ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal) | Impact Factor: 7.249|

Visit: www.ijirset.com

Vol. 8, Issue 1, January 2019

3D Rendering on Mobile Devices: A Challenge

Vishal Verma

Department of Computer Science, M L N College, Yamuna Nagar, India

me_vishaal@hotmail.com

ABSTRACT: With the advancement of technology, mobile devices have now become an important computing platform. They are portable and suitable for variety of work environments. Nowadays, these devices are capable of supporting complex applications such as graphical user interface, audio and video playback, wireless communication etc. However, rendering 3D (Three-dimensional) graphics on these devices is still considered a difficult task. The main reason is that mobile devices have limited CPU speed, limited memory, no graphics hardware, a small display with lower resolution, and the battery of device can't last for long compared to a personal computer. Moreover, the absence of hardware accelerators (which are available on PC-compatible cards) makes the possibility of 3D graphics challenging on the mobile devices and PDAs. Besides these challenges, there has not yet evolved a single standard for graphics on hand held devices. Solutions have been suggested for these problems but the most suitable solution is yet to come in. For these reasons, research on mobile 3D rendering is still limited, but now with the advancement of mobile devices, it is increasingly easier to experiment with mobile 3D. This paper explores the solutions like remote rendering, image based rendering and their variants available to render 3D Graphics on mobile devices.

I. GRAPHICS ON MOBILE DEVICES

The early mobile phones were bulky blocks with separate handsets. They were designed to be lugged around rather than carried in a pocket, and they operated using analog radio networks. With the advancement in technology, mobile phones started to become truly portable rather than just movable in the early 1990s. Mobile phones were pocket-sized by that time, but they were still primarily used for talking.

Eventually, features like address books, alarm clocks, text messaging etc. started to appear. The early alphanumeric displays replaced with dot matrices, and simple games, like the Snake started to available in many Nokia phones. Calendars and email applications quickly followed. In the late 1990s, the mobile phone feature palette has exploded with FM radios, color displays, cameras, music players, web browsers, and GPS receivers. The technological advancements again added lots of new features in mobile phones viz. high resolution display quality, storage memory in gigabytes, high processing power and advance operating systems. Apart from this, the phones became compatible to run a plethora of applications.

II. MOBILE GRAPHICS STANDARDS

Since mobile technologies are getting better, interactive 3D graphics is becoming feasible. The initial standardized 3D programming interfaces for mobile devices available to hardware vendors and application developers include OpenGL ES for native C/C++ and Mobile 3D Graphics (M3G) for Java applications [1].

2.1 OpenGL ES

OpenGL is the most widely adopted low-level cross platform graphics API. However, OpenGL in itself is too large for mobile devices. It needed to be cleaned of rarely used and redundant functionality and adapted to better address the restrictions of mobile devices. The Khronos Group set out to do these changes and create OpenGL ES, where ES stands for embedded systems. This standard 3D graphics API for embedded systems makes it easier and affordable to offer a variety of advanced 3D graphics and games across all major mobile and embedded platforms. With the introduction of OpenGL ES [2], Mobile 3D graphics have made significant progress. Manufacturers are now adding hardware graphics

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal) | Impact Factor: 7.249|

Visit: www.ijirset.com

Vol. 8, Issue 1, January 2019

to the new generation of mobile devices. However, the lack of hardware accelerated graphics in mobile devices has motivated the development of Klimt. Klimt is an open-source software library that implements the OpenGL ES interface.

2.2 M3G

Java 3D is a high-level API that provides extensive support for scene graph operations and some support for animation and compressed geometry. Therefore, a subset of Java 3D could probably be designed so it can be implemented on top of OpenGL ES, allowing both APIs to benefit from the same underlying rendering engine. The JSR- 184 Expert Group (EG) not only make a subset of Java 3D to design M3G [3], but modify and extend it so much that it would have become a different API in the end. The new design offers essentially the same core functionality as Java 3D, but avoids the redundant methods and overly generalized class hierarchies that hamper Java 3D. As a result, the API footprint (the number of classes and methods) is an order of magnitude smaller than that of Java 3D.

III. 3D RENDERING ON MOBILE DEVICES

Despite of the availability of abundance of 3D rendering literature, 3D rendering on mobile devices is still a scarcely explored topic. Typical 3D applications on mobile devices include 3D games, 3D environmental analysis, virtual reality, personal navigation and many geographic information system (GIS) applications. To realize the efficient and real-time 3D Rendering computer graphics rapidly advancing in mobile devices. Even many 3D graphics libraries such as OpenGL ES, M3G etc are also available. Still, mobile devices have limitations of memory and processing power. Furthermore, the most of them don't offer hardware accelerated graphics. As a result, it is difficult to render and interact with large 3D scenes on these devices. Many approaches were proposed in the literature to render interactive 3D scenes on mobile devices. We can divide these approaches in to two main categories:

- Remote Rendering, and
- Local Rendering.

In addition to above two approaches, some researchers also propose hardware support [4, 5, 6] for 3D rendering on mobile devices. These researches aim at producing very small graphic accelerator chips with high performance and low power consumption. Unfortunately, this approach lacks flexibility, because new hardware must be developed for every new emerging rendering technique. Moreover, if multiple, diverse applications have to be running on the same mobile device, adding hardware accelerator for each application incurs a high cost w.r.t hardware area and power consumption.

3.1. Remote Rendering

Remote rendering is the most common solution adopted in the literature for mobile devices. In this type of rendering, the process is carried out on a remote computer with powerful graphic acceleration and the results are sent to the mobile device using a wireless network. As a result, the mobile device is just a simple client. Remote rendering has been used to display complex objects, such as 3D anatomy models, on mobile devices [7, 8, 9], and implement augmented reality systems [10].

The need for a wireless network is a major disadvantage of remote rendering solutions: wireless networks can't be set up for every possible environment. Moreover, wireless networks usually have limited bandwidth and these solutions need complex algorithms for the preparation of data to be sent to the client.

3.2. Local Rendering

There are three different concepts to achieve interactive 3D rendering on mobile devices, among them are

- Polygon-based rendering [11],
- Image-based Rendering [12, 13], and
- Point-based Rendering [14, 15]

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal) | Impact Factor: 7.249|

Visit: www.ijirset.com

Vol. 8, Issue 1, January 2019

3.2.1 Polygon-based rendering

Traditional Polygon-based Rendering or Geometry-based Rendering (GBR) algorithms uses polygons as primitives which makes rendering complexity dependent on complexity of the model to be rendered. Such systems typically use their own graphics library, although some systems use PocketGL [16]. These systems render 3D scenes with a software rasterizing using the native 2D API for the graphics device.

In [17], a 3D viewer for PocketPC devices is presented which offers a stand-alone solution for accessing 3D models during stages of the manufacturing process. However, the problem of this system is that it sends the complete model to the viewing device, instead of sending a simplified version. Therefore, it is not very well suited for large scale geometric models. Zunino et al. [18] presented a PocketGL-based continuous multiresolution rendering engine for PDAs. The engine can dynamically adjust the objects detail level according to the target frame rate specified by the user. The frame rate is considered as an objective function that determines both the quality and the interactivity of the scene. The viewer requires that the geometric models must be stored in the PDA's memory. This limits the amount of geometry that can be processed by the viewer. Sanna et al. [19] proposes a general architecture for searching, retrieving and rendering complex 3D models on PDA's. The architecture includes rendering servers, geometry servers and client applications that display the resulting images.

3.2.2 Image-based rendering

Image-based Rendering (IBR) techniques use images to substitute some or all of the scene's geometry in 3D rendering. Using precomputed images speeds up the rendering algorithm and provides realistic effects at a low cost. Images can also be rendered at a server system and sent to a client system for remote display. Chang and Ger [12] suggest a client-server system that renders geometry at the server and sends the generated images to a PocketPC client for display. The depth information is also provided with images. Using the depth information and 3D warping technique, the client can create new images without more input from the server. V´azquez and Sbert [20] present a client-server system that delivers new images to a remote PC. Instead of sending entire images, the system uses a prediction technique to determine which pixels need to be updated for each new frame. This significantly reduces the bandwidth requirements of the client-server communication. Woodward et al. in their publication [21] describes a server system that sends a compressed video stream to a client. The stream has been rendered using the client-provided camera parameters. This system enables engineers to view and interact with 3D CAD files using a mobile phone connected to a personal data assistant (PDA).

3.2.3 Point-based rendering

Point-based Rendering (PBR) techniques have becoming popular for rendering complex scenes because the traditional graphics pipeline spends a lot of time in transforming and rasterizing several geometric primitives that may not be visible on the screen. Point-based rendering displays complex geometries by rendering a set of points that are located on the surface of the objects. Thus, the number of point samples rendered depends on the complex object's screen size.

One approach to point-based rendering is to create unstructured point sets by generating point samples whereas another approach is to generate structured point sets, for instance by creating a hierarchy. Structured point sets hierarchy also permits visualization of complex models, especially those that don't fit in main memory. Such algorithms create a hierarchy of bounding volumes and attains flexible rendering by stopping the descent into the hierarchy. Intermediate normal and color attributes are stored in this hierarchy. Duguet and Drettakis [14] use a structured point sets hierarchy representation for rendering. The method supports traversal of the hierarchy in a specific order, resulting in a one-pass fast shadow-mapping algorithm.

The major challenge of point-based rendering (PBR) algorithms is to produce a continuous interpolation between discrete point samples that are unevenly distributed over a smooth surface. Furthermore, correct visibility as well as efficient level-of-detail (LOD) must be supported for rendering of large data sets.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal) | Impact Factor: 7.249|

Visit: www.ijirset.com

Vol. 8, Issue 1, January 2019

IV. COMPARATIVE ANALYSIS

Table 2.1 contrasts between image-based rendering (IBR) and geometry-based rendering (GBR) on mobile devices.

Technique	Rendering Load	Complex Scene Support	Interaction	Network Load	Display Quality
GBR	High	Low	Complex	High	Low Polygon
IBR	Low	High	Simple	Moderate	High Resolution

Table 2.1 IBR Vs GBR on Mobile Devices

GBR renders geometry on the client and therefore complex scenes can't be supported and only low polygon, low quality textured models can be displayed. It consumes a lot of bandwidth to stream the 3D models. Techniques discussed in section 3 are used to minimize the network load. GBR offers the remote user more interactivity in terms of user navigation freedom and scene changes where IBR usually delivers static scenes to the client.

IBR is lighter to render on the client since rendering images is much faster and lighter than rendering geometry. As a result, complex environments can be rendered with high resolutions since IBR is independent of scene complexity, but instead depends on the image resolution. Furthermore, IBR consumes less bandwidth.

V. CONCLUSION

To make rendering on mobiles possible, one approach is to rely on a client/server architecture in which server computes images of 3D scene and sends them to a client which visualize them on its screen. As this approach is highly demanding in terms of network bandwidth, a preferable approach is to distribute rendering among the server and the client. In fact, the server computes a set of key images that are sent to a client which computes in-between images using IBR techniques.

While traditional rendering methods need data representing the geometry and the photometry of the objects making up a 3D scene, IBR methods uses as input a set of images (synthetic or real) sometimes augmented with depth maps. When rendering complex scenes, the computation cost of traditional rendering is proportional to the number of objects within a scene, while it is only proportional to the image resolution for IBR methods. IBR techniques proved that they are fast and easy to implement. They only consist in calculating for intermediate camera positions, in-between frames from key frames and depth-maps.

Therefore, it is beneficial to use IBR methods in order to complex scenes on a mobile device within the context of a client/server architecture. The mobile device is the client of a server which computes a very small set of key images (to avoid latency time that would affect interactivity) of a complex 3D scene and transmits them to the client on demand through a low bandwidth network. The client uses these images to compute new images as seen by intermediate cameras (using an IBR technique) the positions and directions of which are interactively selected by the user via a mobile device.

REFERENCES

[1] Aarnio, T., Miettinen, V., Roimela, K., & Vaarala, J. (2008). Mobile 3D Graphics: With OpenGL ES and M3G. The Morgan Kaufmann Series in Computer Graphics. Elsevier Science & Technology.
[2] Lluch, J., Gaitán, R., Camahort, E., & Vivó, R. (2005, June). Interactive three-dimensional rendering on mobile computer devices. In Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology (pp. 254-257).

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal) | Impact Factor: 7.249|

Visit: www.ijirset.com

Vol. 8, Issue 1, January 2019

[3] Hosseini, M., Ahmed, D. T., & Shirmohammadi, S. (2012, February). Adaptive 3D texture streaming in M3G-based mobile games. In Proceedings of the 3rd Multimedia Systems Conference (pp. 143-148).
[4] Nam, B. G., Kim, H., & Yoo, H. J. (2008). Power and area-efficient unified computation of vector and elementary functions for handheld 3D graphics systems. IEEE Transactions on Computers, 57(4), 490-504.
[5] Sohn, J. H., Woo, R., & Yoo, H. J. (2004, August). A programmable vertex shader with fixed-point SIMD datapath for low power wireless applications. In Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware (pp. 107-114).
[6] Marek, T., & Krejcar, O. (2015). Optimization of 3d rendering in mobile devices. In Mobile Web and Intelligent Information Systems: 12th International Conference, MobiWis 2015, Rome, Italy, August 24-26, 2015, Proceedings 12 (pp. 37-48). Springer International Publishing.
[7] Lamberti, F., & Sanna, A. (2007). A streaming-based solution for remote visualization of 3D graphics on mobile devices. IEEE transactions on visualization and computer graphics, 13(2), 247-260.
[8] Lamberti, F., Zunino, C., Sanna, A., Fiume, A., & Maniezzo, M. (2003, March). An accelerated remote graphics architecture for PDAS. In Proceedings of the eighth international conference on 3D Web technology (pp. 55-ff).
[9] Shi, S., & Hsu, C. H. (2015). A survey of interactive remote rendering systems. ACM Computing Surveys (CSUR), 47(4), 1-29.
[10] Chatzopoulos, D., Bermejo, C., Huang, Z., & Hui, P. (2017). Mobile augmented reality survey: From where we are to where we go. Ieee Access, 5, 6917-6950.
[11] Pulli, K., Aarnio, T., Roimela, K., & Vaarala, J. (2005). Designing graphics programming interfaces for mobile devices. *Computer Graphics and Applications, IEEE*, 25(6):66-75.
[12] Chang, C. F., & Ger, S. H. (2002). Enhancing 3D graphics on mobile devices by image-based rendering. In Advances in Multimedia Information Processing—PCM 2002: Third IEEE Pacific Rim Conference on Multimedia Hsinchu, Taiwan, December 16–18, 2002 Proceedings 3 (pp. 1105-1111). Springer Berlin Heidelberg.
[13] Lluch, J., Gaitán, R., Camahort, E., & Vivó, R. (2005). Interactive three-dimensional rendering on mobile computer devices. *ACE '05: ACM SIGCHI*, pp. 254-257. New York, NY: ACM.
[14] Duguet, F., & Drettakis, G. (2004). Flexible point-based rendering on mobile devices. IEEE Computer Graphics and Applications, 24(4), 57-63.
[15] He, Z., & Liang, X. (2006, November). A Point-Based Rendering Approach for Mobile Devices. In 16th International Conference on Artificial Reality and Telexistence--Workshops (ICAT'06) (pp. 26-30). IEEE.
[16] Jackson, J., & Dixon, M. R. (2007). A mobile computing solution for collecting functional analysis data on a pocket PC. Journal of Applied Behavior Analysis, 40(2), 359-384.
[17] D'Amora, B., & Bernardini, F. (2003). Pervasive 3D viewing for product data management. IEEE Computer Graphics and Applications, 23(2), 14-19.
[18] Zunino, C., Lamberti, F., & Sanna, A. (2003). A 3d multiresolution rendering engine for pda devices. In Titolo volume non avvalorato (pp. 538-542).
[19] Sanna, A., Zunino, C., & Lamberti, F. (2004). A distributed architecture for searching, retrieving and visualizing complex 3D models on Personal Digital Assistants. International journal of human-computer studies, 60(5-6), 701-716.
[20] Vázquez, P. P., & Sbert, M. (2002, April). Bandwidth reduction techniques for remote navigation systems. In International Conference on Computational Science (pp. 249-257). Berlin, Heidelberg: Springer Berlin Heidelberg.
[21] Woodward, C., Valli, S., Honkamaa, P., & Hakkarainen, M. (2002, May). Wireless 3d cad viewing on a pda device. In Proceedings of the 2nd Asian International Mobile Computing Conference (AMOC 2002) (Vol. 14, p. 17).