ACCELERATED VIDEO STREAMING FOR GAMING ARCHITECTURE

Arto Laikari, Philipp Fechteler, Benjamin Prestele, Peter Eisert, Jukka-Pekka Laulajainen
arto.laikari@vtt.fi, {philipp.fechteler, benjamin.prestele, peter.eisert}@hhi.fraunhofer.de, jukka-pekka.laulajainen@vtt.fi

VTT Technical Research Centre of Finland
Software architectures and platforms
Vuorimiehentie 3, Espoo, FI-02044 VTT, Finland

Fraunhofer Institute for Telecommunications - Heinrich-Hertz-Institute
Image Processing Department
Einsteinufer 37, D-10587 Berlin, Germany

ABSTRACT

Computer game processing requirements for the CPU and graphics performance are growing all the time higher. At the same time many low cost and modest performance CE devices are gaining popularity. People are already used to mobile life style inside home and on the go and they want to enjoy entertainment everywhere. This paper describes an accelerated video streaming method used in a gaming system, called Games@Large, which enables heavy PC game playing on low cost CE devices without the need of game software modification. The key innovations of the Games@Large system are game execution distribution, game streaming, including graphics/video and audio streaming and game control decentralization as well as network quality of service management. This paper concentrates on the advanced video streaming techniques used in Games@Large.

Index Terms — games, graphics, rendering, video coding, video codecs

1. INTRODUCTION

Future home is considered to be an always-on connected digital home with wide variety of appliances. Entertainment equipment have already conquered the homes, although some other future home appliances are still not so popular.. Computer gaming has been utilizing network infrastructures at home already for a long time. Modern games have become highly realistic and they are consumed by a wide population, not only youngsters. As the games have turned to be realistic connected virtual worlds they have become even more demanding towards the computer hardware. High CPU processing power and graphics performance is required to play these games.

Mobility and digital home entertainment appliances have generated the desire to play not only in front of a home PC, but everywhere inside the house and also on the go. As the result of TV evolution, Set-Top Boxes (STBs) have entered homes and mini-laptops have gained popularity. Several low cost consumer electronic end devices (CE) are already available at home. Although these devices are capable to execute software, modern 3D computer games are too heavy for them. Games@Large system enhances the usage of these modest execution power CE devices also to heavy PC game playing terminals without the need of modifying the games. Commercial games are supported by the Games@Large platform regardless of the 3D technique used in the game, which can be either DirectX or OpenGL, because the platform provides cross-streaming from DirectX to OpenGL and vice-versa.

The Games@Large project [1] has developed a system for gaming both for homes and for enterprise environments, like hotels, internet cafés and elderly homes. [1, 5]

In the Games@Large system the key concepts are execution distribution, audio and graphic streaming and rendering and decoupling of input control commands. Games are executed in one or more servers and the game display and audio is captured and streamed to the end device, where the stream is rendered and the game is actually played. Game control is captured at the end device and streamed back to the server and injected to the game.

There is a number of commercial Gaming on Demand systems, overviewed in [5], that have been presented to the market. More recently, there have been some new announcements about the upcoming systems such as Playcast Media Systems, Gaikai's Streaming Worlds technology, Onlive and GameTree.tv from TransGaming Technologies. However, there is very little detailed technical information publicly available about the commercial systems.

This paper presents briefly the Games@Large architecture and concentrates on the accelerated video streaming system used in the project. The work builds on the basic video encoding presented in [5] adding advanced methods especially suitable for gaming purposes. In [5] the video streaming is presented as a research proposal and in this paper more details and results of the implemented solution are presented.

2. GAMES@LARGE ARCHITECTURE

The Games@Large system depicted in **Figure 2** consists of three major element classes: servers, end devices and access points. Games are played on the end devices and executed on the servers. Games run on the Local Processing Server (LPS), which utilizes also the Local Storage Server (LSS). In the home version, these logical entities will be located in the same physical computer. In an enterprise version, the server entities are

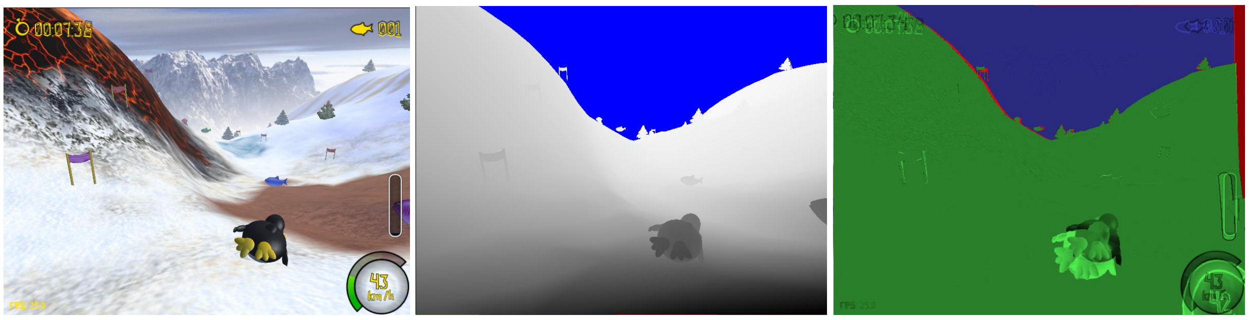

Figure 1: Input frame-buffer from game (left), corresponding depth map with blue colored skybox region (middle) and difference image of input frame-buffer and per pixel prediction based on previous frame (right)

distributed into several physical computers. End devices, like Setup-top-Boxes (STB) or Enhanced Multimedia Extenders (EME), Enhanced Handheld Devices (EHD) and notebooks are connected to the server either with a wireless or wired connection.

The system exploits an adaptive streaming architecture and uses Quality of Service (QoS) functionalities to ensure good quality gaming to a variety of devices connected to the wireless network. The details of each component have been presented in earlier publications *[1,5]*

Games@Large platform interacts with the games without the need to make any modifications to the games.This brings the advantage that there is no need to create specific Games@Large- designed games, but almost any existing game can be used with the platform as they are. Platform provides also the cross steaming option from DirectX to OpenGL and vice-versa, which makes it possible to mix various operating systems for the end devices, which are used to play the games. Because the games are executed in the server, the end device does not need to possess the processing power, which the game would require, if it would be run natively on the client.

The objective of the streaming architecture is to support various end devices with an efficient and high quality game experience, independent of software or hardware capabilities. To meet these demands a streaming architecture has been developed that is able to support two streaming strategies to display the game remotely: graphics and video streaming.

Graphics Streaming is used for end devices with accelerated graphics support, like computers or set-top-boxes, typically having screens of higher resolution. Here the graphics commands from the running game are captured, transmitted and rendered locally on the end device. In this way, high image quality is achieved, since the scenes are always directly rendered for the desired screen.

The alternative approach, video streaming, is used mainly for end devices without a GPU, like handheld devices, typically having screens of lower resolution. Here the graphical output is rendered on the game server and the frame-buffer is captured and transmitted encoded as H.264 video stream. In Video Streaming the bit rates are in general much more predictable in comparison to Graphics Streaming. However, H.264 encoding on server side as well as decoding on end devices is computational demanding. This is described in more detail in the next chapter. Skybox/skysphere techniques are used in many current 3D games, which are on the market. Games@Large platform can exploit this during the encoding for acceleration as is described in this paper and more detailed in [2].

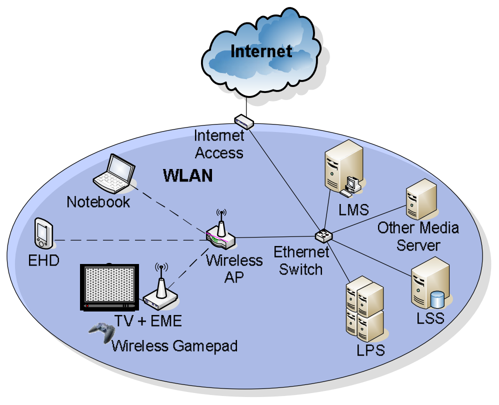

Figure 2: Games@Large architecture

3. ACCELERATED VIDEO STREAMING

3.1 Low complexity H.264 video encoding on server-side

Since video encoding is typically demanding in terms of computational load and because the CPU is already consumed by the running computer game, as well as additional gaming sessions, several optimizations have been developed in order to reduce the computational complexity of encoding. The basic idea is to exploit additional information which is accessible from the render context. This is achieved by overloading the original graphics library (described in [3] for Linux and OpenGL) without the need of any game modifications. This idea has already been published in [5] as a research proposal, without any details or results, which are presented here. In contrast to adaptive streaming techniques, where the video stream is adapted to the client after its creation, e.g. during transmission [6], in the Games@Large framework the visual output is already generated with respect to the clients capabilities. As each graphics command issued by the game passes through our proxy DLL library, the parameters are adapted to render images which are optimal for streaming to the corresponding client: for example the viewport is adapted to the client's screen size, so that no additional image adjustment besides video encoding is needed. This in turn reduces the delay, computational load and provides maximum image quality, because e.g. rescaling artifacts do not occur.

A method called skybox/skysphere, which is used by many 3D computer games, can be exploited to achieve acceleration in H.264 encoding [2]. The skybox technique is used to render the far away environment, e.g. sky, mountain panoramas etc. Since the depth buffer is disabled during skybox rendering, such regions can be detected unambiguously by checking the z-buffer, as can be seen as the blue region in the middle of **Figure 1**. Since skybox regions are homogenous moving textures, corresponding macroblocks need not to be split into smaller partitions, and assigning one common motion vector is sufficient. By identifying the macroblocks which reside completely within skybox regions, further checks for partitions with finer granularity can be omitted, hence accelerated encoding is achieved in comparison to macroblock partitioning techniques of generic H.264 encoders.

Additionally, skybox regions are rendered at the beginning of each frame-buffer update so that the scene is rendered in front of it. This allows capturing the projection matrices used to render the skybox regions without any ambiguity. The captured projection information of the current and previous frame provides the basis for the direct calculation of motion vectors, similar to the method presented in [3]. The computationally demanding motion vector search algorithms of generic H.264 encoders are omitted. Results of per pixel prediction in skybox regions are shown as blue area of the right image in **Figure 1**. The impact gained by exploiting the skybox region and corresponding projection information for macroblock partitioning and direct motion vector calculation is depicted in **Figure 3**.

In order to calculate the motion vectors directly for non-skybox regions, suitable projection information is needed. For major parts the projection information can be captured by intercepting each drawing command, registering the used projection matrices and analyzing which projection matrices where used for most objects. By choosing these most-used projection matrices for prediction, the calculated motion vectors are sufficient for major parts of the frame. In cases where other scene elements were rendered, like figures etc, a threshold on the rate-distortion-cost may be used to decide for such disadvantageous cases to fall back to generic H.264 motion vector search algorithms. Results of per pixel predictions for the scene are shown in the green part of the right image of **Figure 1**. The acceleration we achieved by integrating the direct motion vector calculation into the H.264 encoder can be seen in **Figure 3**.

The H.264 encoded video is then transmitted to the client, using standard compliant RTP over UDP streaming protocols.

3.2 Client for low delay video streaming

While any standard compliant client software will be able to play back the RTP packetized H.264/AAC streams, most media players do not fulfill the requirements of very low delay decoding and rendering. They are usually implemented with the focus on smooth and reliable playback of accurately synchronized audio and video: meeting these goals in a robust fashion, especially in the case of wide area networks with potentially large network jitter, inevitably requires generous buffering at many stages of the processing chain. This in turn results in typical delays of approx. 100ms to few seconds with most players. Such large delays on the client side would obviously not be acceptable for gaming and also cancel out much of the effort put into low delay encoding and streaming on the server side and the networking layer.

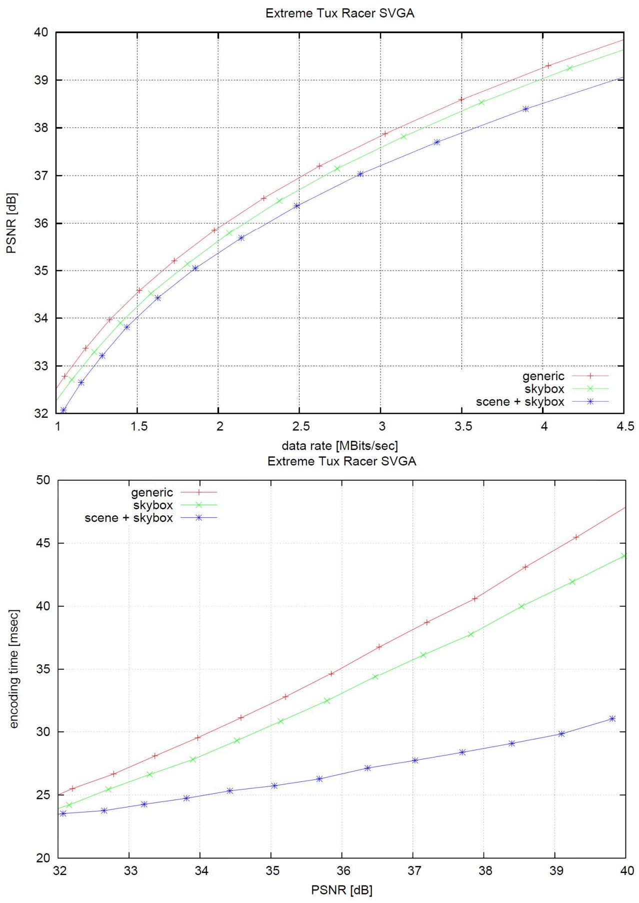

Figure 3: Results of H.264 encoding based on skybox and scene enhancement

However, the specific scenario and system design allow relaxing some of the aforementioned restrictions. Firstly, the network is a dedicated and controlled environment with optional QoS optimizations put in place to reduce network jitter and delay. Secondly, many games use sound only for background music or to provide aural feedback to user interactions, which usually is not vital to the game's functionality. Since audio and video streams are encoded and transmitted with very low delay, and since there is no increasing drift between the streams over time, we found it beneficial for this scenario to waive perfect synchronization in favor of instantaneous playback.

Custom client software has been implemented for this purpose, relaxing on the synchronization requirements and focusing on minimal buffering and instant playback of video and audio instead. While each video frame is being decode and displayed immediately after the corresponding H.264 NAL units have arrived at the client, the decoded AAC audio frames undergo an additional preprocessing step prior to being rendered on the sound card. This showed to be necessary, since even small temporal interruptions of the audio bit stream may result in very distracting discontinuities (clicking noise) in the PCM audio signal to be rendered. Therefore an adaptive retiming algorithm is being applied to the yet unplayed audio samples, which smoothly stretches or compresses available samples to reflect the current arrival speed of new audio samples. This allows the client to compensate for low to medium jittering on the network level with almost no noticeable distortion of the audio, while on average still maintaining minimal buffering and perceived delay.

4. CONCLUSION

Games@Large has implemented an innovative architecture, transparent to legacy game code, that allows distribution of a cross-platform gaming and entertainment on a variety of low-cost networked devices. Virtually extending the capabilities of such devices the Games@Large system is opening important opportunities for new services and experiences in a variety of fields and in particular for the entertainment in the home and other popular environments. This paper has presented the advancements in the field of video encoding optimized for game use.

5. ACKNOWLEDGEMENTS

This work has been carried out in the IST Games@Large project (http://www.gamesatlarge.eu) [1], which is an Integrated Project under contract no IST038453 and is partially funded by the European Commission.

6. REFERENCES

[1] Games@Large project website, http://www.gamesatlarge.eu

[2] P. Fechteler, P. Eisert, "Depth Map Enhanced Macroblock Partitioning for H.264 Video Coding of Computer Graphics Content," in *Proc. of IEEE Int. Conf. on Image Processing (ICIP2009)*, Cairo, Egypt, Nov. 2009, pp. 3441-3444

[3] S. Stegmaier, J. Diepstraten, M. Weiler, T. Ertl, "Widening the Remote Visualization Bottleneck," in *Proc. of 3rd Int. Symposium on Image and Signal Processing and Analysis (ISPA2003)*, Rome, Italy, Sept. 2003, vol. 1, pp. 174– 179.

[4] L. Cheng, A. Bhushan, R. Pajarola, M. El Zarki, "Realtime 3D Graphics Streaming Using MPEG-4," in *Proc. of IEEE/ACM Workshop on Broadband Wireless Services and Applications (BroadWise '04)*, pp. 1–16, San Jose, Calif, USA, July 2004.

[5] A. Jurgelionis, P. Fechteler, P.Eisert, et al., "Platform for Distributed 3D Gaming", in *Intern. Journal of Computer Games Technology*, Article ID 231863, 2009

[6] J. Brandt, L. Wolf, "Adaptive Video Streaming for Mobile Clients", in *18th Intern. Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV2008)*, Braunschweig, Germany, May 2008