ETSI TR 102 794 V1.1.1 (2010-12)

Technical Report

**Media Content Distribution (MCD);
3D Gaming Graphics Delivery Overview**

Reference

DTR/MCD-00010

Keywords

content, transport, protocol

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Contents

Intellectual Property Rights

Foreword

This Technical Report (TR) has been produced by ETSI Technical Committee Media Content Distribution (MCD).

The main purpose of the present document is to provide an overview of the topic of 3D Gaming Graphics Delivery. The present document is meant to create an ETSI MCD view on this topic, and includes use cases, high-level requirements, an overview of potential solutions and a gap analysis.

The present document's conclusions should assist in the definition of new Work Item(s) for actual standardization, provided this is necessary and ETSI MCD is the right body.

Introduction

By 2010, there will be over 420 million broadband households worldwide. With the standard set for super-high speed, always-on connectivity, the way people view entertainment has fundamentally changed and created a new standard for consumption. Consumers no longer expect their internet access to be only from a desktop PC, now they want it through the TV in the living room or in the palm of their hand, inside the house and on the go.

Video games are officially a mainstream leisure time activity, and consumers are increasingly demanding their entertainment to be fully networked and available over multiple platforms. Digital distribution of games is growing and is becoming a common way to acquire games. Several game publishers and game retailers are using this model as a way to reach consumers. The main deployment model being used at the moment of writing the present document is where an end consumer buys the game online after which the game executable and (parts of) the game data are downloaded to his PC or game console, using a run-time environment that protects the game from being copied.

Consumer electronics devices, such as set-top boxes, TVs, mobile phones and Netbook PCs are not considered real gaming devices, since they lack the necessary CPU power, graphical performance and memory/storage capacity. So a download distribution model of 3D games is not very likely to be a viable way to reach consumer electronics devices. Also, there is a large catalogue of existing games that simply are not written with these platforms in mind.

In order to play 3D games on this kind of devices a streaming model is more appropriate. So, instead of downloading the game to the end user device, the game is executed on a high-performance server, after which the game output is streamed to the client. Discussion about the nature and content of what is exchanged between servers and clients constitutes the topic of the present document. Several technical solutions could be possible. The present document will describe the benefits and drawbacks of several of these technical solutions.

The end goal of this work is to standardize the protocol(s) needed for streaming the game output to a client. The present document is meant to create an ETSI MCD view on this topic, and includes use cases, high-level requirements, an overview of potential solutions and a gap analysis to see which actual standardization work needs to be performed.

1 Scope

The present document provides an overview of delivery of 3D graphics of games that are running on a high-performance server to client devices that would otherwise not have the resources to run these games natively. The present document describes the use cases, high level requirements and different solution approaches, and identifies the main area(s) where standardization work would be needed. Exploring the nature and content of what is going to be exchanged between servers and clients constitutes the main topic of the present document

The primary aim for this technology is to enable casual gaming scenarios, and it is not the primary aim to try to satisfy hardcore gamers, since we have to be realistic about what can be achieved with the current state of networking and server technology.

Because of the nature of the content and in order to focus on feasible solutions, we may exclude certain type of network connections to be used, such as using "unreliable" or high-latency wireless connections such as 802.11b/g. We expect to have a broadband connection between the client and the server, without too many intermediate hops.

In the present document we do not address deployment issues, such as integration into a Content on demand infrastructure (e.g. user registration, pricing, purchasing), or integration into a QoS framework (e.g. prioritization of gaming streams compared to other types of streams). Also, use cases and synchronization issues related to multi-player gaming, whereby different connection speeds/types could result in different response times, are out of scope of the present document.

2 References

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found at http://docbox.etsi.org/Reference.

NOTE: While any hyperlinks included in this clause were valid at the time of publication ETSI cannot guarantee their long term validity.

2.1 Normative references

The following referenced documents are necessary for the application of the present document.

Not applicable.

2.2 Informative references

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] Alf Inge Wang, Martin Jarrett and Eivind Sorteberg: "Experiences from Implementing a Mobile Multiplayer Real-Time Game for Wireless Networks with High Latency", International Journal of Computer Games Technology Volume 2009, Hindawi Publishing Corporation.

NOTE: Available at: http://downloads.hindawi.com/journals/ijcgt/2009/530367.pdf.

[i.2] Console Gaming: "The Lag Factor", Eurogamer® Network Ltd., 5 September 2009.

NOTE: Available at: http://www.eurogamer.net/articles/digitalfoundry-lag-factor-article?page=3.

[i.3] OpenGL™ ES: "The Standard for Embedded Accelerated 3D Graphics", Khronos Group.

NOTE: Available at: http://www.khronos.org/opengles/.

[i.4] Jurgelionis et al.: "Platform for Distributed 3D Gaming", International Journal of Computer Games Technology Volume 2009, Article ID 231863.

NOTE: Available at: http://iphome.hhi.de/fechteler/papers/ijcgt2009_JurgelionisFechtelerEisertBellottiDavidLaulajainenCarmichaelPoulopoulosLaikariPeraelaeGloriaBouras.pdf.

[i.5] Philipp Fechteler and Peter Eisert: "Depth Map Enhanced MacroBlock Partitioning for H.264 Video Encoding of Computer Graphics Content", 2009 IEEE International Conference on Image Processing.

NOTE: Available at: http://iphome.hhi.de/eisert/papers/icip09a.pdf.

[i.6] Hugues Hoppe: "Progressive meshes", SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, Pages: 99 - 108, ACM Press/Addison-Wesley Publishing Co., 1996.

NOTE: Available at: http://research.microsoft.com/en-us/um/people/hoppe/pm.pdf.

[i.7] "Progressive forest split compression", SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques, Pages 123-132, ACM Press/Addison-Wesley Publishing Co., 1998.

NOTE: Available at: http://mesh.brown.edu/taubin/pdfs/taubin-etal-sg98.pdf.

[i.8] ISO/IEC 15444-1 (2004): "Information technology - JPEG 2000 image coding system: Core coding system, latest revision 2004".

[i.9] ISO/IEC 14496-1 (2010): "Information technology - Coding of audio-visual objects - Part 1: Systems".

[i.10] ISO/IEC 14496-10 (2009): "Information technology - Coding of audio-visual objects - Part 10: Advanced Video Coding".

[i.11] ISO/IEC 19775-1.2 (2008): "X3D™ Architecture and base components, Edition 2".

NOTE: Available at: http://www.web3d.org/x3d/specifications/X3DPublicSpecifications.zip.

[i.12] WebGL™ Specification, Working Draft (27 May 2010), Khronos Group.

NOTE: Available at: https://cvs.khronos.org/svn/repos/registry/trunk/public/webgl/doc/spec/WebGL-spec.html.

[i.13] Adobe Systems Incorporated: "Adobe Flash™ Player".

NOTE: Available at: http://www.adobe.com/products/flashplayer/.

[i.14] ISO/IEC 14496-11 (2005): "Information technology - Coding of audio-visual objects - Part 11: description and application engine".

[i.15] OpenGL™ Graphics with the X Window System (GLX), Version 1.4, December 2005.

NOTE: Available at: http://www.opengl.org/documentation/specs/.

[i.16] John Stavrakakis et al.: "Exposing Application Graphics to a Dynamic Heterogeneous Network", Proceedings of the 14-th International Conferences in Central Europe on Computer Graphics, Visualization and Computers.

NOTE: Available at: http://wscg.zcu.cz/wscg2006/papers_2006/full/g23-full.pdf.

[i.17] Greg Humphreys et al.: "Chromium: A Stream-Processing Framework for Interactive Rendering on Clusters", SIGGRAPH 2002.

NOTE: Available at: http://chromium.sourceforge.net/.

[i.18] Micah Dowty et al., VMWare™ Inc.: "GPU virtualization on VMware's hosted I/O architecture", USENIX Workshop on I/O Virtualization 2008, ACM SIGOPS Operating Systems Review , Volume 43 Issue 3.

NOTE: Available at: http://www.usenix.org/event/wiov08/tech/full_papers/dowty/dowty.pdf.

[i.19] Gaikai™, Gaikai™ Inc.

NOTE: Available at: http://www.gaikai.com/.

[i.20] T5 Labs Limited.

NOTE: Available at: http://www.t5labs.com/.

[i.21] Onlive™, Onlive™ Inc.

NOTE: Available at: http://www.onlive.com/

[i.22] Tristan Richardson et al.: "Virtual Network Computing (VNC)", IEEE Internet Computing Volume 2, Number 1, 1998.

NOTE: Available at: http://www.cl.cam.ac.uk/research/dtg/attarchive/pub/docs/att/tr.98.1.pdf.

[i.23] XRT Self-contained Remoting Protocol, ANSI/CEA-2014-A: "Web-based Protocol and Framework for Remote User Interface on UPnP™ Networks and the Internet (Web4CE), Annex O".

NOTE: Available at: http://global.ihs.com/search_res.cfm?MID=W097&input_doc_number=CEA-2014.

[i.24] The VirtualGL project.

NOTE: Available at: http://www.virtualgl.org/Main/HomePage.

[i.25] ISO/IEC 14772-1 (1997) and ISO/IEC 14772-2 (2004): "Virtual Reality Modeling Language (VRML)".

NOTE: Available at: http://www.web3d.org/x3d/specifications/vrml/VRML97Am1/.

[i.26] COLLADA™: "Digital Asset Exchange Schema for Interactive 3D", Khronos Group.

NOTE: Available at: http://www.khronos.org/collada/.

[i.27] Dr. Rémi Arnaud (Sony Computer Entertainment) et al.: "Developing Web Applications with COLLADA™ and X3D™ Whitepaper", March 25 2007.

NOTE: Available at: http://www.khronos.org/collada/presentations/Developing_Web_Applications_with_COLLADA_and_X3D.pdf.

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the following terms and definitions apply:

casual games: games that typically do not require long-term time commitment or special skills to start playing, targeted at casual consumers, who comes across the game and can get into gameplay almost immediately

NOTE: Casual gaming does not necessarily mean that it has to be offered for free or that the graphics are primitive.

hardcore gamer: gamer who spends a considerable amount of time and money on games, and who prioritises gaming over other leisure activities

input lag: time between a user action and its response on the screen

3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply:

AVC	Advanced Video Coding
BIFS	Binary Format for Scenes
COLLADA™	COLLAborative Design Activity
GLX	OpenGL™ Extension to the X Window System
GPU	Graphics Processing Unit
MCD	Media Content Distribution
MMORPG	Massive Multiplayer Online Role-Playing Game
OpenGL™ ES	OpenGL™ for Embedded Systems
OpenGL™	Open Graphics Library
RTP	Realtime Transfer Protocol
SDK	Software Development Kit
STB	SetTop Box
UDP	User Datagram Protocol
VNC	Virtual Network Computing
VoD	Video on Demand
VRML	Virtual Reality Modeling Language
X3D™	eXtensible 3D Graphics

4 Use case(s) and high level requirements

4.1 Primary use case

The primary use case is to offer the ability for a user to play 3D games while part of the game resources (such as the game graphics, game logic, etc.), are not located on the end user's device but are hosted on a server in the network. The game's 3D graphics (possibly in conjunction with some game logic) are delivered to the end user's device in a streaming fashion. This allows an easy and flexible way to access games on a variety of different (thin-client) devices. An example of such a scenario could be a person on the couch watching TV, that whilst waiting for his favourite show to start, wants to (spontaneously) play a game to pass the time without having to insert a game DVD or wait for a game download to finish.

The server device can be located anywhere in a network, for example server in a data center ("cloud server") or a PC at home. We expect the games to be written for commonly used 3D rendering APIs such as DirectX™ or OpenGL™.

The resulting system's architecture is typically one of the two following ones:

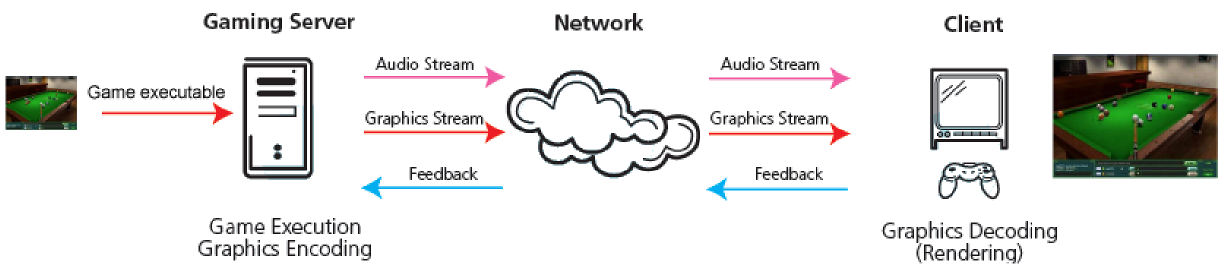

Figure 1: Graphics streaming architecture with game logic running on server

In this case, a gaming server loads the game executable (and perhaps some related metadata), and starts running the game on the server. The graphical output and the audio are both encoded into a streaming format, after which the encoded content is streamed to the client. The client receives the graphics and audio stream content, decodes and renders the output. The user input (e.g. from a game controller) is sent back to the server and taken into account by the game for subsequent rendering. Since the client device does not need to resources to run the game, it enables devices with limited resources, such as TVs, STBs, Netbook PCs etc. to have access to compelling 3D games that would otherwise not be able to run.

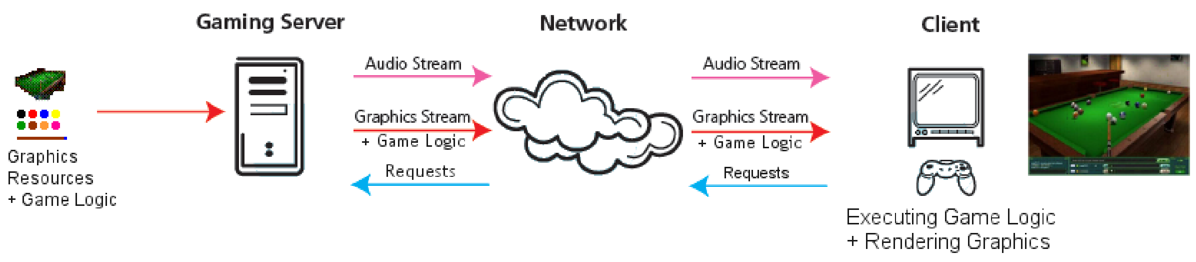
Gaming Server
Network
Client
Graphics Resources + Game Logic
Audio Stream
Graphics Stream + Game Logic
Requests
Audio Stream
Graphics Stream + Game Logic
Requests
Executing Game Logic + Rendering Graphics

Figure 2: Graphics streaming architecture with (parts of) game logic running on client

In this second architecture, (parts of) the game logic is delivered to (or already present on) the client and executed on the client device while graphics and audio data are hosted on the server and streamed to the client on request.

4.2 High level requirements

The main requirement for a streaming and distributed processing system for games is to address the concerns of game play. As any other gaming platform, streaming games from a server faces strong constraints against which an appropriate solution is required so that gamers do not simply run away from the platform because their experience is not up to their expectations. The four main constraints for the solution to take into account are: input lag, image quality, scalability of the service and availability of the service. See clauses 4.2.1 through 4.2.4 for more details. The list below also contains a couple of other important high level requirements in clause 4.2.5 onwards.

NOTE: As mentioned in the Scope (clause 1), the primary focus is on the requirements for the 3D gaming graphics delivery protocol. Deployment issues such as user registration, content selection and purchase are outside the scope of the present document. Also, inter-destination synchronization of multi-user games graphics is out of scope of the present document.

4.2.1 Minimize input lag

Input lag, i.e. the time between a user action and its response on the screen, is probably concern #1 for many of the gamers.

Input lag can be broken down into the following subcomponents:

- input detection time (usually negligible, if we assume the latest wired USB and wireless controllers)
- input information transport time (if information is not handled locally)
- input information management in the game or game engine
- game rendering after input management (actual creation of the output frame)
- transport time of the game rendering data (if not displayed locally)
- display rendering (actual display of the video frame on screen)

Network phenomena, such as latency, jitter, lack of bandwidth or packet loss can introduce additional lag. The solution needs to take these networking aspects into account and minimize their effects on the gameplay.

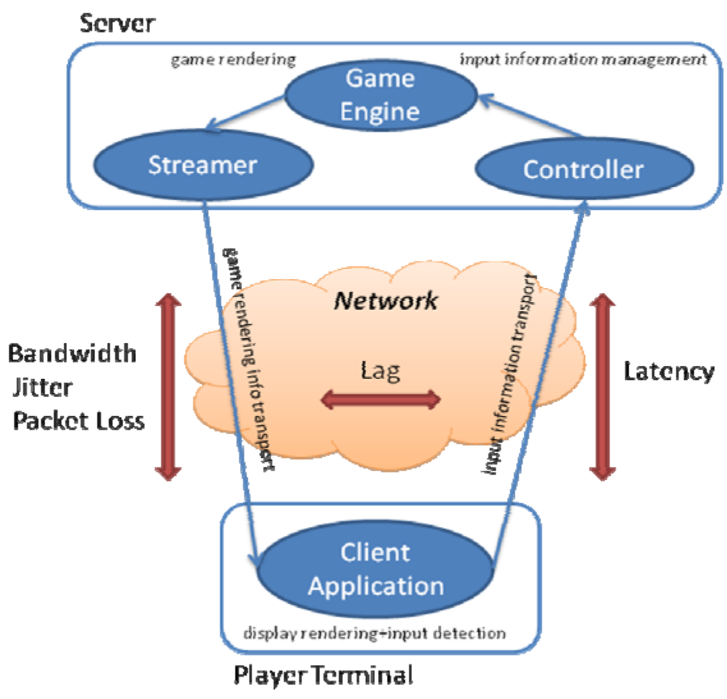

Figure 3: Graphical illustration of input lag

The effects of input lag on the user depend on the user's own sensitivity to it. A hardcore gamer would certainly not put up with a minimum amount of lag while an occasional player would probably not even notice it.

In the application of gaming, the type of game being played is another important factor that may increase one's perception of lag as players' expectations on game play may considerably change depending on the type of the game.

In the present document, we will distinguish:

- Slow paced games (such as chess games and typical MMORPGs such as World of Warcraft™): in slow paced games, input lag is usually not a problem. Slight delays (up to 500 ms) are usually tolerable (see section 2 of [i.1] for more information).
- Medium paced games (such as platform games like Super Mario™ 64, and sports games): in medium paced games, delays higher than 150 ms or 200 ms would probably be noticeable by gamers (see section 2 of [i.1] for more information).
- High paced games (such as fast-action shooters like Unreal Tournament™ or Crysis™, or fighting games): in high paced games, the reaction time requirement for the user is usually very strict and typically any lag over 100 ms affects the gaming experience (see section 2 of [i.1] for more information).

NOTE 1: Current console games typically have an average lag of about 133 ms (see [i.2]).

NOTE 2: There may not be one unique solution that can fit the constraints imposed by any type of games streamed on any kind of network with the current state of technology. As a consequence, in the present document, we may envisage to recommend a different approach in function of game streaming context and its effects on user's quality of experience.

4.2.2 Maximize image quality and framerate

For some gamers, the image quality of the video game is very important. This is especially true for PC gamers who have the capability to play their video games with different quality settings. In order to obtain the highest possible quality; some would regularly update their hardware or even sacrifice framerate (and consequently input lag as well). However, others prefer a higher framerate over highest possible image quality.

It is perceived that under a certain level of image quality, the gaming experience becomes highly impacted. Nevertheless, Nintendo™ proved with the Wii™ that image quality is not necessarily #1 criterion for all gamers. For framerates below a certain level, the gaming experience also becomes highly impacted.

So on the one hand we need to maximize image quality and framerate, on the other hand, we should make the solution flexible enough to allow selection of different quality levels (e.g. level of details rendered by the client).

4.2.3 Allow Scalability

Being able to play a game with the same experience whether there is just one player or thousands of players playing together is also an important requirement for gamers. For games running on a server, this is a major issue since many players are sharing the same server even if playing a mono-player video game. While players may understand that the gaming experience in multi-player games may decrease while the number of players increase, it may be very difficult to have them understand that the same thing happens with mono-player games too.

Scalability is not only important for gamers, but also for the service providers that distribute the game, since the scalability of the solution directly affects the cost involved to run the game. The more players that can be served by a single server, the lower the cost to deploy the game. This affects the price range at which streaming of games can be offered.

The solutions studied in the present document should take into account the eventuality of adapting quality depending on the server's resources.

4.2.4 Maximize availability

Availability is not an issue players are usually familiar with, since on their PC or game consoles they can pretty much play their video games whenever they want. But with games running on a server in the network, the situation is different and it is obviously important that the service is always available to users. The solution needs to be robust in dealing with certain network aspects (i.e. such as jitter and packet loss). If the connection to the server is lost, the solution should be able to detect if the network and/or the server are not available anymore, and should make the user aware of this.

4.2.5 Distribution of game processing resources

Transferring most of the game processing from the local client to a server, eventually shared by several gamers, is the core of this subject. All proposed solutions need to obey this distribution of computation resources between client and server and the associated handling of input events. The distribution of resources needs to be done in such a way as to minimize the effects on game play (i.e. as defined by clauses 4.2.1 through 4.2.4).

4.2.6 Reuse existing infrastructure

It is desirable that the solution(s) can run on existing network and server infrastructures, without requiring specific adaptations. This reduces the cost of deployment.

In order to bootstrap the problem of the availability of content, it is also important that the solution allows reuse of the large catalogue of existing 3D games based on DirectX™ or OpenGL™, with little or no adaptation needed to the source code of the game.

4.2.7 Support for a variety of devices

The solution should allow for a multitude of different devices to be used as clients, and should not depend for example on a particular operating system or single screen resolution.

The solution should allow at least have the ability to use keyboard, mouse and standard game controllers, or a combination thereof as user input. The solution should preferably be extensible to other types of user input devices for games.

4.2.8 Authentication and security

The solution should support some form of client authentication to make sure that only authorized clients can start/enter a game session provided by the server.

5 Solution approaches for delivery of 3D output of games

The following clauses describe different approaches for the delivery of 3D graphics output of games to thin-client devices.

5.1 Streaming 3D graphics primitives

5.1.1 Main concept

The main concept behind this solution approach is the following:

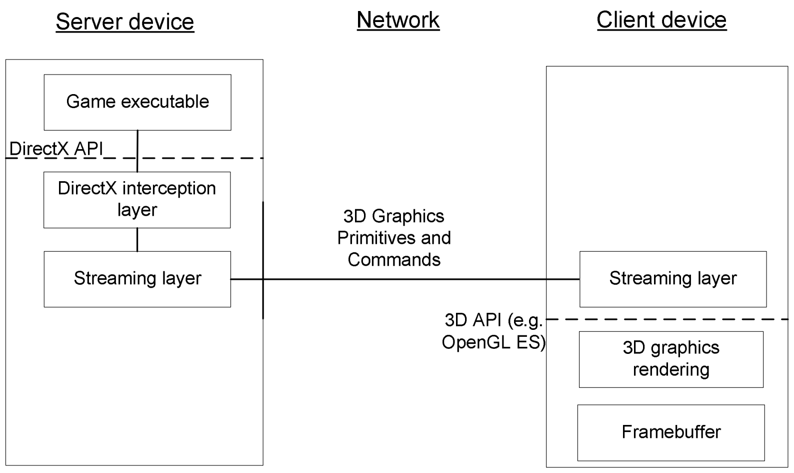

Figure 4: Streaming 3D graphics primitives

The server device executes the game executable. Instead of sending the DirectX™ commands to the graphics card (i.e. the Graphics Processing Unit (GPU)), a software layer intercepts the DirectX™ commands. A streaming layer than analyzes and processes these commands to make them suitable to be streamed over the network to the client device using a protocol suitable for sending and encoding these 3D graphics primitives and commands.

The client device then receives the stream of 3D graphics primitives and commands and forwards them to the 3D graphics API, for example using OpenGL™ for Embedded Systems (OpenGL™ ES) [i.3]. The client device then renders the 3D graphics output into its framebuffer, which is shown to the user.

This solution approach assumes the presence of some hardware acceleration functions to enable 3D rendering by the client device. With the uptake of 3D hardware acceleration for mobile phones and other embedded devices (e.g. using OpenGL™ ES), such support is likely to become commonplace.

See [i.4] for more information about this approach.

5.1.2 Benefits/Drawbacks of this approach

Benefits:

- The server does not need to render the actual graphics of the game. This makes it a very scalable solution. A single server can serve many clients. It also means that common data center technology available today can be used for streaming games. It does not need GPU support, which is typically available only in specialized data centers. This is important to keep the deployment costs down.

- Since it does not need to render the graphics on the server, the latency is usually lower than for example using a video streaming solution.

- Since it is primarily the client who is in control of the output to be rendered, the graphical output remains sharp whilst allowing for scaling to different resolutions.

Drawbacks:

- The graphical detail and/or resolution may need to be adjusted to suit the hardware capabilities of the client device's 3D rendering functionality.
- The bitrate is typically not constant. Complex scenes require more bandwidth than simple scenes.

5.2 Streaming Video

5.2.1 Main concept

The main concept behind this solution approach is the following:

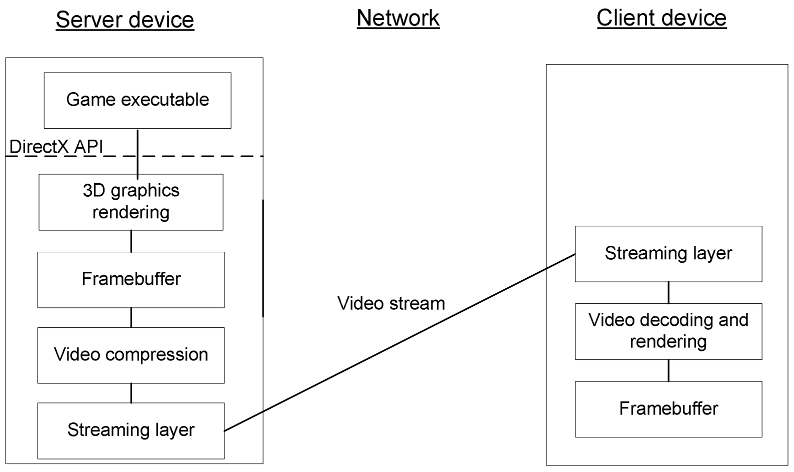

Figure 5: Video streaming

The server device executes the game executable. It renders the 3D graphics of the game (using its GPU) into a framebuffer. The graphics of this framebuffer are captured and compressed into a video stream, which is sent to the client, using a (standard) video streaming protocol, such as H.264/MPEG-4 AVC [i.10] over RTP/UDP.

The client device receives the video stream, decodes it and renders it into its framebuffer, which is shown to the user.

Note that because of the low latency demands of streaming 3D games, the video streaming solution needs to be implemented with low-latency encoding/decoding an a minimum of buffering on both client and server end:

- For the server it means that it cannot perform its compression based on interpolating multiple input frames, such as is done e.g. for compressing B-frames, multi-pass encoding, multi-frame encoding. Basically, the frame has to be compressed and sent to the client as soon as it has been rendered in the server's framebuffer.
- For the client it means a video frame has to be decoded and displayed as soon as it is received. This is different to Video on Demand (VoD) streaming where typically large buffering is used to remove the effects of network jitters. It more resembles solutions that are used for video conferencing, where low-latency encoding and decoding is an important aspect.

Note that standard H.264/MPEG-4 AVC [i.9] is flexible enough to cope with these low latency requirements. See [i.5] and also [i.4] for more information.

5.2.2 Benefits/Drawbacks of this approach

Benefits:

- No dependency on 3D hardware acceleration on the client device. The graphical details and resolution solely depends on the server's rendering capabilities (and server resources available).
- Bitrate can remain quite constant.

Drawbacks:

- Since the server needs to render the actual graphics of the game, the solution is very heavy for server devices and is therefore less scalable. The solution depends on 3D GPU hardware acceleration and preferably also video encoding hardware acceleration, both of which are typically only available in specialized data centers. This is likely to increase the deployment cost.
- Since it needs to render the graphics and perform video compression, the latency is typically higher than with a 3D graphics streaming solution.
- During the video rendering encoding/decoding process some sharpness of the graphics may be lost.

5.3 Transport of 3D models

5.3.1 Main concept

The main concept behind this solution approach is the following:

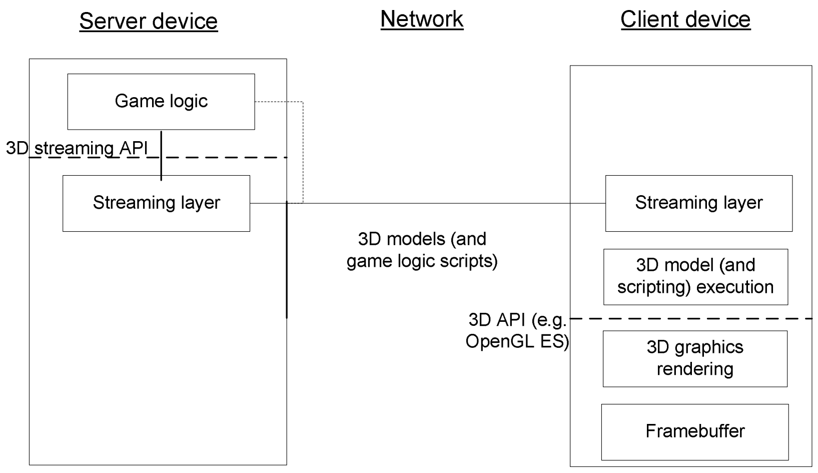

Figure 6: Transport of 3D Models

The server device executes the game executable. Instead of calling the low-level DirectX™ API, it calls a higher level 3D streaming API, which allows the author of the game to send a description of (parts) of the 3D models that constitute the 3D scene to the client. This solution could be extended by allowing the author to send scripts along with the 3D models for execution by the client. This could then handle some of the game logic, e.g. to change the 3D model's view or the viewing angle into the scene, on the client without needing a roundtrip to the server.

The client receives the description of the 3D models and renders them. Depending on the solution, it may also need to execute scripts that allow the user to interact with the 3D models without needing a roundtrip to the server.

This solution is typically used for 3D games or 3D applications for deployment in web browsers. The game publisher needs to develop its game specifically with this deployment model in mind, since it relies on a different API than e.g. DirectX™. Various solutions are being developed in this area that allow for this new breed of web applications to be created, such as X3D™ (successor of VRML) [i.11], WebGL™ [i.12], and Adobe Flash™ [i.13]. Also extensions to MPEG were developed to allow for this kind of scenarios, such as MPEG-4 Part 11 Binary Format for Scenes (BIFS) [i.14].

5.3.2 Benefits/Drawbacks of this approach

Benefits:

- The server does not need to render the actual graphics of the game. This makes it a very scalable solution. A single server can serve many clients. It also means that common data center technology available today can be used for the deployment of games. It does not need GPU support, which is typically available only in specialized data centers. This is important to keep the deployment costs down.
- Since it does not need to render the graphics on the server, the latency is usually lower than for example using a video streaming solution.
- Since it allows some of the 3D model manipulations to be performed by the client without needing a roundtrip to the server, which can be supplemented with scripts provided by the server, the latency can be further reduced.
- Since it is primarily the client who is in control of the output to be rendered, the graphical output remains sharp whilst allowing for scaling to different resolutions.
- Low bandwidth requirements.

Drawbacks:

- The game needs to be specifically created or redesigned to fit this deployment scenario. This makes it very hard to deploy existing games.
- 3D model manipulation and in particular execution of scripts using a scripting environment demands resources on the client. A scripting solution is typically a lot slower than a high performance game executable written in efficient C++ code.
- The graphical detail and/or resolution may need to be adjusted to suit the hardware capabilities of the client device's 3D rendering functionality.

5.4 Progressive transport of adaptive 3D models

5.4.1 Main concept

The main concept behind this solution approach is the following:

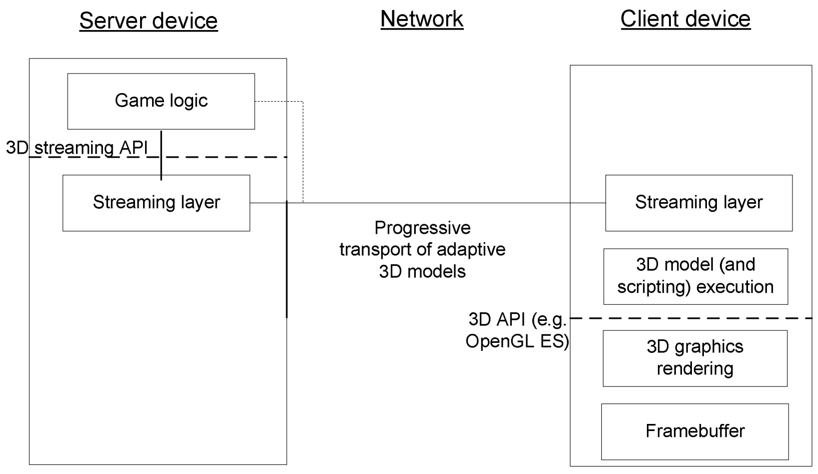

Figure 7: Progressive transport of adaptive 3D models

This is a variant of the model defined in clause 5.3 to optimize the transport and rendering of 3D models. In this case the 3D model descriptions are **progressive** and **adaptive**. The idea is that objects and textures of the scene are sent with a very low level of detail. So their representation is much lighter than the equivalent OpenGL™ primitives. They can be transported and rendered very quickly, and hence may improve the framerate, reduce initial startup time, and reduce memory and bandwidth usage. Of course, such a low level of detail may not be satisfying. For each object of the scene, a set of very small rules can, then, be sent progressively which allow the client to re-construct higher levels of details up to the original one. Rules are fast to download and are requested on demand, depending on the need of details of the renderer (objects far in the scene do not need a high level of detail to be correctly rendered) and/or on the abilities of the client device (in terms of CPU, memory, GPU, bandwidth). Hence, the format is **progressive**.

Independently from the downloading of rules, for each frame, the renderer computes the most relevant level of details for each object and texture depending on a large set of criteria. These could be the position and/or the speed of this object in the frame of reference, the abilities of the client device, or a level of interest of the object according to the game logic. Then rules are applied on each object in order to refine or to degrade it as requested by the renderer. Hence, the format is **adaptive**.

Note that there are several ways in which (part of) the relevant game logic/rules could be deployed:

a) The game logic/rules can be preloaded on the device (e.g. by loading it from a CD or by downloading it in advance of execution). In this case it will run "natively" on the machine. This program is responsible for requesting the root part of the scene graph from the server.

b) The game logic is not pre-loaded (i.e. it is scripted). In this case the client is requested to download the root part of the Scene Graph that contains the scripts. These scripts are then transferred from the server and executed on the client.

Up to now, this solution is typically used for 3D games or 3D applications for deployment in web browsers (e.g. Google™ Earth). In order to allow the 3D model descriptions to be progressive and adaptive, the game publisher typically needs to take these aspects into account whilst developing its 3D game, by using dedicated SDK features.

This kind of solution may be applied as an extension to 3D description formats such as X3D™ [i.11] or 3D APIs such as OpenGL™ or Direct3D™. Direct3D™ implements the "Progressive Meshes" technology as described in [i.6] [i.7]. Also the extensions to MPEG, such as MPEG-4 Part 11 Binary Format for Scenes (BIFS) [i.14] could be used as a basis for progressive transport of adaptive 3D models. Some progressive and adaptive textures formats are already standardised, such as JPEG2000 [i.8].

5.4.2 Benefits/Drawbacks of this approach

Benefits:

- Same as for "Transport of 3D Models" defined in clause 5.3. In addition:
  - It reduces initial startup time, and reduces memory and bandwidth usage, since never too much data is sent across.
  - It allows easy optimization of the framerate.
  - The graphical detail and/or resolution can be automatically adjusted to suit the hardware capabilities of the client device's 3D rendering functionality.

Drawbacks:

- The game typically needs to be redesigned or created specifically to fit this deployment scenario. This makes it very hard to deploy existing games.
- Since the client device has some automatically adjustable control over the level of graphical details (depending on various factors, such as the graphical capabilities of the client), the quality of the graphical output may not be predictable to content authors.
- Dynamic 3D model manipulation and in particular if this is done using a scripting environment demands resources on the client.

6 Overview of existing standards

Analyze existing standards that may have overlap and/or may be applicable to parts of the solution.

Streaming 3D graphics:

This solution needs an efficient encoding format for 3D graphics primitives. Here are some standardized and non-standardized solutions that may be relevant in this area:

1) **GLX [**i.15**]** - GLX is an acronym for "OpenGL™ Extension to the X Window System". This protocol provides the transport and remote rendering of 3D OpenGL™ graphics commands. It also includes the binding to specific parts of the X Window System to control the remote display configuration. The protocol (although not actively developed anymore) is specified by the Khronos industry consortium and falls under the realm of OpenGL™ protocols/APIs developed by that consortium. Despite that it has several dependencies on the X Window System typically only supported by Unix/Linux clients, and the solution should not be limited to devices having support for the X Window System, there may be elements that can be reused, in particular the binary representation of OpenGL™ commands. Note that GLX does not define compression of the data stream. Compression of the GLX datastream may need to be defined, in order to reduce the network load. GLX itself does not describe the backchannel for user input (such as keyboard and mouse events). It does depend on the X Window System protocol to support these events.

2) During the years, several non-standardized and commercial solutions, such as Lumino [i.16], Chromium [i.17], and VMWare™ [i.18], have been implemented that provide some alternative solutions to GLX for streaming OpenGL™ commands or that extend GLX with additional features such as session suspend/resume mechanisms and GPU virtualization for cloud computing environments. The documentation of these technologies shows several techniques on how to improve GLX that may be interesting to take into account whilst developing a protocol for streaming of 3D gaming graphics.

Streaming video:

This solution needs low latency encoding/decoding of video (similar to video chat). Here are some standardized and non-standardized streaming video formats that may be relevant in this area:

1) **MPEG-4** [i.9] - MPEG-4 is an ISO/IEC standard developed by MPEG (Moving Picture Experts Group). MPEG-4 is a framework of technologies for compression of audio/video content and the delivery of multimedia. When considering video streaming of games, MPEG-4 Part 10 Advanced Video Coding (AVC) (also known as H.264) [i.10], is especially worth attention as this codec achieves very high data compression and it is also designed with network streaming in mind. However, encoding A/V content in H.264/MPEG-4 AVC is typically slow, and steps need to be taken in the implementation to achieve low-latency encoding. Several ways exist to achieve low latency encoding of games. One of the ways is to make use of the game's 3D depth map and projection matrices to allow for efficient calculation of the motion vectors in a scene, as described in [i.5]. It makes sense to combine the low latency encoding with a transport protocol such as RTP over UDP to stream the video, and as little buffers on the client and server as possible, in order to speed up the transport and rendering. Several companies are looking into this type of streaming, such as Gaikai™ [i.19], T5 Labs [i.20] and Onlive™ [i.21]. Note that the latter has developed its own proprietary video codec, whereas the first two are looking into using standard MPEG-4 encoding to deliver 3D games.

2) A solution that is similar to MPEG-4 video streaming and that is supported as a mode by many standardized and non-standardized Remote User Interface protocols, such as VNC [i.22] and XRT [i.23] is by making use of Motion JPEG. Motion JPEG (also known as M-JPEG) is an informal name for a class of video formats where each video frame of a digital video sequence is separately compressed as a JPEG image. Whereas traditional remote framebuffer based remote UI solutions such as VNC are not suitable for 3D gaming (because they are optimized for sending only those parts of the screen that have been updated), several variants of these protocols, such as VirtualGL/TurboVNC [i.24], have been developed that can compress every 3D frame as a JPEG image by using a highly optimized JPEG codec and send the generated sequence of JPEG images to the client. This enables remoting of 3D games in a similar fashion as solutions based on MPEG-4 video streaming.

Transport of 3D models:

This solution needs a generic format for describing 3D models (and its interaction). Here are some standardized and non-standardized streaming video formats that may be relevant in this area:

1) **X3D™ [**i.11**]** - X3D™ is an ISO Standard, maintained by the Web3D Consortium, targeted at encoding of interactive 3D content, intended to be used in web or broadcast-based environment. X3D™ is the successor of the well known language VRML [i.25]. X3D™ is an XML based specification for describing and communicating 3D scenes and objects. It can also include runtime behavioural descriptions (using scripting). It operates on higher level of abstraction than 3D APIs such as OpenGL™ and DirectX™, which allows flexibility in the transmission and execution. Because of its higher level of abstraction it is also more concise, which can contribute to lower bitrates for transmission. In addition, the standard also defines a mapping to a compact binary encoding of the XML data in order to further reduce the network load. Unfortunately, the higher level of abstraction also means that it would be difficult to use it for the delivery of existing DirectX™ or OpenGL™ based games, without needing to rewrite the source code of those games. However, for new web-based 3D applications that do not depend on these APIs, it is quite a powerful format.

2) **MPEG-4 BIFS** [i.14] - MPEG-4 BIFS is a scene description and application engine that has been developed by the MPEG consortium as part of the MPEG-4 framework of standards and technologies [i.9]. BIFS or Binary Format for Scenes is a binary format for 2D and 3D multimedia content is defined in MPEG-4 Part 11 [i.14]. The scenes are described in language constructs inherited from VRML [i.25].However, its bitstream representation is completely different, and MPEG-4 BIFS adds several distinguishing features on top of VRML, in particular data streaming which means that the client can receive the data incrementally and renders it as needed. In basic VRML the scene would first have to be completely transferred from the server to the client before rendering, leading to long latency. MPEG-4 defines a scene update mechanism using BIFS commands as a way to allow scene components to be animated. MPEG-4 BIFS is designed to be used in broadcast and video streaming based applications. Therefore, in MPEG-4 BIFS the 3D application can be seen as a temporal stream. It contains various mechanisms to allow the 3D VRML content to be synchronized with playback of audio video content.

3) **COLLADA™** [i.26] - COLLADA™ is an interchange file format for interactive 3D applications. It is a standard created by the Khronos group and was designed as an intermediate format aimed at gaming industry and intended for bringing together content and assets from different editing tools. Several game publisher use this intermediate format whilst developing their games. The COLLADA™ format is typically used during the process of transformation the game assets as they journey from content creation tools that use higher level description paradigms to platform-specific optimized API descriptions. Although it shares many commonalities with X3D™, it was not designed to describe detailed lower level run-time behaviour and for efficient delivery of the network, some aspects which are addressed by X3D™ (see [i.27] for more information), so it is not the most obvious choice for delivery of 3D gaming graphics, although the direct link with content authoring tools is definitely an advantage.

4) **WebGL™** [i.12] - WebGL™ is a standard created by the Khronos group aimed at bringing 3D graphics to the Web by introducing a Javascript API derived from OpenGL™ ES 2.0 that can be used in HTML5 canvas elements. Unlike X3D™, it operates at a much lower abstraction level. This means application development can stay much closer to well known OpenGL™ API constructs. However, it does make extensive use of Javascript, which does come with a performance penalty compared to native C++ code. Because typically a big part of the 3D application scripting sources get transferred to the client before executing the application inside the web browser, WebGL™ shares many of the characteristics as the more pure model based approaches such as X3D™ and COLLADA™. A list of these characteristics can be found in clause 5.3 of the present document.

Progressive transport of adaptive 3D models:

There are currently no known standardized solutions in this area, or standards that clearly make use of a progressive model for transporting adaptive 3D models.

However, there are some standards such as JPEG2000 [i.8] and MPEG-4 BIFS [i.14] that could be used as a basis for progressive transport of adaptive 3D models. The ability of adapting the 3D content based on criteria such as bandwidth, rendering speed, etc. is a powerful concept, and hence may be useful to take into account whilst designing any network based solution of delivery of 3D graphics.

7 Conclusions and recommendations

Various technical solution approaches can be considered for the delivery of 3D gaming graphics from a client to a server. These approaches range from streaming 3D graphics primitives, video of the rendered graphics output, to the (progressive) transport of 3D models. Each of these solution approaches has some possibilities of delivering 3D gaming graphics to thin clients, however, each of the solutions comes with its own set of advantages and disadvantages, as described in clause 5 of the present document.

A solution based on streaming 3D graphics primitives would seem to be the best solution to fit the high-level requirements as described in clause 4, with the main advantage being the ability to reuse existing infrastructure and the scalability of the server. Since the 3D rendering capabilities of the embedded devices are becoming commonplace and more powerful, the disadvantage that the solution depends on the 3D capabilities of the client seems to become less of a problem.

A video streaming solution does not have this dependency on the availability of 3D rendering hardware support on the client, the main disadvantage of the video streaming solution is that it is very heavy on the server, because of the rendering and video compression. This makes the solution more costly to deploy, less scalable, and therefore less attractive. The additional steps (as graphically depicted in clause 5.2) are likely to increase the latency of the solution. However, as the costs for server based graphics solutions will decrease, there will be deployments in the market that make use of a video streaming solution.

Model based solutions are often used for adding 3D content to websites and for a variety of 3D applications, including 3D games (locally run). However, they are not a generic solution for delivery of games. For example, it would not be possible to reuse existing games created for DirectX™ execution environments, without rewriting of the game's source code. No commercial study has been done to estimate whether rewriting of part of the code of a game has to be considered as a major drawback or not, as happened for example in the past to support new devices/platforms (such as smartphones, game consoles). For model based solutions based on scripting environments, and other high-level technologies such as XML, a possible performance penalty needs to be taken into account when porting games to such environment.

The solution of using progressive transport of adaptive models was also described in the present document. There are no clearly defined standards in this area, and the general idea and ability of adapting the 3D content based on criteria such as bandwidth, rendering speed, etc. is a very useful idea to take into account in designing any network based solution of delivery of 3D graphics, and may offer some technical advantage to game developers. Therefore, this technology should not be ruled out of the proposition.

The overview of technologies and standards in clause 6 shows that there is currently not really a standardized solution for streaming 3D graphics primitives of games. In particular a solution that is:

a) geared towards OpenGL™ ES, which is designed for embedded devices such as cellphones and TVs,

b) does not depend on X Window System,

c) for which a clear backchannel is defined for user input devices such as game controllers,

d) which is optimized for low latency.

For the other solution approaches, such as video streaming, there seems to be a less obvious gap in standardization. Analysis shows that it would be possible to use existing standards, such as video streaming based on H.264/MPEG-4 AVC in conjunction with a low latency encoder, to achieve the ability to stream video games. The model based/scripting based approaches typically target a different use case, and various standards have already been devised in that area, such as X3D™ of the Web3D consortium, and WebGL™ and COLLADA™ of the Khronos group. None of these standards currently address progressive transport of adaptive models. This is something however that would be interesting to consider for any solution dealing with the delivery of 3D gaming graphics.

Given the focus of games running on high-performance servers and the current gaps in standards for 3D graphics streaming, the recommendation of the present document is to define and standardize a protocol for streaming of 3D games, based on streaming OpenGL™ ES graphics primitives that is independent of X Window System, and that takes into account the high-level requirements as described in the present document. This recommendation should not preclude that ETSI may work later on complementary solutions such as model based/scripting based approaches.

History

Document history		
V1.1.1	December 2010	Publication