A Web Architecture for progressive delivery of 3D content

Efi Fogel
Enbaya, Ltd.

Daniel Cohen-Or[*]
Tel Aviv University

Revital Ironi
Enbaya, Ltd.

Tali Zvi
Enbaya, Ltd.

Enbaya, Ltd.[†]

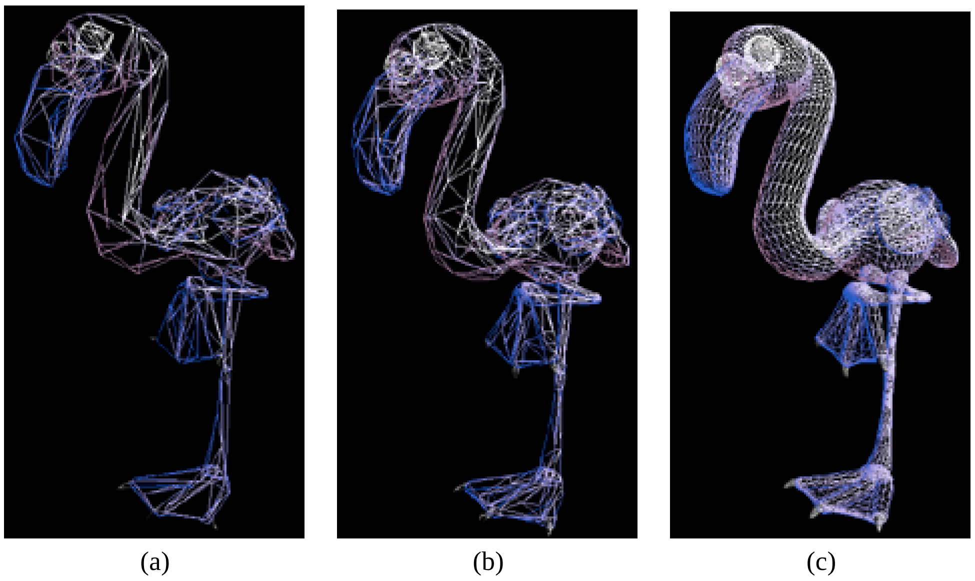


Figure 1: The Flamingo model at 3 resolution Levels, wire-frame view. (a) Lowest Resolution Level, (b) Medium Resolution Level, and (c) Highest Resolution Level.

Abstract

In this paper a Web architecture for 3D content delivery is presented. The architecture is based on a progressive compression representation integrated into the X3D framework. The architecture is designed to enhance the Internet user experience by delivering 3D content quickly, reliably, and with high quality. The progressive compressed stream enables handling 3D models containing large amounts of polygonal and textual data. At the client end the stream is progressively decoded up to the "best" level-of-details as defined by the client computational resources and the user inclination.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representationsGeometric Algorithms, Languages, and Systems; I.3.6 [Computer Graphics]: Methodology and Techniques—Graphics Data Structures and Data Types; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation;

Keywords: multi-resolution, streaming, geometry compression, progressive meshes.

[*] daniel@math.tau.ac.il
[†] {efi,tali,taliz}@enbaya.com, www.enbaya.com

1 Introduction

X3D is the latest evolution in 3D graphics specification. One of its main driving forces is the need to transmit 3D graphics content over the Internet quickly, reliably, and with high quality. That means providing the end user with an immediate response while enhancing her experience over time. A compact, progressive, and yet simple format is introduced here to achieve this goal. A compact format reduces the data size for fast transmission, and a simple file structure allows for fast reading and loading. A progressive scheme exploits asynchronous transmission for better interaction and quicker response time.

Asynchronous transmission can be employed in different ways. Consider, for example, a very large environment that is transmitted to a client along with a view volume that contains only a small portion of the entire environment. The visible objects can be rendered and displayed regardless of the transmission of the rest of the data. If the visible volume contains detailed objects in terms of their geometry or texture, it is desirable to render the object in low level of details (LOD) and allow the user to interact with the scene before the transmission of all LOD's is completed. The asynchronous transmission of the LOD representations of 3D objects is the focus of this paper. Later it is shown how X3D is extended to support this feature.

The most common representation of 3D geometric models to date is the polygonal mesh, in particular the triangular mesh - a collection of triangles that form one or more three-dimensional surfaces in space, and together define the object geometry. This representation is common due to the traditional ability of graphics hardware to draw triangles efficiently. Representing an object with high level of visual realism requires an enormous number of triangles.

The main reason holding back the wide spread of 3D content over the Internet is the large size of 3D models. The initial view is critical, as Internet users are reluctant to wait for "heavy" web pages to download. At the same time, the large size of the models are burdening the web sites in terms of maintenance and storage space. The image of true interactive 3D models should appear on the end-user screen at least as fast as common still images (e.g., JPEG, GIF, etc.) appear, without compromising with the rendering quality. It is crucial to minimize the overall size of the 3D-content stream, and the rendering time of the first view. Our solution is based on the progressive compression technique introduced in [3]. It combines three principals (i) mesh compression, (ii) level-of-details, and (iii) progressive transmission. The above three principals are covered in details in the following section.

2 Progressive Compression Streaming

The problem of minimizing the storage space for a model - represented as a triangular mesh - can be addressed in two alternative approaches. One is lossy compression, that is - there is a loss of information that enables a high compression rate. The other approach is lossless, where the exact original model is retained. A lossy method uses a mesh simplification method to reduce the number of faces (polygons) that represent the model. The lossless approach is a mesh compression method that minimizes the space taken to store a particular polygonal mesh. Some recent mesh compression methods [12, 11, 9, 4, 6] provide extremely efficient mesh compression rates. Other mesh compression algorithms provide the progressiveness property, in which every prefix of the encoded data is a progressive approximation of the original shape [1, 3, 8].

To improve rendering performance, it is common to define several versions of a 3D model, at various levels of detail. A fully-detailed mesh of triangles is used when the object is close to the viewer, and coarser levels of detail are substituted as the object recedes. Each such Level-of-Details (LOD) is an approximation of the full mesh at a lower resolution, containing a smaller number of triangles.

To create the LOD's, a model simplification technique variant is used. (e.g., [7]) that creates a new representation of a given model with fewer primitives (e.g., data points or polygons), while retaining some user-defined error tolerance. Instantaneous switching between LOD's during the display may lead to perceptible "popping" effects. These effects can be alleviated by model simplification. Model simplification (e.g., [7]) is the technique for creating a lighter representation of a given model with fewer primitives (e.g., data points or polygons), while retaining some user-defined error tolerance.

When a triangular mesh is transmitted over a communication line, it is desirable to progressively show better approximations of the model as data is incrementally received. One approach is to transmit successive LOD approximations, but this requires additional transmission time. The adopted technique [3] offers a progressive transmission scheme [5] with no significant overhead. This leads to best visual response time, where the user gets the minimum initial response with a reasonable visual quality. The initial model is immediately ready for interaction and manipulation by the user, because it is a true 3D representation, which can be, for example, rotated for inspection from different angles.

Figure 2 shows a series of frames, which demonstrate the progressiveness of the model transmission. The first image (Figure 2(a)) is created by the lowest-resolution approximation data, and it appears on the end-user screen after no more than 5K bytes of data have arrived over the network (about one second on low-end PCs with a home-modem connection). The second image (Figure 2(b)) is displayed after the arrival of additional 15K bytes only. The final image (Figure 2(c)) is displayed after the arrival of the last 30K bytes. While the stream of data is arriving and the model is being progressively reconstructed, the user can interactively change the viewing parameters.

In comparison to standard image files which are usually included in web pages, the typical size of the JPEG data of the 480x360 image of Figure 2(c)) would be over 50K bytes for a high-quality image, and over 10K bytes for a low-quality image. Other image formats such as GIF are even larger than that, and animated-GIF images are significantly larger. The same holds for Video data. A progressive transmission technology has inherently a fine-grain levels-of-detail support. In addition to the immediate visual response time and the continuous progressive improvement of the visual quality, the levels of detail are advantageous as a means for reducing data transmission as well: it may very likely be that the viewer doesn't need all the levels of detail. Rather, the higher levels of detail may be too complex for rendering on the end-user machine, or maybe the machine is overloaded with other processes. Another possible situation is that the position of the 3D model in virtual space on the end-user display is simply far enough from the viewer so that the higher LOD's do not contribute to the final image at all, and there is no need to download and render them. In such situations the client can avoid the retrieval of the unnecessary data.

3 Mesh Encoding

A lossless compression method based on a multi-resolution decomposition [3] has been adopted. The method uses a hierarchical simplification scheme, which generates a multi-resolution model of the given triangular mesh by applying a simplification technique [10]. On the client side the process is reversed defining a hierarchical progressive refinement stream. A simple prediction plus a correction is used for reconstructing vertices to form a finer level. Furthermore, the connectivity of the triangulation is encoded efficiently and recovered incrementally during the progressive reconstruction of the original mesh. This process is detailed out in the following paragraph.

The series of vertex insertion operations, which reconstruct a given mesh, is computed by reversing a mesh decimation procedure [10]. Given a mesh M_i a simplification algorithm that iteratively removes sets of vertices u_i to yield a simplified version M_{i-1} is applied. However, at each iteration the selected set u_i must be an *independent set* [2], that is, there are no two vertices in u_i connected by a single edge. Removing a vertex from a triangulation requires removing all the edges connected to the vertex and retriangulating the hole with a new set of triangles. Let us define the triangles that cover a given hole as a *patch*. Once all the holes are triangulated, patches are interpolated to predict a set of points, which serves as a base for the displacement vector to the removed vertices. The predicted points are quantized, so the displacement vectors can be represented by a small number of bits, with smaller entropy than the original vertices. For each patch, one displacement vector is stored.

The key idea is to encode the triangles of a patch with colors, such that the decoder can detect the patches during the reconstruction stage based on the triangle colors. Thus, adjacent patches can-

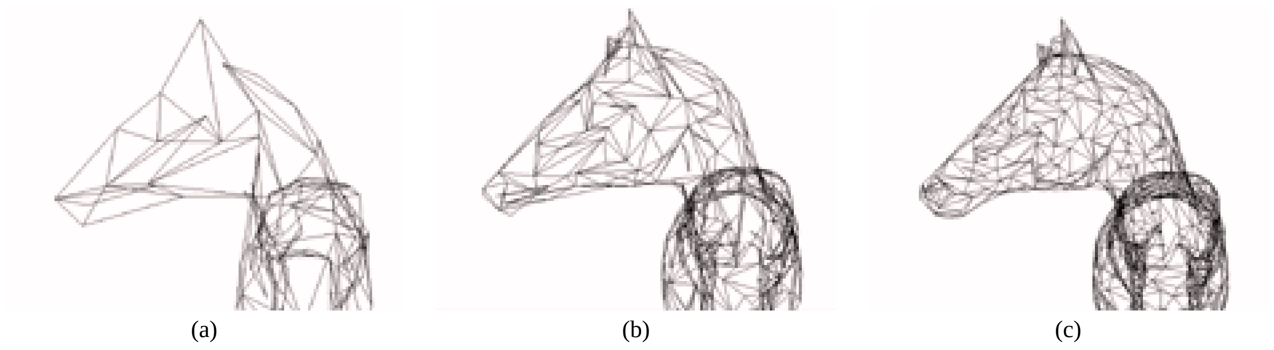


Figure 2: Three resolution Levels. (a) Lowest Resolution Level. (b) Medium Resolution Level, and (c) Highest Resolution Level,

not be assigned the same color, where two patches are said to be adjacent if they share an edge. The triangles of M_i are recursively traversed and each patch is assigned a color that is different from the colors assigned to its adjacent patches. Since the patches do not tessellate the entire mesh, a null color is used for the triangles that are not included in any patch. The rest of the triangles are colored in only three colors (see Figure 3). Three colors are not always enough, but in practice such cases are rare, and can be avoided by excluding the removal of some potential vertices.

This coloring technique requires 2 bits per triangle. Thus, the cost of encoding a vertex is the cost of coloring the triangles of the patch created by its removal. Assuming the degree of a vertex is 6, then its removal requires coloring four triangles, that is, 8 bits per vertex removal. Note that there is some overhead since some triangles are not included in any patch. We strive to create a maximal independent set, to reduce this overhead, when selecting the vertices to be removed.

During reconstruction, for each recovered patch, its triangles are removed, and the location of the vertex that was removed when the patch was created is predicted. By adding the associated displacement vector to the predicted point, the original location of the vertex is recovered. Then, the vertices of the patch are simply connected to the inserted vertex.

VRML Representation

VRML extensions that encode such a progressive LOD mesh are asked for. The VRML form described bellow allows for progressive transmission and provides a uniform VRML framework. Two new nodes, namely *ProgIndexedTriSet* and *ProgLOD,* are introduced.

The *ProgIndexedTriSet* node is an augmentation of the standard *IndexedFaceSet* node. It contains all the fields that an *IndexedFaceSet* node contains. Some of these fields provide general information regarding the mesh construction (e.g., *ccw*, *solid*, etc.). Others represent the initial lowest level triangulation, referred as level 0. Notice that the *convex* field is not included in a *ProgIndexedTriSet* node, as the constructed faces (polygons) must be triangles.

In addition, a *ProgIndexedTriSet* node contains a sequence of level of details. If the *levels* field is not empty, it consists of level of detail nodes, where each additional level is represented by a *ProgLOD* node.

A *ProgLOD* is a node formed by constructing new patches from existing triangles extracted from the previous level and corresponding vertices inserted into the newly formed patches. Each vertex is associated with a patch into which the vertex is inserted. Each patch consists of a number of triangles, which are encoded by a series of triangle indices.

A *ProgLOD* node contain a triangle index and a coordinate mandatory fields, and a color, a normal, and a texture-coordinate optional fields. The coordinate, color, normal, an texture-coordinate fields contain a coordinate, a color, a normal, and texture-coordinate nodes respectively. These nodes contain correction values and are used to construct the 3D vertices inserted into the newly formed patches. The triangle-index field defines the patches by indexing into the triangles of the previous level. An index of ”-1” indicates that the current patch has ended and the next one begins. The last patch may be (but does not have to be) followed by a ”-1” index.

The level-0 triangles indexed by the level-1 patches are explicitly defined according to their order in the *coordIndex* field of the *ProgIndexedTriSet* node. The triangles of levels higher than 0 indexed by patches of levels higher than 1 are deduced by a canonical breadth-first traversal of the triangle of the current evolved mesh.

The *levelsUrl* field contained by the *ProgIndexedTriSet* node provides an alternative method to specify the level of details. If the *levelsUrl* field is not empty, the *coord*, *color*, *normal*, *texCoord*, *coordIndex*, *colorIndex*, *normalIndex*, *texCoordIndex*, and em levels fields are ignored. The *levelsUrl* field contains a sequence of valid URLs. Each URL describes a file (located on a particular server and accessed through a specified protocol), typically binary, that represents a level of details starting with level 0.

A simple example using the prototyping mechanism is provided in Figures 4 and 5. The image in figure 3 is produced by the viewer when fed with the model represented in Figure 5.

4 Results

As discussed above, the color-encoding technique requires 2 bits per triangle. The cost of encoding a d-vertex is $2(d-2)$ bits, since the patch created by removing a d-degree vertex consists of $d-2$ triangles only. Thus, the removal of a 6-degree vertex requires 8 bits, and a 5-degree vertex only 6 bits. In [3] it was shown how these numbers can be further improved by a more elaborate color-encoding scheme in which the encoding of a vertex requires about 4-bits only. Since the independence set is not optimal, there are many triangles that are not included in any of the patches. Thus, in practice the cost is higher than 4 bits per vertex (see Table 1). In any case the stream of bits that encodes the mesh is further compressed by an LZ encoder.

Regarding the compression of geometry, the stream of the displacement is encoded using Huffman encoding. We have tested the results with 12-bit precision per coordinate. Table 2 compares our results with those of Touma and Gotsman's technique [12], which are the best published so far. It should be noted that the exact compression values depend on the specific implementation of the Huffman encoder. However, the combination of compression and progressiveness results with a unique technique.

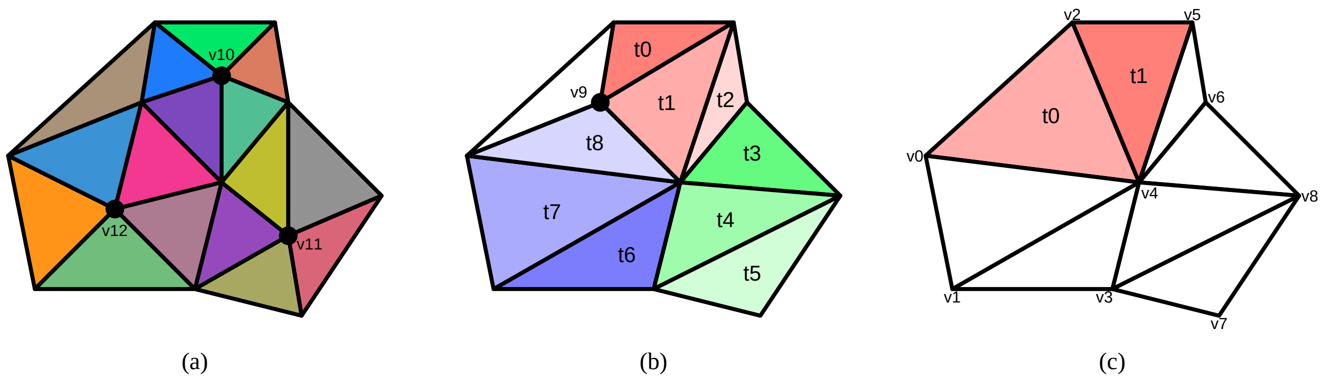


Figure 3: The coloring encoding scheme. (a) The original mesh consists of an arbitrary triangulation. The three black circles are the vertices selected for removal - the independent set. (b) The mesh after the removal of the black vertices and the triangulation of the holes. The new triangles are colored in three colors; the rest remains white. The black circle is the vertex selected for removal to create a lower level. (c) The lowest level after the removal of the selected vertex and the triangulation of the hole with two red triangles.

Table 1: *The models and the connectivity compression results. The first two columns show the number of vertices and triangles in each model. The third column shows the number of vertices removed. The fourth column the number of bytes of the connectivity stream, and the last column shows the number of bits per vertex. The average of the bit/vertex column is 5.98*

model	vertices	triangles	removed	connectivity	bits/vertex
horse	19851	39698	15494	11090	5.72
jaw	12349	24585	9453	6966	5.89
blob	8036	16068	6067	4507	5.94
holes3	5884	11776	4246	3526	6.64
tricerotops	2832	5660	2171	1550	5.71

```
#VRML V2.0 utf8

PROTO ProgIndexedTriSet[
  field         SFBool   ccw             TRUE
  field         SFBool   colorPerVertex  TRUE
  field         SFFloat  creaseAngle     0      # [0, ∞)
  field         SFBool   normalPerVertex TRUE
  field         SFBool   solid           TRUE

  exposedField  SFNode   coord           NULL
  exposedField  SFNode   color           NULL
  exposedField  SFNode   normal          NULL
  exposedField  SFNode   texCoord        NULL
  field         SFNode   coordIndex      NULL # [−1, ∞)
  field         SFNode   colorIndex      NULL # [−1, ∞)
  field         SFNode   normalIndex     NULL # [−1, ∞)
  field         SFNode   texCoordIndex   NULL # [−1, ∞)
  exposedField  MFNode   levels          []

  exposedField  MFUrl    levelsUrl       []
]
PROTO ProgLOD[
  field         MFInt32  triIndex        []     # [−1, ∞)
  exposedField  SFNode   coord           NULL
  exposedField  SFNode   color           NULL
  exposedField  SFNode   normal          NULL
  exposedField  SFNode   texCoord        NULL
]
```

Figure 4: The prototype definitions of the progressive LOD mesh representation in VRML.

```
ProgIndexedTriSet {
  levels [
    coord level0 {
      point[10 70, 20 20, 65 120, 80 20, 90 60,
            110 120, 115 90, 120 10, 150 55]
    }
    coordIndex[4 2 5 -1 0 2 4 -1 4 6 5 -1 4 8 6 -1
               3 8 4 -1 3 7 8 -1 1 3 4 -1 1 4 0 -1]
    ProgLOD {
      coord level1 {
        point[0 0]
      }
      triIndex[0 1 -1]
    }
    ProgLOD {
      coord level2 {
        point[0 0, 0 0, 0 0]
      }
      triIndex[0 1 2 -1 3 4 5 -1 6 7 8 -1]
    }
  ]
}
```

Figure 5: The usage of the progressive LOD mesh representation in VRML. Three LOD's comprise the 3d model. The z coordinates are omitted

Table 2: *Compression results. The same models as in Table 1 with 12 bits per coordinates precision. The second column shows the size of the stream and the base is the size of the base model compressed. Their sum is presented in the next column. The TG column shows the results of Touma and Gotsman's method. The right most column contains the compression ratio between TG and our progressive compression method. The average ratio is 1.08*

model	stream	base	total	TG	ratio
horse	42001	10445	52446	47108	1.11
jaw	29533	8090	37623	34577	1.08
blob	19928	5757	25685	21396	1.18
hole3	8182	4001	12183	13452	0.90
tricerotops	7252	2026	9278	7871	1.17

The above progressive technique has been integrated into a 3D content delivery system. The system accepts 3D content in various representations, an extended VRML being the most common, and translates it into an extended X3D representation off-line. In the final representation critical parts of the data (i.e., meshes and images) are in binary format. In Figure 6 four examples of 3D models that were designed and generated with the Form-Z modeler are shown. The models of the kettle and the ring consist of thousands of polygons and a number of textures, which are used as environment maps. Based on a wavelet decomposition our streaming system translates the input textures into a progressive hierarchy of images which are streamed along the geometry. The model of the Pokemon (see Figure 6(c)) was generated by scanning a physical toy using the 3D scanner of NexTec™.

5 Conclusions

We have presented a VRML/X3D-based representation that supports progressive and selective transmission of 3D scenes or parts of them, and achieves compression rates competitive with flat-compression rates. It includes a textual-format specification, and a compact, yet linear, binary format, used to represent critical sections. It is linear in the sense that it does not require any special streaming capabilities, and allows for quick reading and processing. On the server side, a standard HTTP service application (e.g., Apache) is adequate. On the client side, however, a special component that process the incoming streams must be present. The client component has been implemented as a plug-in to existing browsers and as a stand-alone application. It consists of a few threads that run in parallel; One thread reads the streams from the servers. Other threads decode the streams and produce usable data structures that represent mesh and texture-image hierarchies. Another thread consumes the data that has been produced so far and use it to render the scene.

As a result of employing asynchronous transmission of compact streams, the end-user is exposed to an immediate view of an approximation of the visible 3D models in the scene. While viewing and interacting with the scene, the rest of the data arrives asynchronously, and progressively refines the models in the scene up to the optimal level of details. We are currently working on extending the asynchronous transmission concept, and the file format that comes along, to support other free-from surface and animation sequence representations.

In Addition, we are further improving the process quality in various ways. One direction is to increase the size of the independent set of vertices to remove, while keeping the visual distortion minimal. Another direction is to triangulate the patches such that the visual shape of the resulting mesh resembles the original shape as much as possible.

Acknowledgments

We thank Ron Weitzman and Amir Shaham for their useful comments and suggestions. Also thanks to Gad Freedman and Arik Demi for helping producing the models and the images. The Flamingo model is courtesy of ViewPoint (www.viewpoint.com).

References

[1] Vadim Abadjev, Miguel del Rosario, Alexei Lebedev, Alexander Migdal, and Victor Paskhaver. Metastream. *VRML 99: Fourth Symposium on the Virtual Reality Modeling Language*, pages 53–62, February 1999. ISBN 1-58113-079-1. Held in Paderborn, Germany.

[2] M. De Berg and K. Dobrindt. On levels of detail in terrains. *Graphical Models and Image Processing*, 60:1–12, 1998.

[3] Daniel Cohen-Or, David Levin, and Offir Remez. Progressive compression of arbitrary triangular meshes. *IEEE Visualization '99*, pages 67–72, October 1999. ISBN 0-7803-5897-X. Held in San Francisco, California.

[4] Stefan Gumhold and Wolfgang Straßer. Real time compression of triangle mesh connectivity. In Michael Cohen, editor, *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 133–140. ACM SIGGRAPH, Addison Wesley, July 1998.

[5] Hugues Hoppe. Progressive meshes. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 99–108. ACM SIGGRAPH, Addison Wesley, August 1996.

[6] Martin Isenburg and Jack Snoeylink. Face fixer: Compressing polygon meshes with properties. *Proceedings of SIGGRAPH 2000*, pages 263–270, July 2000. ISBN 1-58113-208-5.

[7] Reinhard Klein. Multiresolution representations for surfaces meshes based on the vertex decimation method. *Computers & Graphics*, 22(1):13–26, February 1998.

[8] Renato Pajarola and Jarek Rossignac. Compressed progressive meshes. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):79–93, January - March 2000. ISSN 1077-2626.

[9] Jarek Rossignac. Edgebreaker: Connectivity compression for triangle meshes. *IEEE Transactions on Visualization and Computer Graphics*, 5(1):47–61, January - March 1999. ISSN 1077-2626.

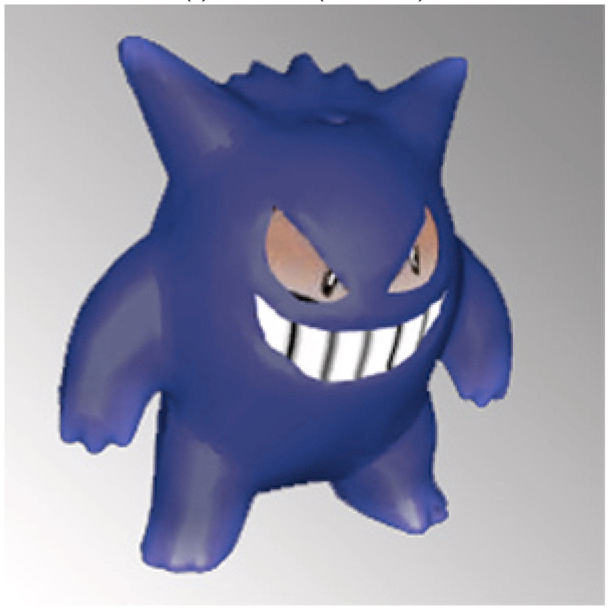
[10] William J. Schroeder, Jonathan A. Zarge, and William E. Lorensen. Decimation of triangle meshes. In Edwin E. Catmull, editor, *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 65–70, July 1992.

[11] Gabriel Taubin and Jarek Rossignac. Geometric compression through topological surgery. *ACM Transactions on Graphics*, 17(2):84–115, April 1998.

[12] C. Touma and C. Gotsman. Triangle mesh compression. *Graphics Interface'98*, pages 26–34, June 1998.

(a) Charizard (36K, 51K)

(b) Ring (15K, 42K)


(c) Gangar (12K, 43K)

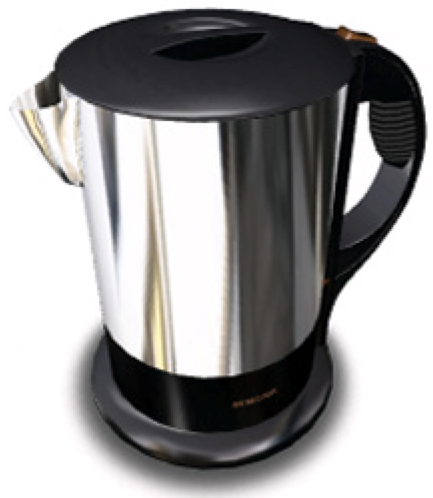

(d) Kettle (16K, 34K)

Figure 6: Four examples of 3D models; Their polygon count and their size, including the textures, in bytes are in parenthesis, respectively