Remote scientific visualization of progressive 3D meshes with X3D

Adrien Maglo[*]
MAS - Ecole Centrale Paris

Guillaume Lavoué[†]
Université de Lyon, CNRS
INSA-Lyon, LIRIS

Céline Hudelot[‡]
MAS - Ecole Centrale Paris

Ho Lee[§]
Universite de Lyon, CNRS
Université Lyon 1, LIRIS

Christophe Mouton[¶]
EDF R&D

Florent Dupont[||]
Université de Lyon, CNRS
Université Lyon 1, LIRIS

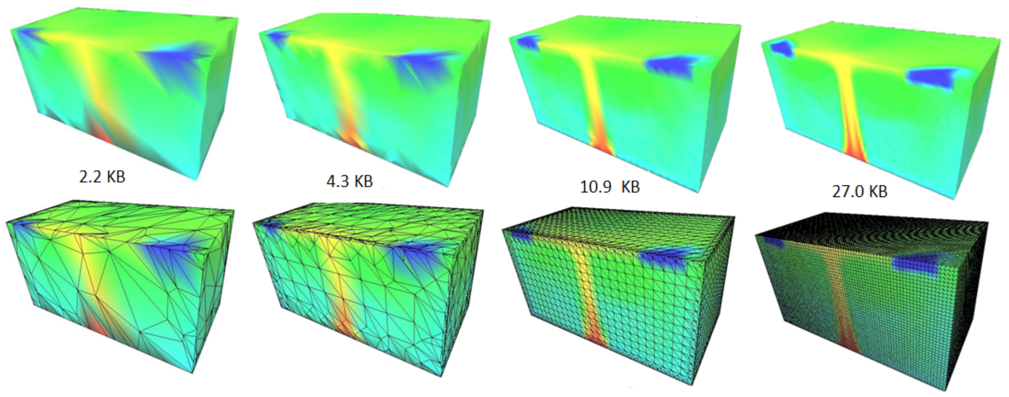

Figure 1: *Progressive decompression of the radiator model (16002 vertices). Its original X3D size is 953 KB.*

Abstract

This paper presents a framework, integrated into the X3D file format, for the streaming of 3D content in the context of remote scientific visualization; a progressive mesh compression method is proposed that can handle 3D objects associated with attributes like colors, while producing high quality intermediate Levels Of Detail (LOD). Efficient adaptation mechanisms are also proposed so as to optimize the LOD management of the 3D scene according to different constraints like the network bandwidth, the device graphic capability, the display resolution and the user preferences. Experiments demonstrate the efficiency of our approach in scientific visualization scenarii.

CR Categories: I.3.2 [Computer Graphics]: Graphics systems—Remote systems I.3.6 [Computer Graphics]: Methodology and Techniques—Graphics data structures and data types

Keywords: 3D adaptation, progressive meshes, scientific visualization.

1 Introduction

Increasing needs for precision and quality in scientific simulation lead to an important increase in the complexity of 3D data sets. For instance, the number of mesh elements used in finite element resolution methods can now reach about one billion. Moreover, these data are nowadays produced in high performance computing centers and have to be analysed by teams of scientists and engineers who are more and more often geographically distant. Nevertheless, even with the increase of available network bandwidth together with the available computational power resources, remote visualization of large 3D scientific datasets still suffers from several shortcomings among which the downloading time, the lack of interactivity and the management of different transmission, visualization and user constraints. Therefore, the idea of providing an efficient and fast remote access to large 3D scientific data has motivated three related research and engineering areas in these twenty last years: 3D mesh streaming, 3D mesh compression and 3D data adaptation.

3D mesh streaming can be defined as the continuous and real-time delivery of 3D content over network connections in order to: (1) enable **user interactions** without a full download of the data; (2) provide a **visual quality** as if the data were stored locally. 3D streaming approaches can be classified into three categories according to the nature of the data that are transmitted: image-based streaming, openGL- based streaming and geometry streaming which is by far the most relevant since it allows full 3D interactions. In particular, progressive geometry streaming enables an incremental rendering of the whole models with increasing details, hence it is directly related with progressive 3D compression. Non progressive geometry streaming [Isenburg and Lindstrom 2005] also exists however the objective is different, it aims at providing an efficient offline mesh format so as to facilitate out-of-core processings of gigantic meshes. In this paper, we only focus on progressive geometry streaming.

Mesh compression techniques can be distinguished in two categories: single-rate techniques, where the mesh data is compressed as a whole and can only be decompressed after receiving the entire

[*]e-mail: adrien.maglo@ecp.fr
[†]e-mail: guillaume.lavoue@liris.cnrs.fr
[‡]e-mail: celine.hudelot@ecp.fr
[§]e-mail: ho.lee@liris.cnrs.fr
[¶]e-mail: christophe.mouton@edf.fr
[||]e-mail: florent.dupont@liris.cnrs.fr

file, and progressive techniques, where the mesh is simplified in a set of levels of detail (LOD) from lossless to coarse, allowing the progressive decompression of the file. In other words, progressive compression permits to transmit different levels of detail (LOD) in a coarse-to-fine way. This functionality is particularly useful in the case of remote visualization since it allows to adapt the level of detail to the power of the client device, the network bandwidth and to the user needs.

3D data adaptation consists in automatically selecting the best set of 3D objects, and their optimal levels of detail in order to propose the best visualization experience to the remote user according to various network characteristics, device capabilities and user preferences. Such adaptation mechanism allows to accurately drive the decoding of some progressively compressed models so as to produce an optimal streaming of the 3D scene.

In this paper, we present a streaming framework for remote scientific visualization of 3D meshes from finite element computation methods. In our specific case, the targeted representations are surfacic and color values representing scalar fields are linked to the vertices of the meshes and are of prime importance for the visualization and the analysis. Our framework is based on a progressive compression scheme providing high quality intermediate color meshes and efficient adaptation mechanisms dealing with different constraints (network bandwidth, display capability, etc.). We provide also a solution to embed this framework into the X3D file format.

This paper is organized as follows: Section 2 details the existing work on progressive mesh compression, 3D streaming and adaptation, particularly in the context of the X3D file format. Then sections 3 and 4 respectively present our progressive encoding method, the proposed adaptation mechanisms and the integration of them in the X3D format. Finally Section 5 illustrates some results of our framework.

2 Previous work

2.1 Progressive Mesh Compression

We provide here a brief review of progressive compression techniques; an extensive state-of-the-art can be found in [Peng et al. 2005].

The concept of progressive compression was introduced for the first time by Hoppe [Hoppe 1996]; in his work, the mesh is iteratively simplified by performing a sequence of edge collapses, whose reverse operation is the vertex split. The edge collapse merges two connected vertices, together with their connectivity. The progressivity is achieved at a high cost, since the positions of the split vertices are explicitly encoded. A bunch of methods were then introduced (like [Taubin et al. 1998; Pajarola and Rossignac 2000]) so as to improve the compression ratio by applying the collapse/split operations on sets of independent vertices. Vertex decimation techniques further improved compression ratios. They consist in removing a vertex and retriangulating the hole left by the deletion at no cost [Cohen-Or et al. 1999; Alliez and Desbrun 2001].

All the progressive algorithms described above are connectivity-driven algorithms, meaning that the priority is given to the connectivity coding. Observing that the mesh's geometry data is larger than the connectivity data, more recent research has focused on geometry-driven compression algorithms. Gandoin and Devillers [Gandoin and Devillers 2002] proposed the first geometry-driven approach based on the kd-tree space subdivision. Then Peng and Kuo [Peng and Kuo 2005] proposed a more efficient geometry-guided technique by using the octree cell subdivision. These

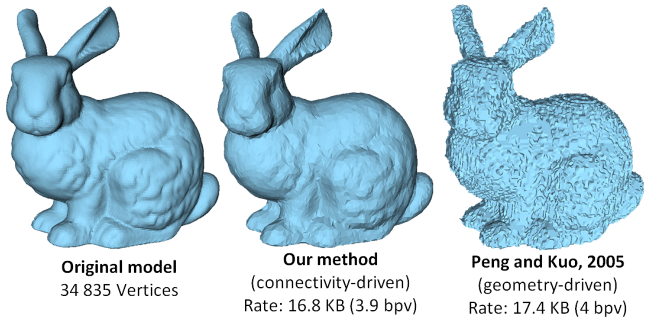

Figure 2: *Original Bunny model and results of progressive decoding at similar bitrates for our algorithm and Peng and Kuo's one.*

geometry-driven algorithms outperform connectivity-driven algorithms in terms of lossless compression ratio, however they provide quite poor results at intermediate resolutions, hence they are not fully efficient for progressive transmission. This is clearly illustrated in figure 2 which illustrates two intermediate levels of detail corresponding to both kinds of algorithms and for the same bitrate ($\approx$ 4 bits-per-vertex). The connectivity-driven algorithm provides a much nicer result.

The compression of mesh associated properties, like colors, plays a secondary role in state of the art compression schemes. However, mesh properties can often be of greater size than the other data flows of a mesh and can carry a great part of the relevant information, in particular in scientific visualization. Basically very few progressive mesh coders allow the encoding of color information; Cai et al. [Cai et al. 2007] and Cirio et al. [Cirio et al. 2010] recently proposed two geometry-based methods (respectively based on octree and kd-tree decompositions) that handle colors; however these methods are limited by the poor quality of the intermediate models.

2.2 Progressive mesh compression and X3D

As X3D is the ISO standard XML-based file format for 3D data delivery, mechanisms have been proposed in X3D to take into account progressive mesh compression and streaming. Works addressing these topics have also been proposed in the context of MPEG-4. First, as it is not always needed to render the finest representation of an object, particularly when it is far from the user viewpoint, the X3D specification includes an adaptation mechanism called LOD[1]. This mechanism enables the manual specification, by the scene designer, of multiple alternatives (e.g. for instance the entire mesh in different level of resolutions) for an object together with a set of distance ranges indicating when alternatives have to be rendered. In [Pasman and Jansen 2002], a similar idea has been proposed based on imposters, accuracy curves describing the minimal amount of resources required to reach an accuracy and a slight extension of the VRML node specification.

These X3D/VRML specifications do not fit yet to the constraints and challenges of remote scientific visualization: (1) the proposed specifications need the complete description of each alternative and thus does not enable progressive representation; (2) the specifications do not define strategies related to the transmission and streaming part, i.e. intelligent transmission of the different alternatives; (3) the LOD techniques are more bounded to the 3D scene and not to the interaction of the user.

[1]`http://www.web3d.org/x3d/specifications/ISO-IEC-19775-1.2-X3D-AbstractSpecification/index.html`

Related to the first issue, approaches based on a client-server architecture have been proposed [Fogel et al. 2001; Guéziec et al. 1998]. In [Guéziec et al. 1998], the authors propose their own progressive mesh representation where an external file, downloaded on demand, holds the LOD data. [Fogel et al. 2001] extends the VRML/X3D language with two nodes: *progIndexedTriSet* which extends the node *IndexedFaceSet* and *ProgLOD* which holds the LOD data (triangle index, coordinates, normal, etc.). An alternative is also the specification of URLs of external files which contain the different binary compressed representations of the data. Both approaches do not specify how to use the progressive representation for progressive download and rendering.

Related to the other issues, the basic transmission strategy consists in downloading the entire scene and in rendering it. In [Arikawa and al 1996], this issue is tackled by the proposition of a dynamic LOD VRML extension and a client server architecture enabling user-dependent LOD streaming and rendering. Nevertheless, this approach does not use progressive representation of meshes. Besides, it is much designed for remote walkthrough and not for remote scientific visualization.

2.3 3D streaming and adaptation

Two kinds of approaches exist related to adaptive 3D streaming. On the one hand, works have been proposed to stream efficiently progressive meshes with a view-dependent way [Southern et al. 2001; Cheng and Ooi 2008], by optimizing the rendering perception [Chen and Nishita 2002] or packetization [Cheng et al. 2007]. On the other hand, the rendering of progressive 3D models taking into account device capabilities and user perception have also been studied in [Funkhouser and Séquin 1993; Gobbetti and Bouvier 1999; Pasman and Jansen 2002]. However, these two problems have not been much studied together, i.e. for the case when 3D models are rendered over networks. The framework built in [Jessl et al. 2005] delivers progressive meshes over networks using subdivision-surface wavelets. Fixed thresholds, related to the transmission and the visualization, are used to drive the transmission. In [Schneider and Martin 1999] the authors propose a client server architecture to deliver 3D model at interactive frame rates over networks. Their approach optimizes trade-offs between all the constraints (networks, device capabilities, user preferences). They use benchmarks to measure the environment characteristics. Martin [Martin 2000] describes a framework dedicated to the automatic selection of the best representation modality for objects in a scene. Their performance model takes into account, for each modality, the estimated delivery time, the quality of the representation and the degree of interaction it supports. In [Deb and Narayanan 2004], an adaptive framework for remote walkthrough applications is proposed. In [Ngoc et al. 2002], a QoS framework for the delivery of 3d content over network is built. The authors define a benefit, cost optimization problem solved by using heuristics. PSNR is used as the quality metric of rendered models. As far as we know, no work have been proposed addressing the particular case of adaptive remote 3D scientific visualization.

3 Progressive encoding method

Our progressive encoding algorithm is described in detail in [Lee et al. 2010]. It is based on the valence-driven progressive approach proposed by Alliez and Desbrun [Alliez and Desbrun 2001]. This approach iteratively decimates a set of vertices by combining decimation and cleansing conquests to get different levels of detail (LOD). Decimation conquest consists in removing vertices patch by patch using a gate-based traversal, and then retriangulating the holes left (see fig.3.b); then, a cleansing conquest removes vertices of valence 3 so as to regularize the simplified mesh (see fig.3.c). Our method [Lee et al. 2010] extends the base algorithm from Alliez and Desbrun [Alliez and Desbrun 2001] in two ways: first, instead of applying a global and uniform quantization to the coordinates of the mesh vertices before starting the decimation, our method automatically adapts the quantization for each intermediate mesh so as to obtain a better rate-distorsion performance (more details are given in [Lee et al. 2009]); second, our method is able to handle colors associated to vertices by introducing an accurate prediction scheme. Hence during encoding, the mesh is iteratively simplified (decimation + cleansing) and at each simplification step, the connectivity, geometry and color information necessary for the inverse operation (i.e. refinement) are encoded; for the connectivity, only the valences of the removed vertices are necessary, and for geometry and color, the 3D position and the color of each removed vertex is predicted according to its neighbors and only the residues are written in the stream. Figure 4 presents the compressed stream; this stream is naturally decomposed into several parts, each standing for a certain level of detail and each containing connectivity (C), geometry (G) and color (Cl) information. The first part is the base mesh (usually several dozens of vertices) which is encoded using a standard mono-resolution compression technique; then each part of the stream, together with the already decompressed level, allows to build the next level of detail. At the end of the stream decoding the original object is retrieved (lossless compression).

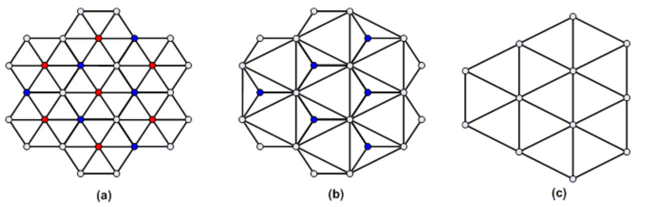

Figure 3: *One iteration of the progressive encoding algorithm. (a) original mesh, (b) intermediate result after decimation and (c) final result after cleansing.*

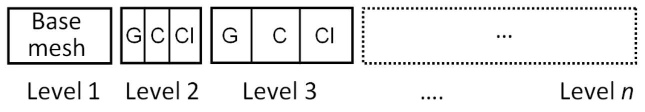

Figure 4: *Format of the encoded stream. Each part of the stream contains geometry (G), connectivity (C) and color (Cl) data needed for mesh refinement.*

4 LOD adaptation and streaming framework with X3D

In this section, we define our adaptation framework for 3D scientific visualization over networks. We use a basic client-server architecture.

4.1 The adaptation parameters

Our adaptation framework takes into account constraints coming from the network, the device graphic capabilities and the user viewpoint and preferences. The algorithm aims to maintain the following metrics below their threshold (experimentally set for a first purpose):

- $T_{dl}(M)$, the total time spent to download successively levels of a mesh M. The download of new level is halted when

$T_{dl}(M)$ overtakes T_{dlmax}.

- Q, the total quantity of memory taken by all the LOD data. New level downloading is stopped when Q is above Q_{max}.
- T_r, the rendering time of an image which measures the difficulty for a device to render a 3D scene. If T_r is above the threshold T_{rmax}, then the scene complexity must be reduced.
- $F_{ED}(M_i, R, P)$, the element distinction metric, where i is the current level index of M. As the visualization devices have not the same resolution, our adaptation framework tries to find the best level of a mesh M to render for a given viewpoint P and a screen resolution R. F_{ED} represents the ability for a user to well distinguish the mesh cells. We compute F_{ED} by randomly extracting a set of triangles of M_i (about 10%) and using the following formula:

$$F_{ED}(M_i, R, P) = \frac{n_1}{n_2}$$

where n_1 is the number of pixels of the lines when the triangles are rendered in line mode and n_2 is the number of pixels of the triangles when they are rendered in filled mode. These two renderings are done with no lighting and anti-aliasing filters and with uniform colors distinct from the background (see Figure 5).

If F_{ED} is near 0, it means that the mesh elements can be well distinguished in the screen space. On the contrary, the more it is far from 0 the more the elements are small and cannot be easily distinguished. F_{ED} has to stay below F_{EDmax}.
The size of the randomly extracted triangle set is important. It sets the accuracy of the metric for the current viewpoint. Indeed, if all the faces are rendered with a depth test, only the visible faces are taken into account in the metric computation. However, this is done at the expense of the computation time. As the following section describes it, we benefit from the inactivity of the computational resources, when the user has found the viewpoint he is interested in, to compute this metric and ask, if needed, for refinements. In our current implementation, we extract 10% of the triangles to compute F_{ED}. For the highest levels, the computation time is approximately half of the rendering time.

4.2 The adaptation algorithm

Our adaptation framework is view-dependent and triggered by events. An event can be: an addition of a new mesh into the scene, a first rendering after a mesh refinement, a first rendering after a mesh coarsening, a changing visibility of one object or a changing view-point. Note that the adaptation algorithm is run when the user has found his interest view-point and does not move anymore. The adaptation algorithm computation time does not impact the frame rate during transient moves.
L is the set of the meshes from the scene that the user has selected to be visible. After each event the adaptation algorithm is launched, it basically chooses among these three choices:

- do nothing,
- if $T_r > T_{rmax} * (1 + r)$, **coarse** the mesh M of L with the highest $F_{ED}(M_i, R, P)$,
- if $T_r < T_{rmax}$, **refine** a mesh M of L given by the findMesh() function.

A hysteresis thresholding (introduced by the r parameter experimentally set) avoids jumps between levels because of T_r varying arround T_{rmax}. It allows T_r to overtake T_{rmax} without spawning detail reduction.

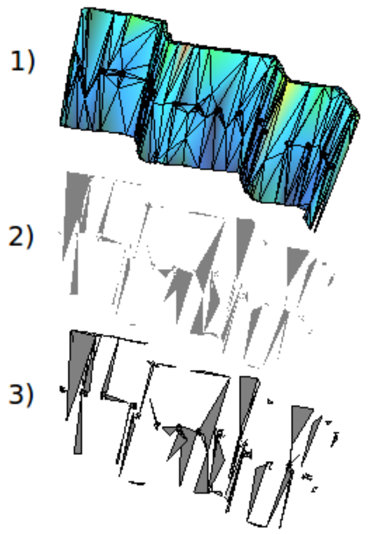

Figure 5: *Computation of metric F_{ED}. 1) The object with its visualization attributes is shown. 2) 10% of the triangles are rendered in filled mode. n_2 is obtained by counting grey pixels. 3) The same triangles are rendered in line mode. n_1 is obtained by counting black pixels.*

```
if $T_r < T_{rmax}$ then
    M = findMesh()
    if $M$ exists then
        if $M_{i+1}$ is not in memory then
            start downloading $M_{i+1}$ and store it in memory
        end if
        replace $M_i$ by $M_{i+1}$
    end if
else if $T_r > T_{rmax} * (1 + r)$ then
    select the visible mesh $M$ with the highest $F_{ED}(M_i, R, P)$
    replace $M_i$ by $M_{i-1}$
end if
```

Figure 6: *The adaptation algorithm. The first rendering after a mesh refining or coarsening triggers a new run of the adaptation procedure.*

The findMesh() function returns the mesh with the minimal $F_{ED}(M_i, R, P)$ which satisfies all the following constraints:

- $Q < Q_{max}$,
- $T_{dl}(M) < T_{dlmax}$,
- and $F_{ED}(M_i, R, P) < F_{EDmax}$.

Figure 6 presents our whole adaptation algorithm.

As the user may be interested in visualizing the highest level of detail of a certain mesh M of the scene, he can force the rendering of its highest level M_n even if it can not be achieved at an interactive frame rate. In this case, all the levels are downloaded and M_n is rendered without taking into account the results of the previous adaptation algorithm.

4.3 Streaming LOD in X3D

At the client side, the resources in terms of network bandwidth to the Internet, device capabilities (CPU, RAM, graphic computational power, etc.) are limited while the resources at the server side

```
<ProgrTriSet>
    <ProgrLOD level="0"
        url="http://collaviz.org/lod0.ps" />
    <ProgrLOD level="1"
        url="http://collaviz.org/lod1.ps" />
    <ProgrLOD level="2"
        url="http://collaviz.org/lod2.ps" />
    <ProgrLOD level="3"
        url="http://collaviz.org/lod3.ps" />
    ...
</ProgrTriSet>
```

Figure 7: *The integration of our progressive compressed remote mesh representation in X3D. '.ps' files correspond to the binary LOD data generated by our progressive mesh encoding scheme.*

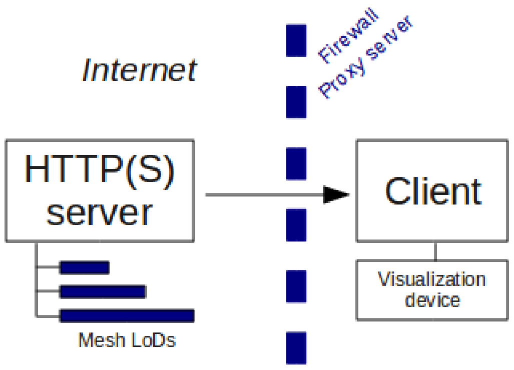

Figure 8: *Our streaming architecture for progressive 3D meshes. Using the HTTP(S) protocol, we are able to pass through most of proxy servers and firewalls.*

can be managed. Therefore, we use a client-server architecture to stream the progressive meshes. We integrate our framework to the X3D language with the same approach as in [Fogel et al. 2001]. The 3D scene is described in a X3D file which contains a $ProgrTriSet$ node for each object. $ProgrLOD$ nodes receive the URL indicating where to download the LOD data (see Figure 7). When a client requests a mesh, it receives this X3D file. Then, it can start downloading the levels according to the previously described adaptation strategy.

Our client asks for new levels of detail using HTTP requests. The server responds by delivering the mesh LOD data that have been already computed off-line by our progressive compression scheme. Once the client has received a new level, it restarts the adaptation procedure in order to maximize the user experience and maintain the metrics below their thresholds. Using the HTTP(S) protocol with its standard TCP/IP port, the proposed traffic can pass through most of the firewalls and proxy servers deployed in companies for their connection to the Internet.

5 Experiments

5.1 The progressive encoding scheme

Our progressive compression method was tested on several 3D models coming from post-processings (cutting-plane, isosurfacing) applied on volumetric scientific simulation results (Fluid Dynamics and geophysics). These six models (see figure 9) are X3D ASCII files and all contain color information on vertices except radiator_Iso.

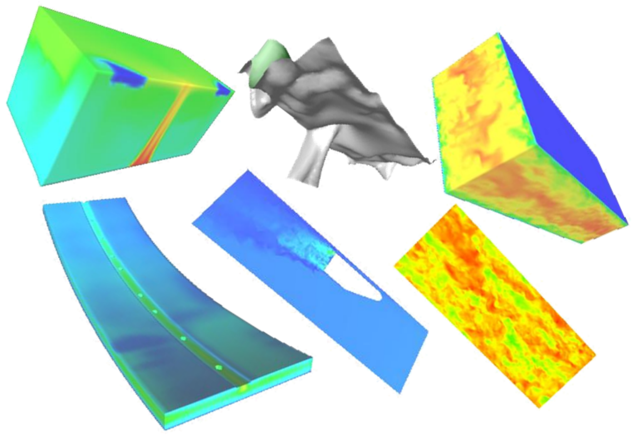

Figure 9: *Our corpus of 3D models. first row: radiator, radiator_Iso and velocity. Second row: tank, vistcurb_CP and velocity_CP.*

Name	Original (KB)	Compressed (KB)	Ratio
velocity_CP	7739	227.9	2.94%
velocity	2979	83.4	2.80%
radiator	953	27.0	2.83%
radiator_Iso	713	19.6	2.75%
vistcurb_CP	354	16.0	4.51%
tank	10065	266.0	2.64%

Table 1: *Lossless compression results for several objects from scientific simulations.*

Table 1 presents respectively the original sizes of the models (X3D ASCII file format) and the sizes of the compressed streams corresponding to a lossless compression (all levels of detail) with 11 bits precision. The compression ratios are very good (between 2.75% and 4.51%); for comparison a binary encoding method like gzip provide compression ratios between 15% and 20%. The whole decompression times (all levels) are between 0.14 and 6,015 seconds, respectively for *vistcurb* (4.3Kvertices) and *tank* (123Kvertices) on a 2Ghz processor. The number of levels is automatically determined so as to obtain a base mesh of around a hundred vertices, it goes from 10 (*vistcurb*) to 26 (*tank*); this number of levels depends on both the number of vertices and the shape complexity of the model. Our method presents a very good compression performance but its very strong point is its capability to produce high quality intermediate levels of detail.

Figure 1 illustrates several levels of detail from the progressive decoding of the *radiator* model; even after decoding only a very small amount of data (eg. 4.3KB, corresponding to a decompression time of 30ms) the resulting models are nevertheless visually correct allowing the client to be able to visualize a nice model even through a very low bandwidth channel.
These levels of detail also allow to adapt the resolution of the scene to the processing capacity of the display device; figure 10 illustrates several levels of detail of the *radiator_Iso* model. If only a small part of the stream is decoded then the resulting model owns only a small number of triangles allowing an efficient rendering even on low capacity devices. As it is shown in the next experiments, these properties combined to our adaptation algorithm will allow a very efficient streaming framework.

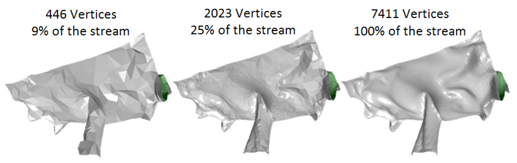

Figure 10: *Progressive decompression of the radiator_Iso model.*

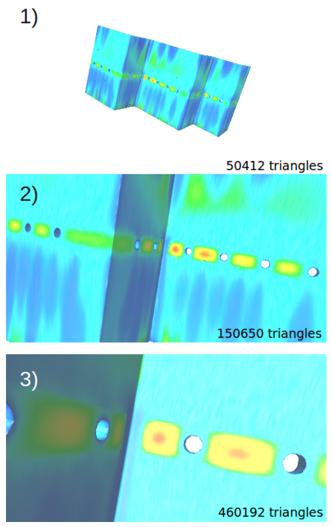

Figure 11: *Snapshots taken during our test of progressive download and rendering of our model with different viewpoints. Our adaptation framework adapts the LOD to the viewpoint.*

5.2 The whole streaming framework

We have tested our whole framework by simulating a remote scientific visualization session with a mesh coming from a real study case [Rupp I. Oct. 2008] of 460192 facets. All the LOD data were stored on an HTTP server. Our adaptation framework has successively downloaded and rendered them. We did two experiments which are described below.

The first experiment consisted in studying the effects of our adaptation framework when the user changed the view point. After each change, we waited for the data requested by our adaptation algorithm to be completely downloaded before taking snapshots of the new rendering. Then, we compared these snapshots with their equivalent taken with the full model by computing the PSNR. The results of this experiment are presented in table 2. Snapshots of the viewpoints used during the experiment can be seen on Figure 11. The experiment ran on a station with 2,8 Ghz processor and a NVIDIA Quadro FX 580. We computed the F_{ED} metrics by extracting 10% of the triangles for each level; it took a maximum time of 110ms for the finest level of detail.
Results demonstrate that our framework permits to achieve a good visualization quality and interactive frame rates for a global viewpoint by downloading a low amount of data (see Figure 11.1). Indeed for this level of detail only 27% of the data need to be downloaded and only 50K vertices have to be displayed. Then when the user zooms on a specific part of the model, more data are downloaded and the LOD increases.

The second experiment consisted in studying the effects of our adaptation framework when considering a single viewpoint (see figure 12) but using three different screen resolutions: one of a mobile device (640x480), one of a standard notebook computer (1280x800) and one (1882x932) which can be reached on a powerful graphic station with a 24 inches monitor. For each screen resolution, our algorithm automatically adapted the LOD of the 3D model. Like in the previous experiment we compared, using the PSNR, the snapshots obtained with the LOD selected by the framework with the snapshots taken with the full mesh; the results are given in table 3 and attest the efficiency of our adaptation algorithm since the PSNR values remain quite high.

Point of view (figure 11)	1	2	3
Level of detail	29/34	32/34	34/34
Nb of triangles	50412	150650	460192
% of downloaded data	26.6	52.1	100
T_r (ms)	45	230	266
PSNR (dB)	35.83	37.80	-

Table 2: *Results of our test of progressive download and rendering of our model with different viewpoints with* $F_{EDmax} = 1.8$, $T_{rmax} = 200$ *and* $r = 0, 5$.

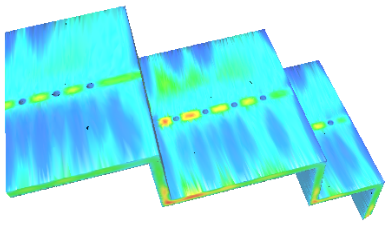

Figure 12: *Snapshot of the viewpoint used during our test of progressive download and rendering of our model with different screen resolutions. Our adaptation framework adapts the LOD to the resolution.*

Resolution	640x480	1280x800	1882x932
Level of detail	30/34	32/34	33/34
Nb of triangles	70862	150650	238720
% of downloaded data	31.5	52.1	70.1
PSNR (dB)	35.61	40.64	44.13

Table 3: *Results of our test of progressive download and rendering of our model with a common viewpoint and different screen resolutions with* $F_{EDmax} = 1.8$, $T_{rmax} = 200$ *and* $r = 0, 5$.

6 Conclusion

In this paper, we have proposed an adaptive and progressive streaming framework dedicated to remote scientific visualization. Targeted data are surfacic representations of 3D meshes computed with finite elements methods. Due to the importance of the scalar fields linked to the vertices of the meshes for the visualization and the analysis, we have first proposed a progressive encoding method which handles adaptive quantization and colors. Then a framework based on efficient adaptation mechanisms has been proposed to tackle different constraints: network bandwidth, device capability and user preferences. Based on previous works on X3D, our proposed framework is also X3D-compliant. Our first results on 3D scientific models are encouraging. Future work will be dedicated to the improvement of the adaptation part by taking into account more complex constraints.

7 Acknowledgement

This work has been founded by French National Research Agency (ANR) through COSINUS program (project COLLAVIZ n°ANR-08-COSI-003).

References

ALLIEZ, P., AND DESBRUN, M. 2001. Progressive encoding for lossless transmission of 3D meshes. In *ACM Siggraph*, 198–205.

ARIKAWA, M., AND AL. 1996. Dynamic lod for qos management in the next generation vrml. In *ICMCS '96: Proceedings of the 1996 International Conference on Multimedia Computing and Systems*, IEEE Computer Society, Washington, DC, USA, 24–27.

CAI, S., QI, Y., AND SHEN, X. 2007. 3d data codec and transmission over the internet. In *Web3D '07: Proceedings of the twelfth international conference on 3D web technology*, ACM, New York, NY, USA, 53–56.

CHEN, B.-Y., AND NISHITA, T. 2002. Multiresolution streaming mesh with shape preserving and qos-like controlling. In *Web3D '02: Proceedings of the seventh international conference on 3D Web technology*, ACM, New York, NY, USA, 35–42.

CHENG, W., AND OOI, W. T. 2008. Receiver-driven view-dependent streaming of progressive mesh. In *NOSSDAV '08: Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, ACM, New York, NY, USA, 9–14.

CHENG, W., OOI, W. T., MONDET, S., GRIGORAS, R., AND MORIN, G. 2007. An analytical model for progressive mesh streaming. In *MULTIMEDIA '07: Proceedings of the 15th international conference on Multimedia*, ACM, New York, NY, USA, 737–746.

CIRIO, G., LAVOUE, G., AND DUPONT, F. 2010. A framework for data-driven progressive mesh compression. In *International Conference on Computer Graphics Theory and Applications (GRAPP)*, Lecture Notes on Computer Science, Springer.

COHEN-OR, D., LEVIN, D., AND REMEZ, O. 1999. Progressive compression of arbitrary triangular meshes. In *IEEE Visualization*, 67–72.

DEB, S., AND NARAYANAN, P. 2004. Design of a geometry streaming system. In *Proc. ICVGIP*, Citeseer, 296–301.

FOGEL, E., COHEN-OR, D., IRONI, R., AND ZVI, T. 2001. A web architecture for progressive delivery of 3d content. In *Web3D '01: Proceedings of the sixth international conference on 3D Web technology*, ACM, New York, NY, USA, 35–41.

FUNKHOUSER, T. A., AND SÉQUIN, C. H. 1993. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 247–254.

GANDOIN, P.-M., AND DEVILLERS, O. 2002. Progressive lossless compression of arbitrary simplicial complexes. In *ACM Siggraph*, 372–379.

GOBBETTI, E., AND BOUVIER, E. 1999. Time-critical multiresolution scene rendering. In *VIS '99: Proceedings of the conference on Visualization '99*, IEEE Computer Society Press, Los Alamitos, CA, USA, 123–130.

GUÉZIEC, A., TAUBIN, G., LAZARUS, F., AND HORN, W. 1998. Simplicial maps for progressive transmission of polygonal surfaces. In *VRML '98: Proceedings of the third symposium on Virtual reality modeling language*, ACM, New York, NY, USA, 25–31.

HOPPE, H. 1996. Progressive meshes. In *ACM Siggraph*, 99–108.

ISENBURG, M., AND LINDSTROM, P. 2005. Streaming meshes. *IEEE Visualization, 2005. VIS 05*, 231–238.

JESSL, J., BERTRAM, M., AND HAGEN, H. 2005. Web-based progressive geometry transmission using subdivision-surface wavelets. In *Web3D '05: Proceedings of the tenth international conference on 3D Web technology*, ACM, New York, NY, USA, 29–35.

LEE, H., LAVOUÉ, G., AND DUPONT, F. 2009. Adaptive coarse-to-fine quantization for optimizing rate-distortion of progressive mesh compression. In *Vision, Modeling, and Visualization Workshop*.

LEE, H., LAVOUÉ, G., AND DUPONT, F. 2010. New methods for progressive compression of colored 3D Mesh. In *International Conference on Computer Graphics, Visualization and Computer Vision (WSCG)*.

MARTIN, I. 2000. Adaptive rendering of 3d models over networks using multiple modalities. *IBM research report*.

NGOC, N., VAN RAEMDONCK, W., LAFRUIT, G., DECONINCK, G., AND LAUWEREINS, R. 2002. A qos framework for interactive 3d applications. In *The 10-th International Conference on Computer Graphics and Visualization '2002*, Citeseer, 317–324.

PAJAROLA, R., AND ROSSIGNAC, J. 2000. Compressed progressive meshes. *IEEE Visualization and Computer Graphics 6*, 1, 79–93.

PASMAN, W., AND JANSEN, F. W. 2002. Scheduling level of detail with guaranteed quality and cost. In *Web3D '02: Proceedings of the seventh international conference on 3D Web technology*, ACM, New York, NY, USA, 43–51.

PENG, J., AND KUO, C.-C. J. 2005. Geometry-guided progressive lossless 3d mesh coding with octree (ot) decomposition. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, ACM, New York, NY, USA, 609–616.

PENG, J., KIM, C.-S., AND KUO, C.-C. J. 2005. Technologies for 3D mesh compression: A survey. *Journal of Visual Communication and Image Representation 16*, 6, 688–733.

RUPP I., PENIGUEL C., T.-M. M. Oct. 2008. Large scale finite element thermal analysis of bolts of a french pwr core internal baffle structure. In *7th I. Topical Meeting on Nuclear Reactor Thermal Hydraulics, Operation and Safety NUTHOS-7*.

SCHNEIDER, B., AND MARTIN, I. 1999. An adaptive framework for 3D graphics over networks. *Computers & Graphics 23*, 6, 867–874.

SOUTHERN, R., PERKINS, S., STEYN, B., MULLER, A., MARAIS, P., AND BLAKE, E. 2001. A stateless client for progressive view-dependent transmission. In *Web3D '01: Proceedings of the sixth international conference on 3D Web technology*, ACM, New York, NY, USA, 43–50.

TAUBIN, G., GUÉZIEC, A., HORN, W., AND LAZARUS, F. 1998. Progressive forest split compression. In *ACM Siggraph*, 123–132.