Cell Segmentation-based Dynamical Loading for Efficient VE Navigation

Xiumei Kang[1] and Qingjin Peng[2]

[1]University of British Columbia, kangx@interchange.ubc.ca
[2]University of Manitoba, Pengq@cc.umanitoba.ca

ABSTRACT

Navigation efficiency is one of the important performance measures of virtual environments. A large-scale virtual environment (VE) is extremely computational demanding for real-time rendering and manipulation. The cell segmentation is an effective method to partition a VE into smaller universes or cells. It is frequently used for rendering large-scale models to minimize demands of the computer capacity. This paper discusses an improved cell segmentation method with a dynamic loading strategy for the performance improvement of large-scale VEs. Case studies are conducted using the proposed method in the product assembly simulation and the building model navigation. Potentially visible sets (PVS) and visual regions are formed based on the attributes of cells in VEs. The VE performance is improved comparedto the non-segmented model.

Keywords:virtual environment, cell segmentation, simulation, 3D navigation.
DOI: 10.3722/cadaps.2011.407-420

1 INTRODUCTION

Virtual environments provide cost-effective user interfaces for the product development and simulation. A virtual environment (VE) can consist of numerous 3D models with different colors, textures, material properties and product behaviors. The construction of a VE includes the object modeling, the model visualization planning, and the model manipulation design and processing using virtual reality (VR) tools. A VE is modeled using texture mapping, lighting, animating and interactive operations.

Alarge-scale VE results in a large computation load for the VR engine to processmodels in real-time. In addition, the computer memory required for a large-scale VE can become very demanding. Although the computer capability has been increased significantly, a large-scale VE can severely influence the performance of a VR system.

Model management techniques are frequently used for VR enginesin rendering large-scale VEs to maintain the simulation quality. The model management techniques include methods of levels of detail, cell segmentation, off-line pre-computation, and database management. A combination of these methods is often used for the simulation improvement.Cell segmentation methods partition a virtual world into smaller parts, or cells. The model complexity can be significantly reduced as only a small amount of objects is rendered within the current visible VE.

As a navigator can only see a small area of the whole model in a VE,alarge-scale VE can be divided into several regions. Since other objects of the model are invisible to the navigator at a particular time, it is possible to only load the visible objects to lessen the computer memory usage. Along with the viewpoint moves, new cells are dynamically added and removed from the scene. Thus, the set of polygons that appears in each view can be kept small. Cell segmentation can not only improve the performance of a simulation, but also benefit designers. One important role of the simulation is to facilitate modifications during the design stage. For example, a designeroftenonly needs to changepartsof a product. In this case, it is possible to only loadthe parts to be changed, not the whole product model.

A VE with the poor performance in navigation is unconvincing, hard to use, and difficult to engage users into the environment [1]. One of the performance measurements for VEs is the frame rate. The frame rate is the refreshed frequency of an image in adisplay device, whichis often expressed in frames per second (fps). If the frame rate is too low, the scene will lag and flash between frames leading to unsatisfactory effects. Frame rate is affected by the number of polygons loaded for a geometry model in a VR simulation. In this research, a cell segmentation method is used to reduce the number of polygons to be loaded to increase the frame rate in simulation. The number of polygons and the frame rate are measured to evaluate the performance improvement using the cell segmentation method.

Fig.1 shows architectureof the dynamic loading VR system based on the cell segment.In the pre-processing stage, cells are grouped based on potentially visible sets (PVS), prototypes are created and stored in a prototype library. During the simulation, a navigator's position is monitored simultaneously. When an entry of a new region is detected, the corresponding region will be searched from the database, and the prototype will be dynamically loaded from the prototype library for the navigation.

In following parts of the paper, related work is reviewed in Section 2. Section 3 presents the cell segmentation and dynamic loading strategies for VEs' performance improvement. Section 4 introduces two case studies for the cell segmentation based dynamic loading followed by conclusions and further work discussed in Section 5.

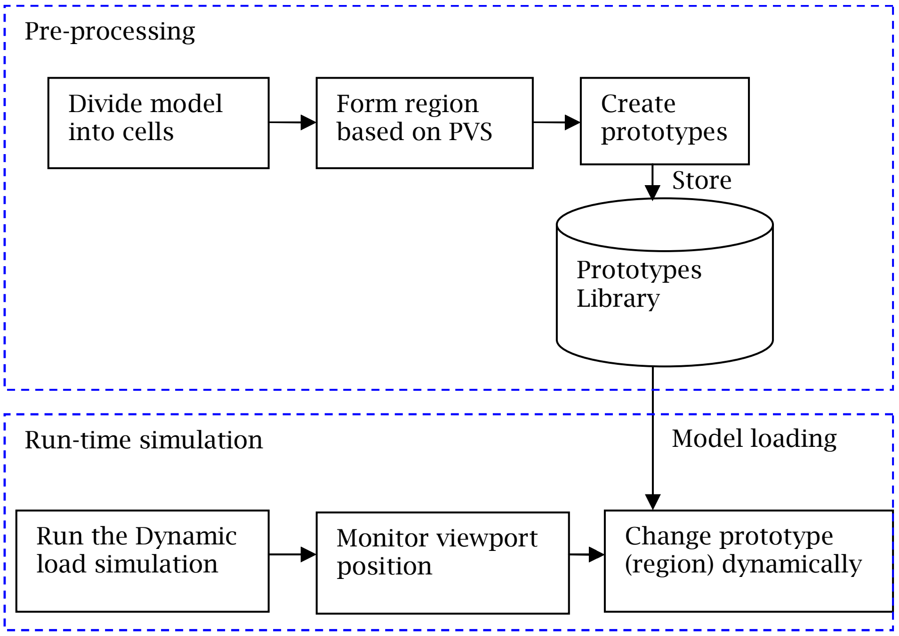


Fig.1: Architecture of the dynamic loading system.

2 RELATED WORK

The large-scale display of VR applications demands the large computer capacity [2]. The behavior modeling of objects and the sophisticated application of VEs lead to rapidly growing demands for computer systems which pose new challenges for the model interaction and visualization [3].Different methods have been proposed to increase the capacity of computers in the process of large-scale display of VEs. The parallel selective rendering was used for a viewer-based scene to achieve high-fidelity virtual environments [4]. Allard et al used a modular design method in developing highly animated VR applications with numerous animated objects managed for physical based simulations [5].

Using levels ofdetail (LOD) methods, the number of polygons can be reduced to meet the need of dynamic real-time simulations [6]. A LODmodelwas used for the generation of effective indoor route visualization [7]. Chu et al used a product model at various LOD to represent the feature hierarchy of CAD construction to enable the real-time visualization of different LOD models in distributed environments[8]. Zhang et al used LOD in real-time visualizationofthe large-scale digital elevation model, which improves the dynamic and visual multi-resolution performance of large-scale terrain in real-time[9]. Zhang and Wu proposed a method to generate multiresolution animation models[10].Different geometrical representations are used for the visualizationof ontologies as a graph[11]. The graph can be built for each concept with different levelsof depth. LOD was used based onmodels' geometric parameters and topology information to achieve considerable speedups in frame rate with little loss in image quality for the visualization of 3D plant models [12]. LOD is also used in the visualizationof complicated 3D buildings [13].

Similar as LOD, an effective approach to improve the system scalabilityis to partition the virtual world in smaller segments called cells. The partition can be used as a standard division of the environment whichis divided into n regions or cells.Cell segmentation has many applications, such as in the 3D models retrieve and reuse [14], in autonomic microcell assignment [15], in the crowd simulation [16], and in the understanding of biological regulations [17]. Zhang scales virtual environments to improve the integration of spatial knowledge and spatial action [18]. It uses a multiscale progressive model in navigation of virtual environments.

Highly interactive visualization of large-scale models for VR facilities is a challenge [19]. By organizing the model into a multi-resolution partition hierarchy, the client-end visualization ensures a fast view reconstruction with efficient occlusion culling and view-dependent levels of detail control. Cell segmentation decreases the dependence on remote computation performance and network requirements for the interactive visualization.

When the environment is dynamic, the optimal partitioning changes over time. Tracking this optimal process is the key challenge of load balancing. For the object extraction in highly complex scenes, Merchán and Adán proposed a method based on a distributed segmentation technique to explore 3D data by establishing a set of suitable observation directions [20]. Using partition for parallel simulation, the dynamic load balancing can significantly reduce the overall calculation time required for the problem [16].

Airey et al [21] presented an automatic model-space subdivision and potentially visible set calculation methods. Potential visible set (PVS) is the union of visible polygons for all viewpoints in a cell. For any viewpoint in a cell, rendering the PVS for that cell gives the same scene as if the whole model is rendered. Since the PVS is much smaller than the total size of the model, rendering takes much less time.

Forming potentially visible sets based on the attribute of cells is the next step after the cell segmentation, which also refers to the visibility calculation. The main algorithms can be classified as point visibility culling, and region visibility culling. The point visibility culling method computes the visibility from a point which is applied in each frame during rendering [22]. The region visibility culling method computes visibility for a region rather than a single point [23].

Visibility culling algorithms have been used for large models. There are exact pre-processing algorithms and conservative runtime algorithms of visibility culling algorithms [24].When an observer moves in VEs for a real-time simulation, the relevant PVS are retrieved from storage. An approach for dynamically determining PVS of cells in the real-time simulation was developed by Luebke and Georges [25]. It uses a screen space projection to compute a conservative estimate of PVS at rendering time, providing increased interactive performance for large architectural models.

The existing research mainly focuses on the visibility algorithmbased on object polygons and meshes without considering the nature of objects. It normally tags each polygon with a number andstores the geometric structure in a database. Instead of this method, a practical solution is proposed in this research to form a local region and store it as a PVS group, which allowsVR systems to only load a part of the model. It improves the navigation performance in large-scaleVEs.The following section describes the proposed cell segmentation and dynamic loading method.

3 CELL SEGMENTATION, REGION CONNECTION, AND DYNAMICAL LOADING

It is highly computational demand for the simulation of large product assembly models in VEs, such as models of an aircraft, a ship or a mine machine. The loading time may be too long if many detailed objects are to be loaded simultaneously. Effective rendering and manipulation techniques are required to overcome such computational demand. A user may observe details of a small portion of the whole product at a time. Many unseen objects can be unloaded at that time to reduce the computer memory usage so that the navigation performance is improved. A product assembly can be modeled as two levels, a detailed level of the model from CAD design, and a level of the model from the product assembly. If a subassembly is defined as a cell or a region,anassembled product can be represented as a simplified product model consisting of several link parts and subassemblies. The component models in the detailed level can be included into cells for the dynamic loading.

3.1 Forming Visual Regions based on Subassemblies

A productcouldconsist of several sub-assemblies. Each sub-assemblyor cell consists of several components. Partitioning a product model into cells to form visual regions is a pre-processing for the dynamic loading.Besides the separation of subassemblies, there are cross parts linking these sub-assemblies. Such parts are defined as cross links. As avisual region is defined based on the subassembly, the cross parts are used as links of regions. The region formation is based on the PVS of cells and cross links.The adjacent visible cells are linked together to form the region for visulization.

3.2 Forming Visual Regions based on PVS

Let's examine the visibility of a product 2D outline shown in Fig.2 (a). Assume that the productconsistsof subassemblies or parts, SA_1,$SA_2$and SA_3. If the part $SA_2$is a fastener to link other cells,we call this partasa link of cells. SA_3 is visible when a designerexamines the subassemblySA_1. As thecell SA_3 is a neighbor cell of SA_1 which is visible to the designer and should be included in the PVS.

On the other hand, if there is not a direct link between the subassemblies, for example, subassembly SA_7 in Fig.2 (b). This cellwill be ignored for the detail review of the cell SA_3 as no direct relation between $SA_7$and SA_3.

Fig.2 (a) shows the cellSA_1's PVS group G_{SA1}based on SA_1. Taking a cellSA_1 as the example, a link matrix $L_{SA1}(+x, -x, +y, -y)$ is defined as the linking property of the cell in the +x, -x, +y, *and* -y directions, respectively. If there is a link in the direction, the value is 1, otherwise, the value is 0. A neighbor matrix $N_{SA1}(SA_{+x},SA_{-x},SA_{+y},SA_{-y})$ defines the neighbors of this cell in eachdirection, respectively. For example, the value $SA_{+x}$is the cell number connected to cellSA_1in the +x direction so SA_{+x} equals to SA_3. Then the group G_{SA1} of cell SA_1 is expressed as:

$$G_{SA1}= SA_1+L_{SA1}*N_{SA1}^T =SA_1+ (1010) * (SA_3 0\ SA_2\ 0)^T= SA_1+SA_2+SA_3 \quad (1)$$

Since the cell$SA_3$is included in the group, its neighbors which have a link to SA_3 may be visible if the navigator reviewsSA_1. Thus the group G_{SA3} has to be calculated and included in the group G_{SA1}. This calculation is recursively used until no further cell is included in the group G_{SA1}.

Fig.2 (b) shows a link P_1's PVS group. It has neighbors in the +y and -ydirections. LinkP_1's PVS depends on its neighbors' property. If a subassemblyhas a link to other subassembly, it will be included in the PVS group. The cellsSA_1 SA_2 SA_3toP_1 have link matrixes $L_1(0,0,0,1)$, $L_2(0,0,0,1)$, $L_3(0,0,0,1)$. The link property of SA_1withP_1 can be expressed as $L_1(1,4)$=1, which is the value at the first row, forth column of matrix $L_1(0,0,0,1)$. If this value equals to 1, this cell group should be included in the PVS group of P_1. For example, the P_1's +y PVS groupcan be calculated as:

$G_{P1+y}= L_{P1+y}*N_{P1+y}^T=(L_1(1,4)\ L_2(1,4)\ L_3(1,4))(G_{SA1}\ G_{SA2}\ G_{SA3})^T=(1,1,1)\ (G_{SA1}\ G_{SA2}\ G_{SA3})^T= G_{SA1}+ G_{SA2}+ G_{SA3}$(2)

The P_1's - y PVS group can be calculated as:

$G_{P1-y}= L_{P1-y}*N_{P1-y}^T=(L_4(1,3), L_5(1,3), L_6(1,3), L_8(1,3))(G_{SA4}\ G_{SA5}\ G_{SA6}G_{SA8})^T =(1,1,1,1)\ (G_{SA4}\ G_{SA5}\ G_{SA6}\ G_{SA8})^T= G_{SA4}+G_{SA5}+ G_{SA6}+G_{SA8}$(3)

P_1'sPVS group is the sum of P1 and its 4 direction links as follows:

$G_{P1}= P_1+ G_{P1+x}+ G_{P1-x} + G_{P1+y}+ G_{P1-y}$(4)

Besides the cells included in the PVS group, there are some surfaces of other parts which are potentially visible to the navigator. Here the decision making is based on the importance of these surfaces to get a reasonable view. For example, P_2 and P3 that linkP_1, are included in the PVS group to decrease the effect of sudden switch from one cell to another. Removing the repeated nodes from the final PVS group, the whole group can be formed and ready to create a prototype for dynamical loading and unloading in real-time simulation.

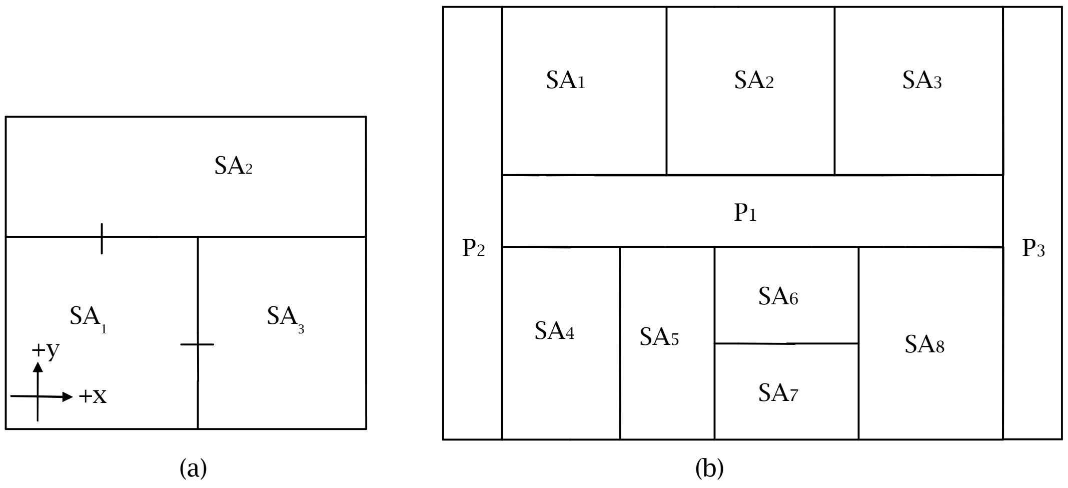


Fig.2:(a) SA_1's PVS group, (b) P_1's PVS groups.

3.3 Dynamic Loading Strategy

Thedynamic loading is used to load, unload the cell prototypesdynamically during simulation. All loaded models are stored in a prototype library.*DynamicPrototype* nodeloads and viewsthe model dynamically. This node has a *PrototypeName* field that holds the name of the modelprototype of library files. There is a default prototype tobe loaded upon the start of the simulation.

During the simulation, the *PrototypeName* field can use the corresponding prototypefilename, which means that a new prototype will be loaded and inserted into the running of the simulation. The loading order is controlled using a script code to track current viewport. When the navigator moves to the link of another region, the system will search the database and find out which region should be loaded. In the meantime, the previous region is unloaded.

Fig.3 shows the flowchart of the dynamic loading. The position of the viewport is sent to a script code. In the script code, whenever it receives a new position (*x, y, z*), it will determine whether this position relates to anlink, if yes, it will search the corresponding prototype in the database according to current prototype and the link number. The new prototype will then be loaded and old prototype will be unloaded. If the position is not located in any link, the process will continues.

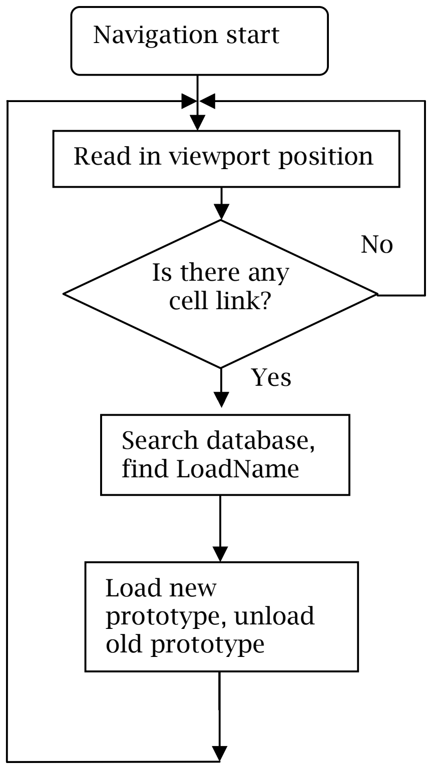

Navigation start
Read in viewport position
Is there any cell link?
No
Yes
Search database, find LoadName
Load new prototype, unload old prototype

Fig. 3: Flowchart of the dynamic loading.

4 CASE STUDY

4.1 Dynamic Loading for a Product Assembly Simulation

A product assembly model called six motor drive assemblies is used for this case study. In the pre-processing stage, the product assembly is divided into subassemblies based on the function and structure. The motor drive assembly is divided as a pneumatic shifting mechanism, a transmission, a dual motor, and an output shaft subassembly, where the last three subassemblies are located in the left and right sides. There are seven subassemblies in total. A subassembly consists of a group of parts. Fig.4 shows the product assembly model and its seven subassemblies.

Each subassembly is defined as a visual region, the region division based on subassemblies and their constituted parts are shown in Fig.5.Two subassemblies may be connected to each other by one or several parts. The connecting parts are essential to show the adjacency of these regions. Fig.6 shows the adjacency graph of these regions by connecting parts. For example, region 1 is connected with region 2 by part 27. The connecting parts are included in the initial product assembly to demonstrate the adjacency between regions.

In order to provide a general product assembly, the initial scene includes a simplified product assembly consisting of several link parts. The link parts are identified based on its profile and function. The parts forming the outer profile of a subassembly and the connection parts are identified as link parts. For example, Twolink parts are identified in region 1 including parts 115 and 27. Part 115 is assigned as a link part because of its big size. Part 27 is chosen since it is a connecting part. All link parts in each region form a simplified product assembly model for the initial display. A click sensor is attached to link parts for responding to user's action.

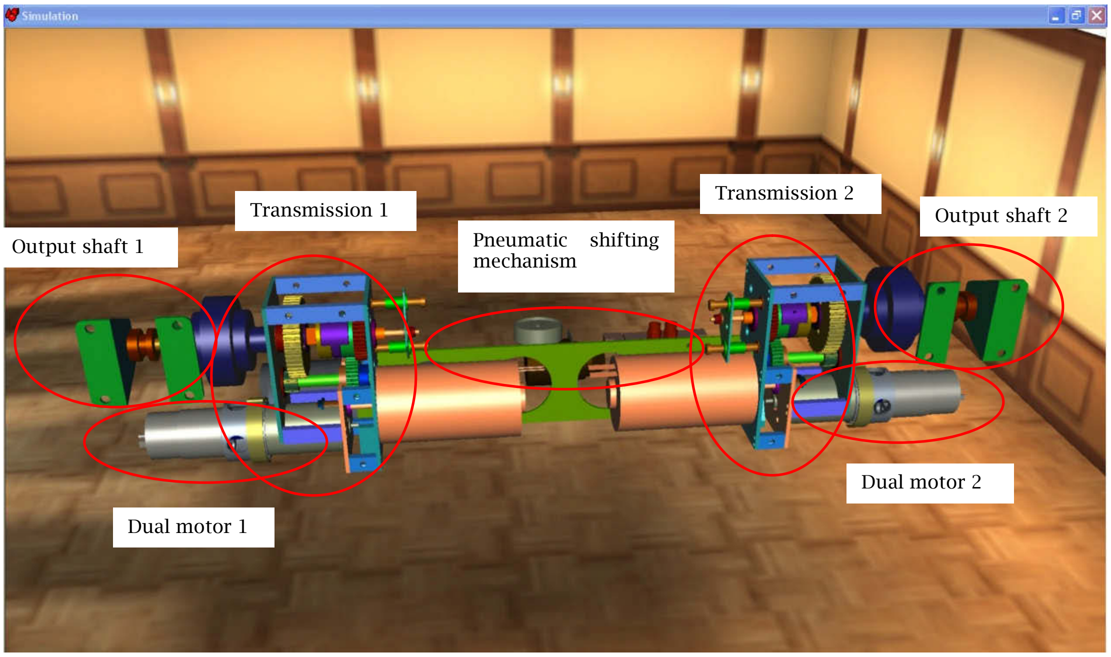


Fig.4: Motor drive assembly.

A region is determined based on the parts included in a subassembly. Since link parts have been included in a simplified product assembly model, they are excluded from the region formation. The remaining parts in each region are saved in prototype files and are stored in a prototype library. Tab.1shows all link parts identified in each region and the prototypes generated.

These link parts form a simplified product assembly model shown in Fig.7. The details of each subassembly are omitted from the simplified product model in Fig 7. But the main structure and connection relationships are clear. A user may proceed to examine the detailed structure of a subassembly by clicking alink part.

The creation of prototypes follows five steps: (1) import models into the VE, configure the materials and actions; (2) copy relevant nodes into the same model frames, such as mesh, and materials; (3) Right click the model frame, select “Create Prototype”, a prototype will be created; (4) in the New Component window, click New, save it as a prototype file;and (5) move the generated prototype from a local prototype window to a new component window.

During the model simulation, the initial scene displays a simplified product assembly without details of most parts. Only link parts are included in the simplified product assembly. These link parts are selectable. When a link part is selected by a user, details of the corresponding visual region will be dynamically loaded. In the meantime, the scene will navigate automatically to a closer viewport for the observation.

Tab.2 shows the relationship of link parts and their dynamic loading prototypes. For instance, if a user selects the part 115 in the product assembly level, the corresponding prototype “R1.eop” will be loaded. The scene will automatically move to that region for a detailed view. When a user selects the part 115 again, the scene will restore to its initial product assembly model. As soon as another link part is selected, a current prototype will be dynamically replaced with a new prototype.

The dynamic loading is implemented using a *Script*node and a *DynamicPrototype* node. The *ClickSensor* sends the part name to the *script*. In the *Script*, the required loading region is found based on the selected part. This region’s name is then sent to the *PrototypeName* field of *DynamicPrototype* node, and the prototype is loaded automatically. The routing is as follow:

ClickSensor.Target->Script.ObjectName
Script.RegionName->DynamicLoading.PrototypeName

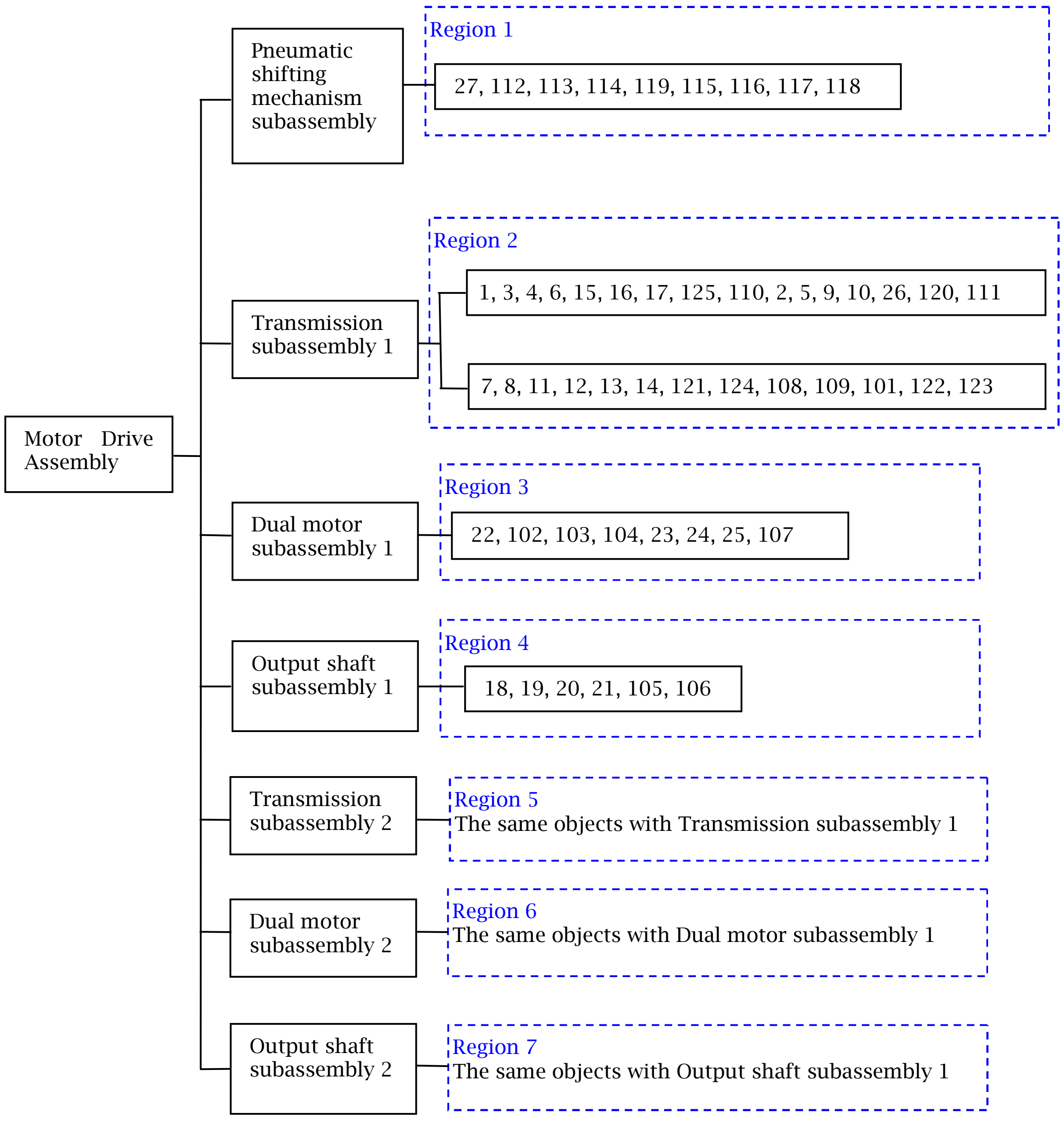


Fig.5: Region division and its constituted parts.

When a subassembly is loaded, the scene will automatically navigate to the newly loaded subassembly. An *ObjectNav*prototype is assigned to each subassembly. *ObjectNav* is a prototype that allows a user to rotate, zoom, and pan a 3D object using mouse or mouse and key combinations. It automatically resets the camera when *ObjectNav* becomes active by presetting the camera's coordinates. *ObjectNavManager*prototype controls the shifting between *ObjectNavs*. It turns off the previous *ObjectNav* when a new *ObjectNav* becomes active. Fig.8 shows the system implementation in EON Studio [26]. The left side is the routing between different nodes and the *Script* code. The middle is the simulation tree of all nodes and models.

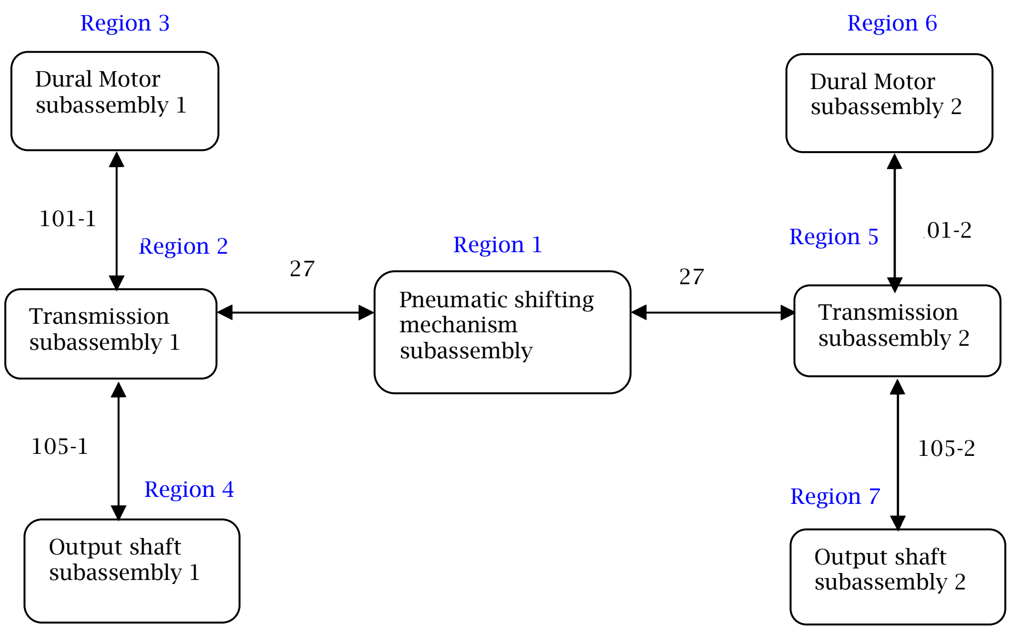

Region 3
Dural Motor subassembly 1
101-1
Region 2
27
Region 1
27
Region 5
01-2
Region 6
Dural Motor subassembly 2
Transmission subassembly 1
Pneumatic shifting mechanism subassembly
Transmission subassembly 2
105-1
Region 4
Output shaft subassembly 1
105-2
Region 7
Output shaft subassembly 2

Fig.6: Adjacency graph representation of subassemblies.

Region No.	Link parts	Parts in Prototype	Prototype Name
R1	115, 27	The rest parts of R1	R1.eop
R2	1-1, 2-1, 3-1, 4-1, 5-1, 6-1, 101-1	The rest parts of R2	R2.eop
R3	None	All parts of R3	R3.eop
R4	18-1, 105-1	The rest parts of R4	R4.eop
R5	1-2, 2-2, 3-2, 4-2, 5-2, 6-2, 101-2	The rest parts of R5	R5.eop
R6	None	All parts of R6	R6.eop
R7	18-2, 105-2	The rest parts of R7	R7.eop

Tab. 1:Link parts in the simplified model and prototype forming.

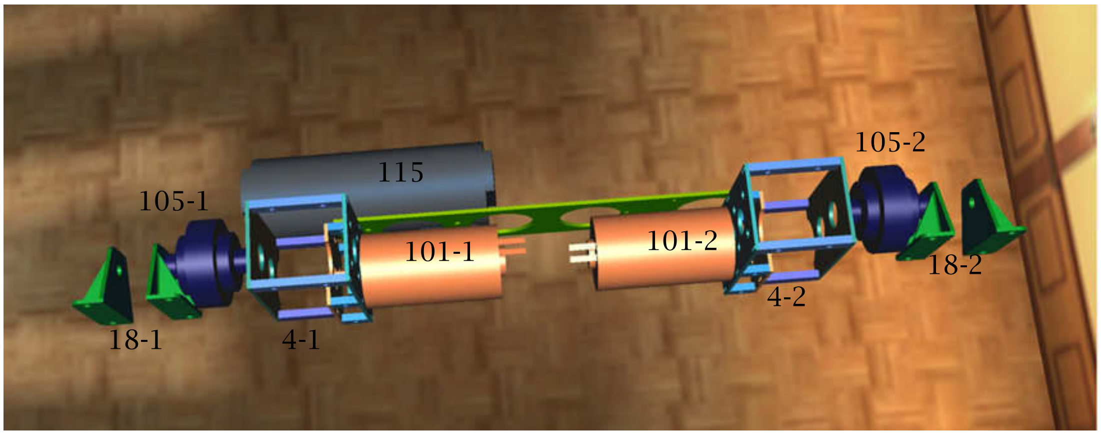
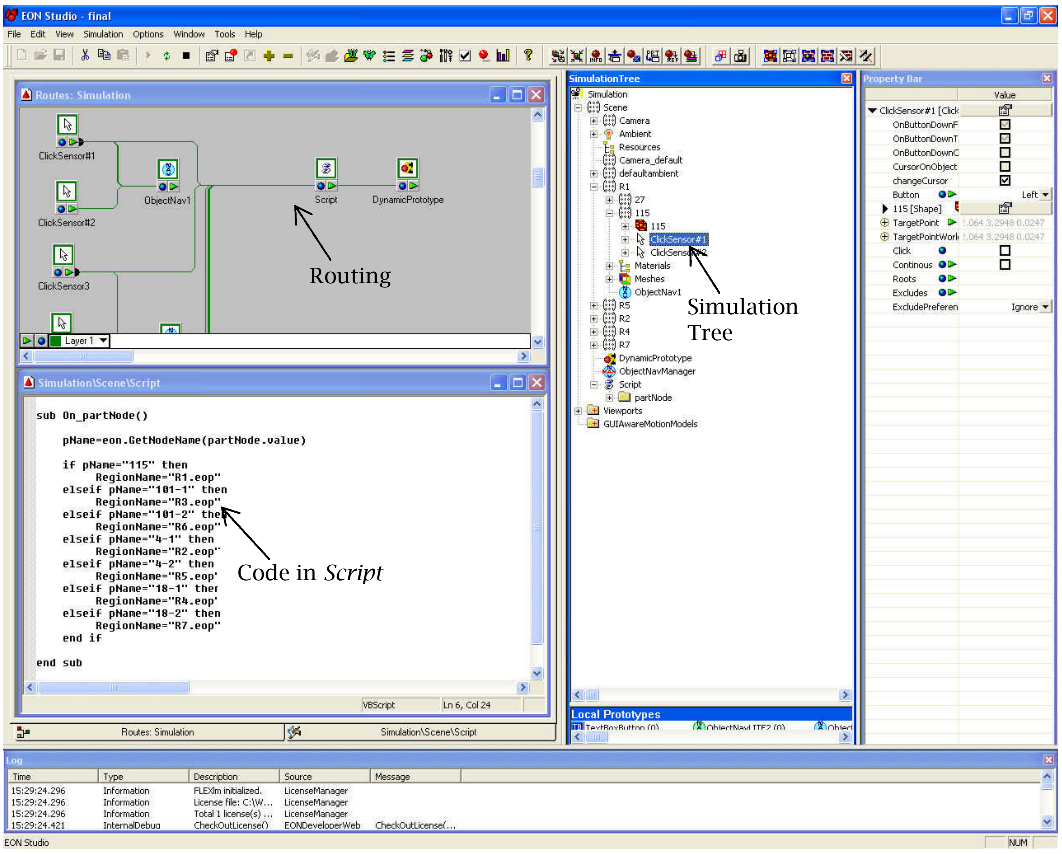


Fig.7: Simplified product assembly model with link parts.

Part Name	Prototype Name
115	R1.eop
4-1	R2.eop
101-1	R3.eop
18-1	R4.eop
4-2	R5.eop
101-2	R6.eop
18-2	R7.eop

Tab.2: The relationship of link parts and dynamic loading prototypes.



```
sub On_partNode()

    pName=eon.GetNodeName(partNode.value)

    if pName="115" then
         RegionName="R1.eop"
    elseif pName="101-1" then
         RegionName="R3.eop"
    elseif pName="101-2" then
         RegionName="R6.eop"
    elseif pName="4-1" then
         RegionName="R2.eop"
    elseif pName="4-2" then
         RegionName="R5.eop"
    elseif pName="18-1" then
         RegionName="R4.eop"
    elseif pName="18-2" then
         RegionName="R7.eop"
    end if

end sub
```

```
Time            Type            Description            Source              Message
15:29:24.296    Information     FLEXlm initialized.    LicenseManager
15:29:24.296    Information     License file: C:\W...  LicenseManager
15:29:24.296    Information     Total 1 license(s) ... LicenseManager
15:29:24.421    InternalDebug   CheckOutLicense()      EONDeveloperWeb     CheckOutLicense(...
```

Fig.8:Implementation of the dynamic loading in Eon Studio.

The cell segmentation based dynamic loading for the product assembly shows the performance improvement in the VE navigation. The performance measure uses the number of polygons and the frame rate in the real-time simulation. Fig.9 shows the dynamically loaded regions based on user's selection. In Fig.9 (a), the parts in region 1 are dynamically loaded when part 115 is selected. In Fig.9 (b), the initial transmission model is a frame without the inside parts. When part 4-2 is selected, the parts in region 5 are loaded including gears, axes, and dog mate. Fig.9 (c) shows the loaded dual motors. Fig. 9 (d) shows the loaded parts in output shaft 2.

Tab.3 shows a comparison of the number of polygons and frame rate before and after the cell segmentation. There are 65,484 polygons in the original assembly. The number of polygons is reduced to 32,816 in the simplified assembly, which accounts to 50% reduction of polygons. The frame rate increases from 50 fps to 63fps, a 26% performance improvement. When a region is dynamically loaded, the polygon's reduction rate ranges from 23% to 47%, and the frame rate increases up to 20%, comparing to the original assembly. It shows that the cell segmentation and dynamic loading decreases the number of polygons in the computer's memory, thus the navigation performance is improved.

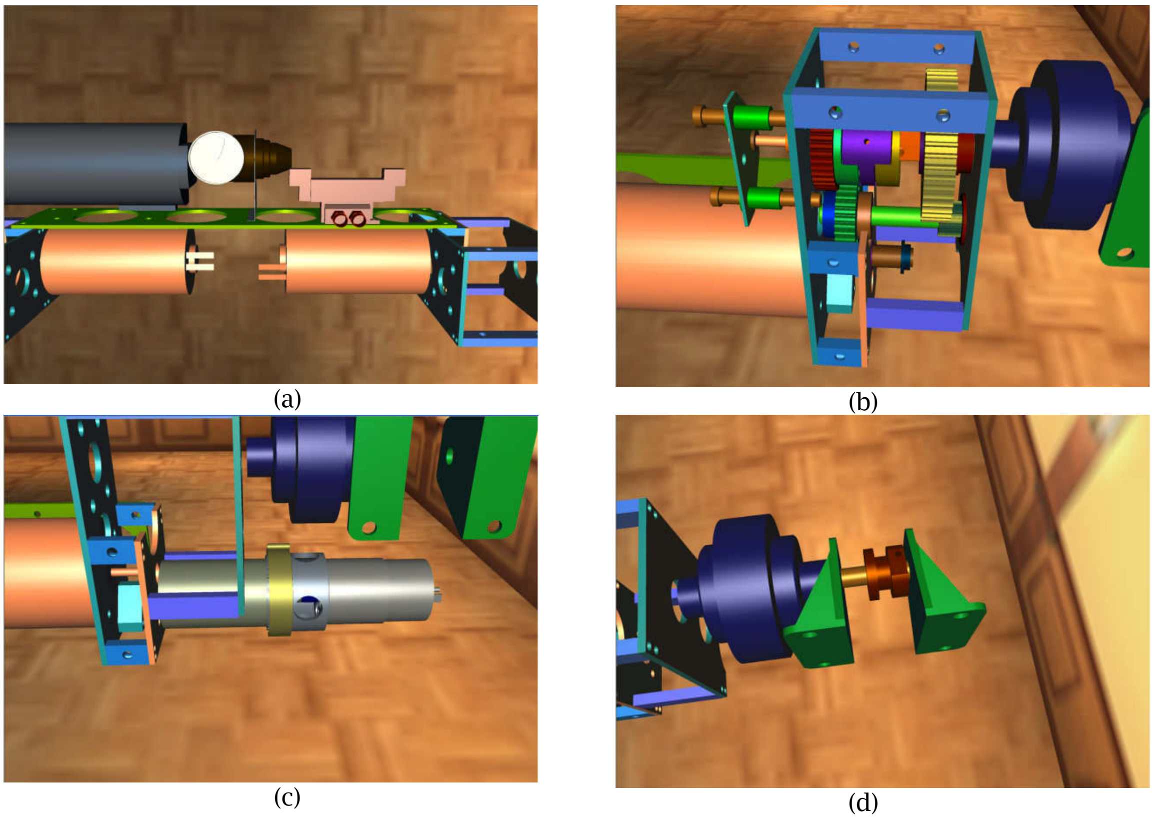

(a) (b) (c) (d)

Fig.9:Four dynamic loaded regions based on a user's selection (a) the transmission (b) the pneumatic shifting mechanism (c) the dual motor (d) the output shaft.

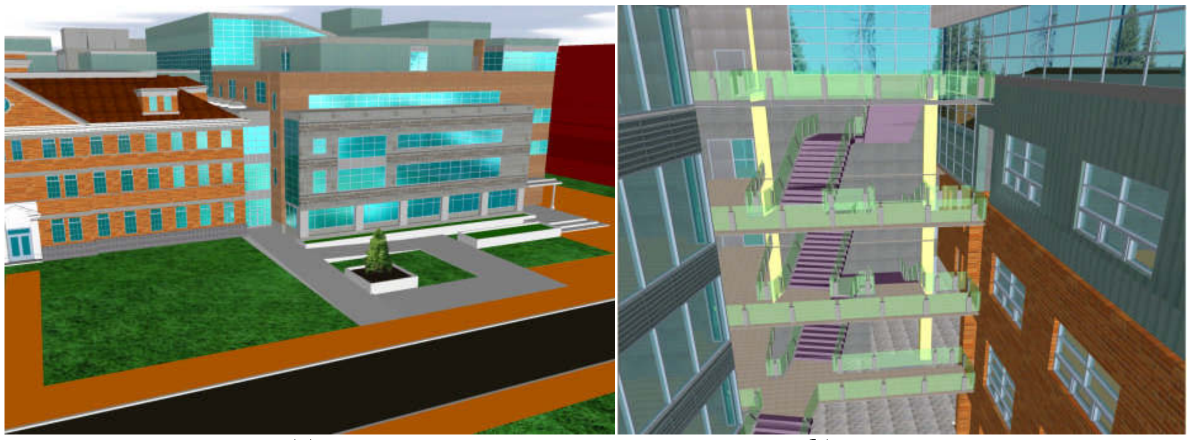
Models	Number of polygons	Reduction rate	Frame rate (fps)	Frame rate increased
Original assembly	65484	0	50	-
Simplified assembly	32816	50%	63	26%
Simplified assembly + R1	36976	44%	59	18%
Simplified assembly + R2	50610	23%	55	10%
Simplified assembly + R3	42580	35%	57	14%
Simplified assembly + R4	34616	47%	60	20%

Tab.3: Comparison of the number of polygons and frame rate.

4.2 Dynamic Loading for a Building Model Simulation

In order to test the performance of the proposed method, a much larger model is used. The proposed method is tested usinga building model. The Engineering Buildings E1 and E2 at the University of Manitoba, shown in Fig. 9, are segmented and then grouped into PVS regions according to the cell properties.

Tab.4 shows a comparison of non-segmented models and segmented models of Building E1 and E2. The test is conducted in a HP XW4400 Workstation, with Intel (R) Core (TM) 2 CPU 6700@2.66 GHz, and 2.0GB RAM. The segmented modelsare simplified into smaller models that only include external walls, atrium, ceilings, railings, and a default region. Separated regionsare dynamically loaded according to navigator's position. The number of polygons is read from the *PolygonCountAfterReduction* in the *Meshes* node. The frame rate is read from the*StatisticData*under *Simulayion* node. For the Building E1, the number of polygons in non-segmented models is 181,408, while the number of polygons in segmented models decreases to 60,472. The reduction rate of E1 is 67%. The frame rate increases by 71%, from 62 fps to 106 fps. For Building E2, the number of polygons in non-segmented model is 297,124, while the number of polygons in segmented models decreases to 65,976. The reduction rate of E2 is 78%. The frame rate increases by 88%, from 49 fps to 92 fps. It shows that the simulation performance is improved after using the cell segmentation method.



(a) (b)

Fig.9: Building models, (a) Exterior of Engineering Building E1 (left) and E2 (right), (b) Interior of Engineering Building E1 (right side) and E2 (left side).

Models of Building		Number of Polygons	Reduction Rate	Frame rate (fps)	Frame rate increased
E1	Non-segmented model	181,408	-	62	-
	Segmented model	60,472	67%	106	71%
E2	Non-segmented model	297,124	-	49	-
	Segmented model	65,976	78%	92	88%

Tab.4: Comparison of polygons and frame rates for non-segmented and segmented models.

5 CONCLUSIONS AND FURTHER WORK

This paper presentedanimproved navigation and dynamic loading method in VEs. For the product simulation, regions were formed based on subassembly divisions and visibility. Cell properties were used to form regions according to the potential visibility during simulation. Dynamic loading strategy was applied using real-time monitoring of viewers' position or conducted by user's selection on some predefined link parts. The developed methods are very useful in the navigation of large-scale VEs. It has advantages of reduced computer memory usage and improved navigation performance.

Two case studies were conducted. The frame rate was improved significantly for large models. By loading separate regions, the number of polygons in simulation can be greatly decreased while maintaining the visulization performance of VEs. Although the real-time monitoring of viewers' position takes time leading to lower performance than what was expected, the size of loaded region is much smaller than the whole model.

A problem in the dynamic loadingis the sudden switch between different regions when the model is loaded and unloaded. One possible solution is to includemore polygons in the overlapped area.

Another strategy isto maintain two adjacent regions simultaneously, and to load a newregionbefore the navigator approaches the region. It is promising to dynamically generate regions and link parts based on user's viewpoint in VEs.

ACKNOWLEDGMENTS

This research is supported by the Discovery Grants of Natural Sciences and Engineering Research Council of Canada (NSERC), and NSERC Postgraduate Scholarships.Thanks Mr. MawuliGavifor modeling of the motor drive assembly.

REFERENCES

[1] Just, C. D.: Performance analysis of a virtual reality development environment: Measuring and tooling performance of VR Juggler, Master thesis, Iowa State University, 2000.
[2] Yang,X.B.; Choi,S. H.; Yuen K. K.; Chan,L. K. Y.: An Intuitive Human-Computer Interface for Large Display Virtual Reality Applications, Computer-Aided Design & Applications, 7(2), 2010, 269-278.
[3] Klein, R.:Data preparation for real-time high quality rendering of complex models, Computer Graphics Forum, 25(3),2006, xviii.doi:10.1111/j.1467-8659.2006.00943.x
[4] Debattista, K.; Chalmers, A.; Gillibrand, R.; Longhurst, P.; Mastoropoulou, G.; Sundstedt, V.: Parallel selective rendering of high-fidelity virtual environments, Parallel Computing, 33, 2007, 361–376.doi:10.1016/j.parco.2007.04.002
[5] Allard, J.; Raffin, B,:Distributed physical based simulations for large VR applications, Proceedings - IEEE Virtual Reality, 2006, 12.
[6] Liu, S.Q.;Ong, S.K.;Chen, Y.P.;Nee, A.Y.C.: Real-time, dynamic level-of-detail management for three-axis NC milling simulation, Computer-Aided Design, 38, 2006, 378–391.doi:10.1016/j.cad.2005.11.003
[7] Hagedorn, B.; Trapp, M.; Glander, T.; Dollner, J.: Towards an indoor level-of-detail model for route visualization, 2009 Tenth International Conference on Mobile Data Management: Systems, Services and Middleware, 2009, 692-697.
[8] Chu,C-H.;Wu,P-H.; Hsu Y-C.: Multi-agent collaborative 3D design with geometric model at different levels of detail, Robotics and Computer-Integrated Manufacturing, 25, 2009, 334–347.doi:10.1016/j.rcim.2007.01.005
[9] Zhang, J.; Fei, L.; Chen, Z.:Quadtree of TIN: a new algorithm of dynamic LOD, Proceedings of the SPIE - The International Society for Optical Engineering, 7492, 2009, 749210 (9 pp.).
[10] 10Zhang, S.; Wu, E.: An improved method for generating multiresolution animation models, Proceedings - 2009 11th IEEE International Conference on Computer-Aided Design and Computer Graphics, CAD/Graphics 2009, 58-61.
[11] Dmitrieva, J.; Verbeek, F.J.: Different geometries in ontology visualization,Proceedings of the SPIE - The International Society for Optical Engineering, 7530, 2010, 753007 (12 pp.).
[12] Su, Z.; Xia, M.; Li, W.; He, T.;Tang,W.: Feature-based simplification of process plant models over network, International Journal of Virtual Reality, 8(2), 2009, 51-58.
[13] Zhu, Q.; Zhao, J.; Du, Z.; Zhang, Y.: Quantitative analysis of discrete 3D geometrical detail levels based on perceptual metric, Computers & Graphics, 34(1), 2010, 55-65.cdoi:10.1016/j.cag.2009.10.004
[14] Li, M.; Zhang, Y. F.; Fuh,J. Y. H.:Retrieving Reusable 3D CAD Models Using Knowledge-Driven Dependency Graph Partitioning, Computer-Aided Design & Applications, 7(3), 2010, 417-430.
[15] Bossche, B.;Vleeschauwer,B.; Verdickt,T.; Turck,F.;Dhoedt,B.; Demeester, P.: Autonomic microcell assignment in massively distributed online virtual environments, Journal of Network and Computer Applications, 32, 2009, 1242–1256.doi:10.1016/j.jnca.2009.04.001
[16] Wang, Y.; Lees, M.;Cai, W.; Zhou, S.; Yoke, M.; Low, H.: Cluster based partitioning for agent-based crowd simulations, Proceedings of the 2009 Winter Simulation Conference, 1047-1058.
[17] Kojima, K.; Nagasaki, M.; Miyano, S.: An efficient biological pathway layout algorithm combining grid-layout and spring embedder for complicated cellular location information, BMC Bioinformatics, 2010, 10.1186/1471-2105-11-335.doi:10.1186/1471-2105-11-335

[18] Zhang, X.:A multiscale progressive model on virtual navigation, Int. J. Human-Computer Studies, 66, 2008, 243–256.doi:10.1016/j.ijhcs.2007.09.004
[19] Ge, J.; Sandin, D. J.; Johnson, A.; Peterka, T.; Kooima, R.; Girado, J. I.; Defanti, T. A.: Point-based VR visualization for large-scale mesh datasets by real-time remote computation, Proceedings - VRCIA 2006: ACM International Conference on Virtual Reality Continuum and its Applications, 2006, 43-50.
[20] Merchán, P.; Adán, A.: Exploration trees on highly complex scenes: A new approach for 3D segmentation, Pattern Recognition, 40, 2007, 1879- 1898.doi:10.1016/j.patcog.2006.11.017
[21] Airey, J.M.: Increasing update rates in the building walkthrough system with automatic model-space subdivision and potentially visible set calculations, PhD thesis, University of North Carolina, 1990.
[22] Bittner, J.;Havran,V.; Slavik, P.: Hierarchical visibility culling with occlusion trees, Proceedings of Computer Graphics International'98, 1998, 207-219.
[23] Leyvand, T.;Sorkine, O.; Cohen-Or.D.: Ray space factorization for from region visibility, ACM SIGGRAPH, 2003, 595-604.
[24] Roden, T.;Parberry, I.: 2005. Portholes and Planes: Faster Dynamic Evaluation of Potentially Visible Sets,ACM Computers in Entertainment, 3(2), 2005, 1-9.
[25] Luebke, D.; Georges, C.: Portals and mirrors: Simple, fast evaluation of potentially visible sets.Proceedings of the Symposium on Interactive 3D Graphics, 1995, 105-106.
[26] Eon Studio, Eon Reality Inc. www.eonreality.com, 2010.