3D Visualization of particle system with extracted data from sensor

ABSTRACT

This paper deals with 3D visualization and interaction techniques used to analyze data necessary to manage energy consumption. Data comes from sensors located in a green data center in the IBM Montpellier (France) site. The main goal of this work is to visualize, in real-time, the large amount of data produced by different sensors in the data center. A visualization engine has been developed to understand the data extracted from sensors. This engine is based on a particle system which allows representation of the room's interior.
Data centers are large warehouse servers, which represent a gap in building's energy consumption. In this paper we present a visual method developed to improve energy management of data center by supporting IT architects activities. Our system use sensors to monitor data center, and allows visualizing temperature air flow. We use 3D models and several multiresolution visualization methods to allow our system to be deployed in different devices.

Keywords: 3D Visualization, Sensors, Particles, Data center, Level Of Details

1. INTRODUCTION

In this paper, we present a method to produce a 3D visualization for analyzing and managing temperature. Our project is a part of RIDER (Research for IT Driven Energy Efficiency) project who aims to produce a green box to manage energy consumption. This scalable box will be used in different kinds of buildings: flats, data centers, theatres and commercial buildings. This box is designed around advice systems, see Figure 1. Several sources compose inputs of the box. Building Management System (BMS) provides data from sensors located in the building and external data are sent from providers like Météo France. A rule system manages the green box and defines user or administrator parameters. Each rule or advice change will affect all system, and new information is broadcasted to each advice system.

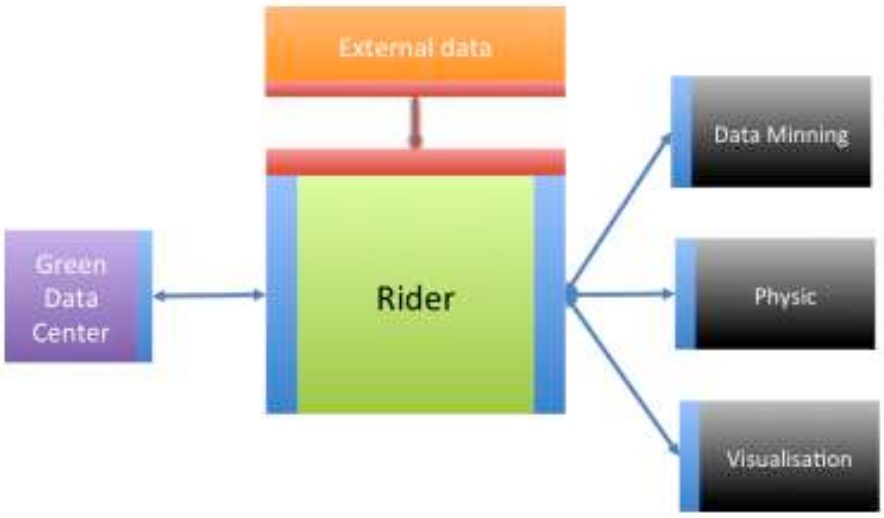


Figure 1. RIDER overview, Purple box represents BMS system, Orange box represents external data and black boxes are for decisonal engine

Our sandbox data come from sensors located in the IBM Green Data Center of Montpellier, France. Sensors are timed vector composed of type, value, time and location. They measure temperature, hygrometry, pressure and UPS (Uninterruptible Power Supply). In our system, sensors are placed in a virtual room and the internal space is modeled using particles. The main constraint here is to produce a real-time rendering because latency appears due to the number of vertices. To deal with this problem, we use LOD (Level Of Detail) to produce multi resolution 3D objects and allow interactive rendering. This solution has been introduced in 1976 by J. Clark (Clark 1976). In his paper, J. Clark introduces the use of several mesh resolutions to simplify the 3D scene complexity. In this paper, we describe our data center 3D model and the methods used to produce visualizations at different resolutions. Our goal is to produce a low network consomption protocol to transmit data, in order to allow visualization in mobile devices.

In Section 2, we introduce related work on visualization, particles systems and LOD. In Section 3, we expose our solution to simplify particles system. In Section 4 we give some results and finally, in Section 5 we present our conclusions and future work.

2. RELATED WORK

In this section we present relevant previous works concerning data visualization, particle systems and level of detail methods.

Some previous work present solutions to visualize large data flow extracted from mantle convection. M. Damon et al. (Damon et al., 2008) and K. E. Jordan et al. (Jordan et al.,1996) present interactive viewers for this kind of data. These data are computed by using Hight Performance Computing (HPC) and visualized on a large display. The rendering is calculated by using another HPC. The data flow is very important and a real-time 3D simulation is hard to obtain.

Particles systems are introduced in 1960. They were used in the earlier days to simulate 2D pixel cloud (Reeves, 1983). In 2004, L. Latta (Latta, 2004) presents a solution to compute a particle system using a Graphical Processor Unit (GPU). There are few possible options for simulating a particle system. For example, some particle systems can be entirely stateless, they depend only on their initial values and the elapsed time. Stateless particles are based on mathematical equation. It was used for rendering fireworks, explosion. Statefull methods produce motion for each particles. They know their neighbors and their own location in space. However, in many cases, using ordinary computing styles has some limitation for rendering particles, they cannot compute in real-time a large number of them. GPU provides a new way to solve this issue by computing data in a parallel way.

W. Kapferer and T. Riser (Kapferer and Riser, 2008) introduce how to use particle system to visualize astronomic simulation, particles representing space objects. The number of particles is extremely important for computing motion in real-time. GPU computing is preferred to render instead of a common HPC solution. To display their data, they have developed their own 3D graphical engine. The space objects are represented by point sprite instead of sphere. Lights are used to give a spherical aspect to the point sprite. This solution allows to render more stars than spherical object method. The 3D engine provides different rendering methods to group space objects: cell simplification or extraction of isosurface. The use of GPU seems quite well for a particle solution, parallel processing allows to render large data; the astrological data seems to be well suited.

In 1976, J. Clark introduces Level Of Detail (LOD) concept (Clark 1976). LOD consists to produce several resolution meshes for using them at different distance from the camera. Firstly, designer produces these meshes. First algorithms, in 1992 Schroeder et al. developed a method by decimation for simplify the mesh[?]. It analyses mesh geometry and evaluates the complexity of triangles. Vertices are removed if only constraints set by the user are respected. Vertices are removed and gaps are filled using triangulation.

These algorithms of simplification are not enough to simplify mesh efficiently because shape is not always totally respected. D. Luebke, in 1997 (Luebke, 1997), has proposed a taxonomy of mesh simplification. He presented the most used algorithms and benchmarks. He extracted different ways to use each algorithm. But in this paper, only

one solution works with volumetric mesh. T. He et al. propose a method based on voxel simplification by using a grid for clustering voxels (He et al., 1995). A marching cube (Lorensen and Cline, 1987) algorithm was applied to produce a surface mesh. But this simplification algorithm did not preserve the shape of the mesh.

In our work, we look for point cloud simplification. Indeed, previous methods which deal with simplification for surface point cloud like (Moening et al., 2004; Pauly et al., 2002; Song and Feng, 2009) are not adapted to our case. All of these methods produce LOD for surface mesh, point cloud is extracted from scanner.

3. PROPOSED APPROACH

This section presents the different methods that we use to visualize data from Green Data Center (GDC). The main goal is to be able to visualize in real-time the evolution of temperature in the data center. For this, we use a special particle method. Particles are located using a segmentation algorithm based on Voronoï cell extraction and Delaunay triangulation. The latency due to the large flow of particles is avoided by using a client server paradigm. We improve our solution by using LOD methods to simplify rendering.

3.1 PARTICLE SYSTEM

The data center is composed by two rooms: an high density and a low density area. The density depends on the number of CPU unit by square centimeter. Formally, we can define each room by three measures: length (l Є R+), width (w Є R+) and height (h Є R+). Figure 2 gives a schema of a room. In each room, some sensors are dropped and shaped as layers. Layers are defin ed with: L = {L1, ..., LnbL}, where nbL is the number of layers. The sensors are set on layers and are defined by: S ={S1, ..., SnbS}, where nbS is the number of sensors. A sensor is characterized by: Si = {xi, yi, Lj , di1, ..., dia}, with i Є N+ and I ≤ nbS, x and y are the coordinates of sensor and are defined in R, j Є N+ and j ≤ nbL.

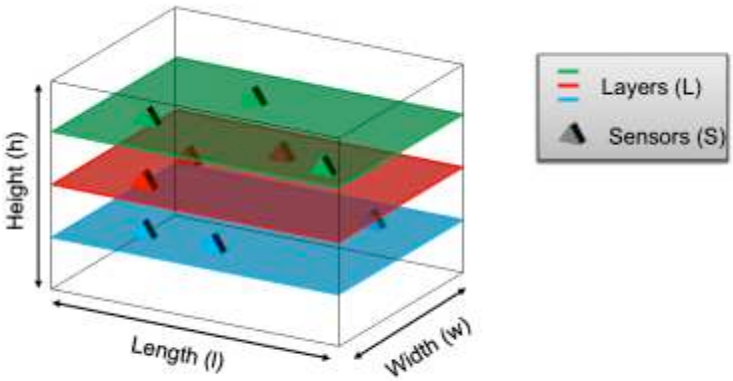


Figure 2. Room model with sensors layers

In our solution, room space is visualized by particles placed on a regular grid. Particles are defined by: P = {p1, ..., pnbP }, where nbP is the number particles and nbP Є N+. Each particle owns his own coordinate pk = {xk, yk, zk}, k ≤ nbP, where 0 ≤ xk ≤ l, 0 ≤ yk ≤ w, 0 zk ≤ h. For the regular grid, coordinates of each particle can be calculated with:

$$p_k(k \in \mathbb{N}, k \leq K_{max}) = \begin{cases} x_k = b \cdot \rho, \\ y_k = c \cdot \rho, \\ z_k = d \cdot \rho \end{cases}$$

where b Є N+, 0 ≤ b ≤ l, c Є N+, 0 ≤ c ≤ w, d Є N+, 0 ≤ d h and f Є R+. f represents the step between two particles. We use this measure to produce a multi resolution mesh designed for different devices. Each device can visualize a limited number particles before getting some latency. We can calculate the numb of particles for this method by the formula:

$$nbP = \frac{((l+1) \times (h+1) \times (w+1))}{f^3}.$$

This grid is regular and fills the entire structure of the room. Figure 3 presents a schema of a regular grid in a room.

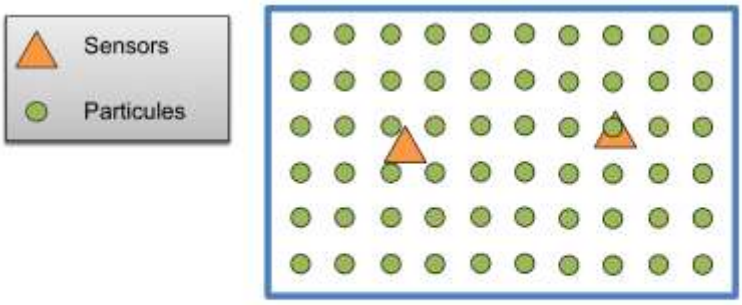


Figure 3. Regular grid

3.2 SEGMENTATION ALGORITHMS

In our solution, each particle has a influence on surrounding sensors. To calculate the set of particles in the sensor range, we use two methods: Voronoï cells extraction and Delaunay triangulation.

Voronoï diagram is a special kind of decomposition of space, mainly used in 2D, but also useful for 3D space. In our particle system, this solution can be assimilated to a collision between sphere and a point. Tools for extracting 3D Voronoï diagrams exist Voro++ and QHull but particles are discrete and these solutions are not suitable because they extract Voronoï diagram in a continuous way. Then we designed our own method based on sphere expansion in order to find nearest sensors for each particle.

The second solution used for solving the problem of space partitioning is the Delaunay triangulation, the dual function of Voronoï cells. This method allows us to give a better approximation to particles weight. The particles are influenced by sensors, and with this method we can determine the location of a particle compared to the mesh of sensors. Delaunay works by looking the nearest neighbors of a point. In 2D, the extracted structure is triangular mesh whereas in 3D, we obtain a tetrahedron soup. Figure 4 represents a tetrahedron mesh.

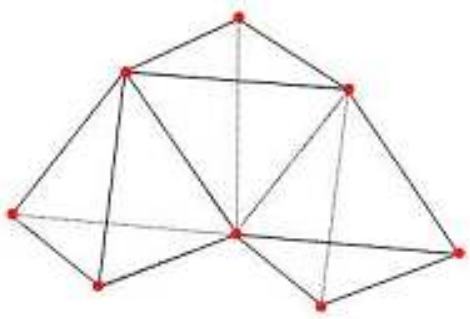


Figure 4. 3D Delaunay triangulation

After the extraction of the different triangles, we need to analyze particles compared to the different tetrahedrons. The goal of this step are to determine in which tetrahedron the particle is located. This method is based on the ray tracing method introduced in . For each particle, we analyze its position compared to the different faces of the tetrahedrons. We compute the surface normal of each face and we apply them to the particle. If the ray intersects more than 3 faces, the particle belongs to the tetrahedron. For the particles outside all tetrahedrons, we use Voronoï cell to assign them to the nearest sensor. The particles concerned by these solutions are the particles outside the set of sensors. The complexity of this computation is high. The computational cost is around $O(N \times M \times 4)$, with N the number of particles and M the number of sensors.

Some improvements can reduce complexity. We create a box of particles around each tetrahedron. This solution compute a Hull box around the tetrahedron. Particles are compute at only one time for each four vertex of a tetrahedron. The box was created in extracting extremal point of the tetrahedron, minimal and maximum coordinates. The complexity of this solution decreases and we only compute at $O(N \times 4)$. Moreover, this calculation is only done in preprocessing because our particles are static and position does not need to be update.

3.3 CLIENT/ SERVER PARADIGM

To improve computation, a client server paradigm is used. We define a low cost communication protocol to transfer data from a server to a client. Server computes the modification of particles and client displays the results. This protocol works in five steps. These steps are: sending header, sending sensor data, sending particle data, sending footer and receiving acknowledgment/command from client. At each step, the server waits the acknowledgment from the client. We develop two ways to send data. The first sends the entire point cloud (sensors and particles). The biggest problem of this method is the transmission of data. Sensors are sent with their coordinates and their value. We encode these data in bit words. For the particles data, the same method was used. The footer was sent for closing the communication..

The second method is used to reduce efficiently the communication cost. We use the same first part, but we only send modified sensors and particles, the id is sent instead of coordinates and the new value. The next step is the command sent by the client. It allows the user to interact with the server. We use it to modify the camera viewpoint.

3.4 LOD

Level of detail (LOD) is one of the most important methods in computer graphics. It allows to solve rendering problems or performance problems. In this method several resolution models of a 3D object are produced. In our works, we use hardware and viewpoint to define the object resolution. LOD was defined by two problems statement. The first one uses a sample of original points, the second one uses a new point data set. In this part, we define six methods to produce LOD. The four first methods are for the client, the other are for the server.

Problems statement:

For this two approaches, we have a set ω of vertices V, $V = \{V_1 , ..., V\omega \}$. Each vertex is defined in R3. Simplify a mesh using a sample vertex means $\omega > \omega 2$, where ω 2 is the size of the second data set. For approach 1, we obtain a new object $V2 = \{V2_1 , ... , V2\omega \}$ with fewer point than V but V2 is a subset of V. For approach 2, we obtain a new object $V3 = \{V3_1 , ... , V3\omega \}$ with fewer point than V but each point in V3 is a new vertex.

In Section 2 we have presented methods to produce simplification. A few were designed for volumetric simplification. In this section, we propose several methods to produce different volumetric simplifications on our client. We develop four approaches to simplify 3D objects: clustering, neighbor simplification, LOD 0 and LOD -1, and two more approaches based on server.

Clustering method was based on He et al.[He et al., 1995] works, it consists of clustering particles using a 3D grid. Cells sizes of grid are set depending to the viewpoint of the camera. Clusters were being weight with the average of the different values of particles. The position is the barycenter of these particles. Figure 6 give some examples of simplification using clustering solution. Figure 6a) present the original point of cloud mesh. Figure

6b and 6d give two different methods for clustering. And finally, Figure 6c and 6e give the results of clustering methods.

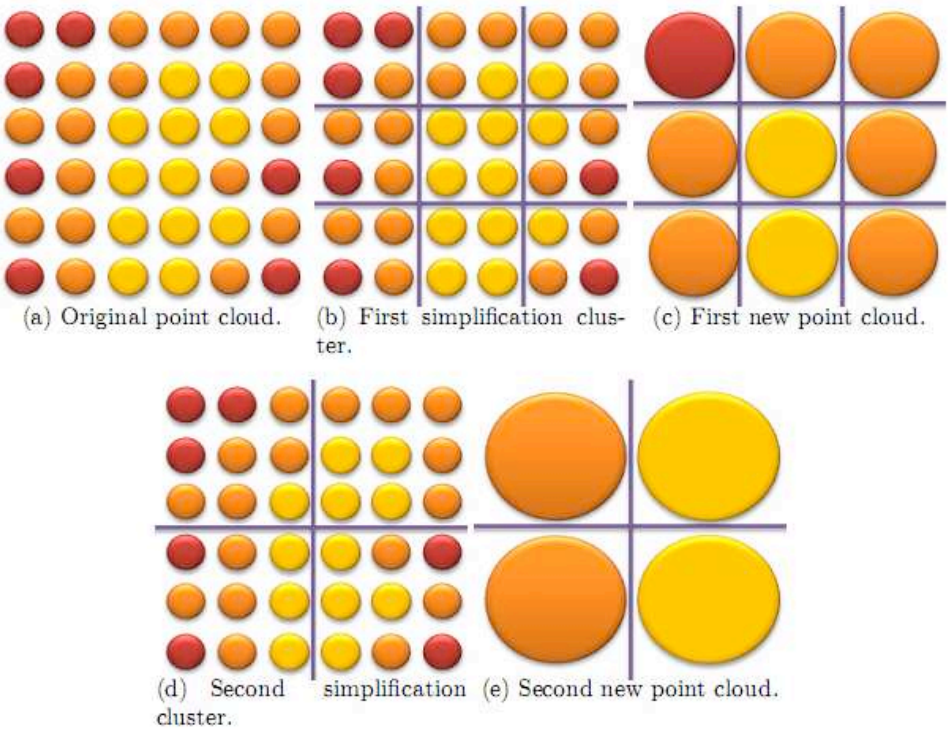


Figure 5. Clustering method for point cloud simplification

The second solution used is based on neighborhood extraction. Before runtime, we extract all neighbors of a particle. We measure the distance between each particle. Some optimization can help to decrease complexity: we can estimate easily in our structure which particle is closer to another one (using the fact that particle grid is regular). After this, we extract the main value of particles. We explore each neighbor of particles and we keep the most important. In some cases, the most important can be the high values, in other the low values and in other both of them. This solution is able to produce a low resolution model with the most important information structure. Several low resolution models are created by exploring deeper in neighborhood. Figure 7 illustrates a neighbor, and two simplifications of this mesh.

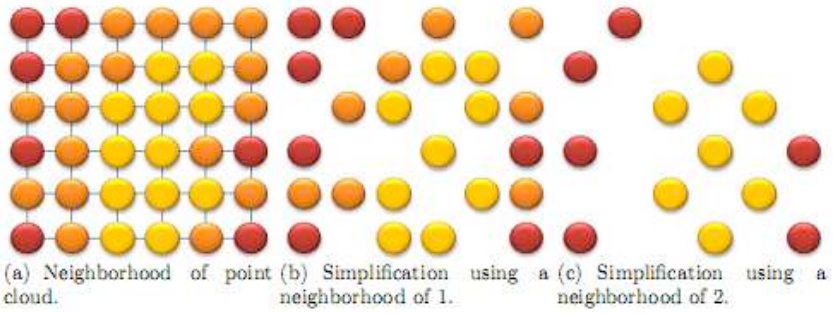


Figure 6. Neighbor method simplification

In LOD produce a LOD 0 means to create the lower resolution mesh of a model. It was introduced by H. Hoppe[?], he calls this model a based mesh. In this paper, the low mesh resolution is the sensor mesh. It is used when the distance from the camera is too important. Particles are replaced by sensors. This solution produces a

low resolution mesh with high fidelity. To continue, we can determine a lower resolution, which can be called LOD -1. This kind of LOD is the barycenter of mesh sensors. We compute the average value of sensors. Figure 8a shows sensors and particles, Figure 8b presents the solution of LOD0 and finally, Figure 8c shows the LOD -1.

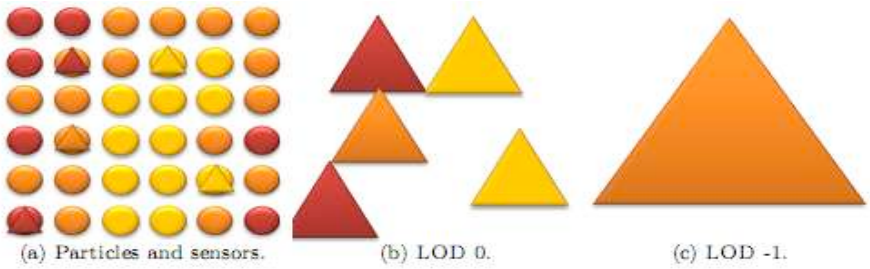


Figure 7. The smallest LOD

Other methods were based on server instead of client. Client sent via TCP connection his viewpoint. The server recomputes the particles structure and recreates the entire structure. With this solution, it is possible to produce many particles cloud resolutions. Figure 9 presents particles rendering at different distances.

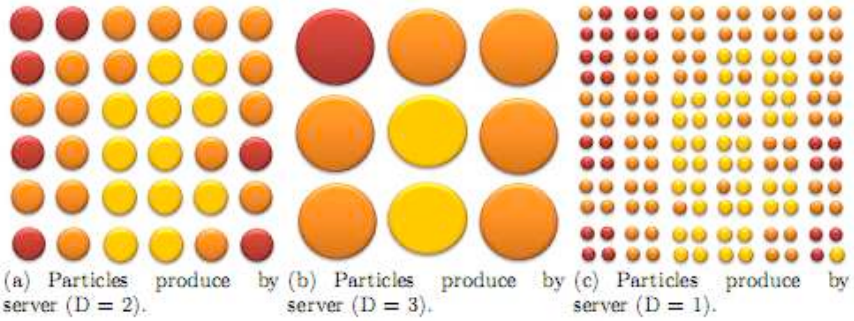


Figure 8. Particle simplification using server and distance D

Another method was based on Voronoï diffusion of temperature. The bandwidth for transmitting data is limited. We developed Vorono temperature diffusion to solve this communication. In this approach, we update data using sphere expansion. Each time, we update some particles depending on their distance from sensors. The more particles are distant from sensors the later they will be refreshed. This method sends only modified particles. The bandwidth is saved and the visualization gives a flow effect. Figure 10 represents values at time 0. At time 1, values of sensors change, 10b. After time 2, we update a first range of particles 10c and finally the second range 10d.

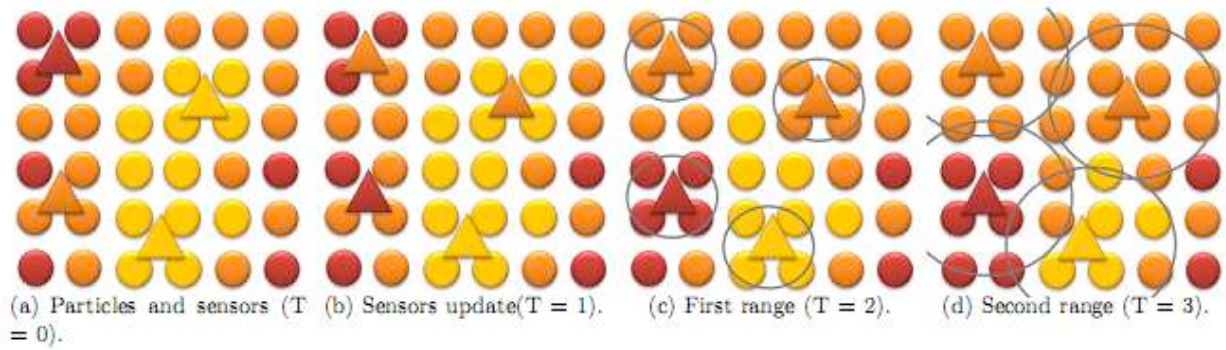


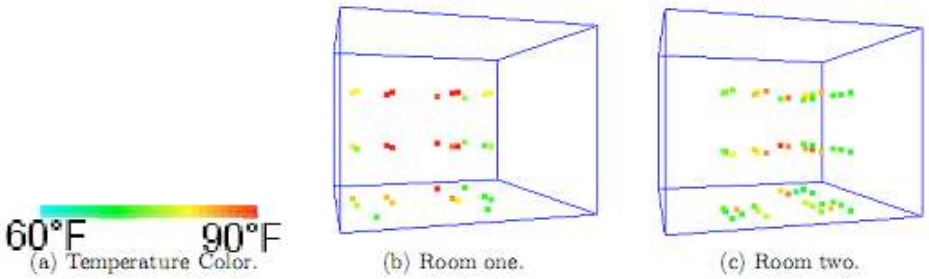
Figure 9. Simplification using bandwidth size

4. EXPERIMENTAL RESULTS

The data are extracted from two rooms of the IBM data center. Firstly, we present our method for rendering the room, and later we present our results using Level Of Detail methods.

4.1 DATA VISUALIZATION

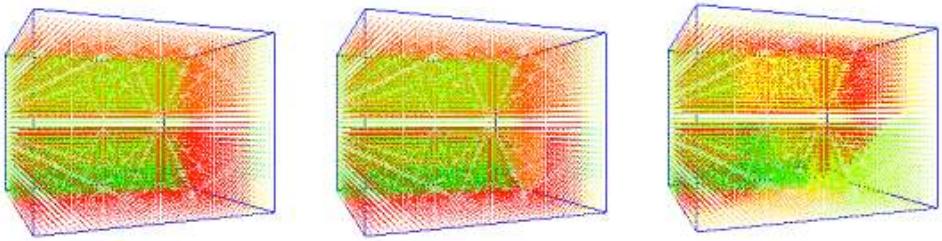
We want to visualize and manage the consumption of a data center. For the visualization, we want to use an IFC viewer. But the IFC model for GDC is not available yet. Data center extraction of the room space is for the moment done by hand. The room is empty and was represent by a simple shape a box with 4 meters length, 3 meters width and 2.5 meters height. We use point cloud visualization based on particle paradigm. We use the two rooms of the data center and we put the same number of particles (30000) and 35 sensors distributed on three layers at 1 meter, 2 meter and on the ground. We define high and low temperature regarding the real sensors value. Figure 11a presents temperature color scale, Figures 11B and c present data center sensors.



(a) Temperature Color. (b) Room one. (c) Room two.

Figure 10.Data center 3D model

The next step is to interpolate data from sensors. Figure 12 give some first results with the partition method described in Section 3.2.



(a) Real room at T = 0. (b) Real room at T = 1. (c) Real room at T = 100.

Figure 11. Particles visualization

4.2 LEVEL OF DETAILS

In the earlier days of this project, first solution proposed gives a low frame rates, about 15 FPS (Frame Per Second): visualization was not in real-time (real-time is about 24 FPS). For solving this problem, we define a client server paradigm. This solution allows to produce a real-time rendering on the client.

We use Openscenegraphas a 3D engine. It owns several features useful in LOD. A special object is defined to manage multi-resolution model. It calculates the distance of the object from the camera. For our experimentation we use five resolutions of mesh. The first mesh was the original mesh, it is set at 0 to 500. The next mesh was set at 500 to 1000, the next at 1000 to 1500 and the other at 1500 to 2000. These three meshes were constructed by specific LOD methods: clustering and significant vertices. LOD 0 and -1 are used at 2000 to 2500 and 2500 to 4000 pixels.

Clustering defines a 3D grid inside the room. The size of each cell depends on the viewpoint location. The size of the cluster depends on the visibility of the clustered particles. Value of cluster is an average of clustered value. The number of points of the final mesh depends on the grid size. Table 1 shows the results at several distances.

	D = 0 to 500	D = 500 to 1000	D = 1000 to 1500	D = 1500 to 2000
C = X	30 000	3 900	240	36

Table 1. Results of clustering simplification.

Significant points method extracts the neighbors for each particle. We extract the highest and lowest temperatures, by exploring the neighborhood of a particle, in order to have significant vertices of the model. For the first step of simplified model we explore neighbor. For the second model, we explore neighbor and neighbor of neighbor, etc. This solution simplifies drastically the model. Table 2 shows the number of vertices at several distance.

	D = 0 to 500	D = 500 to 1000	D = 1000 to 1500	D = 1500 to 2000
C = X	30 000	22 950	4 554	3 524

Table 2. Results of neighbor simplification.

LOD 0 is the most simplified mesh. It was used at far distance and did not need to keep a high resolution for mesh. In this work, we use only sensors to visualize temperatures of the room to produce this low resolution model. Figure 11 presents results of this method at far distance. As we can see, we only keep the important value. Another simplification consists of producing another simplified model by clustering sensors. When the viewpoint is too far from the data, we can compute barycenter of the room and apply the average of temperature of each sensor. This method can simplify the 3D scene and gives some information at far distance.

The first server solution receives orders from client as presented Section 3.4. We calculate the viewpoint distance and we send data according to it. A new structure is recalculated if the camera is too far from the object. The computational cost of this method is high but client server paradigm reduces work on the client. After the recomputing, we send the new data. This solution allows the user to receive more or less data according to its distance to the object. Table 3 shows some different resolutions produced with this method.

	D = 0 to 500	D = 500 to 1000	D = 1000 to 1500	D = 1500 to 2000
C = X	120 000	30 000	7 500	1 875

Table 3. Several resolution of model.

Another solution is to use bandwidth latency. We send data at several times, we do not send the entire set of data but only modified particles. We send at first time the sensors data, and subsequently we send a range of data (the nearest). After few minutes, all data are sent. This solution gives good results, and simulates a thermal diffusion in the whole structure of particles. We use real update time as the maximum update time.

5. CONCLUSION

In this paper, we have presented a method to visualize sensors data extracted from a Green Data Center. This approach produces interpolation visualization for managing and visualizing data. This interpolation used a Delaunay triangulation and a cell extraction based on Voronoï. An unusual way of use particles helps to process data. First results present the solution proposed to visualize the inside of a GDC space.

The second method proposed in this paper aim to improve the rendering. For this, first step introduces a client/server protocol and a second step illustrates methods to simplify the model. With these different approaches

we improve the rendering time, preserving most important data are kept. In future works, we will work on data "dressing". We want to find a way to improve rendering of the scene using meatballs or marching cube algorithms. A main constraint of this work is real-time computation. Future work also concern to add rooms to the visualization. At present, we only visualize a single room. We want to visualize building and complex form, and adapt visualization to several devices as smartphones.

ACKNOWLEDGMENTS

We want to thanks the PSSC (Products and Solutions Support Center) team of IBM Montpellier for having provided the necessary equipment and data needed for this experimentation. And we thank the FUI (Fonds Unique Interministeriel) for their financial support.

REFERENCES

Clark, J. H., Hierarchical geometric models for visible surface algorithms, Communications of the ACM 19(10), 547{554 (1976).

Damon, M., Kameyama, M., Knox, M., Porter, D., Yuen, D., and Sevre, E., Interactive visualization of 3d mantle convection," Visual Geosciences (2008).

He, T., Hong, L., Kaufman, A., Varshney, A., and Wang, S., Voxel based object simplification, in Proceedings of .SIGGRAPH Symposium on Interactive 3D Graphics , 296{303 (1995).

Jordan, K. E., Yuen, D. A., Reuteler, D. M., Zhang, S., and Haimes, R., Parallel interactive visualization of 3d mantle convection, IEEE Comput. Sci. Eng. 3(4), 29{37 (1996).

Kapferer, W. and Riser, T., \Visualization needs and techniques for astrophysical simulations, New Journal of Physics 10(12), 125008 (15pp) (2008)

Latta, L.,Building a million particle system," (2004).

Lorensen, W. E. and Cline, H. E., Marching cubes: A high resolution 3d surface construction algorithm,SIGGRAPH Comput. Graph. 21(4), 163{169 (1987).

Luebke, D., A survey of polygonal simplification algorithms, (1997).

Moenning, C., , Moenning, C., and Dodgson, N. A., Intrinsic point cloud simplification," (2004

Pauly, M., Gross, M., and Kobbelt, L. P., Efficient simplification of point-sampled surfaces, (2002).

Reeves, W. T., Particle systems - a technique for modeling a class of fuzzy objects, ACM Transactions on Graphics 2, 359{376 (1983).

Song, H. and Feng, H.-Y., A progressive point cloud simplification algorithm with preserved sharp edge data, The International Journal of Advanced Manufacturing Technology 45, 583{592 (November 2009)

Authorization and Disclaimer