Optimal Combination of Multi-resolution Models in Visualization Techniques

Xiaolong Yang[1,2, a *], Hao Gu[1,2] and Fengju Kang[1,2]

[1] School of Marine Science and Technology of Northwestern Polytechnical University, Xi'an 710072, China

[2] National Key Laboratory of Underwater Information Process and Control, Xi'an 710072, China

[a]yangxl060120@aliyun.com

Keywords: Multi-resolution model, Optimal combination, Visualization, Lagrange multiplier.

Abstract. Complex VR scenes, containing hundreds of multi-resolution models with millions polygons, cannot achieve real-time performance and verisimilitude at the same time. How to adaptively choose the appropriate level of resolutions according to the data volume and hardware/software environment is therefore a critical issue. This paper we propose an optimization algorithm to choose the model resolutions that maximize scene quality while meeting timing constraints. The mathematical model is given by means of the relationship between scene representation and model resolution. Lagrange multiplier method is used to solve the problem of extreme value solution under time constraints.

Introduction

With the rapid development of virtual reality technology, scenarios become increasingly complex, the number of models (including three-dimension solid models, special effects, and environment models, etc.) is increasing, while the data volume in each model (i.e., the number of vertices and triangular faces) also increasing sharply. Although the performance of current graphic hardware rendering system is also rising, but still cannot meet the large volume of data in real-time scene rendering needs. Construction of multi-resolution modeling of three-dimensional model simplify the models with different levels of details which can be selected as an alternative for different drawing demand, thus can effectively reduce the amount of data volume and improve rendering speed.

Details of expression can provide different resolution models is not the same, the details of the high-resolution model of high performance, but the time required to draw a corresponding increase. Therefore, how to draw the time constraints under suitable conditions for the resolution of different models to get the highest allocation model expression become multi-resolution model of the key issues.

The traditional model resolution are based on the experience of application developers to determine, developers and hardware performance depending on the complexity of the scene rendering environment, pre-established rules of the respective resolution of the model chosen. [1] LOD models use a simplified method for the establishment of three-dimensional city model provides a way to reference [2] for the virtual campus programs achieve three levels of detail model, and select the level of detail based on the model from the viewpoint of the distance. This approach not only spend a lot of time and effort, but once the environment changes need to draw re-enact the resolution selection rule for the model. [3] The importance of index model, considering the distance, speed, user concerns and other factors model, we propose a model for complex multi-resolution scene selection algorithm. The algorithm can realize limit drawing complex scenes, but could not get the optimal solution of the model of multi-resolution selected. Therefore, a multi-resolution model limit an adaptive rendering method, the method by plotting time and quantitative model to predict performance capabilities, the establishment of a multi-resolution models show that the mathematical model painting problems, and using Lagrange multiplication real time solve the optimal solution of the model to determine the optimal multi-resolution rendering of the model.

Multi-resolution Mesh Representation

In graphics, the triangular mesh is a common representation for three-dimensional shape and free-form surface. The so-called triangular mesh is a collection of triangular facets with different sizes which constitute the surface of an object. The higher resolution model uses the more triangles.

Multi-resolution representation of three-dimensional model can be divided into two ways, discrete representation and continuous representation. Discrete multi-resolution model refers to the original model produced in advance for a plurality of different degrees of approximation model. In real-time rendering, an appropriate approximation of the model is selected to draw according to the viewpoint of the current frame. Many commercial systems they use this method for advantage of simply used. But the disadvantage of this approach is obvious, a) approximation models are not continuous resulting image hysteresis while switching models with different solution, b) choose different resolution approximation model requires manual intervention, c) additional memory is required to store different resolution approximation models.

Multi-core technology Continuous-resolution model is a simplified model, which uses less triangular facets within a certain error range instead of the original model. Geometric elements operating foundation currently has a triangular model simplification algorithm can be divided into vertex deletion algorithms, triangular facets delete algorithm and three edge collapse algorithm [4]. Wherein the edge collapse algorithm because of the speed, robustness advantages, as well as the incremental generation has a natural advantage to simplify multi-resolution models, gained extensive application.

Edge Collapse Algorithm

Hoppe et al. [5] describe a method, mesh optimization that can be used to simplify an initial mesh to a much simpler one. Based on edge collapse, the basic idea is: In each iteration an edge is selected as deleted object and add a new vertex, all vertices linked with the deleted edge are connected with the new vertex to maintain a triangular grid. In this process an edge is "folded" into a vertex, so it is called "edge collapse". Repeating this process, we can get simplified triangle mesh at any resolution.

In Fig. 1, the deleted side is $e = < v_1, v_2 >$, and v_1, v_2 is collapsed to the new vertex v_0, two triangles connected are removed. If e appears on the boundary of a mesh, only one triangle will be deleted. With collapsing edge the number of triangles reduces, in the meantime the resolution decreases.

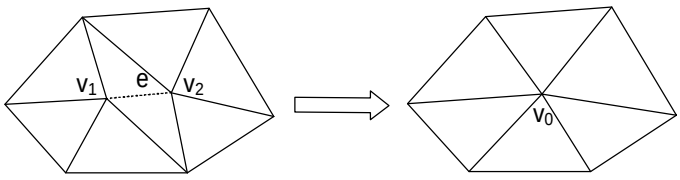

Fig.1 Edge Collapse Algorithm

The main steps of edge collapse algorithm are:

1) The weight of each edge $W(e)$ is obtained according to a predetermined algorithm;
2) Select the smallest weight edge (v_i, v_j) to perform edge collapsing operation, delete edge (v_i, v_j) and its adjacent triangles;
3) Updated the set of edges to be deleted;
4) Exit the algorithm if the condition meets demand, otherwise go to step 1.

Edge collapse algorithm termination condition is generally the following two conditions: simplify error reaches a certain threshold δ; triangular grid resolution to the desired value.

Progressive Mesh Representation

On the basis of edge collapse algorithm, Hoppe proposed a PM (Progressive Mesh) construct method [6]. A progressive mesh is consist of a simplified grid $\mathbf{Mr}$ and a vertex list, which can be expressed as $\{Mr, Vsplit_0, Vsplit_1, \cdots, Vsplit_{n-1}\}$. The vertex list keeps a record of deleted vertices and their orders, it shows how the mesh $\mathbf{M}$ is converted to $\mathbf{Mr}$.

Progressive mesh include two reciprocal processes mesh simplification and reconstruction. First, simplification is achieved by edge collapsing operation. In the iterative process, each collapsing operation remove one edge and the adjacent triangles are removed, the resolution of mesh decreases. Finally we get a low-resolution mesh $\mathbf{Mr}$ and a record of vertex information that are deleted. Otherwise, the reconstruction of the grid is the inverse operation of the edge collapsing operation which is called point splitting. According to the record $\{Vsplit_0, Vsplit_1, \cdots, Vsplit_{n-1}\}$, we can insert vertices and triangles to a simplified mesh $\mathbf{Mr}$ and finally reconstruct the original mesh.

Problem Description

Generally a triangular mesh can be represented as a data set $M = (K, V, A)$, where K represents the connection between the mesh vertices; $V = \{v_1, v_2, \cdots, v_m\}$ is the data set of vertices; $A = \{a_1, a_2, \cdots, a_m\}$ is the set of corresponding attributes set, including normals, texture coordinates and other attributes. Suppose a triangular mesh Mcontains $N_{maxvert}$ vertices and N_{maxtri}triangular facets, the simplified model with a resolution $r \in [0,1]$ can be expressed as $M(r)$, and $M(r)$ contains approximately $\lfloor rN_{maxtri} \rfloor$ triangular facets.

In each frame of the scene rendering, the rendering pipeline obtains the combination of models to be drawn by visibility culling, in which each model can be drawn in their own resolution $r \in [0,1]$. Therefore, assuming that the current frame is composed of n models, define $S(\mathbf{r}) = \{M_1(r_1), M_2(r_2), \cdots, M_n(r_n)\}$ as a set of n simplified triangular meshes, where $\mathbf{r} \in [0,1]^n$. The purpose of optimal combination is to find the optimal solution $\mathbf{r}^*$ in which models can get the highest performance in constraint time. Define $cost(S(\mathbf{r}))$ as rendering time that is required to render n models in resolution $\mathbf{r}$, $representation(S(\mathbf{r}))$ as the sum of contributions that is made to the scene by n models.

The optimization problem can be described by Eq.1.

$$\begin{cases} \max\{representation(S(\mathbf{r}))\} \\ cost(S(\mathbf{r})) \le t^* \end{cases} \tag{1}$$

Where t^* is the given constraint time.

In order to solve combinatorial optimization problem, we should get the expressions for $representation(S(r))$ and $cost(S(r))$, then the problem is transformed into solving the extreme value of representation under the constraint of time.

Cost

For a visual system, it is prerequisite to guarantee real-time rendering, which requires time for the allocation of resources and choosing appropriate multi-resolution triangular meshes to draw. To choose a resolution, the final result should satisfy that models can be drawn in the limitation of the predetermined time. To achieve this goal, we need to estimate the rendering time of each model.

A rendering frame can be divided into three phases: the initialization phase (configuring pipeline, clearing the cache), the mesh rendering phase, the drawing phase (exchanging buffers) [7]. In most current graphics pipeline, the speed of CPU can be considered fast enough, the time of initialization phase and drawing phase can be considered to be a fixed value. On most hardware configurations, the time to define rendering attributes and process all the triangles is just dictated by the speed of the graphics pipeline for all operations before rasterization. On very high-end graphics systems, actually fetching triangles from the multi-resolution structure may become the dominant phase. In both cases,

assuming that the rendering time is proportional to the number of triangles in the mesh, the cost time of mesh M drawn in resolution $\mathbf{r}$ can be expressed as Eq.2.

$$t_{const} + \max\begin{Bmatrix} t_{tri} r N_{maxtri}(M) \\ t_{pix} N_{pix}(M) \end{Bmatrix} \tag{2}$$

where, t_{const} represents the time required to configure render environment (texture binding, etc.) for the pipeline, t_{tri} is the time to fetch a triangle from memory to the graphics card, t_{pix} is the time to fill a pixel, $N_{pix}(M)$ is the number of pixels occupied by mesh M.

By Eq.2 the minimum resolution r_{min} of mesh M can be obtained as Eq.3, when the grid resolution reaches r_{min}, reducing the grid resolution will not reduce rendering time.

$$r_{min} = \frac{t_{pix} N_{pix}(M)}{t_{tri} r N_{maxtri}(M)} \tag{3}$$

Model combinations $S(r)$ rendering time can be expressed as Eq.4.

$$r_{min} = \frac{t_{pix} N_{pix}(M)}{t_{tri} r N_{maxtri}(M)} \tag{4}$$

Wherein, $T_{fixed} = T_{init} + T_{final} + \sum_i T_{const\,i}$ there is no direct relationship with the grid resolution, T_{init}, T_{final} denote pipeline initialization time and buffer exchange time, $\mathbf{t}_{max}$ for each grid maximum rendering time $t_{tri} N_{maxtri}(M)$ constituting the vector. t_{const}, t_{tri}, t_{pix}, T_{init}, T_{final} contour lines drawn by the hardware configuration decisions can be obtained at the program preprocessing stage measurements.

Representation

The $representation(S(\mathbf{r}))$ refers to the ability that meshes in solution $\mathbf{r}$ can exhibit the details of original meshes. The literature [8] gives a simplified model of expression.

$$representation(S(\mathbf{r})) = \sum_i importance(M_i) accuracy(M_i, r_i) \tag{5}$$

Wherein, $importance(M)$ indicates the importance of meshes M evaluation factor, $accuracy(M, r)$ represents the ability that the mesh in solution r can exhibit the details of the original mesh M.

The importance factor of mesh M can be obtained from three elements: the number of pixels that are occupied by M, the distance from the center of the screen, and user's self-awareness, so you can get:

$$importance(M) = coverage(M) focus(M) semantics(M) \tag{6}$$

Where, $coverage(M)$ represents the number of pixels occupied by mesh M, which can be obtained after visibility culling; $focus(M)$ is the distance of the mesh M and the center of the screen; $semantics(M)$ is the user-defined awareness, which is given in advance or determined according to the interaction of user.

Visually pleasing results were obtained using $accuracy(M, r) = \sqrt{N_{maxvert} r}$ for a mesh with $N_{maxvert}$ vertices at the highest level of detail, which provides diminishing returns at higher resolutions and, intuitively, relates representation accuracy to an indication of the distance between samples on the surface.

Optimization Algorithm

With our cost and representation heuristics, the resolution optimization problem in Eq.1 can now be written as:

$$\max\{f(\mathbf{r}) = \sum_i \frac{\sigma_i}{\alpha} r_i^{-\alpha} \,|\, \mathbf{Ar} \leq \mathbf{b}\} \tag{7}$$

where

$$\mathbf{A}=\begin{pmatrix} t_1^{(max)} & t_2^{(max)} & \cdots & t_n^{(max)} \\ -1 & & & \\ & -1 & & \\ & & \ddots & \\ & & & -1 \\ 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{pmatrix}, \mathbf{b}=\begin{pmatrix} t_{desired}-t_{fixed} \\ -r_1^{(min)} \\ -r_2^{(min)} \\ \vdots \\ -r_n^{(min)} \\ 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}, \mathbf{c}=\begin{pmatrix} c_1 f_1 s_1 \\ c_2 f_2 s_2 \\ \vdots \\ c_n f_n s_n \end{pmatrix}$$

The Lagrange multiplier algorithm to solve Eq.7 is described in Fig.2.

Algorithm 1: Active-set strategy for solving $\min\{f(r):\ \mathbf{a}_i^{\mathrm{T}}\mathbf{r} \le \mathbf{b},\ i \in \mathscr{I}\}$.

Given: Starting point $\mathbf{r}^{(0)}:\ \mathbf{a}_i^{\mathrm{T}}\mathbf{r}^{(0)} \le \mathbf{b},\ i \in \mathscr{I}$
Given: Timing constraint $t^{(\mathrm{opt})}$
$k \leftarrow 0$; *done* $\leftarrow$ false; *stopped* $\leftarrow$ false
$\mathscr{W}^{(k)} \leftarrow \{i \in \mathscr{I}:\ \mathbf{a}_i^{\mathrm{T}}\mathbf{r}^{(k)} = \mathbf{b}\}$
while not (*done* **or** *stopped*) **do**
  $\mathbf{d}^{(k)} \leftarrow \arg\min_{\mathbf{d}}\{f(\mathbf{r}^{(k)}+\mathbf{d}):\ \mathbf{a}_i^{\mathrm{T}}(\mathbf{r}^{(k)}+\mathbf{d}) = \mathbf{b},\ i \in \mathscr{W}^{(k)}\}$
  if $\|\mathbf{d}^{(k)}\| = 0$ **then**
    $\mathbf{r}^{(k+1)} \leftarrow \mathbf{r}^{(k)}$
    $\lambda^{(k+1)} \leftarrow \lambda : \nabla f(\mathbf{r}^{(k+1)}) + \sum_{i \in \mathscr{W}^{(k)}} \lambda_i^{(k+1)}\mathbf{a}_i = 0$
    $j \leftarrow \arg\min_i\{\lambda_i^{(k+1)}\}$
    if $\lambda_j^{k+1} < 0$ **then**
      $\mathscr{W}^{(k+1)} \leftarrow \mathscr{W}^{(k)} \setminus \{j\}$
    else
      $\mathscr{W}^{(k+1)} \leftarrow \mathscr{W}^{(k)}$; *done* $\leftarrow$ **true**
    end if
  else
    $\mu^{(k)} \leftarrow \arg\max_{\mu}\{\mu = \min(1, (b_i - \mathbf{a}_i^{\mathrm{T}}\mathbf{r}^{(k)})/\mathbf{a}_i^{\mathrm{T}}\mathbf{d}^{(k)}) : \mathbf{a}_i^{\mathrm{T}}\mathbf{d}^{(k)} > 0,\ i \in \mathscr{I} \setminus \mathscr{W}^{(k)}\}$
    $\mathbf{r}^{(k+1)} \leftarrow \mathbf{r}^{(k)} + \mu^{(k)}\mathbf{d}^{(k)}$
    $\mathscr{W}^{(k+1)} \leftarrow \mathscr{W}^{(k)} \cup \{i \in \mathscr{I}/\mathscr{W}^{(k)}:\ \mathbf{a}_i^{\mathrm{T}}\mathbf{r}^{(k)} = \mathbf{b}\}$
  end if
  $k \leftarrow k+1$; *stopped* = Time elapsed $> t^{(\mathrm{opt})}$
end while
$\mathbf{r}^* \leftarrow \mathbf{r}^{(k)}$
$\mathscr{W}^* \leftarrow \mathscr{W}^{(k)}$
Ensure: $\mathbf{r}^*$ is feasible and $f(\mathbf{r}^*) \le f(\mathbf{r}^{(0)})$
Ensure: *done* $\Rightarrow \mathbf{r}^*$ is optimal and $\mathscr{W}^*$ is the optimal active set
Ensure: *done* $\Rightarrow$ Time elapsed $\le t^{(\mathrm{opt})}$
Ensure: not *done* $\Rightarrow$ Time elapsed $\approx t^{(\mathrm{opt})}$

Fig. 2 Lagrange Multiplier Algorithm

Conclusion

Visual scene from the 3D solid model environment effects and composition, drawing mutual fidelity and real-time constraints of the problem of multi-resolution of these models, presented under a time constraint multi-resolution model optimized combination method. Analysis of the current rendering process graphics system, designed a multi-resolution model of continuous drawing time prediction method given model performance evaluation method including distance, speed, user awareness and simplify error and other factors, then Lagrange multiplier method to solve the problem of multi-model expressive extreme value under time constraints to obtain the optimal model for multi-resolution mapping program.

References

[1] CAO Bing-xia, MENG Yong-fei, WANG Qian-jin, A New Method of Three Dimensional LOD Model Simplification, Surveying and Mapping. 37 (2014) 3-6.

[2] ZHANG Dian-hua, CHEN Yi-min, Design and Implementation of Multi-platform Virtual Campus Based on Unity3D, Computer Technology and Development. 24 (2014) 127-135.

[3] KONG Chui-liu, Application of simplification of three-dimensional architecture model based on asymptotic grid, Journal of Changchun University. 21 (2011) 29-31.

[4] GUO Li-zhen, WU En-hua, Survey of Polygonal Model Simplification Algorithms, Application Research of Computers. (2005) 20-23.

[5] LI Feng-xia, ZHAO Deng, LI Zhong-jun, Mesh Simplification for 3D Models with Exterior Maintaining, Journal of System Simulation. 24 (2012) 1-5.

[6] Hoppe H., Progressive Meshes, In SIGGRAPH'96 Proc. 1996, pp. 99-108.

[7] Gobbetti E, Bouvier E. Time-critical multi-resolution scene rendering, Proceedings of the conference on Visualization'99: celebrating ten years. IEEE Computer Society Press, 1999: 123-130.

[8] Funkhouser T A, Séquin C H. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments, Proceedings of the 20th annual conference on Computer graphics and interactive techniques. ACM, 1993: 247-254.