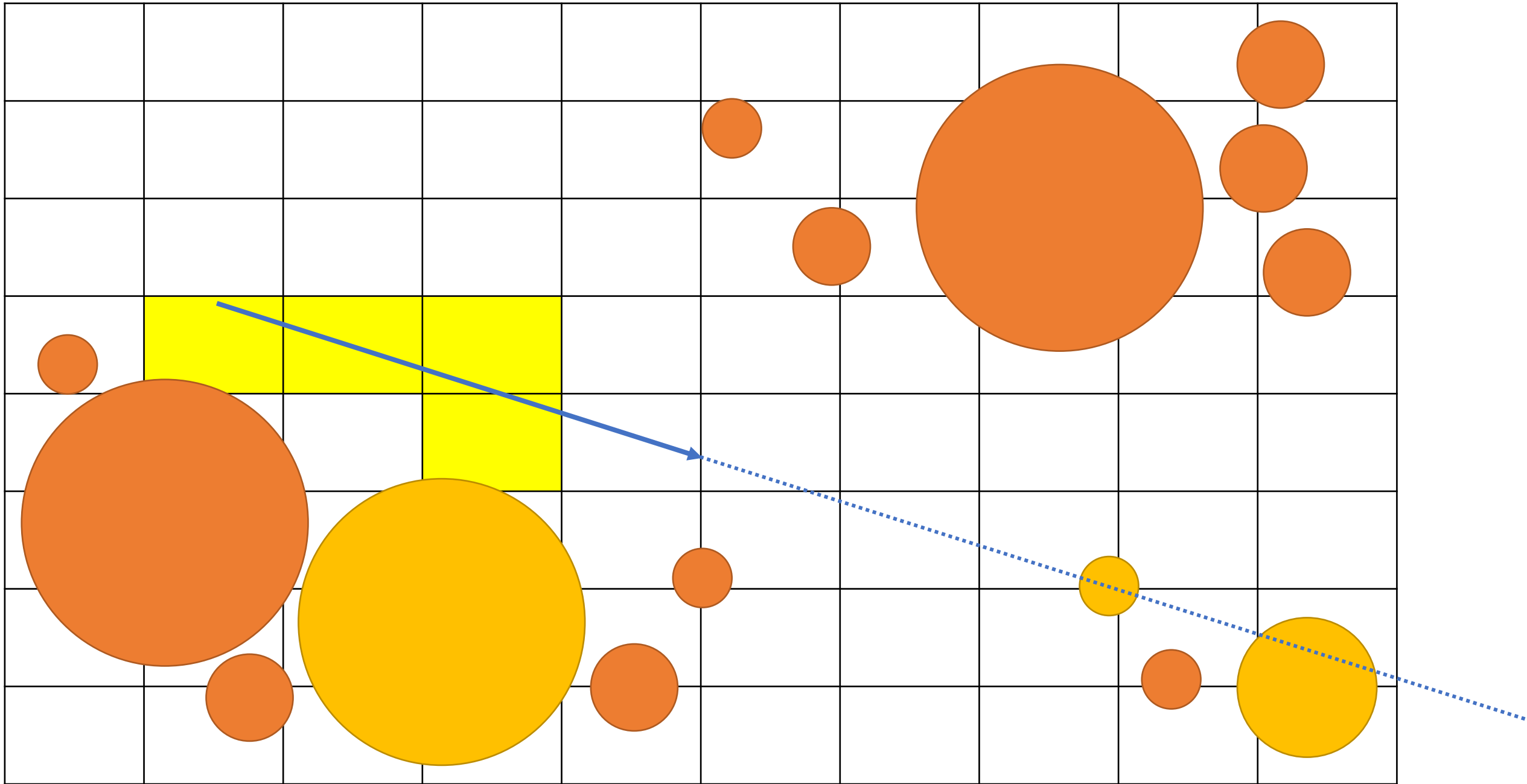
2. Keep a **list of objects** per cell.

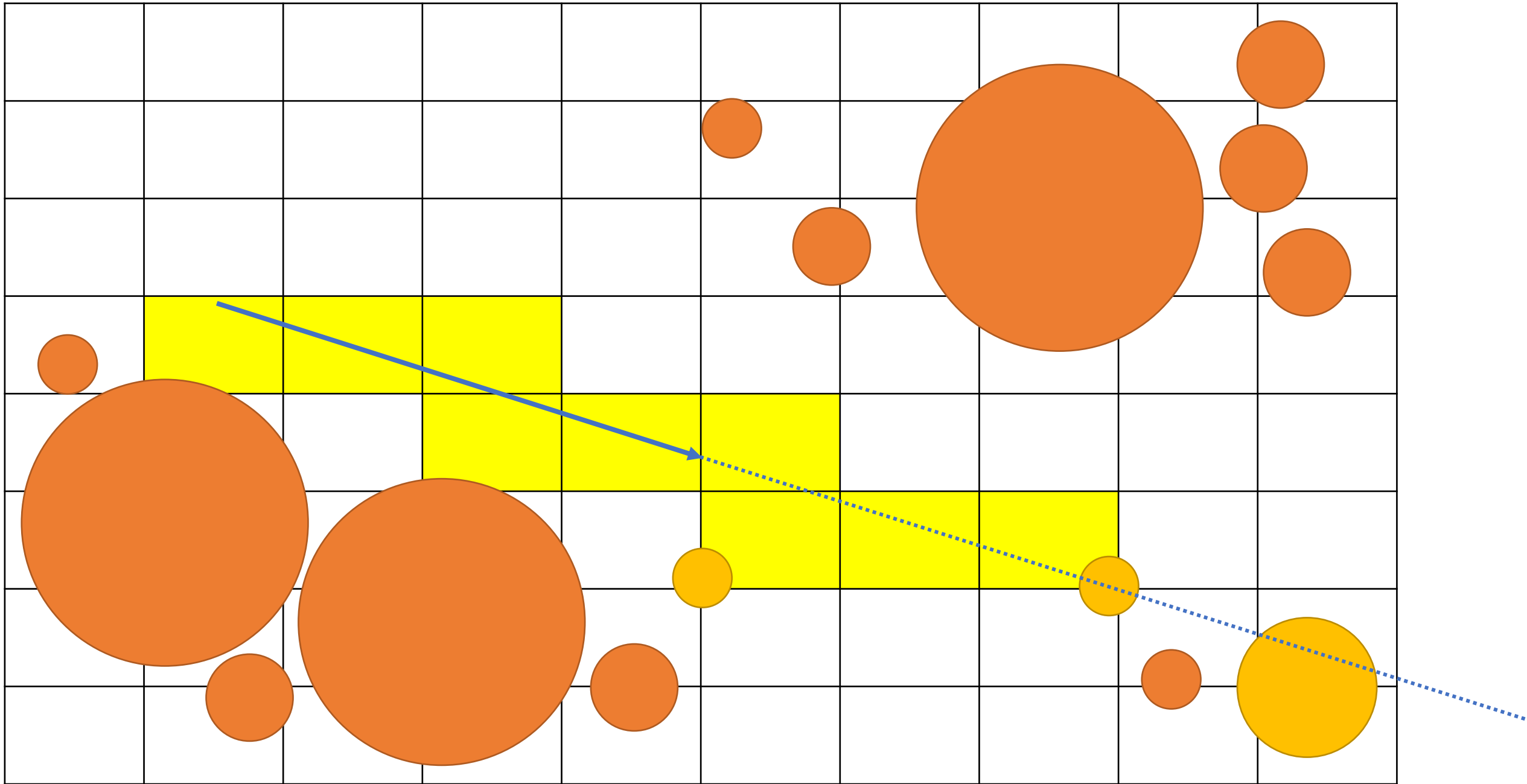












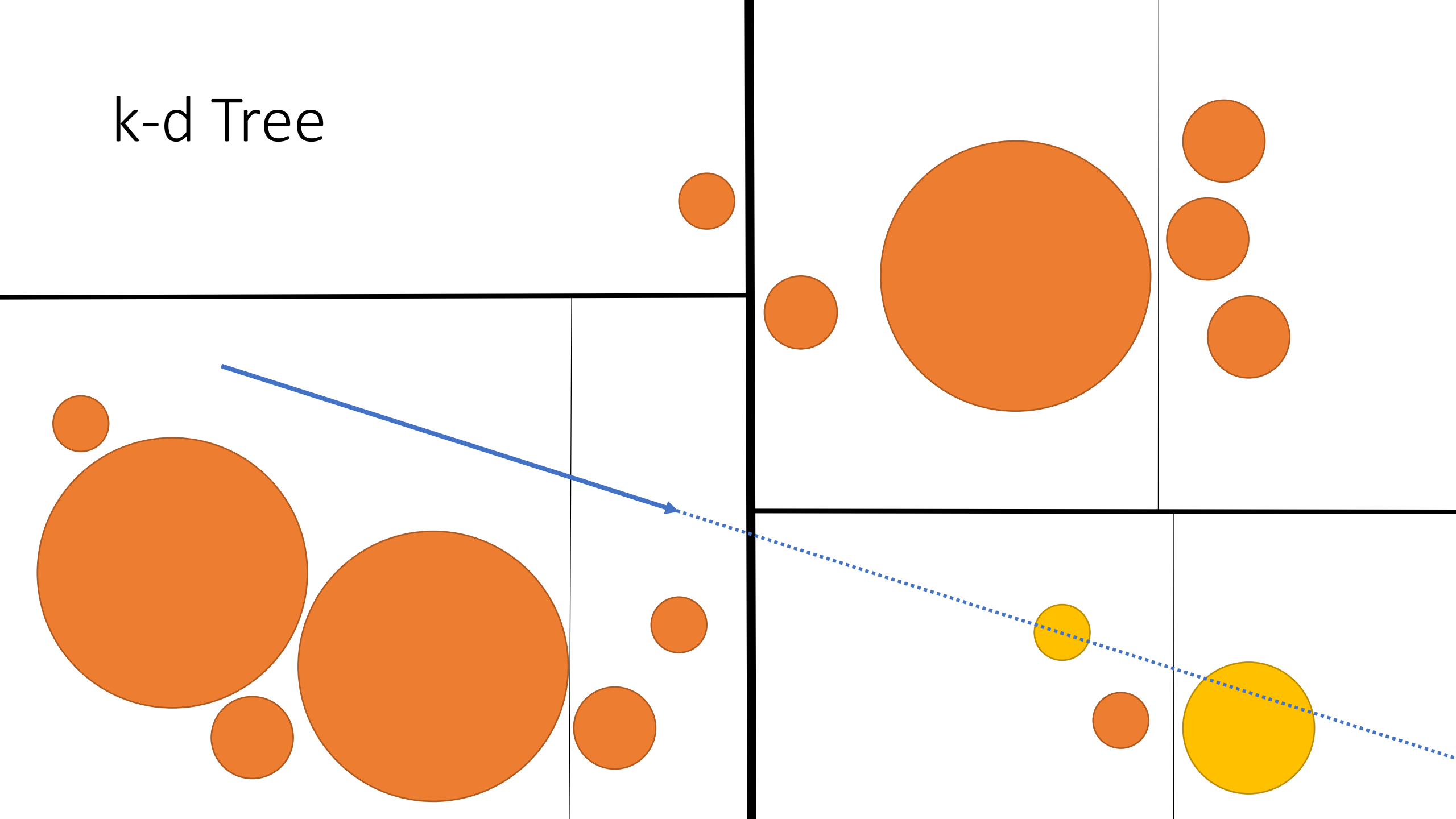
Acceleration Structures

Grids

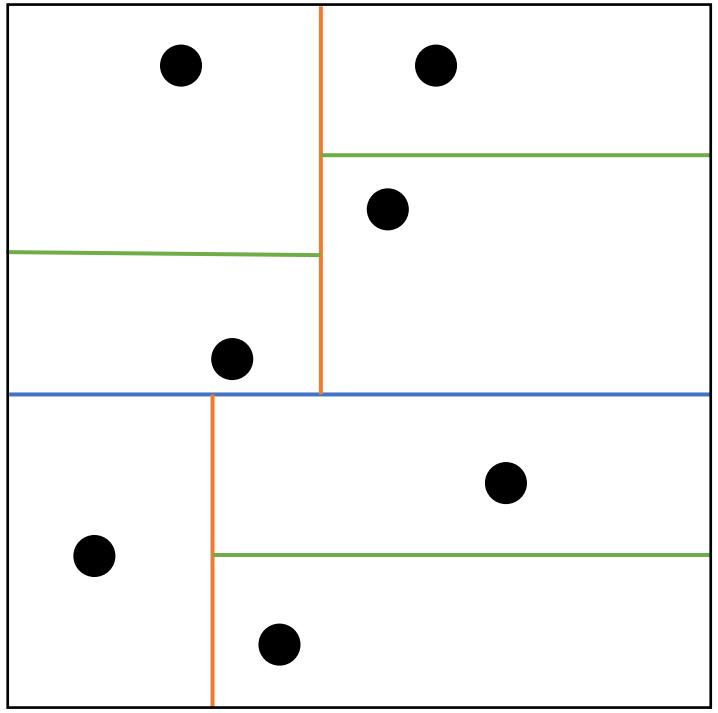
Trees

Bounding Volume Hierarchies (BVH)

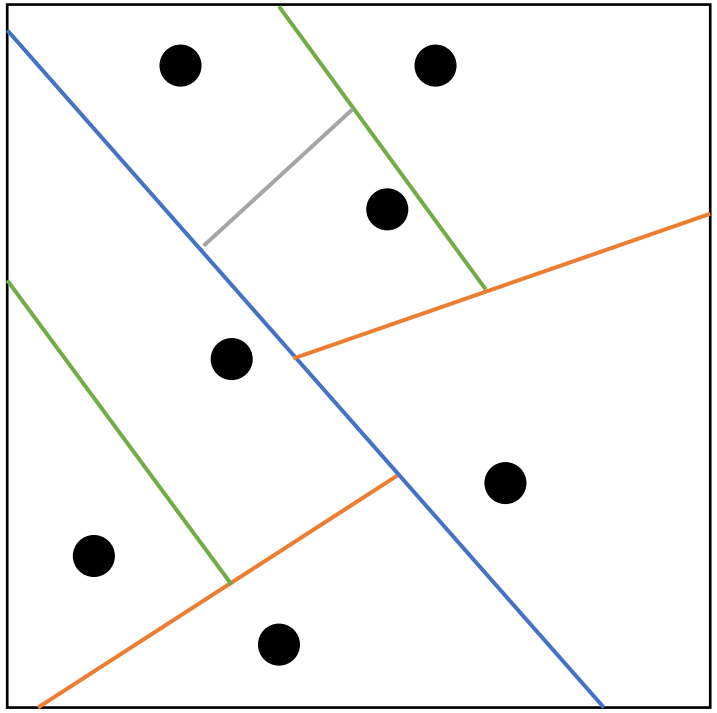
k-d Tree



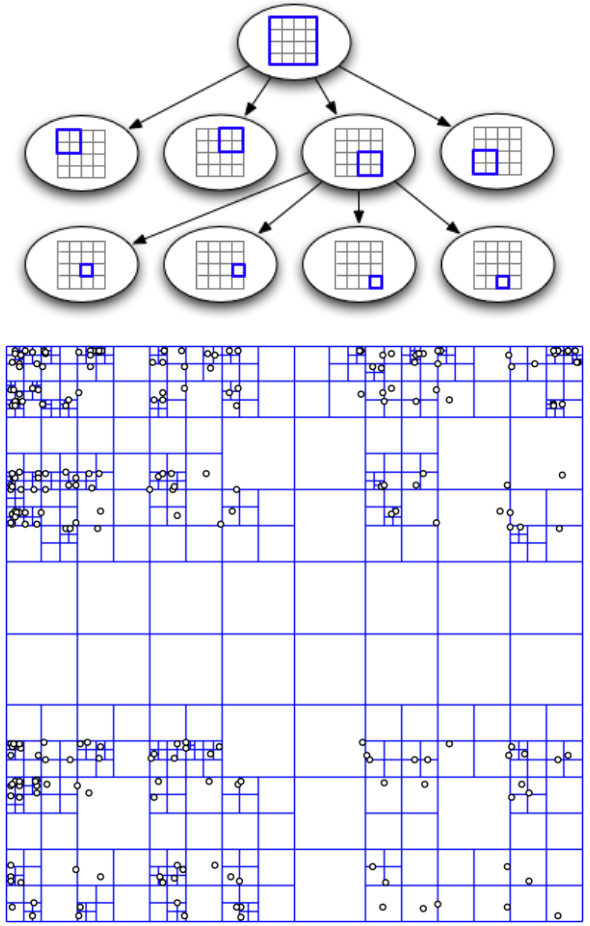
Tree Design Space



k-d tree



Binary Space Partitioning (BSP)



Quadtree/Octree

Images: Wikipedia & Nick Johnson

Acceleration Structures

Grids

Trees

Bounding Volume Hierarchies (BVH)

Bounding Volumes

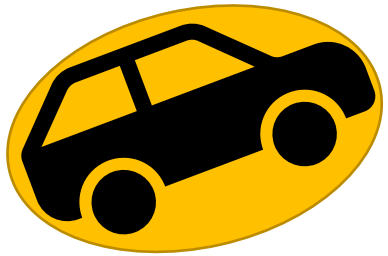
Bounding volumes **fully enclose** objects.

Bounding volumes **over-approximate** the true volume

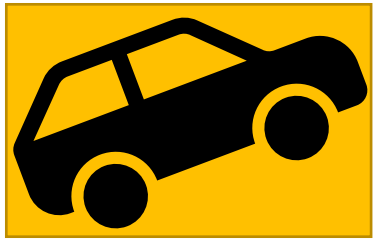
Design space

- **Axis-aligned** vs. **oriented**
- **Boxes** vs. **Ellipsoids** vs. **Spheres** vs. **many others**

Bounding sphere

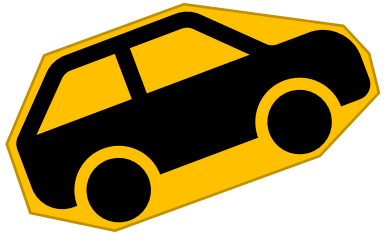


Bounding ellipsoid



Axis-aligned bounding box (AABB)

Oriented bounding box (OBB)



Convex Hull

Bounding Spheres

Simple Approximation

$C = \frac{1}{n}\sum_i p_i$ *(Center of mass)*

$r = \max_i \|p_i - C\|$

Ritter Approximation

1. Take 2 extremal points, p_a and p_b
2. $C = \frac{p_a + p_b}{2}$

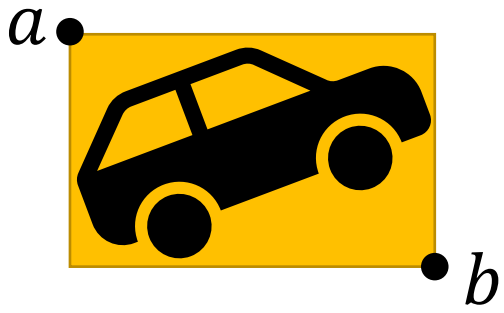
$$r = \frac{\|p_a - p_b\|}{2}$$

3. If any point is outside sphere, determine a new bounding sphere, covering both $Sphere(C, r)$ and p_i

Exact Algorithms

- LP/Simplex
- Randomized algorithm (Welzl)
- …

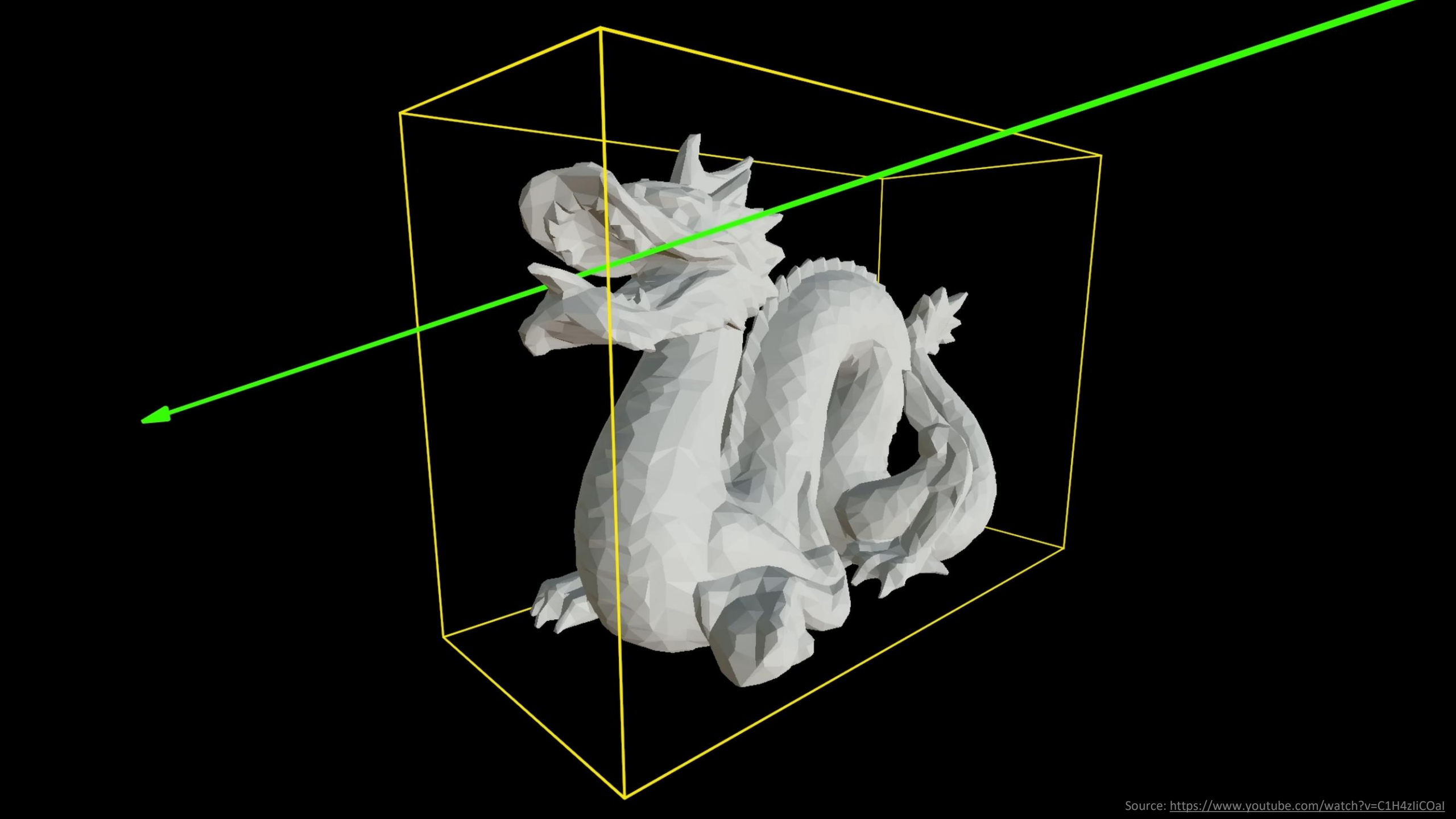
Axis-Aligned Bounding Boxes (AABB)



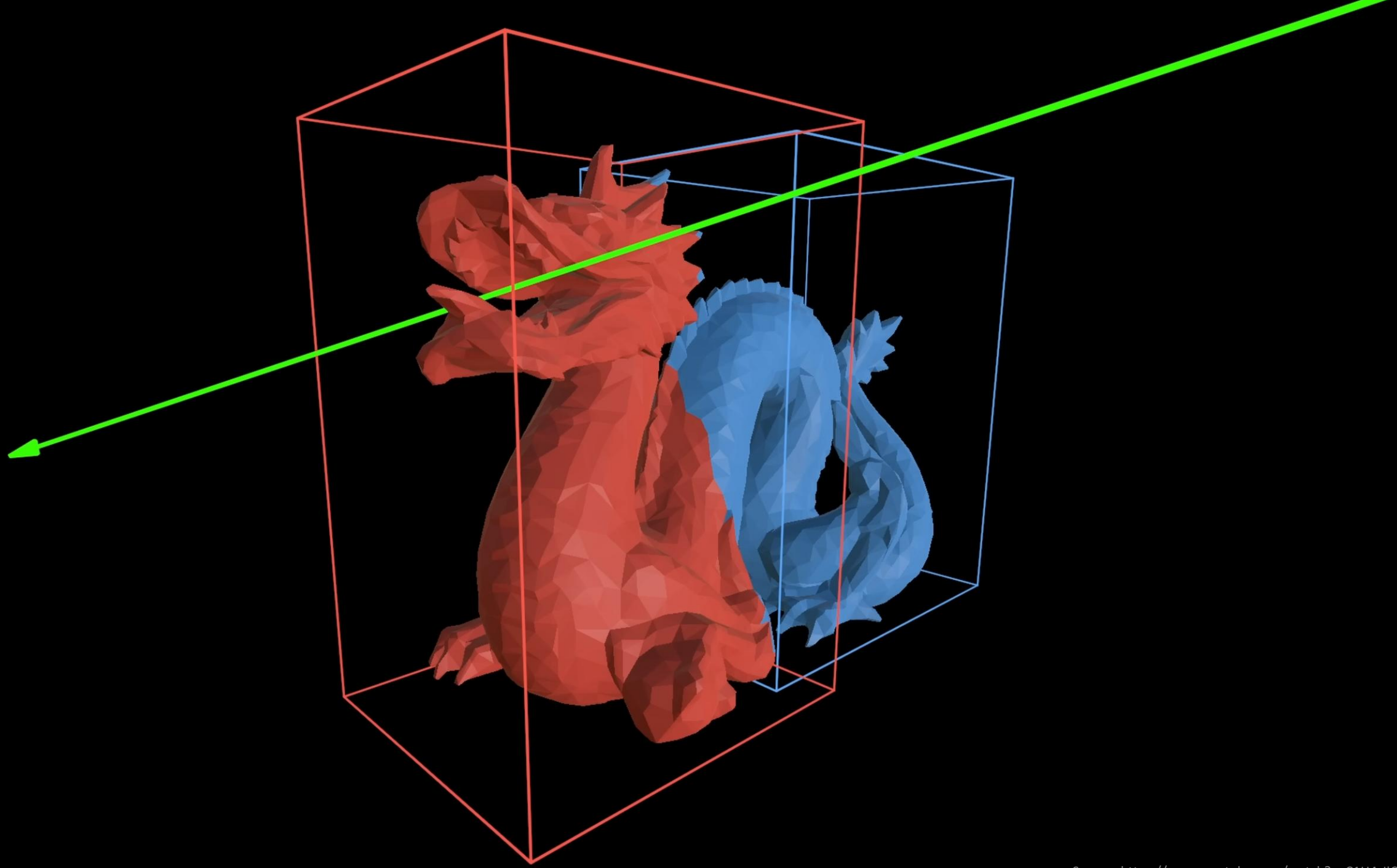
Axis-aligned boxes are uniquely defined by **two corner points**.

$$a_x = \min_i p_{i,x}$$

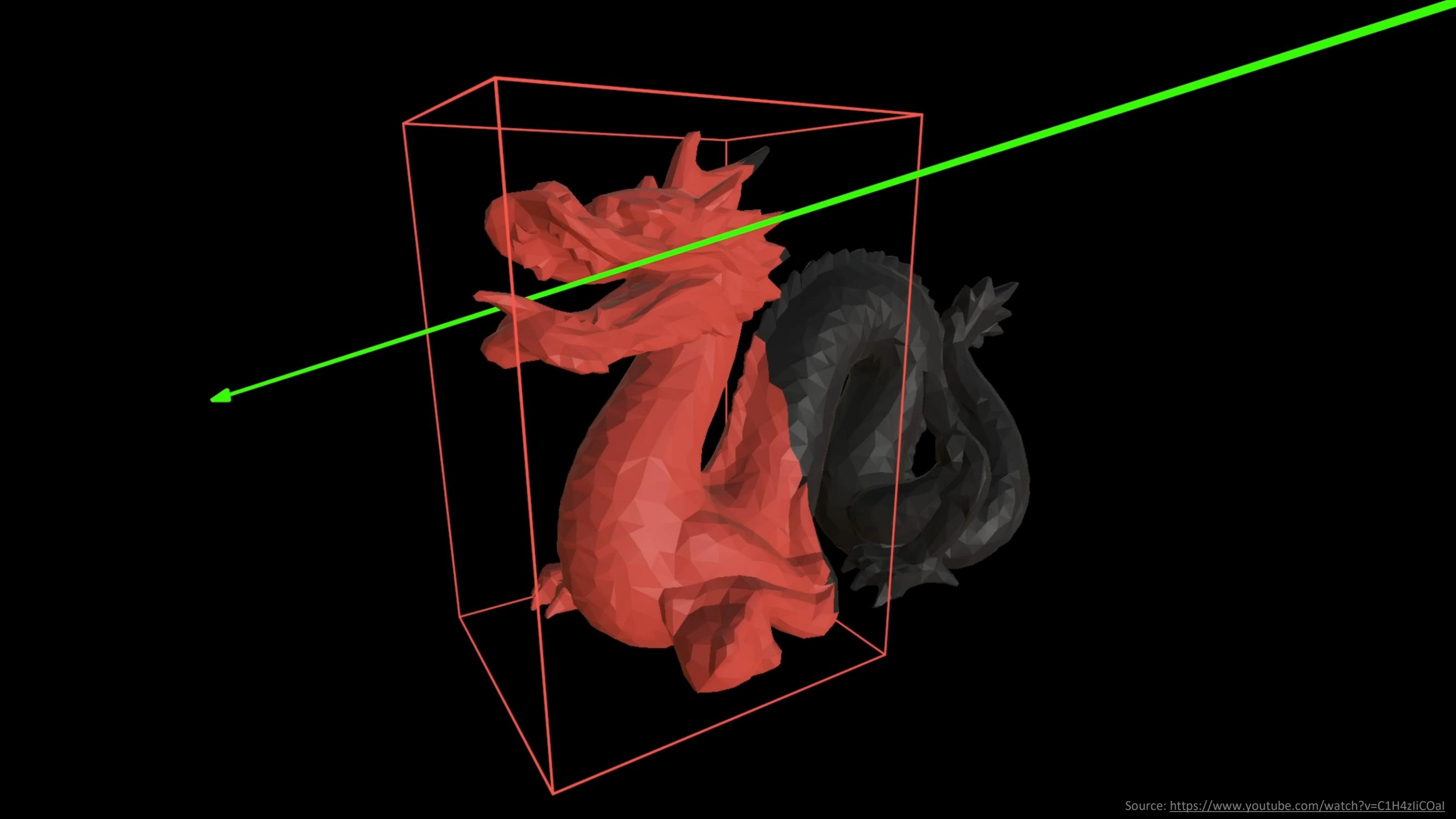
$$b_x = \max_i p_{i,x}$$



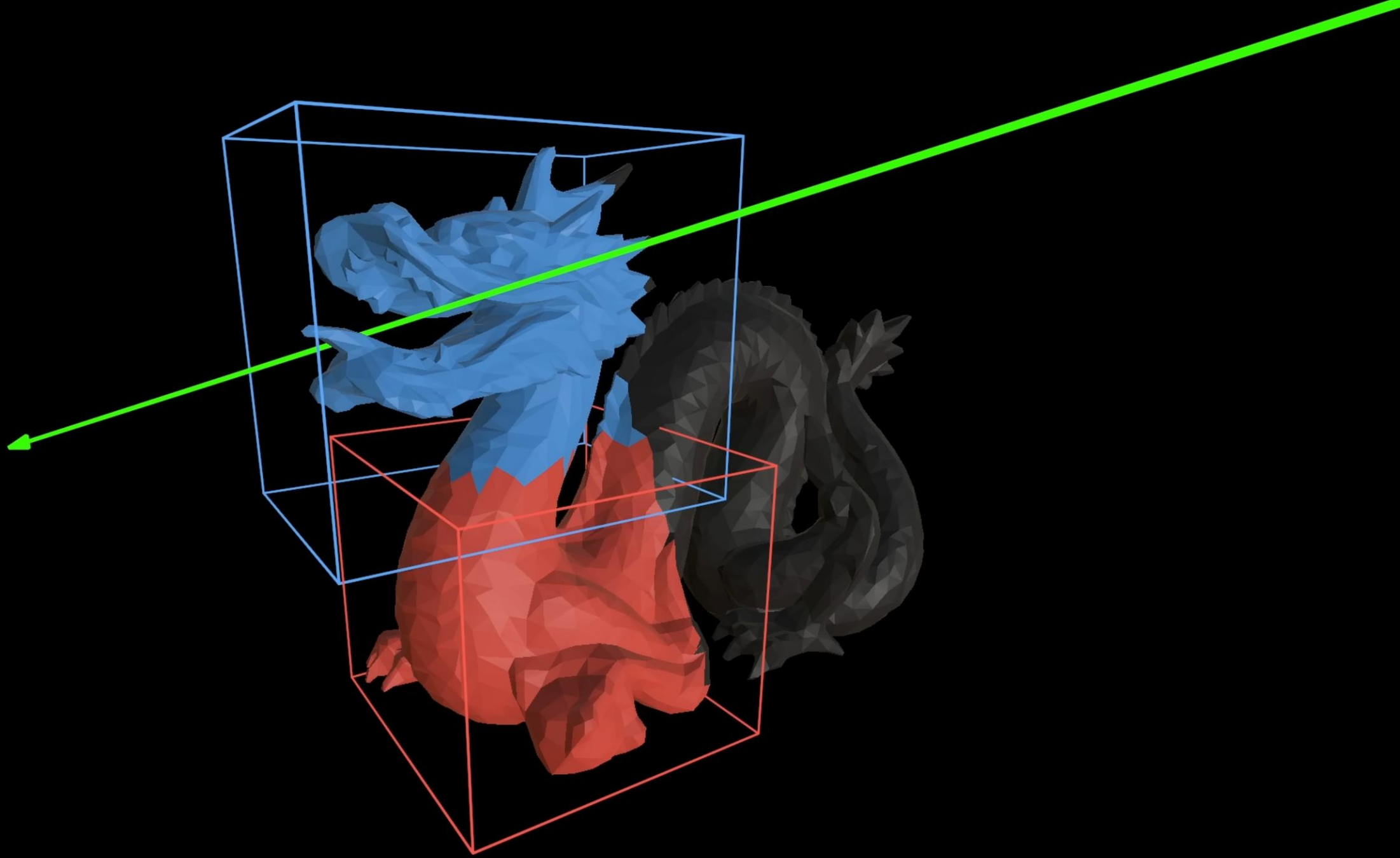
Source: https://www.youtube.com/watch?v=C1H4zIiCOaI



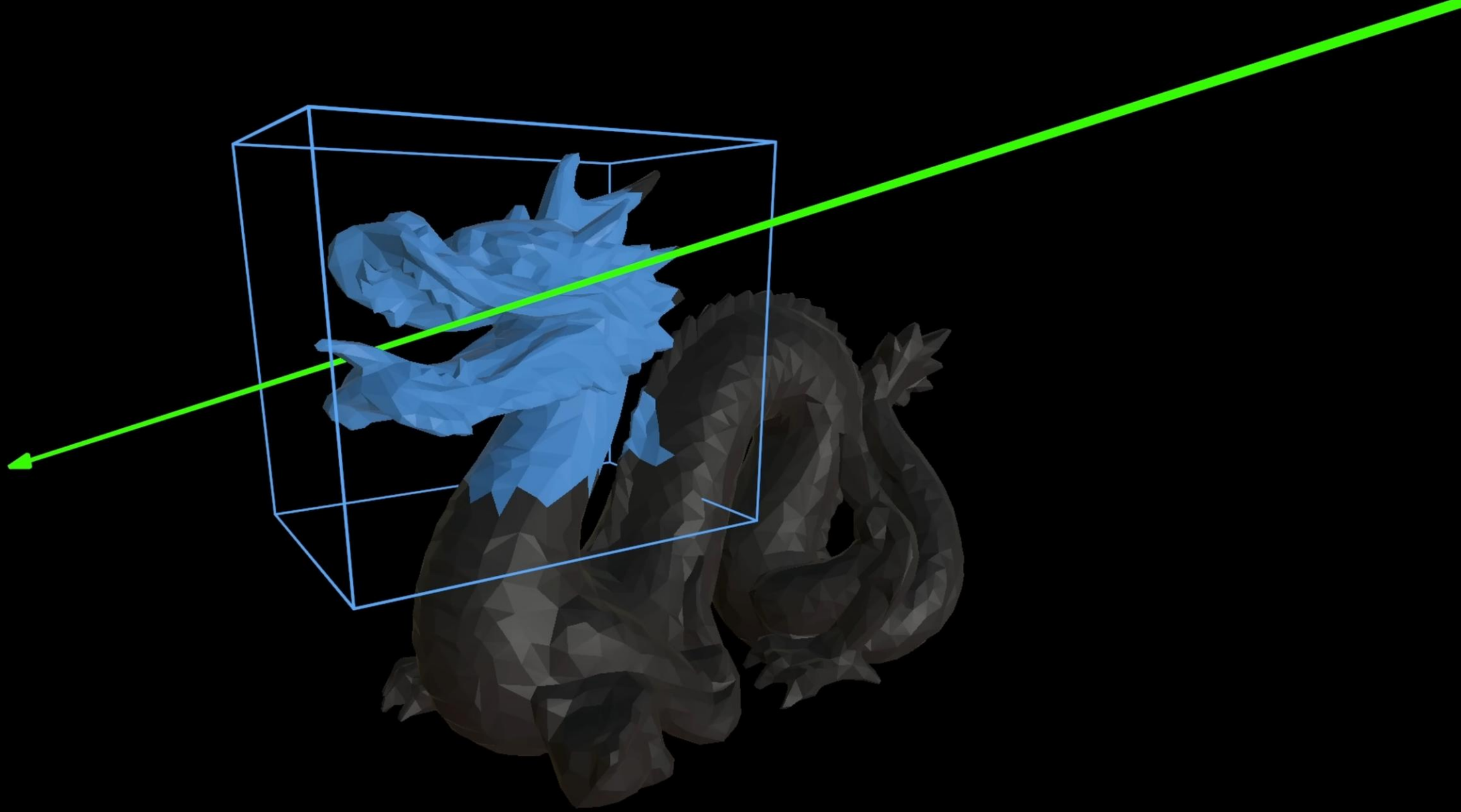
Source: https://www.youtube.com/watch?v=C1H4zIiCOaI



Source: https://www.youtube.com/watch?v=C1H4zIiCOaI

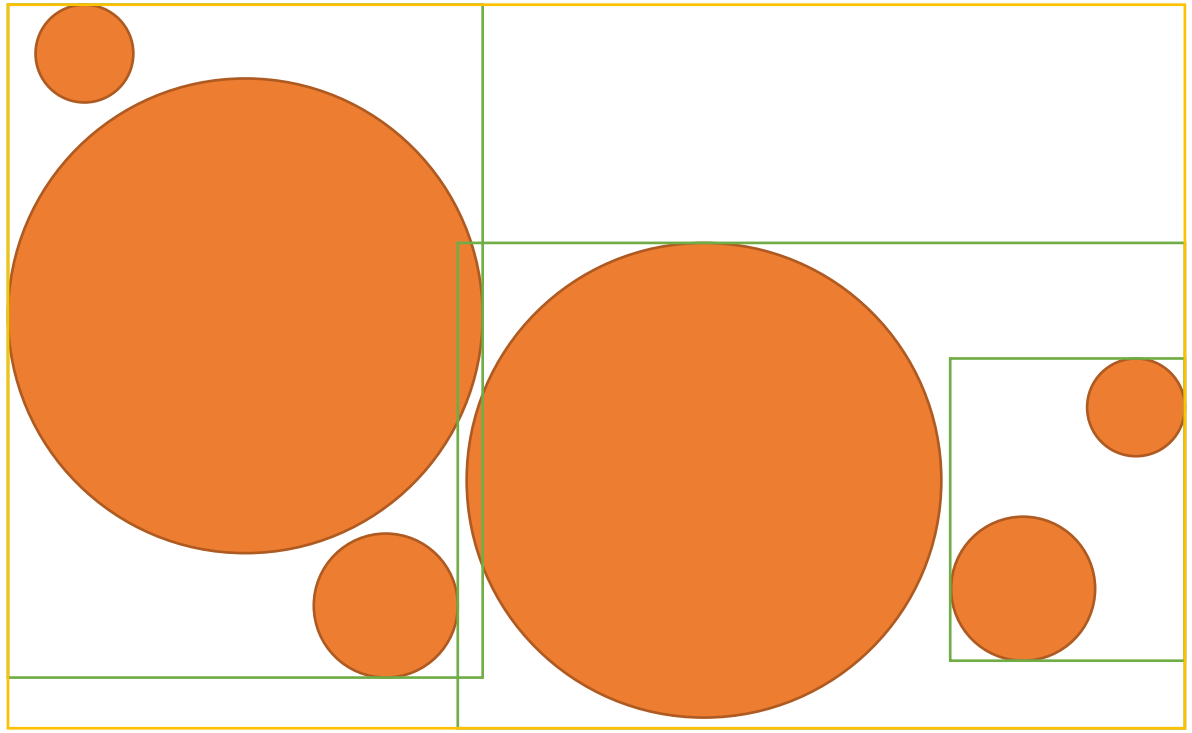


Source: https://www.youtube.com/watch?v=C1H4zIiCOaI



Source: https://www.youtube.com/watch?v=C1H4zIiCOaI

Bounding Volume Hierarchies (BVH)

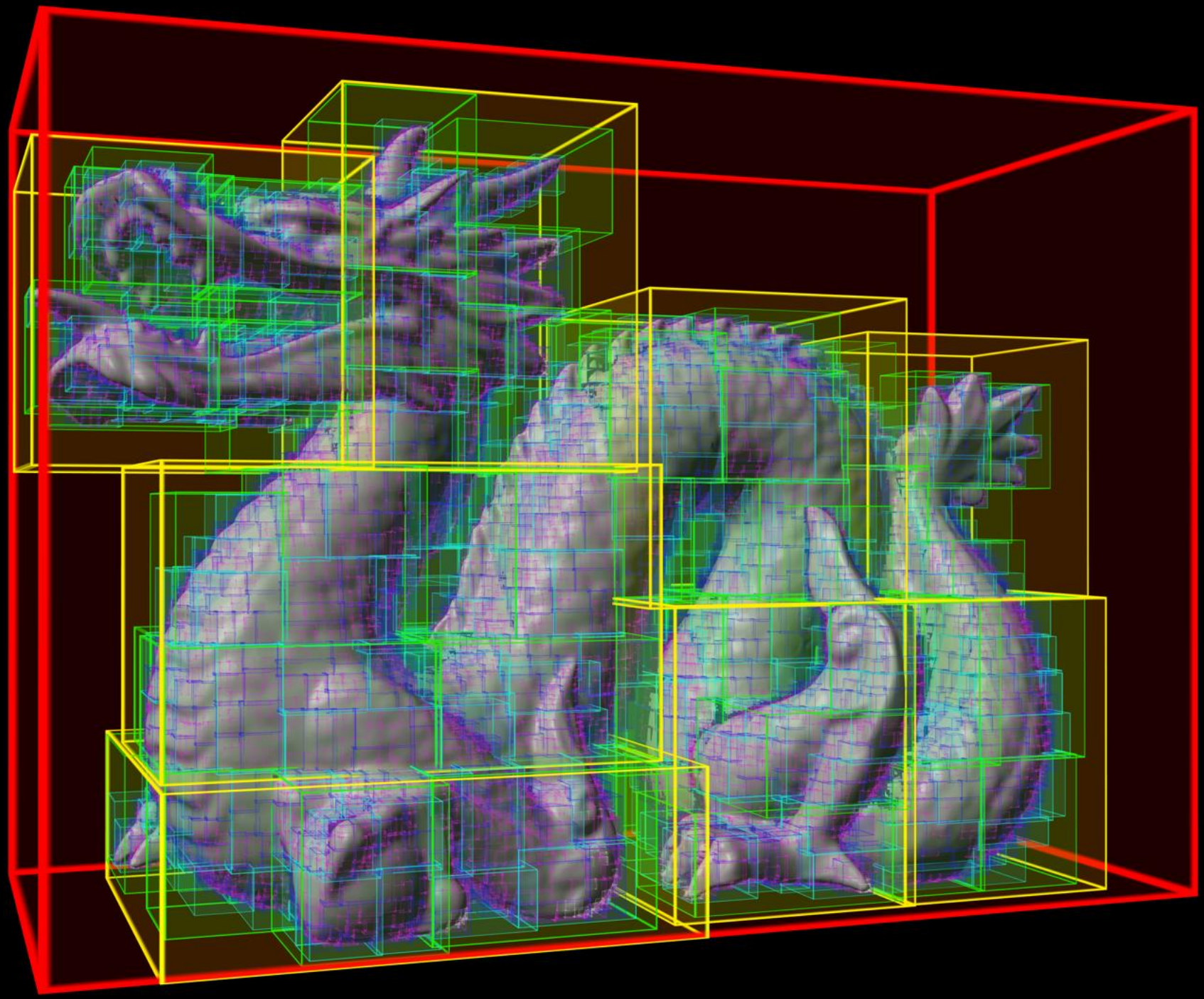


BVHs can be built **top-down** or **bottom-up**.

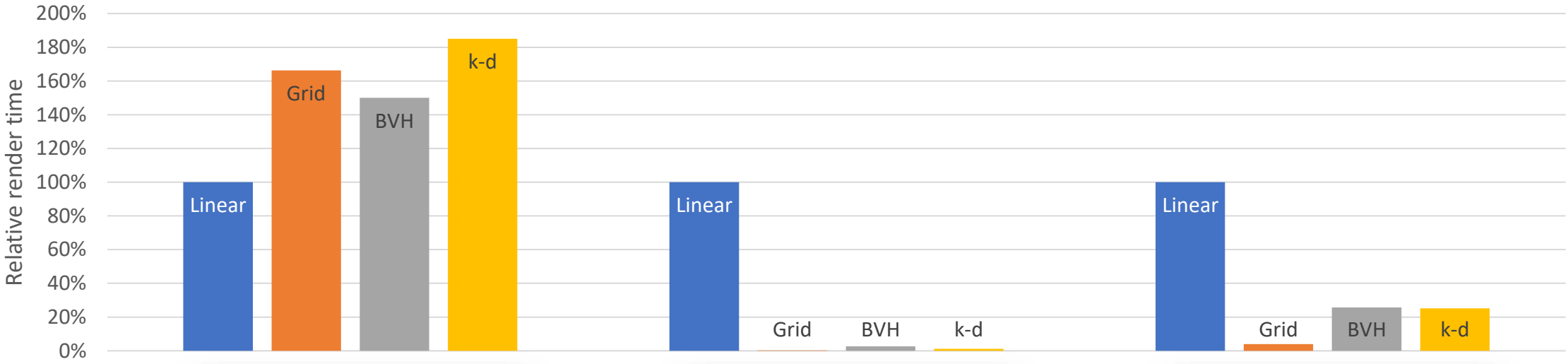
BVHs can be **wide** or **narrow**.

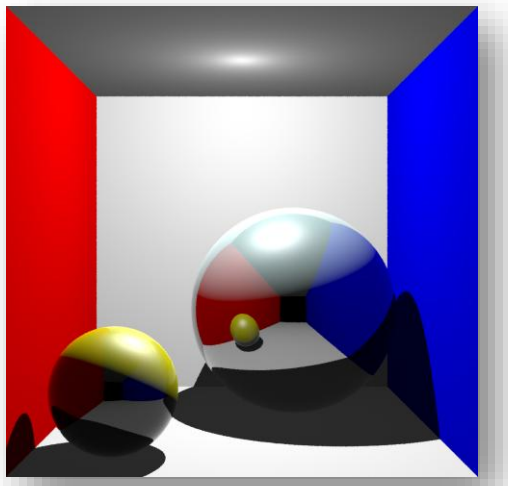
BVHs should be **balanced**.

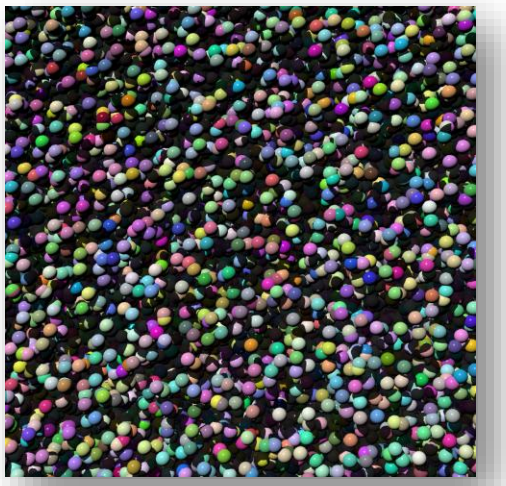
Volumes should be **minimal**.

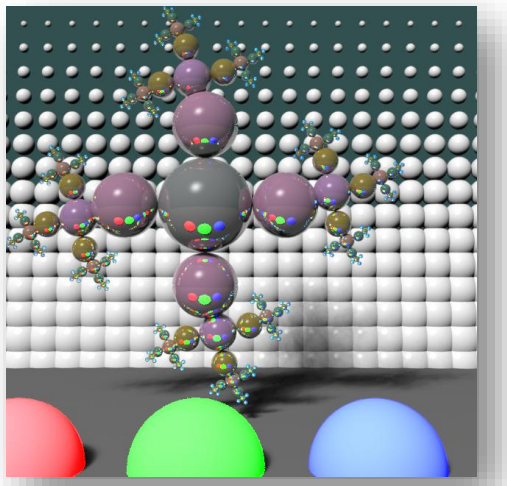


Comparison


Relative render time
200%
180%
160%
140%
120%
100%
80%
60%
40%
20%
0%
Linear
Grid
BVH
k-d
Linear
Grid
BVH
k-d
Linear
Grid
BVH
k-d

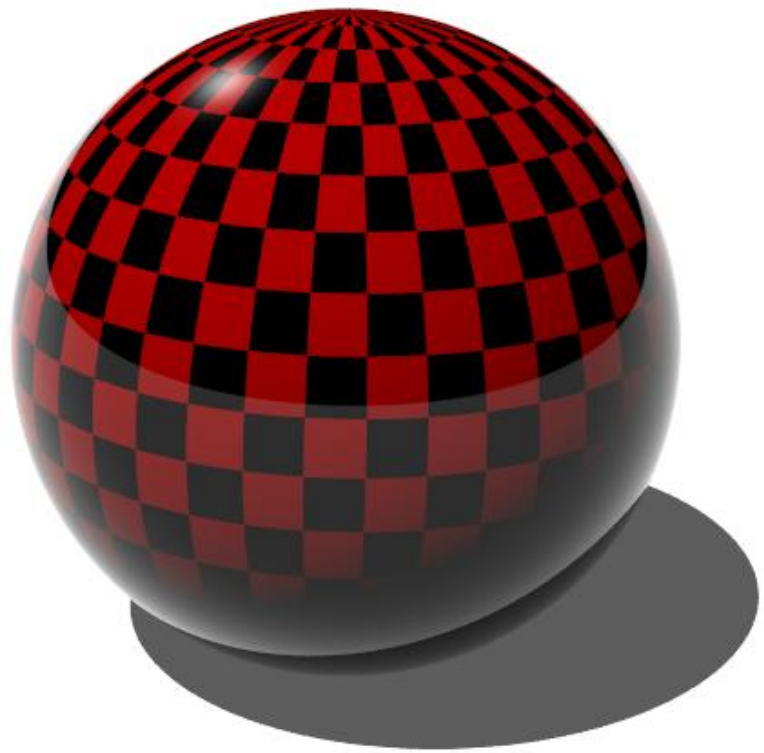






With suitable acceleration structures, intersections with n objects can be determined in $O(\log n)$.

Thus, path tracing can **scale** to enormous scenes.



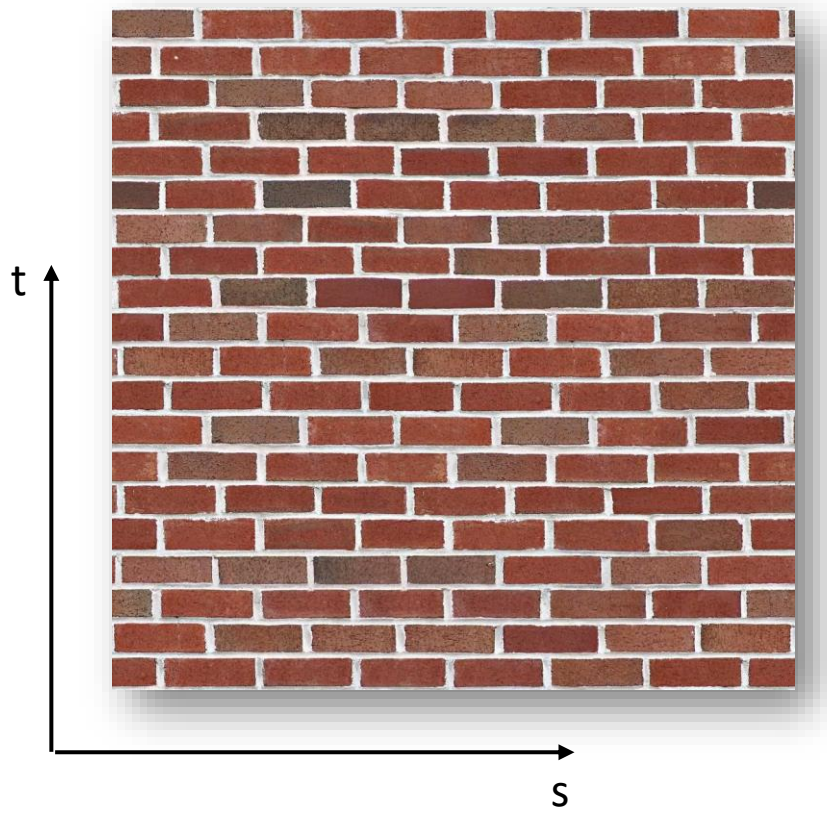
Texture Mapping

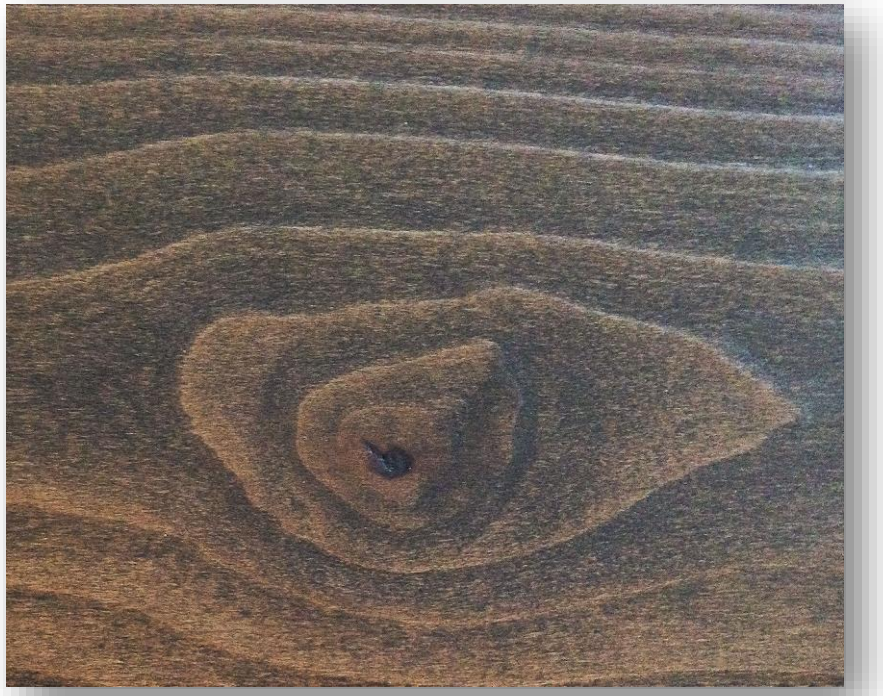
Bitmaps

Procedural

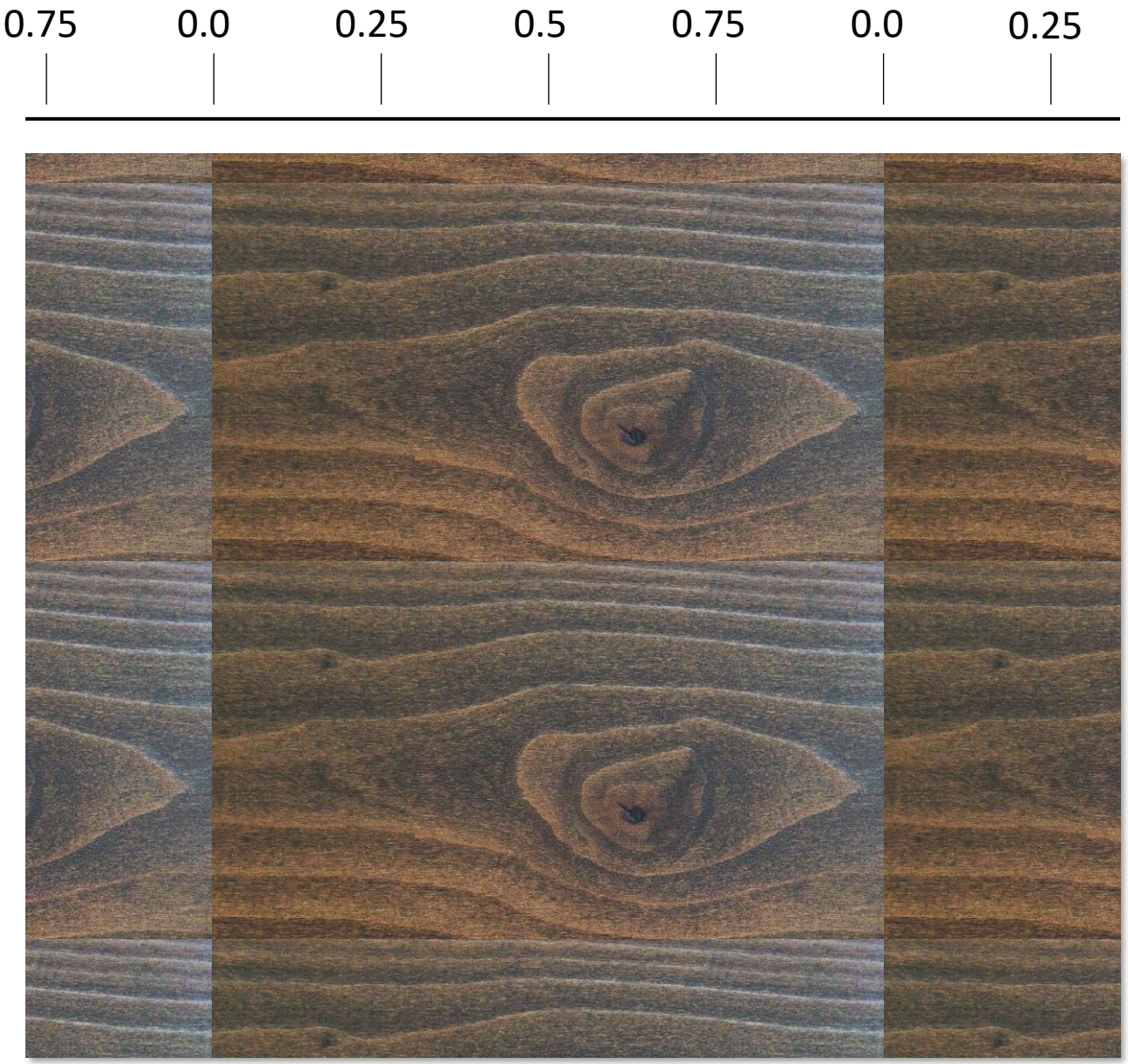
Bitmaps

We can look up material colors from a **bitmap**.





Bitmaps



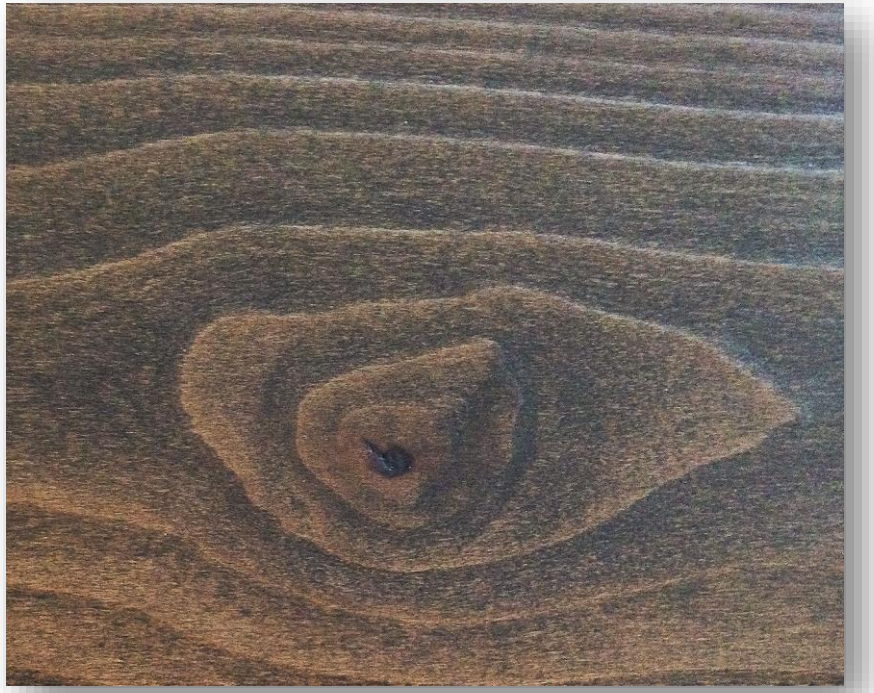
Bitmaps have a **limited resolution**.

Bitmaps are **finite**.

Bitmaps are **two-dimensional***.

*) usually

Tiling Bitmaps

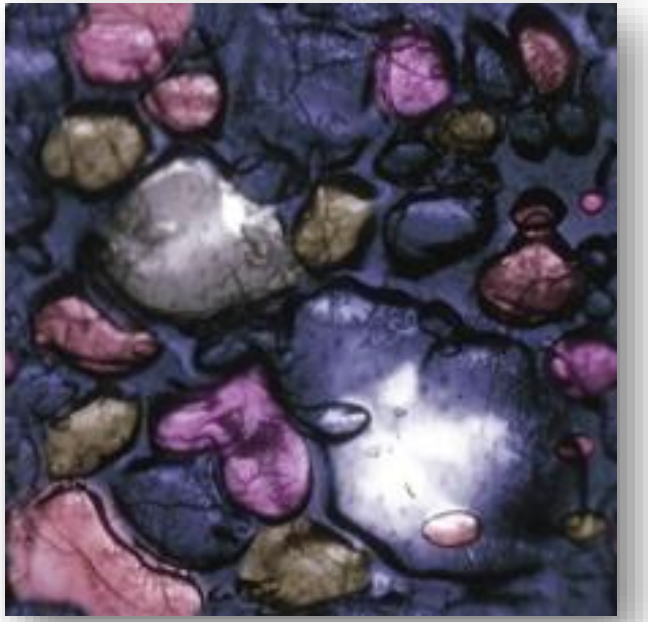


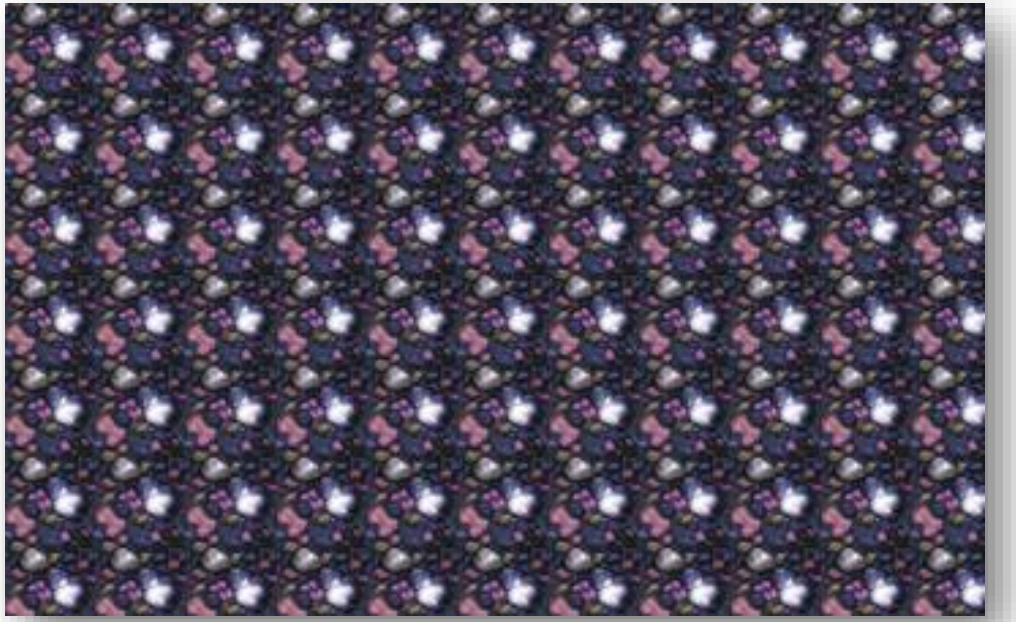
1

2

3

Tiling Bitmaps

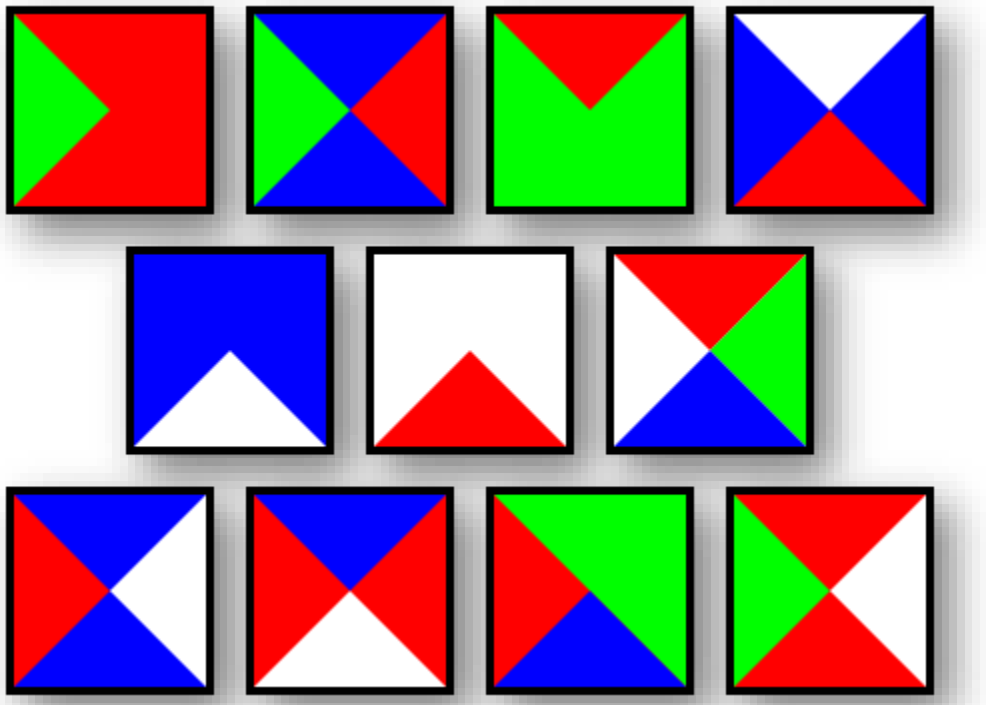


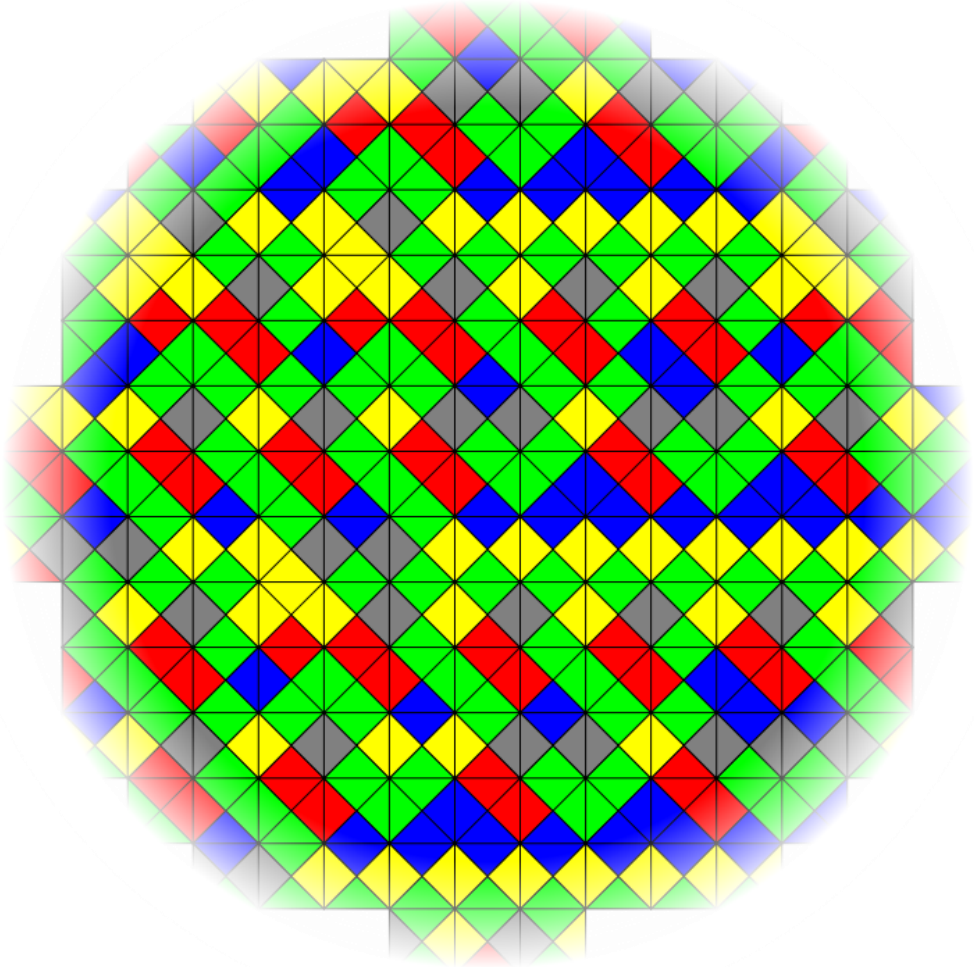


Tiled textures look **repetitive** and unnatural at distance.

Images: Inigo Quilezles

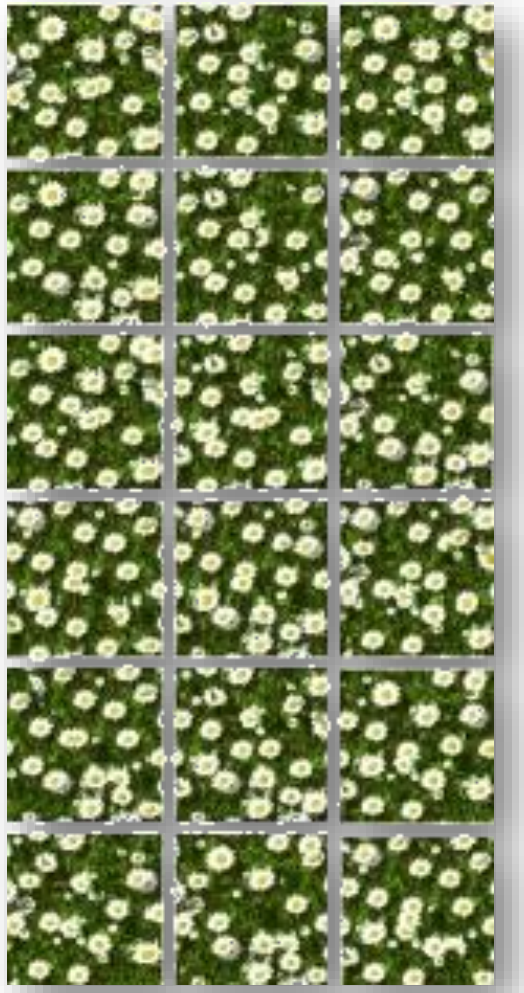
Tiling Bitmaps – Wang Tiles

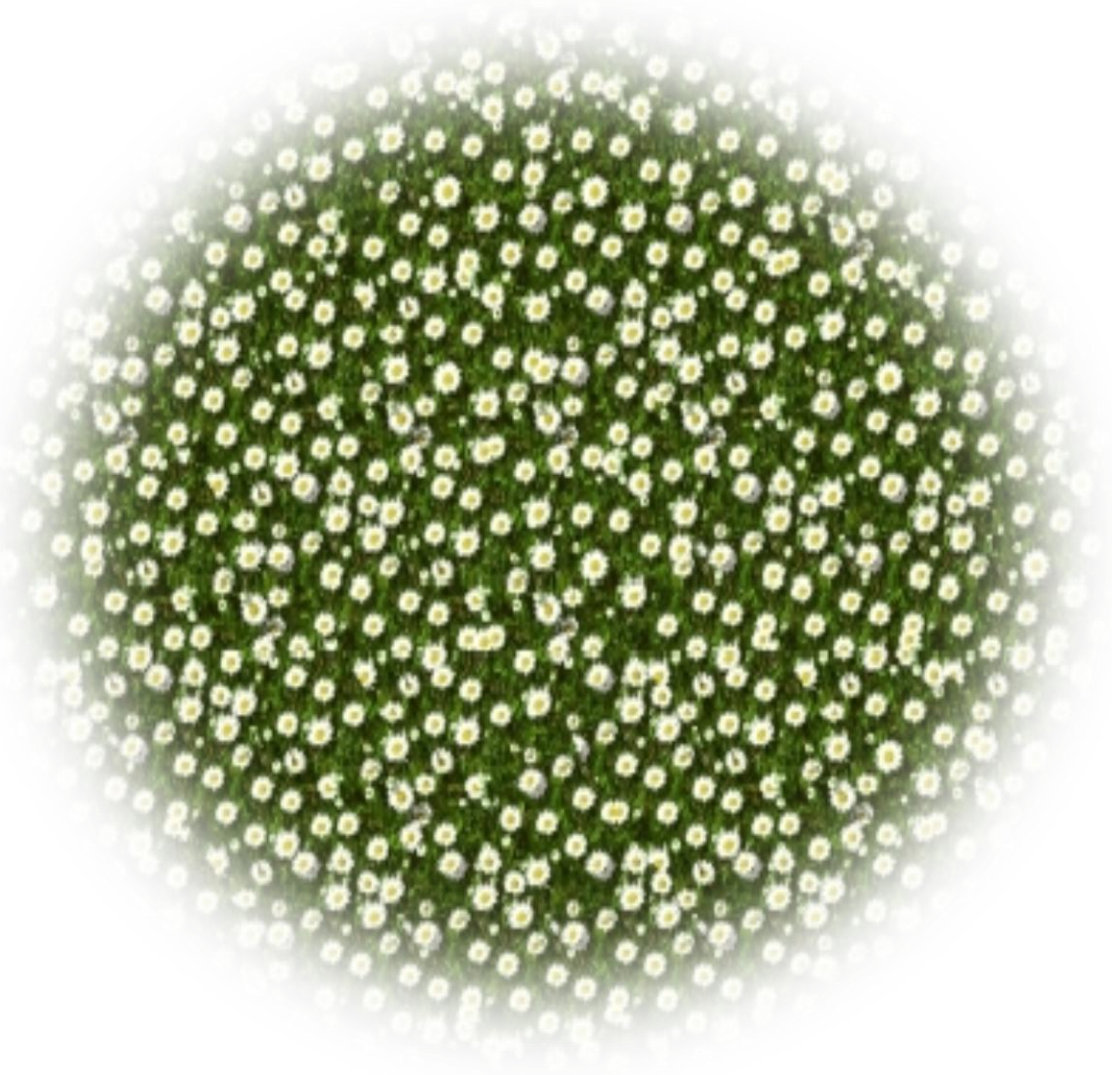




Images: Parcly Taxel & Claudio Rocchini

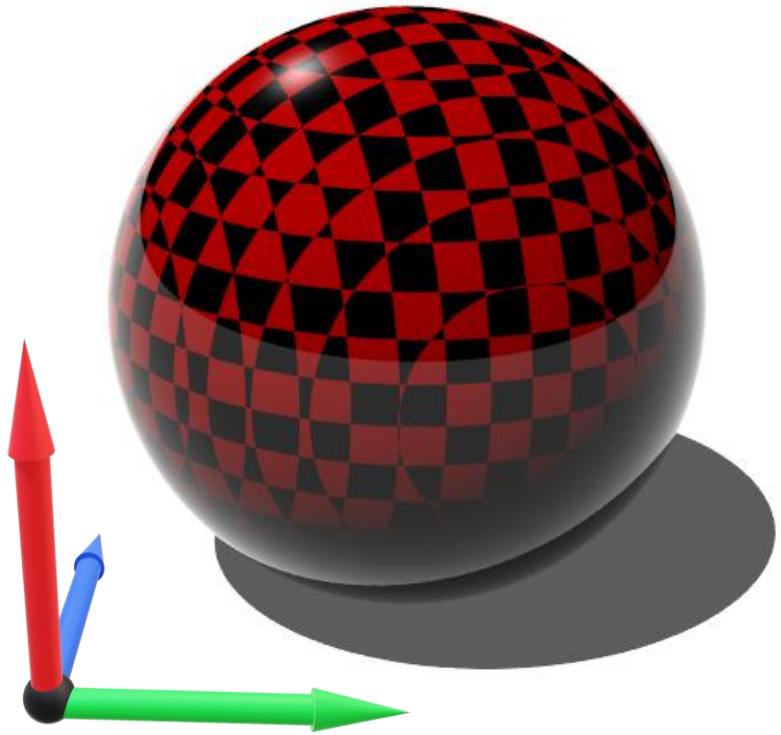
Tiling Bitmaps – Wang Tiles

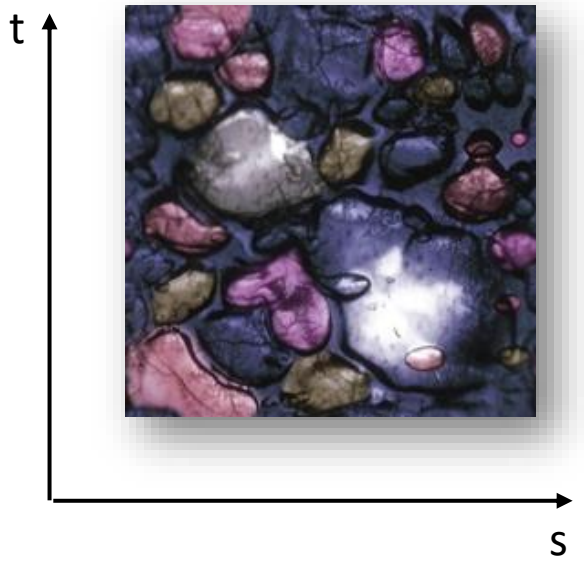




Images: Cohen, Shade, et. al

3D vs. 2D





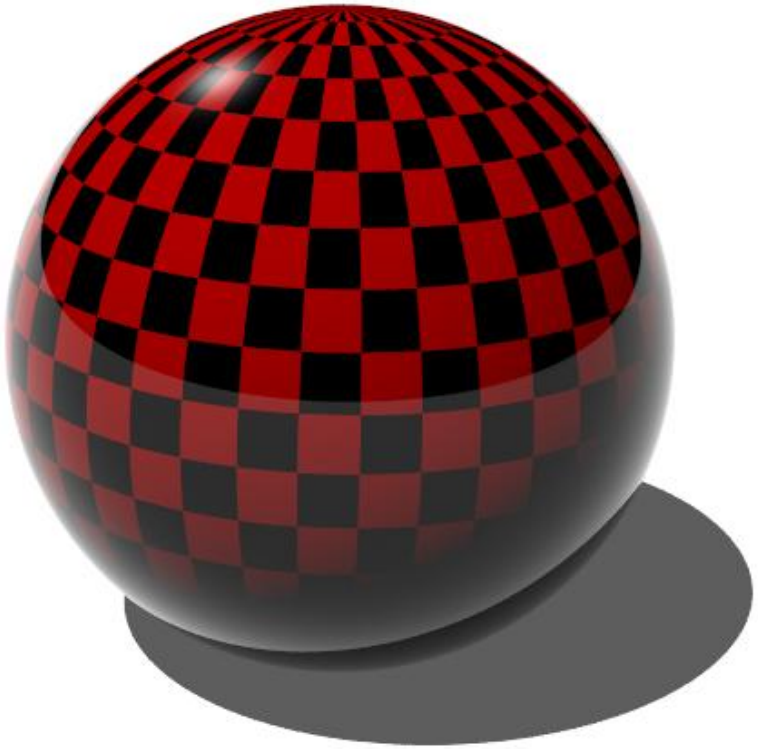
$$P = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

$$f(s,t)$$

Planar XY projection

$$s = \hat{\vec{n}}_x$$
$$t = \hat{\vec{n}}_y$$

Spherical projection

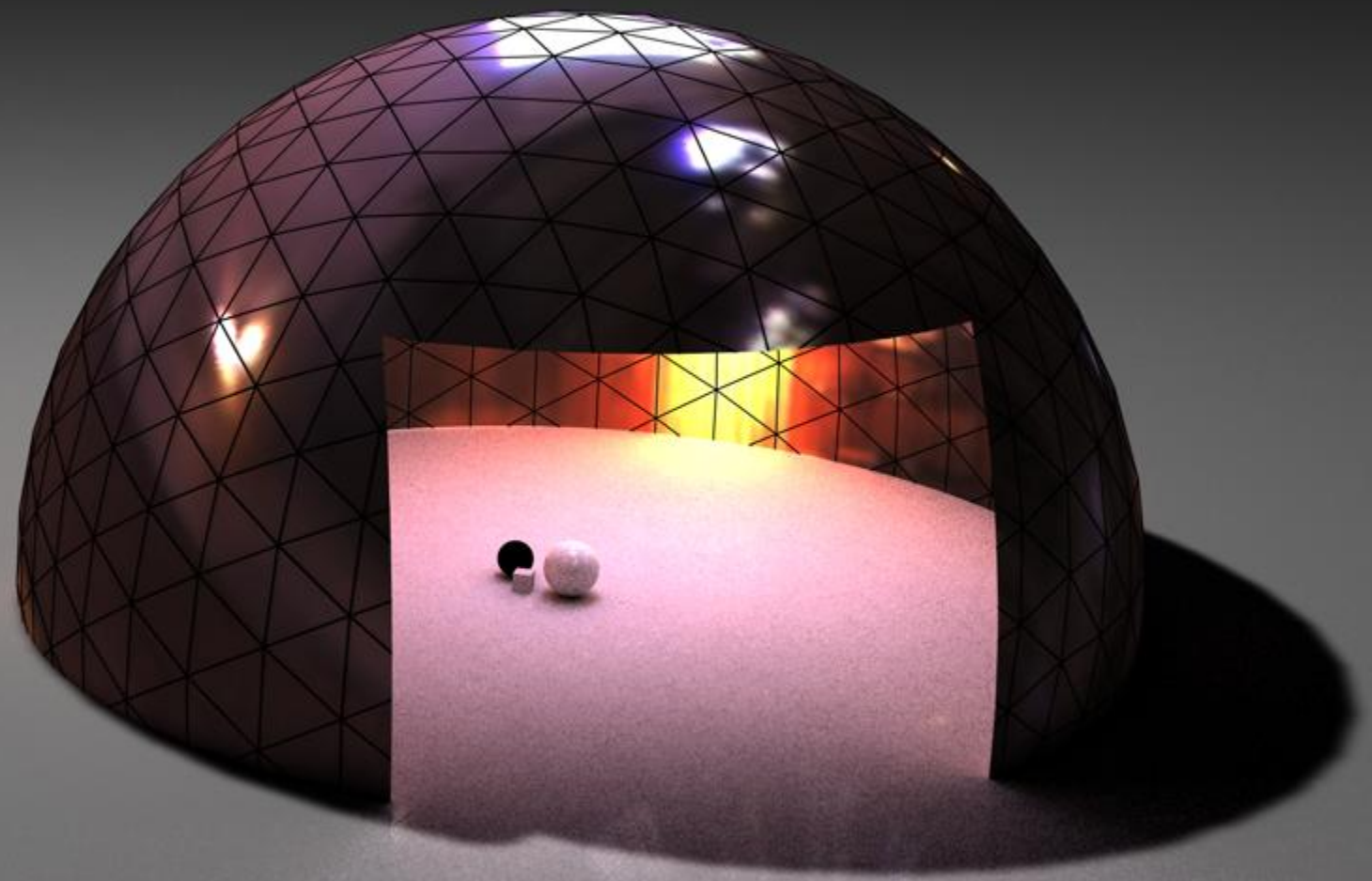


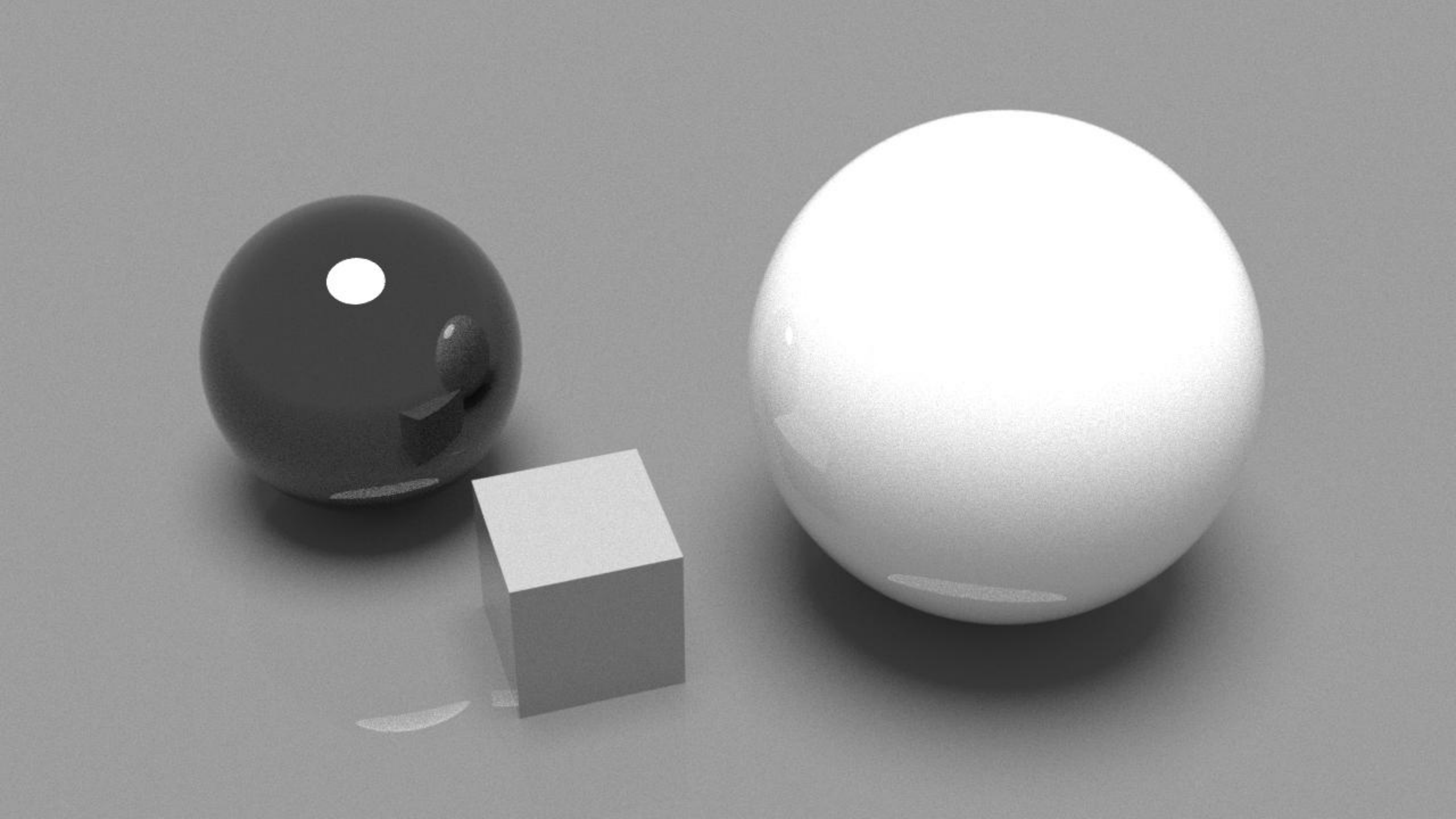
$$s = atan2\left(\hat{\vec{n}}_x, \hat{\vec{n}}_z\right)$$
$$t = \cos^{-1} \hat{\vec{n}}_y$$

Image-Based Lighting

Using Textures as Emission Color

Textured **Light Domes**
simulate realistic lighting.





Grace Cathedral, San Francisco
200'000:1 dynamic range, light probe by P. Debevec

High Dynamic Range (HDR) textures store brightness in special floating-point formats to capture both **very dark** areas and **bright lights** accurately.

The `.float` image format uses **no header**, and describes each pixel with **three single-precision floating-point numbers** (RGB).

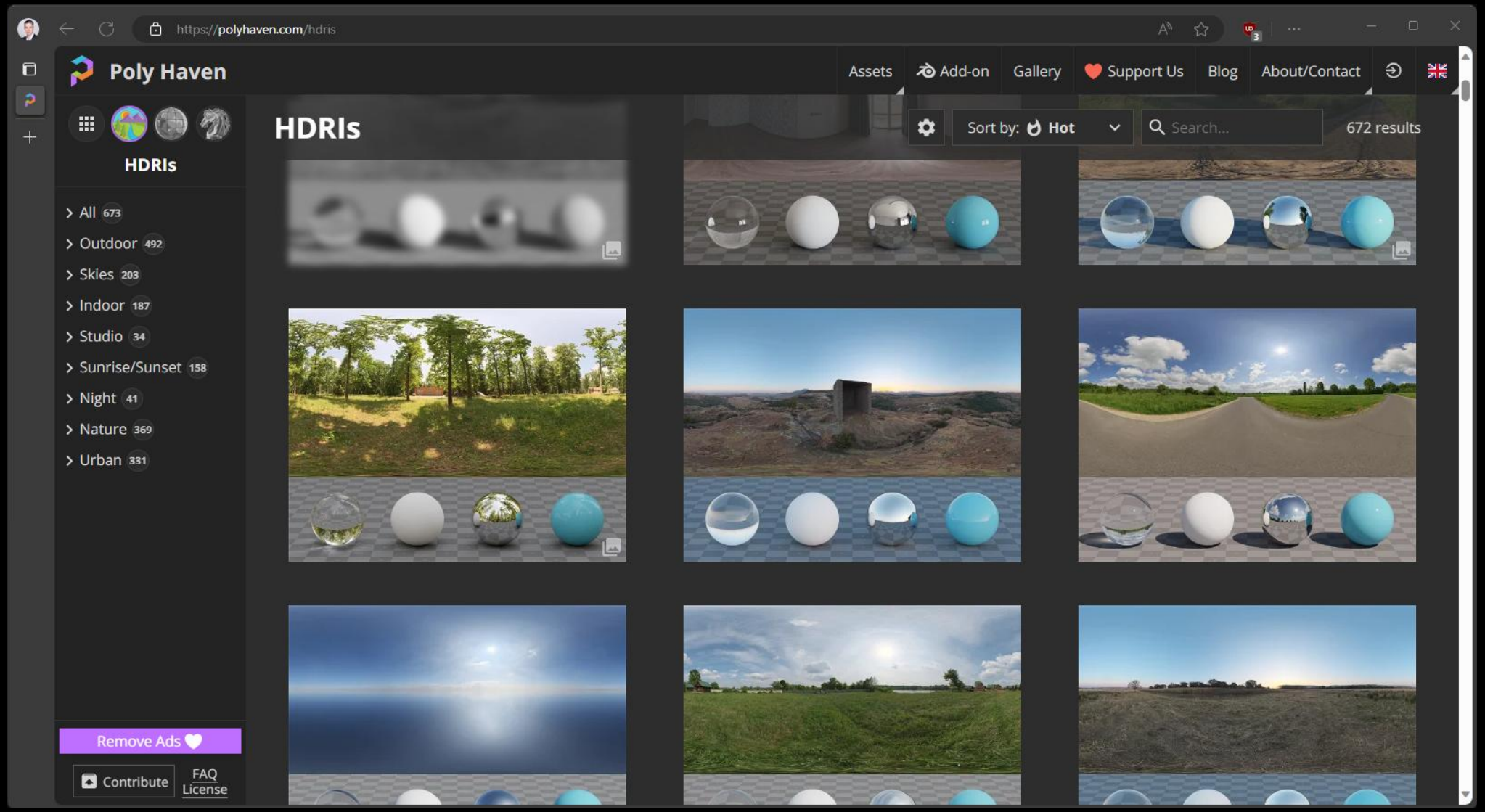
Grace Cathedral, San Francisco
200'000:1 dynamic range, light probe by P. Debevec

Spherical

Cube Map

Environment Maps can be stored in many alternative projections.

Eucalyptus Grove, UC Berkeley
5'000:1 dynamic range, light probe by P. Debevec



Environment Maps are freely available online.

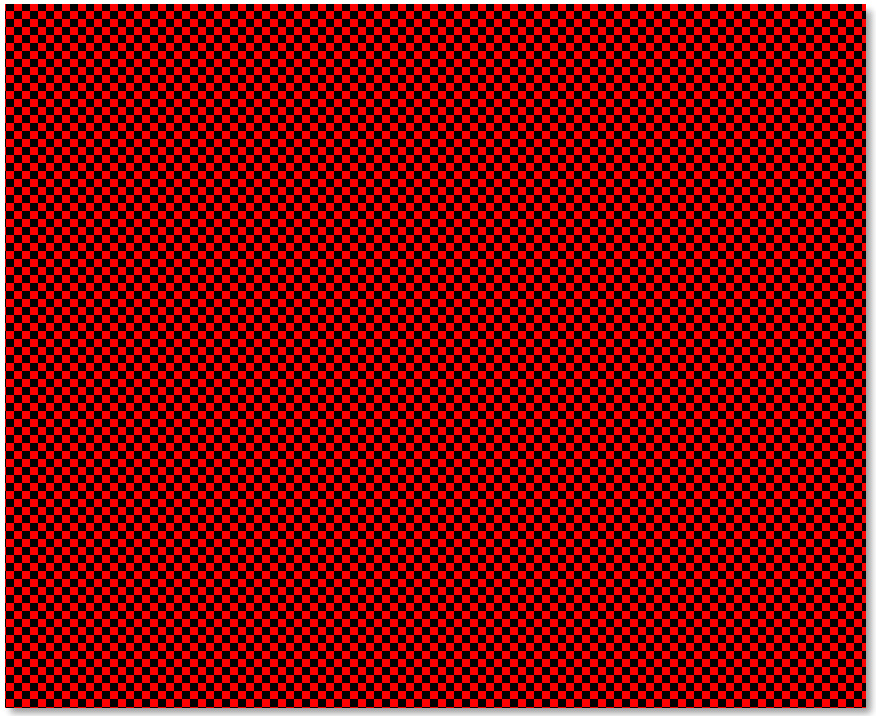
Texture Mapping

Bitmaps

Procedural

Procedural Textures

$$C_{checker}(x,y,z) := \begin{cases} \blacksquare, & frac(x) < 0.5 \oplus frac(y) < 0.5 \oplus frac(z) < 0.5 \\ \blacksquare, & otherwise \end{cases}$$

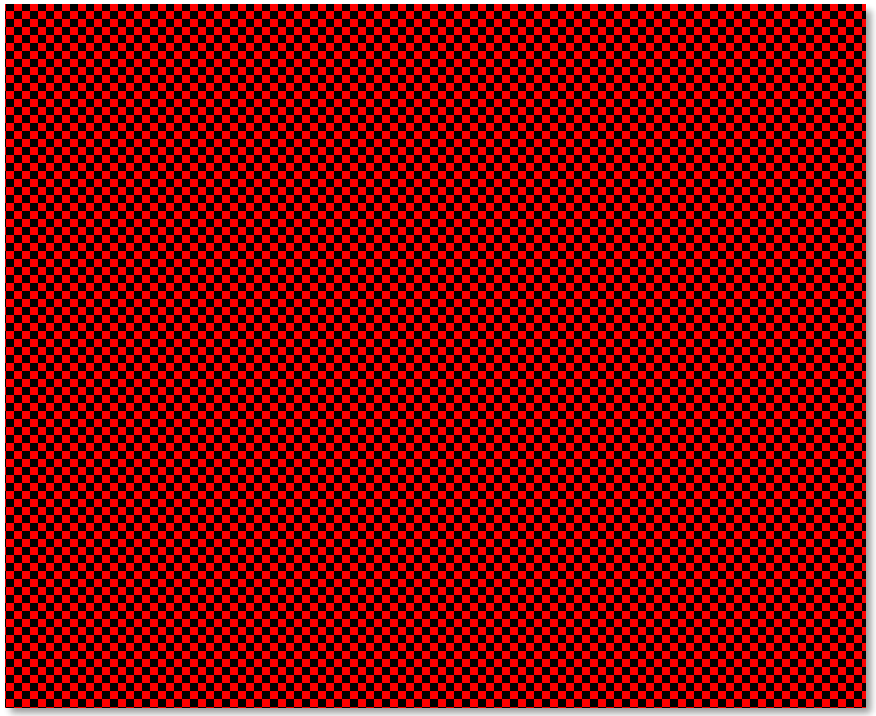


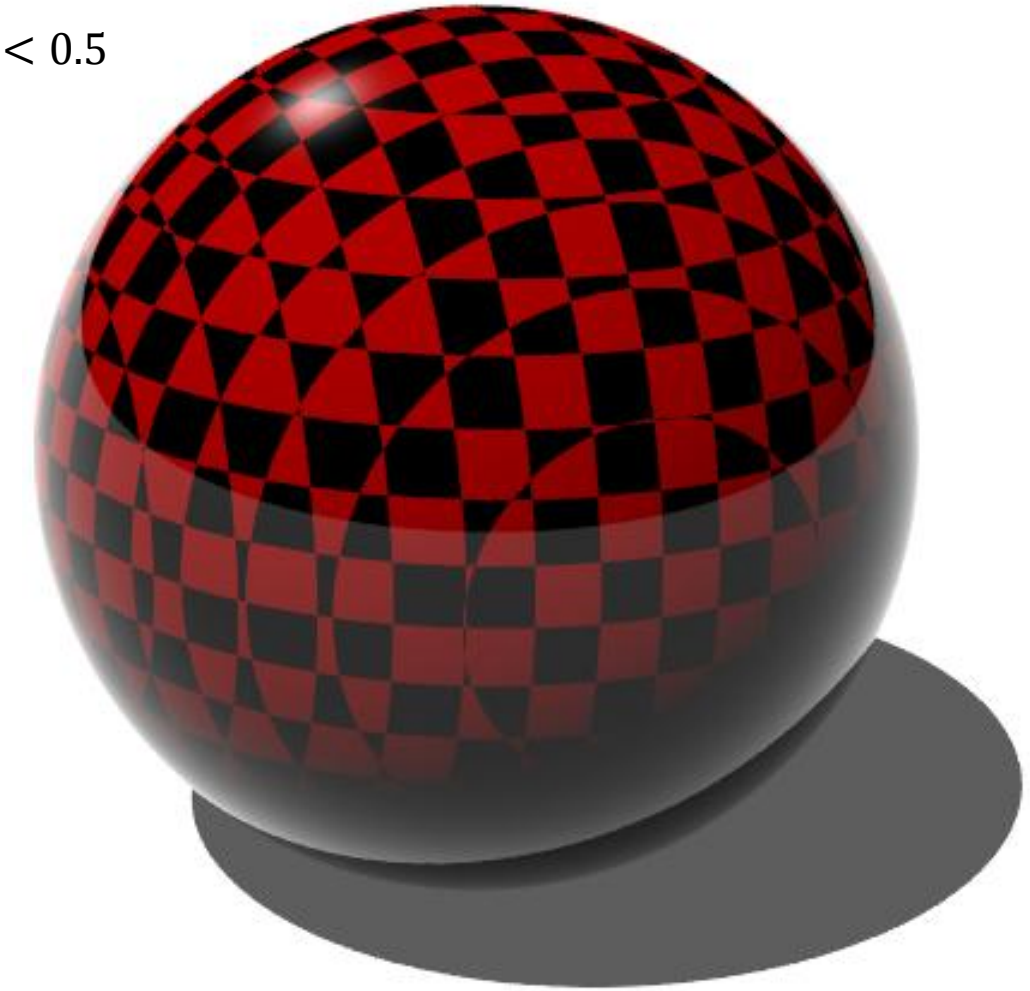
Images can be defined **mathematically**.

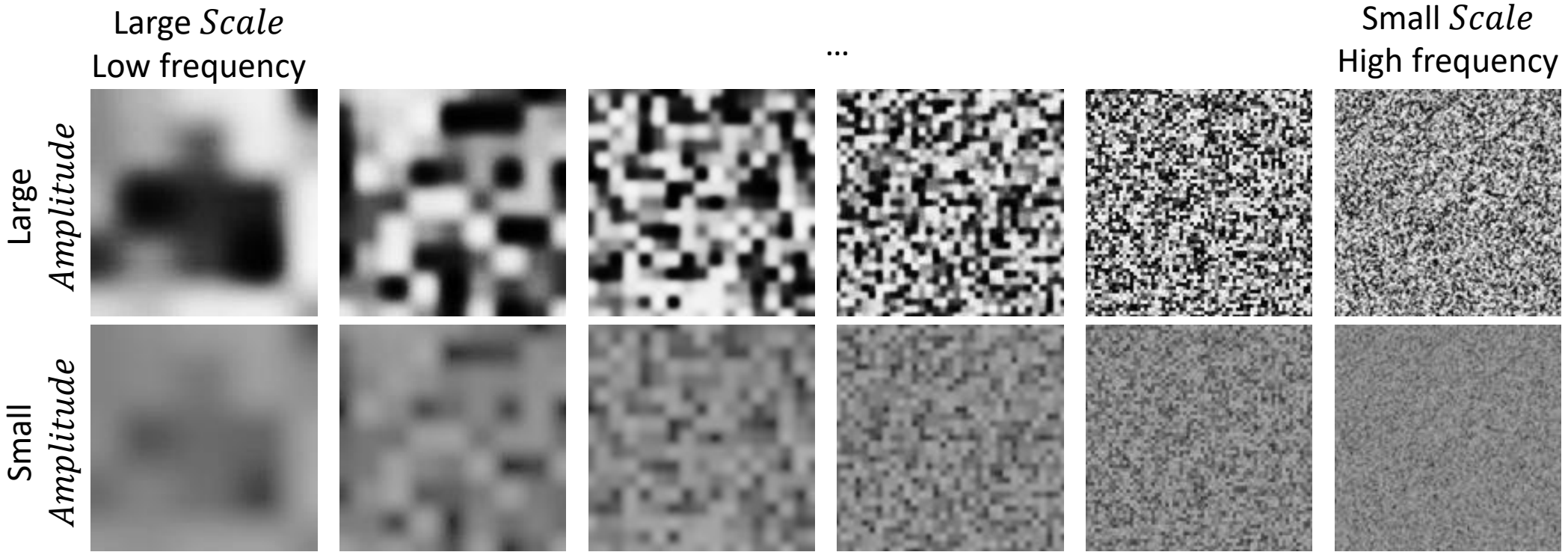
Procedural images have **infinite resolution** over an **infinite domain**.

Procedural Textures

$$C_{checker}(x,y,z) \coloneqq \begin{cases} \blacksquare, & frac(x) < 0.5 \oplus frac(y) < 0.5 \oplus frac(z) < 0.5 \\ \blacksquare, & otherwise \end{cases}$$







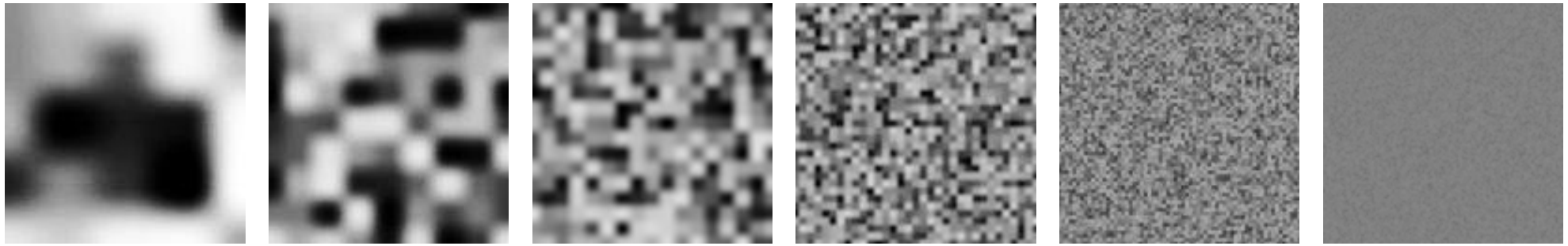
Pseudo-random numbers from a hash function look noisy

$$C_{noise}(x,y) \coloneqq Interpolate(Hash(\left\lfloor\frac{x}{Scale}\right\rfloor,\left\lfloor\frac{y}{Scale}\right\rfloor)) \cdot Amplitude$$

Large $Scale$
Low frequency

Small $Scale$
High frequency

$\sum$



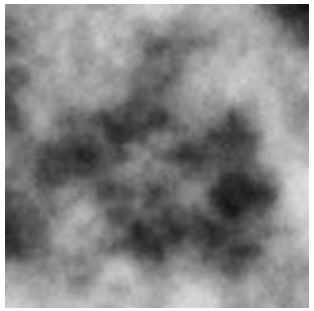
…

Large
$Amplitude$

…

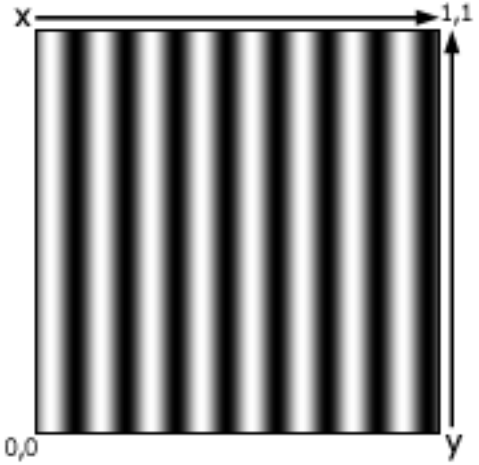
Small
$Amplitude$

$=$

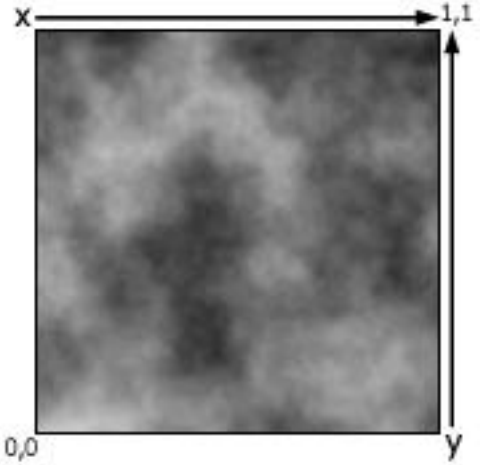


“Perlin Noise”

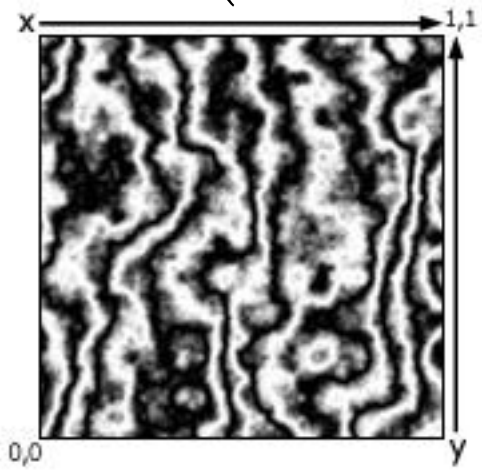
$C_{grid}(p)$



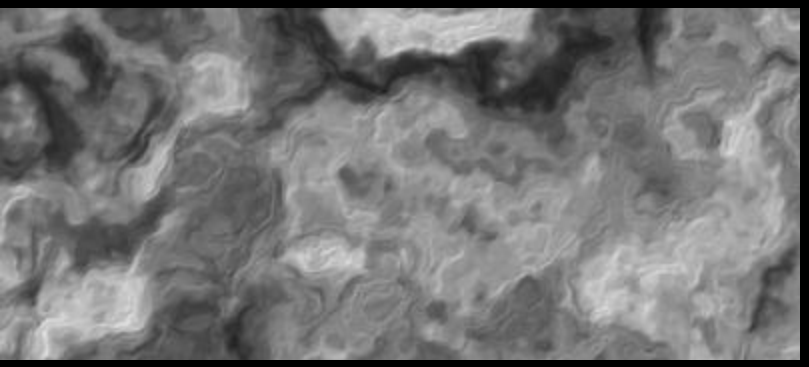
$C_{perlin}(p)$

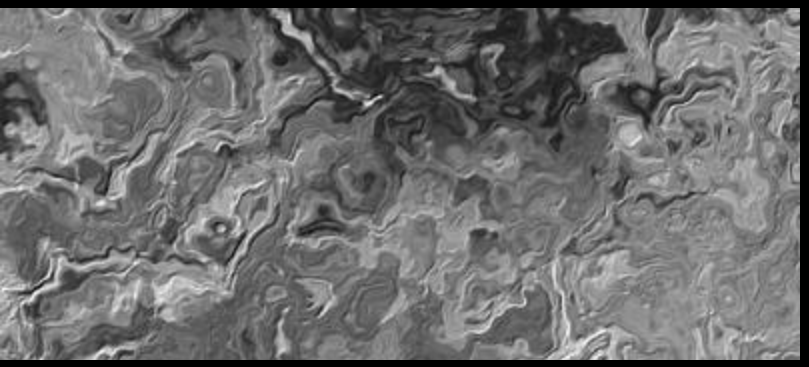


$$C_{grid}(p) \coloneqq C_{grid}\left(p + \begin{bmatrix}1\\0\end{bmatrix} C_{perlin}(p)\right)$$



Images: Shea McCombs





$$C_{perlin}(\mathrm{p})$$

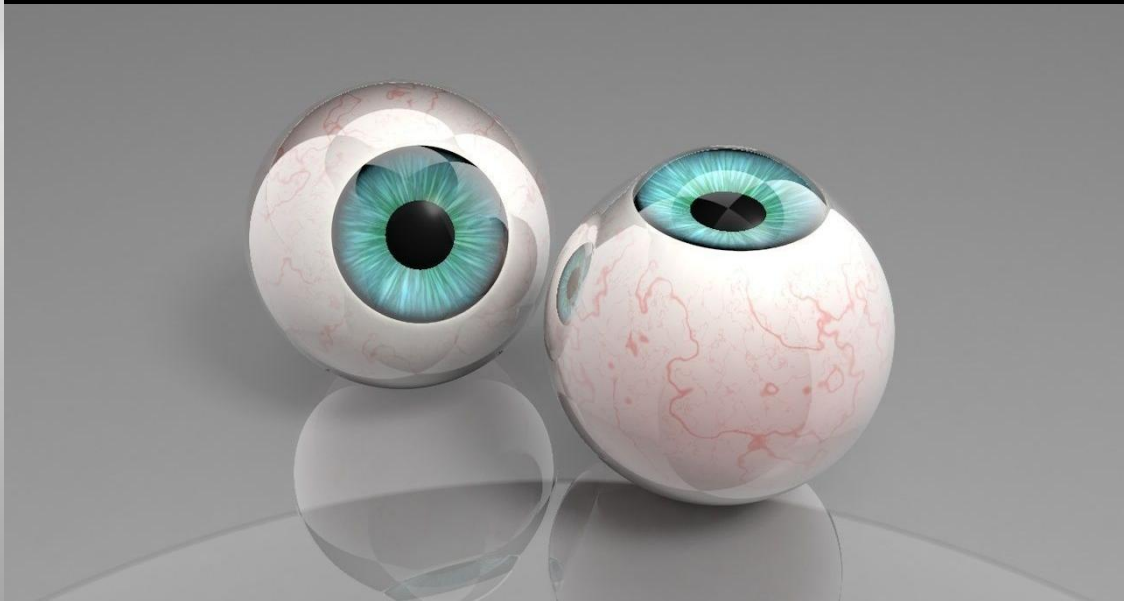
$$C_{perlin'} \coloneqq C_{perlin}\left(p + \begin{bmatrix} C_{perlin}(\mathrm{p}) \\ C_{perlin}(\mathrm{p} + [5.2 \quad 1.3]^{\tau}) \end{bmatrix}\right)$$

$$C_{perlin''} \coloneqq C_{perlin}\left(p + \begin{bmatrix} C_{perlin'}(\mathrm{p} + [1.7 \quad 9.2]^{\tau}) \\ C_{perlin'}(\mathrm{p} + [8.3 \quad 2.8]^{\tau}) \end{bmatrix}\right)$$

Colorize using $Lerp$

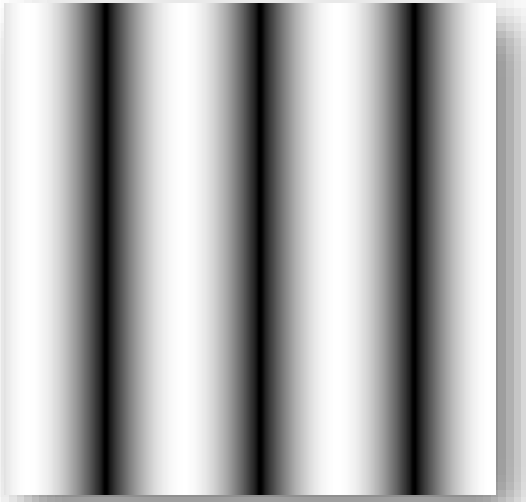
Images: Inigo Quilezles

Procedural materials are widely used to render grass, wood, leaves, rocks, mountains, water, …



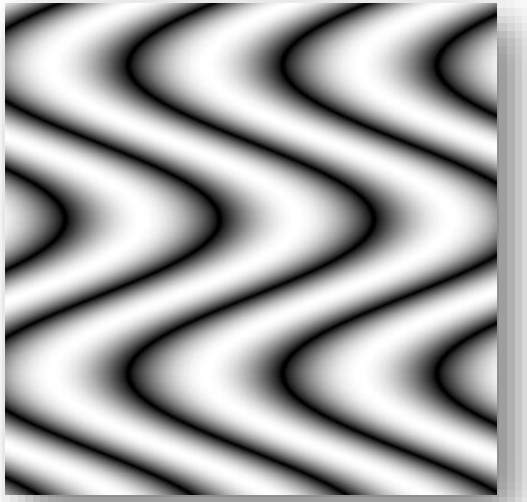
$$\begin{bmatrix}1\\1\\1\end{bmatrix}\sin(x)$$

1

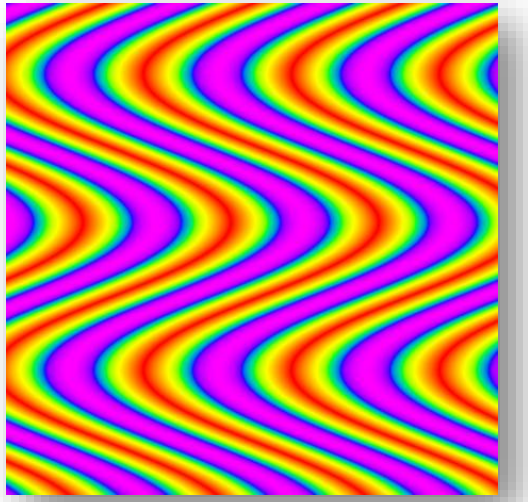


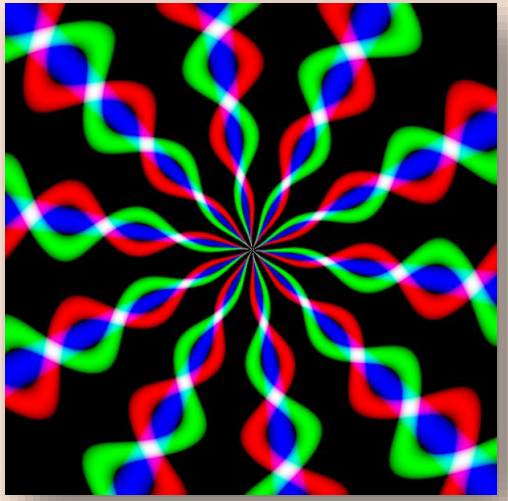
$$\begin{bmatrix}1\\1\\1\end{bmatrix}\sin(x+\cdots)$$

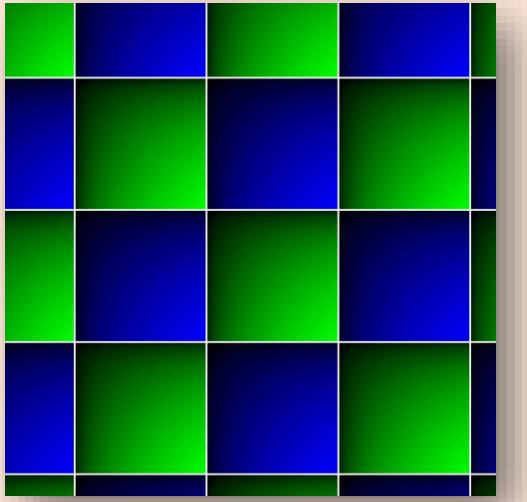
2

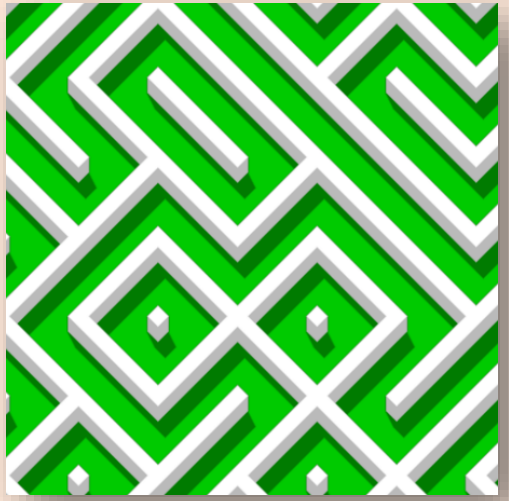


3









This Week's Lab

Add to the project:

- Gamma-correct **bitmap textures**, read from a file
- Create a new, **custom** scene using textures

1. Comment out the code from last week
2. Take screenshots of the results