Activity-Centric Prioritized Streaming of Games to Mobile Devices

HESAM ALDIN RAHIMI KOOPAYI

A thesis submitted to the Faculty of Graduate and Postdoctoral Studies
in partial fulfillment of the requirements for the degree of

MASTER OF APPLIED SCIENCE
IN ELECTRICAL AND COMPUTER ENGINEERING

Ottawa-Carleton Institute for Electrical and Computer Engineering
School of Electrical Engineering and Computer Science
University of Ottawa
Ottawa, Canada

Abstract

As mobile devices still have limited battery life, processing power, memory, and display size, they cannot yet execute gaming applications with the same fidelity and quality as their PC counterparts. In response, researchers have recently performed research with the goal of the real-time delivery of game content specifically to fit within mobile devices' limitations. In this thesis, we present a novel approach to tackling the streaming of objects to mobile devices. Our goal is to reduce the number of objects subject to streaming from the server to the target devices, while not violating the user-defined limitations through an efficient, context-aware 3D object selection and prioritization scheme. We take advantage of the game context to stream only the most relevant objects. Our evaluations have shown that this technique not only leads to better performance in general, but also increases the gameplay experience by helping the player to achieve a higher score.

Acknowledgements

This dissertation would not be possible without the many people who contributed to it at various stages.

First and foremost, the main reason for my entrance into a master's program in Canada, aside from my personal desire to continue my education abroad, was my honorable supervisor, Professor Shervin Shirmohammadi, who accepted me as his student. The first stroke of luck came when I met Prof. Shirmohammadi while he was on sabbatical in my home country, after which he encouraged me to apply at the University of Ottawa and later accepted me as his student.

I'm grateful to Prof. Shirmohammadi for the opportunity he gave me in Canada and for giving me space to explore and make mistakes. Without his tolerance, patience, acceptance, encouragement, and support, I would not be graduating. He morally and financially supported my studies in Canada, and helped me expand and develop myself in many aspects of life, academic as well as personal. He is the model of a true scholar, and I hope that I can imitate his kindness. It has been a pleasure and a privilege to learn from and work with him, and I am deeply honoured to have him as my supervisor and mentor.

I'm also particularly thankful to Dr. Ali Nazari. He demonstrated not only how to perform superior research, but how to act as a good person. Dr. Nazari's generosity, support, and understanding helped me through various difficult stages in this work. His time spent

helping me with this thesis is greatly appreciated. Without his dedicated efforts, all of this work simply would not have happened. I enjoyed so many helpful discussions with him, which are all now wonderful memories.

I must also acknowledge all of my friends who helped me during my difficult situations in Ottawa. Also my special thanks are for another friend in my home country who helped me in implementation part of this thesis.

Dearest of all are my family, my parents, my older brother, and my younger sister. They are my greatest source of strength, with the endless love and support they have given me in all my endeavors. Being far from the family was hard for all of us, but my mother and my sister's love was the best encouragement for me in coping with all the difficulties over the past two years. I dedicate this work to my family.

Hesam Rahimi

Ottawa, November 2011

Table of Contents

Table of Figures

Table of Tables

List of Acronyms and Definitions

VE	Virtual Environment
AOI	Area of Interest
DVE	Distributed Virtual Environment
FPS	First Person Shooter
IP	Internet Protocol
LOD	Level-of-Detail
CSG	Constructive Solid Geometry
MMOG	Massively Multiplayer Online Game
PM	Progressive Meshes
P2P	Peer-to-Peer
TCP	Transport Control Protocol
UDP	User Datagram Protocol

Chapter 1 - Introduction

There is no doubt that in the near future, mobile computing will replace PC-based computing. Reuters [1] has reported that more than half of the world's population already uses cell phones or other mobile devices; hence, people are more likely to use a handheld device than a PC to browse the internet [2]. In recent years, mobile devices have integrated video cameras, fast or even dual core CPUs, very large memories, 3D hardware accelerators, WiFi, etc. In addition to this, the rapid advancement of wireless networking technologies has led to the introduction of the fourth generation of cellular wireless standards (4G), which is a successor of 3G and the 3G families of standards. Therefore, facilities such as ultra-broadband Internet access, IP telephony, gaming services, and streamed multimedia may be provided to mobile users [3]. At the same time, we are witnessing a move towards the mass consumption of 3D media, with a significant rise in applications such as 3D video, 3D virtual environments, and 3D games [4].

The idea of fully displaying and navigating websites using 3D was introduced and called Web3D, which now refers to all interactive 3D contents that are embedded into webpages' html and that can be seen through a web browser [5]. Currently, many formats and tools are available for browsers to incorporate 3D display and processing capabilities. X3D,

which is the successor to Virtual Reality Modeling Language (VRML), O3D, Java 3D, and WebGL, is among the many available Web3D technologies.

Of the different types of interactive 3D contents on the web, one of the most prevalent recent trends has been that of online video games and online 3D virtual environments. Video games, which are among the most profitable economic sectors and subject to both academic and industry research, have become an important part of the 21st century's entertainment landscape and are growing in popularity at a substantial rate. Currently, multiplayer networked games are not only for entertainment. They can be employed for socializing, business, commerce, scientific experimentation, and many other practical purposes. This is due to the fact that they make it possible for hundreds of thousands of players to simultaneously interact with one another.

Studies have been done in North America (Canada and the US) regarding the growth of the gaming industry, especially online games. As reported by [6] , in 2011, 59% of Canadians were gamers, a gamer being a person who has played computer or video games in the past four weeks. Another study by [6] shows that in 2011, 47% of Canadians households had at least one video game console, such as Xbox 360, Wii, or PlayStation3, and that 30% of gamers play every day. Moreover, a study shows that on October 30, 2011, Canada boasted the third largest video game industry in the world [7]. Is has been reported that the industry is poised to grow 17% over the next 2 years (2011 – 2013). In Appendix A, more reports have been shown regarding the Canadian gaming industry.

In the US, as reported by the NDP Group, 59% of the total US population (ages two years old and older) play games, with 56% of them doing so online [8]. Therefore, almost one third of the population is playing online games. On January 28, 2009, comScore Inc. [9], a leader in measuring the digital world, released an analysis of Americans' usage of online gaming sites, which showed that this category has grown 27% during the past year to 86 million visitors in December of 2008. The total time spent playing online games has jumped 42%.

Mobile gaming is growing too, and it is estimated that 78.6 million people in the U.S. alone played mobile games in 2009. Downloads of mobile games increased tenfold as compared to 2003 [8]. Not only is mobile gaming growing, but mobile social media usage is increasing as well. On October 20, 2011, comScore Inc. released the results of a study on mobile social media usage based on data from its comScore MobiLens service, which showed that 72.2 million Americans accessed social networking sites or blogs on their mobile device in August of 2011, an increase of 37% in the past year [10]. "Social media is one of the most popular and fastest growing mobile activities, reaching nearly one third of all U.S. mobile users," said Mark Donovan, comScore senior vice president for mobile. "This behaviour is even more prevalent among smartphone owners, with three in five accessing social media each month, highlighting the importance of apps and the enhanced functionality of smartphones to social media usage on mobile devices" [10].

These studies strongly show that millions of people spend their time and money on online games and in online virtual/social environments, especially using their handheld devices.

Therefore, these advancements led researchers to consider mobile online gaming as an avenue for research to further manage and support this emerging massive industry as well as its traffic on the network and user's quality of experience.

1.1 Motivation

3D virtual environments and games are becoming more realistic and detailed, with higher resolution and near-reality texture maps. To achieve this high level of fidelity, a considerable amount of information needs to be both processed (for logic and rendering) and delivered over the network.

Due to the large and dynamic contents of common 3D virtual environments and games, the prior download of such worlds is not a practical solution. For instance, Second Life [11] hosted 34 terabytes of user-generated, dynamic content in 2007 [12], [13]. Therefore, the game companies prefer to distribute their products in the form of CD/DVD media.

A pre-installation of such environments using CDs/DVDs is an un-realistic approach for three main reasons: First, it restricts the distribution process of such games. Secondly, casual players who want to try many different games and environments to decide which one to buy will be unable to do so [14]. Thirdly, it is impractical for game developers to update their products after their final releases without re-building and re-selling the whole products. Moreover, due to the limited capabilities of mobile, handheld devices in terms of storage, they cannot have the entire VE installed on the device.

In addition to the impracticality of pre-installing such worlds, the prior download of these large games/VEs through the Internet is very time-consuming and not pragmatic. In VEs

and games, the user must be able to view the game content almost immediately and start playing the game as soon as possible, instead of waiting for a progress bar, which is common in presentational applications such as audio/video broadcasting. Most online game portals will not accept a game in which some sort of gameplay does not start after downloading, at most, 1 MB of data [15].

Therefore, the continuous and real-time delivery of 3D content over the network is needed in order to allow user interaction without a full download. This is called *3D streaming* [16]. These 3D contents could be anything related to 3D objects, such as textures, animation, and scene graphs.

In other words, in the case of online 3D games and online virtual environments, 3D streaming techniques facilitate the need for loading new 3D models, textures, materials, etc. at runtime. This feature also lets designers update the models without rebuilding the whole project, i.e., some 3D contents that are not originally built into the game can be streamed and added later on. These additional contents can be anything, such as additional weapons, environments, characters, or even full levels. The delivery of such contents must be dynamic and interactive on demand, for example, when different items and 3D objects come into our view, certain contents must be delivered.

By its very nature, a dynamic 3D streaming approach at runtime inherently addresses many of the above-mentioned issues, which are associated with online 3D games and virtual environments.

Given the apparent advantages provided by dynamic 3D streaming techniques, it is therefore desirable to employ it as a feasible alternative to traditional approaches in supporting the operation of online 3D video games and virtual environments.

1.2 Research Problem and Objective

While applying novel mechanisms for the dynamic loading of 3D objects at runtime will solve problems such as a very time-consuming download process as well as problems associated with the distribution process of CDs/DVDs that are required for the pre-installation of large 3D games and virtual environments, it does introduce its own set of problems, which must be addressed. Creating online 3D virtual environments in a dynamic manner is often more complex without the presence of a specific algorithm that knows which objects should be streamed in each level.

As reported in [4], despite the promising nature of 3D media, significant challenges remain to be solved in order to bring them into next generation computing platforms, including mobile devices. As each mobile handheld device has limited capabilities and resources, such as limited battery life, limited bandwidth, higher network latency due to the nature of wireless networks, limited processing, and limited display, the usage of 3D media in mobile platforms will be restricted [4]. In addition to this, using more bandwidth, even if it is available, will contribute to more energy consumption and faster battery drainage. Therefore, even with 4G wireless networks that provide bandwidths of up to 100 Mbps, we cannot translate the availability of higher bandwidth into the continuous consumption of it [4].

As such, novel techniques must use the local resources of different handheld users efficiently in order to accomplish tasks as optimally as possible.

The focus of this work lies in the 3D streaming aspect of online mobile gaming. Each handheld user must receive the 3D content dynamically from the server, based on his specific game context. As such, our design must allow the game to start quickly and for the content to continuously load and interact with the user as fast as possible. In our approach, we provide such features based on which content is more important for the player at a specific point in time in the game, i.e., our approach is context-based.

In the case of the mobile online 3D gaming experience, the goal of the system is for a handheld user to be able to receive the game contents that are matched with his specific configurations and his specific device variations. Therefore, it is crucial for our system to be context- and content-aware in such a way that it can adapt 3D media to the mentioned specific needs.

Specifically, the goal is to take advantage of users' specific contexts and needs and to offer a new approach towards the delivery of live game information for mobile games over the network. By doing so, we are trying to reduce the number of objects subject to streaming, and as a result we can address the challenges of limited bandwidth and resources that mobile games are faced with. In other words, our research focuses on reducing the total size of data needed to be streamed from the server and be downloaded by the clients. Therefore, reducing the battery usage, processing power, etc. is a consequence of our method.

1.3 Research Contributions

This thesis presents an activity-centric and context-aware 3D streaming architecture as a feasible solution to make online 3D gaming experience available for mobile handheld devices, taking into account their specific needs and configurations. The system is based on the virtual context in which the player is playing the game.

Several contributions are included in this new system, which are as follows:

- A fast, context-aware algorithm for the streaming of 3D contents to mobile handheld device users.
- Definition and formulation of an "importance factor" for each object, considering the importance of that object for the current situation of the player within the virtual gaming world.
- A method for prioritizing different 3D objects based on their importance factor and selecting and streaming only the most relevant objects to the players.
- Optimizing the object selection and prioritization with respect to the globally defined constraints of maximum download size, maximum bandwidth usage, and energy consumption defined by each mobile handheld device user.
- Design, implementation, and evaluation of a framework which could be easily deployed to many different games without much effort so as to make the context-aware 3D streaming mechanism available for them.

1.4 Research Publications

The peer-reviewed publications related to the field of networked gaming are given below.

Journal Papers:

Hesam Rahimi, Mahdi Hemmati, Ali Asghar Nazari Shirehjini, Shervin Shirmohammadi, "Activity-Centric Streaming of Virtual Environments and Games to Mobile Devices," in *Springer's Multimedia Systems Journal (MMSJ),* (submitted under review), 2011.

Conference Publications:

Hesam Rahimi, Ali Asghar Nazari Shirehjini, Shervin Shirmohammadi, "Activity-Centric Streaming of Virtual Environments and Games to Mobile Devices," in *Proceedings of IEEE Symposium on Haptic Audio Visual Environments and Games (HAVE 2011)*, Qinhuangdao, Hebei, China, October 14-17, 2011. [17]

Hesam Rahimi, Ali Asghar Nazari Shirehjini, Shervin Shirmohammadi, "Context-Aware 3D Object Streaming for Games," in *Proceedings of ACM/IEEE Network and Systems Support for Games (NetGames 2011),* Ottawa, Ontario, Canada, October 6-7 2011. [18]

Hesam Rahimi, Ali Asghar Nazari Shirehjini, Shervin Shirmohammadi, "Context-Aware Prioritized Game Streaming," *Proc. Workshop on Interactive Ambient*

Intelligence Multimedia Environments, in Proc. of IEEE International Conference on Multimedia & Expo (ICME 2011), Barcelona, Spain, July 11-15, 2011. [19]

Hesam Rahimi, Saurabh Ratti, Ali Asghar Nazari Shirehjini, Shervin Shirmohammadi, "Unsynchronized Multiplayer Networked Games: Feasibility with Time Rewind," in *Proceedings of ACM/IEEE Network and Systems supports for Games (NetGames 2010), Taipei, Taiwan, November 16-17, 2010.* [20]

1.5 Thesis Outline

The majority of this thesis presents the design and evaluation of our activity-centric framework for 3D object streaming, which uses the game context for efficient object selection and prioritization. Accordingly, the remainder of this thesis is structured as follows:

Chapter 2 - Background provides background information on 3D levels of detail (LOD) as well as runtime frameworks for LOD.

Chapter 3 - Related Work discusses existing approaches taken by a variety of 3D streaming frameworks and architectures.

Chapter 4 - Overview of Proposed Approach specifies the design of the activity-centric architecture, specifically regarding context-awareness and using players' game states in order to stream 3D objects accordingly.

Chapter 5 – Implementation details the architecture's proof-of-concept implemented for evaluation purposes.

Chapter 6 - Performance Evaluation presents the evaluation results of simulations carried out to determine the performance of the aforementioned proof-of-concept.

Chapter 7 - Conclusion, Discussion & Future Work summarizes and concludes the thesis while outlining venues for future research.

Chapter 2 - Background

In order to make this thesis a self-explanatory document, in this chapter, we will explain some fundamental concepts that will help an ordinary reader to better understand the thesis material. In this regards, we will explain what the concept of level of detail (LOD) is in regards to 3D models.

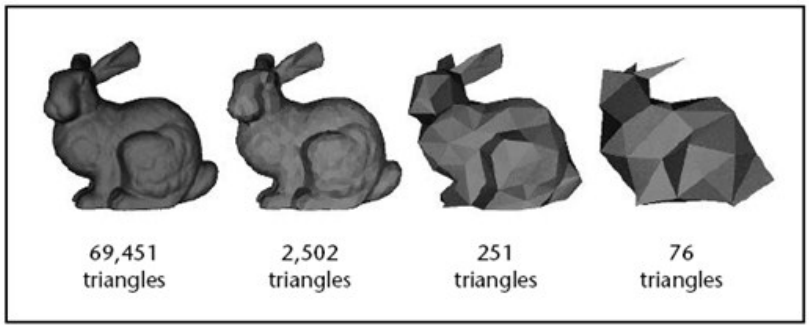
2.1 Level of Detail

Level of detail is one of the important concepts in the field of computer graphics. The first research project performed in this area belongs to James Clark's 1976 Communications of the ACM paper entitled "Hierarchical Geometric Models for Visible Surface Algorithms" [21].

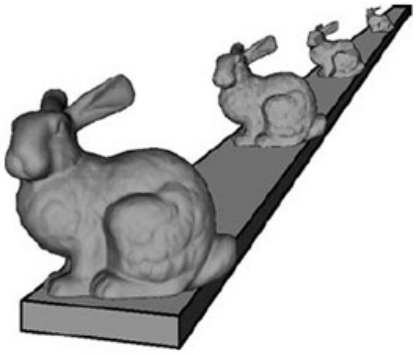
The development of today's 3D models, which can be measured by the number of polygons, is not proceeding at the same pace as the improvements in the hardware required to render them. This is why we are interested in not exceeding the affordable number of polygons for our platforms, mainly due to our hardware limitations. The problem could be formulated in terms of limited rendering resources, limited energy, and limited bandwidth available in handheld devices. Moreover, using state-of-the-art techniques, even the offline rendering of animations can take advantage of regulating the level of detail.

The primary idea of LOD is summarized in Figure 1. The entire concept can be stated as: using a less detailed representation for small, distant, or unimportant portions of the scene.

As can be seen in Figure 1, we have a few versions of a given object, which are known as levels of detail or LODs, and each version is less detailed and faster to render than the one before it. In the next section, we will explain some existing frameworks for LOD [22].



(a)



(b)

The fundamental concept of LOD. (a) A complex object is simplified, (b) creating levels of detail or LODs to reduce the rendering cost of small, distant, or unimportant geometry [Luebke 01a]. Copyright © 2001 IEEE.

Figure 1: fundamental concept of level of detail [22].

2.1.1 LOD Frameworks

The three basic frameworks for managing level of detail are [22]:

2.1.1.1 Discrete

Discrete level of detail is the traditional approach to LOD, which was proposed by Clark in 1976. As described before, the goal of this approach is to create multiple versions of every object that have different levels of detail. This is done during an offline pre-process. Then, the proper version or LOD is chosen to demonstrate the object at run-time. Since the creation of the LODs has been done during an offline preprocessing, the simplification process cannot predict from what direction the object will be viewed. That is why the details of the objects decrease uniformly, and because of this, we refer to discrete LOD as isotropic or view-independent LOD. The simplicity of discrete LOD is its main advantage. We only need to create as many versions (i.e., details) of the object as needed during the pre-process and select some of those versions for rendering at run-time.

2.1.1.2 Continuous LOD

In continuous LOD, instead of creating LODs during the pre-processing stage, a data structure encoding a continuous spectrum of detail is being created, and at run-time, the desired LOD is then taken out from this data structure. In this approach, rather than selecting a detail from a list of pre-created versions for a given object, the LOD for each object is specified exactly. Therefore, better granularity is the main advantage of this approach. This approach results in having exactly as many polygons as needed and leads to a better use of resources and a higher overall fidelity for a given polygon count. The streaming of polygonal models can be done using continuous LOD because a simple base model can be streamed first. Refinement levels are then streamed, followed by the base model.

2.1.1.3 View-dependant LOD

This is an extension to continuous LOD, which uses view-dependent simplification in order to dynamically select the most proper LOD for the current view. This is why view-dependent LOD is also known as anisotropic LOD. In other words, using this approach, nearby portions of the object can be shown at a higher resolution than distant portions.

2.1.1.4 Level of Detail in Practice

Due to the extra processing at run-time and the extra memory required for a view-dependant and continuous approach, discrete LOD is still the most common approach in practice [22].

2.1.2 Runtime Frameworks for LOD

After finishing the offline task of LOD creation, we must somehow manage the created LODs at runtime. Designing a run-time framework for LODs in a discrete approach is quite simple. In each frame, we merely need to select the appropriate LOD for the representation of a given object. However, this is more complicated in regards to a continuous or view-dependant approach. In this section, these methods are studied.

Screen shots of an *Unreal* tournament player showing how level of detail reduces as distance increases. The original model (a) contains over 600 polygons (excluding the weapon), whereas (b), (c), and (d) show, respectively, vertex counts reduced to 75%, 50%, and 25%. Copyright © 1999–2001 Epic Games, Inc.

Figure 2: an example of the Unreal game engine using different LODs ([22]).

Switching from a lower to a higher resolution model in LOD management is one of the most crucial issues. Generally, although the size and the distance of the targeted objects are the significant factors, the more important and challenging concern is to find out, before transitioning to a simpler model, how small and far away the object should be. In the following paragraphs, we present two major runtime frameworks for LOD.

2.1.2.1 Distance

In a distance-based framework, the levels of the detail can be managed easily. In order to perform this task, a distance could be allocated to each LOD that can represent the object. By choosing a lower LOD for far objects, the fidelity of the image has not been influenced that much, because fewer high-detail features of a distant object are visible [22].

Historically, flight simulators were among the first applications to use distance-based LOD heavily [23]. Figure 2 is an example of the Unreal game engine, in which a support for distance-based continuous LOD for 3D models has been provided as well as the smooth adaption of texture map detail [24].

For the implementation of such a distance-based scheme, a data structure is needed to store the different levels of detail for each object. A list of distance thresholds to indicate when each LOD should be used is needed as well.

In order to determine whether the distance exceeds the specified threshold, few conditional statements are required. The only computation required is calculating the 3D Euclidean distance between two points. Although distance LOD is simple, it also has some disadvantages. Considering the fact that the distance to the viewpoint can change depending on the orientation, selecting an arbitrary point for all distance calculations leads to inaccurate results and popping effects. Perhaps, the best method is to calculate the nearest distance to the viewer of the object, scale the object, use a different resolution, and change the fields of view. On the other hand, simply measuring the distance does not consider the parameters of the perspective projection used to render all objects [22].

2.1.2.2 Size

A sized-based technique has some advantages over distanced-based LOD. It is insensitive to factors such as display resolution, object scaling, or field of view. A size-based technique is an alternative in that the system can use a screen space criterion, where distance-based criteria measure the distance from viewpoint to object in world space. Size-based LOD

techniques don't switch between LODs based on a series of distances, but rather switch based on a series of size thresholds. This can be done because objects get smaller as they move further away [22]. Perhaps, objects that appear larger to the observer "contribute" more to the image (see Figure 3).

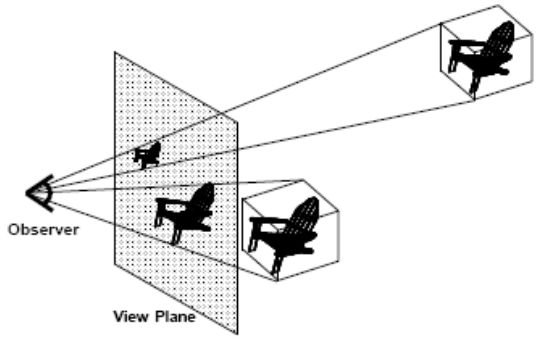


Figure 3: objects that appear larger "contribute" more to the image ([25]).

Another advantage of using size-based LOD is that rather than asking the user to choose an arbitrary point for the calculation, it uses the entire object. Therefore, the results from size-based LOD techniques for modulating LOD will be more precise than the results from distance-based techniques. Although this technique has some advantages, it should be taken into account that this method requires more computation because a number of world coordinates must be projected into screen coordinates.

Chapter 3 - Related Work

Much research has been done in recent years in 3D content delivery for virtual environments and video games, in which the dynamic loading and streaming of 3D objects has been discussed as an option for its advantageous properties. These works contain varying approaches to applying 3D streaming concepts and technologies.

In this thesis, 3D streaming is a technique that delivers 3D content over networks in real-time to allow users to navigate a virtual environment without a prior download or complete content installation. As reported in [26], the main resource bottleneck is usually assumed to be the bandwidth [27]. Therefore, the goal of 3D streaming is to provide a mechanism to somehow reduce the bandwidth usage in a proper way that guaranties maximum satisfaction. The simplification and progressive transmission of the content are among the dominant strategies regarding this [28].

In [29], Hu categorizes the current 3D streaming techniques into four main types:

- Object streaming
- Scene streaming
- Image-based streaming
- Visualization streaming

In the following sections, we will discuss the related works in each category except the fourth one, which is beyond the focus of this thesis. After that, we briefly compare how 3D

streaming would be different from media streaming. We also discuss the concept of gaming on-demand or cloud-based online gaming as well as some recent developments regarding this. Then, we explain the problem associated with the discussed related works. We will also compare our approach to these existing works and explain how our proposed system differs from them.

3.1 Object Streaming

The first category in current 3D streaming techniques is "object streaming," which deals with the streaming of a single object.

3.1.1 Progressive mesh

In order to help clients interact with the 3D data without a complete download, Hoppe [30], in 1996, was the first person to introduce the concept of a *progressive mesh*. Progressive mesh is a technique that stores an arbitrary triangular mesh as an appearance-preserving but much coarser base mesh and a number of refinement pieces [26]. Using this technique, once the base mesh is downloaded, a remote client is able to begin interacting with the object. Additional pieces will be streamed later. The base mesh will be refined incrementally by receiving these additional pieces, called vertex splits. Figure 4 (Hoppe [30]) shows the overall concept of progressive meshes.

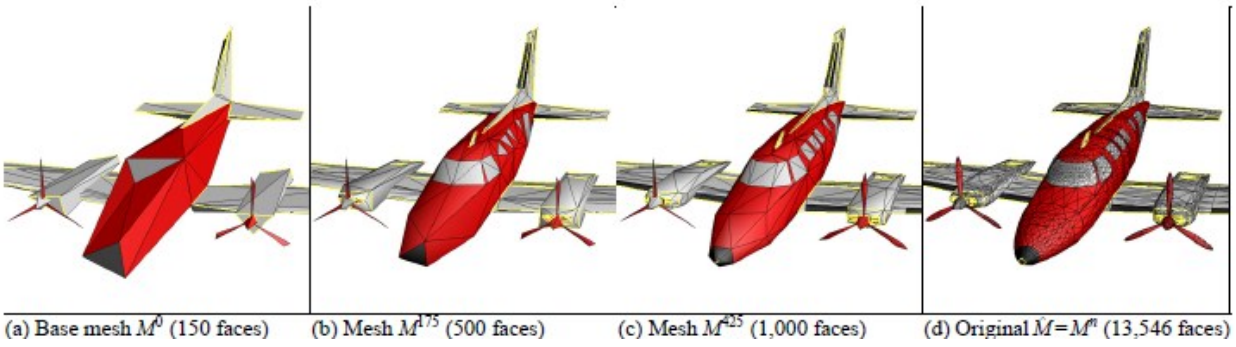


Figure 4: progressive mesh (Hoppe 1996 [30]).

The user can quickly see a coarse version of the mesh and may stop the transmission if he finds the quality of the already-received mesh to be good enough or even if he loses interest in the mesh [12].

3.1.2 View-dependent streaming

The idea of *view-dependant* streaming [31], [32], [33], [34] was introduced to consider the user's viewing angle in order to improve visual relevance by giving priority to the currently visible areas because the original progressive mesh technique does not consider the user's viewing angle. These methods are based on the view-dependant refinement methods proposed by many researches, for example, Xi et al. [35].

Since typical 3D content may include other data types besides polygonal meshes [26], streaming techniques are also available for other data types, such as textures [36], animations [37], scene descriptions, [38] and others.

3.2 Scene Streaming

Scene streaming is the second category of currently available 3D streaming techniques. In scene streaming, as opposed to object streaming, an entire scene, which usually has many objects placed in different positions within a virtual environment, will be streamed.

There are usually more objects in the scene than the user can see at a given time; therefore, in [26], Hu has divided scene streaming into two stages: *object determination* and *object transmission.* The first stage deals with the selection of objects in a given scene, based on visibility determination techniques used at the server. After culling away invisible objects (from the user's point of view) for the second stage, the methods described in Chapter 2 - could be used. Figure 5 ([39]) shows an example of a 3D scene with many visible objects.

Figure 5: an example of a 3D scene ([39]).

The way in which to perform the first stage, i.e., object determination, properly requires further examination. Different objects in the scene should be streamed from the server to the client according to the user's ever-changing position and angle of view. Therefore,

many techniques are available to determine a set of objects in a given scene that is subject to streaming over the network.

Normally, in virtual environments (VEs), which can be very large, a user only visits a small region of the environment that is visible to him at a given time. In order to save bandwidth and computing power, a client receives the geometric information for only that specific region. Existing VEs based on this region-based approach include DIVE [40], CALVIN [41], Spline [42], and VIRTUS [43]. In such systems, the environment is divided into a number of "pre-defined" regions and before the user interacts with a given region, the full content of the region must be downloaded. However, in such systems, whenever the user changes the region, the interactivity of the system becomes slow because multiple pauses are needed for a region to be completely downloaded.

Other works propose an interest-based approach [44], [45]. Such techniques use an area of interest (AOI) to determine object visibility, and the client only receives update messages within his AoI. NetEffect [46] is among such systems.

These mentioned systems do not provide any mechanism to control or guarantee visual quality, although they may reduce the amount of game content for downloading [47]. Moreover, the download time might still be too long. In recent high-quality games, there may be too many objects inside the mentioned AOI or the pre-defined region. Also, some objects might be very large in size.

To remedy this problem and in order to build up the visible region of the game scene very quickly, Constructive Solid Geometry (CSG) has been used in Second Life [11] to allow a

primitive of the object to be transmitted with a higher priority if it is a component of multiple visible objects. Moreover, Cavagna et al. [48] represents each object using several levels of detail (LODs). The lowest LOD will be transmitted first, and higher LODs will be gradually streamed upon request. This is similar to progressive mesh transmission, which was explained before, but multiple copies of each object will be sent each time if there is a request for higher LODs.

3.3 Image-based Streaming

In Hu's categorization in [26], image-based streaming is the third type of 3D streaming.

In image-based streaming techniques, such as [49] and [50], 3D content is stored in the server and clients only receive 2D rendered images generated in real-time by the server. This approach is good for clients with thin functionalities (i.e., no 3D acceleration capability and low processing power). However, it suffers from poor scalability and interactivity because the server processing power is limited as well.

In recent years, even mobile phones have become equipped with 3D rendering capability. Therefore, there is no concern about thin clients anymore.

In the case of online virtual environments and games, 3D streaming is the more suitable and preferred method as compared to image-based streaming. This is firstly because it provides the client with more interactivity. In 3D streaming, the user would be able to manipulate the 3D object and zoom in so as to see greater detail, while with an image-based approach, each time the user wants to manipulate the 3D object, new images must be rendered and streamed [16]. Secondly, rendering images of very large virtual

environments, those that have many objects, will impose a very high workload on the servers because different clients may have different views of the VE with different quality requirements [47].

3.4 Media Streaming vs. 3D Streaming

As reported in [26], in media streaming, the *content access pattern* is linear because media streaming views content as being one-dimensional (i.e., time) and sequentially accessible. However, in 3D streaming, the content access pattern would be non-linear and less predictable because 3D streaming views content in a multi-dimensional space (i.e., the x and y axis, view orientation, etc.) and accesses it according to user behaviours [51].

3.5 Cloud Gaming Platforms

Taking advantage of video streaming, cloud gaming, also called gaming on-demand, has been introduced recently. Cloud gaming is a type of online gaming in which the actual game is stored in the operator's or game company's server and is streamed directly to the users [52]. In this type of online gaming, all of the user inputs will be sent to the server and processed in data centers on specialised servers, and the server will then send back the response to the clients as rendered videos over the internet.

OnLive [53] and Gaikai [54] are two examples of recent developments in cloud gaming platforms. With OnLive, one can play a game instantly on one's TV, PC, Mac, or even mobile handheld devices like iPhones or Android-based phones. Using OnLive, users do not need to install games. Instead, they can instantly play the game and receive the streamed game online over their Internet connection. Gaikai also offers a cloud platform that can deliver

video games within seconds to all web browsers, operating systems, and devices. On February 27, 2011, Gaikai officially launched its game streaming service [55], which provides its users with a demo of a game using only a web browser, without downloading the game to the user's computer or installing the game using disks.

However, all these cloud gaming platforms are based on video streaming, which is very bandwidth-consuming and is only useful if the player does not have a computer or terminal. For example, it is helpful for thin clients like TVs. OnLive's lowest connection speed of 1 Mb/s would consume around 450 MB per hour, which is by far more than the common cap of broadband packages. They have relatively low cap limits, such as 5 or 10 GB/month [56].

3.6 Problems Associated with Related Works

Due to the limited network bandwidth of the mobile clients, these above-mentioned methods still cannot guarantee that every visible object can be sent to the mobile target device fast enough. In addition, all objects are treated the same, no matter how important they are for the user's current activity and context. The work in [47] addresses this issue to some extent by focusing on the prioritization of objects in a given scene based on object scope and viewer scope. In [47], the idea is to transmit only the objects whose scope has some overlap with the player's viewing scope. While this improves performance, priority is simply defined by whether or not an object's scope is inside the viewer's scope. Game context is not considered.

In all the above-mentioned approaches to object determination in a given scene, priority will be given always to those objects that are closer to the player. In other words, the proximity of the player to objects in the scene is the only parameter considered for culling away other objects. We argue that proximity, distance, or user visibility is not always the best method of selecting a subset of objects for streaming. The reasons for this will be discussed in following paragraphs.

While such distance-based approaches provide a better visual quality for closer objects, other objects that are further away are not transmitted or only streamed at a lower quality. However, the context in which the virtual world is rendered does not depend only on the distance between objects and might be determined by other priorities. For example, when fighting an enemy in a jungle, the enemy has a higher priority than the trees and surrounding plants and should be updated with higher quality in terms of both resolution and update frequency, regardless of how close or far the enemy is in the visible range. Distance-based approaches, on the other hand, would simply render whatever is closer to the player with the higher quality. As a consequence, many objects such as trees are sent to the player at a higher quality than required and without having a semantic relevance to the game context. While this might be acceptable on a personal computer, it leads to a waste of resources on devices such as mobile phones and other portable devices. In addition, some relevant objects might be transmitted at a lower quality only because they are far, even though they may be highly important.

In contrast, our approach is to consider the importance of an object for the current situation of the player within the virtual gaming world. As will be described later in detail,

in an activity-centric approach, our algorithm considers the context parameters relevant to the activities inside the game and uses this information to select and prioritize objects for progressive streaming purposes. This should ensure that only the most important objects are streamed to the clients, where importance is defined by the game designer within the game context. Our dynamic algorithm can optimize the object selection and prioritization with respect to the globally defined constraints of energy consumption and bandwidth constraints or qualitative requirements, such as immersion and visual quality.

Here, we introduce the idea of prioritized context-aware progressive streaming. In each frame of gameplay, we consider the context of the virtual game world to decide which objects are more important for the accomplishment of the player's current activity. This is different from visibility-based or interest-based approaches because visibility becomes just one factor among many in our context model. Also, in our work, based on current network conditions and some other user-defined restrictions, such as the available battery level of the mobile device and/or the available download limit, we optimize our prioritized list and select the most important objects to be streamed given the resource restrictions of the receiver.

To the best of our knowledge, no other research project has investigated utilizing the game context in a virtual environment for object selection and prioritization in a given scene. Additionally, existing approaches do not consider the receiver's resource restrictions. Hence, they are not suitable for mobile clients.

Chapter 4 - Overview of Proposed Approach

The overall architecture of our system consists of three major components, as shown in Figure 6. First, we perform a segmentation of the virtual world. This means that for a player's current scene in the game, we extract all the objects in that specific scene. Then, we prioritize all of the objects in that scene. This prioritization is based on several heuristics, which we will explain in Section 4.2. After prioritization based on the objects' ranking and based on the global constraints defined by different users, an optimal subset of the objects will be chosen for streaming to the target device for rendering.

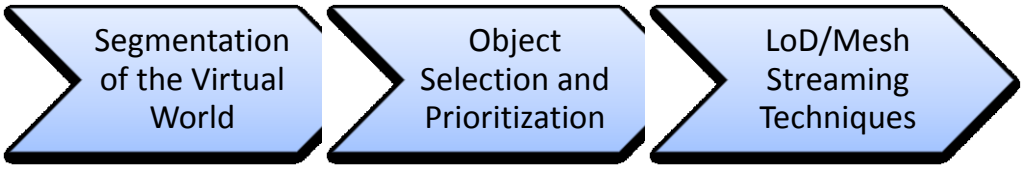


Figure 6: major components of our system.

This subset contains only the most relevant 3D objects for the current context in which the player is playing the game. Existing methods for level of detail and progressive mesh streaming, which was explained in detail in Chapter 2 -, can then be applied to the selected objects prior to their transmission. This is mainly due to the fact that sending the whole 3D model of each object to the client's handheld device may cause a long pause in his experience. Therefore, progressive mesh streaming can be used to let the user have a glance at the object immediately and then receive the object in full detail progressively. In

other words, the system sends a small portion of the model to the client's handheld device, where the whole object will be progressively reconstructed later on. This will offer an additional reduction in model complexity.

In this thesis, our main focus and contribution is to step 2: object selection and prioritization. Here, for each activity, we either select an object for streaming or we skip it because it is irrelevant to the current scene. This has been done in this thesis because the focus of this work is designing and developing an object selection and prioritization algorithm based on the game context. However, practical 3D streaming techniques, in reality, should not ignore irrelevant objects. Instead, they should stream those objects in lower levels of detail (LODs), while more important objects will be streamed with higher LODs as required. This can help the player to have a better visual impression of the environment by having a glance at all the objects together at the same time. In our proposed approach here, however, if the user limitations allows, the server will stream irrelevant objects after the most important ones.

4.1 System Requirements

Our proposed system requires the following components:

- A context model that is aware of the importance of different objects in different situations inside the game environment. This context could be represented by ontology, which will be described in detail next.
- An object selection algorithm for sorting, culling, optimizing, and streaming the most important objects.

In the following sections, we will describe the design of these components in detail.

4.2 Ontology-Based Context Definition

Ontology is a formal representation of knowledge as a set of concepts within a domain and the relationships between those concepts. Ontologies are the structural frameworks for organizing information. Ontology-based data integration involves the use of ontology(s) to effectively combine data and/or information from multiple heterogeneous sources [57]. We have used this technique in order to combine semantic information related to the game.

In order to carry out this research, we have analyzed a few first person shooter games to develop a context model. It should be noted that, while our analysis in this paper was done on First Person Shooter (FPS) games, this does not reduce the generality of our proposed approach. A similar analysis can be performed to define the ontology-based contexts of other games. Our context model for the FPS games consists of the following entities:

- player
- activities
- location
- stationaryObjects
- surroundingEnvironment
- buildingsObjects
- ownableObjects
- enemies

Again, the exact entities are not the issue here, and those above are merely examples. A game designer can very specifically and completely define all the entities for a given game, and our approach would still be valid. In a game such as Battlefield Heroes [58], which is shown in Figure 7, there are several activities one could perform inside the game, i.e., any kind of action or movement. In our context model, we store all of these activities. Every activity requires a set of objects. Each of these objects is associated with a 3D model inside the environment, and the user needs those objects in order to accomplish his desired activity. This set of objects is also stored in our context model, and each is related to its associated activity.

Figure 7: a screen shot of Battlefield Heroes [58].

Moreover, each object offers a "service" to the player, and each service has a "service zone," in which the service is performable. There are different types of zones, so each object has its specific services and service zones. For example, a weapon in an FPS game offers a

"shooting" service, and this service has a "shooting zone." However, the "shooting zones" of different kinds of weapons might be different from one another. A pair of binoculars in an FPS game offers magnified visibility to the user and has a "visibility zone." Therefore, the context model consists of all potential and achievable activities inside the environment, followed by all associated objects needed for each specific activity and also all the services offered by each object, followed by their service zones. In our context model, we also store all the associations between different activities and objects. Considering the service zone of each object, we should be able to find out which objects are necessary and important for a particular activity. As an example, in an FPS game, depending on the type of weapon the player has and depending on its "shooting zone," some objects would be necessary for that activity (shootable objects that are in the effect area of that particular weapon). Finally, for the purpose of context-aware object selection and prioritization, we need to know the importance of every single object for a given activity. This is done by defining several aspects and heuristics, such as the visual satisfaction of the player, immersion, how each object affects the task accomplishment of its associated activity, and also how each object affects the orientation of the player inside the environment. We will describe this in detail in the next section.

For a given activity, every single object has a value for all of these aspects, which is known as the importance ratio. This ratio is a gauge indicating to what extent an object is important for the fulfillment of that aspect of that specific activity. As an example, the importance ratio of a weapon for the task accomplishment aspect of the shooting activity is

γ_w (see Table 1), for the visual quality aspect of the shooting, it is β_w, for the immersion aspect, it is α_w, and lastly, for the orientation aspect, it is θ_w.

Table 1: importance Value Table.

Activity/Object	%	weapons	teammates	enemies	walls	trees
Navigation and Orientation	Immersion (K_i)	α_w	α_t	α_e	α_s	α_r
	Visual quality (K_V)	β_w	β_t	β_e	β_s	β_r
	Task accomplishment (K_T)	γ_w	γ_t	γ_e	γ_s	γ_r
	Navigation (K_N)	θ_w	θ_t	θ_e	θ_s	θ_r
Shooting	Immersion (K_i)	α_w	α_t	α_e	α_s	α_r
	Visual quality (K_V)	β_w	β_t	β_e	β_s	β_r
	Task accomplishment (K_T)	γ_w	γ_t	γ_e	γ_s	γ_r
	Navigation (K_N)	θ_w	θ_t	θ_e	θ_s	θ_r

These importance factors are designed and measured by game designers, who are the most knowledgeable persons about the game scenario. The availability of such important factors is one of the requirements of our framework. It is worth mentioning that the measurement of these importance factors does not cause much overhead for the designers. The cost/benefit ratio for spending time to design such importance factors is low enough that it makes this feasible for game designers. Also, they only need to do this once, at the time of game scenario design. This will be explained further.

To assign units to these importance factors, we have implemented a normalization mechanism. As a result of this normalization, game designers are free to use any number and any measurement unit. For example, object (A) could be worth (α) importance units for a given activity, while object (B) could be worth ($n \times \alpha$) importance unit for the same activity, meaning that the importance of object B in a specific aspect is *n*-times more important than that of object A. After assigning importance factors to all objects for every activity, in order to normalize these factors, the importance of each object is divided by the sum of the importance factors of all objects. The resulting normalized importance factors, which we also call *priority indices* would sum up to one. This normalization scheme is justified by the fact that our object selection algorithm needs comparable importance factors for different activities. Since we are optimizing the "sum of importance factors," we will then normalize all importance factors in such a way that their sum becomes equal in a given activity. For example, after normalization, the total sum of the importance factors of all objects for "walking" should be equal to the total sum of the importance factors of all

objects for “shooting” and so on. This would help us to have a “fair” optimization and a “fair” object selection for streaming.

4.3 Proposed Context-Aware Streaming Framework

As mentioned before, our approach streams only a small portion of the environment to the client in each game frame. The difference between our approach and existing ones is in how we define this “small portion.” As discussed before, this small portion contains the objects that the user is interested in most, i.e., objects that are needed to fulfill the gaming tasks and keep the game enjoyable. Based on which parameters can we select and prioritize such objects? Unlike existing methods, which are based only on distance or region and provide a better visual quality for closer objects, we focus on the context in which the game is played. This context does not depend only on distance between objects and might be determined by other factors. For example, when fighting an enemy in a jungle, the enemy has a higher priority than the trees and surrounding plants and should be updated with higher quality in terms of both resolution and update frequency, regardless of how close or far the enemy is in the visible range. Distance-based approaches, on the other hand, would simply render whatever is closer to the player with higher quality. As a consequence, many objects are sent to the player at a higher quality than is required or without having a semantic relevance to the game context. While this might be acceptable on a personal computer, it leads to a waste of resources on devices such as mobile phones and other portable devices. In addition, some relevant objects might be transmitted at a lower quality only because they are far, even though they might be highly important.

In our approach, an algorithm considers context parameters relevant to the activities inside the game, knows the importance of each object, and uses this information to select and prioritize objects for progressive streaming purposes. This should ensure that only the most important objects are streamed to the clients, where importance is defined by the game designer within the game context. Our dynamic algorithm can optimize object selection and prioritization with respect to globally defined constraints of energy consumption and bandwidth or qualitative requirements, such as immersion and visual quality.

This algorithm is described in detail in Section 4.4.

In each frame of the gameplay, our framework is able to determine the current activity undertaken by the player. After recognizing the current activity, objects are prioritized based on their importance for that activity. In other words, using normalized importance factors, we set the "priority" property of each object for the current activity. After this prioritization, we sort objects based on importance, and we select the objects to be streamed to the client. This can be done in three different ways:

- We could stream all the prioritized objects in a decreasing order, taking into account their priority: objects with more priority will be streamed first. However, to stream all objects in real time, this method requires that there be no limitation on the client's maximum download size, which might not be the case for many mobile devices.

- Depending on the type of Internet connection in use, we could define a maximum download size, set by the service provider or the mobile client. Our framework then assures that this maximum download size will not be violated. This will be done by selecting and streaming an optimum subset of objects from the prioritized list. This subset contains some objects that the sum of total importance of those objects is the maximum comparing to all other possible subsets. Also, the total size of those objects is less than the user constraint. For example, if a player's mobile client does not want to have more than 10 Megabytes download while playing the game, our system ensures no more than 10 Megabytes worth of objects from the prioritized list are sent. This is in addition to the fact that those selected objects would have the maximum total priorities among all possible subsets with a size of less than 10 MB. This optimization mechanism will be discussed further in Section 4.4.
- Also, not only can we abide by a specific constraint defined by a mobile client, but also we can abide by several limitations and perform an optimized object culling from our prioritized list, taking into account those limitations. Different types of players might have different types of constraints. As an example, apart from having a maximum download size, a player might have limitations regarding the battery life of the mobile device, so the maximum energy usage of the game cannot exceed X watts.

Based on these specific needs of a specific player, we can select an optimized subset of objects from our prioritized list, and by doing so, our framework guaranties that all the

limitations and player constraints will be met during the gameplay experience in an optimized way.

4.4 Object Selection and Prioritization Algorithm

Our algorithm is summarized in Algorithm 1. In each frame of the game, the current activity undertaken by the player is fetched (*act*).

The current activity of the player comes from an activity recognition module that is inside the game engine. This module is aware of the current activity the player is involved with in each frame of the game and also predicts future activities. Our research is not about this activity recognition module. We assume that this component already exists, although we mention here very briefly that approaches such as HMM models can be used for such game activity predictions [59]. Another similar module, called a Context Management module, is aware of game states. In order to conduct our research, in our implementation, the player sets his current activity manually.

Algorithm 1: Context-aware Streaming Algorithm

Input: $batteryLevel$: Available battery at client
Input: $maximumDownloadSize$: Maximum download size limited by the user
Input: $priorityIndex[]$: An array containing normalized importance factors
Input: $objectSizes[]$: An array containing the size of different objects

$act \leftarrow getCurrentActivity();$
$listOfAvailableSceneObjects \leftarrow getSceneObjects\ ();$
// the function *getSceneObjects()* uses the current position of the player and
// enemies. It also checks whether an object has already been streamed or not.

// get user constraints
$batteryLevel \leftarrow getBatteryLevelFromClient();$
$maximumDownloadSize \leftarrow getMaximumDownloadSizeFromClient();$

$L \leftarrow length\ (listOfAvailableSceneObjects);$

for $i \leftarrow 0$ **to** $L - 1$ **do**
{
  $priorityIndex[i] \leftarrow normalized(listOfAvailableObjects[i]);$
  $objectSizes[i] \leftarrow getSizeOf\ (listOfAvailbleObjects[i]);$
}
$listOfSelectedObjects \leftarrow optimize(listOfAvailableObjects, act,$
$priorityIndex, objectSizes, maximumDownloadSize)$

$prioritizedListOfObjects \leftarrow$ *Sort listOfSelectedObjects based on their priority index*

$L2 \leftarrow length\ (prioritizedListOfObjects);$

for $i \leftarrow 1$ **to** $\boldsymbol{L2} - 1$ **do**
{
  Stream the object[i] of the prioritized list ($prioritizedListOfObjects[]$) to the client;
}

Algorithm 1: context-aware streaming algorithm.

As mentioned before, we need to perform a segmentation of the virtual world, which means that for a player's current scene in the game, we must extract all the objects in that specific scene. This is mainly due to deciding which objects can potentially be streamed to

the player in that scene. This object extraction should take into account the current game state, including player and enemy position and orientation. In our pseudo-code, we have shown this as a list named listOfAvailableObjects.

Our algorithm then looks for the *importance factors* (explained in Section 4.2) of those selected objects and sets the *priority index* of each object. This *importance factor* for each object is based on a set of heuristics that are described in Section 4.2. We use those heuristics to develop a final parameter, which is named the importance factor. In other words, the importance factor is a result of several heuristics being considered in our prioritization. The higher this parameter, the more important the object will be. This priority is based on the current activity of the player. This means that given the same scene, two different players with two different activities would have different priorities for each object in that specific scene. We sort all objects based on their importance factor or their priority index, which is actually their normalized importance factor.

After prioritization, we have a sorted list of all the required objects for a given activity undertaken in a given scene. Next, we must select a subset of these objects that will be supported by the user constraints, such as battery level, available bandwidth, maximum download size, and the screen resolution of the target device, among other possible endpoint restrictions. This subset should be an optimized subset. In the following section, we will explain our approach to selecting an optimized list of objects based on user constraints.

4.5 Optimization of Prioritized Objects based on User Constraints

As explained in the previous section, we must select a subset of objects from our prioritized list of objects, and this subset should be created in such a way that it meets the user constraints. Therefore, object selection could be seen as an optimization problem. We are seeking to pick and stream a selection of objects in the visibility range of the player, which means the current segment or the current scene that the player is involved in. Those selected objects are the most important ones for the player to use in accomplishing his task based on his current activity. In other words, we are in search of a subset of objects with a maximum of total importance for the current activity of the player.

In this regard, there exist some technical constraints as well. The maximization of the total importance of the selected objects could be subject to constraints on total download size and battery usage, among others.

There is a vast and growing literature on power-aware computing [60] that provides various physical, empirical, abstract, and mixed models of battery discharge behavior, which could be utilized to design and implement a power-efficient computing and communication scheme for a mobile device. However, in the case of object selection and streaming, the most significant factor in battery usage is the capacity of data streamed to, *i.e.,* downloaded by, the mobile device. Therefore, it would be reasonable to consider total download size as the main constraint regarding the object selection problem.

Total download limit (*TDL*) is a real, tangible constraint for the game player because he is supposed to pay for his data usage. We assume that the user will set a download limit for each level of the online game to be played. This user input serves as the main constraint of our optimization problem. To sum up, we will formulate and solve an optimization problem, seeking to maximize the total importance of the selected subset of objects so that the total size of the selected objects does not exceed the download limit determined by the user.

Assume that there are *n* objects in the visibility range of the player in a certain scene of the game. If the *priority indices* of these objects for the current activity of the player are denoted by vector $\boldsymbol{p} = [p_1, p_2, \dots, p_n]^T$ and if their *download sizes* are $\boldsymbol{s} = [s_1, s_2, \dots, s_n]^T$, then we can formulate our optimization problem as a standard *binary integer programming* problem as follows:

$$\max_{\boldsymbol{x}} \ \boldsymbol{p^T}.\boldsymbol{x}$$

$$s.t. \ \ \boldsymbol{s^T}.\boldsymbol{x} \leq L$$

where $\boldsymbol{x}$ is required to be a binary vector of dimension *n,* and *L* is the download limit for the current round of object streaming. Both the objective function and constraints are linear functions of optimization variable $\boldsymbol{x}$**.** Therefore, the problem could be solved using a *linear programming* (LP)-based *branch-and-bound* algorithm [61]. Numerically robust and efficient implementations of this algorithm are available as commercial software packages [62], [63]. The resulting binary solution vector $\boldsymbol{x}$ determines which objects should be streamed and which ones should not, in order to maintain the download size below the limit while maximizing the total importance of streamed objects.

It is important to note that, theoretically speaking, the above optimization problem should be solved for each frame of the game. In practice, the frequency of running the optimization algorithm might be lowered to once every few frames. It is noteworthy that the so-called download limit, which is denoted by *L* in the following equation, is not the same as the *total* download limit entered by the user, which we denote by *TDL*. The gameplay of each level of the game normally consists of numerous rounds of object selection and streaming. While *TDL* is the allowed download throughout the entire gameplay of a certain level, *L* is the download limit for the current round of the optimization problem. Our optimal object selection algorithm includes a budgeting plan to fairly distribute the total download limit across the entire level, encompassing several rounds of object streaming.

Specifically, assume that *N* is the number of *all* objects in this level of the game. Moreover, suppose that I_j denotes the set of indices of those objects present in the visibility range of the player at the j^{th} round that have not been streamed until this round of object selection. A fair value for the download limit of the j^{th} round of running the optimization problem is:

$$L_j = \frac{\sum_{i \in I_j} s_i}{\sum_{i=1}^{N} s_i} . TDL .$$

The above budgeting plan maintains a uniform distribution of the download quota during the entire level. This budgeting plan is required to solve the problem of "meeting the user's maximum download size (his TDL)" early in the game by downloading the most relevant objects in the first iterations of the algorithm. In other words, the question is: How is the global limit achieved? How can we ensure that a complex, object-rich situation that

happens late in the session can be rendered without much loss of important items due to an earlier over-spending or a too-low general limit? Therefore, if we do not have a proper download budgeting, the user cannot continuously receive the related objects. Instead, after being involved in one activity, for example, shooting, he will receive all the relevant objects for shooting and his maximum download limit (or TDL) might be met by that time, so there is no chance for him to receive objects relevant to his other activities, such as walking, navigating, etc. Other relevant objects will not have an equal chance to be downloaded in the game later on. As a result, the player will encounter a "freeze" in receiving more objects. As such, we need to properly select the most relevant objects in a way that the user's Maximum Download Limit is achieved after several iterations or at the end of the game, and therefore, we must somehow allocate a "maximum download budget" in each iteration of the algorithm in order to guarantee that the user will still receive some objects in the future if he continues navigating the game world (until all the relevant objects get a chance to be downloaded). However, the user must at least have a reasonable minimum download limit if he wants to have a reasonable gameplay. We are not actually handling the cases in which the user's download limit is too low. We are trying to improve the relevance of the objects that are to be streamed as well as considering specific constraints from the users. Therefore, our framework is more efficient in comparison to existing approaches since we are trying not to waste the user's download limit, while other approaches will stream all the objects anyway.

The optimization algorithm is a sort of cut-off mechanism for the selection of a subset of objects in our prioritized list of objects. This subset should only contain the objects which

are optimum in regards to the available bandwidth and battery level of the client. This means that we will not go further than what is tolerable for the target device in each frame. By doing so, we have not only considered the context of the virtual environment, but also taken into account how much bandwidth and battery level or download limit is available on the target device. We only stream the most important objects with respect to those restrictions.

Chapter 5 - Implementation

Our framework is designed to be platform-independent. This means that our framework has the ability to be used and implemented in a variety of platforms and for different game genres. In this section, we explain our methodology regarding the implementation of different parts of the framework. Moreover, we will explain how easily this framework could be deployed in different games. This will be explained by showing two different proof-of-concept games, both using our streaming method. The second game has also been ported to a mobile device that is Android-based [64] in order to show the feasibility of our framework in mobile handheld devices.

5.1 System Architecture

Figure 8 presents the different processes in our framework. As can be seen in the figure, this system uses the game context and the game state in each frame of gameplay in order to decide which objects are the most important ones in a given segment/scene of the VE. Our framework then uses the importance values table to prioritize the objects. Finally, using the battery level and available network conditions of the client, the most optimum subset of objects from the prioritized list will be streamed towards the target device. Incremental Level of Detail (LOD) can then be used as well as progressive mesh streaming techniques in order to send the 3D object over the network. By doing so, we reduce the number of objects as well as the less-important parts of the objects to be streamed.

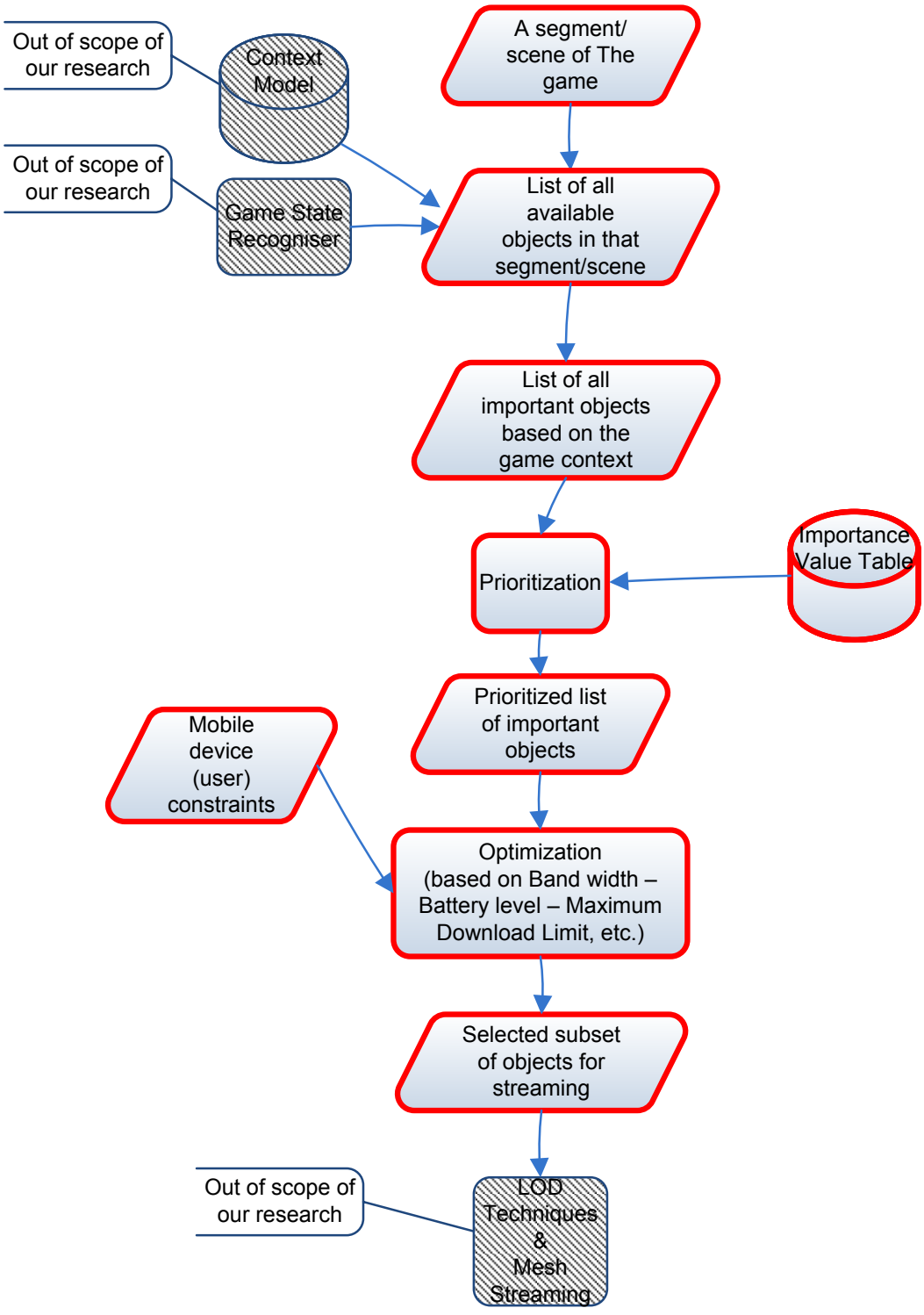


Figure 8: our system architecture.

By reducing the number of objects, the total download size will be reduced therefore the energy consumption of the handheld devices will be decreased, while the streamed objects still fulfill their role in the game. In other words, players can perform their activities as well as possible and have the maximum level of fun in the context of the specific game, given the resource-limited device they are using. The server further uses existing techniques for mesh streaming and levels of detail and streams the required objects in each frame of gameplay to the mobile clients.

5.2 The Game Context: Bootcamp

Following our approach, the ontology and game context should be provided at the time of game design and development. However, using reverse engineering methods, we modeled the context of an existing game: Bootcamp. This allows us to conduct the rest of our research, which requires such a context model. We played the game several times, captured the video of each gameplay instance, and analyzed the video in order to understand the functional importance of each object for each activity. The result is sufficient to test our method. However, for a complete and absolutely correct version of the object and activity "fact book," the game must be analyzed, and the context model must be generalized, preferably by the game designers themselves.

We maintain a list containing the importance of each object, designed by game designers, who know exactly what the game context is and what the importance of each object is for different activities inside the game.

For our game, we have designed an importance factor table similar to that in Table 1, which is shown in Table 2. Our importance factors here are designed only for the "task accomplishment" aspect of the gameplay, as explained before. Table 2 has only a small subset of all the objects used in our game. The complete object list as well as their size is shown in Appendix C.

Table 2: an example of importance factors for our proof-of-concept game.

Activity/Obj.	warehouse	fence	barrel	concrete_beam	iron_beam
Shooting	1	0	0	0	0
Walking	0.1	1	0.8	0.9	0.8

As previously mentioned, the goal of this research is to offer a new technique for object selection and prioritization based on virtual game context. Therefore, the main requirement of our system is to have a general context model for games. The context model described here and the importance values shown in Table 2 have been developed in an experimental way, using reverse engineering techniques. This is just an example to demonstrate the technical feasibility of our approach. In reality, the context and the importance ratios would come from game designers. The question now is: When should such a context be designed, and who should be responsible for it? The separation of the roles of game designer, engine developer, and game developer is as follows: First, game designers, at the time of game scenario design, are required to provide such a context for

engine developers. Game engine programmers, using the context model provided by designers, are required to develop a module inside the game engine that is responsible for the recognition of the current game activity in each frame of gameplay and for the recognition of current game states. Game developers could then take the advantage of our framework in order to select the required objects for each frame and stream only the most important ones. In other words, game design should proceed in such a way that it assures that the requirements of our framework will be used by game developers. Appendix B describes the entire process of game development.

Now, the question is: How much overhead does the proposed method introduce into the design/creation of a game? To answer this question, we must consider that having the game scenario available at the time of game design, the aforementioned context model is not anything further than merely formalizing the game scenario in a specific way, and this does not require a significant increase of game development costs. Fundamentally, the context model models the game scenario. The importance values table complements the context model and provides detailed information about the game scenario, meaning that for a given task in a given scene of the game, the context model together with the importance value table are not something further than information such as what objects are needed for the accomplishment of a certain task and to what extent each object is important for that task.

In other words, writing a story, sketching 3D models, animations, and textures, and designing the interaction, having in mind the different game situations in which this object will be used, and following an iterative design and development approach is costly. The

designer behind all these tasks will be forced to analyze all the realistic scenes and situations in which his object will be deployed. He must know with which other objects and activities his designed object will interact. Therefore, he will be easily capable of filling out an importance matrix.

Actually the importance factors already exist for a given game, but they are implicit and hidden in the minds of designers or within the iterative design processes. Those importance factors must be made explicit and ready for our framework in order to use them.

However, it is crucial to note that we do not mandate that one person create the importance factors for the entire game. In fact, this job can be divided among all the designers contributing to the game. Every designer will create the matrix for those objects created by him.

Moreover, these efforts could be justified by the benefits of our framework, which would result in the feasibility of massively multiuser online gaming (MMOG) for mobile devices. Thus, the cost/benefit ratio of our system is defensible.

5.3 Environmental Setup

For implementation, we have used the Unity 3 game development tool and engine [65]. This engine supports rendering, lighting, audio, physics, terrains, networking, and programming, which are among the most important features of game development. Unity supports three scripting languages: C#, Java Script, and Python. It can also link to Microsoft .NET libraries for networking and other tasks. It should be mentioned that for the sake of

simplicity, in our implementation, we have only considered the “task accomplishment” aspect of gameplay, and we have designed our importance factors only for this aspect. However, our implemented framework is able to support all the other game aspects mentioned above.

5.4 Proof-of-Concept Game

Our proof-of-concept game is based on a demo project that comes with Unity 3 by default, named Bootcamp. Fundamentally, it is an open source and simple 3rd person shooter environment in which the player is able to navigate and shoot, with no other features. We have used this environment as a base of our framework in order to apply our approach to object streaming inside the game. We have added the arrangement of objects in the scene, dynamic assets steaming support (from a web-server), context-awareness, and multiplayer capability to Bootcamp. Figure 9 shows a snapshot of Bootcamp running on a laptop.

Figure 9: Bootcamp running on a PC/laptop.

In order to show that our framework could be easily deployed in different games and in order to show that it is a feasible approach for mobile devices, we have also developed another proof-of-concept game for mobile devices that are Android-based [64]. This second proof-of-concept game is based on another demo project that comes with Unity 3.4, which is named *Angry Bots.* Angry Bots is a top-down First Person Shooter (FPS) game. Figure 10 shows Angry Bots running in a laptop/PC, and Figure 11 is a screenshot of Angry Bots running on an Android-based mobile device.

Figure 10: Angry Bots running on a laptop/PC.

Figure 11: Angry Bots running on an Android-based handheld device.

5.4.1 MMOG Capability

Our proof-of-concept also supports a massively multiplayer online gaming (MMOG) experience. For this purpose, we have used the Photon socket server [66]. Using a free version of Photon, we can create multiple "game groups" consisting of up to 20 players per group. Players who are in a same game group can then play together in the same level and have interactions with one another. The Photon socket-server utilizes the UPD protocol for

ultimate speed and performance, which is fast and reliable, though also prone to loss and without guaranteed delivery.

5.4.2 Gameplay Mechanics

In our proof-of-concept game, players can play in a local area network or via the Internet. To make Internet-based gameplay achievable, we have placed all 3D objects and assets that are to be streamed in a web-host in a 3D database server. Therefore, object streaming will be performed via the Internet, and as a result, different players can connect with each other and receive the game content dynamically from the web server. At the very beginning of the game, objects are not stored in the players' machines, but must be downloaded from the server dynamically on demand. The goal of the game is for a player to try to hit the weapon warehouses in the game as fast as he can. Since the game is multiplayer and different players are in competition with each other, when a player hits a warehouse first, the score for that hit will go to that player. The game will be finished when all the warehouses are hit. Also, players are able to shoot at and kill one another.

5.4.3 The Game

Figure 12 to Figure 15 show multiple snapshots for the same frame of our game, using different methods. The first two figures (Figure 12 and Figure 13) show the game's rendering using our method, and the second two figures (Figure 14 and Figure 15) show the game's rendering using a distance-based approach. Each group shows two activities, aiming and walking. In these pictures, the streamed objects are highlighted in red for better visual identification. They are not red in the actual game. As we can see, in our method, when the player is aiming and shooting (Figure 12), only important objects for shooting are

streamed, in this case warehouses. When the player is walking, only important objects for walking are streamed, in this case obstacles such as fences, barrels, etc. (Figure 13). However, in the distance-based approach, no matter what the current activity is, only the objects that are within certain distance of the player are streamed. As can be seen in the picture, there are two problems with this approach. First, when aiming (Figure 14), some warehouses are not shown because they are farther from the player than the obstacles. The latter are shown, but are useless for the aiming activity. Due to limited bandwidth, only objects closer to the player have been streamed and are shown, which do not include some of the warehouses needed for shooting. Second, the obstacles that are close to the player have been downloaded and are shown, so bandwidth has been wasted on objects that have no impact on aiming and shooting, wasting resources and reducing the quality of the interaction between the game and the player. This is also the case for other activities like walking (Figure 15). Using our activity-based methodology, on the other hand, we have not only skipped the streaming of irrelevant objects for a given activity, but we have also increased the overall experience and game quality on the mobile device for the player.

Figure 12: activity-based (aiming).

Figure 13: activity-based (walking).

Figure 14: distance-based (aiming).

Figure 15: distance-based (walking).

5.5 Android Implementation

In this proof-of-concept game, we showcase a method for the selective streaming of 3D objects in online mobile games. Our goal is to show the difference between our streaming approach and the distance-based approach. As such, we run our proof-of-concept game in three different methods at the same time in a multiplayer environment. These three different methods are as follows:

- The game is run on a PC, which has a larger display than mobile devices, in its full version with all objects streamed to the player.

- The game is run on a mobile device, and the distance-based approach is used for streaming 3D objects.

- The game is run on another mobile device, and our context-aware method is used for streaming 3D objects.

The first version of the game is used as a reference to see the game in its full version and to make comparisons with the mobile devices easier. As mobile devices still have limited battery lives, processing powers, memories, and display sizes compared to their PC counterparts, we show the full version of the game running on a PC, and we also show the game running on mobile devices with limited resources. Comparing the two mobile clients described before, each using a different method as their streaming technique, we can see that the client who uses our context-aware approach will give the player a higher quality of

experience with better interaction and higher scores in the gameplay experience, compared to the other mobile client. In the next section, we will describe our demo setup.

5.5.1 Demo Setup

Our demo environment, which we showed in the 10th Anniversary of the annual International Workshop on Network and Systems Support for Games (NetGames 2011) [18], consists of one PC as a server and two mobile devices (Android-based) as clients. All of these devices are Wi-Fi capable. We also have a wireless access point (WAP), which is connected to the Internet, offering Internet connection for all devices. Figure 16 shows our demo setup environment.

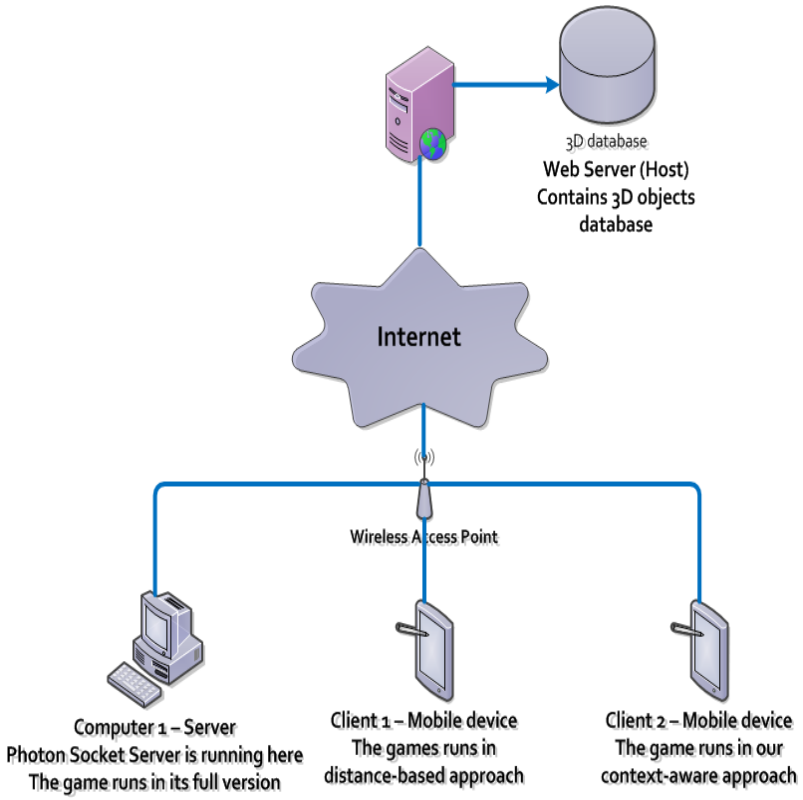


Figure 16: environmental setup for our demo as shown in NetGames 2011 [18].

We placed all 3D objects and assets to be streamed in a web server. The PC will play the role of the server in our environment. All other clients, i.e., mobile handheld devices, are connected to the server by using its IP address and a specific port number.

The next section describes the differences between the gameplay in our server and our different clients (mobile handheld devices), each running the game in two different activities.

5.5.2 Snapshots of the gameplay on different platforms

In this section, we show snapshots of a same scene on three different platforms: one laptop and two mobile handheld devices.

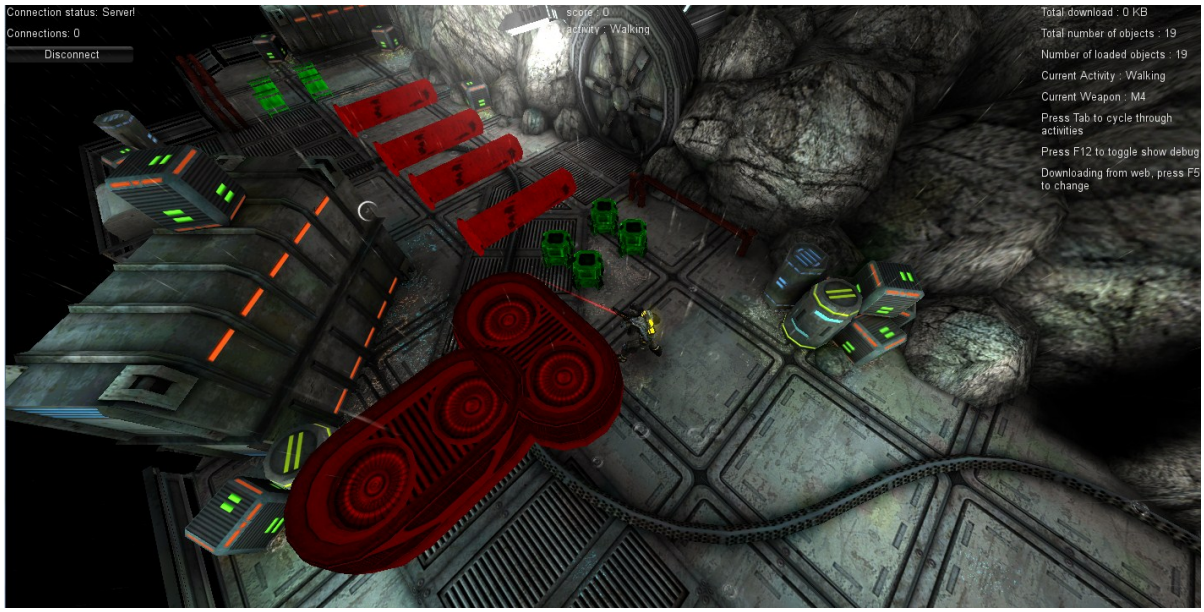


Figure 17: fully loaded scene, running on a PC/laptop, regardless of the current activity

Figure 17 is a screenshot of the laptop/PC (the server), which has the game in its full version (with all objects being loaded, regardless of the current activity). This can be used

as a reference to distinguish the differences between our two handheld devices, each using a different 3D streaming approach. Important objects for the "walking" activity are shown in red and important objects for "aiming" or "shooting" are shown in green.

Similar to what has been shown in Section 5.4.3, Figure 18 to Figure 21 show multiple snapshots of the same scene in our proof-of-concept game, using different methods. All of these snapshots have been taken from two different android-based mobile handheld devices, which we used as clients in our demo environment. Figure 18 and Figure 19 are the snapshots of our first client, in which other approaches (distance-based) have been used for 3D streaming. Figure 20 and Figure 21, on the other hand, are snapshots of our second client, in which our activity-centric streaming method was used. In each client, we show the same scene with two different activities, aiming and walking.

In the first client (Figure 18 and Figure 19), which is using the distance-based approach, no matter what the current activity is, only the objects that are within a certain distance of the player are streamed. Therefore, some objects that are irrelevant for the current activity of the player have also been loaded, while more important objects for the current activity that are far from the user have not been streamed. Figure 20 and Figure 21, on the other hand, show a mobile device using our activity-based streaming technique, and therefore, irrelevant objects (either close to or far from the player) have been skipped in each activity, and only the objects most relevant to the current activity have been streamed for the client. The loaded objects are shown red for better visual identification.

In this thesis, we either select or ignore an object completely for streaming. This was done because the focus of this work is to propose the idea of context-aware object selection and prioritization for streaming. However, as mentioned before, in reality, the server can also stream irrelevant objects in the scene in lower levels of detail (LOD).

Figures are shown on the next page.

Figure 18: the first mobile device, using a distance-based approach (walking).

Figure 19: the first mobile device, using a distance-based approach (aiming).

Figure 20: the second mobile device, using our activity-centric approach (walking).

Figure 21: the second mobile device in our activity-centric approach (aiming).

Chapter 6 - Performance Evaluation

Since the objective of our evaluation is to confirm that using our approach for object streaming is more efficient in terms of resources and also increases gameplay experience and quality, we have not focused on the MMOG aspect. Our test sessions are in single player mode because we want to show the functionality of our method and to compare it with other approaches. However, our conclusions are general and apply to the MMOG case as well because in both single player and MMOG mode, 3D objects are downloaded from a web-server. We therefore present two different test cases:

- A game with prioritization based on distance.
- A game with prioritization based on our context-aware model, considering the user-defined constraints and limitations.

Considering the goal of our evaluation, we define several metrics that can help us investigate the differences between the two test cases explained above. Each metric indicates a specific subject that is somehow related to the point that we are evaluating, and also explains gameplay experience, efficiency and success of the streaming technique. In the following section, we will explain our evaluation metrics.

The metrics we use to conduct our evaluation are:

- The total amount of data transmitted from the server to the client in each frame of gameplay.

- The percentage of importance of the loaded objects in the scene for all activities performed in the game.
- The ratio of the total importance of objects downloaded to total KB downloaded, which shows how much "importance" has been downloaded. This importance is the sum of the importance factors of all of loaded objects for different activities performed in a game session. For example, consider a scenario in which we have four different activities inside the game, but during a test session we only use two of them. The above-mentioned ratio, in this case, would be the total importance of the loaded objects for those two activities performed in the game, divided by the total size of the objects downloaded. This metric is good from a game designer's point of view.
- The total size required to obtain a specific percentage of importance in the scene.
- The ratio of the percentage of importance of the loaded objects in the scene (for the current activity) to the total importance of all objects available for that activity.
- The time during which the player has managed to hit each of the weapon warehouses. This information helps us to compare the total time required for game completion under different object streaming approaches and is an indication of the quality of the interactivity and of the game enjoyment experienced by the player.

In the following section, we explain all the above mentioned metrics in different graphs, and then we compare our proposed approach in many different aspects with distance-based approaches.

It is necessary to mention that each metric has been measured in a separate test session; this means our proof-of-concept game has been tested several times, each time measuring a specific metric explained before. This is due to the fact that each metric has its own set of requirements. For example, in order to assess the total amount of data transmission, the game environment must be explored completely and all available activities must be done in a game session; but for measuring the 5th metric for example, we don't need to have all activities done in our test session.

6.1 Results and Analysis

Figure 22 shows the amount of data transmitted during a game session. It is clear that the distance-based approach consumes more resources than our activity-based approach. Let us describe some of the details in the figure.

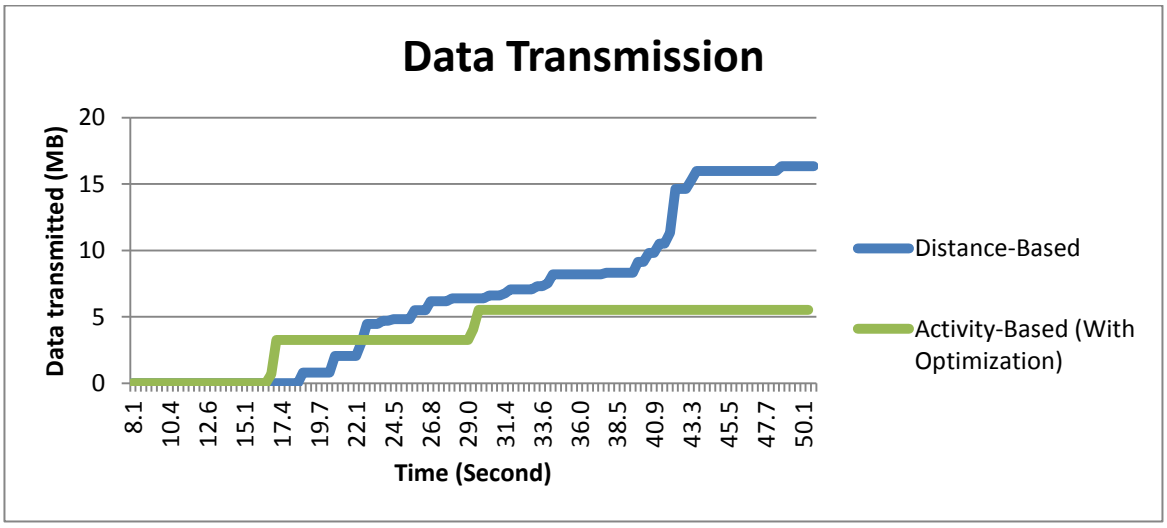


Figure 22: data transmission vs. time.

For our activity-based approach, around the 16th second, the player has switched to "shooting" mode, and this is why the amount of data transmitted has increased to ~3 MB in

our approach. This means that by going from “idle” to “shooting” status, ~3 MB of objects that are important for shooting are needed. From second 16 to second 29, the player was in shooting mode, and that is why there is no more data downloading reported during this time period. At second 29, the player has switched to the walking activity, so more objects required for walking have been transmitted to the player. At this point, all the important objects for these two activities, shooting and walking, have been transmitted (~6 MB). The player then continues the game without downloading anything.

For the distance-based approach, we played the game from second 0 and began to explore the entire scene during this time in order to let the server stream all the visible objects to the player. At around the 43rd second of the game, the exploration was finished, and all objects had been transmitted (~16 MB). The total data downloaded in the distance-based approach is, therefore, more than the total data downloaded in our proposed approach because in the distance-based approach, regardless of the current activity, all close objects will be streamed to the player whether the players needs them or not. Therefore, our approach utilizes the available resources much more efficiently.

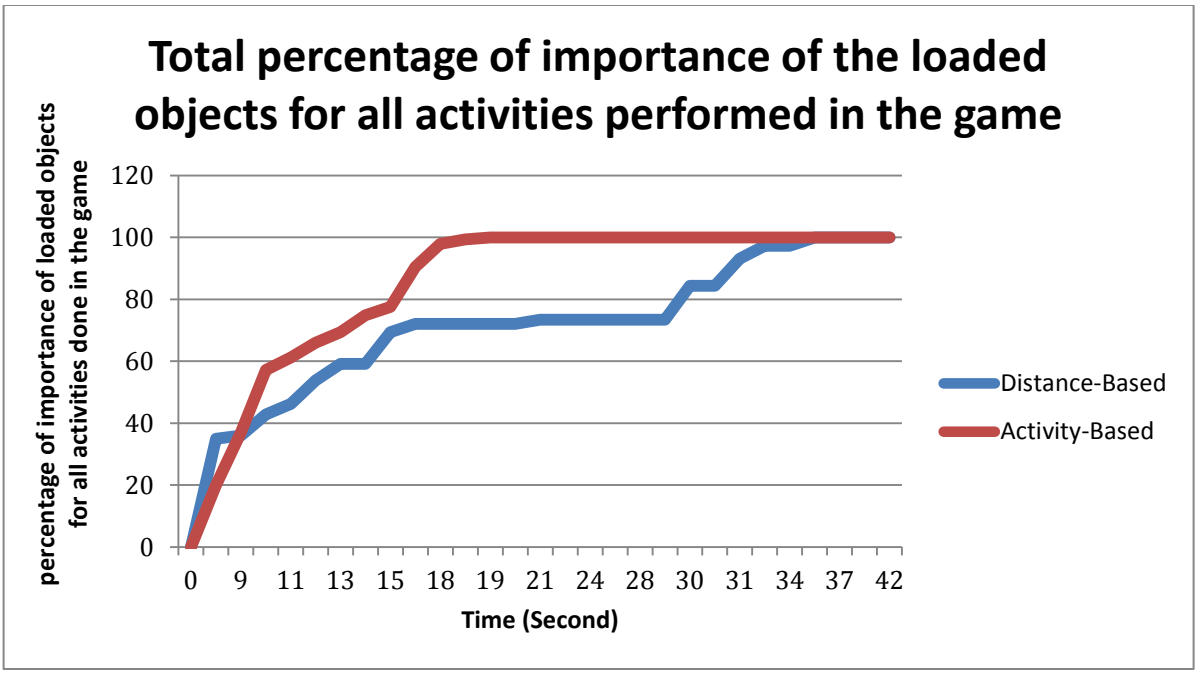


Figure 23: total percentage of importance of the loaded objects for all activities performed in the game.

We have another metric, which shows the total percentage of importance of the loaded objects in the scene for all activities performed in the game (not only for the current activity, but also for all activities performed). If this metric becomes 100%, it means that the loaded objects in the scene are all of the important objects available for all activities performed from the beginning of the game until the time when the metric becomes 100%. At that point, we can say that there is not actually any other relevant object that could potentially be loaded in that scene, based on the activities that have been done up until that moment.

Figure 23 draws the above-mentioned metric over time, and as can be seen from the graph, our activity-centric method has generally greater value in comparison to the distance-based approach. Also, in our approach, this metric reaches 100% much faster (~second 17)

as compared with the distance-based approach (~second 43). Our approach has significantly more value for this metric. Also, examine the 17th second and 43rd second in the previous graph and note the correlation.

It is necessary to indicate that for our activity-based approach, the above graph has been drawn only for a single activity, in this case shooting. Once the metric reaches 100%, it will remain the same value, even if we change our activity to walking, for example, because all the objects that have been loaded until that moment would be important for either shooting or walking.

The above-mentioned metric is a helpful way for game designers to see that the usage of our framework in their games would provide players with much more relevant and important objects during their gameplay.

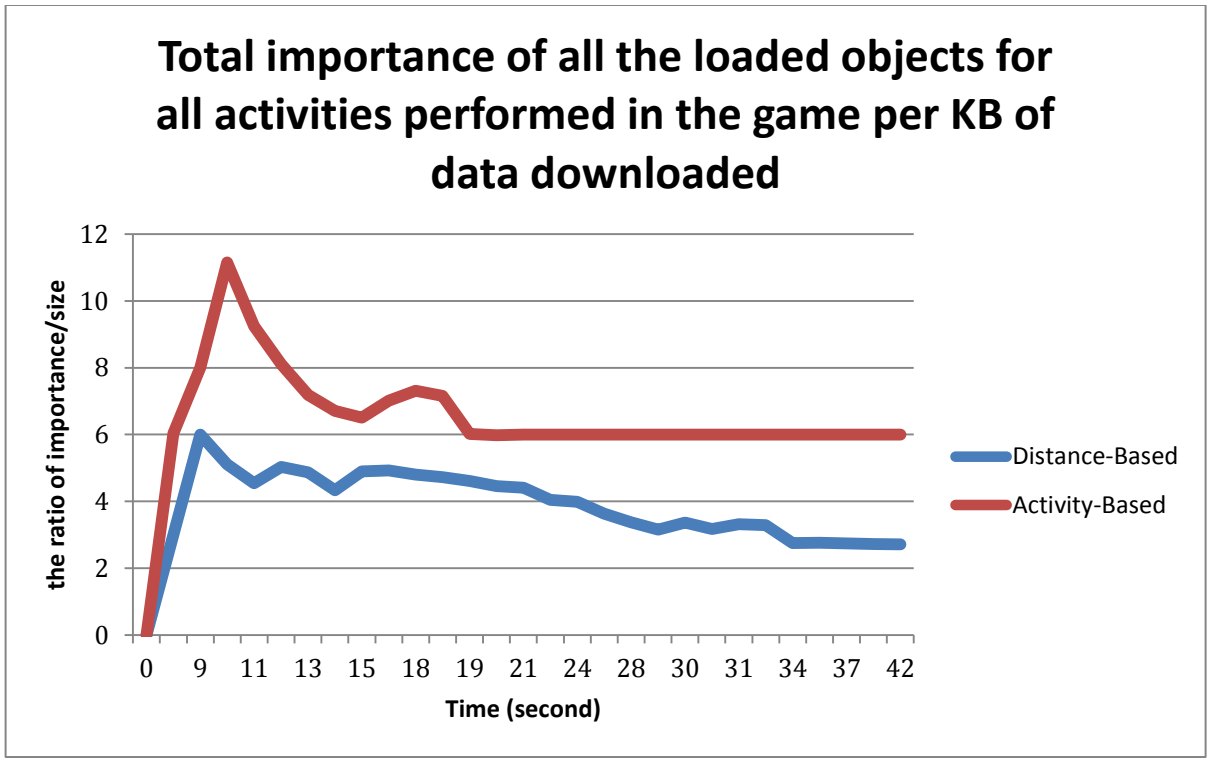


Figure 24: total importance of the loaded objects per KB downloaded.

Another metric that can be used by game designers to compare our approach with distance-based techniques is the amount of importance we would have per MB/KB of data downloaded from the server. In other words, what is the efficiency of data streaming in terms of importance? How much importance we will receive by downloading 1 KB of data from the server? How relevant is each KB of data downloaded from the server? The mentioned metric can answer all these questions.

Figure 24 draws the above-mentioned metric versus time during gameplay, and it shows that our activity-centric method always has greater values in comparison to a distance-based approach. This means that if we download a specific amount of data from the server, using the two different methods, our approach always has greater importance in total for

those loaded objects, or if we examine this from another perspective, it means that if we download a specific amount of importance from the server, using the two different methods, our approach can do so with much less data size than distance-based approaches.

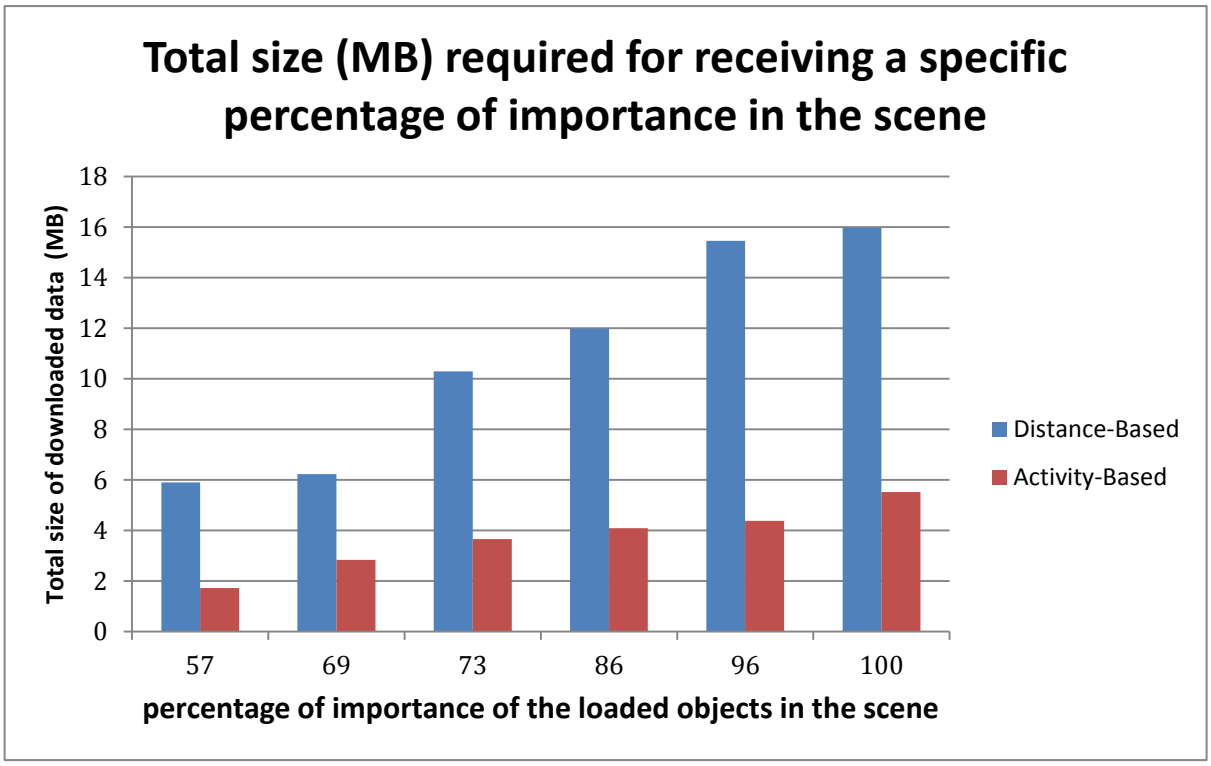


Figure 25: total size (MB) required for getting a specific percentage of importance in the scene.

In order to show the differences between our approach and distance-based approaches in terms of the amount of data downloading they require to obtain a specific amount of importance in a scene, we have drawn Figure 25.

As can be seen from the graph, different percentages of the importance of loaded objects in the scene have been shown on the horizontal axis, and the amount of data in MB required for each percentage has been shown on the vertical axis. Our approach always has a

significant difference with the distance-based approach, and it requires much less data size to be downloaded in order to obtain a specific percentage of importance in the scene.

For example, when the total importance of loaded objects in the scene is 57% of all the objects that are potentially important for the specific activities undertaken in the session, we can see which approach has downloaded less data in order to achieve this 57% importance level.

For the distance-based approach, however, this metric is greater because irrelevant objects have been transmitted during various activities, without increasing the total importance of the loaded objects in the scene. Those irrelevant objects have a “0” importance value for the activities performed in that session. Therefore, by downloading more data, there is no added importance of the loaded objects. For our activity-based approach, this ratio is much higher because we have skipped those cases in which the total percentage of the importance of the loaded objects is less than 50% and because we have mostly transmitted important objects for a given activity.

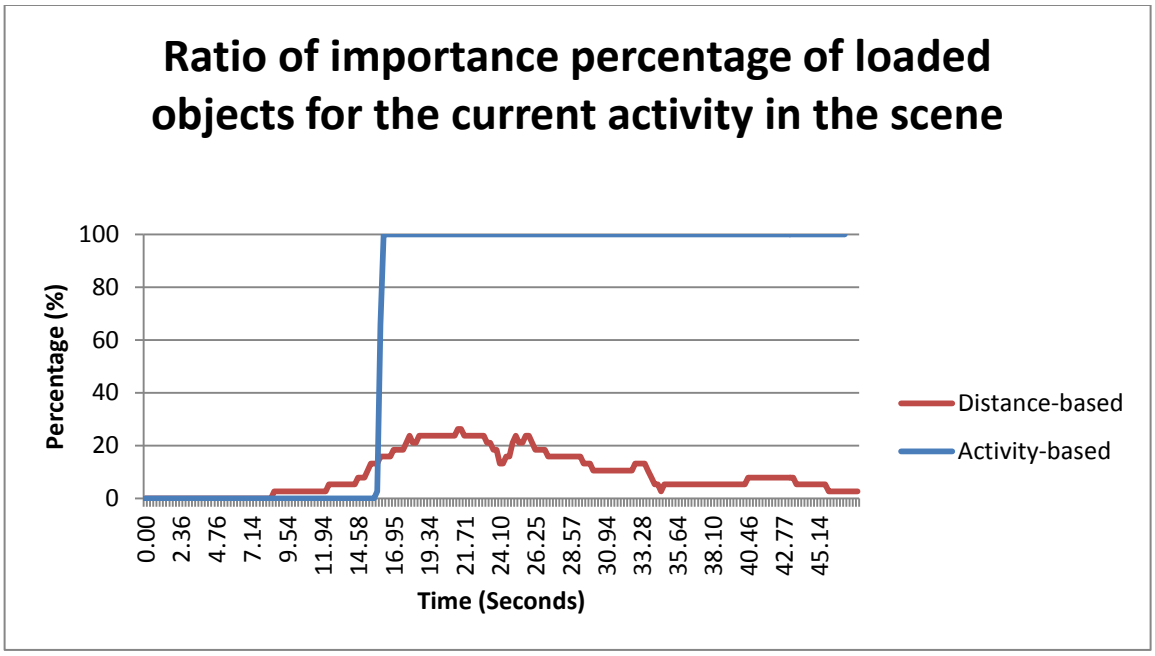


Figure 26: percentage of importance of the loaded objects in the scene for the current activity (activity-based).

Figure 26 show the relevance of the loaded objects for the current activity in each scene. This can be used to examine what percentage of our total important 3D-objects are still available for the current activity in each segment/scene.

In an activity-based approach, at around second 15 of gameplay, the player has switched from idle mode to an activity, either shooting or walking, and all the required objects for that activity in that specific scene/segment have been transmitted to the player. This means that all the loaded objects in the scene are important for the current activity and that the percentage of importance of the loaded objects is 100% for the current activity.

In other words, using our streaming technique guarantees that whatever is being streamed is 100% important for the current activity of the game. Therefore, no irrelevant or unimportant objects, from the game designer's point of view, will be streamed for the

player. Therefore, our approach has used the bandwidth in an efficient way by not wasting it on the streaming of irrelevant objects.

Please note that the Figure 26 is for a single activity, in this case shooting, and the metric has been reported for a single activity here in order to demonstrate the importance of the loaded objects for the current activity.

In a distance-based approach, the player explored the scene from the 0^{th} second up to 46^{th} second. At each second, a visible part of the scene was streamed to the player, no matter what the current activity was. This is why the percentage of importance of the loaded objects for the current activity is not high in this approach. Every single visible and close object will be streamed in this approach, and the selected objects are not necessarily important ones for the current activity.

Therefore, there is no guarantee that an under-streaming object will be an important one for the current activity of the player. As such, the bandwidth has been used in an inefficient way.

It is worth mentioning that an object could be "important" for any "aspect" of a specific activity. Different aspects of an activity were explained in Section 4.2. They include, but are not limited to:

- An object could be helpful for the "task accomplishment" aspect of an activity.
- An object could help the player in his game experience by providing him with a better "immersion" in the game while he is involved with an activity.

- An object could be useful for the "visual quality" of a specific activity.
- An object could be effective in the "orientation" of the player involving a specific activity.

As explained previously, *importance factors* have been designed in such a way that all different aspects of an activity can be considered in the value of an importance factor.

The above figure (Figure 26) proves that our method always has a better rate regarding this metric.

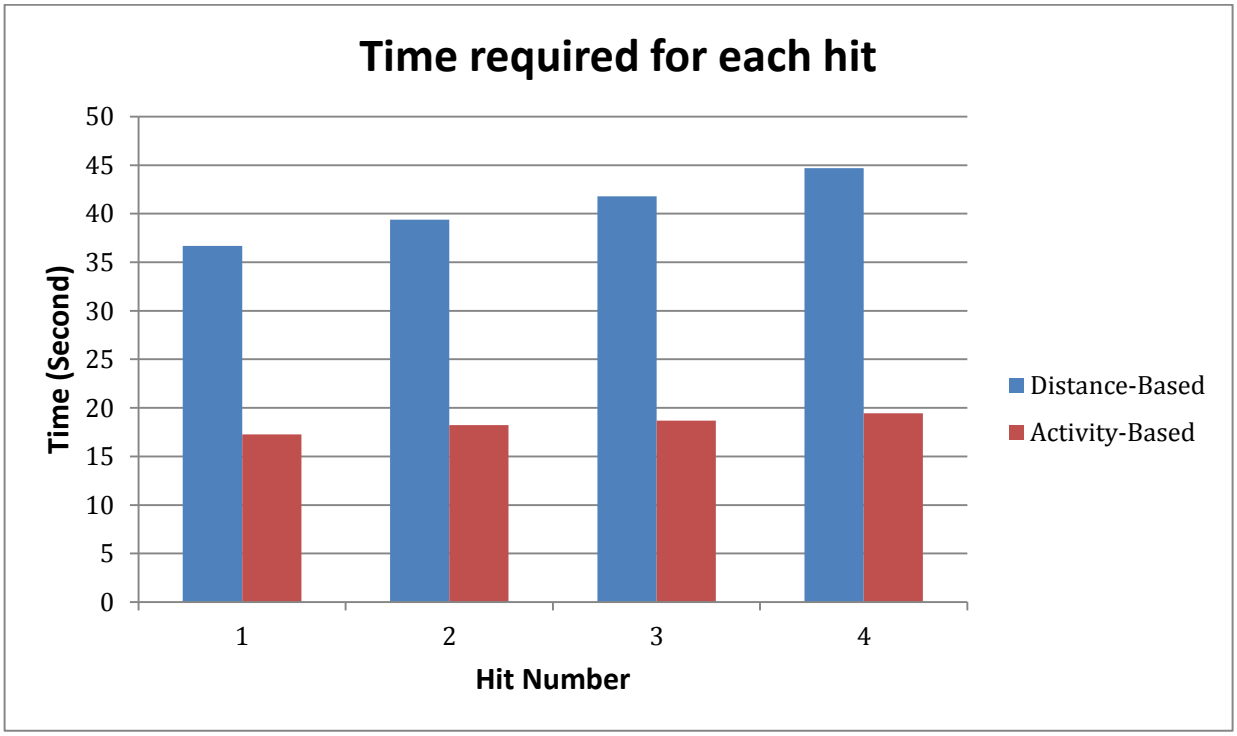


Figure 27: time required for each warehouse hit.

We have set the game such that it will end if a player manages to hit four warehouses. In our approach (activity-based), when we are in "shooting" mode, our algorithm will stream

all the important objects required for shooting to the player, and as a result, the player is able to shoot the objects faster than he would under the distance-based approach. This is why less time is required to hit four warehouses in an activity-based approach, but in the distance-based approach, the player needs to explore the scene more fully and get close enough to the objects to be able to see and hit them. As a result, he requires a longer time to finish the game. This is confirmed by our evaluations, as can be seen in Figure 27.

Chapter 7 - Conclusion, Discussion & Future Work

7.1 Conclusion

Taking into account the increasing popularity of online mobile gaming popularity and the potential application of online 3D virtual environments in a variety of scenarios, it would be advantageous to have these environments work smoothly and quickly, providing high levels of interactivity for their users. Therefore, the domain of online mobile gaming is fertile ground for novel approaches and new technologies.

This thesis presents an architecture for activity-centric and context-aware 3D object selection, prioritization, and streaming for networked virtual environments and multiplayer games. Our architecture uses the game context and streams only the most relevant objects for players in each frame of gameplay. We tested our architecture by implementing a proof-of-concept game and evaluating it through different metrics. Our evaluation results show promise. Our approach not only reduces resource utilization, but also increases the quality of interactivity and gameplay as experienced by the player. Therefore, this architecture can be pursued further in order to provide a basis for the live deployment of online mobile game frameworks.

7.2 Discussion

It is necessary to mention that in this thesis, we either select/choose an object for streaming or tag it as irrelevant/unimportant and, therefore, ignore it for streaming purposes. This was done because the focus of this work was to propose a novel approach to object selection, object prioritization based on game context, and object streaming. However, in reality, the 3D streaming technique used does not need to ignore irrelevant objects. Instead, it can stream those objects in lower levels of detail (LODs), while more important objects will be streamed with higher LODs as required. This can help the player to have a better visual impression of the environment by having a glance at all objects at the same time.

In our testing and analysis, we find that depending on the object arrangements and their positions in the scene, on the number of total activities in the game, and on whether we have used all the available activities or not, we might have some scenarios in which the distance-based approach would have similar results to our context-aware approach. As an example, in a scenario in which all the warehouses were close to the player, the distance-based approach and our context-aware approach would have similar results. These are the worst cases for our approach, yet our approach has the same result as the distance-based approach. In general, our algorithm always performs better or, at the very least, the same as distance-based approaches. The superiority of our algorithm will be more obvious in scenarios with a large number of activities, many objects, and many downloadable files with large sizes. The larger the level, the more efficient our algorithm will be.

7.3 Avenues for Future Research

While the presented design accomplishes activity-centric and context-aware 3D streaming for online mobile games, assumptions have been made with regard to having a complete game context model with all the importance factors for all the objects and activities being set in advance by game designers as well as having a module inside the game responsible for recognizing the current activity of the player. These two pre-requisites are outside the scope of this particular work, but in order to conduct our research and evaluate our proposed object streaming technique, we implemented a proof-of-concept game with importance factors being set and explained how game designers could use a similar approach to their games to design them in a way that meets the requirements of our framework.

Therefore, one avenue that can be explored by future research is developing a mechanism that will allow different game objects to have their importance factors set automatically during the game. This would result in a self-organizing 3D objects approach. In this approach, during initial gameplay, different 3D objects will learn how important they are for the current activity, and their importance factor will then be set automatically by the framework and not by game designers in advance. This requires applying a specific method in order for objects to become intelligent in such a way that they learn from gameplay.

Moreover, as another avenue of future work, the battery usage in different streaming methods can be measured precisely to study the exact amount of battery saving caused by our method. One possible approach for this is to measure the battery usage of handheld devices caused by downloading of each 3D object in the scene. This can be done in an

experimental way, by running the game several times and measuring how much battery is consumed after downloading of that specific object. Therefore, we can roughly figure out how much battery is needed for the entire game session, and we can then compare different 3D streaming methods with each other in terms of their battery usage.

References

[1] Global Cell Phone Use at 50 Percent (November 29th 2007). [Online]. http://www.reuters.com/article/2007/11/29/us-cellphones-world-idUSL2917209520071129

[2] Mobile Phones Used More than PCs to Browse the Internet (March 31st, 2009). [Online]. http://news.softpedia.com/news/Mobile-Phones-Used-More-than-PCs-to-Browse-the-Internet-108254.shtml

[3] Wikipedia. [Online]. http://en.wikipedia.org/wiki/4g

[4] CFP Special Issue on 3D Mobile Multimedia, ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP). [Online]. http://www.site.uottawa.ca/~shervin/3dmmsi/

[5] Wikipedia. [Online]. http://en.wikipedia.org/wiki/Web3D

[6] Entertainment Software Association of Canada. [Online]. http://www.theesa.ca

[7] pcw-tech. [Online]. http://www.pwc-tech.ca/pwctechnologyteam/24f6ca4d-693b-4bef-abbf-9a5ce46889af/Canada-s-Video-Game-Industry-in-2011--Infographic------Techvibes-com

[8] CFP Special Issue on Network and Systems Support for Games in Springer's Multimedia Systems Journal (MMSJ). [Online]. http://www.site.uottawa.ca/~shervin/netgamessi/

[9] comeScore Inc. (January 28, 2009). [Online]. http://www.comscore.com

[10] Social Networking On-The-Go: U.S. Mobile Social Media Audience Grows 37 Percent

in the Past Year. [Online]. http://www.comscore.com

[11] Second Life. [Online]. www.secondlife.com

[12] M. Chan, J. Jian, C. Hung, and W. Tsang Ooi, "Group-Based Peer-to-Peer 3D Streaming Authentication," in *15th International Conference on Parallel and Distributed Systems*, Shenzhen, China, December 2009.

[13] J. Jiang, B. Chen and S. Hu, "Peer-to-Peer 3D Streaming," *IEEE Internet Computing*, vol. 14, no. 2, pp. 54-61, March/April 2010.

[14] W. B. Li Frederick, W. H. Lau Rynson, K. Danny, "GameOD: an internet based game-on-demand framework," in *Proceedings of the ACM symposium on Virtual reality software and technology (VRST'04)*, Hong Kong, November 2004, pp. 129 - 136.

[15] "Web Player Streaming", Unity Manual, Unity Technologies. [Online]. http://unity3d.com/support/documentation/Manual/Web%20Player%20Streaming.html

[16] H. R. Maamar, E. Petriu and A. Boukerche, "MOSAIC - A Mobile Peer-to-Peer Networks-based 3D Streaming Supplying Partner Protocol," in *14th IEEE/ACM Symposium on Distributed Simulation and Real-Time Applications*, Fairfax, Virginia USA, October 2010.

[17] H. Rahimi, A. A. Nazari Shirehjini, and S. Shirmohammadi, "Activity-Centric Streaming of Virtual Environments and Games to Mobile Devices," in *Proceedings of IEEE Symposium on Haptic Audio Visual Environments and Games (HAVE 2011)*, Qinhuangdao, Hebei, China, October 14-17 2011.

[18] H. Rahimi, A. A. Nazari Shirehjini, and S. Shirmohammadi, "Context-Aware 3D Object Streaming for Mobile Games," in *Proceedings of ACM/IEEE Network and Systems Support for Games (NetGames 2011)*, Ottawa, Ontario, Canada, October 6-7 2011.

[19] H. Rahimi, A. A. Nazari Shirehjini, and S. Shirmohammadi, "Context-Aware Prioritized Game Streaming," in *Proc. of Workshop on Interactive Ambient Intelligence Multimedia Environments, in Proceedings of IEEE International Conference on Multimedia & Expo (ICME 2011)*, Barcelona, Spain, July 11-15, 2011.

[20] H. Rahimi, S. Ratti, A. A. Nazari Shirehjini, and S. Shirmohammadi, "Unsynchronized Multiplayer Networked Games: Feasibility with Time Rewind," in *Proceedings of ACM/IEEE Network and Systems supports for Games (NetGames 2010)*, Taipei, Taiwan, November 16-17, 2010.

[21] J. H. Clark, "Hierarchical Geometric Models for Visible Surface Algorithms," *Communications of the ACM*, vol. 19, no. 10, pp. 547-554, 1976.

[22] M. Reddy, J. D. Cohen, A. Varshney, B. Watson, R. Huebner, and D. Luebke, "Level of Detail for 3D Graphics", ISBN: 1-55860-838-9, 2003rd ed.

[23] J. K. Yan, "Advances in Computer Generated Imagery for Flight Simulation," *IEEE Computer Graphics and Applications*, vol. 5, pp. 37-51, 1985.

[24] V. Miliano, "Unreality: Application of a 3D Game Engine to Enhance the Design, Visualization and Presentation of Commercial Real Estate," in *Proceedings of VSMM '99*, 1999, pp. 508-513.

[25] T. A. Funkhouser and C. H. Sequin, "Adaptive Display Algorithm for Interactive Frame Rates During Visualization of Complex Virtual Environments," in *Proceedings of SIGGRAPH'93*, 1993, pp. 247-254.

[26] S. Hu, "Peer-to-Peer 3D Streaming," National Central University, Taiwan, PhD Thesis November 2009.

[27] E. Teler and D. Lischinski, "Streaming of Complex 3D Scenes for Remote Walkthroughs," in *EUROGRAPHICS*, 2001, pp. 17-25.

[28] S. Rusinkiewicz and M. Levoy, "Streaming QSplat: A Viewer for Networked Visualization of Large, Dense Models," in *Proc. Symp. Interactive 3D Graphics*, 2001, pp. 63-69.

[29] S. Hu, "A Case for 3D Streaming on Peer-to-Peer Networks," in *Proceedings of the eleventh international conference on 3D web technology (Web3D)*, 2006, pp. 57-63.

[30] H. Hoppe, "Progressive Meshes," in *Proceedings of SIGGRAPH*, 1996, pp. 99-108.

[31] S. Lee, L. Kobbelt, and J. Kim, "View-dependent Streaming of Progressive Meshes," in *Proceedings of Shape Modeling Applications*, 2004, pp. 209-220.

[32] B. Li, K. Nahrstedt, and Y. Cui, "ostream: Asynchronous Streaming Multicast in Application-Layer Overlay Networks," *IEEE JSAC*, vol. 22, no. 1, pp. 91-106, 2004.

[33] W. H. Lau Rynson, H. Leong, A. Si, and J. Chim, "Cyberwalk: A Web-based Distributed Virtual Walkthrough Environment," *IEEE Transaction on Multimedia*, vol. 5, no. 4, pp. 503-515, 2003.

[34] D. Cohen-Or, R. Ironi, T. Zvi, and E. Fogel, "A Web Architecture for Progressive Delivery of 3D Content," in *Proceedings of ACM Web3D*, 2001, pp. 35-41.

[35] J. C. Xia and A. Varshney, "Dynamic View-dependent Simplification for Polygonal Models," in *Proceedings of IEEE Visualization'96*, 1996, pp. 327-334.

[36] J-E. Marvie and K. Bouatouch, "Remote rendering of massively textured 3d scenes through progressive texture maps," in *Proceedings of 3rd IASTED Conference VIIP*, 2003, pp. 756-761.

[37] K. Nishitani, T. Cornish, T. Naka, S. Asahara, and T. Hijiri, "A spatial hierarchical compression method for 3d streaming animation," in *Proceedings of ACM VRML*, 2000, pp. 95-101.

[38] I. Soetebier, H. Birthelmer, and J. Sahm, "Eficient representation and streaming of 3d scenes," *Computers & Graphics*, vol. 28, no. 1, pp. 15-24, 2004.

[39] T. Borich and I. Suen, "3D Visualization in Community-based Planning," *Journal of Extension*, vol. 42, no. 6, 2004.

[40] O. Hagsan, "Interactive Multiuser VEs in the DIVE System," *IEEE Multimedia*, vol. 3, no. 1, pp. 30-39, 1996.

[41] A. Johnson, C. Vasilakis, T. DeFanti, and J. Leigh, "Multi-perspective Collaborative Design in Persistent Networked Virtual Environments," in *Proc. IEEE VRAIS*, 1996, pp. 253-260.

[42] D. Anderson, J. Barrus, D. Brogan, M. Casey, S. McKeown, T. Nitta, I. Sterns, W. Yerazunis, and R. Waters, "Diamond Park and Spline: A Social Virtual Reality System with 3D Animation, Spoken Interaction, and Runtime Modifiability," *Presence*, vol. 6, no. 4, pp. 461-480, 1997.

[43] K. Saar, "VIRTUS: A Collaborative Multi-User Platform," in *Proc. VRML*, 1999, pp. 141-152.

[44] M. Zyda, D. Pratt, R. Mackey, and J. Falby, "NPSNET: Hierarchical Data Structures for Real-Time Three-Dimensional Visual Simulation," *Computers & Graphics*, vol. 17, no. 1, pp. 65-69, 1993.

[45] M. Zyda, D. Pratt, D. Brutzman, P. Barham, and M. Macedonia, "Exploiting Reality with Multicast Groups: A Network Architecture for Large-scale Virtual Environments," in *Proc. IEEE VRAIS*, 1995, pp. 38-45.

[46] G. Singh, A. Mitchell, P. Kumar, K. McGhee, and T. Das, "NetEffect: Network Architecture for Large-scale Multi-user Virtual World," in *Proc. ACM VRST*, 1997, pp. 157-163.

[47] W. H. Lau Rynson, D. Kilis, W. F. Li Lewis, and W. B. Li Frederic, "Game-on-demand: An online game engine based on geometry streaming," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, vol. 7, no. 3, August 2011.

[48] R. Cavagna, C. Bouville, and J. Royan, "P2P network for very large virtual environment," in *Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST'06). ACM*, New York, 2006, pp. 269-276.

[49] C. Chang and S. Ger, "Enhancing 3D Graphics on Mobile Devices by Image-Based Rendering," in *Proceedings of the IEEE Pacific Rim Conference on Multimedia*, Los Alamitos, CA, 2002, pp. 1105-1111.

[50] R. W. N. Pazzi, A. Boukerche, and T. Huang, "Implementation, measurement, and analysis of an image-based virtual environment streaming protocol for wireless mobile devices," *IEEE Transaction of Instrumentation and Measurement (TIM)*, vol. 57, no. 9, pp. 1894-1907, 2008.

[51] K. Mayer-Patel and D. Gotz, " Scalable, Adaptive Streaming for Nonlinear Media," *IEEE Multimedia*, vol. 14, no. 3, pp. 68-83, 2007.

[52] Wikipedia. [Online]. http://en.wikipedia.org/wiki/Gaming_on_demand

[53] OnLive. [Online]. http://www.onlive.com/

[54] Gaikai. [Online]. http://www.gaikai.com/

[55] GamesBeat. [Online]. http://venturebeat.com/2011/02/28/gaikai-launches-game-streaming-beta-tests-for-four-electronic-arts-games/

[56] BBC News. [Online]. http://www.bbc.co.uk/news/technology-14481101

[57] Wikipedia, Ontology (information science). [Online].

http://en.wikipedia.org/wiki/Ontology_(information_science)

[58] Battle Field Heroes. [Online]. http://www.battlefieldheroes.com

[59] B. Hariri, S. Shirmohammadi, and M.R. Pakravan, "Hierarchical HMM Model and Measurement of Online Gaming Traffic Patterns," in *Proc. IEEE International Instrumentation and Measurement Technology Conference*, Victoria, Canada, May 12-15 2008, pp. 2195 – 2200.

[60] R. Rao, S. Vrudhula, and D. N. Rakhmatov, "Battery Modeling for Energy-Aware System Design," *Computer*, vol. 36, no. 12, pp. 77-87, December 2003.

[61] M. L. Fisher, "The lagrangian relaxation method for solving integer programming problems," *Management science*, vol. 27, no. 1, pp. 1-18, January 1981.

[62] IBM. (2011, October) IBM ILOG CPLEX Optimizer. [Online]. http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/

[63] Mathworks. (2011, October) Optimization Toolbox. [Online]. http://www.mathworks.com/products/optimization/

[64] Android. [Online]. http://www.android.com/

[65] Unity3d Game Development Tool. [Online]. http://unity3d.com/

[66] Photon Socket Server. [Online]. http://www.exitgames.com/Photon

[67] pwc-tech. [Online]. http://www.pwc-tech.ca/pwctechnologyteam/24f6ca4d-693b-4bef-abbf-9a5ce46889af/Canada-s-Video-Game-Industry-in-2011--Infographic------Techvibes-com

[68] Wikipedia. [Online]. http://en.wikipedia.org/wiki/Game_design

[69] Purdue University. [Online]. http://www.e-

games.tech.purdue.edu/GameDevProcess.asp

Appendix A – Canadian Gaming Industry

Figure 28: video Game Industry in Canada ([67])

Appendix B – Game Design Process

Game design is one of the subsets of game development in which the contents of a game, as well as its rules will be designed in the *pre-production* stage and the design of gameplay, environment, storyline, and characters will be done during *production* stage [68]. Figure 29 ([69]) shows different stages of the game development process.

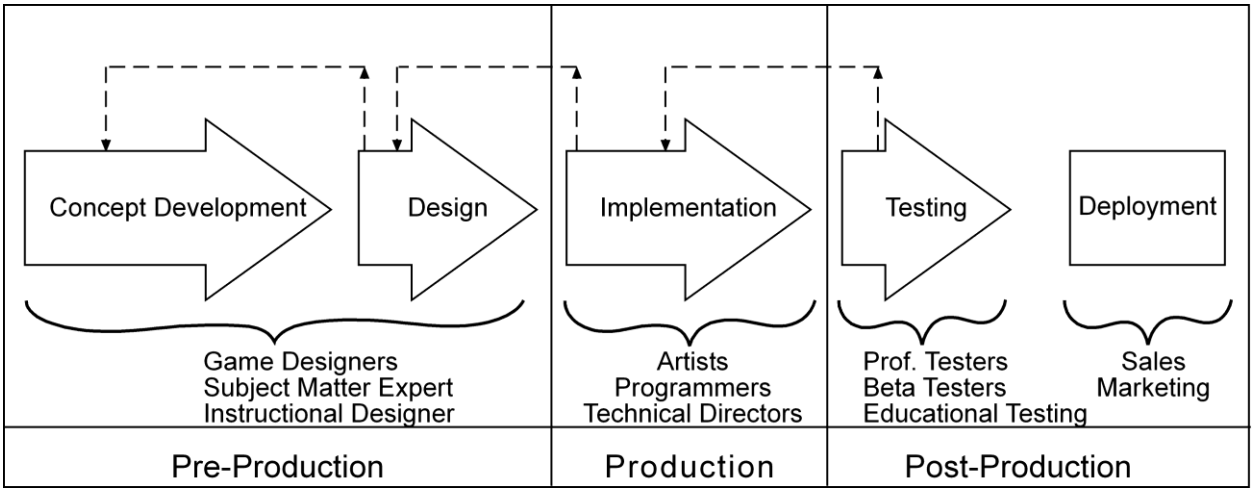


Figure 29: game development stages [69]

The responsibility of the game designers is to define the puzzles, rules, and rewards that will be entertaining and challenging to the player [69]. After all these steps, game designers should provide the production team with a roadmap that is specific enough for them to create a game [69]. Like a writer or director in a film, a game designer is aware of all of the assets created by the art team and the activities programmed by the software engineers [69]. As such, it is not too much overhead for them to define the importance of ach object/asset for each activity doable in their game.

Video games are usually developed by a team composed of programmers, artists, game designers and technical directors, as shown Figure 30 (derived from [69]).

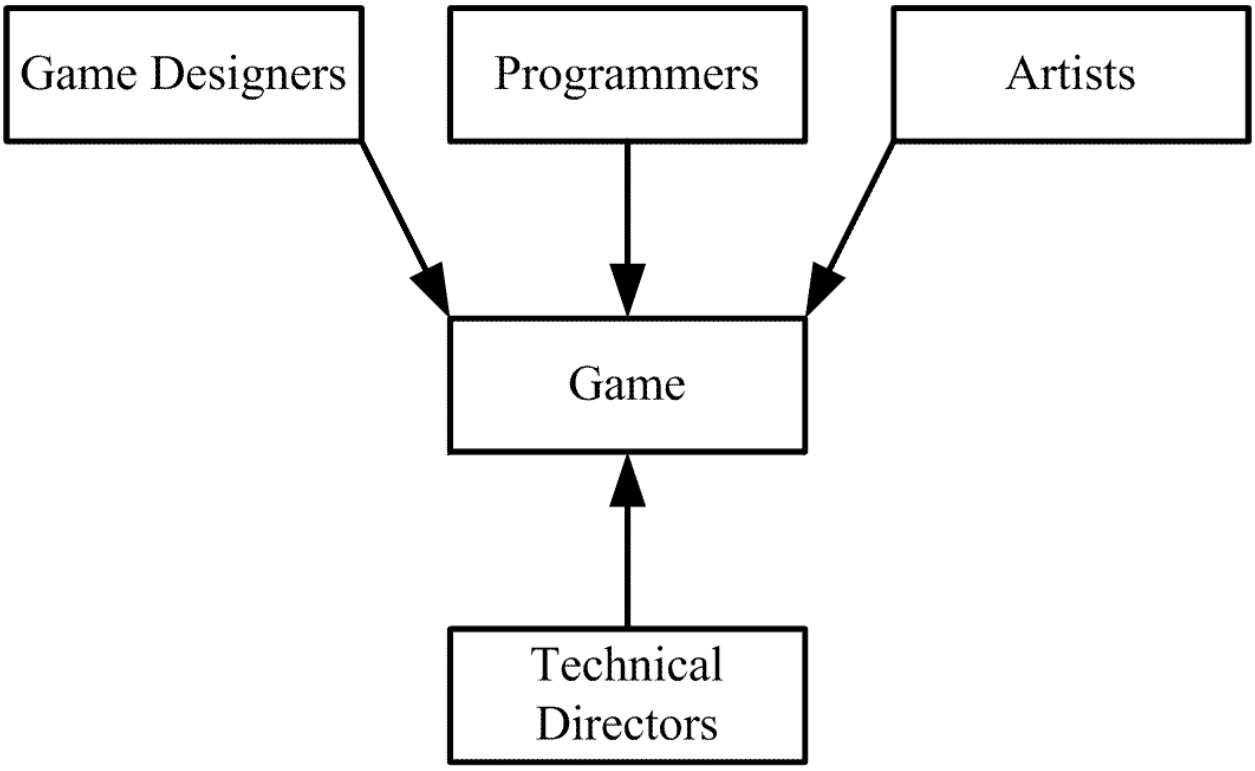


Figure 30: different roles in game development process

Appendix C – A Complete List of Objects Used In Our Implementation

OBJECT NAME	OBJECT SIZE IN KB
barrel_a	216.1777
barrel_b	214.959
barrel_c	218.3408
barrel_d	217.8721
boot	168.1914
cardboard_box_c	105.3955
cardboard_box_c1	658.6426
concrete_beam_large	670.3379
concrete_beam_medium_a	670.0781
concrete_brick	108.875
concrete_wall_a	667.3135
concrete_wall_g	659.373
fence_fixed	797.7041
fence_fixed_b	801.0801
gate	1029.243
grid	795.0986
guard_rail_medium	671.0322
guard_rail_small	672.2148
hesamKalleh	2558.167
hesamOutPost	1205.915
house_new	336.7031
ink_can	118.668
interlaced_wood_a	671.9082
iron_beam_b	659.9961
roofing_a	659.2275
tire_a	237.5576
tire_b	161.4629