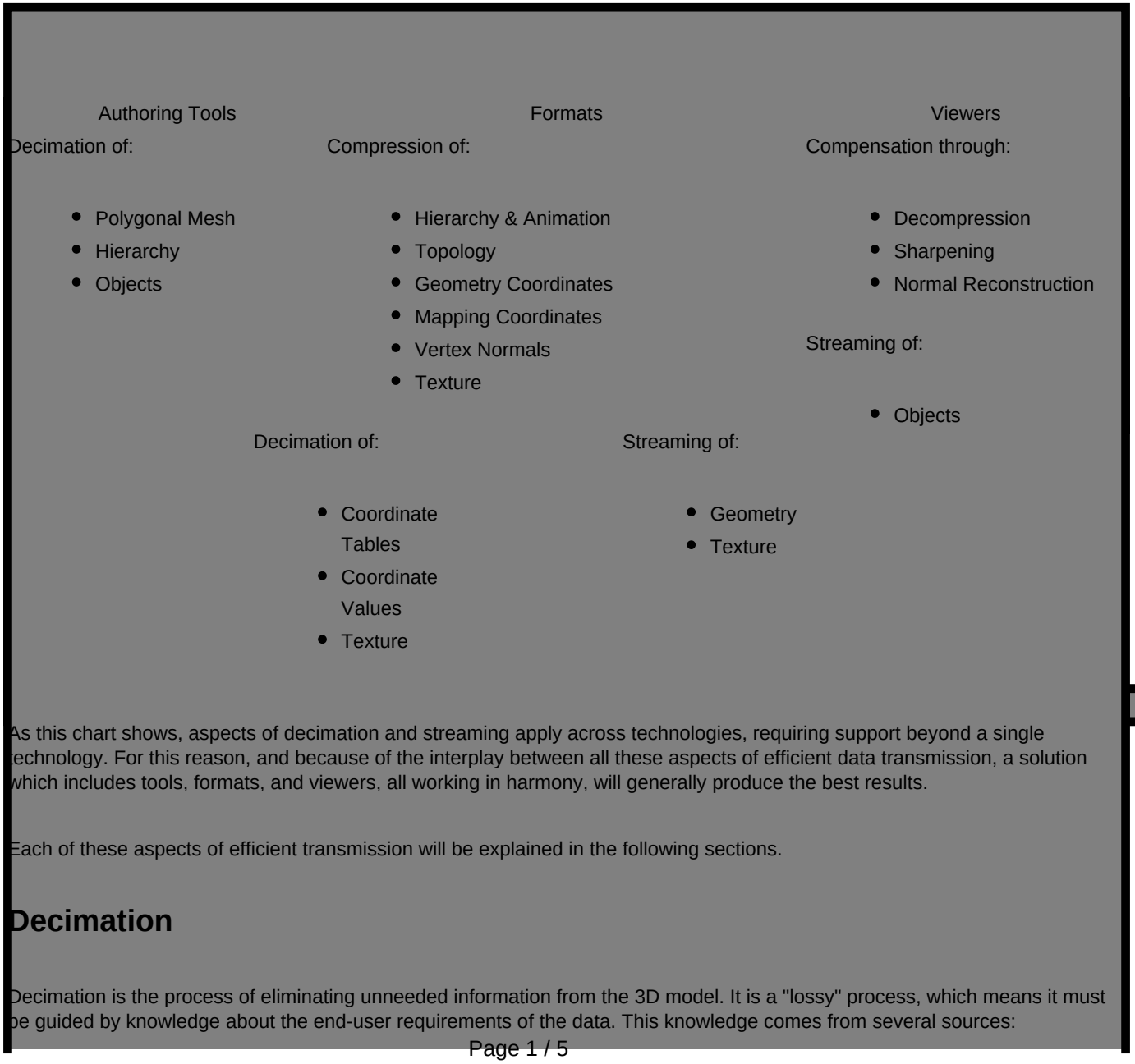
Advanced Editing

Technical aspects of optimizing large 3D models for the web

3D models are among the most complex kinds of data to transmit efficiently. A comprehensive web optimization solution requires Decimation, Compression, and Streaming. Failure to address any one of these areas, even in part, can make an otherwise good approach extremely inefficient.

Decimation is a "lossy" process, in which high-resolution data, which is not required in a web application, is removed. *Compression* is the process of representing a large data set using a smaller data set, by exploiting patterns or other kinds of predictability in the data.

Streaming is a technique used to send data while the user is looking at the running application, to reduce the perception of delay. These three techniques apply across three domains of technology — Authoring Tools, Formats, and Viewers — as shown in the following chart:



As this chart shows, aspects of decimation and streaming apply across technologies, requiring support beyond a single technology. For this reason, and because of the interplay between all these aspects of efficient data transmission, a solution which includes tools, formats, and viewers, all working in harmony, will generally produce the best results.

Each of these aspects of efficient transmission will be explained in the following sections.

Decimation

Decimation is the process of eliminating unneeded information from the 3D model. It is a "lossy" process, which means it must be guided by knowledge about the end-user requirements of the data. This knowledge comes from several sources:

- **Technicial Input**

  Technicians select parameters based on their understanding of end-user requirements.

- **Viewing Platform Limitations**

  By making the software aware of limitations (such as maximum zoom) of the viewing platform, it can eliminate data which would not be visible to the end-user.

- **Behavioral Analysis**

  By making the software aware of the behavior of the 3D model, it can eliminate data which will not be relevant for a given application.

A 3D model is generally represented by a "scene graph," in which a hierarchy of transform nodes tie together various parts. At each of these nodes, there may be geometry. There are many parts of this graph which may be decimated. The next sections discuss each of the candidates.

Mesh Decimation (Polygon Reduction)

Tessellated 3D models often have more triangles in each mesh than are really necessary for a given end-user application. In some cases, such as co-planar polygons, this decimation requires no technician input; but in general, a technician will need to make quality determinations to decide how aggressively the system should remove polygons from the tessellation.

In the Strata Live 3D software system, Strata Live 3D uses a quality-driven approach to decimation. Rather than giving an exact polygon count goal, which is impossible for the technician to know in advance, the software is given an abstract quality metric, and the reducer will eliminate as many polygons as possible without dropping below that quality level. This allows the technician to work on large hierarchies of objects at a time, and then do fine-grained adjustments to the decimation as needed.

Hierarchy Decimation (Merging Nodes and Meshes)

3D scenes graphs often have far more hierarchy than is necessary for an end-user application. For example, an Alias Studio model of an MRI machine which a Strata Live 3D client was using, had about 3000 nodes in its hierarchy, but only about a dozen moving parts. Analogous to automatic "inlining" in a compiler, the Strata Live 3D software will analyze the motion and visibility operations actually used in an application, and identify the levels of hierarchy that are irrelevant to animation. It can then decimate these out of the scene graph, apply the necessary transforms, and merge child node mesh data.

Coordinate Table Decimation (Welding)

Triangular meshes are generally represented using tables of coordinate values (3D positions, vertex normals, and texture mappping coordinates). Often these tables will contain values which are duplicated, or very close to duplicated. This redundant data can be decimated through a "welding" process, in which values which are within a defined threshold are unified. The maximum threshold to use can be determined with knowledge of the resolution limitations of the end-user application. However, placing this value under control of a technician allows the use of a more aggressive weld threshold to further reduce model size.

Coordinate Value Decimation (Quantization)

3D scene graphs generally use floating-point representations for all coordinate data (positions, normals, and mapping coordinates). However, floating point data is extremely difficult to work with in compression, so this data will first be quantized into an integer representation for subsequent compression. Quantization is a kind of decimation, because it effectively decimates the least significant digits of each coordinate value. The choice of the number of bits to use for each integer might be determined through statistical analysis of the dynamic range, and variance, of the sample data. However, again it is best to leave this under technician control, since an algorithm cannot effectively determine which variance is noise, and which is actually important geometry.

Texture Decimation (Cropping, Down-Scaling, Noise Reduction)

For many classes of 3D scenes, the texture data can dwarf 3D geometry. For example, Strata Live 3D's photographic process

regularly produces models with a hundred megabytes of texture. This texture data must be aggressively decimated as well. Three approaches are used by Strata Live 3D's Strata Live 3D software to get the number of pixels down to a reasonble level: cropping, down-scaling, and noise reduction.

First, the texture coordinates used by the meshes are analyzed to determine which image pixels are actually mapped to triangles, and unused pixels are cropped out of the image (either by reducing the rectangular boundary of the image, or by replacing these pixels with an average color value).

Next, the images are scaled. By default, Strata Live 3D uses a quality-based approach to scaling images. The image is scaled down as much as possible, without dropping the quality of the image below a threshold (relative to the unscaled original). Using this approach, a largely uniform texture might be scaled down a lot, whereas a highly variable texture will be scaled very little. In addition to this quality-based approach, a technician may choose individual scale factors directly, for more optimal reduction.

Finally, under technician control, the images may be noise-reduced. Noise reduction is a common data pre-conditioning technique used on image data, to ensure the compressed image is as small as possible. For example, in the Save to Web feature of Photoshop, applying a Gaussian blur is an option in the JPEG compression step. Of course, Strata Live 3D uses a more sophisticated noise reducer than simple Gaussian blur

Both noise reduction and scaling can make images look blurry, so it is often desirable to augment these techniques with a post-decompression sharpen filter in the viewer, to re-introduce much of the original clarity of the image.

Object Decimation (Invisible Node Removal)

Object decimation is analogous to dead-code removal in a compiler — by performing static analysis of the potential dynamic nature of a scene graph, the software can determine that some parts will never be visible in the final application. These objects, and texture images used by these objects and no others, can be dropped from the scene.

Compression

After decimating out as much data as possible, the remaining data must be effectively compressed. Strata Live 3D uses a proprietary technique that we call Dedkov compression. Here, we discuss the kinds of data that must be compressed, but we do not get into the details of our proprietary approach.

Hierarchy & Animation Compression

After Decimation, there is usually not much hierarchy left to compress, and animation information is generally fairly small to begin with, so ordinary ZIP-style compression is adequate for this data.

Topology Compression

The topology of the mesh describes how the various 3D coordinates are connected together into triangles.

Geometry Coordinate Compression

The geometry coordinates give the exact 3D location of each vertex in the mesh.

Mapping Coordinate Compression

Mapping coordinates dictate the method by which 2D image maps should be overlaid onto the geometry. As 3D modeling techniques evolve, more and more maps are being applied to geometry (diffuse, bump, displacement, etc.), so any technique should support multiple mapping coordinates per vertex.

Vertex Normal Compression

Vertex normals make up a large part of the data to be transmitted, so special attention is often paid to compressing this data. However, for almost all tessellated models, the vertex normals can be easily calculated by averaging some of the face normals of triangles adjacent to each vertex. Exactly which faces to include in this calculation can be encoded using a technique called "smoothing groups." In some cases, the smoothing groups will be available in the source data, because they were used in the modeling process to drive surface normals to begin with. In other cases, they must be computed by analyzing the normals. Encoding smoothing groups is a far more effective compression technique than any other option for vertex normal compression.

Texture Compression

For many models, texture image data will be many times larger than all other data combined. Thus, any comprehensive solution to compression must focus on texture compression as well. There are two classes of lossy image compression, characterized by the familiar JPEG and GIF approaches. However, these formats are quite outdated, and there are better alternatives for both, as described below.

Lossy Frequency Domain Image Compression

JPEG is the dominant image format which uses lossy frequency domain image compression. In this technique, color information for blocks of pixels is transformed from the amplitude (intensity) domain, into the frequency domain, filters are applied to remove high-frequency changes, and the filtered data is encoded.

A modern technique which works similarly, but produces consistently superior results, is wavelet compression. The JPEG-2000 standard is an example format for wavelet compression. Strata Live 3D's Strata Live 3D software uses a proprietary wavelet-based image compression technique, which is similar to JPEG-2000 but has a much smaller decoder footprint, so it is practical to use in a Java applet.

Paletted Image Compression

For certain kinds of images, such as screen shots, JPEG-style compression requires very large files to produce acceptable quality. For these images, it can be beneficial to use a paletted compression technique instead. In this approach, a palette of colors is built for the image, and each pixel uses a color from this palette. GIF and PNG-indexed are popular formats which uses this technique. Strata Live 3D's Strata Live 3D software uses a proprietary technique which achieves dramatically better compression than these formats for paletted images.

Streaming

After decimation and compression, a hundreds-of-megabytes data set may have been reduced to hundreds-of-kilobytes. While 1000:1 compression is admirable, files spanning hundreds of kilobytes will still take a long time to load over most Internet connections. To reduce the perceived delay, some form of streaming is often desirable.

The exception to this is cases, like PDF, where the usual user paradigm is to download, then view. For formats like that, using a streaming mechanism is pointless, and probably counterproductive, since the streaming algorithms will undoubtedly make it take longer to load the 3D model.

But for web-based applications where streaming makes sense, there are a few variations to consider: streaming objects, geometry, and/or texture. The are examined in the next sections.

Object Streaming (Instancing)

Object-level streaming is an application-specific optimization, which has nothing to do with the decimation, compression, or file formats used to transmit data. A good example of object streaming would be a product catalog that only loaded 3D models selected by the user, instead of loading all the 3D models in the entire catalog and selectively changing visibility of individual models. Clearly, in a case like this, object-level streaming is a requirement. But all that it requires is that each viewable object be

stored separately (or at least clearly indexed within a larger file), and that viewer logic be included to download the visibile parts as they are needed.

Geometry Streaming (Progressive Meshes)

A great deal of research has been devoted to streaming geometry data, and it is the primary claimed benefit of the U3D ECMA standard. However, we have elected to omit this kind of streaming from Strata Live 3D solutions for two reasons. First, it is not necessary. After decimation and compression, the geometry component of a model is rarely more than a few tens of kilobytes. For example, with a typical compression rate of 8 bits per triangle, a 50,000 triangle CAD model (which may have been decimated down from 500,000 triangles) will only be 50KBytes. For models created with 3D visualization tools, polygon counts exceeding 20,000 are rare. Adding the complexity of streaming geometry to change the user experience during the first 5 seconds of viewing hardly seems worth the trouble.

However, if geometry streaming had no down side, it might be worth doing anyway. But the downsides, are, in fact, huge. First, progressive meshes cannot be efficiently compressed. In addition to all the data which already needed to be encoded, the rules to migrate the initial, low-resolution mesh, to transform it into the final high-resolution mesh must also be encoded. This vastly increases the entropy of the system, resulting in much lower compression ratios.

Second, unpacking progressive meshes takes a lot of computation. Thus, the user may be seeing a progressively improving visual appearance, but there is no opportunity for the user to interact with the lower resolution data, because the CPU is spending all its time decoding the data and applying the progressive mesh transforms. For large data sets, splitting the CPU cache between unpacking the data, and displaying it on the screen, can lead to serious performance degradation.

Texture Streaming

For photographic-quality models which have been properly decimated and compressed, the amount of data devoted to texture is typically 10 times the data devoted to geometry. In order to provide the best user experience, streaming in progressive resolution in texture is a requirement (except in the download-and-view case, such as in PDF files). And unlike progressive mesh encoding, progressive encoding of images generally has no detrimental effect on overall compressed size. For example, the familiar progressive-JPEG approach will typically yield a file somewhat smaller than an ordinary JPEG encoding. And interlaced GIF files are exactly the same size as non-interlaced GIFs.

The same principle holds true in wavelet encoding, which is the dominant texture encoding method used by Strata Live 3D's software for photographic-quality, textured models. Each "band" of wavelet data progressively increases the resolution, color fidelity, or intensity fidelity of the image. So merely showing this data as it is decoded gives the users a progressively improving quality level, at almost no added cost.

Summary

Many tools, formats, and viewers focus on individual parts of the problem of efficient 3D data transmission, but by only working on one part of the problem, they fail to create useful end-user applications. Large 3D models and scenes require a holistic approach for web optimization, including Decimation, Compression, and Streaming. By addressing all these technology needs, Strata Live 3D and the Meson viewing platform, together provide a comprehensive solution to make high-fidelity 3D feasible in a web environment.

Unique solution ID: #1092
Author:
Last update: 2010-10-27 00:28