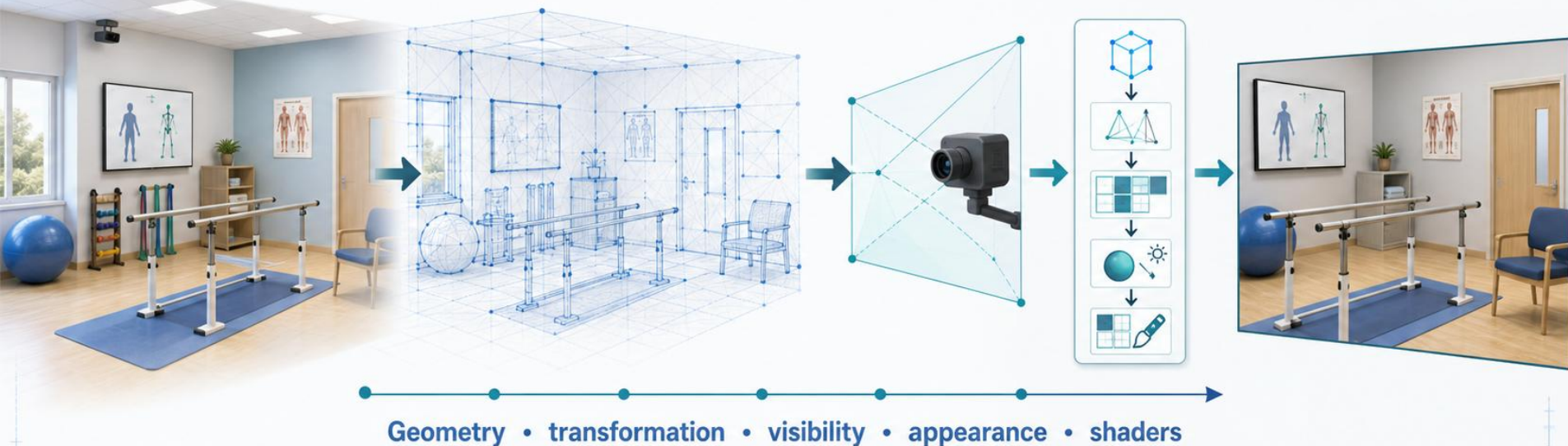


Graphics turns world representations into observations

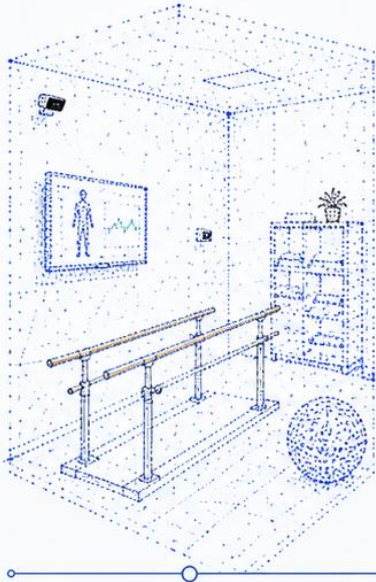
Computer Graphics Foundations

World representation $\rightarrow$ camera $\rightarrow$ image

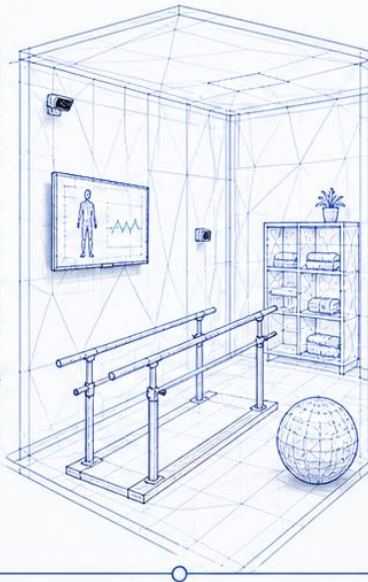


From represented state to observer-specific pixels

One place, five representations



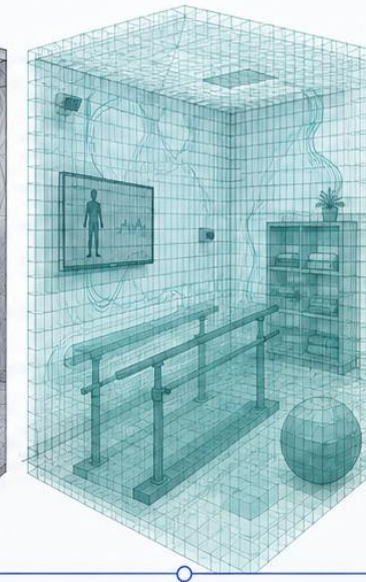
Points



Wireframe



Triangle surface

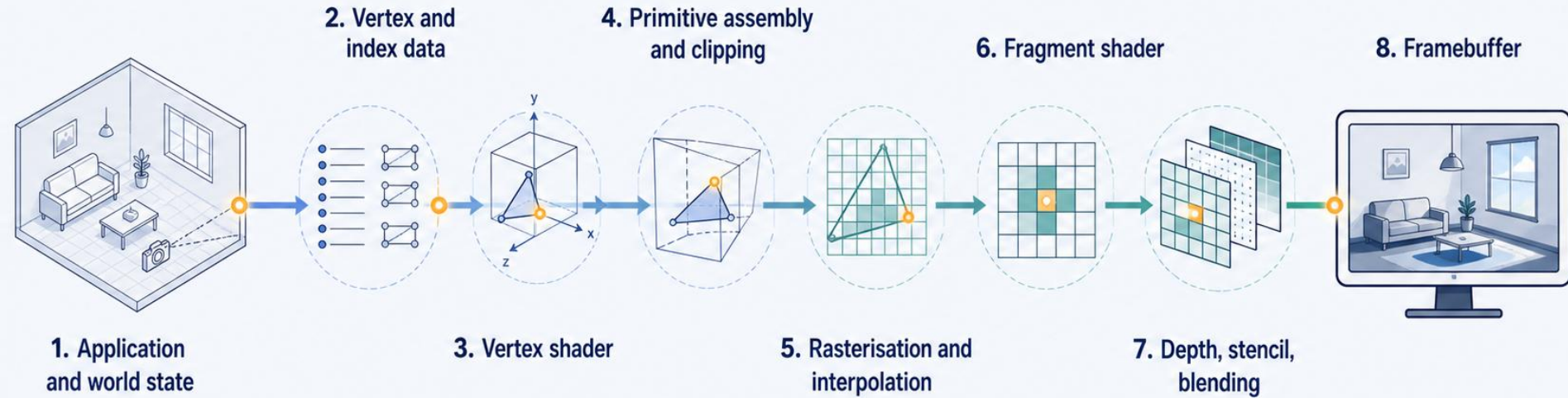


Volume or field

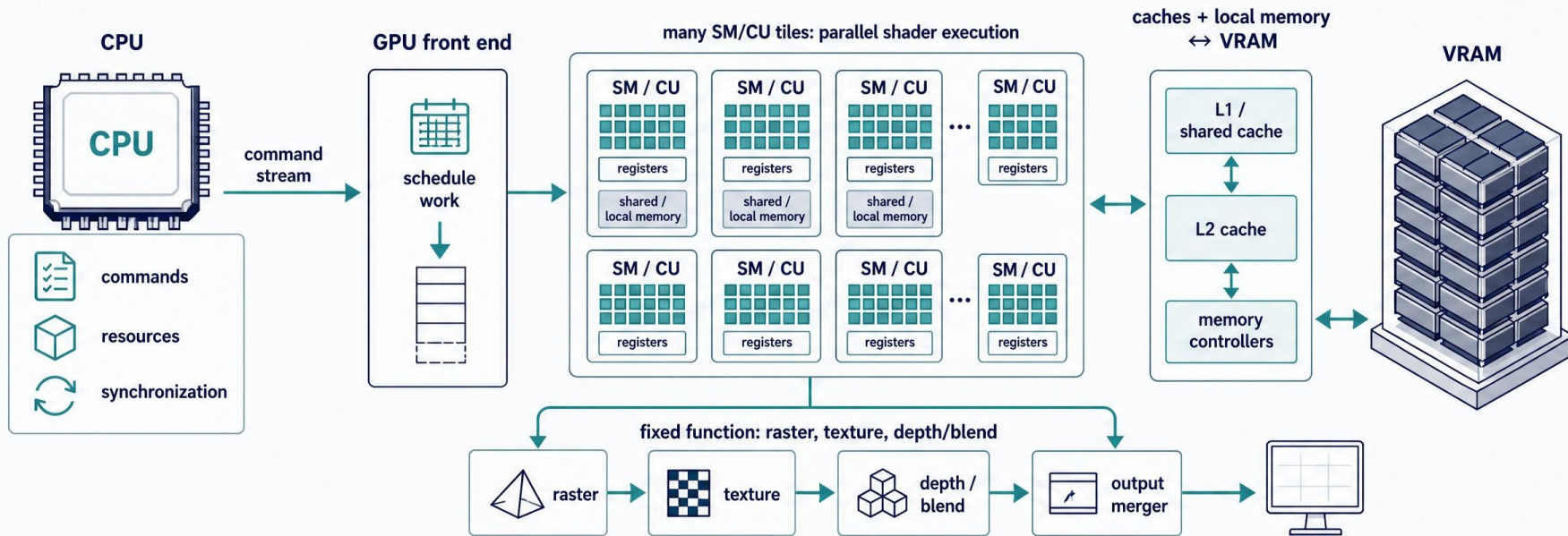


Rendered observation

A frame is produced by a pipeline



GPUs trade sequential control for massive throughput



CPU: commands, resources, synchronization



GPU front end: schedule work



Many SM/CU tiles: parallel shader execution

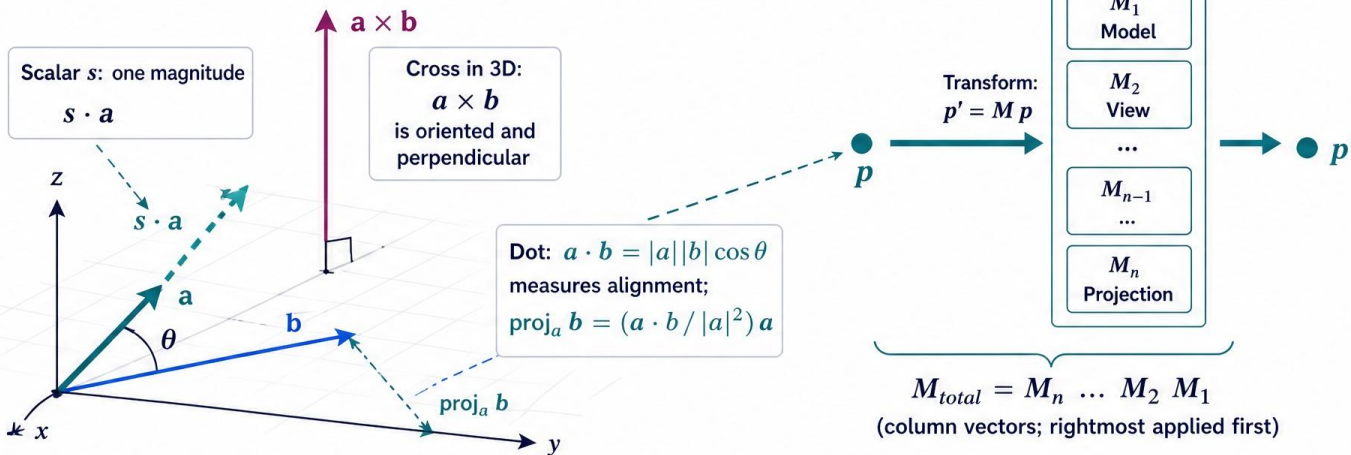


Caches + local memory ↔ VRAM



Fixed function: raster, texture, depth/blend

Graphics composes geometry with linear algebra



S Scalar s :
one magnitude

V Vector v :
components, direction,
displacement

D Dot:
 $a \cdot b = |a||b| \cos \theta$

C Cross in 3D:
 $a \times b$ is oriented
and perpendicular

M Transform:
 $p' = M p$

Homogeneous coordinates

Point (position):

$$p_h = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$w = 1$

Direction (vector):

$$v_h = \begin{bmatrix} x \\ y \\ z \\ 0 \end{bmatrix}$$

$w = 0$

4×4 homogeneous matrix maps points:

$$p'_h = M p_h$$

$$M = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix}$$

Order matters: matrix multiplication is not commutative

A then B (apply A, then B)
result = BA

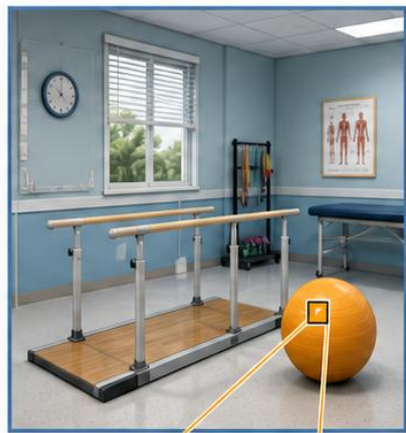


B then A (apply B, then A)
result = AB

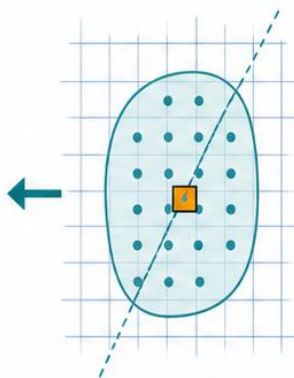


⚠ Different orders produce different results (unless matrices commute).

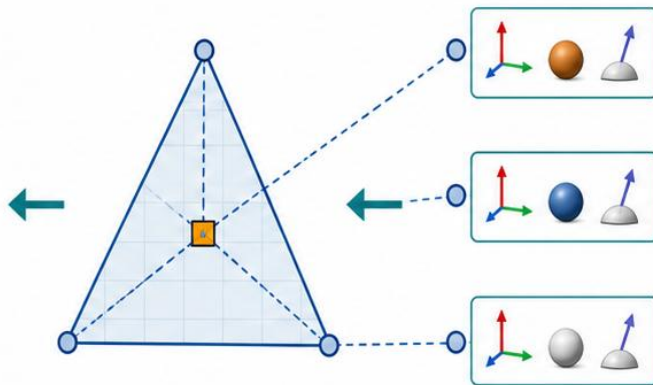
Pixels know neither objects nor meaning



Pixel value

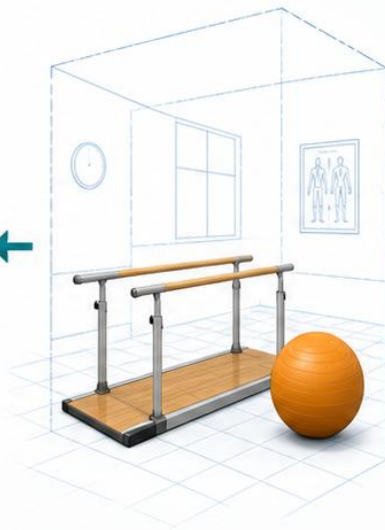


Fragment



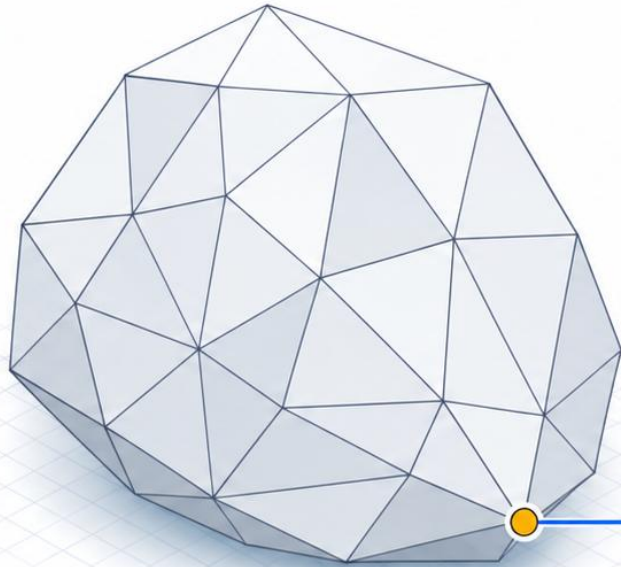
Triangle

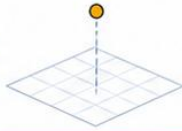
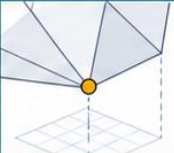
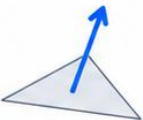
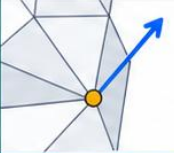
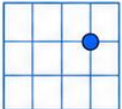
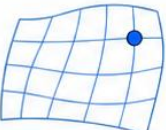
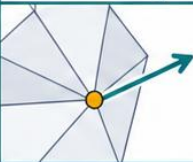
Vertex attributes



Object representation

A vertex is a record of attributes



	Position	
	Normal	
	Colour	
	Texture coordinate	
	Tangent	

Position is only one vertex attribute

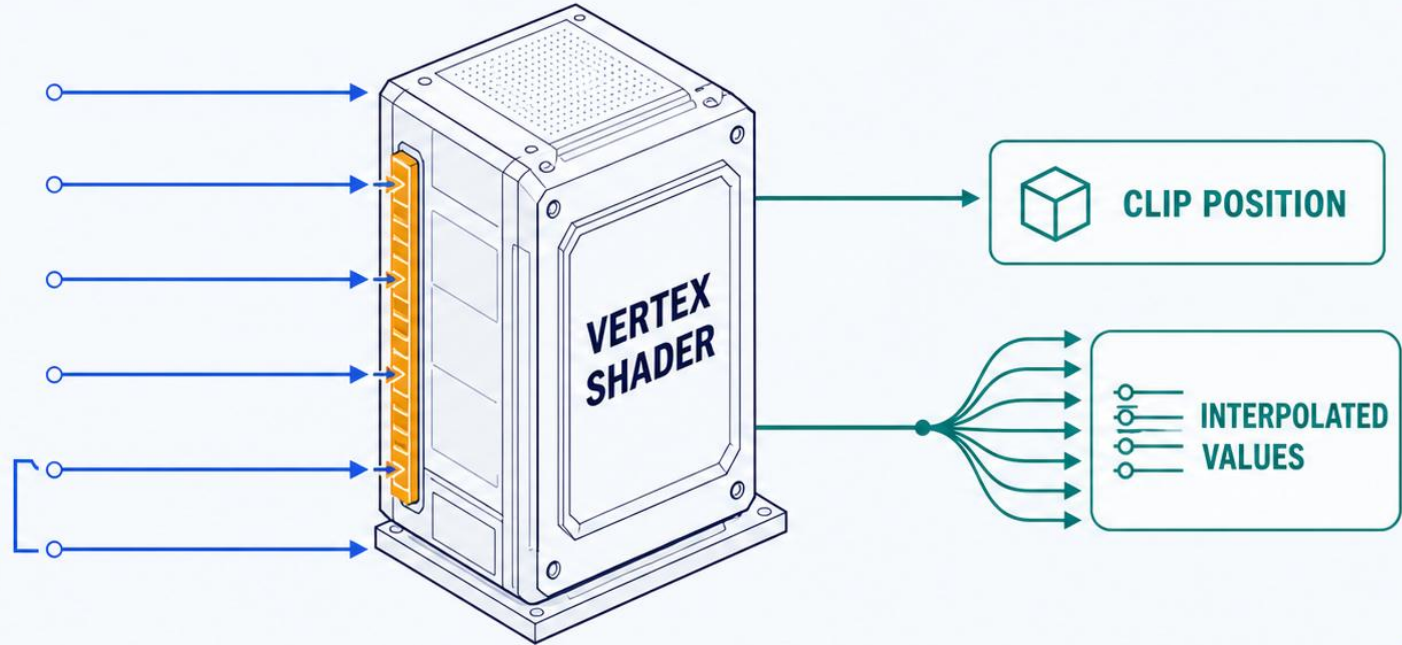
`a_position` → geometry

`a_normal` → orientation

`a_uv` → texture addressing

`a_color` → colour or mask

`a_joints` + `a_weights` →
deformation



Buffers give the GPU structured streams

Typed-array bytes

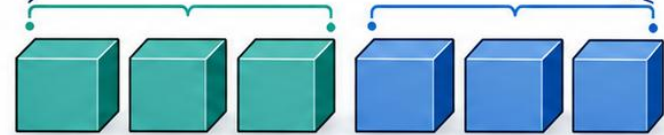


Stride and offset

Buffer object



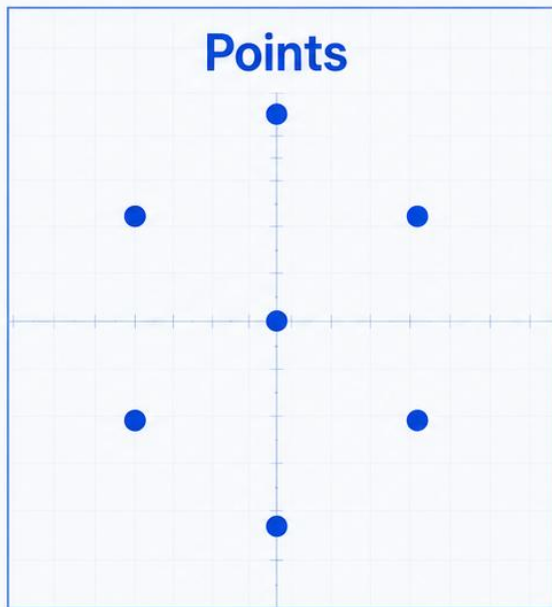
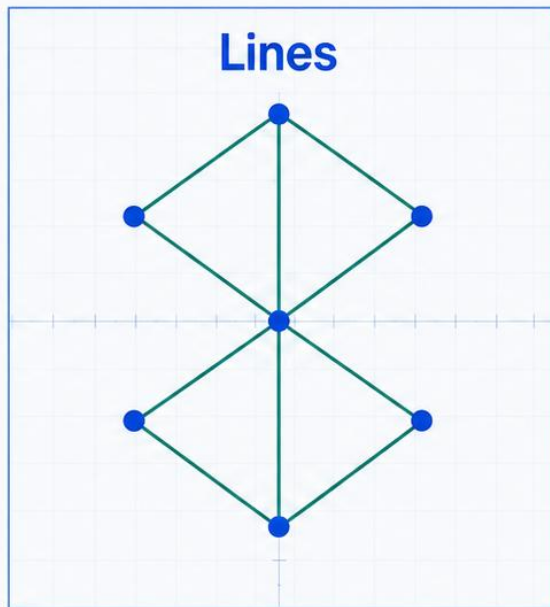
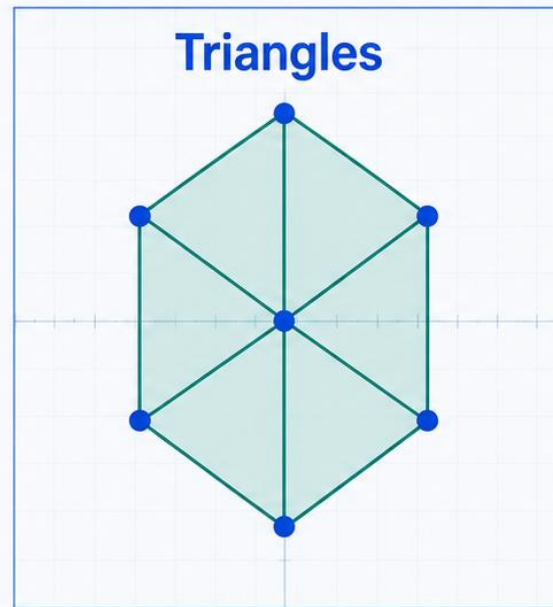
Component type and count



Interleaved versus separate data



Primitive assembly gives vertices geometric meaning

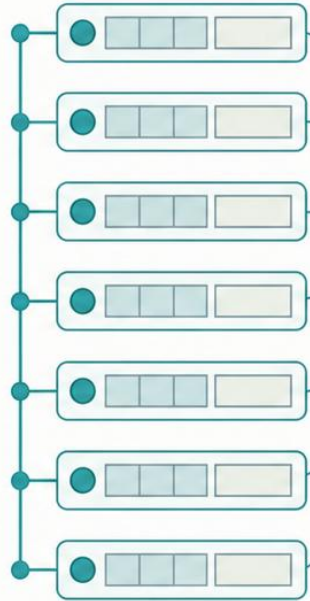


Same vertices, different assembly

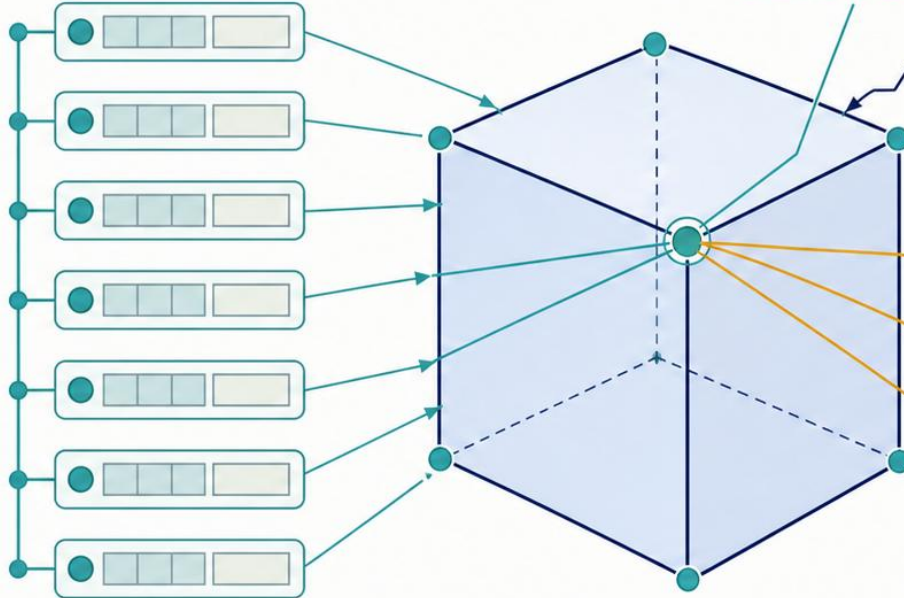
Primitive $\neq$ mesh

Indices reuse vertices, but seams require duplicates

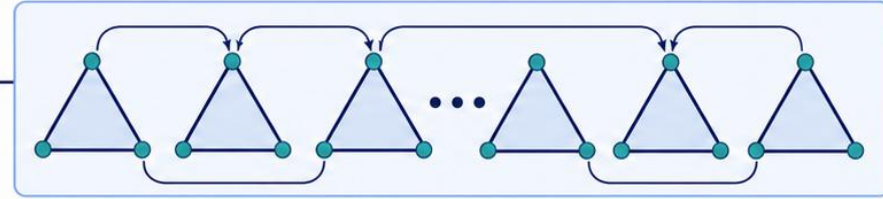
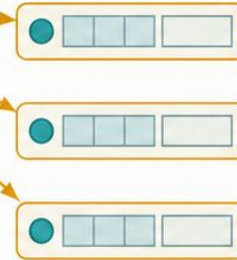
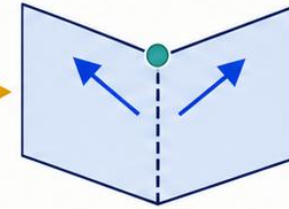
Vertex buffer = records



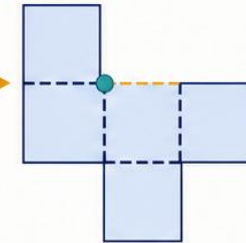
Shared position



Index buffer = connectivity and reuse

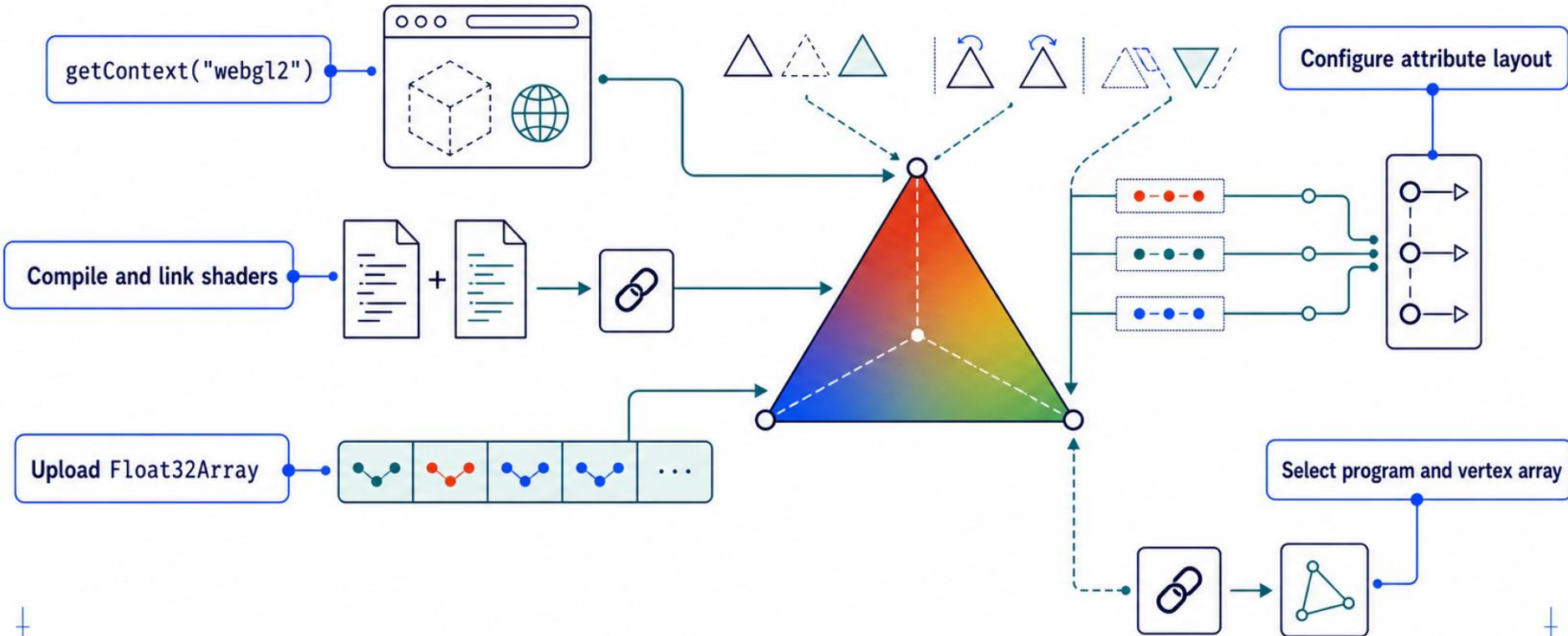


Hard-normal seam



UV seam

Raw WebGL exposes the contract

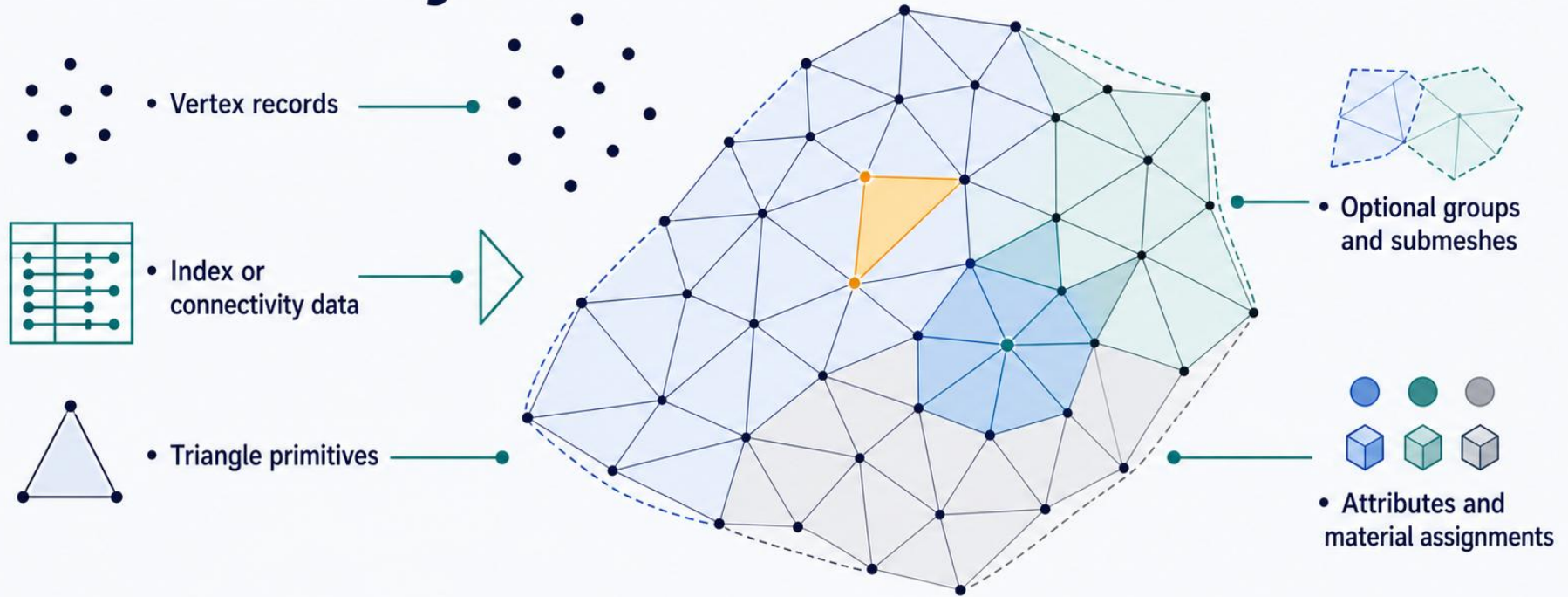


A point cloud samples space without declaring a surface



Point cloud data $\neq$ point primitive

A triangle mesh adds explicit connectivity



Line meshes visualize structure without defining a surface



Indexed segments and polylines



``LINES`` • ``LINE_STRIP`` • ``LINE_LOOP``



Wireframes • paths • skeletons • graphs



Bounds • contours • axes • diagnostics



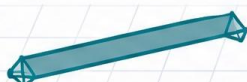
Native wide lines are not portable in WebGL

Native one-pixel line vs. triangle-based thick stroke

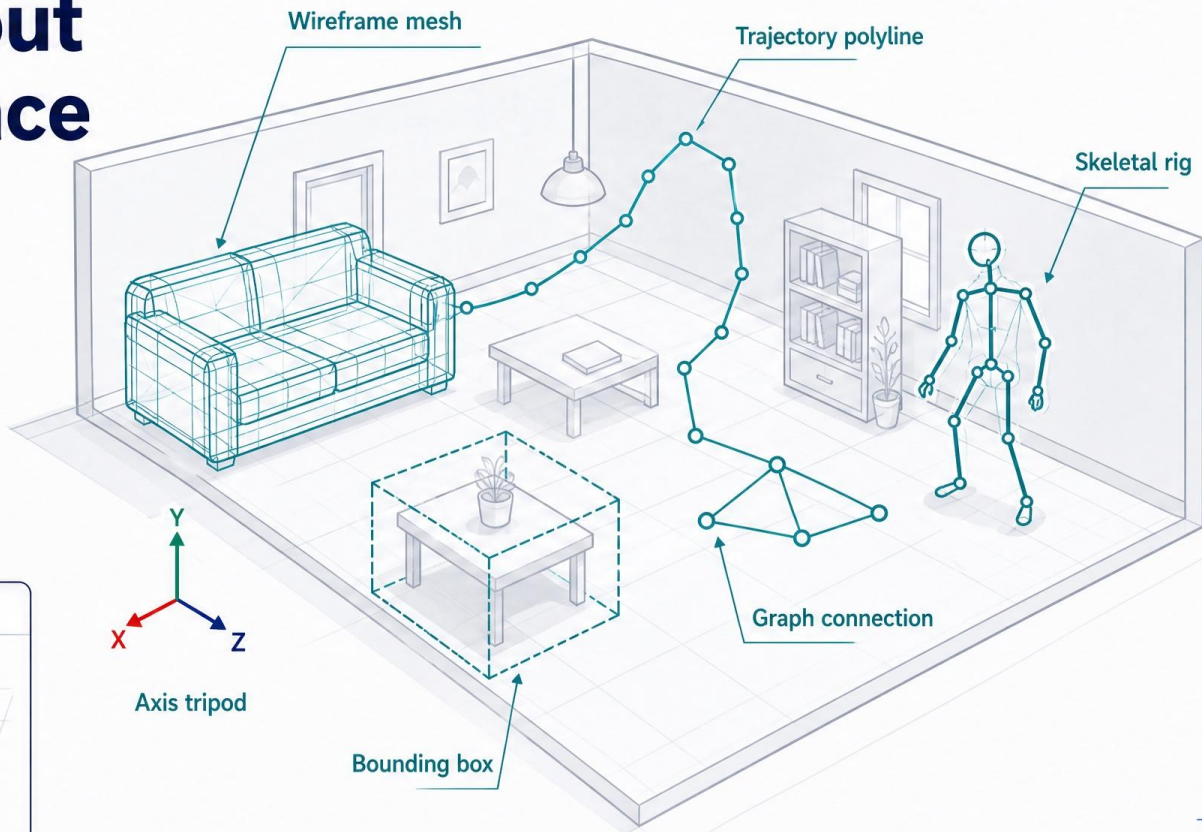
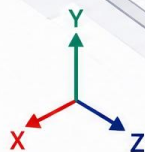
Native one-pixel line



Triangle-based thick stroke



Thick strokes built from camera-facing triangles are portable.



Axis tripod

Bounding box

Graph connection

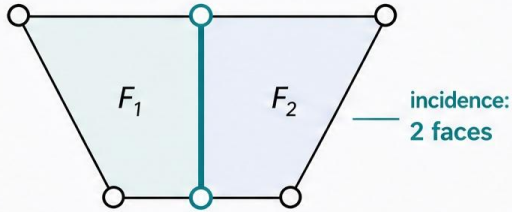
Skeletal rig

Trajectory polyline

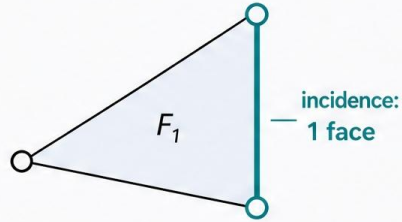
Wireframe mesh

Topology determines valid operations

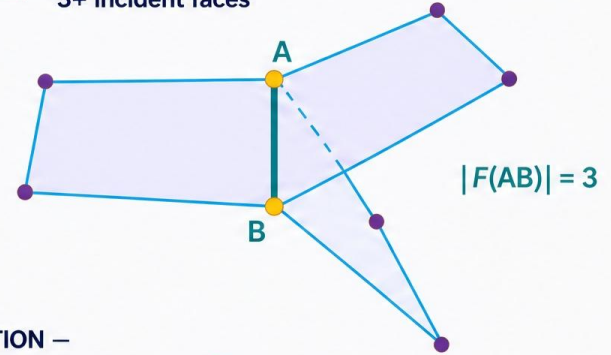
- 1** MANIFOLD INTERIOR EDGE –
2 incident faces



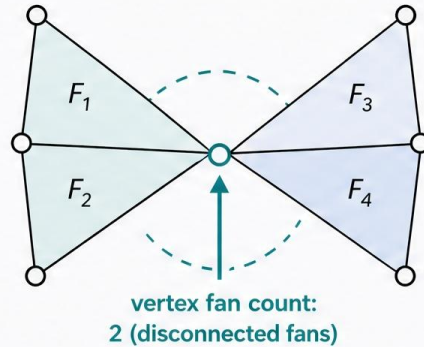
- 2** BOUNDARY EDGE –
1 incident face



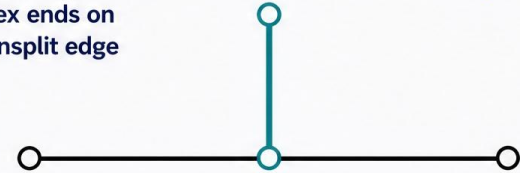
- 3** NON-MANIFOLD EDGE –
3+ incident faces



- 4** NON-MANIFOLD VERTEX –
disconnected face fans meet
at one vertex

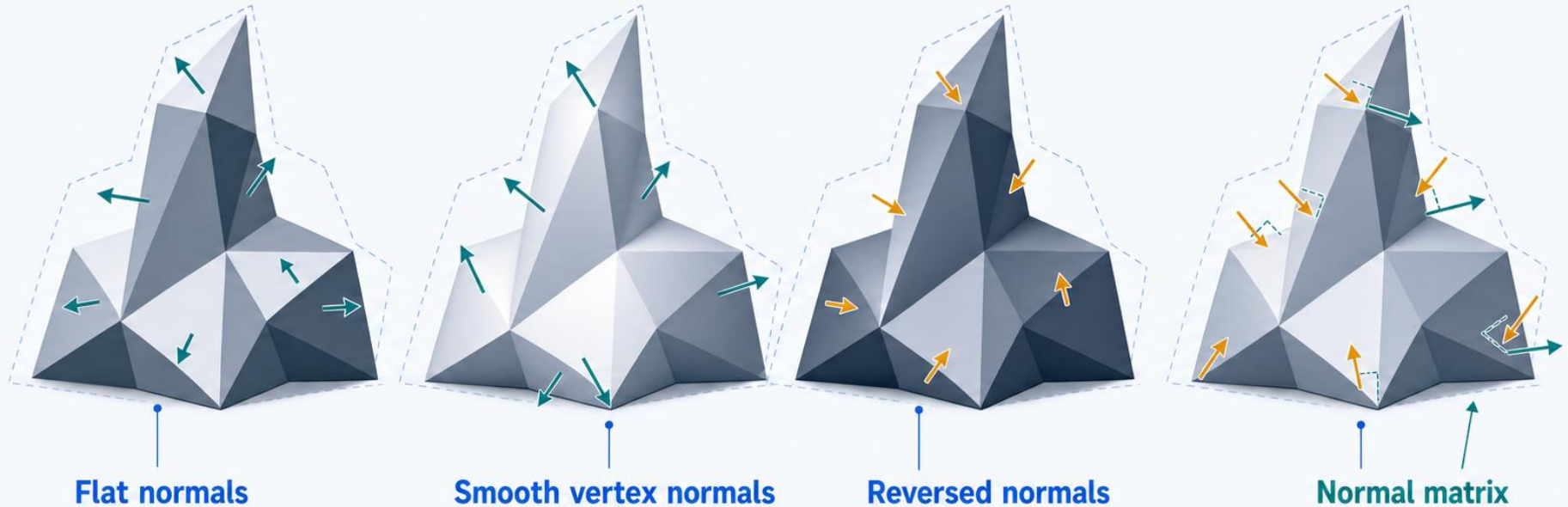


- 5** T-JUNCTION –
vertex ends on
an unsplit edge



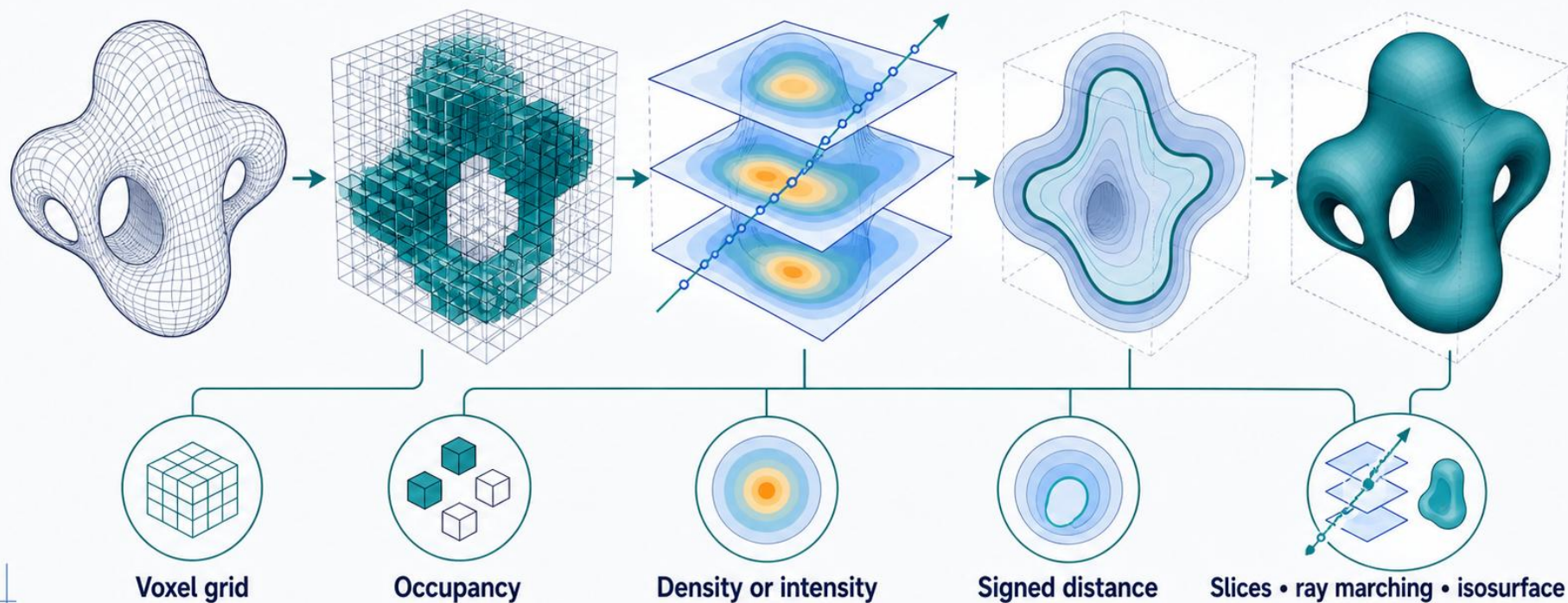
Closed • consistently oriented • watertight are separate global checks

Normals change shading, not shape



Same positions and silhouette

A volume stores values throughout a 3D domain



Representation should follow operation and evidence

Opaque display → triangle surface



Raw depth acquisition → point cloud



Density inspection → volume



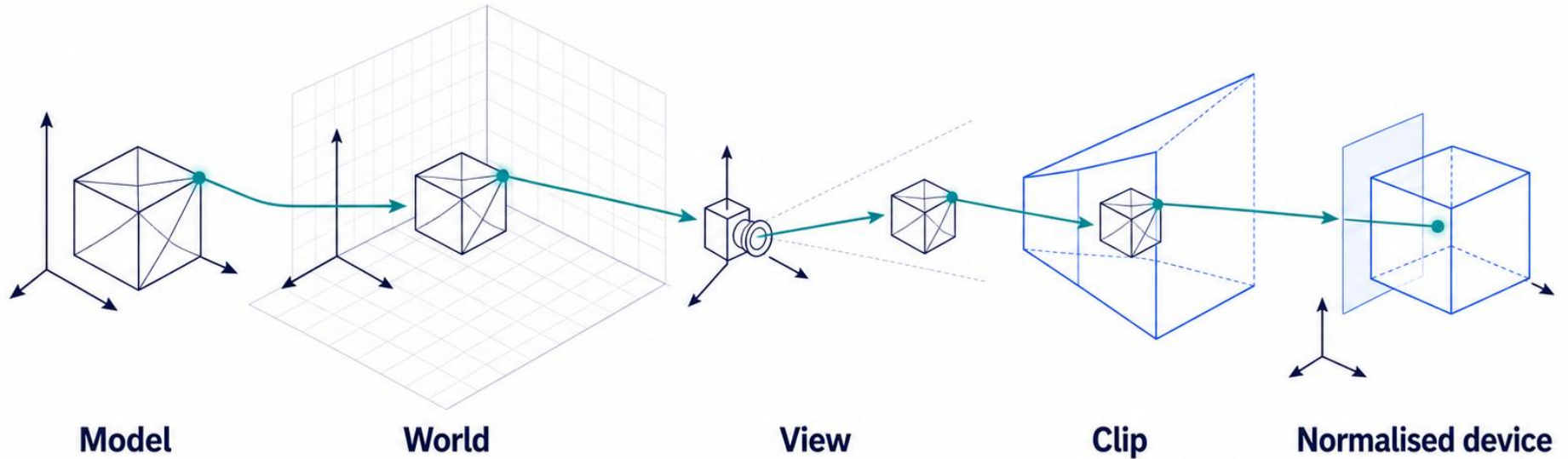
Distance query → signed distance field



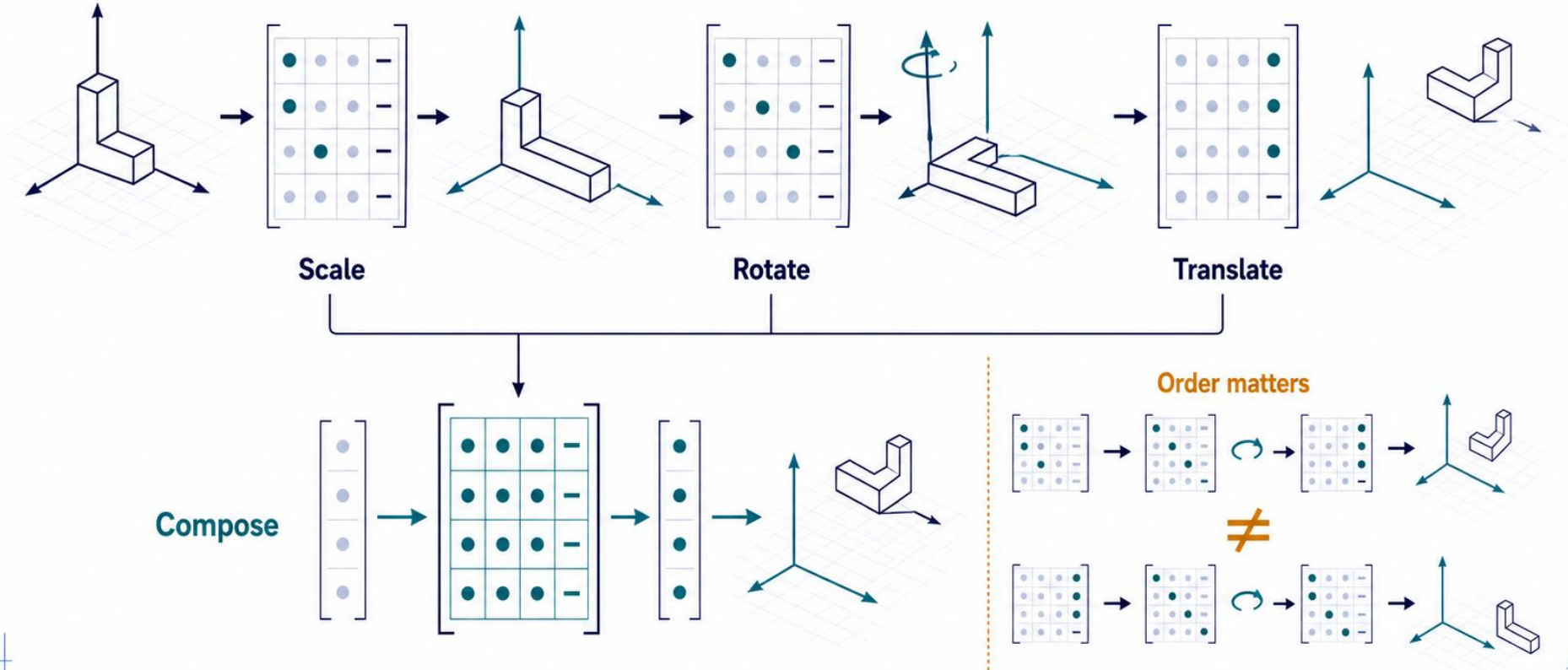
Navigation → graph or semantic structure



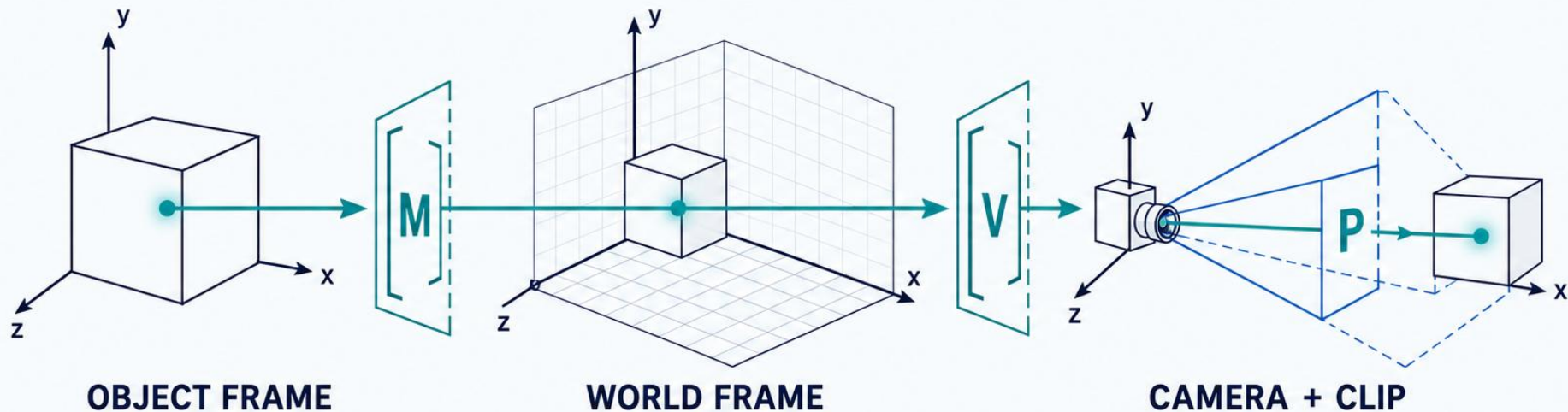
Every coordinate belongs to a frame



Homogeneous matrices compose transformations



Model, view, and projection answer different questions



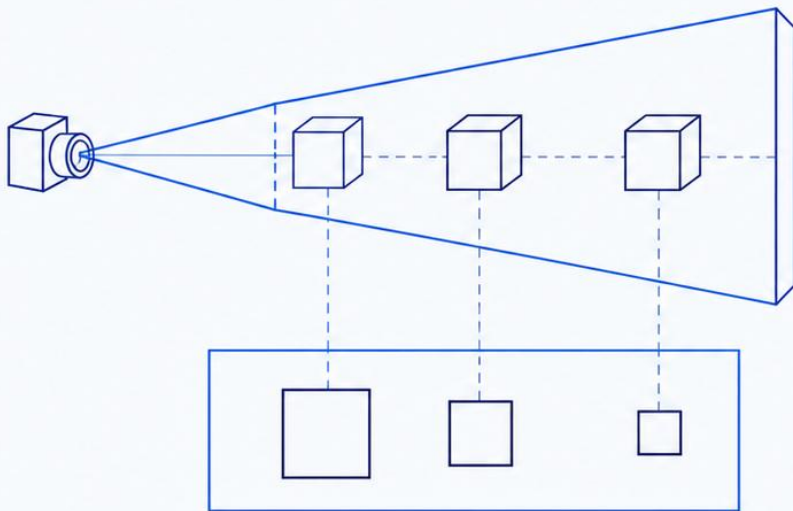
$$\mathbf{p}_{\text{clip}} = \mathbf{P} \mathbf{V} \mathbf{M} \mathbf{p}_{\text{model}}$$



Rightmost transform acts first

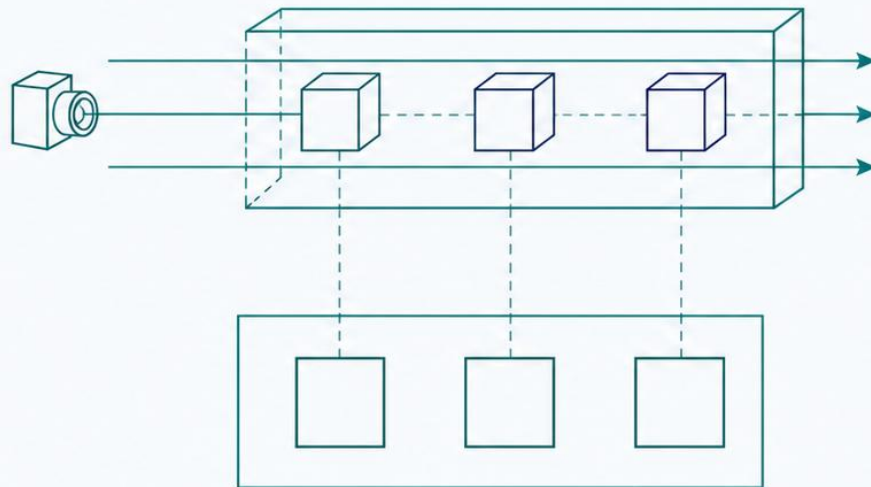
Perspective is a projection choice

Perspective: projected size changes with distance



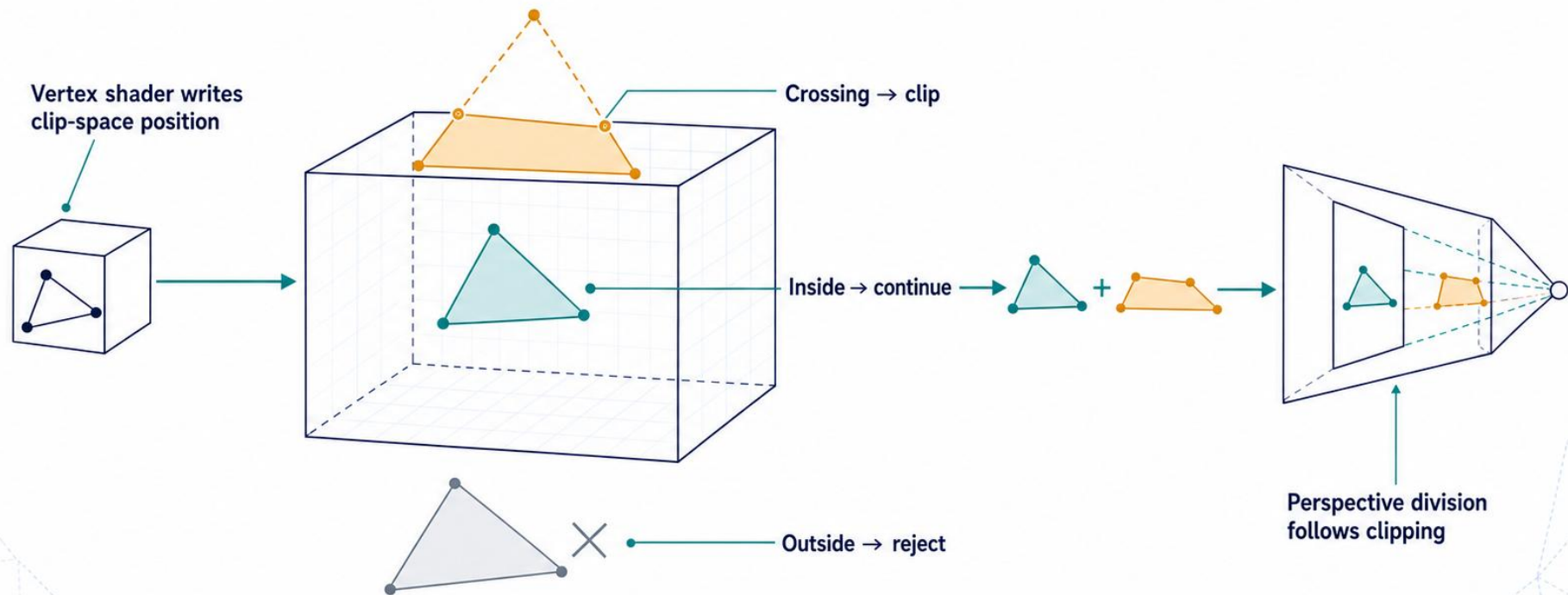
Field of view • aspect • near • far

Orthographic: projected size is distance-independent

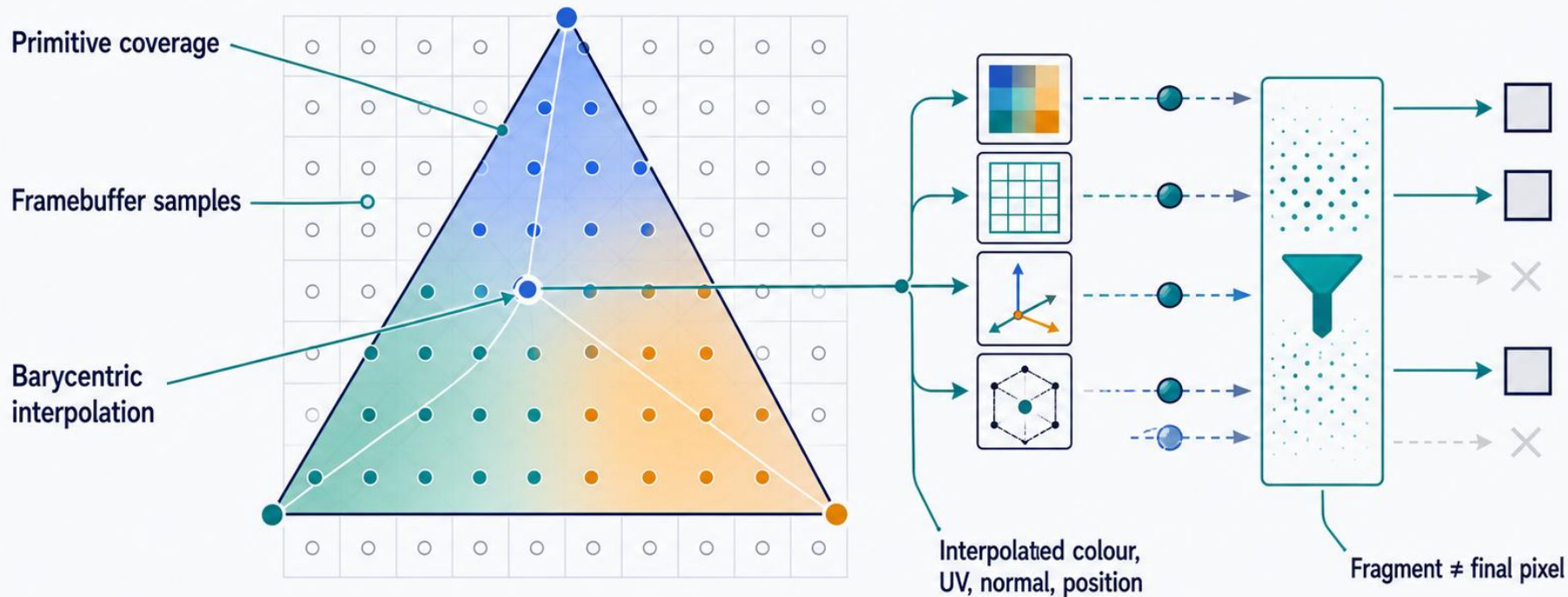


Both render 3D geometry

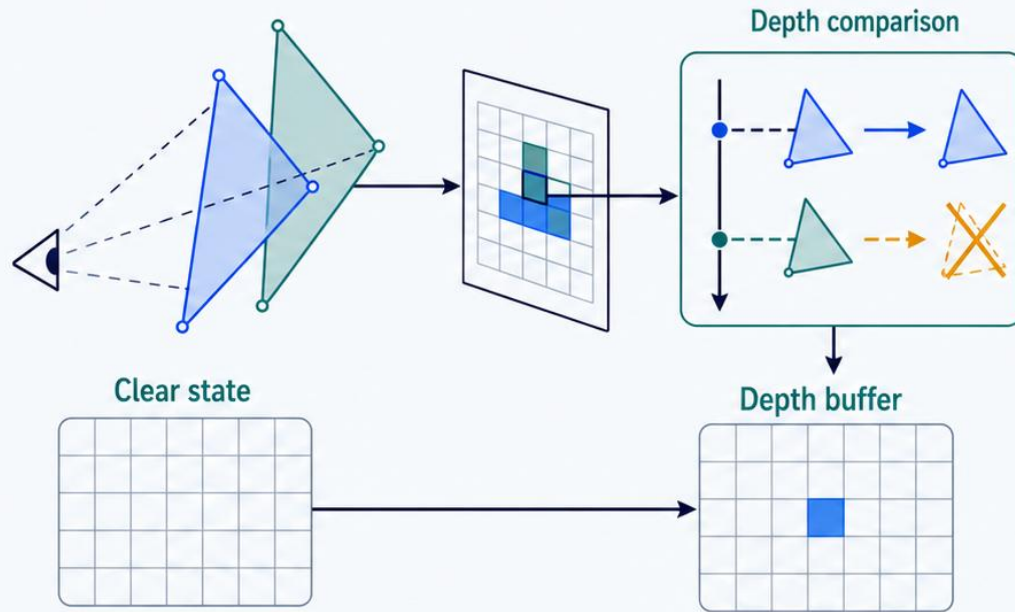
Clipping decides what can reach rasterisation



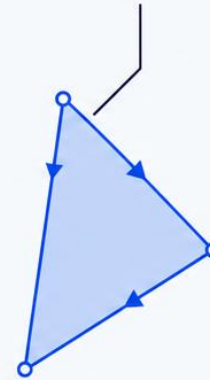
Rasterisation creates candidate fragments



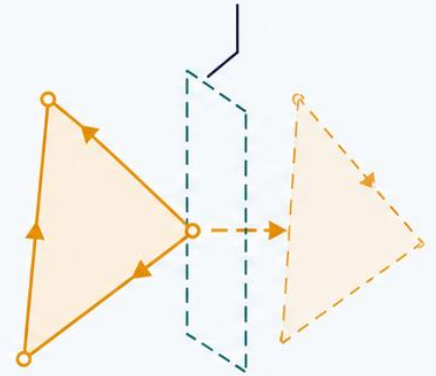
Depth and culling approximate visibility efficiently



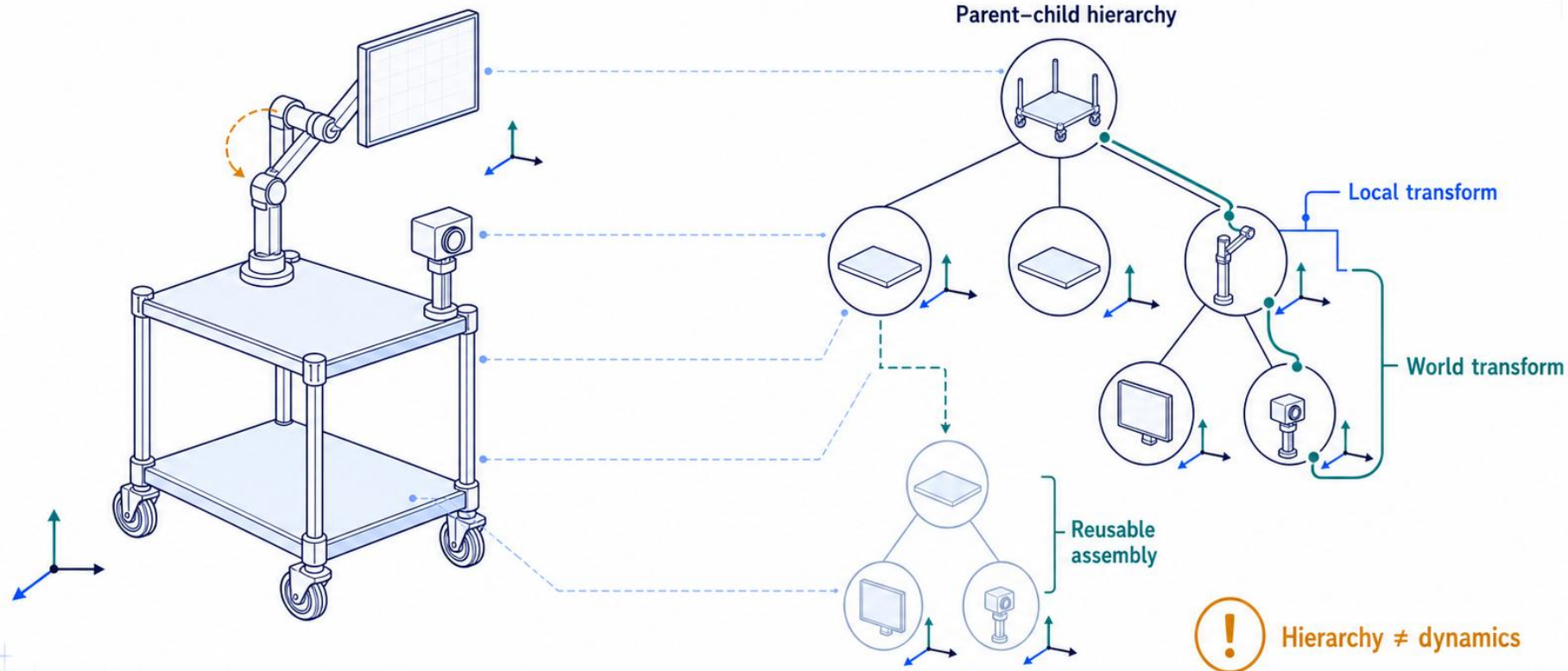
Front-face winding



Back-face culling

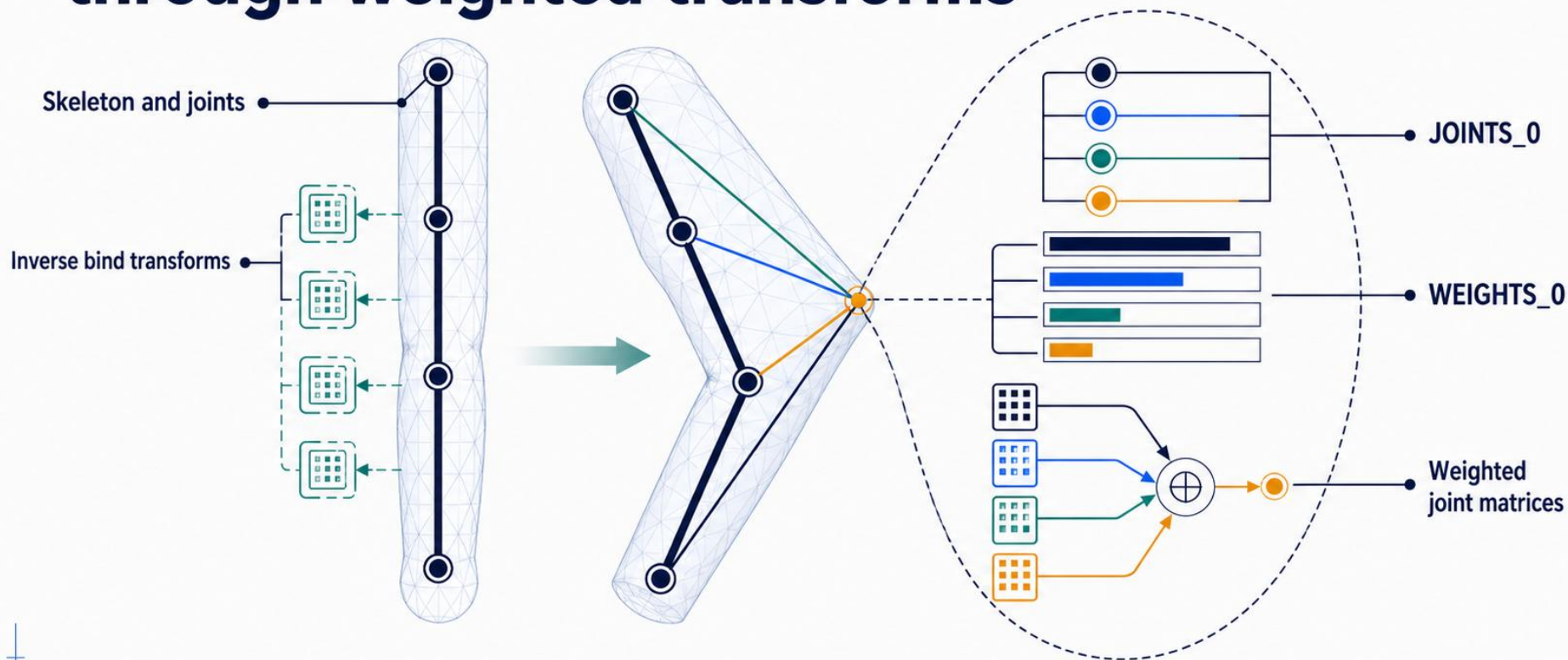


Scene graphs make transforms relative

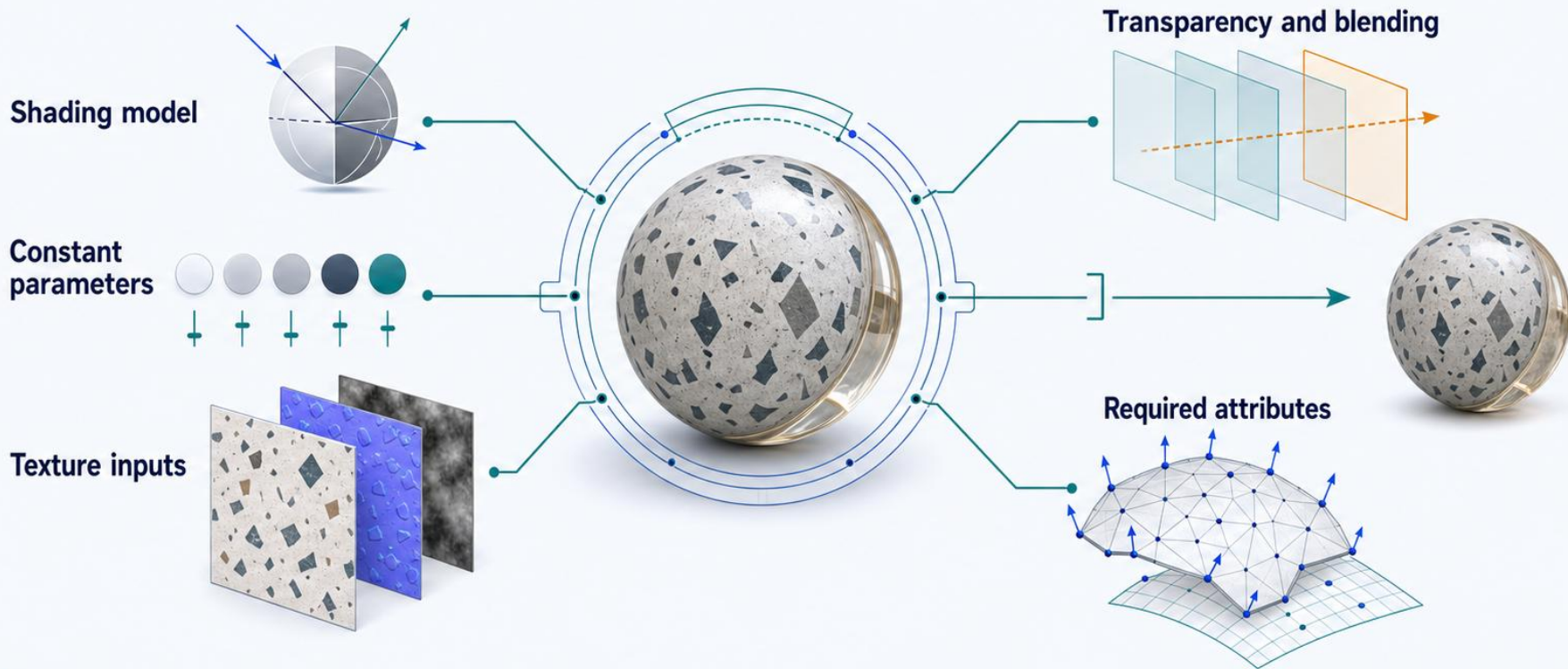


! Hierarchy $\neq$ dynamics

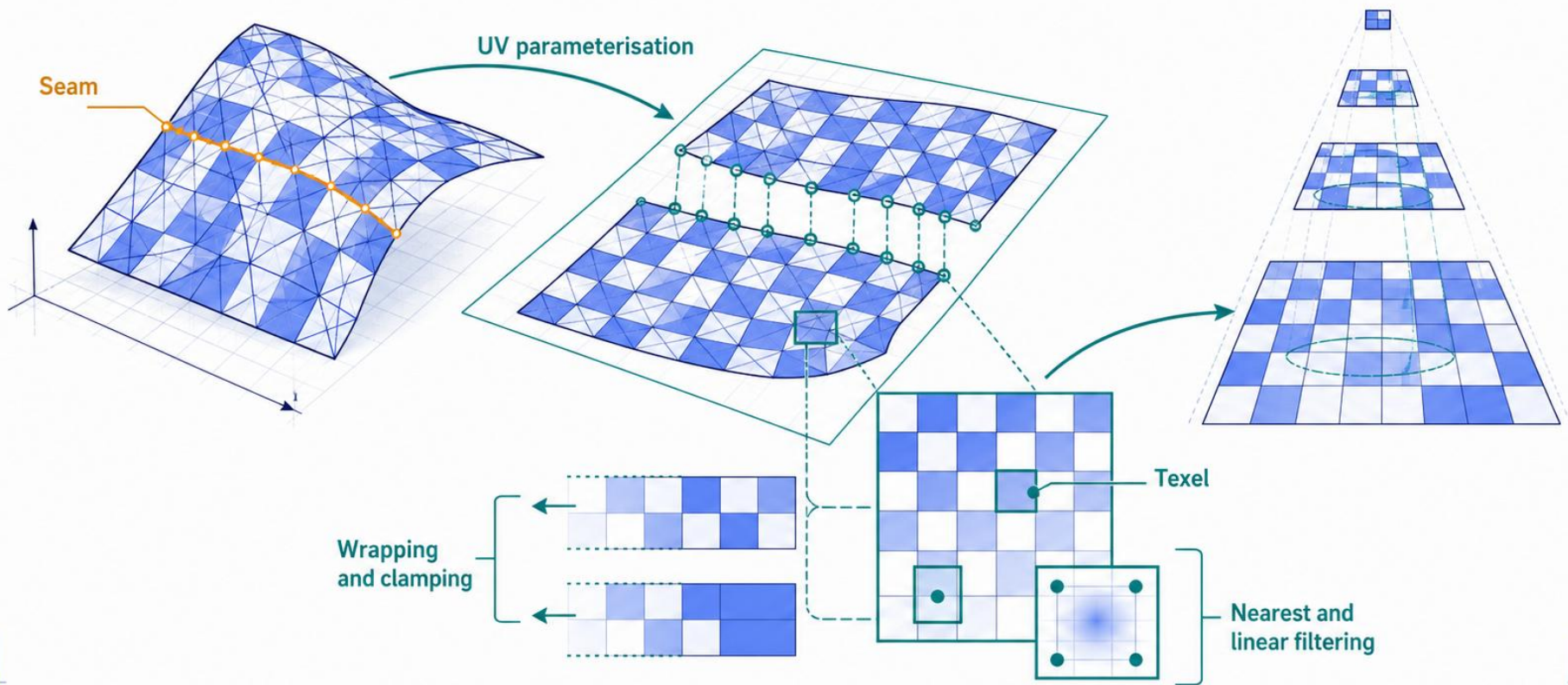
Bones deform a skin through weighted transforms



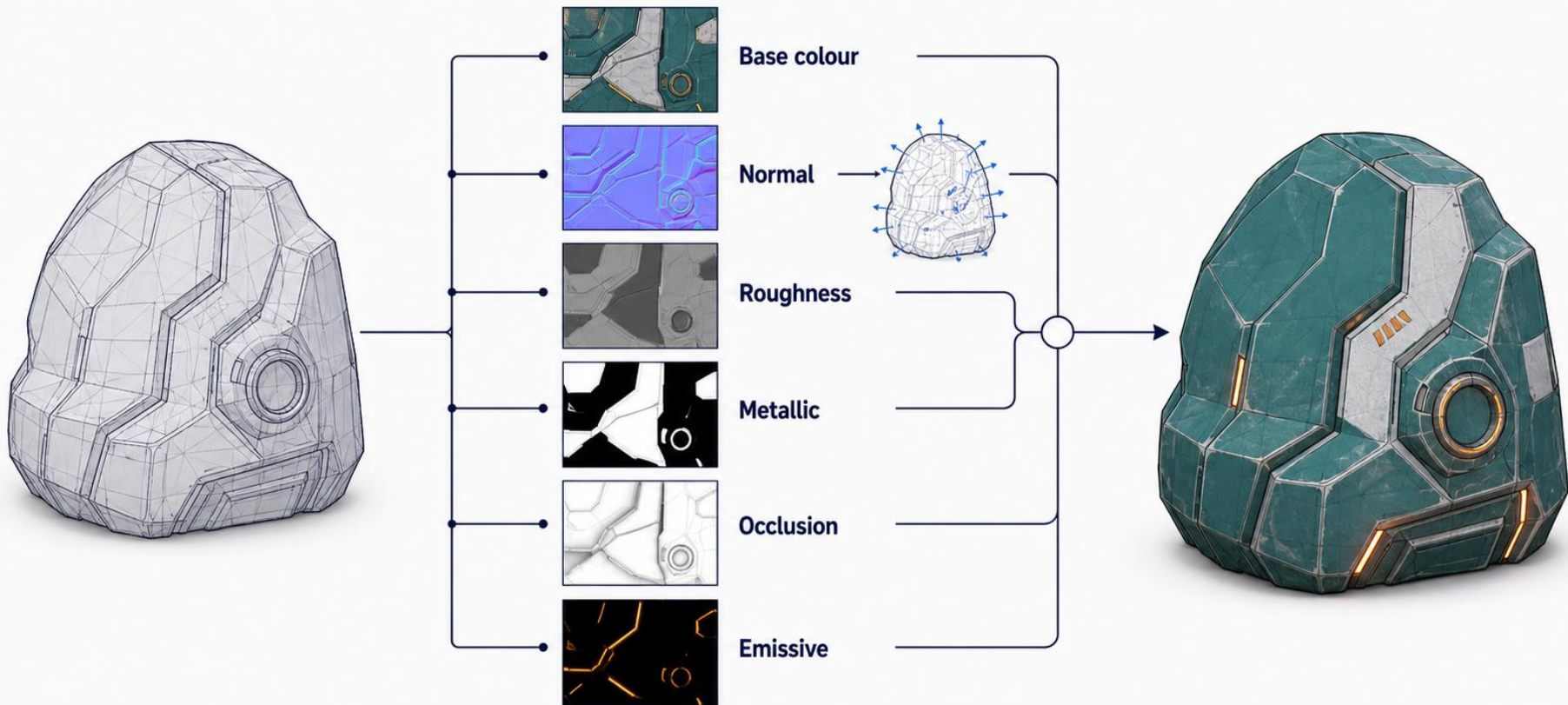
A material is a model plus parameters and state



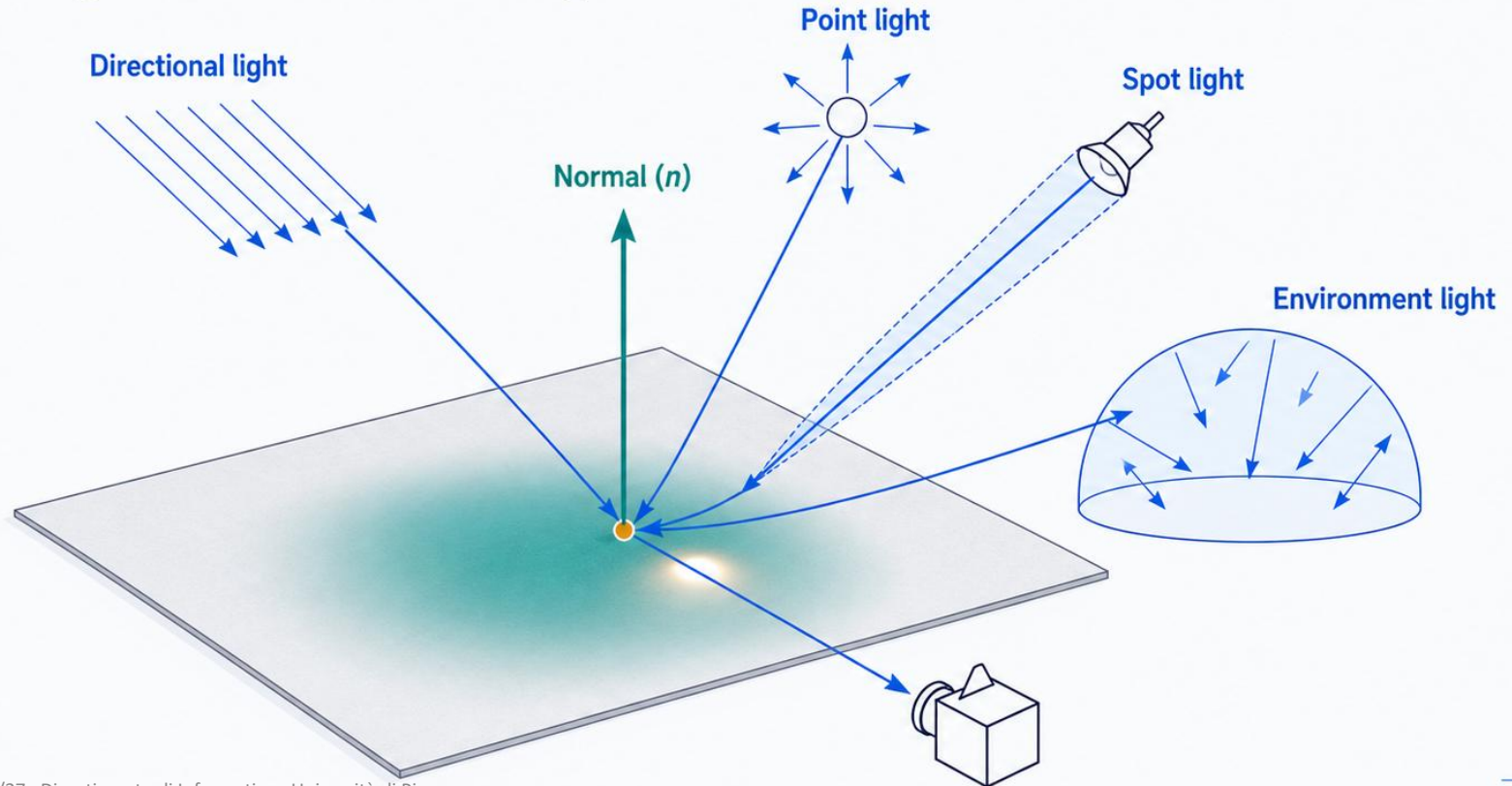
Textures are sampled data



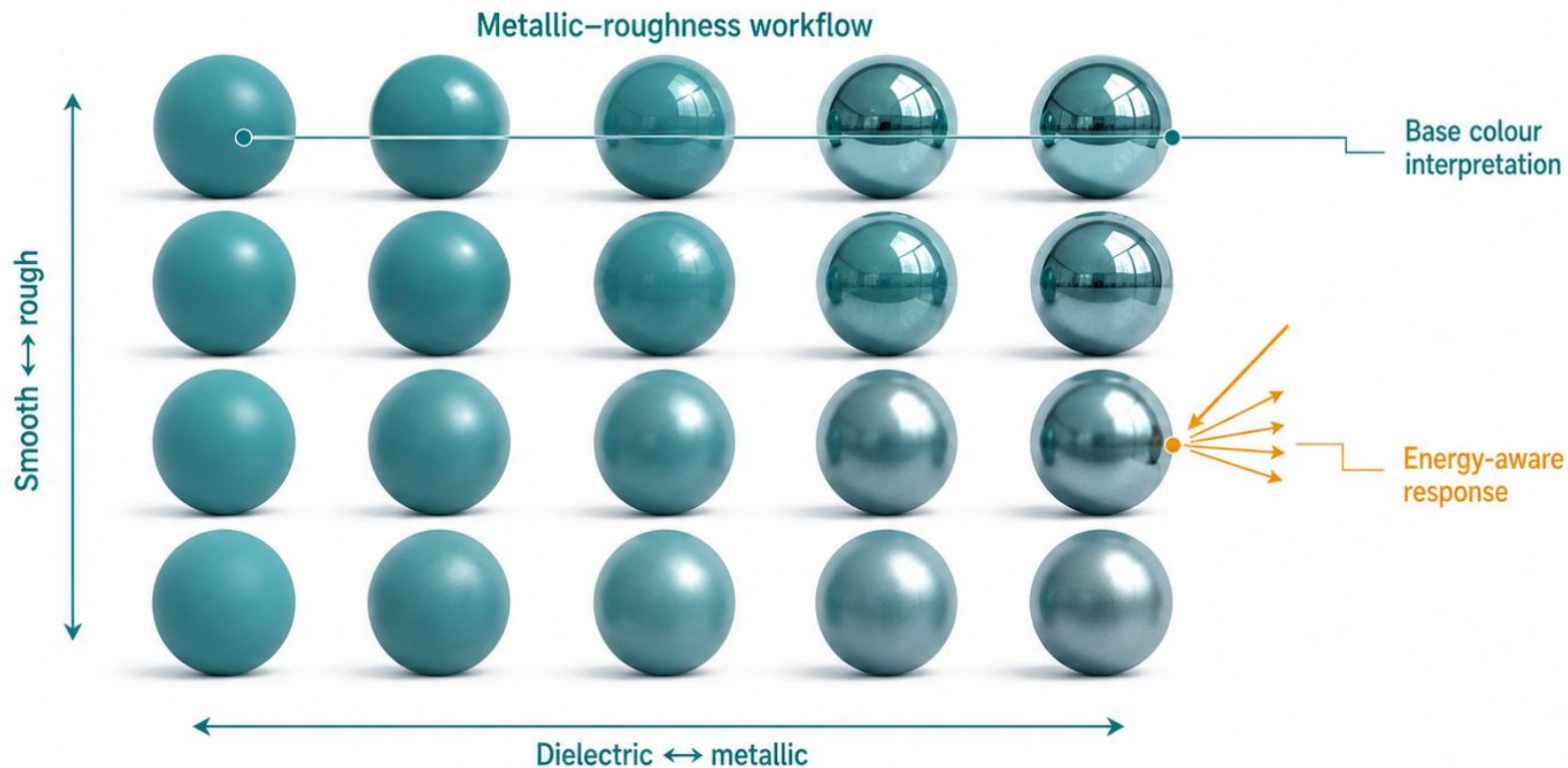
Texture maps have different roles



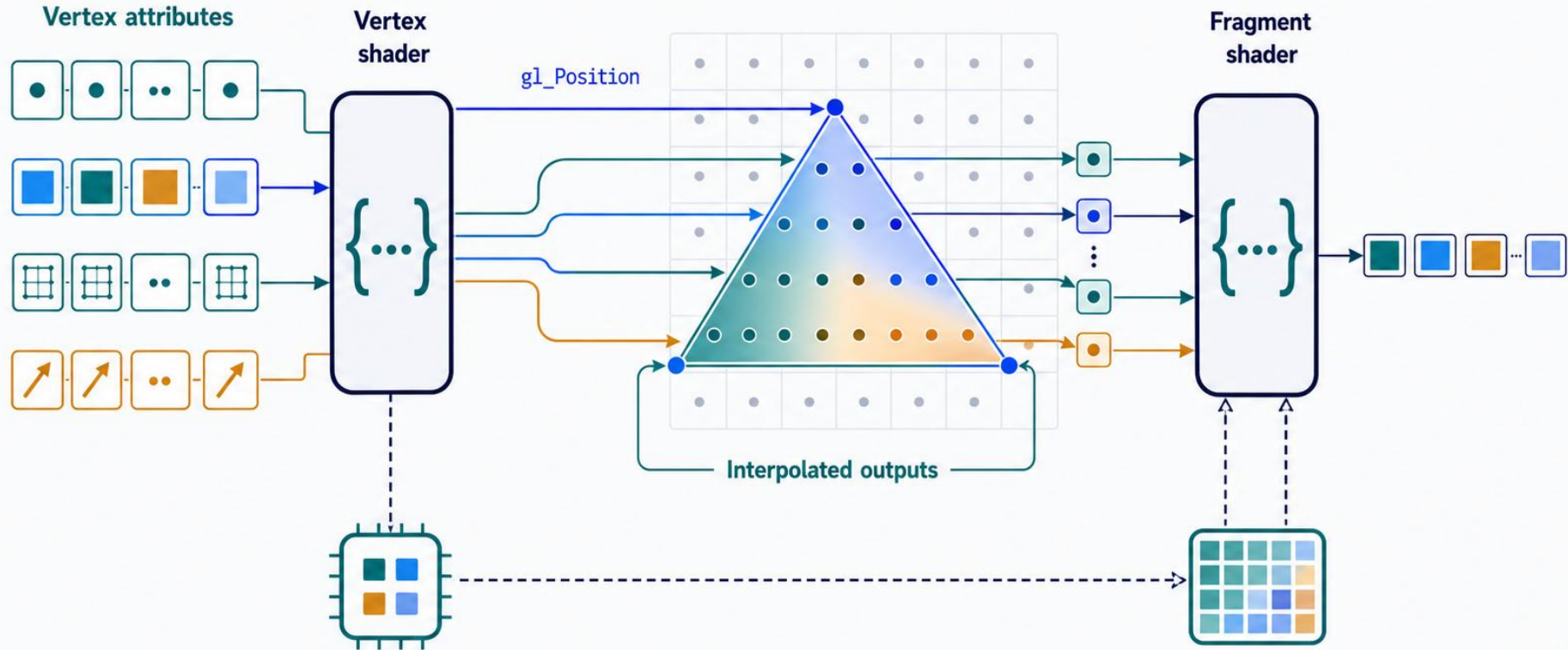
Lighting describes sources; shading evaluates response



Physically based shading is a disciplined approximation



Shaders make the pipeline programmable

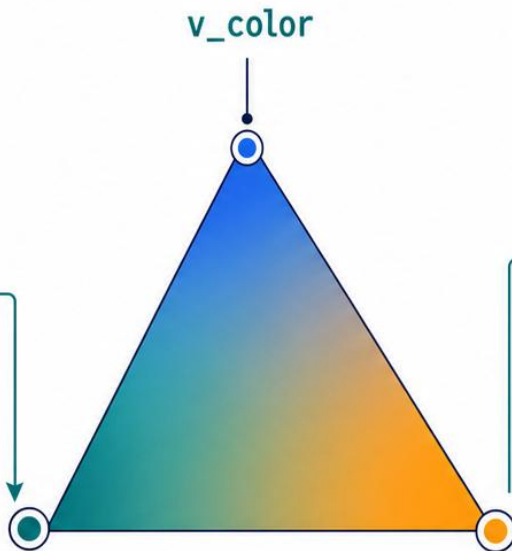


Minimal GLSL connects attributes to fragments

Vertex shader:

```
#version 300 es
in vec2 a_position;
in vec3 a_color;
out vec3 v_color;

void main() {
    gl_Position = vec4(a_position, 0.0, 1.0);
    v_color = a_color;
}
```

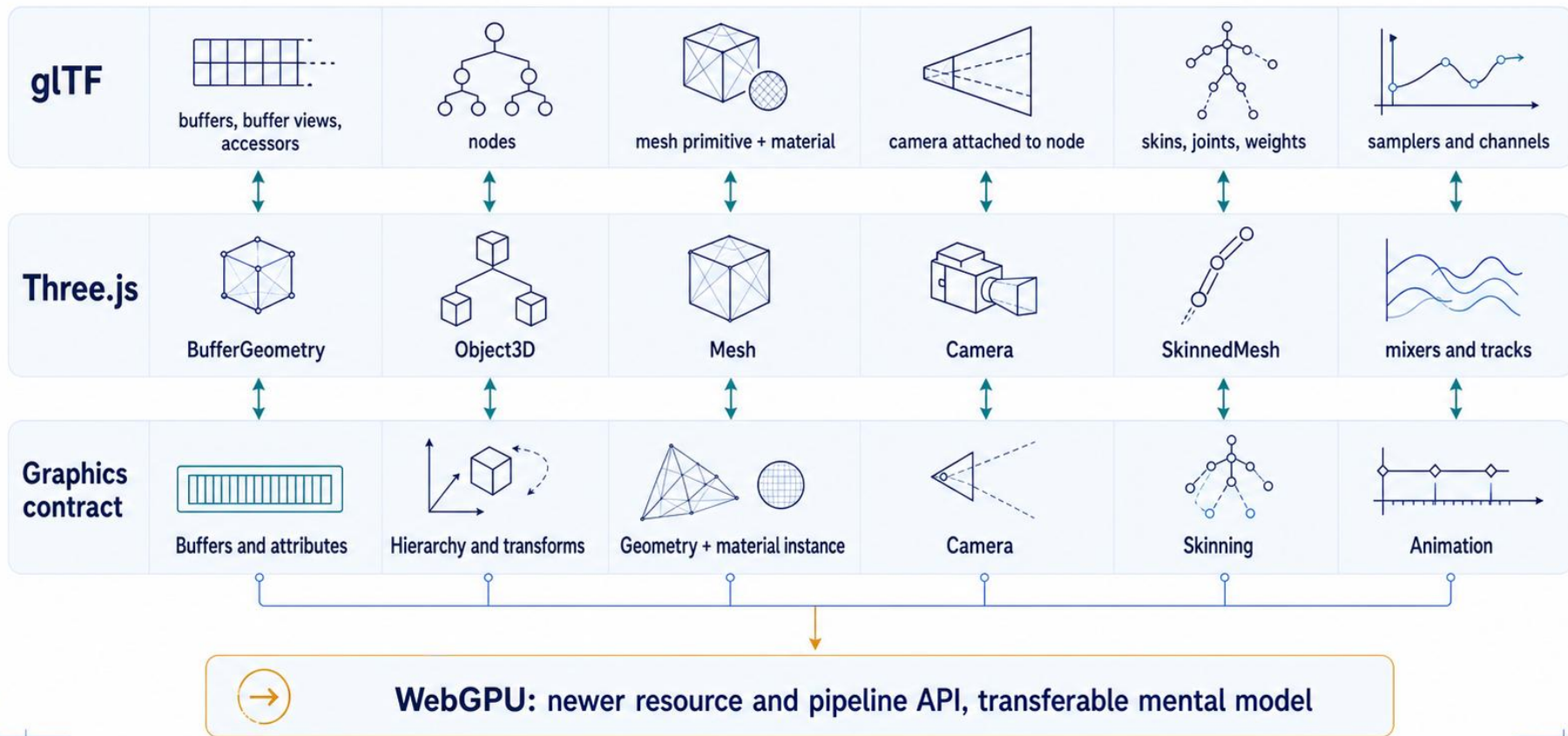


Fragment shader:

```
#version 300 es
precision highp float;
in vec3 v_color;
out vec4 outColor;

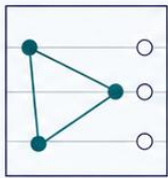
void main() {
    outColor = vec4(v_color, 1.0);
}
```

Three.js and glTF package the same contracts



Activity: diagnose the first broken contract

1. Triangle is in the buffer but invisible



2. Texture stretches and breaks at a seam



?

For each:

Likely stage or contract



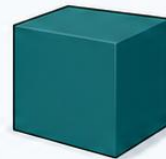
Evidence to inspect



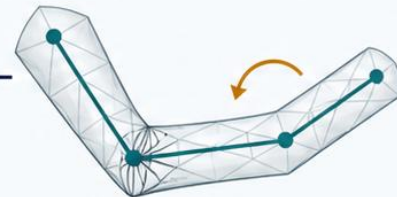
One experiment that could falsify the hypothesis



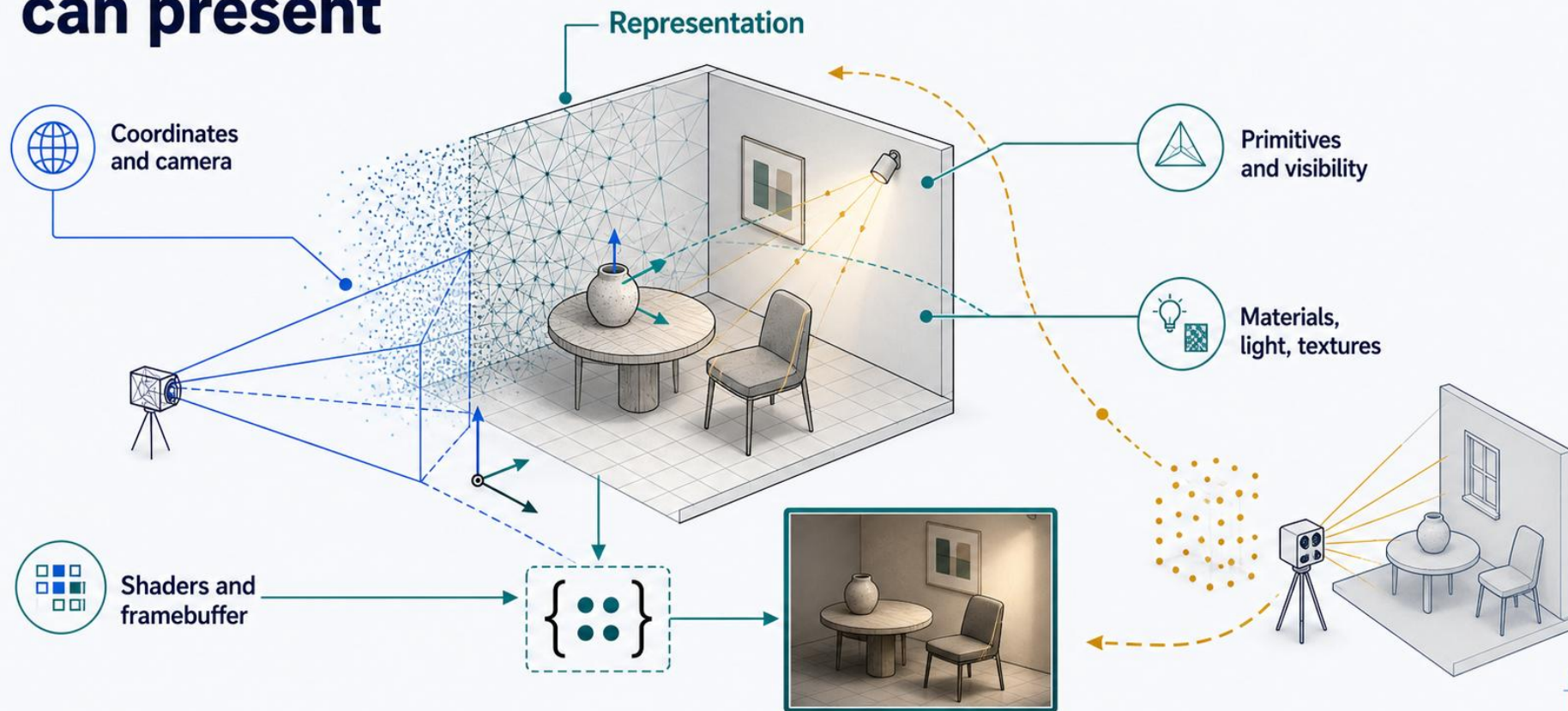
3. Lit material is black; unlit material is visible



4. Elbow collapses when a joint rotates



Graphics defines what a world can present



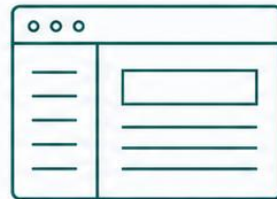
References and further reading



**Khronos Group —
WebGL and WebGL 2.0
Specification**



**Khronos Group —
glTF 2.0
Specification**



**Three.js —
official manual and
documentation**



**W3C GPU for the Web
Working Group —
WebGPU**



Akenine-Möller, Haines, Hoffman — *Real-Time Rendering*