Journal of Naval Science and Engineering
2015, Vol. 11, No. 3, pp. 43-74

SURVEY ON VISIBILITY AND DATA DISTRIBUTION IN DISTRIBUTED VIRTUAL ENVIRONMENTS

Yekta KILIÇ[1]
Gürkan KOLDAŞ[2]
Şafak Burak ÇEVİKBAŞ[3]

[1]*Naval Science and Engineering Institute, Turkish Naval Academy, Istanbul, yektakilic2011@gmail.com*
[2]*Computer Engineering Department, Turkish Naval Academy, Istanbul, gurkankoldas@gmail.com*
[3]*Computer Engineering Department, Middle East Technical University, Ankara, safakburak@gmail.com*

Abstract

The importance of data distribution increases in Multi-User Distributed Virtual Environments (DVEs) in parallel to both the complexity of virtual scenes and the number of clients. Main challenges are to load the relevant part of the scene, estimate and render visible primitives by keeping the frame rate fluent, since each user sees different part of the shared scene. In order to achieve the frame rate goal, data distribution should be well managed and optimization approaches based on network and visibility should be applied according to the requirements of each DVE. This is substantially handled by considering the area of interest for each user and estimating visible primitives as early as possible. This paper surveys the research on visibility culling, data management based on area of interest in DVEs and consequently summarizes thirteen implementations in the literature.

Keywords: *distributed virtual environments; data distribution management; visibility culling; area of interest*

1. INTRODUCTION

Nowadays, everyone has a computer or smart devices such as smart phones, tablets, TVs, game consoles with the recent developments in technology. People may interact with anyone in the world with these devices and share documents, videos, pictures, programs, ideas etc. Multi-Player Online Games and Simulations in DVEs1 increase with the broad bandwidth usage in the Internet access in addition to the number of implementations on shared portals, forums, instant messaging, teleconference etc. With an exciting new hardware, outstanding creativity and software, game industry explode in the last decade. Especially, distributed multi-player online games among the teenagers guide to firms in the field. To take the attention of the customers, firms aim to increase the immersion with more accurate 3D models and complex scenes while keeping interactivity and frame rate fluent. This is still the main challenge of distributed games or virtual environments.

In this paper we focus on 3D data distribution management approaches considering visibility. A consistent visibility and interaction on the remote user is the main purpose of multi-player online games like stealth games. In such games, some optimization methods should be applied to achieve the frame rate goal since there are usually constraints on network bandwidth, hardware and software properties of both server and clients or data sets those will be distributed.

As a course of DVEs' nature, improvements have been done in different domains like network, distributed simulations and graphics. In network, different network topologies have been implemented such as broadcasting, multicasting or centralized server approach [1]. In addition, High Level Architecture provide a Data Distribution Management that is implemented over Run-Time Infrastructure for distributed simulations [2], [3]. In computer graphics, temporal and spatial subdivisions of scenes and

[1] In this paper, the term of DVEs is used as general term for Distributed Virtual Environments, Networked Virtual Environments, Multi-User Online Games and Distributed Virtual Simulations. Details and differences between them are not discussed.

visibility algorithms like occlusion culling are utilized for this purpose [4].

Visibility has been remarkable subject over the last few decades. The aim of the visibility is to decide whether an object is visible from a given viewpoint in the viewing direction, so that invisible ones would be culled. It is basically performed by using traditional depth buffer, but more efficient methods should be used in order to increase performance in much more complex scenes and situations which have large data set [4].

In terms of efficiency, there is an important similarity between visibility and data distribution. In visibility, it is aimed to reject invisible polygons as early as possible so that either CPU or GPU wouldn't be occupied in working on them [4]. Likewise, in DVEs, if invisible or unrelated data are not distributed to clients, network performance is improved [5]. Because of this similarity in conceptual view, some of methods related to the data distribution takes the advantage of visibility culling approaches.

Although there are large numbers of studies on individual topic of DVEs, data distribution and visibility, there is not any study which covers both visibility and data distribution. In consequence of this motivation, we aim to survey data distribution approaches considering visibility methods for consistent visibility and interaction in DVEs. As illustrated in Figure 1, general purpose of this paper is showing that network bandwidth is used more efficiently by using data distribution and interest management (IM) techniques along with visibility algorithms.

The rest of the paper is organized as follows. Firstly, we compare our paper with related studies in section II. Then, we describe visibility culling in section III and basics of data distribution and IM in terms of DVEs in section IV. In section V, data distribution management approaches in DVEs which take advantage of visibility techniques are discussed. Finally in section VI, we conclude our survey.

2. RELATED SURVEYS

Our paper contains three different topics as visibility, data distribution and IM each of those have been studied in several surveys.

In the survey of Cohen-Or et al. [4], they classify visibility algorithms as point-based and from-region and then summarize individual approaches in terms of walkthrough applications. Bittner and Wonka [6] have also studied and compared visibility algorithms however they present new taxonomy

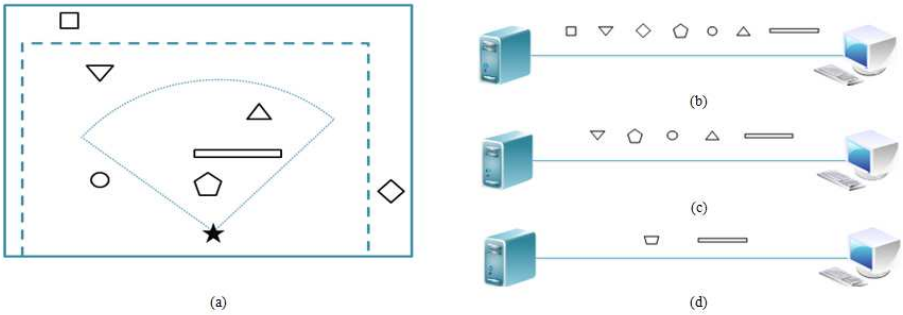


Figure 1. (a) A scene representation seen in the server. Filled star represents a client, as dashed line do Area of Interest (AoI) and dotted line do Viewing Frustum. (b) In brute force approach all geometries are sent to the client from the server that causes occupying network bandwidth a lot. (c) In case of considering AoI of the client, some geometries (diamond and square) outside the AoI are not sent. (d) When visibility culling computations are taken into account, more geometries are also culled and network usage is relatively decreased. In this sense, reverse triangle and circle culled as a result of viewing frustum test, triangle is culled because of being occluded by rectangle and some part of pentagon is culled since back-face culling is performed.

according to problem domain. Both of these studies focus on visibility algorithms on discrete systems and do not consider distributed virtual applications and data distribution or IM.

Boulanger et al. [7] survey IM techniques for massively multiplayer games and compare eight algorithms. Carter et al. [8] discuss area of interest management (AoIM) and the distribution and communication protocol to be considered when building P2P massive multiplayer online games. These surveys cover interest and data distribution management methods in DVEs while they don't take visibility computations into consideration. In Liu and Theodoropoulos' detailed survey on IM [9], they give fundamentals about DVEs and classify IM algorithms into six categories one of those is visibility-based. In their survey some IM algorithms which use visibility computations are discussed some of those are also explained in our paper.

There are several other works on DVEs which survey network overlays, architectures and designs [10], [11], [12]. These don't contain neither visibility nor IM algorithms, only focus on network properties such P2P overlays.

3. VISIBILITY

The topic of visibility emerged in 1960s on the purpose of determining visible lines of surfaces [6]. As well as hidden surface removal (HSR) algorithms, in 1975 z-buffer algorithm was presented by Catmull and has been widely acclaimed for long years [13]. With the increase in data sets in the last few decades, traditional HSR and z-buffer algorithms remained incapable. Therefore, new visibility culling algorithms have been implemented to fulfill the need in different problem domains [4], [6].

It is aimed to identify visible or invisible parts of scene as early as possible in visibility algorithms [14]. Thus some computations and tests are performed to determine whether a polygon is visible or not. Fundamentally, there are three technique for this purpose: Back-face culling, occlusion culling and viewing-frustum culling. These techniques constitute the base of visibility culling algorithms where a polygon is determined as invisible if it faces away from the viewpoint, is occluded by another part of the scene, or is outside the current viewing frustum respectively as seen in Figure 2 [4], [15].

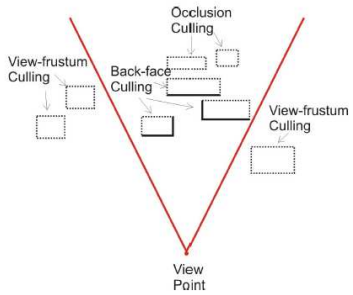


Figure 2. Types of visibility culling methods. A polygon is determined as invisible if it faces away from the viewpoint (Back-Face Culling), is occluded by another part of the scene (Occlusion Culling), or is outside the current viewing frustum (View-Frustum Culling).

Human visual system perceives pictures at higher frame rates more than 10-12 frame per second (fps) as a moving pictures. For example silent films has frame rates between 16 and 24 fps while standard filming and projection format are 24 fps in motion film industry [16]. Digital video and television formats support more than 24 fps where most of games and simulations requires at least 30 fps for interactivity and fluent rendering. Thus, determining visible parts and rendering them should be performed in less than 1/30 sec. to obtain frame rates more than 30 fps. This is the most desired objective to reach in computer graphics applications and games [17]. To accelerate this process, most of the approaches offered in visibility culling aim to find potential visible set instead of exact one. Exact visible set (EVS) contains all polygons which are at least partially visible for a given viewpoint while potential visible set (PVS – in literature also called conservative visibility set) may include some invisible polygons in addition to all exactly visible ones [4], [17], [18]. Obtaining EVS in one shot is costly, slow and underperformance in most applications which has complex virtual scene. Some of the implementations uses graphics hardware to estimate the EVS with the help of depth buffer and send all the primitives in the scene to the graphics hardware. If the virtual scene is more complex and

larger than the memory capacity of graphics hardware, then a bottleneck occurs during the translation between main memory and graphics hardware during the implementation. For that reason, the preferred strategy is to estimate PVS by culling most of invisible objects as early as possible in graphics pipeline in low cost initially, and to remove the relatively small amount of invisible primitives in the second step by determining EVS in previously overestimated PVS. The main goal in this approach is to estimate PVS as much as close to EVS in optimum cost and save up time rather than processing all primitives with respect to the requirements of implementation. Thus, the quality of the PVS that is related to size of invisible polygons is crucial matter [17].

A user usually views only small portion of whole scene in complex and large environments such as urban. Working on great deal of irrelevant and invisible data increases the computations and running time in an implementation. Visibility culling algorithms aim to be output sensitive. If an algorithm is output sensitive, its running time is proportional to process time of the size of its visible set instead of entire scene [4], [6]. In other words, the more the visibility algorithm is output sensitive, the faster it is.

Until this point, we mention about the goals and performance issues of a visibility algorithm in computer graphics implementations. We now explain the basic classification of visibility algorithms as point-based and region-based (or from region visibility) with respect to computations made for only current viewpoint or a defined region respectively [4], [19].

3.1. Point-Based Visibility Algorithms

In point based visibility culling algorithms, visibility computation is performed with respect to current viewpoint. Whenever viewer's location or looking direction changes, the number of visible primitive changes and new computations should be executed. This characteristic of point-based visibility algorithm necessitates to estimate EVS in each frame when user moves in the virtual environment. This is a significant drawback where the sequence of computations in runtime occupies hardware a lot and therefore limits or incapacitates the usage of these methods in DVEs.

Several methods has been presented to overcome point-based techniques' stated disadvantage. Most of them uses the occlusion culling approaches which estimate the invisible primitives behind the close ones which are called occluder. In the method that is based on selecting large convex occluders, Coorg and Teller offers tracking visual events to exploit temporal coherence [20]. In their subsequent work, they also proposed selecting occluders in preprocessing stage [21]. Hudsan et al. make some improvements about occluder selection to be used more efficiently [22]. In addition, additive individual approaches, hierarchical data structures and subdivision of scene such as octree [20], [23], [24], Shadow Volume Binary Space Partition [25], hierarchical z-buffer [23] and hierarchical occlusion map [17] are utilized.

Point-based methods are implemented in object or image space, or in both. In object precision methods, computations are performed on the basis of objects, whereas pixel-based processes are done in image-precision techniques [4]. Since image-precision methods are performed in rasterization stage, its performance is up to capacity and productivity of hardware. In some complex scenes where working on object would be costly, image-precision approaches may suit well according to the implementation requirements.

3.2. Region-Based (or From Region) Visibility Algorithms

Point-based visibility algorithms can be utilized on a small scale scenes where number of visible primitives are limited such as indoor scenes. Because of the weakness of computing visibility for each frame in point-based methods, they are not applicable to walkthrough applications in outdoor scenes where a lot of primitive may be seen from an opening. To overcome this drawback, rather than performing computations for a view point, PVS for a region is computed and utilized when the user is in the corresponding region. This way, minimizing processes executed in render time using computed PVS for the region puts region-based visibility algorithms forward. As well, most region-based approaches defines regions in precomputation and in this way prefetching is provided especially for

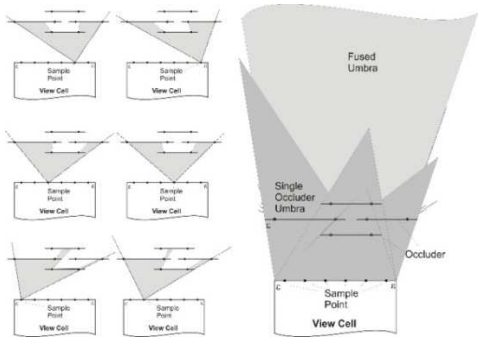
DVE implementation. On the other hand, these algorithms may need preprocessing phase and large size of memory for the overestimated PVS for potential regions or view-cells [4].

Commonly used region-based visibility method is cell-and-portal approach. It is suitable for indoor scenes where the environment is divided into cells such as rooms. Portals are the openings such as doors or windows defined between cells. Room walls bound cells where the user move in. Even though this approach is utilized in point-based visibility methods, the PVS of each cell is estimated by the neighborhood relation constructed by portals of the cell. It means that if a cell is seen by any portal of the corresponding cell, then the primitives of seen cell is added to the PVS. This approach is utilized both point-based and region-based visibility culling methods using neighboring relationship. In region-based visibility method, cells and portals are usually identified in preprocessing stage and for each cell possible visible adjacent regions are computed and added to PVS of each cell.

The most important benefit of cell-and-portal approach is prefetching. During run time, when a viewer transits to another cell, related scene information, which is the precomputed PVS of cell, is retrieved from storage. By means of prefetching, possible visible cells and thus transited cell information are known. Also, whenever user stays in a cell there is no need to make computation or retrieve data from memory. Using the precomputed PVS which is significantly a small of the entire scene, facilitates fluent display and prevents unwanted flickering of instant visible primitives. As we will discuss later, this prefetching approach is greatly desired in walkthrough applications in DVEs.

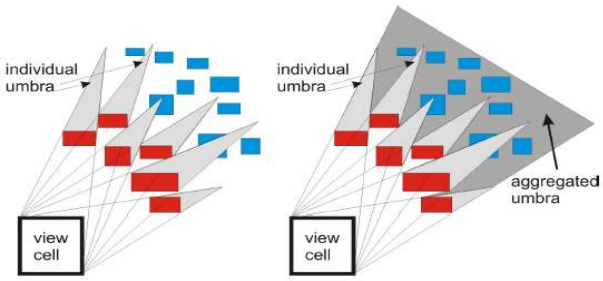
Though its momentous advantages, unfortunately, cell-and-portal algorithms are not well suited to outdoor scenes. Not only this method but also other approaches are difficult to implement effectively because this kind of scenes may contain enormous objects and polygons. Its reason is that a lot of primitives behind may be visible through the opening such as streets. To overcome this, generic techniques were developed which is

based on defining view cells and testing occlusion between the view cell and objects. Some methods utilizes occluder fusion approach to estimate the occluder shadow constructed by close and small occluders as seen in Figure 3. Then these methods use the occluder shadow to cull the invisible primitives in behind [15]. Some region-based methods utilizes sample points on the view cell to estimate PVS of the cell. The difficulty of estimating PVS from sample points is that an invisible primitive from the sample points may be visible between sample points as seen in Figure 4 [6], [15]. These methods use occluder shrinking or extended projection approach to in addition to occluder fusion overcome this problem [4], [15], [26].



(a) (b)

Figure 3. (a) Sampling of the occlusion from six sampling points. (b) The fused umbra from the six points is the intersection of the individual umbra.



(a) (b)

Figure 4. (a) Umbra of individual occluders. (b) Aggregated umbra.

4. DATA DISTRIBUTION IN DVES

In DVEs, users share a common environment and be aware of other participants via exchanging data such as position and orientation. Each distributed simulation implementation has different strategy for data distribution to obtain fluent and consistent simulation since latency between clients disturbs the user and reduce the interactivity and consistency of the simulation. It is not efficient to receive all data or messages those are flowing over network because of both discrete workstations and network constraints [27]. For this reason, data distribution whose task is managing data transfer has great importance to avoid delays in such systems. The importance of data distribution management increases especially on interactive graphics implementation such as stealth games since they are less tolerable to latencies on network and rendering.

In virtual applications, since visualization pipeline starts with retrieving data from storage [28], bottleneck in this stage would affect the rest of rendering pipeline. Thus, it is crucial for a remote user to have data on time prior to process. In terms of distribution time, as Hesina and Schmalstieg summarized [29], data can be distributed off-line (usually used in simulations and CD-ROM based games), before use (preloading) or on-the-fly (distributing individual objects during rendering). For a consistent simulation, the amount of data needed by any user should be estimated initially and then distributed to the corresponding client on time. These are the challenges of each DVE to be fulfilled considering the virtual environment, user's area of interest (AoI), hardware capability of server and client, network latency, number of users etc.

DVEs are implemented over two main network architectures: server-client and peer-to-peer (P2P) [30]. Hybrid architecture involving both server-client and P2P can also be used in some applications. In server-client, there is a centralized server where clients cannot communicate with each other directly but only through server. Management of server client architecture is easy, but it has some drawbacks like single point of failure, bottleneck on server, not being scalable and cost of running server. In

contrast to server-client, there is no centralized server in P2P architecture and clients can communicate with each other directly. This architecture is highly scalable, but it is difficult to manage and has a security vulnerability. The appropriate architecture is chosen with respect to the requirement of DVE by taking their advantages and disadvantages into consideration.

The objective of data distribution strategy in DVE is to utilize the limited resources such as network bandwidth, memory and process power of both server and clients more efficiently. In this way, we improve the scalability of the DVE and prevent possible errors caused by latency while keeping its consistency and interactivity. Apart from field of network, this can be accomplished by reducing message traffic by filtering them according to the interest of clients, which is a quite separate topic called interest management [7]. In IM, a client declares its interests with regards to his location, what he sees or what his sensors engaged to. For instance, a user walkthrough in urban environment and he only needs to get the visible primitives in his AoI or his viewing direction. These primitives may be static objects like buildings, roads or be dynamic objects like autonomous cars, people or avatars of remote clients. This way, we may limit the data distribution regarding to the requirements of each client and enable server and client to utilize their resources more efficiently [31] [32].

IM approach in data distribution is usually used in message passing related to dynamic objects. It can be classified as class-based and space-based [7]. A client states its interests about object's attributes in class-based IM. For example, in an airport, a surveillance radar may only subscribe to airplanes therefore it doesn't take any message from other dynamic agents such as busses, cars or people. In space-based, subscription is done in terms of positions of agents. Turning back to our example, radar station may subscribe only aerodrome control zone of that airport but not other airports'. Space-based can also be categorized as region-based and aura-based [33]. Main distinction of them is that in region-based the scene is partitioned into static regions, while in auras-based spaces are determined dynamically according to agents and their interests [7], [33].

As a conclusion, seizing the frame rate objective is vital in rendering. Since rendering pipeline starts with retrieving data from either network or storage, bottleneck in this stage directly affects rendering time. Thus, latencies and overloads in both network and storage should be minimized. One way to accomplish this drawback is to distribute data to only relevant clients. For this purpose, some IM and visibility techniques have been proposed in time. In the next section, we review the well known thirteen DVE implementations in the perspective of data distribution management.

5. VISIBILITY USAGE IN DVES

5.1. RING (1995)

RING presented by Funkhouser [34] is a system whose aim is to reduce the number of messages sent from server to clients based on possible visual interactions between entities in a virtual environment. The main idea behind the RING is that clients should take update messages only if updates are relevant to them. By using visibility algorithms, server decides which update message should be sent to whom.

The implementation of RING is as follows. At the beginning, virtual environment is divided into cells which are axis aligned and have static boundaries. Then, the portals which provide line-of-sight visibility between cells are defined. During the simulation, server keeps track of clients and their regions. Thus, when a client informs server about an update, server computes visible cells of the client and forward update messages to relevant server or clients.

It would be an advantage of RING that storage, processing and network requirements of clients individually are independent from number of active clients since each of the clients keep and process only its local data. Besides, performing computations on server side improves network performance. However, because there is additional process in server, great deal of latencies may occur.

5.2. ϵ-Neighborhood (1998)

In a DVE using server-client architecture, the server should send only relevant data instead of entire scene to the client. Common method to accomplish this is calculating PVS in server side. However, as stated in Section II, little change in viewpoint of a client may cause great deal of difference in visibility. Therefore server might overwork on computing new visible sets and sending them to the clients. In order to disburden the server and reduce message traffic between the server and the client, Cohen-Or and Zadicario [35] proposed an algorithm for computing superset of PVS by taking a few neighbors of current viewpoints into consideration. By this way, as long as the client is on these region there is no need to ask the server to compute PVS.

Implementation details of the algorithm are as follows: From a viewpoint, visibility of an object is determined by selecting a convex occluder and testing whether the object is hidden by the occluder or not. If this process is executed for two individual viewpoints and the object is determined as invisible from both of them because of being hidden by the same occluder, it is inferred that the object is invisible from any point between selected viewpoints. Based on this, a tetrahedron shadow umbra is constituted by combining edges of occluder and object and then the object is accepted as invisible whenever the clients stay in the umbra. The superset of the PVS is computed in server by the help of this umbra.

In this algorithm, the most expensive process is ray shooting to construct shadow umbra. Besides, occlusion fusion is not supported. Another disadvantage is that occluders must be in convex form although some methods can be performed to overcome this.

5.3. Update Free Regions (1999)

In server-client architecture, a server knows where each client is located at. Therefore, the server is able to filter messages, and can send update messages to only relevant clients. However, in P2P architecture, it may not be possible since keeping data of all clients causes additional cost.

To handle this, Makbily et al. proposed a new concept that uses Update Free Regions (UFRs) to compute visibility and reduce message traffic [36].

UFRs are determined in preprocess stage for each agent pair in their algorithm. UFR is a region where an agent doesn't send update messages to another while it stays in. In other words, as long as an agent stay on its UFR, it knows that the other agent cannot see it. When it leaves UFR, it comes to mean that these agent pair may see each other; so update messages are sent (Figure 5). As seen easily, this method is not efficient in multi-user environments in the case UFRs are computed for each pair.

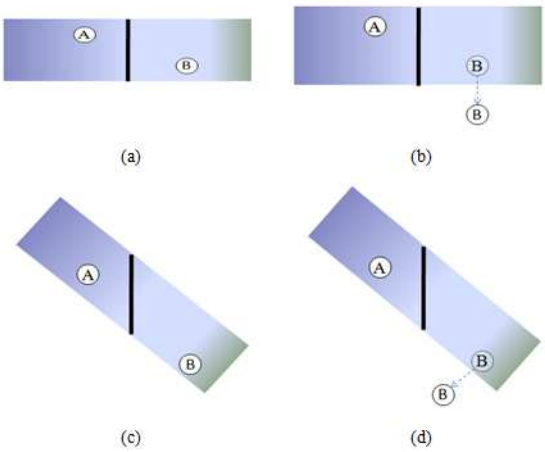


Figure 5. (a) According to initial positions of A and B, UFRs are identified where A and B cannot see each other. As long as both of them stay in their own UFR, they don't send update messages to the other. (b) When one of the clients leave its UFR, this come to mean that they can see each other and thus it is tested whether new UFRs can be identified or not. (c) Since new regions which fulfill the conditions can be identified, UFRs are updated. (d) In the case one of the clients leave its region and new UFRs cannot be identified, clients begin to send update messages to the other.

5.4. Smart Visible Sets (2002)

In spite of the common acceptance of the PVS methods for reducing the data amount flowing over network, in some complex environments PVSs might not be small and simple enough. Moreira et al. [37] thus present a Smart Visible Sets (SVS) concept that enables partitioning PVS into smaller subsets and setting priorities to them. In this way, it is aimed to decrease data amount sent from server by sending the most prioritized set instead of less important ones in the event of congestion over network.

SVS is initially computed by splitting viewing frustum into pieces. Each of the pieces covers specified angle. As the angle can be constant, it can also be adaptive because defining constant angles may cause more than needed split cells. In both cases, depth-traversal of the Binary Space Partitioning (BSP) is performed to determine which geometries are located in cells. Once division of the cells is completed, cell-to-cell visibility test is performed by checking divided cells' boundaries according to current frustum. Besides angle specification, distance parameter can also be used in partitioning by considering the distance from PVS region to the divided regions. Hence, distances between divided cells are estimated and viewing distance as well as visible other cells of the selected cell is stored.

The other purpose of this work is setting priorities to cells in order to sort them in addition to defining SVS. Because some losses or latencies may occur in the case of network bandwidth overload, some data might be sent lately or incompletely from the server. Prioritization information is maintained on the server and the server tries to send most desired and most necessary cell first to overcome this drawback.

5.5. A Navigation System by Marvie et al. (2003)

In the work of Marvie et al. [38], they present a navigation system that allows real-time remote walkthrough built on a server-client architecture. Considering network bandwidth restrictions; visibility, prefetching and Level-of-Details (LoD) techniques have been adapted in their system.

In this method, region-based visibility computations are used by

defining view cells and computing PVSs for each of them. In addition, adjacency graph is also constructed to enable next cell estimation and thus prefetching. Most important part of the approach within our context is that the system takes the advantage of visibility computation results while deciding LoD in order to reduce the amount of polygons to be rendered. Different from traditional metric-based LoD determination, Average Coverage Hint (ACH) is computed. ACH is the average surface area of the object when projected from a viewcell. In the final rasterization, ACH is used to determine LoD as a percentage of covered pixels.

5.6. CyberWalk (2003)

CyberWalk [39] is an on-demand distributed virtual walkthrough system that enables walking through a virtual environment over internet. It is built upon server-client architecture and all objects are stored in central server. There are three featured issues in CyberWalk. Firstly, models are maintained in compact form in order to reduce transmission and thus rendering time. Secondly, the system keeps clients alive in case of disconnection problems. Lastly, it provides catching and prefetching mechanism.

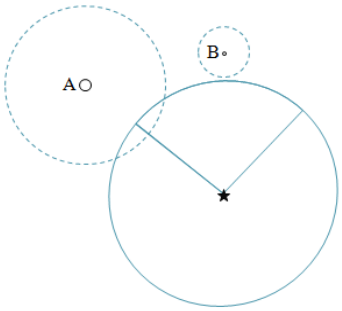


Figure 6. In CyberWalk, since the user's view scope (straight line) overlaps with object A's object scope (dashed line), A is handled for further resolution determination. Though B is closer than A to the user, it is not handled because most probably B's size is much smaller.

Most approaches identify AoI according to viewer. In CyberWalk, its scope is extended by considering AoI from both viewer and object. The former is called the viewer scope and the letter is called the object scope. A viewer scope is an area bounded by viewer's depth of sight, while object scope indicates the area where the object can be seen by agents. In run time, the visibility test is performed by comparing these circular regions' (scopes) relations between each other. As a result of the test if two scopes intersect, it comes to mean that the viewer may see the object. Then, according to the viewer's viewing angle and distance between the viewer and the object, resolution of the object is determined (Figure 6). Finally, object model with the computed resolution is sent from server to client.

5.7. From-Point Based Prefetching (2003)

Correa et al. [40] presented a from-point visibility-based prefetching algorithm for interactive out-of-core rendering. Their method uses prioritized-layered projection (PLP) algorithm to compute approximate visible set (AVS), cPLP to compute conservative visible set and a point-based visibility algorithm to determine geometries that the user may need in the near future. PLP performs computation like view-frustum culling except the traversal of the nodes from the highest to lower priority, while it is executed in pre-defined order in view-frustum culling.

Whenever camera position or orientation is changed, new visible set is computed by the system according to the user's selection (AVS or PVS). Then the look-ahead thread estimates next camera position in regard to camera's direction and speed. After defining possible camera positions, the look-ahead thread performs PLP to decide which nodes are possibly visible and for each likely visible node it sends a prefetch request to geometry cache. In this way, likely visible sets are predicted by the system and transferred to memory, thus data is retrieved from memory instead of disk in run time which is a less expensive process.

This algorithm has some advantages over region-based approaches. First of all, it doesn't need great preprocessing time since scene is not partitioned to regions like cells. In this stage only hierarchical structure is

created and thus in the rendering time visible set is computed without traversing over whole scene. Besides, because this algorithm handles smaller amount of data relative to region-based techniques, it reduces disk access. Lastly, there is no restriction like defining cell and portals or identifying axis aligned bounding box as done in region-based methods.

5.8. Frontier Sets (2004)

Steed and Angus [41] has presented a new data structure called a frontier set that is used to define mutually invisible cells between client pairs in region-based visibility. Frontier is a region (may consist of nodes or cells) where clients cannot see each other as long as both of them stay in their own frontier. Frontiers are defined for client pairs and both of them have knowledge about these frontiers. Whenever one of them leaves its frontier, it comes to mean that it may see the other. In this point, client left its frontier informs the other one and they renegotiate to identify new frontiers. Frontier definition and updates are illustrated in Figure 7.

Creation of frontiers is based on cell-and-portal approach. Cells are linked to each other in an adjacency graph and from the cell where clients exist in, traverse is executed to decide whether nodes (cells) are visible or not. Although frontier creation may take long time, in the run time data amount flowing over network is reduced. This data structure is adapted to P2P network architectures and provides well scalability.

In authors' succeeding work [42], they aimed to reduce frontier definition algorithm complexity. For this purpose, they proposed computing enhanced PVS that is including visibility distance metric. By this way, frontiers can be created during run time dynamically and though it needs larger storage capacity because of distance parameter, complexity dropped from O(N3) to O(N2).

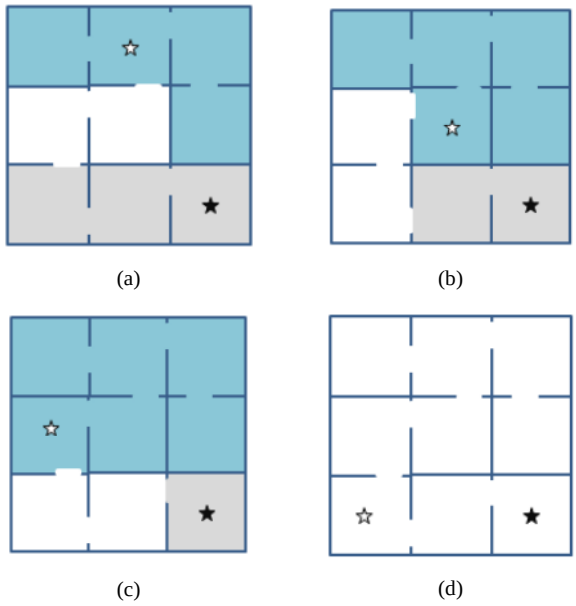


Figure 7. A client pair negotiate and define their frontiers in initial position (a). When at least one of them leave its frontier, they renegotiate and define new frontier regions (b and c). If there is a line of sight between cells those contain client pair, frontier cannot be constructed (d).

5.9. Partially Ordered Delivery for 3D Scenes (2006)

The method presented by Tian and AlRegib [43] aims to prevent server side to send objects with unnecessarily high resolutions. To accomplish this, a multiresolution method is implemented by considering the fact that there are usually multiple objects in the same viewing frustum. Thus, interactions between objects are taken into account for assigning them a weight value that represents relative importance of the object in the rasterization.

The weighting algorithm performs two processes. At first, well known viewing frustum culling is executed. As a result of this test, two sets are created: outside of the frustum and at least partially inside of the frustum. For the second set, distances between view point and the center of the

bounding boxes surrounding objects is sorted with the closest objects first order. In the second process, image-spaced mapping is performed by using bounding volume projection map. From the sorted list, objects are mapped to the scene projection according to their weight factor, and farther objects are therefore not rendered. In subsequent stages of the method, LoD and thus resolution are determined after visibility computations. Finally, the server decides objects to send and their LoD characteristics.

5.10. Object-Initiated View Model (2006)

A client can see only objects lay inside its AoI or viewing frustum. While the client cannot see objects immediate ahead of the viewing frustum, those objects may suddenly be appear after a while during navigation [44]. Thus, this problem which cause discontinuities in walkthrough disturbs the client's view seriously and called popping problem [45].

In order to minimize popping problem, object-initiated view model that take also the AoI of objects as well as users into consideration is proposed by Seo and Zimmermann [44]. AoI for each object is computed considering several parameters like illumination, distance, size. The basic idea behind the visibility determination is testing whether the AoI of the object covers the client or not. Seo and Zimmermann emphasize the drawback of storing and retrieving such a big data rather visibility determination. Therefore they present a new indexing method called edge indexing which is out of our scope. In another work [45], they have implemented the same paradigm into different environments as a stationary user in a stationary environment, a moving user in a stationary environment and a stationary user in a moving environment.

5.11. Flowing Level-of-Details (2008)

In DVEs, one method to distribute data to clients is delivering content before application runs that is named as pre-downloading or pre-installing [32] or offline distribution [29]. However, in the case there is much larger and more dynamic content or larger number of objects, this method becomes inadequate. For this purpose, Hu et al. presented a P2P 3D

streaming framework that is called Flowing Level-of-Details (FLoD) [46].

Main idea behind the FLoD is to enable clients obtaining data from others those have similar AoI or overlapped visibility with that client. To handle this, three requirements are emerged. First, because all data is maintained in server initially, it is required to partition and deliver some related scene data to clients in time. Second, clients must have knowledge about others at least those share common AoI with it. Third, clients must be able to select available peers among others from whom it can take scene data efficiently. To meet these requirements, FLoD performs five tasks as partition, fragmentation, prefetching, prioritization and selection.

FLoD uses Voronoi-based Overlay Network to organize peers according to their AoI and boundary neighbors. When a peer login in first time, it informs server about its initial location and AoI. Then, to decide which part of the scene is visible, it requests likely visible objects from its neighbors by reason of the fact that its neighbors might visit this peer's current location before, thus may have visibility information about that area. Whenever the peer moves in VE, it updates its neighbors and make new requests to them for visibility determination. By this way, server gets rid of sending repeating data to clients.

5.12. Distributed Massive Model Rendering (2012)

Since CPU, GPU, or memory in a single host fail to cope with the processing large amount of 3D geometries, a new framework distributing discrete processes to individual hardware and hosts have been proposed by Revanth and Narayanan [47]. Coarsely, in their distributed rendering solution that is built upon a server-client architecture, the server performs view-frustum culling, one GPU in the client performs visibility culling and another GPU in the client executes rendering.

The proposed pipeline consists of four parallel modules. In the load balanced frustum division module, viewing-frustum culling is performed by the server. As the server has the whole knowledge about the scene, this process can be handled efficiently according to the client's position and

bounding boxes (or spheres) of objects. Thus, the server sends only relevant data to the client instead of the entire scene. In the second module named as visibility determination module, the client does visibility culling, particularly occlusion test in the first GPU. In the sub-frame rendering module which is the third stage, the second GPU of the client is used to render sub-frame. Lastly, these sub-frames are sent to the server to be assembled in the assemble frame module.

5.13. A3 (2008) and EA3(2013)

In order to use network bandwidth efficiently, a proximity-based IM algorithm called A3 is presented by Bezerra et al [48]. By calculating distances between the client and entities, it is decided whether the entity is relevant to the client. For optimization, view distance, field of view (FoV) and critical area of the client is considered in the algorithm. By defining critical area that is a circle whose centre is the location of the client and radius is the critical distance, it is provided that the client can take most relevant updates in the immediate vicinity as soon as possible because network bandwidth is allocated to critical area with priority (Figure 8.a).

A3 is improved by Vatjus-Antilla et al. [49] considering occlusions. In their algorithm, which is named as EA3, they use ray visibility algorithms additionally. Basic rational is as same as A3 in defining critical area and FoV. Differently, obstacles are identified, then by using axis aligned bounding box (AABB) ray casting algorithm occluded part of the FoV is discarded and the area of the FoV is thus Minimized (Figure 8.b). As a result, message traffic flow from the server to the client is greatly decreased as this process is executed in the server.

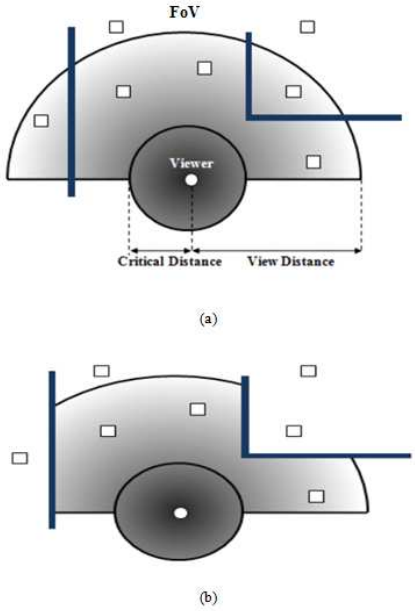


Figure 8. (a) In A3 algorithm, critical distance and view distance are identified. Primitives in the critical area have more priority than others which exist in rest of the FoV. (b) In EA3, only difference from A3 is taking the obstacles into consideration and thus reducing the area of FoV.

6. CONCLUSION

In this paper, we aimed to propose a survey to the researchers who plan to work on data distribution based on visibility and AoI in DVEs. Even though there are individual surveys on visibility, data distribution with or without AoI approach, our aim is to bring all of them together in the perspective of visibility based data distribution.

We firstly discussed visibility that has the objective of rejecting invisible primitives as early as possible in virtual scenes. By using effective visibility culling techniques, unnecessary primitives are not handled and

therefore processes must be performed is reduced. Visibility algorithms are classified as point-based and region-based according to considering visible primitives from the current view point or a region. After examining methods in both types, it is clearly seen that region-based algorithms have enormous advantages over point based algorithms in providing prefetching, fluency in scene transitions, exploiting coherence and less run time processes. On the other hand, region-based approaches may need longer preprocessing time for defining cells, portals, viewing cells etc. To seize the frame rate objective, most suited techniques should be preferred considering scene complexity, data sets, hardware and software properties.

Secondly, we mentioned the importance of data distribution and IM in DVEs. Parallel to the recent developments in computer technology, data amount in DVEs increase a lot as a consequence of increase at details and reality of scenes. Thus, data transfer between participants should be well managed. To overcome this challenge, data distribution management techniques most of whose are based on considering AoI are used. By this way, some data set are eliminated before transfer thus network bandwidth is used more efficiently. Besides, each participant has only local data about the scene and doesn't have to cope with the entire scene that contains mostly irrelevant data set.

Lastly, we surveyed thirteen data distribution techniques which are summarized in Table 1 in terms of visibility and IM in chronological order. All these methods have in common that they perform visibility methods before distributing data to participants. They consider participants' interests and thus send related data set. To accomplish this, most of them benefit by dividing scene into regions and construct relations between regions and user interests. While some of them are directly focus on visibility culling, some other use visibility computations in a small part of pipeline for example to decide LoD or resolution.

Table 1. Summary of Discussed Methods

Technique	**Server-Client/ P2P**	**Point-Based/ Region-Based**	**Visibility Issues**	**IM and DDM Issues**
RING	Server-Client	Region-Based	Cell-and-Portal approach is used.	Server keeps track of cells in which clients exist and distribute data to relevant clients.
ϵ-Neighborhood	Server-Client	Region-Based	Superset of PVS is computed by constituting shadow umbra from a selected convex occluder.	Since superset of PVS is computed, client request updates from server rarely.
Update Free Regions	Server-Client	Region-Based	Regions in which clients cannot conduct line-of-sight due to an obstacle are determined.	Unless clients leave their region, no message is requested from server.
Smart Visible Sets	Server-Client	Region-Based	PVS is partitioned into smaller cells and cell-to-cell visibility is tested according to the viewing frustum.	Smart visible sets are obtained by assigning priorities to partitioned cells and server send the highest priority data first.
A Navigation System by Marvie et al.	Server-Client	Region-Based	View cells are defined and PVSs are computed for each of them.	Resolution of geometries sent from server is decided as a result of visibility computations by computing ACH.
Cyber Walk	Server-Client	Point-Based	PVS is computed by testing whether objects' and viewer's scope overlap.	AoI is taken into consideration for both objects and viewer.
From-Point Based Prefetching	Server-Client	Point-Based	PLP or cPLP algorithms are used to compute AVS or PVS. Besides, the algorithm determines geometries which the client may need in near future.	Likely necessary geometries are tested firstly thus prefetching is enabled.
Frontier Sets	P2P	Region-Based	Cell-and-Portal approach is used.	As long as peers are in their own frontier, no message or update is transferred between them.
Partially Ordered Delivery for 3D Scenes	Server-Client	Point-Based	Firstly, viewing frustum culling is performed. Then, objects are sorted upon their distances from viewer.	By considering distance factor of objects, LoD of objects are determined and this prevents server to send objects with unnecessarily high resolution.
Object-Initiated View Model	Server-Client	Region-Based	Visibility is determined by testing whether object's AoI covers the client.	Edge Indexing method is presented in order to overcome the drawback of storing each object's AoI.
Flowing Level-of-Details	P2P	Region-Based	A peer request likely visible objects from its neighbors.	Each connected peer declares its AoI and position.
Distributed Massive Model Rendering	Server-Client	Region-Based	Visibility computations are dispatched to server and client. View-frustum culling is performed in server, while visibility culling is done in GPU of client.	By performing visibility culling in server side, data amount to send from server is decreased.
A^3 and EA^3	Server-Client	Region-Based	Traditional view-frustum culling is performed. Differently, in EA^3, obstacles are also taken into consideration to reduce FoV.	Proximity based interest management algorithm is used and objects are taken in sort of priority according to each client's critical area.

REFERENCES

[1] Michael Capps and Seth Teller, "Communication visibility in shared virtual worlds," in *Enabling Technologies: Infrastructure for Collaborative Enterprises, 1997. Proceeding, Sixth IEEE Workshops on*. IEEE, 1997.

[2] Richard M. Fujimoto, "Parallel and distributed simulation," in *Proceedings of the 31st conference on Winter simulation: Simulation-a bridge to the future-Volume 1*., 1999.

[3] Katherine L. Morse and Jeffrey S. Steinman, "Data distribution management in the HLA: Multidimensional regions and physically correct filtering," in *Proceedings of the 1997 Spring Simulation Interoperability Workshop*, 1997.

[4] Daniel Cohen-Or, Yiorgos L. Chrysanthou, Cláudio T. Silva, and Fredo Durand, "A survey of visibility for walkthrough applications," in *Visualization and Computer Graphics, IEEE Transactions on* 9.3 (2003): 412-431.

[5] Azzedine Boukerche, Nathan J. McGraw, and R. B. Araujo, "A novel data distribution management scheme to support synchronization in large-scale distributed virtual environments," in *Virtual Environments, Human-Computer Interfaces and Measurement Systems, 2005. VECIMS 2005. Proceedings of the 2005 IEEE International Conference on*. IEEE, 2005.

[6] Jiri Bittner and Peter Wonka, "Visibility in computer graphics," in *Environment and Planning B: Planning and Design* 30 (2003): 729-755.

[7] Jean-Sébastien Boulanger, Jörg Kienzle, and Clark Verbrugge, "Comparing interest management algorithms for massively multiplayer games," in *Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games*, 2006.

[8] Chris Carter, Abdennour El Rhalibi, and Madjid Merabti, “A survey of AoIM, distribution and communication in peer-to-peer online games,” in *Computer Communications and Networks (ICCCN), 2012 21st International Conference on*. IEEE, 2012.

[9] Elvis S Liu, and Georgios K. Theodoropoulos. "Interest management for distributed virtual environments: A survey," in *ACM Computing Surveys (CSUR)*46.4 (2014): 51.

[10] Bingqing Shen, Jingzhi Guo, and Peng Chen. "A survey of P2P virtual world infrastructure,," in *e-Business Engineering (ICEBE), 2012 IEEE Ninth International Conference on*. IEEE, 2012.

[11] John S. Gilmore and Herman A. Engelbrecht. "A survey of state persistency in peer-to-peer massively multiplayer online games," *Parallel and Distributed Systems, IEEE Transactions on* 23.5, 2012, pp. 818-834.

[12] Eliya Büyükkaya, Maha Abdallah, and Gwendal Simon. "A survey of peer-to-peer overlay approaches for networked virtual environments," *Peer-to-peer networking and applications* 8.2, 2013, pp. 276-300.

[13] Edwin Catmull, “A subdivision algorithm for computer display of curved surfaces,”, Ph.D. dissertation, University of Utah, Salt Lake City, Utah, 1974.

[14] James T. Klosowski and Cláudio T. Silva, "The prioritized-layered projection algorithm for visible set estimation," in *Visualization and Computer Graphics, IEEE Transactions on* 6.2 (2000): 108-123.

[15] Gürkan Koldaş, “Efficient visibility estimation for distributed virtual urban environments,” Ph.D. dissertation, Middle East Technical University, Ankara, 2008.

[16] Frame rate, Wikipedia, [online] 2015, https://en.wikipedia.org/wiki/Frame_rate (Accessed: 06 October 2015).

[17] Hansong Zhang, “Effective occlusion culling for the interactive display of arbitrary models,” Ph.D. dissertation, The University of North Carolina, Chapel Hill, 1998.

[18] Seth J. Teller and Carlo H. Séquin, "Visibility preprocessing for interactive walkthroughs," *Computer Graphics*. Vol. 25. No. 4., 1991.

[19] Ned Greene and Michael Kass, "Error-bounded antialiased rendering of complex environments," in *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, 1994.

[20] Satyan Coorg and Seth Teller, "Temporally coherent conservative visibility," in *Proceedings of the twelfth annual symposium on Computational geometry*, 1999.

[21] Satyan Coorg and Seth Teller, "Real-time occlusion culling for models with large occluders," in *Proceedings of the 1997 symposium on Interactive 3D graphics*, 1997.

[22] T. Hudson, D. Manocha, J. Cohen, M. Lin, K. Hoff, and H. Zhang, “Accelerated occlusion culling using shadow frusta,” in *Proceedings of the thirteenth annual symposium on Computational geometry*, 1997, pp. 1-10.

[23] Ned Greene, Michael Kass, and Gavin Miller, "Hierarchical Z-buffer visibility," in *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, 1993.

[24] Fausto Bernardini, James T. Klosowski, and Jihad El-Sana, "Directional discretized occluders for accelerated occlusion culling," in *Computer Graphics Forum*. Vol. 19. No. 3. Blackwell Publishers Ltd, 2000.

[25] Jiří Bittner, Vlastimil Havran, and Pavel Slavik, "Hierarchical visibility culling with occlusion trees," in *Computer Graphics International, 1998. Proceedings*. IEEE, 1998.

[26] Frédo Durand, George Drettakis, Joelle Thollot, and Claude Puech, "Conservative visibility preprocessing using extended projections," in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, 2000, pp. 239-248.

[27] Mojtaba Hosseini, Steve Pettifer, and Nicolas D. Georganas, "Visibility-based interest management in collaborative virtual environments," in *Proceedings of the 4th international conference on Collaborative virtual environments*, 2002.

[28] Huy T. Vo et al., "iRun: Interactive rendering of large unstructured grids," in *Eurographics Symposium on Parallel Graphics and Visualization*, 2007.

[29] Gerd Hesina and Dieter Schmalstieg, "A network architecture for remote rendering." *dis-rt*. IEEE, 1998.

[30] James F. Kurose and Keith W. Ross, "Computer networking a top-down approach," Fifth Edition, Pearson Education, 2010, ch. 2, pp. 112-116.

[31] Katherine L. Morse, "Interest management in large-scale distributed simulations," in Information and Computer Science, University of California, Irvine, 1996.

[32] Shun-Yun Hu, "Spatial publish subscribe," *Proc. of IEEE Virtual Reality (IEEE VR) workshop, Massively Multiuser Virtual Environment (MMVE'09)*, 2009.

[33] Graham Morgan, Fengyun Lu, and Kier Store, "Interest management middleware for networked games," in *Proceedings of the 2005 symposium on Interactive 3D graphics and games*, 2005.

[34] Thomas A. Funkhouser, "RING: a client-server system for multi-user virtual environments," in *Proceedings of the 1995 symposium on Interactive 3D graphics*, 1995.

[35] Daniel Cohen-Or and Eyal Zadicario, "Visibility streaming for network-based walkthroughs," in *Graphics Interface*. Vol. 98. No. 1, 1998.

[36] Yohai Makbily, Craig Gotsman, and Reuven Bar-Yehuda, "Geometric algorithms for message filtering in decentralized virtual environments," in *Proceedings of the 1999 symposium on Interactive 3D graphics*, 1999.

[37] Fábio O. Moreira, Joao L.D. Comba, and Carla M.D.S. Freitas, "Smart visible sets for networked virtual environments," *Computer Graphics and Image Processing, 2002. Proceedings. XV Brazilian Symposium on*. IEEE, 2002.

[38] Jean-Eudes Marvie, Julien Perret, and Kadi Bouatouch, "Remote interactive walkthrough of city models." in *Pacific Conference on Computer Graphics and Applications*, 2003.

[39] Jimmy Chim, Rynson W.H. Lau, Hong Va Leong, and Antonio Si, "CyberWalk: a web-based distributed virtual walkthrough environment," *Multimedia, IEEE Transactions on,5*(4), 2003, pp. 503-515.

[40] Wagner T. Correa, James T. Klosowski, and Claudio T. Silva, "Visibility-based prefetching for interactive out-of-core rendering," in *Proceedings of the 2003 IEEE Symposium on Parallel and Large-Data Visualization and Graphics*, IEEE Computer Society, 2003.

[41] Anthony Steed and Cameron Angus, "Frontier sets: A partitioning scheme to enable scalable virtual environments," in *Proceedings of EUROGRAPHICS 2004, Short Presentations and Interactive Demos*, 2004, pp: 13-17.

[42] Anthony Steed and Cameron Angus, "Supporting scalable peer to peer virtual environments using frontier sets," in *Virtual Reality, 2005. Proceedings. VR 2005. IEEE*. IEEE, 2005.

[43] Dihong Tian and Ghassan AlRegib, "PODS: partially ordered delivery for 3D scenes in resource-constrained environments," in

Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on. Vol. 5. IEEE, 2006.

[44] Beomjoo Seo and Roger Zimmermann, "Edge indexing in a grid for highly dynamic virtual environments," in *Proceedings of the 14th annual ACM international conference on Multimedia*, 2006.

[45] Beomjoo Seo and Roger Zimmermann, "Quantitative analysis of visibility determinations for networked virtual environments," *Journal of Visual Communication and Image Representation* 23.5, 2012, pp. 705-718.

[46] Shun-Yun Hu, Ting-Hao Huang, Shao-Chen Chang, Wei-Lun Sung, Jehn-Ruey Jiang, and Bing-Yu Chen, "Flod: A framework for peer-to-peer 3D streaming," in *The 27th Conference on Computer Communications. IEEE*, 2008.

[47] Revanth N. R. and P. J. Narayanan, "Distributed massive model rendering," *Proceedings of the Eighth Indian Conference on Computer Vision, Graphics and Image Processing*, 2012.

[48] Carlos Eduardo B. Bezerra, Fábio R. Cecin, and Cláudio FR Geyer, "A3: A novel interest management algorithm for distributed simulations of mmogs," in *Distributed Simulation and Real-Time Applications, 2008. DS-RT 2008. 12th IEEE/ACM International Symposium on*. IEEE, 2008.

[49] Kari Vatjus-Anttila, Timo Koskela, Seamus Hickey, and Jarkko Vatjus-Antilla, "Occlusion based message dissemination method in networked virtual environments," in *Next Generation Mobile Apps, Services and Technologies (NGMAST), 2013 Seventh International Conference on*. IEEE, 2013.