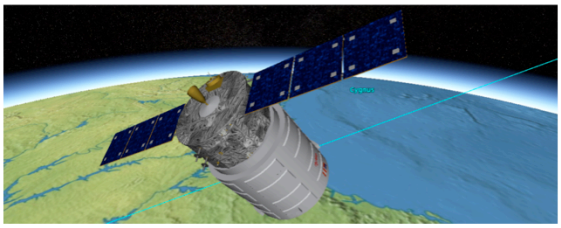
WebGL Content Pipeline with glTF



Patrick Cozzi, @pjcozzi
Analytical Graphics, Inc.
University of Pennsylvania

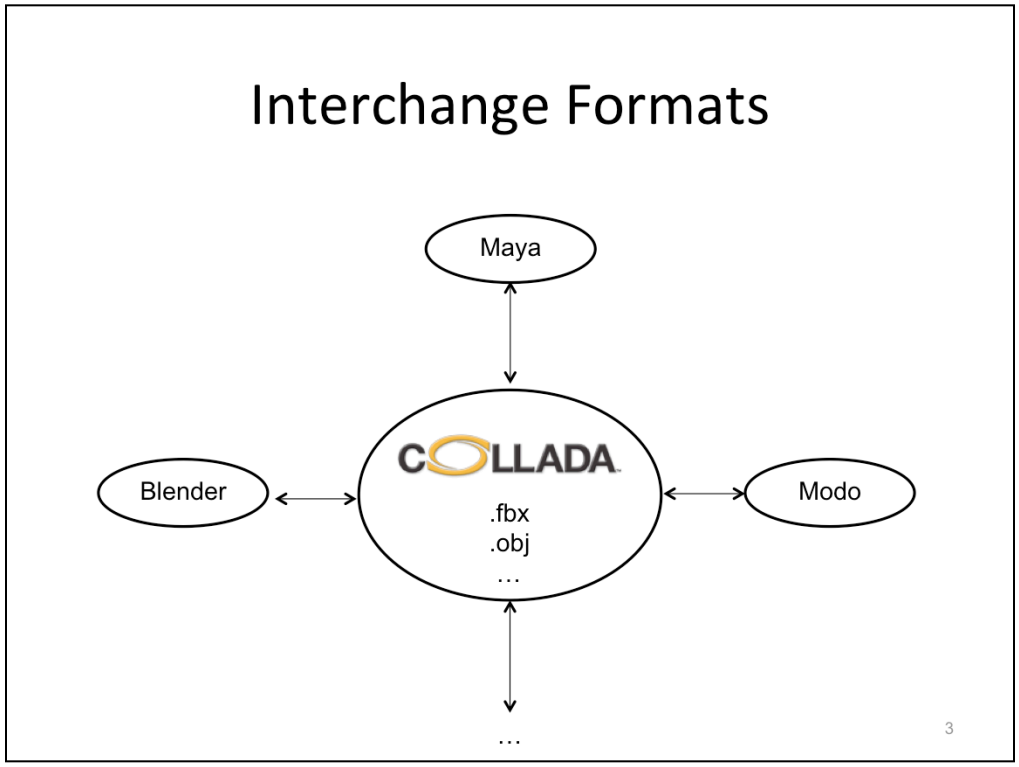
Like OpenGL, WebGL is a rendering API that exposes the capabilities of the hardware. It knows about low-level concepts like buffers, textures, shader programs, and uniforms. Artists, on the other hand, use modeling tools like Maya or Modo, to create assets using much higher-level constructs such as geometries, node hierarchies, materials, and animations. As engine developers, it is up to us to create a content pipeline that brings assets from modeling tools to our WebGL-based engines. Furthermore, this pipeline needs to produce runtime assets that are easy and efficient to use on the web with WebGL.

Historically engine developers have created custom asset formats for their engine and custom exporters for modeling tools or converters from interchange formats like COLLADA. This talk introduces glTF, the runtime asset format for WebGL, OpenGL ES, and OpenGL, which significantly reduces the amount of work engine developers have to do by providing an efficient and extensible format based on JSON and binary blobs, and an open-source content pipeline for creating glTF assets from COLLADA.

// TODO: everything after this!

Outline

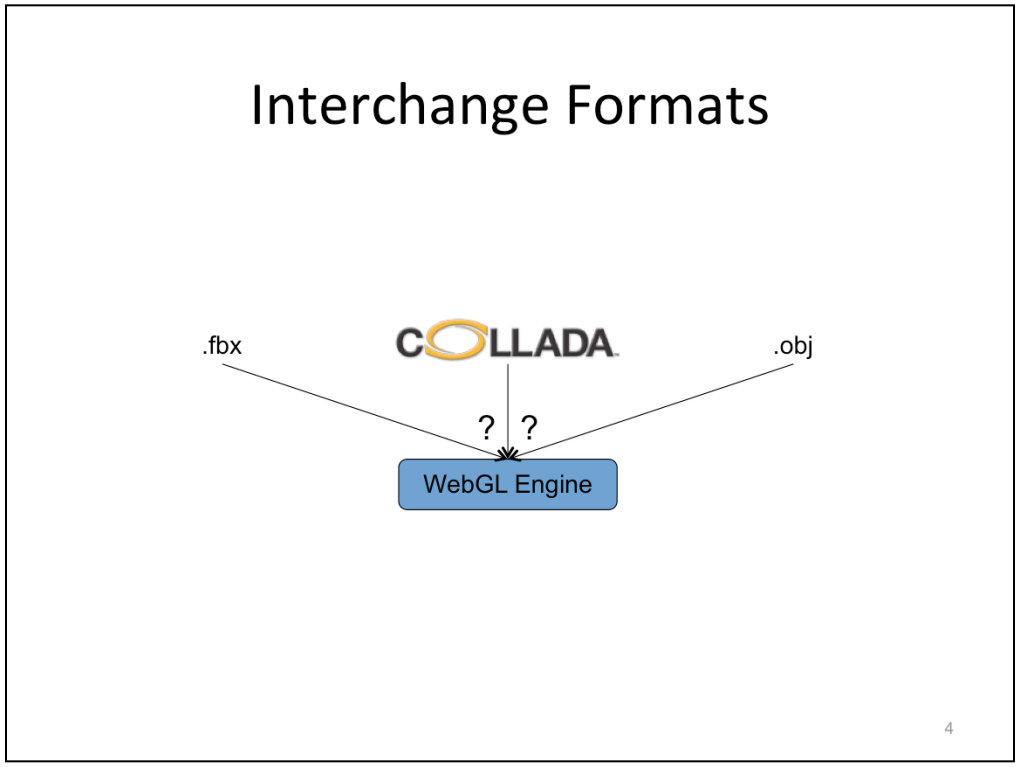
- Interchange and runtime formats
- glTF goals and schema
- COLLADA to glTF content pipeline



COLLADA is an open-standard from Khronos. Open-source OpenCOLLADA and COLLADA DOM read/write COLLADA file.

FBX is proprietary and owned by Autodesk. Autodesk has an SDK to read/write FBX. FBX can be binary or ASCII. There is an unofficial spec.

OBJ is originally from Wavefront. It is geometry only, so doesn't include animations, skins, physics, etc.



Interchange formats can move assets between tools, but what about between tools and the runtime engine?

Interchange Formats

- Target tools, not WebGL
- Example: COLLADA
  - XML + image files
  - One index per attribute, not vertex
  - Unsigned int indices
  - Transform stack per node
  - Polygons and splines
  - Common profile materials
  - Doesn't specify image file format
  - Lots of flexibility and indirection in animations and skins

5

Interchange formats are generally verbose and slow to load for runtime use.

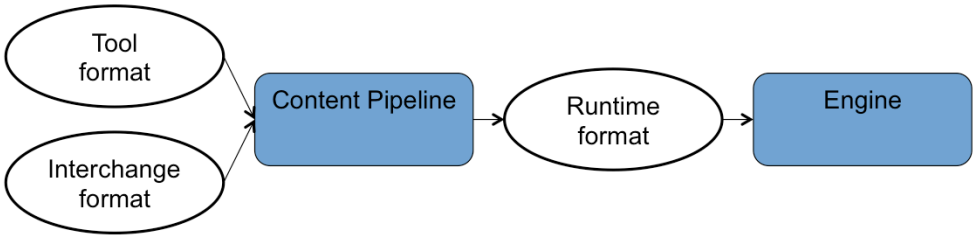
Interchange formats need to go through many conversion steps before a graphics API. This doesn't belong in a runtime; it belongs in the content pipeline.

Common profile materials – need to generate shaders to render. However, some engines will want to do this to match their g-buffer format for deferred shading, for example.

Keyframe animation supports several different splines. Great for interchange, but a runtime usually only needs one or two.

Runtime Format

- Optimized for use in an engine



6

Again, .bmp vs. .jpg example.

The content pipeline runs offline, perhaps as part of the build process. It does not ship with the game.

glTF

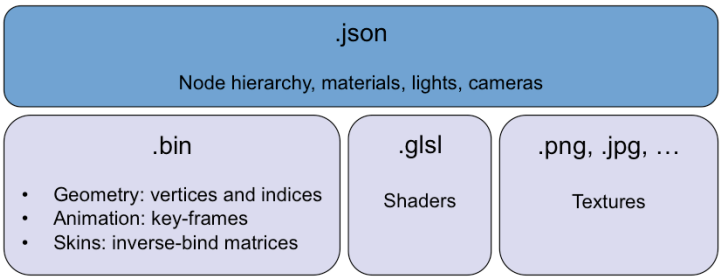
- "the runtime asset format for WebGL, OpenGL ES, and OpenGL"
- jpg, mp3, mpeg, … what about 3D?
- Open standard
- Not ratified yet

7

Story about how I got involved in glTF - http://blog.virtualglobebook.com/2013/03/how-i-got-involved-in-gltf-and-khronos.htm

glTF Goals

- Easy and efficient to render



8

Why JSON - cross-platform, compact, readable, allows validation, and minifies and compresses well

Geometry, animation, and skins are binary, unlike, COLLADA, for example, which uses XML

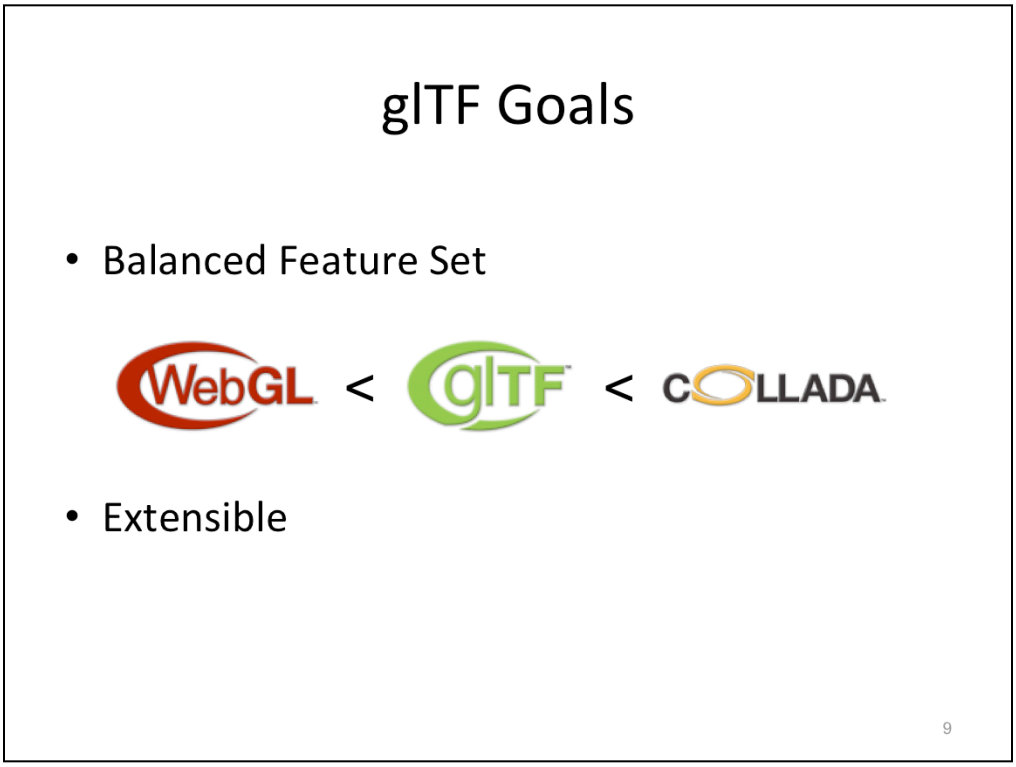
Binary data is little endian

Binary blobs - allow efficient creation of GL buffers and textures since they require no additional parsing, except perhaps decompression

Shaders can be in .json or separate .glsl files

Can have any number of .bin files

Flexible for a wide array of applications.



glTF has more features than a graphics API, like a node hierarchy, animation, and skins, but less features than an interchange format, like physics and spline representations.

Extensible – extra properties – forwards compatible

Logos from http://www.khronos.org/legal/trademarks/

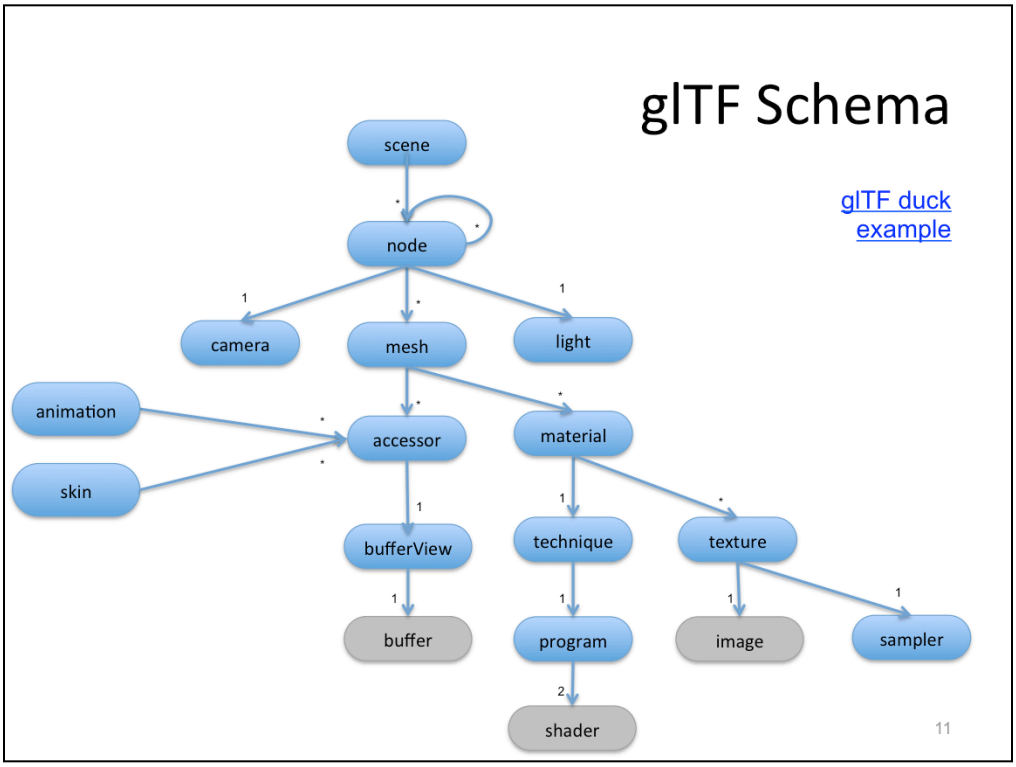
glTF Goals

- Code, Not Just Spec
  - Content Pipeline is key to adoption
  - Three.js is key to adoption
  - Implementations are needed for a sane spec
- Community
  - Grassroots and transparency
- WebGL, OpenGL ES, and OpenGL
  - Initial adoption - WebGL

https://github.com/KhronosGroup/glTF

We developed 4 renderers. Sometimes multiple times each.

Established engines like Unity and C4 already have a runtime format. WebGL engines are still emerging.



Bottom-up:

Geometry

- buffer – binary blob. Can be combination of geometry, animation, and skins
- bufferView – subset of buffer with target info (ARRAY_BUFFER, ELEMENT_BUFFER, animation/skin)
- accessor – subset of bufferView with type info, e.g., float-point. Similar to a call to glVertexAttribPointer
    - For example, a bufferView may be all vertices in the asset (think glBufferData), where as an accessor may be an individual attribute for a mesh (think glVertexAttribPointer)
- mesh – (composed of primitives, not shown) – corresponds to glDrawElements
- node – one or more meshes, plus transform, plus children

Material

- image – Image file
- sampler – texture filter and wrap modes, think glTexParameter
- texture – think glTexImage2D

TODO

- Schema example
- Converter example

Content Pipeline

13

Content Pipeline

- Optimize and package assets for use with the engine
- Several areas
  - Geometry
  - Animation and skins
  - Texture
  - Shaders

14

Usually not all hand coded, but instead a combination of many tools from different third-parties, e.g., texture compression, mesh compression, vertex cache optimize, etc.

Content Pipeline

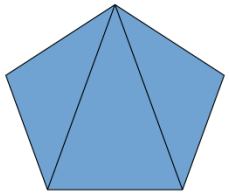
- Cleanup redundancies created by artist/ exporters
  - Remove unused nodes, meshes, materials, techniques, etc.
  - Remove unused vertices. Remove duplicate vertices
  - Remove duplicate materials and techniques
  - Combine primitives with the same material and vertex format

15

SketchUp story with one triangle per primitive.

Content Pipeline: Geometry

- Triangulation
  - Polygons ⇨ Triangles
  - Higher-order surfaces



16

Also, triangle strips/fans to triangles.

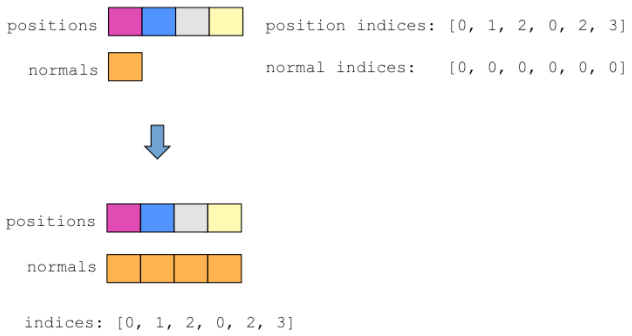
For polygons, this only adds indices so the payload increase is not that bad.

Ear clipping. Accelerate with spatial data structure.

Randomized algorithm. Select random cut. Split polygon if it doesn't intersect any edges.

Content Pipeline: Geometry

- Deindex
  - One index per attribute ⇨ one index per vertex



17

Deindexing reduces the about of index data but can increase the amount of vertex data. A single set of indices is required for glDrawElements and friends.

Example here is one side of a box. It is 2 triangles with one normal. Deindexing requires duplicating the normals.

This example:

Before:

Vertex data: 16 + 4 = 20

Index data: 12 +12 = 24

Total: 44

After:

Vertex data: 16 + 16 = 32

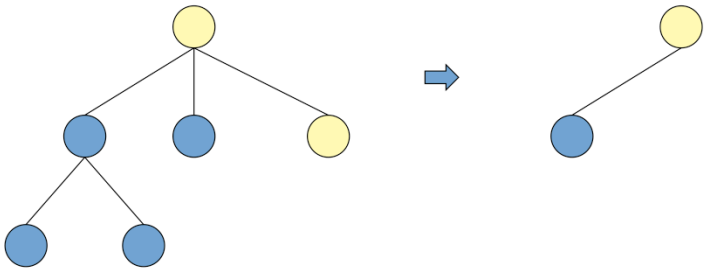
Index data: 12

Total: 46

If most attributes are unique (not shared by multiple vertices), deindexing can

Content Pipeline: Geometry

- Flatten node hierarchy



18

Node hierarchy or “scene graph.”

Increases the batch size and, therefore, reduces the number of draw calls.

Nodes need the same material (and vertex format, which is implied when they share material).

Transform combined meshes into the same coordinate system. Children have their transform applied when they are combined with their parent.

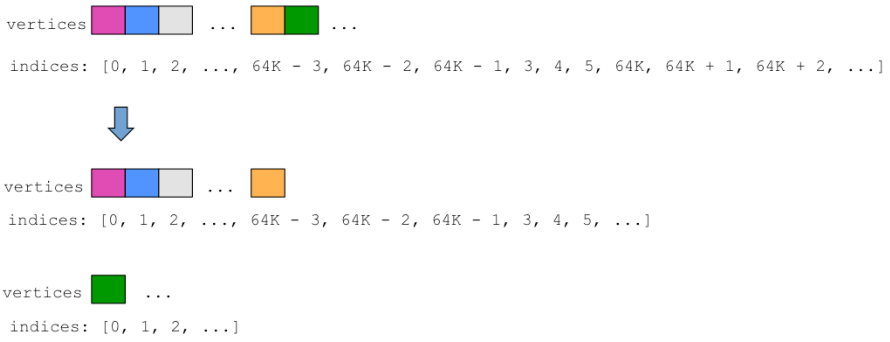
If a node is targeted by an animation, it’s sub-tree can be combined, but it can’t be combined with its parent.

Texture atlases help nodes have the same material since they share the same texture.

Also reduces the number of meshes and combines buffers as needed.

Content Pipeline: Geometry

- Split meshes
  - So indices fit into unsigned short



19

Needs to duplicate some vertices.

Without extensions, this is needed for WebGL 2 and OpenGL ES 2. The unsigned int extension is widely supported, and although it uses more memory and potentially contributes to cache pollution, I have not noticed a performance hit, and it allows for larger batches.

Content Pipeline: Geometry

- Compression
  - Open3DGC (TFAN)
  - Pre-gzip for web deployment
  - Easy tricks
    - Minify JSON, e.g., whitespace
    - Exclude default values, e.g., identity matrix
    - Uniform scale instead of non-uniform scale
    - 4x3 matrices instead of 4x4
    - Quaternions are normalized, only store 3 components

20

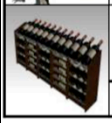
Need to consider decompression time along with the payload savings.

Open3DGC – http://kmamou.blogspot.com/2013/07/open-3d-graphics-compression-open3dgc.html

TFAN – fans, quantize, parallelogram predict, O(n) wrt vertices to decompress.

These transform tricks also apply to animation data.

Also see “WebGL Models: End-to-End” in OpenGL Insights (Chapter 30).

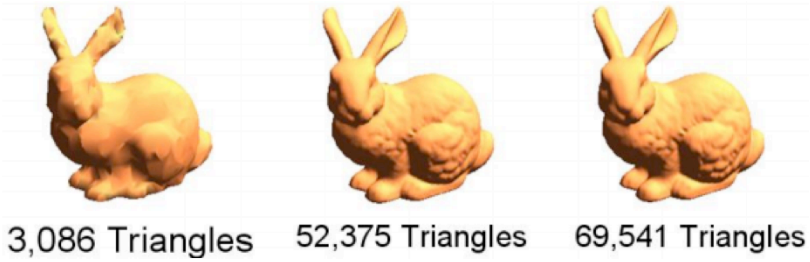
Model	COLLADA		glTF		glTF+Open3DGC ascii		glTF+Open3DGC binary	
	XML	gzip	raw	gzip	raw	gzip	raw	.raw bin .gzip JSON
			. bin:102k . JSON:11k	. bin:81k . JSON:2kb	. ascii:29k . JSON:11k	. ascii:19k . JSON:2k	. bin:18k . JSON:11k	. bin:18k . JSON:2k
	336k	106k	113k	83k	40k	21k	29k	20k
			. bin:9220k . JSON:75k	. bin:3220k . JSON:5k	. ascii:3080k . JSON:151k	. ascii:1510k . JSON:11k	. bin: 1622k . JSON:151k	. bin: 1622k . JSON:11k
	19767k	3417k	9295k	3225k	3231k	1521k	1773k	1633k
			. bin:25224k . JSON:183k	. bin:5738k . JSON:8k	. ascii:7793k . JSON:587k	. ascii:1433k . JSON:29k	. bin:3205k . JSON:589k	. bin:3205k . JSON:29k
	56763k	7378k	25407k	5746k	8380k	1462k	3794k	3234k
			. bin:329k . JSON:255k	. bin:99k . JSON:10k	. ascii:122k . JSON:267k	. ascii:61k . JSON:11k	. bin:71k . JSON:267k	. bin:71k . JSON:11k
	794k	133k	584k	109k	389k	77k	338k	88k

21

Slide from https://www.khronos.org/assets/uploads/developers/library/2013-siggraph-collada-bof/COLLADA-BOF_SIGGRAPH-2013.pdf

Content Pipeline: Geometry

- Generate LODs



Vertex clustering from my master's thesis is shown here.

QEM is most popular.

Many games are not using geometric LOD on their characters.

Many generate different models for different platforms – mobile vs. console vs. desktop.

Content Pipeline: Geometry

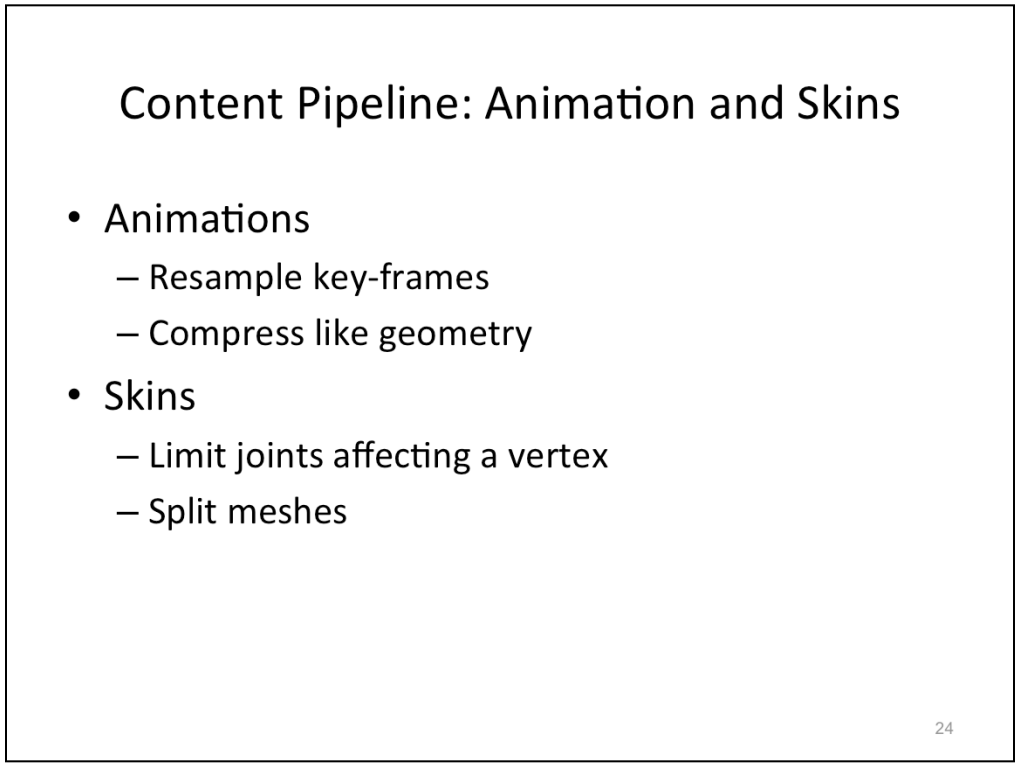
- Others
  - Consistent up axis
    - What's up? y? z? What's forward?
  - Re-order for the pre- and post-vertex-shader caches
  - Interleave vertex attributes?

Vertex cache optimization

* http://home.comcast.net/~tom_forsyth/papers/fast_vert_cache_opt.html

* http://gfx.cs.princeton.edu/pubs/Sander_2007_%3ETR/

Interleaving – see Chapter 3 - http://www.sci.utah.edu/~csilva/papers/thesis/louis-bavoil-ms-thesis.pdf



Animation:

15 fps may be fine instead of 30 or 60

Control points don't need to be uniformly sampled, e.g., samples can be removed for linear parts.

Don't store a channel for scale, for example, if it never changes.

See Section 11.8 in Game Engine Architecture

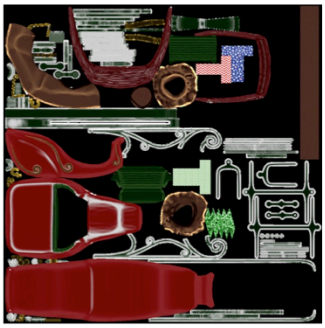
Skins:

2 joints for an elbow. 3 weights for a hip. Games rarely use more than 4 weights.

Mesh splits are required to keep joint matrices in a uniform array.

Content Pipeline: Texture

- Create texture atlas
  - Increases batch size. Reduces individual files



25

Care needs to be taken when mipmapping.

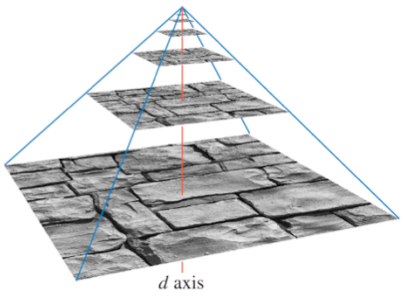
Packing a texture atlas is NP hard, see

- http://clb.demon.fi/files/RectangleBinPack.pdf
- http://clb.demon.fi/projects/even-more-rectangle-bin-packing
- http://www.blackpawn.com/texts/lightmaps/

- See https://developer.nvidia.com/sites/default/files/akamai/tools/files/Texture_Atlas_Whitepaper.pdf

Content Pipeline: Texture

- Generate mipmaps
  - Higher quality than doing it online
  - Increase size by 1/3



26

glGenerateMipmap may use a low-quality filter and/or be slow.

In addition to visual quality, mipmaps also help the GPU cache since sampling from the mip level has better spatial coherence than sampling from the full texture.

Image from Real-Time Rendering - http://www.realtimerendering.com/

Content Pipeline: Texture

- Convert image formats
  - For example, .bmp to .jpg
- Compress images
  - DXT / S3TC
  - ETC2

27

Format for modeling: lossless. Format for runtime: can be lossly. In our engine, we would compress satellite imagery except for the leaf nodes.

Often, we convert .tga to .png.

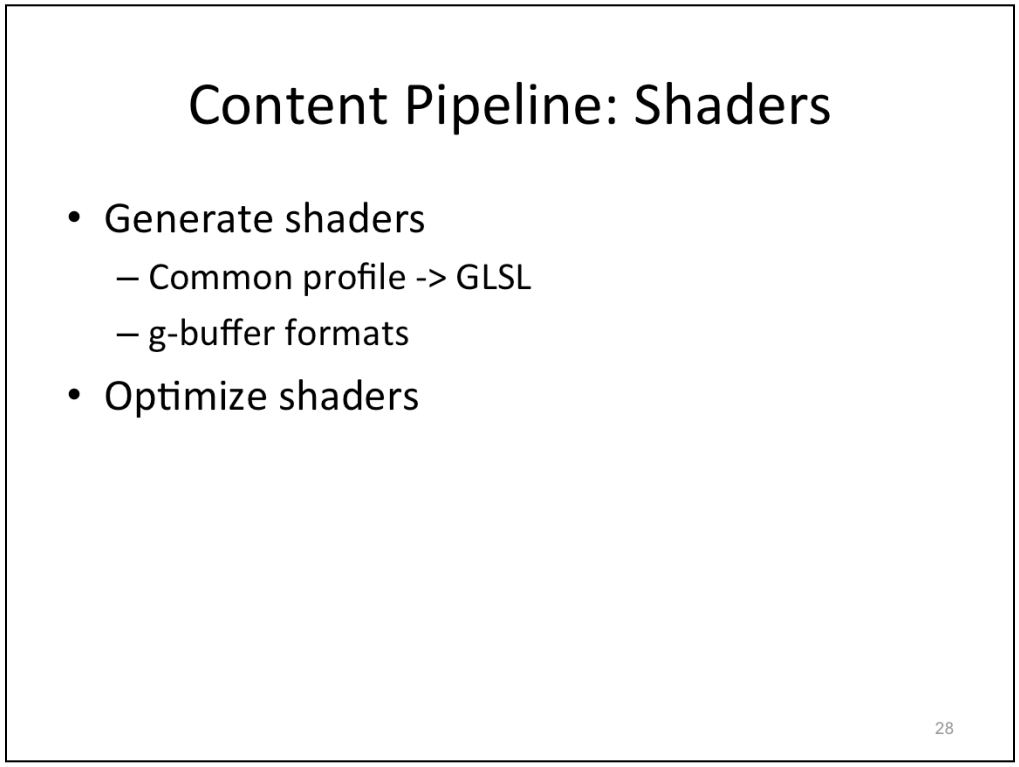
.jpg can have an alpha channel nowadays.

JPEG compression is better than DXT.

DXT – use PCA to fit a line through color space. Lossy. Slow to compress but fast to decompress on-the-fly in hardware. Also higher visual quality if compressing a larger texture, compared to a smaller uncompressed texture. Several versions of DXT, with and without alpha.

ETC2 required in ES 3.0 and GL 4.3. Higher quality than DXT at same bitrate. More flexibility in texture format, e.g., R and RG formats.

Also, ASTC (Adaptive Scalable Texture Compression), but is optional in GL.



g-buffer format is engine-specific.

Less important if shaders are hand-coded.

Combine uniforms. Replace uniforms with constants if they aren't targeted for animation (don't add more materials or techniques though).

Minify or just remove whitespace. Size is nothing compared to textures, geometry, and animations.

See

* https://github.com/KhronosGroup/glTF/issues/34
* https://github.com/KhronosGroup/glTF/issues/36

Themes

- Keep the runtime simple
- Push work to the Content Pipeline

References

Fabrice Robinet et al. glTF: Designing an Open-Standard Runtime Asset Format. GPU Pro 5, 2014

Patrick Cozzi. Building a WebGL Santa with Cesium and glTF. 2013