A Graphics Adaptation Framework and Video Streaming Technique for 3D Scene Representation and Interaction on Mobile Devices

Congdu Nguyen, Minh Tuan Le, Dae-Il Yoon, and Hae-Kwang Kim

School of Computer Engineering, Sejong University, Seoul, Korea
{congdu,tuanlm,daeil}@sju.ac.kr and hkkim@sejong.ac.kr

Abstract. In this paper, we propose a graphics adaptation framework with a mechanism of video streaming to overcome the shortcoming of real-time representation and interaction experiences of 3D graphics application running on mobile devices. We therefore develop an interactive 3D visualization system based on the proposed framework for rapidly representing a complex 3D scene on mobile devices without having to download it from the server. Our system scenario is composed of a client viewer and an adaptive media streaming server. The client viewer offers the user to navigate the 3D scene and interact with objects of interests for studying about them through the responded text descriptions. The server adaptively provides media contents to the client according to the user preferences, interactions, and the condition of wireless network.

Keywords: Video streaming, interactive 3D visualization, adaptive media content, MPEG-4/H.264 standard, color vision deficiency.

1 Introduction

In recent years, real-time representation and interaction for a highly complex 3D scene is one of fundamental problems in 3D graphics application for mobile devices. It is because the performance of 3D graphics requires a powerful computational capability, and large storage memory. In addition, a very huge data size of 3D scenes will also cause problems for storing, transferring and rendering the contents not only on mobile devices but also on PCs.

A number of individual approaches are proposed to overcome these difficulties. Teler [1], [6] presented a method of streaming 3D scene for remote walkthrough by selecting a portion of the 3D scene and reducing quality representation for objects. This method only sends the visible 3D contents at the observer viewpoints. These contents are then kept in the client sides for reusing so these contents will not be sent again. Schneider [2] and Hesina [3] introduced a network graphics framework that includes various transmission methods to provide a constant quality of 3D-contents downloaded with different conditions of environment. AlRegib [7] developed a special network layer called 3D model transport protocol for streaming 3D models over the network. A single-level compression method is used to reduce the transmitted bits.

C. Stephanidis (Ed.): Universal Access in HCI, Part I, HCII 2007, LNCS 4554, pp. 481–490, 2007.

Some other methods attempt to reduce bandwidth requirements by developing compression algorithms [14], [15], [16], [17]. Using arbitrary triangular mesh with an irregular connectivity, a 3D model is compressed by two channels of data (geometry data and connectivity data) before it is streamed to the client. To deal with the scene complexity, Hoppe [4] and Funkhouser [5] proposed approach of level of detail (LOD) models and object progressive rendering. Users can see the objects with different levels of detail depending on transferring intervals of time or distance from the observer viewpoint to the objects.

Similar to our research, many researches invested accelerated rendering algorithms in taking advantages of displaying 2D rendered images for easily visualizing a 3D scene in the client sides. Image-based rendering techniques [8], [9], [10], [11], [12], [13] do not use 3D geometry information of the 3D scenes such as object attributes, vertex points and their connectivity. Thus, the client alternatively receives and displays rendered images in a very short time. To archive high performance, the rendered images usually are compressed using JPEG compression standard [9]. However, the data size of compressed images is still big compared to the bandwidth of wireless links of mobile devices.

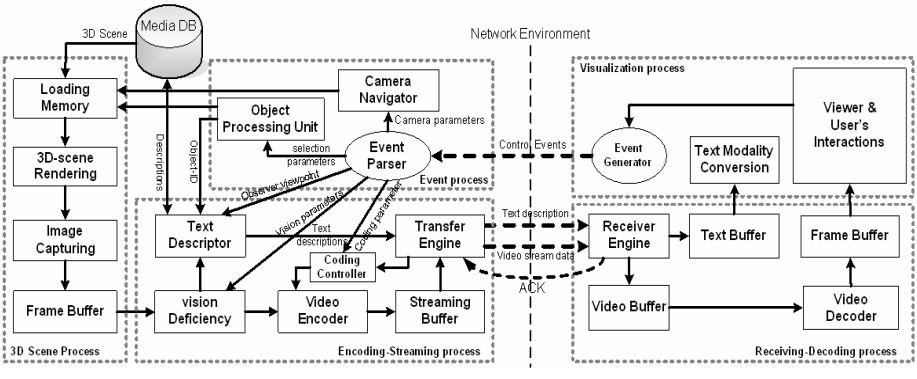

Fig. 1. A graphics adaptation framework for the Inter3DV system, the client can navigate for a large and complex 3D scene without having to download it from the server

The above solutions all, however, require a powerful computational capability, large memory for storing, 3D graphics hardware, and programming capabilities from the client sides. Meanwhile, every mobile device can not embed required hardware for rapidly displaying and memory capacity for storing the large and complex 3D scene. Moreover, the mobile devices are still limited in bandwidth of wireless links so that transferring data of a 3D scene over the network is a big challenge. Whereas, today mobile devices usually support natively video, audio, and text, they offer direct optimization for playing back of such media contents. Consequently, in this paper, a graphics adaptation framework for an interactive 3D visualization (Inter3DV) system is proposed. The Inter3DV system offers the client viewer on mobile devices to rapidly represent a large and complex 3D scene resided in the server without having to download it. The user can remotely navigate the 3D scene and select an object of interests for studying according to the text description introducing about it.

The graphics adaptation framework is providing a representation, interaction and adaptive media content experiences with no additional processing cost on the client side. Our solution is developing an adaptive media streaming (AMS) server for streaming the video stream generated from a remote interaction. The AMS server frequently renders a sequence of images and then encodes them into a video stream. The video stream is then received by the client, which decodes the video stream into video-frames and displays. Video codec processes is based on MPEG-4/H.264 standard. The AMS server provides the client with adaptive video stream in term of color vision deficiency, quality representation according to the user preferences and the variation of network bandwidth and also provides suitable text descriptions depended on the user interactions.

2 A Graphics Adaptation Framework

Figure 1 illustrates a graphics adaptation framework of the Inter3DV system that takes the advantages of video streaming technique for reducing the data transmission rate and accelerated rendering on the client sides. The client viewer offering real-time representing and navigating a complex 3D scene is specially developed for HP iPAQ Pocket PCs. The client viewer provides a user-interface for representing the responded media contents and for controlling visualization. The server, AMS server, plays important roles of transforming the 3D scene into suitable media contents due to the user interactions and streaming these media contents to the client viewer.

2.1 Adaptive Media Streaming Server

Before the media contents are sent to the client, the server has a lot of works to do that can be performed in three processes.

Event Process. This process manages the interactions derived the user on the client. Event process includes *event-parser, camera-navigator,* and *object-processing-unit* modules. On the client side, whenever the user interacts with the 3D scene such as walking through the scene or selecting an interested object, the event-generator creates a control event and sends it to the server. On the server side, the event-parser converts the control event into control parameters (camera, selection, coding parameters, etc.) that will be conveyed to one of the following processes: text-descriptor, camera-navigator, object-processing-unit, vision-deficiency and coding-controller. The observer viewpoint of the 3D scene is updated following the movement and orientation of the virtual camera. The camera-navigator uses the camera parameters to adjust the virtual camera by changing camera coordinates in position (x, y, z) and changing camera orientation in pose (h, p, r), *h* is rotated by z-axis, *p* is rotated by x-axis and *r* is rotated by y-axis. The object-processing-unit uses the selection parameters to obtain the selected object in the 3D scene. The process of object identification is doing as follows. Because the size of image rendered in the server side (called *rendered image*) might be different to the size of image displayed on the client side (called *displayed image*). Therefore, the position of mouse event should be mapped into the rendered image at the current observer viewpoint to archive correct coordinates. Assume that, x_n and y_n are coordinates projected on the rendered image,

x and y are coordinates projected on the displayed image. x_n and y_n are calculated as the following: $x_n = x*r_w$ and $y_n = y*r_h$, where r_w and r_h are ratio by width and ratio by height between the rendered image and displayed image. The x_n, y_n coordinates are then projected into the 3D space of the 3D scene to obtains the object-ID of the selected object. The selected object is marked by adding its triangle mesh. The object-ID is also sent to the text-descriptor for querying its text description in the Media DB**.**

3D-scene Process. 3D-scene process is designed for rendering and grabbing images from the 3D scene at the observer viewpoints. This process is broken into four processing layers: loading-memory, 3D-scene-rendering, image-capturing, and frame-buffer. When a client sends a request to access a 3D scene in the AMS server, the 3D scene is loaded to the loading-memory and then a virtual camera is mapped to the 3D scene. Responding to the observer viewpoints, the 3D-scene-rendering constantly renders raster images. This method uses a special function that is available in OpenGL called *feedback* function [21], [22]. When using feedback mode the 3D scene is not rasterized like the usual way, the transformed data (rendered/raster image) is stored in a feedback buffer. After that, the 3D-scene-rendering signs the image-capturing to read back the rendered image from the feedback buffer using *glReadPixels* function and store it into the frame-buffer preparing for the video encoding process.

Encoding-Streaming Process. The main tasks of encoding-streaming process are adaptively providing and streaming media contents (video stream data and text descriptions) in accordance to the control parameters received from the event-parser to the client. This process is performed as follows.

(1) The images in the frame-buffer first are passed to the vision-deficiency module for pre-processing color vision. If the vision parameters indicate the degree of color vision deficiency then the images will be converted by the visual content adaptation algorithm [19] to suit the visual perception characteristics of the user.
(2) The video-encoder compresses the converted images into a video stream based on MPEG/H.264 standard [18]. The video stream is finally stored in the streaming-buffer. During encoding process, the quality of video compression is always adjusted in order to satisfy the user preferences. *Channel rate* (a), *minimizing distortion* (b), and *quantizer* (c) are three encoding options for the user on the client. The channel rate option requires that, the system should always keep a stable frame rate. The minimizing distortion option requires that, the system provides result with a minimal video distortion for video segment in the streaming-buffer. If the user chooses option (c), the quality of video compression will not be changed. If the user sets up option (a) or (b), the system requires the coding-controller to find an optimal quantizer [20]. The coding-controller receives control parameters from the event-parser or a signal from the transfer-engine it finds an optimal quantizer and sends to the video-encoder for adjusting quality of video compression. The value of quantizer is varied from 1 to 31 in steps of 1. The increment of quantizer value is in inverse ratio to the quality of video compression.
(3) The transfer-engine receives the video stream from the video-buffer and receives text descriptions from the text-descriptor. It creates two data transmission channels,

one is for transferring the text descriptions and one is for transferring video stream data. If the text-descriptor receives an object-ID from the object-processing-unit or the vision-deficiency it tries to obtain the text description of the object-ID and sends to the transfer-engine. Whenever the transfer-engine receives ACK message from the receiver-engine, it indicates the coding-controller to find a new optimal quantizer for the video-encoder due to the rules of the coding options.

2.2 A Client Viewer

A client viewer, a real-time client application offering adaptively representing and navigating a complex 3D scene, is specially developed for mobile devices (HP iPAQ Pocket PCs). In previous works [1], [2], [3], [7], the clients need time-delay for downloading the 3D scene, a large memory space for storing data and required graphics hardware for accelerated 3D rendering. In our method, a video streaming technique is employed instead of streaming 3D scene for solving the above limitations and for accelerated representing a complex 3D scene. Therefore, a complex 3D scene resided in the AMS server is rapidly represented by the client viewer without having to download and store it as well. In this study, we are taking advantages of the video streaming technique due to not only the low computing capacity of the clients and the potential low bandwidth of wireless links, but also security concerns. The user on the client can not access the original data from the server because only generated video stream data is transferred to the client.

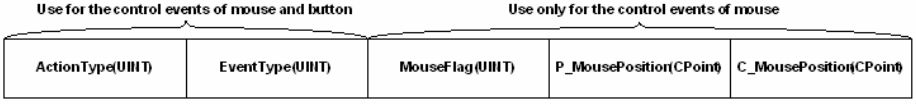

Fig. 2. The *Event_Action* structure of a control event

The media content, received by the receiver-engine, is temporarily stored into the video-buffer if it is a video stream data and stored into the text-buffer if it is a text description data. The video-decoder decompresses the video stream data into frames and stores into the frame-buffer. After that, the visualization process displays the frames and represents the text description through the *viewer & user's interaction* and *text-modality-conversion*.

The client viewer supports the user with walking through the 3D scene, turning view orientation, and selecting objects for studying about them according to their text descriptions. The user interactions are generated to as control events and transferred to the server by the event-generator (see figure 1). Each control event is formed by the Event_Action structure as shown in figure 2. *ActionType* defines type of the interactions (e.g. ACT_WALK or ACT_SELECT). *EventType* defines interface-button or mouse events (e.g. mouse move, mouse click, etc.) *MouseFlag* records statuses of mouse click (e.g. left or right). *C_MousePosition* and *P_MousePosition* record the current and previous position of mouse events based on the size of displayed image.

2.3 Streaming Protocol

Data transmission between the client and the server are conveyed through sockets. User diagram protocol (UDP) is often the favoured transport protocol for delivery of media data over the network [23]. However, as UDP, the sender can continuously send data to the receiver without guarantee the delivery of transmitted data [7]. Therefore, the media streaming system needs to be supported extra information to manage the delivery of lost, delayed, and unordered packets. A real-time media streaming system usually uses UDP with supplementation of framing and feedback information that provided by real-time transport protocol (RTP) [24].

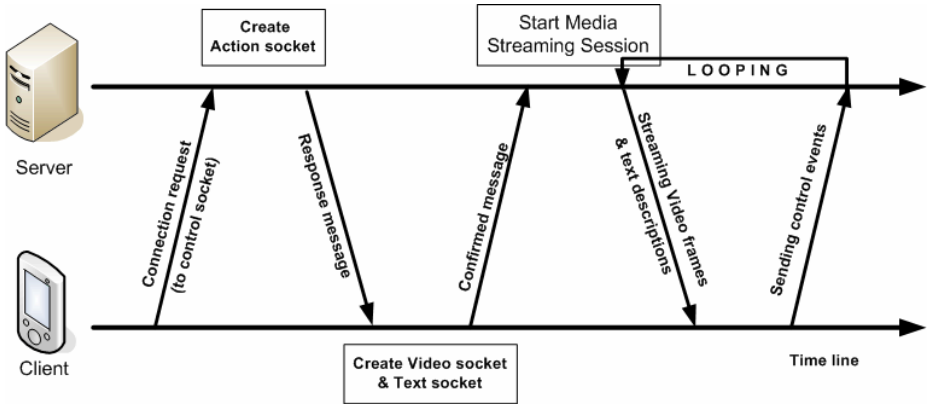

Fig. 3. A mechanism for establishing sockets of the interaction and streaming sessions

In this paper, the socket connections include *control socket*, *action socket*, *video socket,* and *text socket.* The control socket is used to establish and manage all communications between the client and the server. The control socket always listens request messages from the client. User interactions from the client are sent to the server over the action socket. The video socket is created for transferring video stream data. Text socket is created for transferring text descriptions. Figure 3 depicts a mechanism of establishing socket connections for interaction and media streaming sessions. To access and visualize the 3D scene, the client first sends a request message to the server over the control socket. The server creates an action socket and sends a response message to the client. The client then sends the user preferences and device information to the server for starting interaction session and also creates from the server and it again sends a confirmed message to the server. After that, the server opens media streaming session for transferring the media contents to the client.

3 Implementation and Results

To verify the graphics adaptation framework, an interactive 3D visualization (inter3DV) system is developed. We implemented an adaptive media streaming (AMS)

server and client viewer. An implementation based on a central workstation is configured in which the AMS server is running on. The central workstation is equipped with a single Pentium IV, 1.8 GHz, 1 GB of RAM and running on WindowXP platform. Client nodes are HP iPAQ PocketPCs working on WinCE 2005 platform. PocketPC devices currently have a small screen resolution - 240x320 pixels and 16/32 bit color depth. In this experiment, MPEG-4/H.264 standard is used to encode the input images into video stream and UDP/RTP is used to packetize and stream generated video data to the client over the network. The size of video compression is Quarter Common Intermediate Format (QCIF – 176x144).

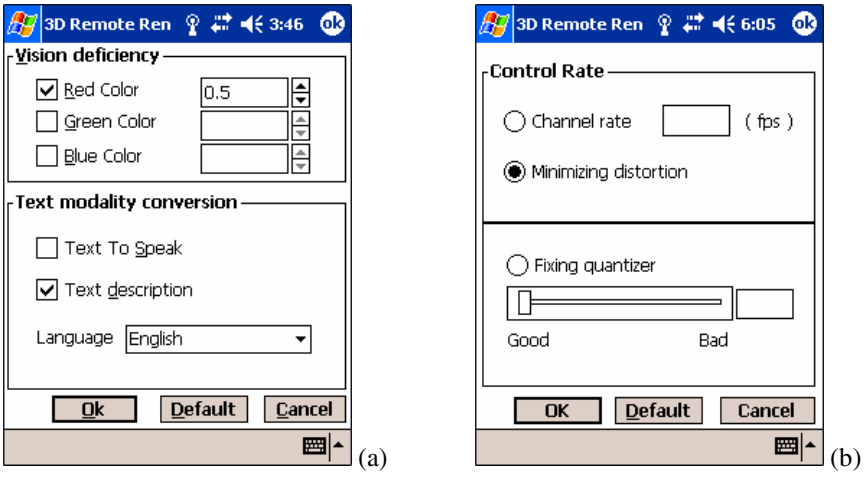

Fig. 4. The user preferences on the client viewer, with (a) vision deficiency and text modality conversion options, (b) control rate option

Figure 4 is two screenshots of the user preferences. In figure 4.a, the user can setup the vision deficiency characteristics based on three primitive colors (Red, Green, and Blue) and text modality conversion such as text-to-speak (TTS), text description, and language for presenting text modality conversion. In figure 4.b, the user can select one of three control rate options. The channel rate allows the system to fix the frame rate and tries to adjust the quality of video compression by finding an optimal quantizer. Similar to the minimizing distortion option, the system adjusts the quality of video compression by finding an optimal quantizer in order to obtain a minimum distortion for decoded video data in the client side. The quantizer option is that the user chooses a quantizer level for the encoding process. The default option for control rate preference is minimizing distortion. We have evaluated the frame rate for HP iPAQ PocketPC that is varied from 14 to 20 fps when the user chooses default minimizing distortion option. Figure 5.a and 5.b show results of choosing the channel rate option with 22 fps and 25 fps, respectively.

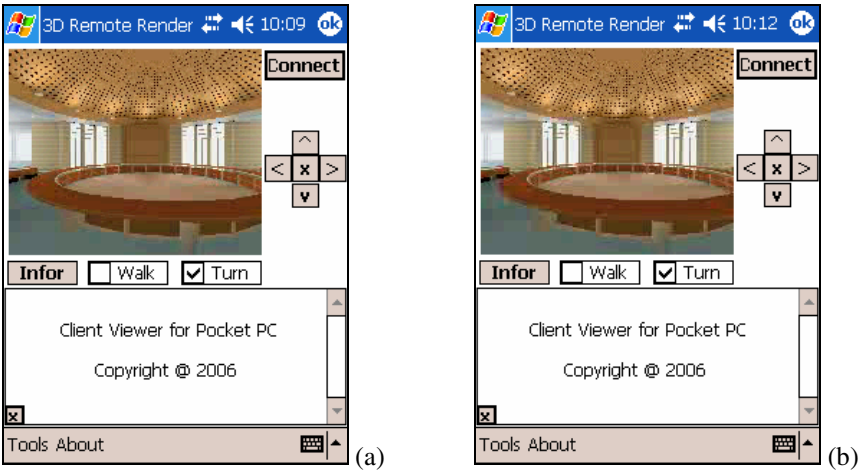

Fig. 5. The screenshots of the client viewer based on the two channel rates (22 and 25 fps)

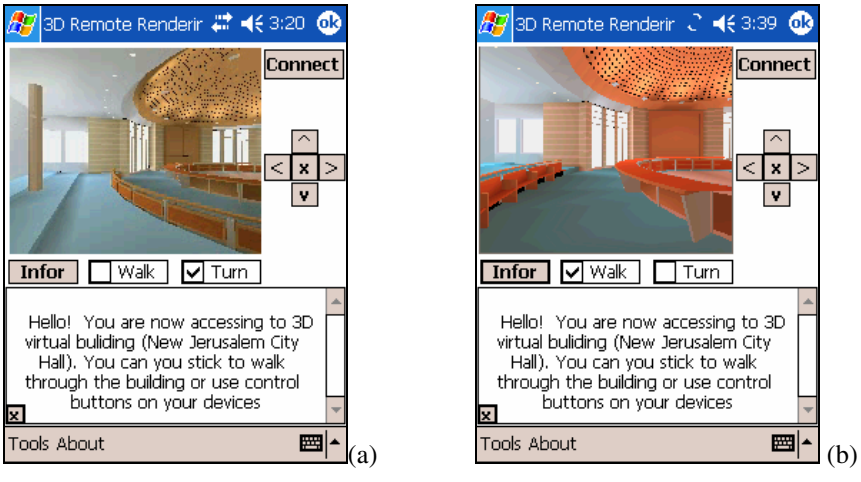

Fig. 6. The screenshots of the client viewer for navigating the 3D virtual building, (a) without setting of vision deficiency, (b) with setting of vision deficiency for red-color at degree of 0.5

Figure 6 is a demonstration for the user walking through the 3D virtual building (New Jerusalem City Hall). Figure 6.a and 6.b are results at two different viewpoints. In figure 6.a, the video frame is represented with the original color. Figure 6.b shows an instance of color adaptation that describes color vision deficiency characteristics of the user who has a certain type and degree of color vision deficiency. In this example, red-color is set as color deficiency of the user and the degree is equal to 0.5.

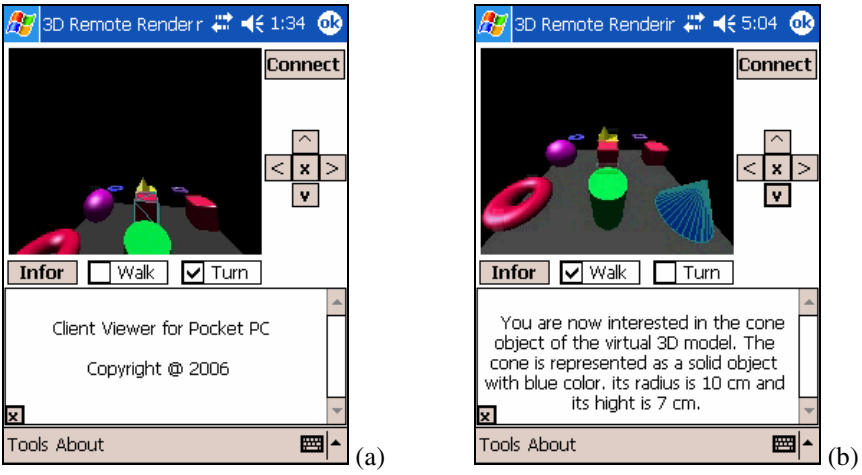

Fig. 7. The screenshots of the client viewer for navigating the primitive 3D object model and selecting object of interests and querying its text description

Figure 7 shows a demonstration for walking through the primitive 3D object model and selecting an interested object. In figure 7.a, the user selected the cube object and queried its text description that was not existed in the Media DB so that the AMS server can not serve. In figure 7.b, the user selected the cone object and queried its description that was displayed in the user interface. The text description of the cone object is a simple one for introducing about the chosen object.

4 Conclusion

In this paper, we proposed a graphics adaptation framework for an inter3DV system. The system takes the advantages of video streaming technique for reducing burden of storing and rendering process on the client side as well as reducing transmitted data (the end-to-end transmission delay). Our proposed system works well for mobile devices with low computational power, memory capacity.

The Inter3DV system has been implemented with an AMS server and a client viewer. We have experimented on 3D virtual building (New Jerusalem City Hall), and a primitive 3D object model. The client viewer, run on a HP iPAQ PocketPC device, allows the user to rapidly represent, easily navigate a complex 3D scene and interact with interested objects for studying according to their text descriptions. Responding to the user preferences, interactions, and the condition of wireless, the media contents are adaptively represented on the client.

Acknowledgments. This work was supported by Seoul City Project. The authors would like to thank the related members of Dept. of Computer Software in Sejong University.

References

1. Teler, E., Lischiniski, D.: Streaming of complex 3D Scenes for Remote Walkthroughs. In: Eurographics, vol. 20(3), pp. 17–25. Blackwell Publishing, Oxford (2001)
2. Schneider, B.-O., Martin, I.M.: An adaptive framework for 3D graphics over networks. Computers & Graphics 23(6), 867–874 (1999)
3. Hesina, G., Schmalstieg, D.: A Network Architecture for Remote Rendering. In: Boukerche, A., Reynolds, P. (eds.) Proceedings of Second International Workshop on Distributed Interactive Simulation and Real-Time Applications, pp. 88–91 (1998)
4. Hugues, H.: Progressive Meshes. Computer Graphics. In: SIGGRAPH '96 Proceedings, vol. 30, pp. 99–108 (1996)
5. Funkhouser, T.A., S´equin, C.H.: Adaptive Display Algorithm for Interactive Frame Rates During Visualization of Complex Virtual Environments. In: ACM SIGGRAPH, pp. 247–254 (1993)
6. Agarwal, P.K., Har-Peled, S., Wang, Y.: Occlusion Culling for Fast Walkthrough in Urban Areas. In: Proc. Eurographics 2001 (September 2001)
7. AlRegib, G., Altunbasak, Y.: 3TP: An Application-Layer Protocol for Streaming 3-D Models. Multimedia, IEEE Transactions on 7(6), 1149–1156 (2005)
8. Shum, H.-Y., Kang, S.B.: A Review of Image-Based rendering techniques. In: IEEE/SPIE Visual Communications and Image Processing (VCIP), pp. 2–13 (2000)
9. Cuntz, N., Klein, J., Krokowski, J.: Real-time Navigation in Highly Complex 3D-scenes Using JPEG Compression. In: Proceedings of 4. GI-Informatiktage (2002)
10. Engel, K., Sommer, O., Ertl, T.: A Framework for Interactive Hardware Accelerated Remote 3D Visualization. In: Proc of EG/IEEE TCVG Symposium on Visualization, pp. 167–177 (2000)
11. Koller, D., Levoy, M.: Protecting 3D graphics contents. Communications of the ACM 48(6), 74–80 (2005)
12. Chang, C.-F., Ger, S.-H.: Enhancing 3D Graphics on Mobile Devices by Image-Based Rendering. In: IEEE Pacific Rim Conference on Multimedia, pp. 1105–1111 (2002)
13. Jeschke, S., Wimmer, M., Purgathofer, W.: Image-Based representations for Accelerated rendering of complex scenes. In: EUROGRAPHICS, pp. 1–20 (2005)
14. Khodakovsky, A., Schröder, P., Sweldens, W.: Progressive Geometry Compression. In: SIGGRAPH, pp. 271–278 (2000)
15. Gueziec, A., Taubin, G., Horn, B.: A framework for streaming Geometry in VRML. IEEE Computer Graphics and Applications 19(2), 68–78 (1999)
16. Cohen-Or, D., Levin, D., Remez, O.: Progressive Compression of Arbitrary Triangular Meshes. In: the Proceedings of Visualization, pp. 67–72 (1999)
17. Chow, M.M.: Optimized Geometry Compression for Real-time Rendering. In: Proc. IEEE Visualization'97, Computer Society Press pp. 347–354 (1997)
18. Joint Video Team of ITU-T and ISO/IEC JTC 1, Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC), document JVT-G050r1 (May 2003)
19. Nam, J.Y., Man, R., Huh, Y., Kim, M.: Visual content adaptation according to user perception characteristics. IEEE Transactions on Multimedia 7(3), 435–445 (2005)
20. Hsu, C.-Y., Ortega, A., Khansari, M.: Rate control for robust video transmission over burst-error wireless channels. IEEE Journal on Selected Areas in Communication 17(5), 756–773 (1999)
21. OpenGL.: http://www.opengl.org
22. SGI - OpenGL Performer.: http://www.sgi.com/products/software/performer/