

Activity-Centric Streaming of Virtual Environments and Games to Mobile Devices

Hesam Rahimi, Ali Asghar Nazari Shirehjini, Shervin Shirmohammadi

Distributed and Collaborative Virtual Environments Research Laboratory (DISCOVER Lab)

School of Information Technology and Engineering, University of Ottawa

800 King Edward Ave. Ottawa, Ontario, K1N 6N5, Canada

{ hrahimi | ali.nazari | shervin }@discover.uottawa.ca

Abstract— As mobile devices still have limited battery, processing power, memory, and display size, compared to their PC counterparts, they cannot yet execute Virtual Environment (VE) and Online Gaming applications with the same fidelity and quality as their PC counterpart. In response, we have recently witnessed some research with the goal of real-time delivery of VE and multiplayer game content specifically to fit mobile devices' limitations. In this paper, we present a novel approach to tackle the streaming of 3D objects for such environments. Our goal is to improve the real-time response and interactivity of networked VEs and games through an efficient context-aware 3D object selection and prioritization scheme, before streaming those objects over the network. In our architecture, we take advantage of game context for culling and streaming only the most relevant objects in each frame of gameplay and stream them from the server to a client. Our evaluations show that this technique not only leads to better performance in general, but also increases gameplay experience by helping the player to achieve a higher score in comparison with traditional approaches.

Keywords- 3D streaming, context-awareness, virtual environments, multiplayer games, network games, mobile games

I. INTRODUCTION

3D virtual environments and games are becoming more realistic-looking and detailed, with higher resolution and near-reality texture maps. To achieve this high fidelity, a considerable amount of information needs to be both processed (for logic and rendering) and delivered over the network. This has been one of the main factors pushing personal computers (PC) in terms of computing power and display capabilities. As a result, PCs have responded by doubling their computing power every 2 years, also known as Moore's law, and have become very powerful computing platforms. While mobile devices have also improved in terms of computing power and display capabilities, they are still far behind their PC counterparts. As such, delivering the content of the above mentioned VEs and games is still challenging for mobile devices, given their limitations of network bandwidth, processing power, memory size, and display size. Due to these restrictions, high fidelity 3D applications must be converted to lower quality and less detailed versions for their mobile counterparts. This is an unavoidable fact, until mobile devices will have the same computing power and display capability as PCs, which will not happen for the next few years. Meanwhile, researchers have proposed various approaches to achieve the

said conversion of VEs and games to lower quality versions for mobile devices. In this paper, we use game context to offer a new approach towards the progressive delivery of 3D models for mobile games over the network. By doing so, we can address the challenges of limited bandwidth and resources that mobile games are faced with. In addition, our approach is complementary to existing methods, such as region-based rendering, area of interest management, progressive streaming, and data reduction by occlusion and other techniques. We proposed a preliminary version of this approach in our previous work [9], which we continue and further complete in this paper.

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 as 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 [12]. 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 feature based on which content is more important for the player at this point in time in the game; i.e., our approach is context based. In addition, in such environments there is usually a need to load new 3D models, textures, and materials (let us call them assets) at runtime. Sometimes after releasing the game, its designers might need to update the models without rebuilding the whole project. In this case some assets which were not originally built into the game must now be streamed, such as additional weapons, environments, characters, or even full levels. Delivery of those assets must also be done dynamically and interactively. Consequently, we can load assets on demand and we can stream in our game world different items and 3D objects come into our view.

The remainder of this paper is organized as follows. The next section describes related works. Our framework is described in Section 3. Section 4 describes our implementation followed by Section 5 that presents our Evaluation results and analysis. Finally, discussion and future works, and conclusions are each described in separate sections.

II. RELATED WORK

Geometry transmission [1] is among the main methods proposed for large virtual environments, where only a specific region of the environment might be visited by the user.

Therefore, the client receives only the geometry information of that specific region to reduce bandwidth usage and computing power. Virtual environments described in [2], [3], [4], and [5] are based on this region-based approach. Some other works like [6] and [7] propose an interest-based approach where an area of interest (AOI) will be used to determine object visibility. However, none all of these methods are concerned with whether every visible object is in fact of equal interest and importance to the player at the moment; they simply stream all visible objects with the same priority and quality. Other works such as that in [8], address this issue to some extent by focusing on prioritization of objects in a given scene based on object scope and viewer scope. The idea here is to transmit only the objects whose scope has some overlap with the player’s viewing scope. While this improves the performance, priority is simply defined as whether or not an object’s scope is inside the viewer’s scope; i.e., priority is defined as distance.

Our work, which is complementary to the above and can be used in conjunction with them, is based on the idea of context-aware prioritized game streaming. As explained in details in our previous paper [9], our method uses an *importance factor* defined for every object for a given game activity. We will set the priorities of the objects based on their importance factors and stream the most important objects to the player first. In other words, taking into account the current activity undertaken by a player, different objects will be sorted based on their priority for that activity, and then they will be streamed to the player dynamically while the player is playing the game. This is described in details next.

III. PROPOSED FRAMEWORK

Similar to previous work, our approach also 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 we previously explained in [9], this small portion contains 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 that are based only on distance or region, and provide a better visual quality for closer objects, we focus on the context under 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, no matter how close or far the enemy is in the visible range. Distance-based approaches on the other hand would simply render with higher quality whatever is closer to the player. As a consequence, many objects are sent to the player at a higher quality than required or without having a semantic relevance for 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 only at a lower quality because they are far, even though they might be highly important [9].

In our approach, the importance of an object is determined

by an algorithm which considers context parameters relevant to the activities inside the game, and uses this information to select and prioritize objects for progressive streaming purposes. This should assure 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.

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. In other words, game designers should list all the possible activities and all the available 3D objects inside the environment, and they need to determine exactly to what extent each object is important for each activity. As discussed in [9], this importance could be defined in many aspects of the gameplay; i.e., for a given activity and a given object, we can have specific importance factor for each aspect of gameplay. For example, an object could be very important for the “task accomplishment” aspect of an activity (like “weapon” is much needed for “shooting”), while the same object is less important for the “visual quality” aspect of the same activity. As reported in [9], the different aspects are “immersion”, “visual quality”, “task accomplishment” and “orientation”, shown in Table 1. 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 as we discussed in [9]. Basically the cost/benefit ratio for spending time to design such importance factors are low enough that makes this feasible for game designers. Also they need to do this just one time, at the time of game scenario designing.

TABLE 1 : IMPORTANCE FACTORS (REPRODUCED FROM [9])

Activity/Object	%	weapon	teammate	enemy	wall	tree
Navigation	Immersion	α_w	α_t	α_e	α_s	α_r
	Visual quality	β_w	β_t	β_e	β_s	β_r
	Task accomplishment	γ_w	γ_t	γ_e	γ_s	γ_r
	Orientation	θ_w	θ_t	θ_e	θ_s	θ_r
Shooting	Immersion	α_w	α_t	α_e	α_s	α_r
	Visual quality	β_w	β_t	β_e	β_s	β_r
	Task accomplishment	γ_w	γ_t	γ_e	γ_s	γ_r
	Orientation	θ_w	θ_t	θ_e	θ_s	θ_r

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 (α) unit important for a given activity while object (B) is ($n \times \alpha$) unit important for the same activity. To normalize these factors, we must first find the highest importance and change it to 1, and then we divide all other importance factors by this highest one. Therefore, we will have numbers between 0 and 1, where

1 indicates the highest importance and 0 shows completely no importance. After normalization of the importance factors for different objects, we prioritize all the objects based on their importance.

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:

1. 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 no limitation on the client’s download rate, which might not be the case for many mobile devices.
2. Depending on the type of internet connection in use, we could define a maximum download rate, set by the service provider or the mobile client. Our framework then guaranties that maximum download rate will not be violated, by streaming objects from the very beginning of the prioritized list and continuing until it reaches the maximum download rate. For example, if a player’s mobile client cannot download more than 100 Kilobytes per second while playing the game, our system ensures no more than 100 kilobytes worth of objects from the prioritized list is sent every second.
3. We can have several limitations and do 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 rate, a player might have limitations in the battery life of the mobile device, and so the game must be played while the maximum energy usage does not go beyond 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.

IV. IMPLEMENTATION

For implementation, we have used the Unity 3 game development tool and engine [10]. This engine supports rendering, lighting, audio, physics, terrains, networking, and programming which are among the most important features for 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 other game aspects mentioned before.

A. Proof-of-Concept

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

B. 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 [11]. Using a free version of Photon, we can create multiple “game groups” consisting of up to twenty players per group. Players who are in a same game group can then play together in the same level and have interaction with each other. The Photon socket-server utilizes the UDP protocol for ultimate speed and performance which is fast and reliable, though also lossy and without delivery guarantee.

C. Gameplay Mechanics

In our proof-of-concept game, players can play it in a local area network or via the Internet. To make the Internet-based gameplay achievable, we have put all 3D objects and assets (that are to be streamed) in a web-host in a 3D database server. Therefore, object streaming will be done via the Internet and as a result different players can connect to 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 of 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 each other.

TABLE 2: NORMALIZED IMPORTANCE FACTORS FOR OUR SPECIFIC 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 explained before, in our approach each object has a priority for a given activity and all these priorities are adjustable. For our game, we have designed an importance factor table similar to that in Table 1, shown in Table 2. Our importance factors here are designed only for the “task accomplishment” aspect of gameplay, as explained before.

D. The Game

Figure 1 shows multiple snapshots of our game, for the same frame using different methods. The top row shows the game’s rendering using our method, and the bottom row shows the game’s rendering using distance-based approach. Each row



a) Activity-Based Aiming (our approach)



b) Activity-Based Walking (our approach)



c) Distance-Based Aiming



d) Distance-Based Walking

Figure 1: Walking and Aiming snap-shots of the game using both our method and distance-based method

shows two activities: aiming, and walking. In these pictures, 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 1a), 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 1b)

But in the distance-based approach, no matter what the current activity is, only the objects which are in certain distance from the player are streamed. As can be seen in the picture, there are two problems with this approach. First, when aiming (Figure 1c), 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. So 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 the aiming and shooting activities, wasting resources and reducing the quality of interaction between the game and the player. Using our activity-based methodology on the other hand, we have not only skipped streaming irrelevant objects for a given activity,

but we have also increased the overall experience and game quality on the mobile device for the player.

V. EVALUATION

Since the objective of our evaluation is to confirm that using our approach for object streaming is more efficient in terms of resources while it also increases gameplay experience and quality, we have not focused on the MMOG aspect and our test sessions are in single player mode where we want to show the functionality of our method and to compare it with other approaches. However, our conclusions are general and apply to 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

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 ratio of the “important” 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 done in a game session. For example, consider the scenario where 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 loaded objects for those two activities done in the game, divided by the total objects’ size downloaded.

- c) 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 in different object streaming approaches, and is an indication of the quality of interactivity and of the game experienced by the player.

A. Results and Analysis

Figure 2 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.

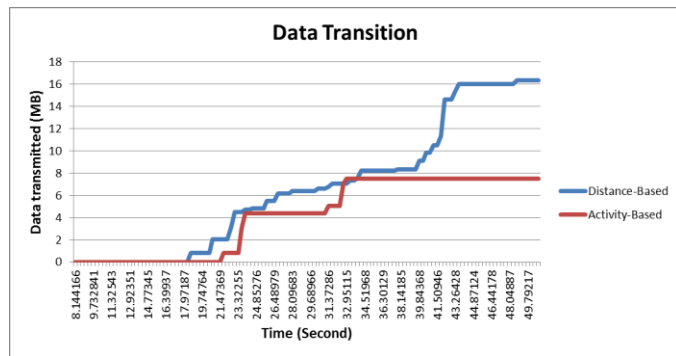


Figure 2: Data Transition Rate

For our activity-based approach, around the 23rd second, the player has switched to “shooting” mode, and this is why the amount of data transmitted has increased to ~4 MB in our approach. This means by going from “idle” to “shooting” status, 4 MB of objects which are important for shooting are needed. From second 23 to second 32, the player was in shooting mode, and that’s why there is no more data downloading reported during this time period. At second 32, player has switched to walking activity and 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 (~7 MB). The player then continues the game without downloading anything.

For the distance-based approach, we played the game from second 0 and started to explore the whole scene during the time, to let the server stream all the visible objects to the player. At around 44th second of the game, the exploration has been finished and all objects have 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 3 shows the importance/size ratio described before. As explained, the rationale behind this ratio is to find how much of the downloaded data have been important for the activities that have been done in the game session. In other words, by calculating this ratio, we want to compare the total download size required to get a specific amount of importance. As can be seen in the picture, for example when the total importance of loaded objects in the scene is 57% of all the potentially important objects (for specific activities undertaken in the session), we can compare which approach has downloaded less data in order to get this 57% importance. The less the download size, the more the ratio will be.

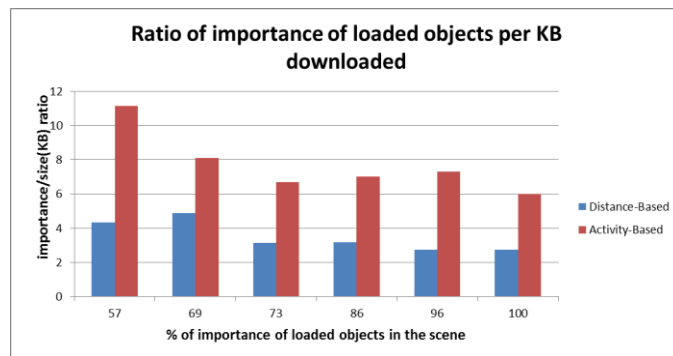


Figure 3: Ratio of importance/file size

For the distance-based approach, this ratio is smaller since irrelevant objects have been transmitted during various activities without increasing the total importance of loaded objects in the scene. Those irrelevant objects have “0” unit of importance for the activities done in that session, therefore, by downloading more data, there is no added importance to the loaded objects. For our activity-based approach, this ratio is much higher because we have skipped cases where the total percentage of importance of loaded objects is less than 50%, and we have mostly transmitted important objects for a given activity.

We have set the game such that it will end if a player manages to hit 4 warehouses. In our approach (activity-based), say we are in “shooting” mode: our algorithm will stream to the player all the important objects required for shooting and, as a result, the player is able to shoot the objects faster than the distance-based approach. This is why in activity-based approach less time is required to hit 4 warehouses, but in the distance-based approach the player needs to explore the scene more and get close enough to the objects to be able to see them and hit them. As a result, he needs a longer time to finish the game. This is confirmed by our evaluations as can be seen in Figure 4.

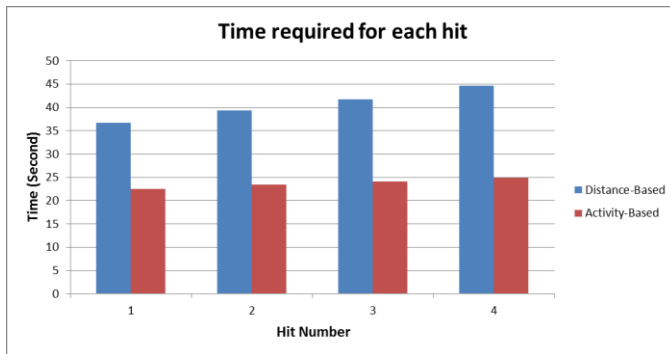


Figure 4: Time required for each hit

VI. DISCUSSION AND FUTURE WORK

In our testing and analysis, we find that depending on the object arrangements and their positions in the scene, depending on the number of total activities we have in the game, and depending on if we have used all the available activities or not, we might have some scenarios where the distance-based approach might also have similar results to our context-aware approach. As an example, in a scenario where all the warehouses are 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, and 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 the scenarios with many numbers of activities and many numbers of objects, with too many downloadable files with big sizes. The bigger the level, the more efficient our algorithm will be.

For our future work, we will do more tests and data analysis with regards to the MMOG aspect of our framework. As well, we will conduct our experiment in situations where there are a number of user-defined restrictions (like maximum download size, limited battery life, etc.).

VII. CONCLUSION

In this paper, we presented 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 we showed that our approach not only reduces resource utilization, but also increases the quality of interactivity and gameplay as experienced by the player.

REFERENCES

- [1] E. Teler, D. Lischinski. "Streaming of Complex 3D Scenes for Remote Walkthroughs", *Procs. Eurographics*, 17-25. 2001.
- [2] O. Hagsand, "Interactive Multiuser VEs in the DIVE System", *IEEE Multimedia*, 3(1):30-39. 1996.
- [3] J. Leigh, A. Johnson, C. Vasilakis, T. DeFanti, "Multi-perspective Collaborative Design in Persistent Networked Virtual Environments", *Proc. IEEE VRAIS*, 253-260, 1996.
- [4] R. Waters, D. Anderson, J. Barrus, D. Brogan, M. Casey, S. McKeown, T. Nitta, I. Sterns, W. Yerazunis, "Diamond Park and Spline: A Social Virtual Reality System with 3D Animation, Spoken Interaction, and Runtime Modifiability", *Presence*, 6(4):461-480. 1997.
- [5] K. Saar, "VIRTUS: A Collaborative Multi-User Platform", *Proc. Symposium on VRML*, 141-152. 1999.
- [6] J. Falby, M. Zyda, D. Pratt, R. Mackey. "NPSNET: Hierarchical Data Structures for Real-Time Three-Dimensional Visual Simulation", *Computers & Graphics*, 17(1):65-69, 1993.
- [7] M. Macedonia, M. Zyda, D. Pratt, D. Brutzman, P. Barham, "Exploiting Reality with Multicast Groups: A Network Architecture for Large-scale Virtual Environments", *Proc. IEEE VRAIS*, 38-45, 1995.
- [8] F. Li, R. Lau, D. Kilis, L. Li, "Game-on-demand: An Online Game Engine Based on Geometry Streaming," *ACM Transactions on Multimedia Computing, Communications and Applications*, accepted 2010.
- [9] H. Rahimi, A.A.N. Shirehjini, S. Shirmohammadi, "Context-Aware Prioritized Game Streaming", *Proc. Workshop on Interactive Ambient Intelligence Multimedia Environments, in Proc. IEEE International Conference on Multimedia & Expo*, Barcelona, Spain, July 11-15, 2011
- [10] <http://unity3d.com/>
- [11] <http://www.exitgames.com/Photon>
- [12] "Web Player Streaming", Unity Manual, Unity Technologies, <http://unity3d.com/support/documentation/Manual/Web%20Player%20Streaming.html>