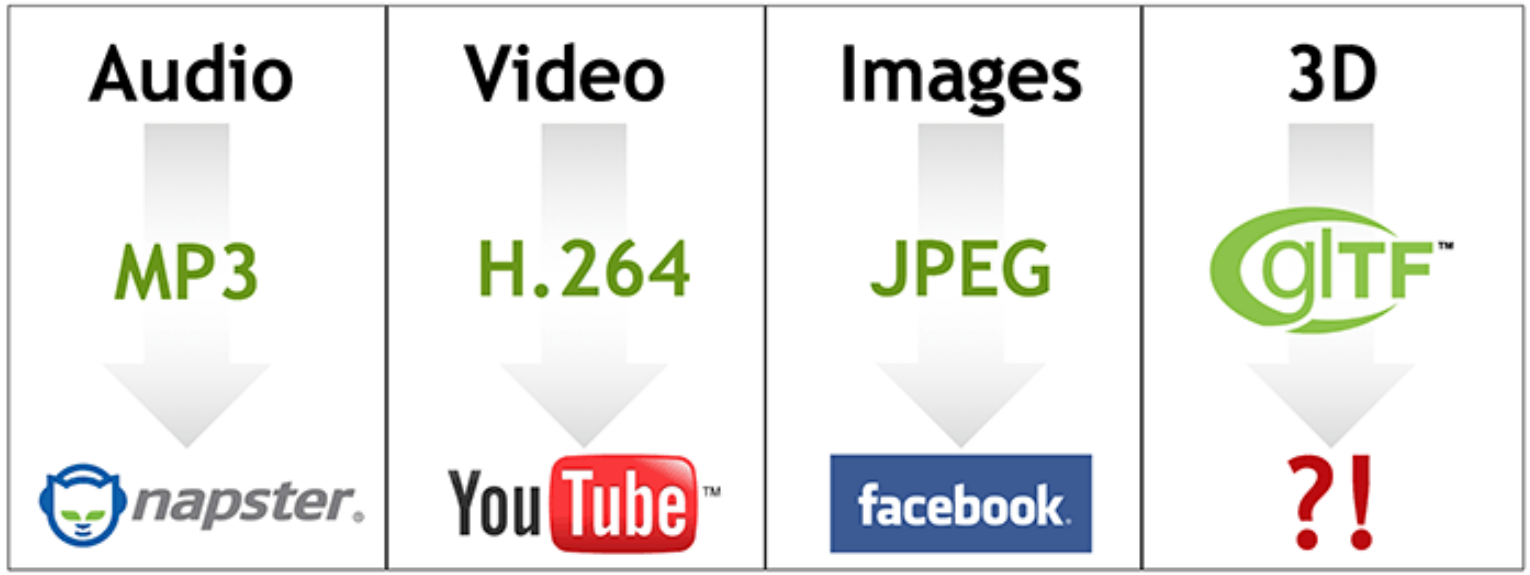
Marco Hutter
gltf@marco-hutter.de

Khronos glTF Webinar

February 2017

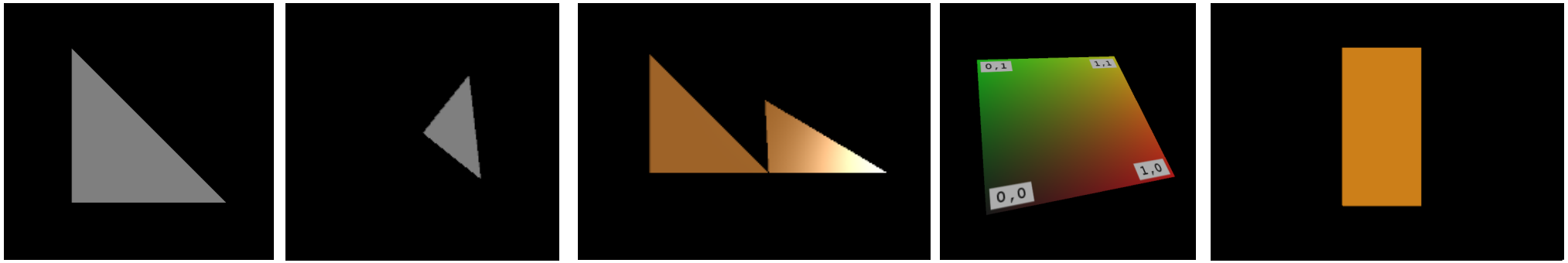
What is glTF?

- **glTF is the *GL Transmission Format***
  - An open standard, developed by Khronos: khronos.org/gltf
- **Designed for the efficient transfer of 3D assets**
  - Versatile, compact, and easy to process by the client

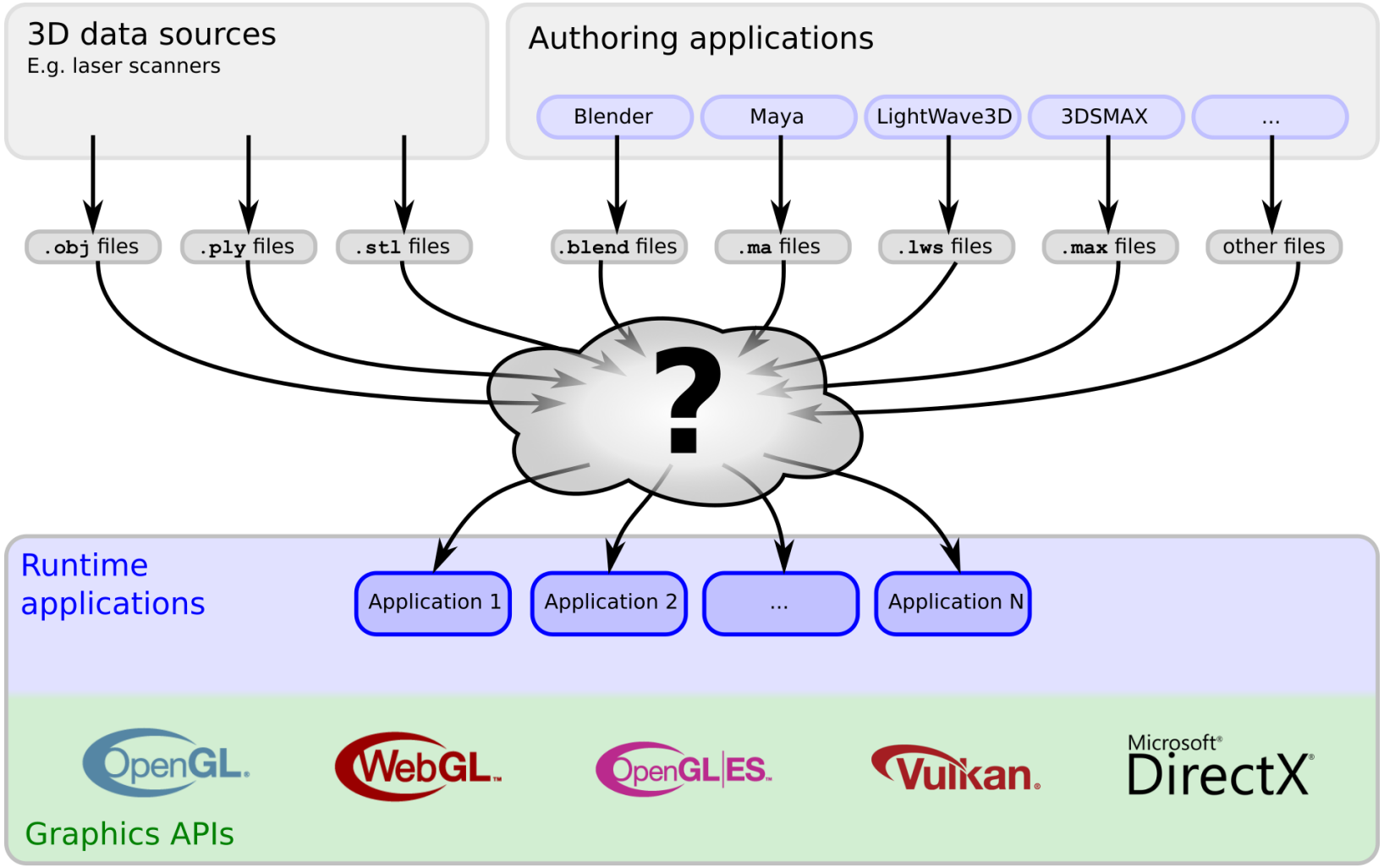


Outline

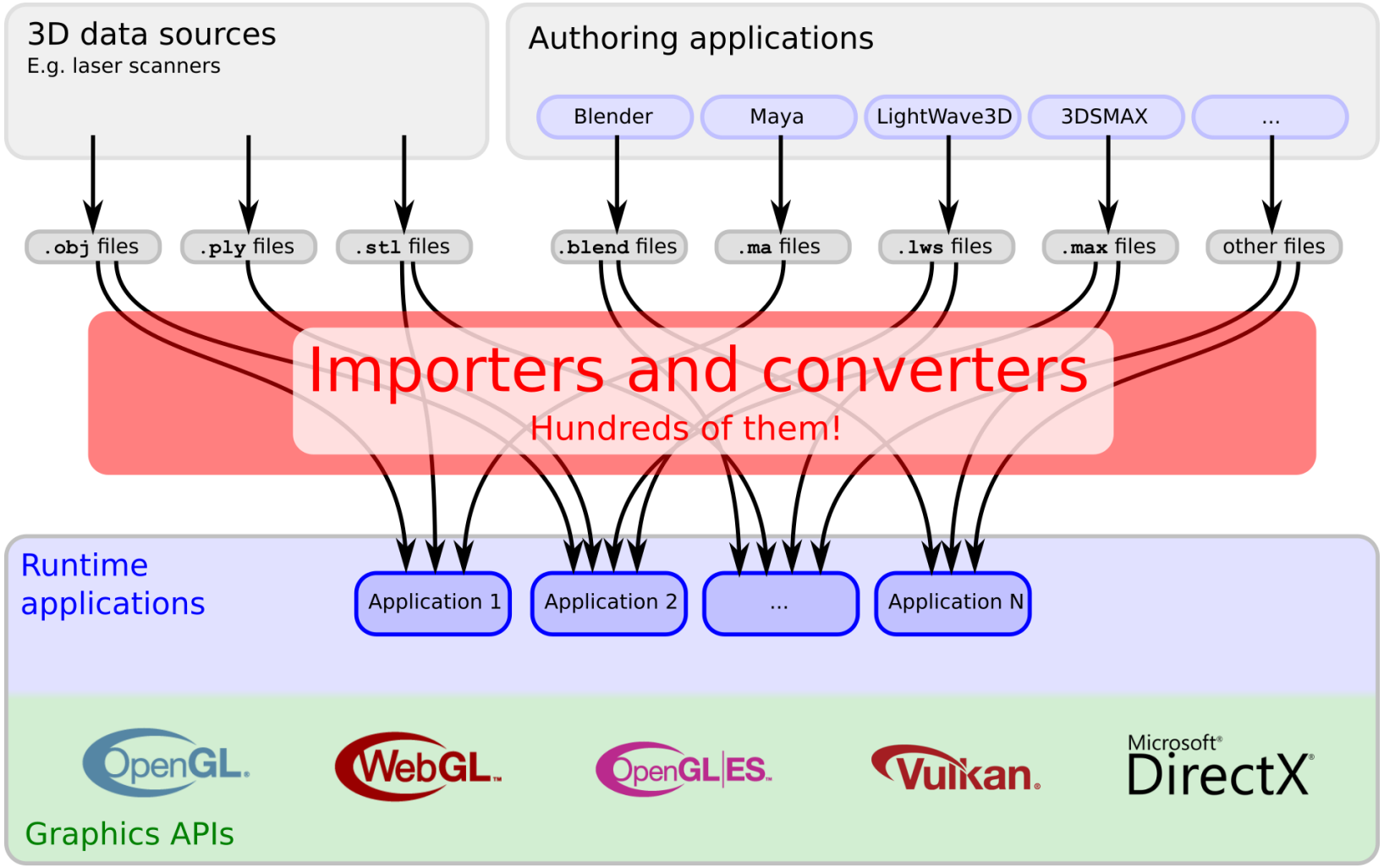
- **The full tutorial is available at github.com/KhronosGroup/glTF-Tutorials**
- **Explains the concepts of glTF, step by step**
  - Each one demonstrated with an actual glTF asset
- **Geometry, animations, materials, textures, skins…**
- **Targeting glTF 2.0: Pull request at github.com/KhronosGroup/glTF/pull/826**



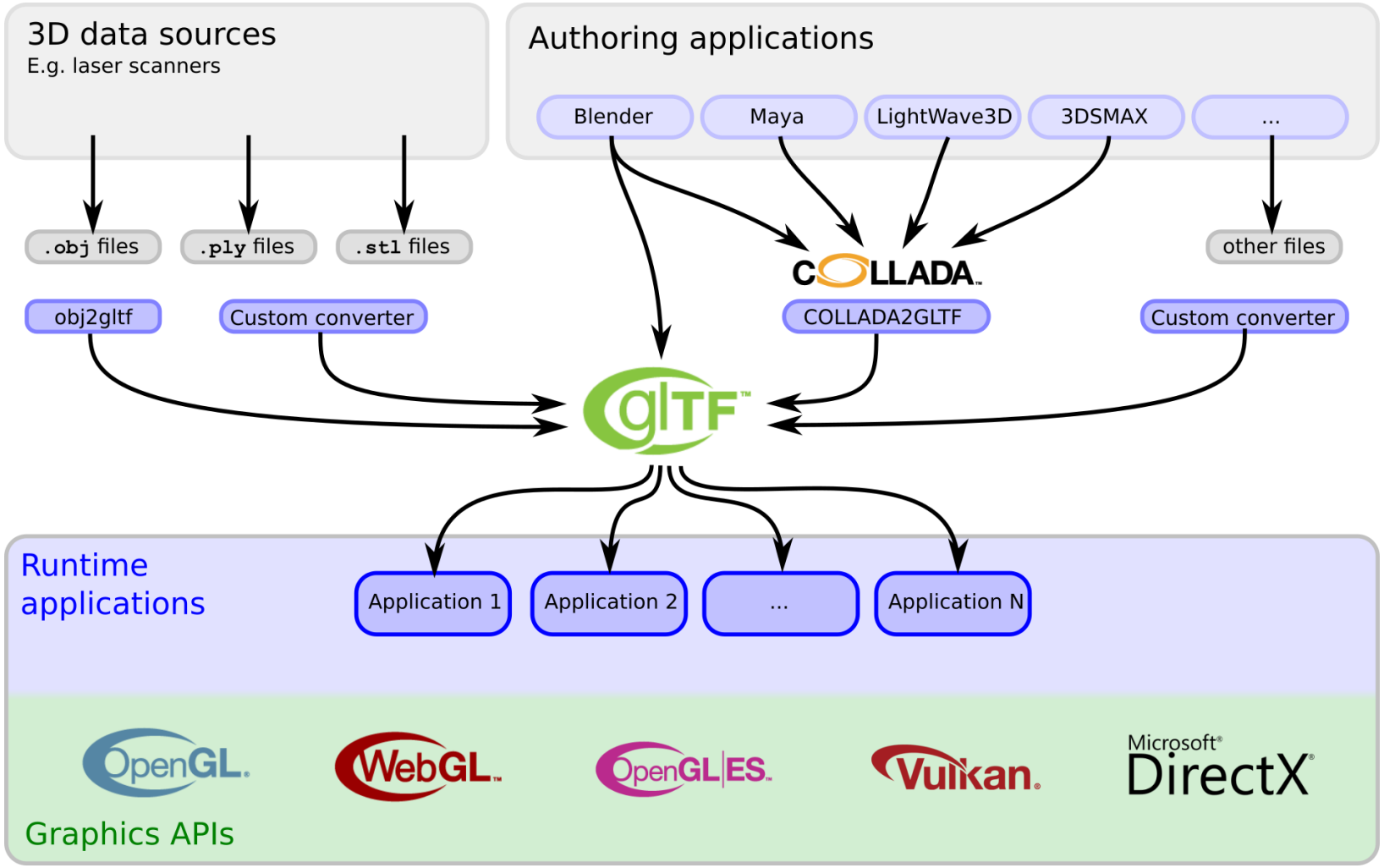
Introduction and motivation



Introduction and motivation

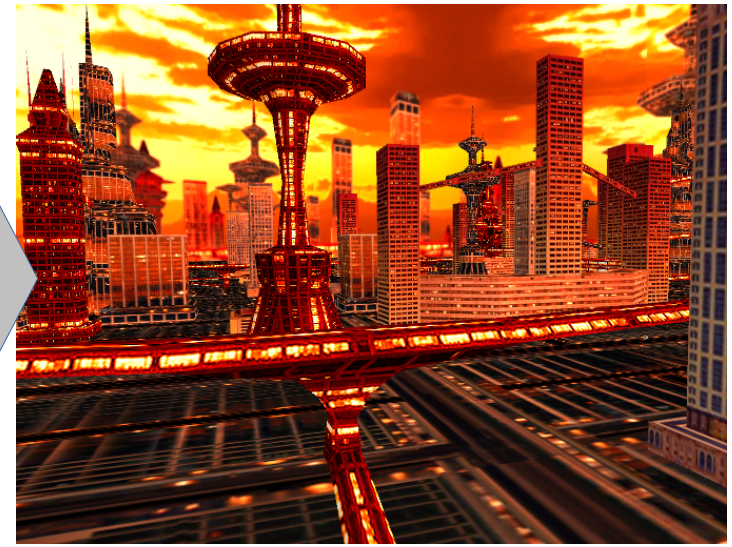


Introduction and motivation



Goals of glTF

- **Versatile:**



- **Plain geometry**
- **Complex scenes with animations, materials, ...**

- **Compact:**

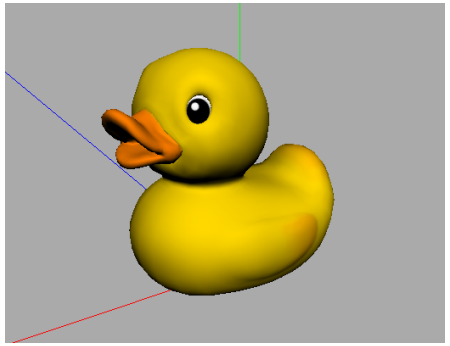
OBJ : **90MB**
glTF: **19MB**

COLLADA: **5.3MB** (+1.9MB textures)
glTF: **2.2MB** (+1.9MB textures)

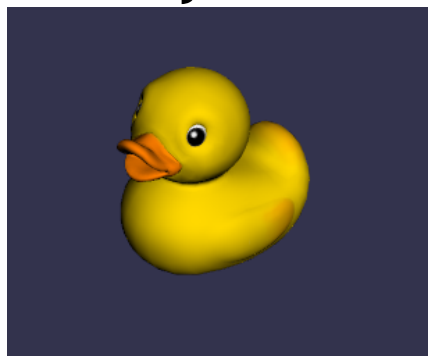
- **Easy to parse**
  - Core format based on JSON
  - Geometry data stored in binary form: No decoding overhead!

Users and supporters of glTF: Libraries

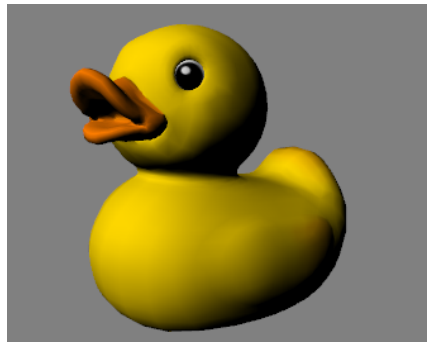
three.js



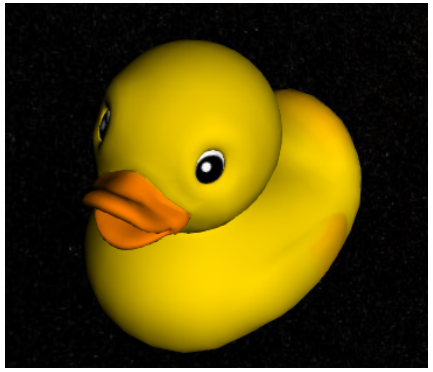
BabylonJS



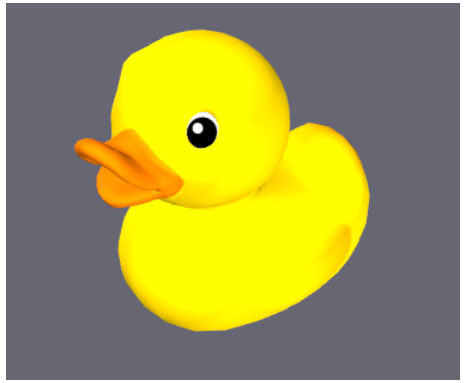
GrimoireGL



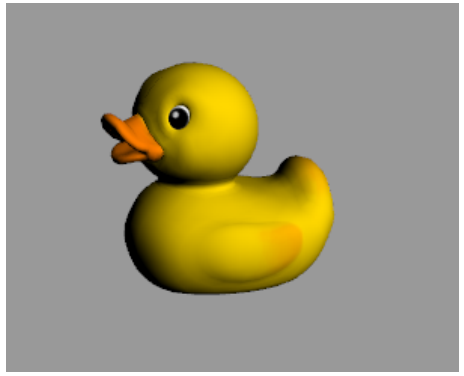
Cesium



xeogl



GLBoost

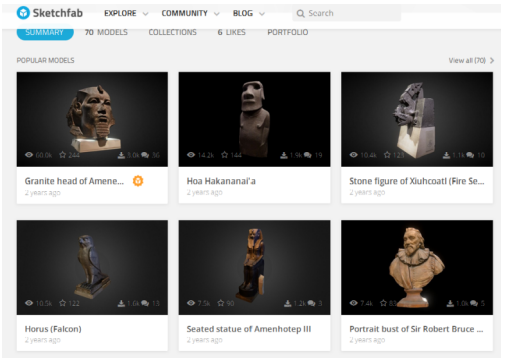


Test and comparison of loaders and viewers at github.com/cx20/gltf-test

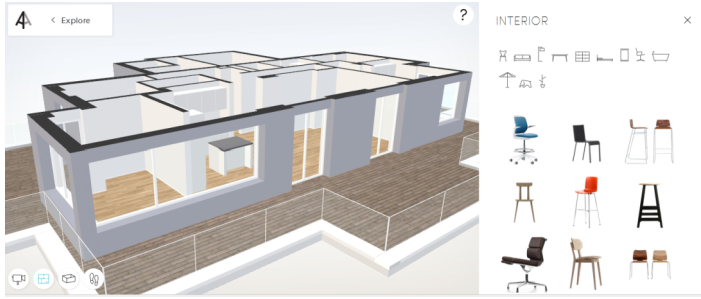
Users and supporters of glTF: Applications

VisCircle

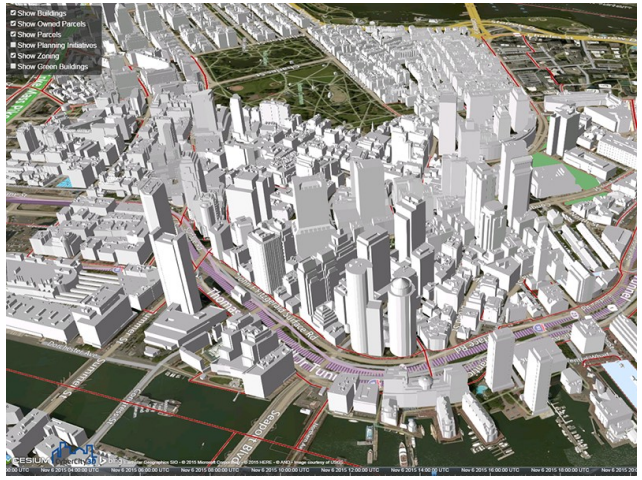
Sketchfab



Archilogic



Cesium

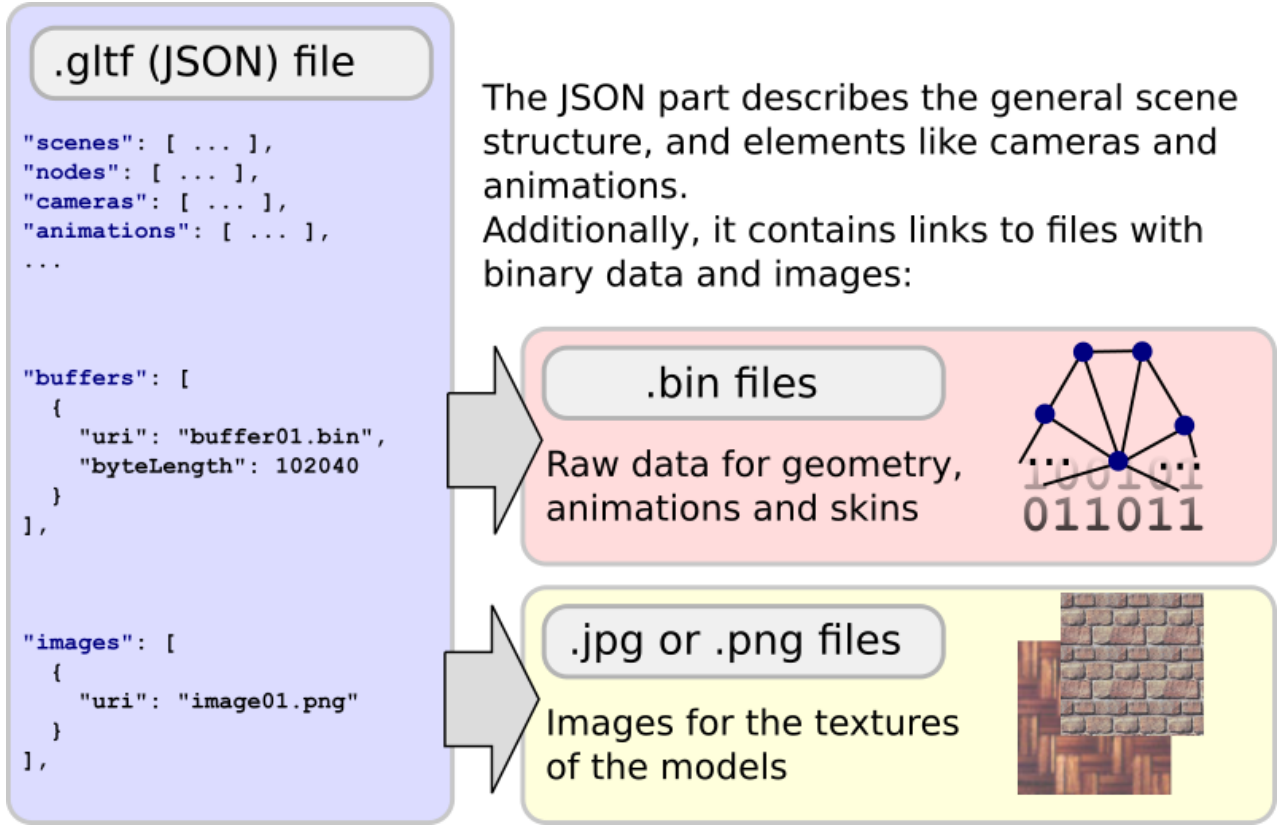


BioDigital

Supporters:

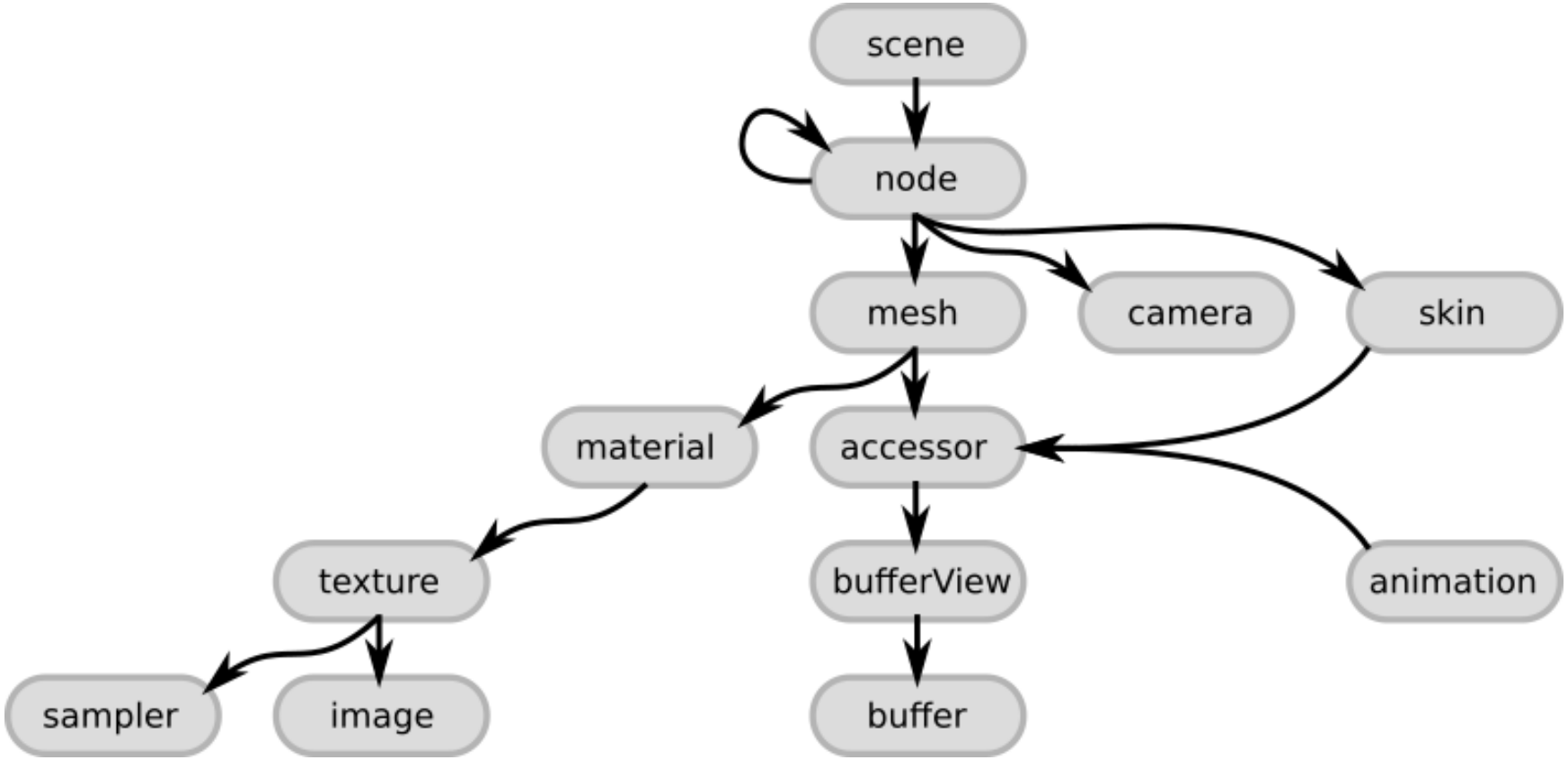
oculus
web3D CONSORTIUM
MPEG
Adobe
blender
OGC
box
Microsoft
Fraunhofer
otoy
NVIDIA
AREA Augmented Reality for Enterprise Alliance
CESIUM
AUTODESK
Google

Basic file structure



- External resources *can* be embedded into JSON, as data URIs

Basic JSON structure



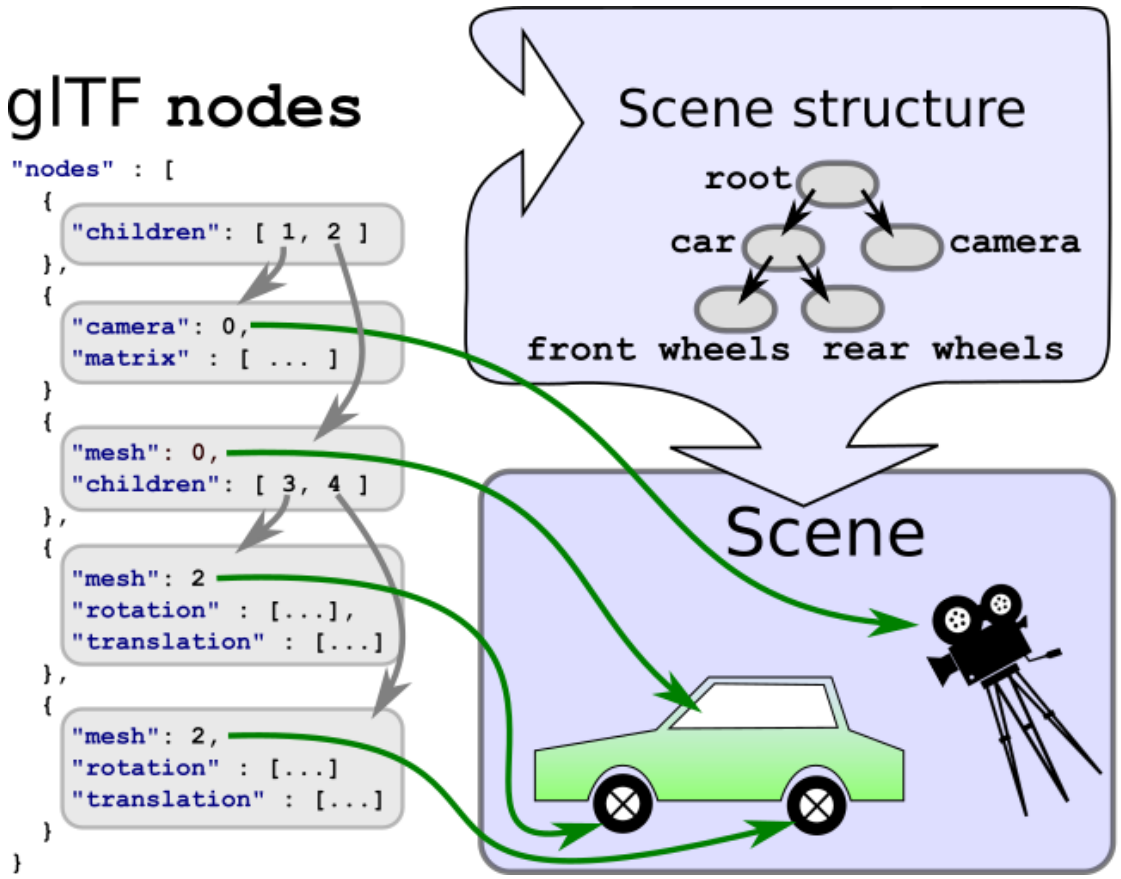
```
{
 "scenes" : [ { "nodes"  : [ 0 ] } ],
 "nodes"  : [ { "mesh" : 0 } ],
 "meshes" : [ {
   "primitives" : [ { "attributes" : { "POSITION" : 0 } } ]
  }
 ],
 "buffers" : [
  {
   "uri" : "data:application/octet-stream;base64,AAAAAAAAAAAAAAAAACAPwAAAAAAAAAAAAAAAAAAAgD8AAAA",
   "byteLength" : 36
  }
 ],
 "bufferViews" : [
  {
   "buffer" : 0,
   "byteOffset" : 0,
   "byteLength" : 36,
   "target" : 34962
  }
 ],
 "accessors" : [
  {
   "bufferView" : 0,
   "byteOffset" : 0,
   "componentType" : 5126,
   "count" : 3,
   "type" : "VEC3",
   "max" : [ 1.0, 1.0, 0.0 ],
   "min" : [ 0.0, 0.0, 0.0 ]
  }
 ],
 "asset" : { "version" : "2.0" }
}
```

This is a complete glTF asset
with an embedded buffer

(Supposed to be *the* minimal glTF asset)

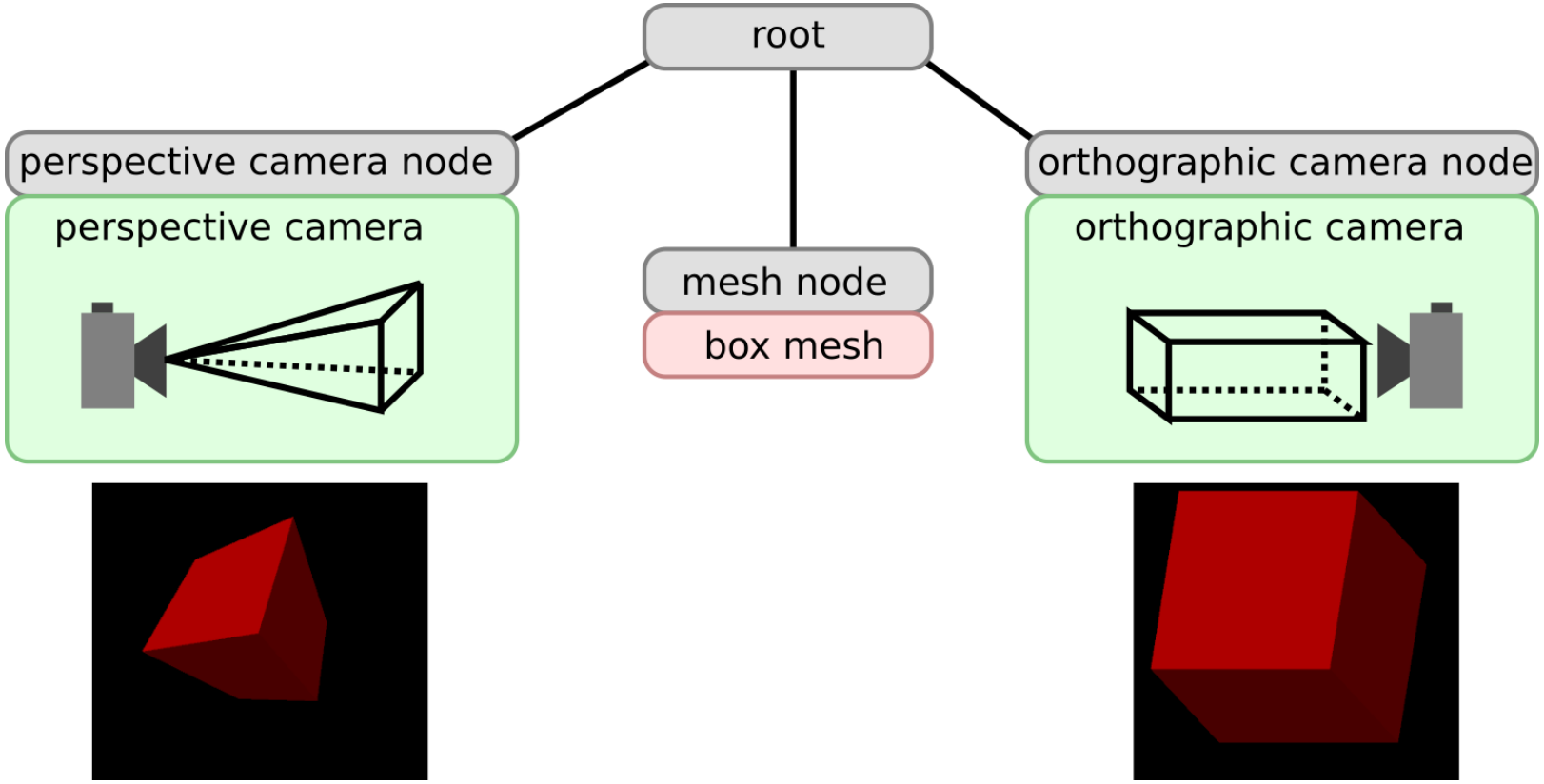
Scenes and nodes

- **Nodes stored in JSON**
  - Define a node hierarchy
  - Can have local transforms
- **Meshes, cameras, etc. are attached to nodes**



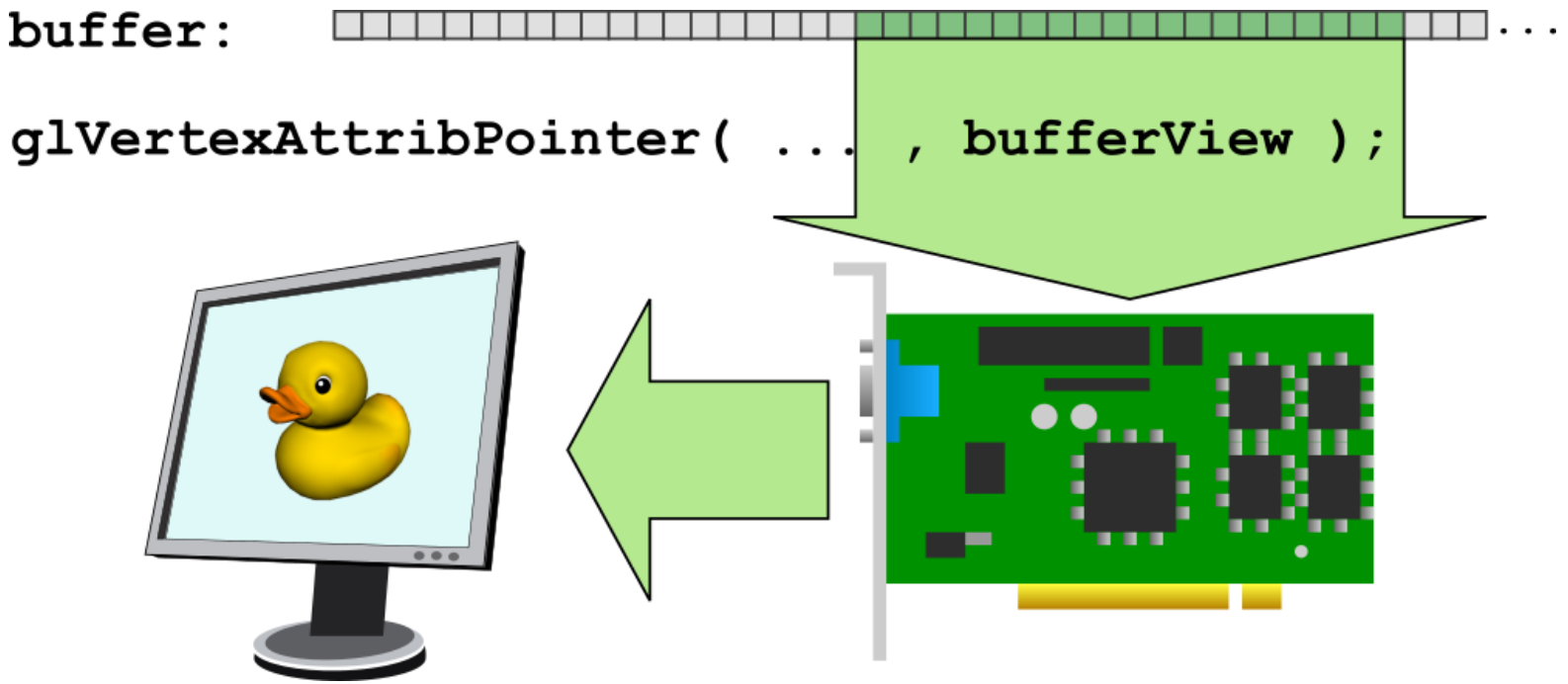
Cameras

- Perspective and orthographic cameras, attached to nodes

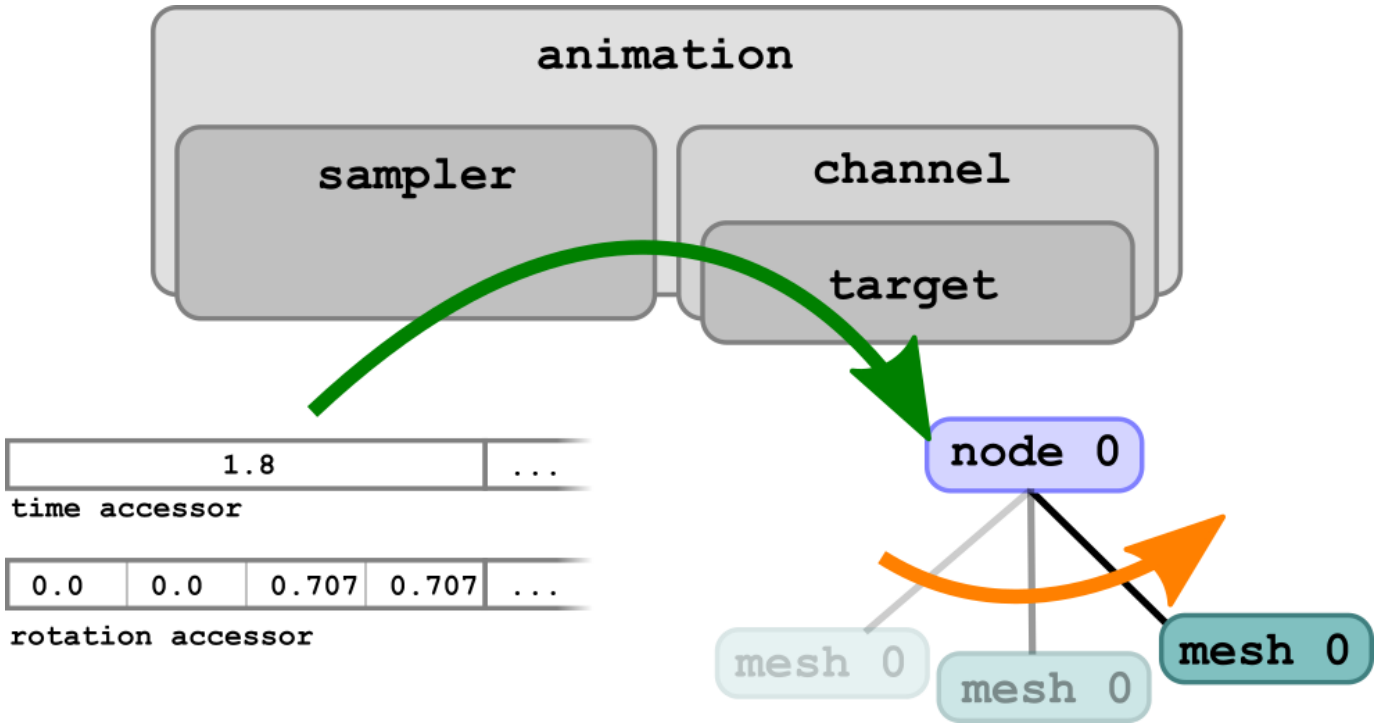


Buffers, bufferViews, and accessors

- A `buffer` is stored in an external file, in binary form
- A `bufferView` defines a part of a `buffer`
- An `accessor` defines the data layout of a `bufferView`



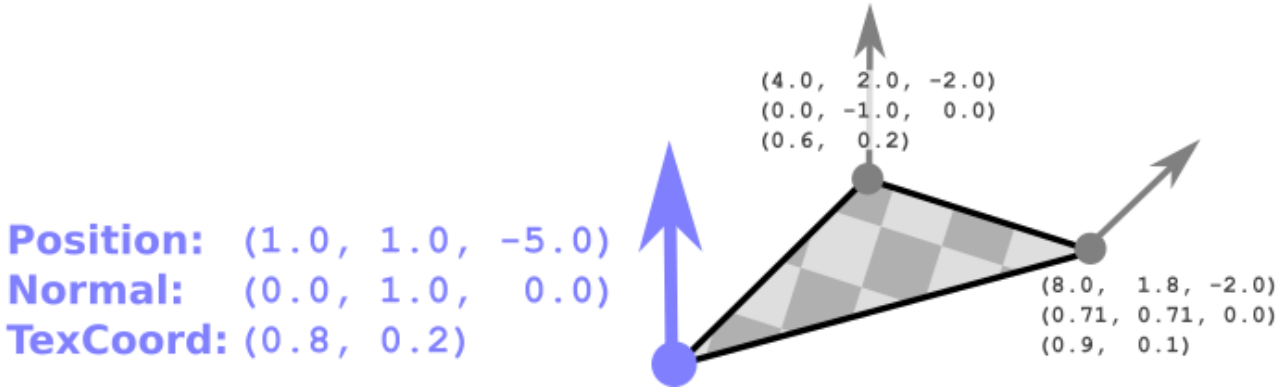
A simple animation



Meshes

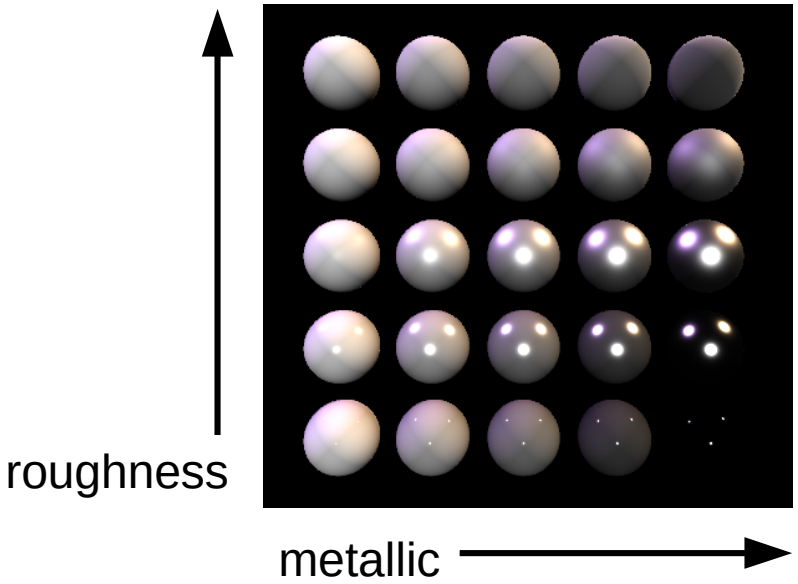
- Consist of mesh primitives that refer to accessors with vertex attribute data

positions accessor:	1.0	1.0	-5.0	4.0	3.0	-2.0	8.0	1.8	-2.0	...
normals accessor:	0.0	1.0	0.0	0.0	-1.0	0.0	0.71	0.71	0.0	...
texCoords accessor:	0.8	0.2	0.6	0.2	0.9	0.1	...			



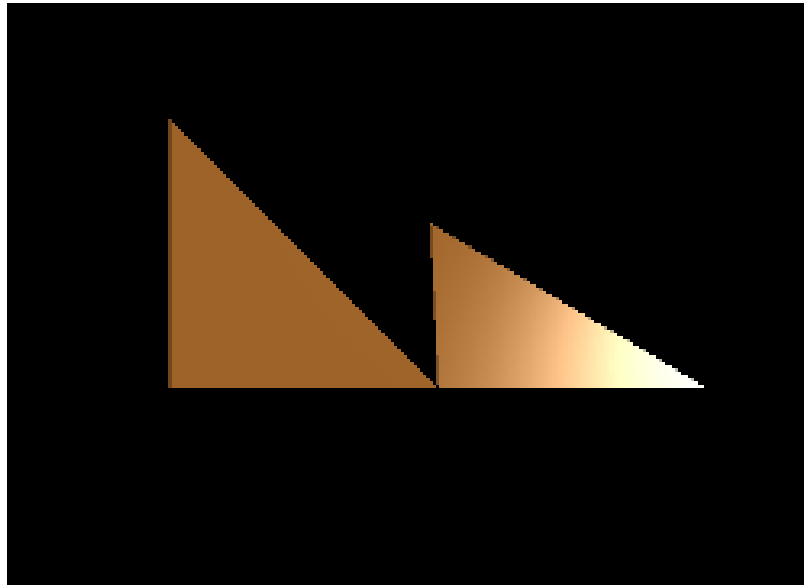
Materials

- A **`material`** stores material parameters
  - For example: Metallic-ness and roughness
  - Can also be given as textures
- Physically based rendering (PBR) part of glTF 2.0
  - Coordinated effort to define a standard for PBR!



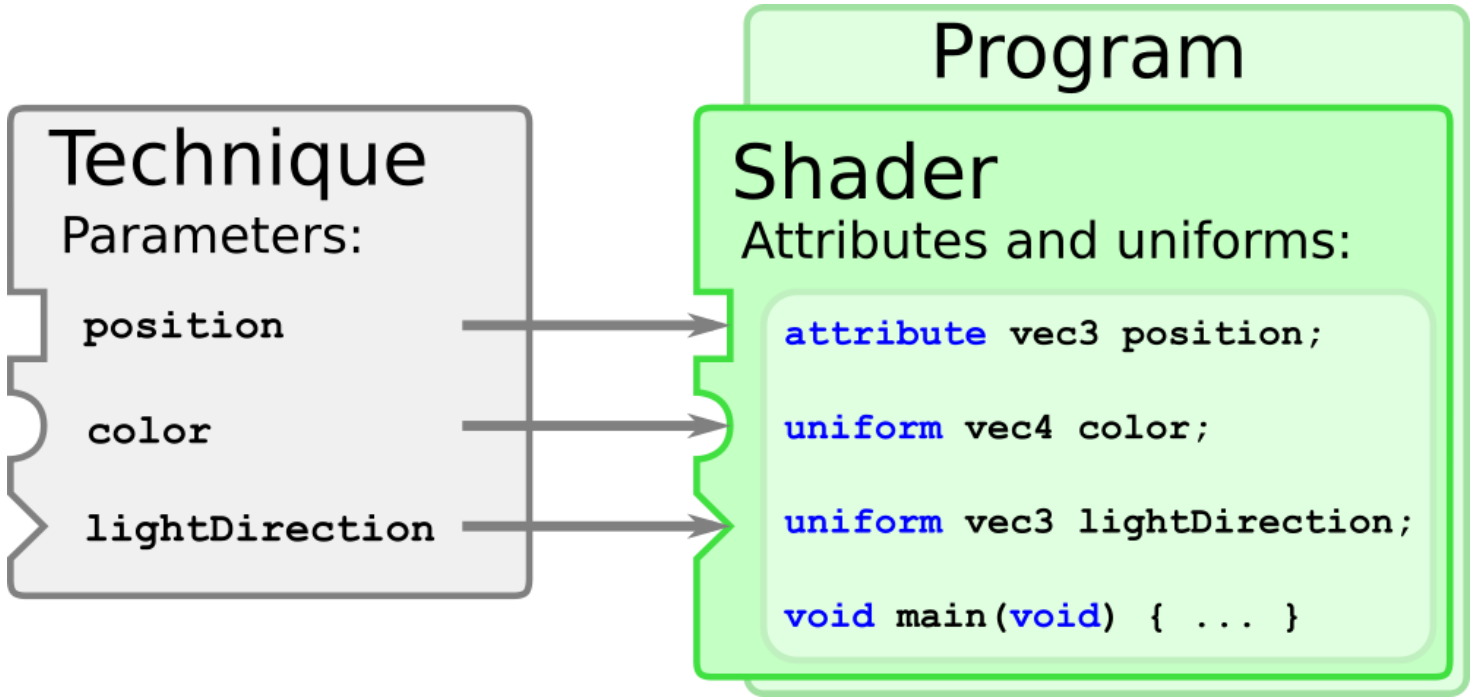
Techniques (extension)

- An extension for GL-based rendering
- Fast forward:
  - A **`material`** is an instance of a **`technique`**
  - A **`technique`** refers to a **`program`**
  - The **`program`** refers to GLSL **`shader`** objects



Techniques (extension)

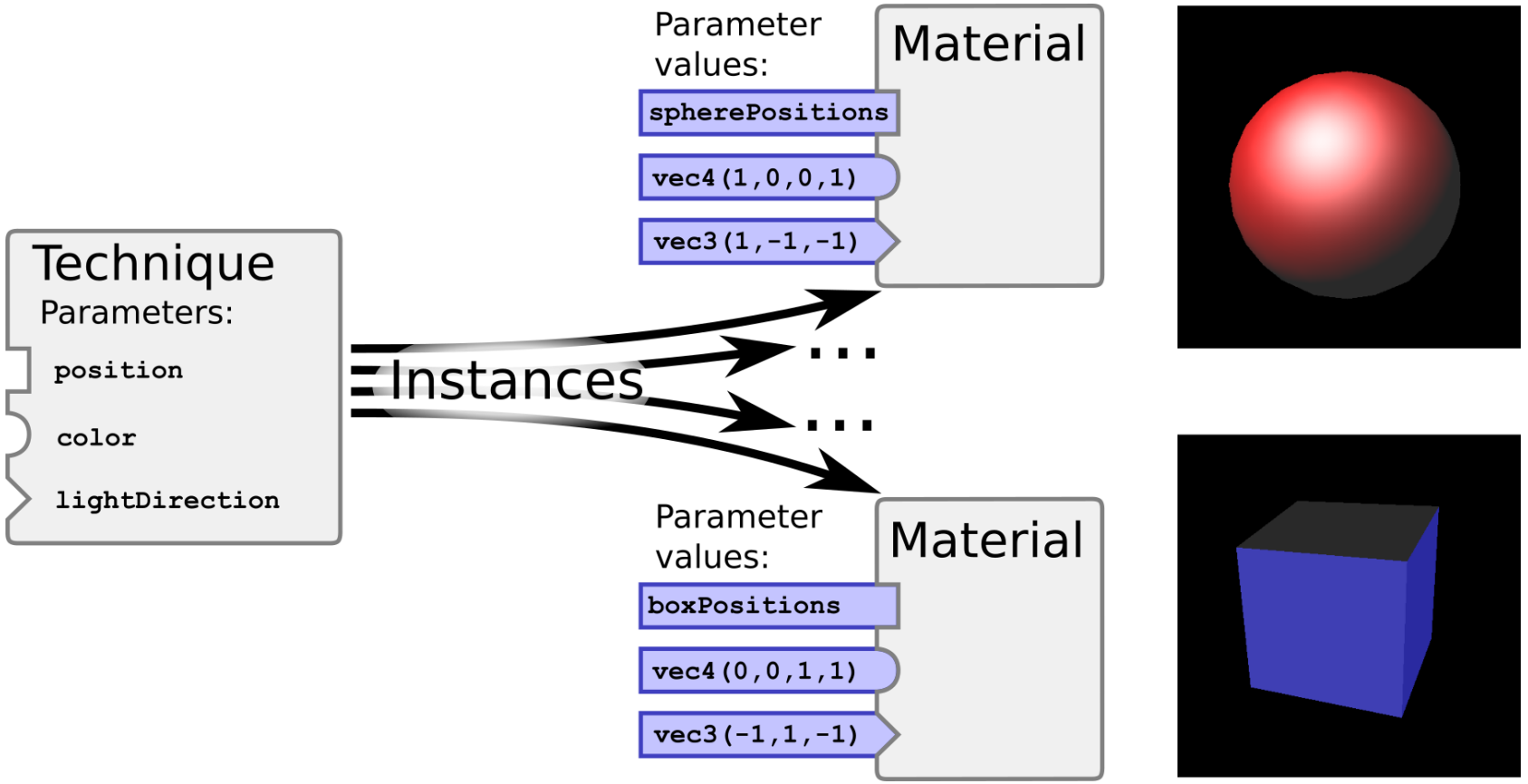
- Technique parameters describe shader attributes and uniforms



- Shaders are stored in external (GLSL) files

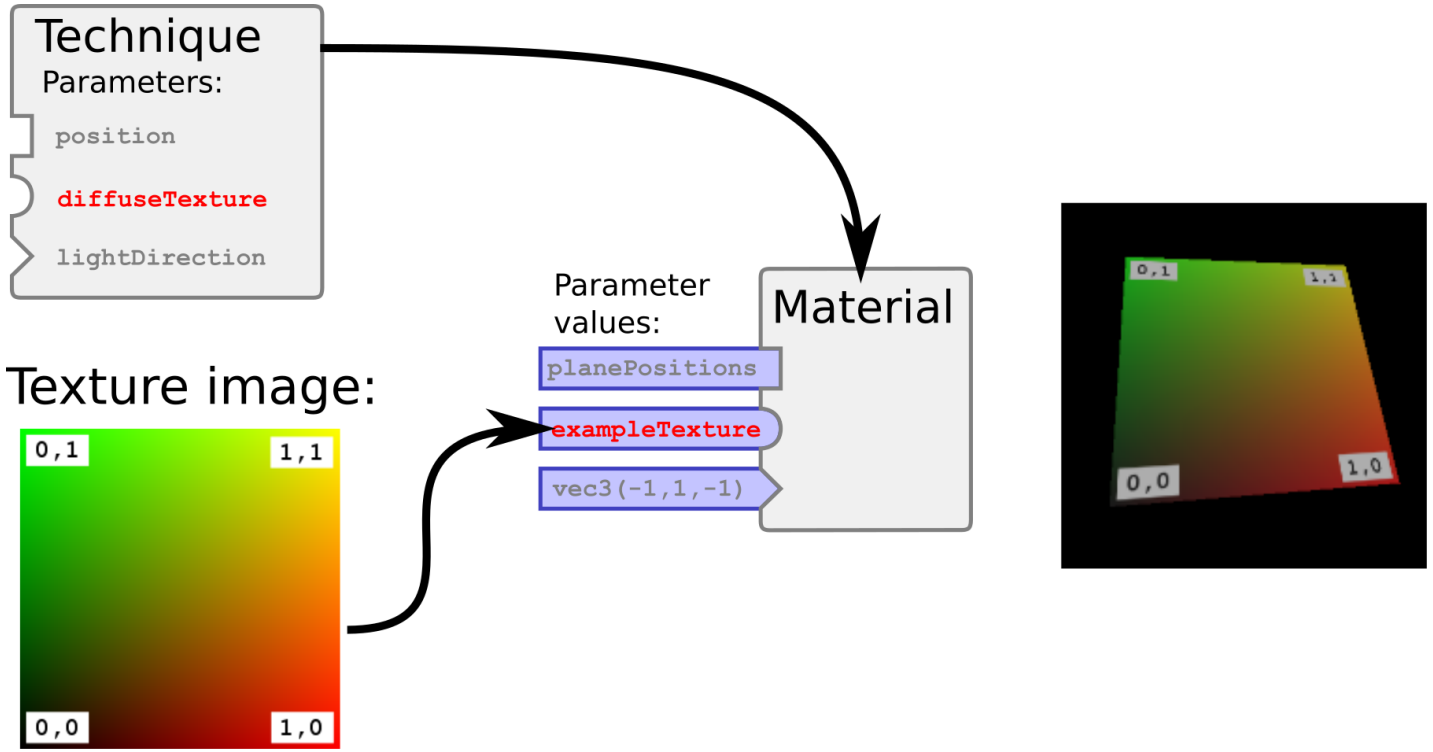
Materials and Techniques (extension)

- Materials are „instances“ of techniques



Textures

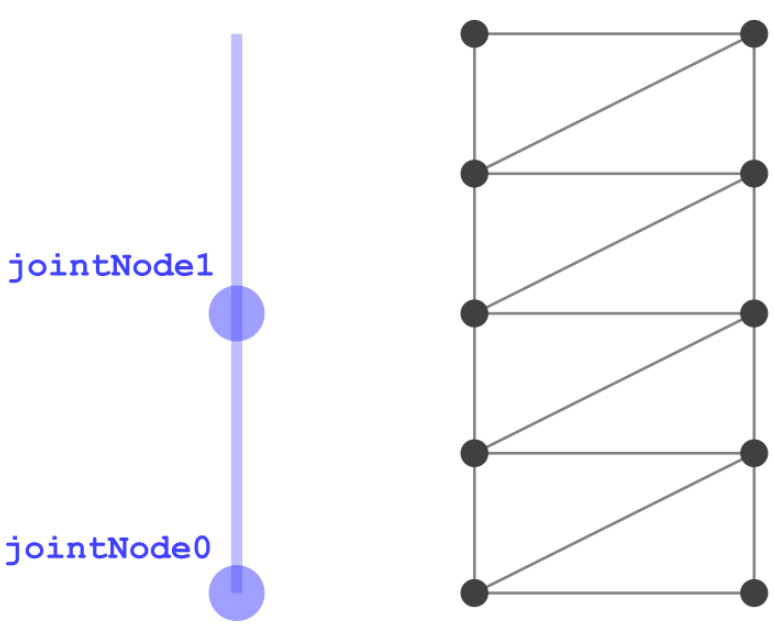
- A texture can be an input for a `sampler2D` uniform variable:

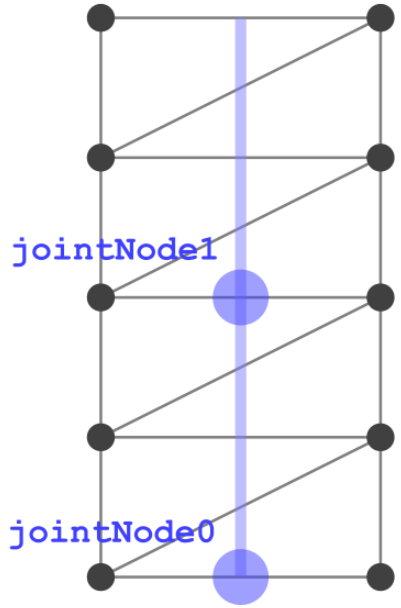


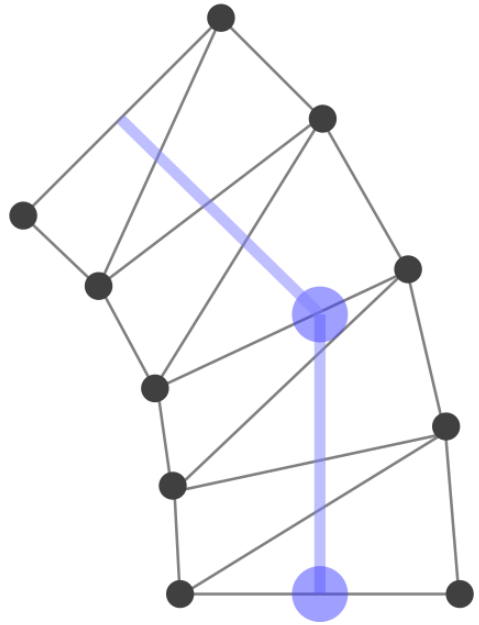
- Reminder: Texture images are stored as external files (JPG, PNG...)

Skins

Skeleton + Mesh = Skinned mesh : Vertex skinning

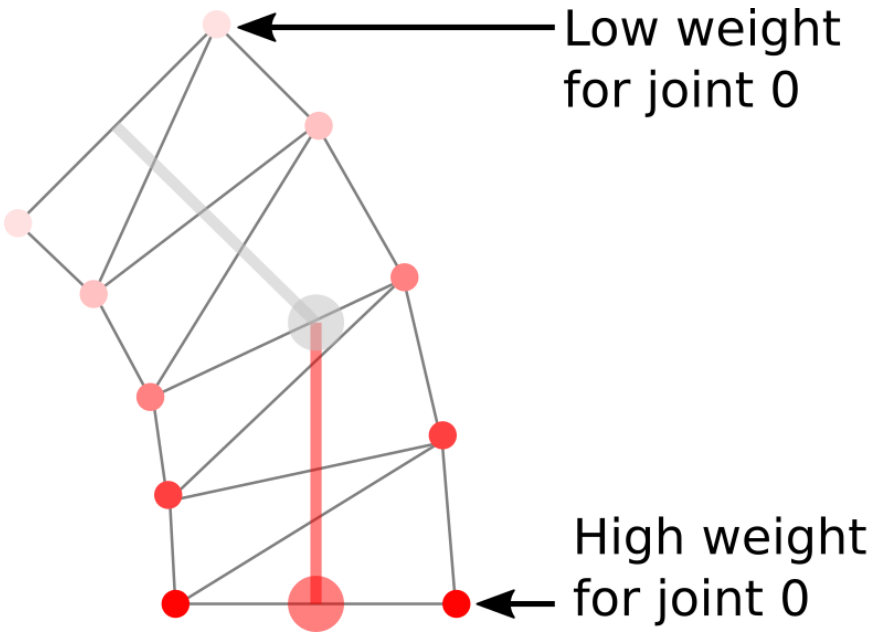


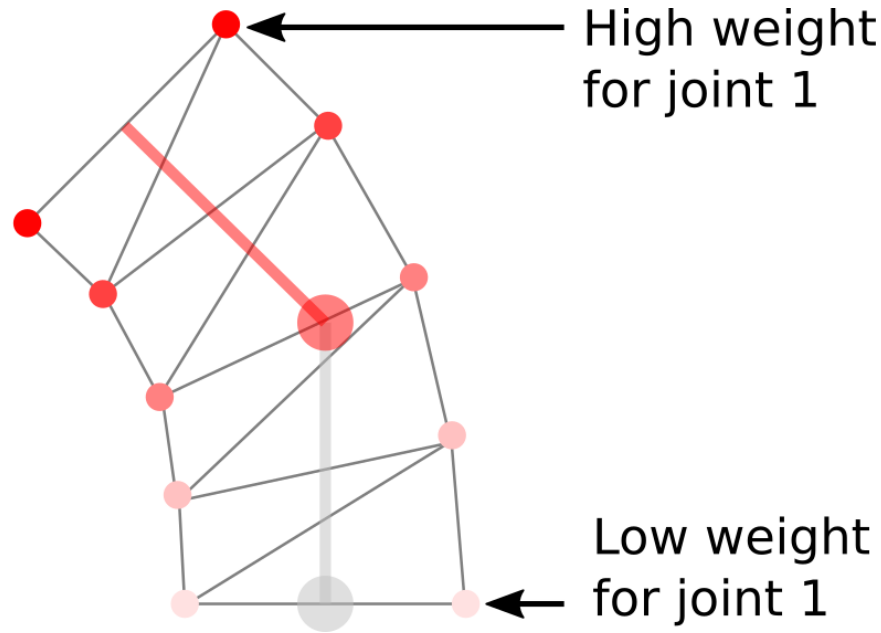




Skins

- The `JOINTS`: Which joints affect the vertices
- The `WEIGHTS`: How strongly the joints affect the vertices





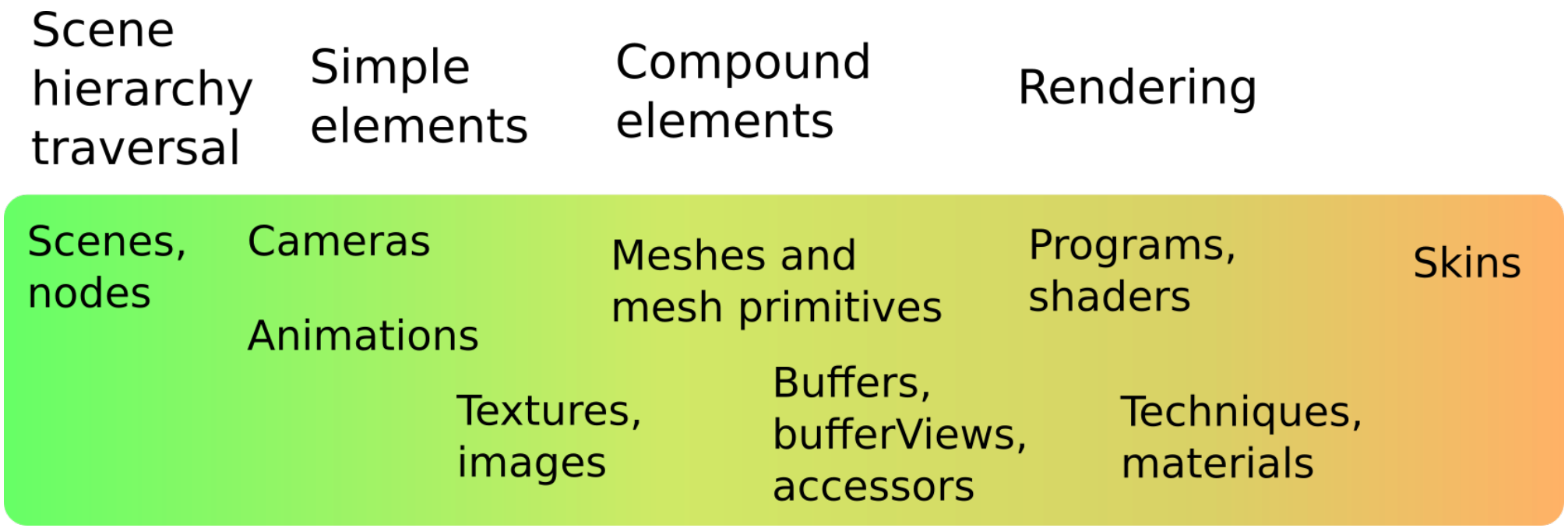
glTF resources

- **Landing page:** khronos.org/gltf
- **GitHub repository:** github.com/KhronosGroup/glTF
  - Links to resources, samples and the specifiction
- **Sample models repository:** github.com/KhronosGroup/glTF-Sample-Models
  - Simple models for learning, complex models for testing
- **Asset validator:** github.com/KhronosGroup/glTF-Validator
- **Converters:**
  - COLLADA: github.com/KhronosGroup/COLLADA2GLTF/
  - OBJ: github.com/AnalyticalGraphicsInc/OBJ2GLTF
  - **+ many others!**
  - Full list at github.com/KhronosGroup/glTF#gltf-tools
- **Try it out with the online drag-and-drop converter:**
  - cesiumjs.org/convertmodel.html

Getting started with glTF

- Get an overview of the glTF concepts and their relationships:
  - github.com/KhronosGroup/glTF#overview
- Explore each concept using the simple test models:
  - github.com/KhronosGroup/glTF-Sample-Models
- Dive deeper into each topic using the tutorials:
  - github.com/KhronosGroup/glTF-Tutorials
- Look up the details in the specification:
  - github.com/KhronosGroup/glTF/tree/master/specification

Writing a glTF loader or viewer



- **Have a look at the existing loaders and viewers:**
  - github.com/KhronosGroup/glTF#loaders-and-viewers
  - For JavaScript/WebGL, C++, C#, Go, Rust, Haxe, Java...

Contributing to glTF

- **Create loaders, exporters, converters or viewers**
  - To be listed at github.com/KhronosGroup/glTF
- **Contribute sample models**
  - To be added to github.com/KhronosGroup/glTF-Sample-Models
- **Write tutorials**
  - To be published at github.com/KhronosGroup/glTF-Tutorials
- **You are already using glTF?**
  - Let us know and share your story!

Khronos glTF Webinar: Questions?

- Download the spec, header files, tutorials, and more:
  - khronos.org/gltf
- Sign up for the Khronos newsletter:
  - khronos.org/news/subscribe
- Learn about becoming a Khronos member:
  - khronos.org/members
- Slides by Marco Hutter
  - gltf@marco-hutter.de
- Thanks to all glTF contributors!
  - Neil Trevett and Patrick Cozzi will now join to answer your questions