Uniquitous: an Open-source Cloud-based Game System in Unity

Meng Luo and Mark Claypool
Computer Science and Interactive Media & Game Development
Worcester Polytechnic Institute
Worcester, MA 01609, USA
{mluo2,claypool}@wpi.edu

1. Introduction

Cloud gaming is an emerging service based on cloud computing technology which allows games to be run on a server and streamed as video to players on a lightweight client. Cloud gaming is estimated to grow from $1 billion in 2010 to $9 billion in 2017 [1], a rate much faster than either international or online boxed game sales.

Figure 1 depicts how games can be run in the cloud. The game computation, normally done on the client's computer or game console is instead done on one of many cloud servers, on the right. The server maintains the game world and computes the game scene images that the player sees on the screen, sending these game frames down to the client as streaming video. The client, on the left, can be "thin" since it no longer needs to do the heavy-weight computation of updating the game world and rendering the game scene – the client only needs to display the game frames as video and manage the player input. The client captures the player input in the form of mouse, keyboard and/or game controller actions, and sends it upstream to the game server in the cloud where the server incorporates the input into the game world as if the entire game was played locally on a traditional, non-cloud-based client.

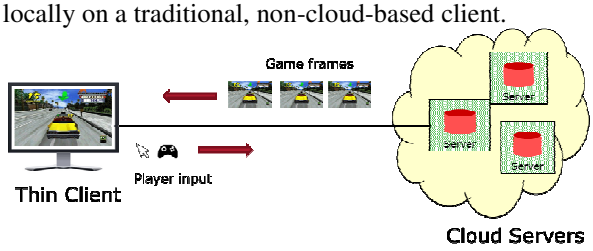

Figure 1. Cloud gaming

Cloud gaming provides benefits to players, developers and publishers over traditional gaming. Cloud gaming allows players to access games streamed as video through mobile devices, such as laptops, tablets and smart phones, which provides the potential to play the same game everywhere without constraints on hardware. Game developers only need to develop one version for the cloud server platform instead of developing a version for each client type, thus reducing game development time and cost. Publishers can more easily protect against piracy since cloud games are only installed in the cloud, thus limiting the ability of malicious users to make illegal copies.

Despite some recent successes, before cloud gaming can be deployed widely for all types of games, game devices and network connections, cloud gaming must overcome a number of challenges: 1) network latency, inherent in the physical distance between the server and the client, must be mitigated; 2) high network capacities are needed in order to stream game content as video down to the client; and 3) processing delays at the cloud gaming server need to be minimized in order for the game to be maintained, rendered and streamed to the client effectively for playing. Research and development required to overcome these challenges need cloud gaming testbeds that allow identification of performance bottlenecks and exploration of possible solutions.

Several commercial cloud gaming systems, such as OnLive [2] and StreamMyGame [3] have been used for cloud gaming research. Although these commercial services can be readily accessed, their technologies are proprietary, providing no way for researchers to access their code. This makes it difficult for researchers to explore new technologies in cloud-based gaming and makes it difficult for game developers to test their games for suitability for cloud-based deployment. While GamingAnywhere [4] provides an open source cloud gaming system, it remains separated from the game itself, not supporting integration and simultaneous exploration of game code and the cloud system. For instance, researchers cannot easily explore latency compensation techniques that require both the game code and the networking code to monitor player lag if the game code is separate from the cloud system.

In order to provide a more flexible and easily accessed platform for cloud gaming researchers and game developers, we present *Uniquitous* [5], a cloud gaming system implemented using Unity [6]. Unity is a cross-platform game creation system with a game engine and integrated development environment. Uniquitous is exported as an independent Unity package, which can blend seamlessly with existing

Unity projects, making it especially convenient for Unity developers, one of the largest and most active developer communities in the world – the Unity community increased from 1 million registered developers in 2012 to over 5 million in 2015 [7].

Uniquitous is open source, allowing modification and configuration of internal cloud gaming structures, such as frame rate, image quality, image resolution and audio quality, in order to allow exploration of system bottlenecks and modifications to meet client-server requirements. In addition to enabling system modifications, by being in Unity, Uniquitous enables game adjustments for further exploring the relationship between the game itself and cloud gaming performance. For example, game objects can be adjusted to study the effect of scene complexity on network bitrates, or camera settings can be altered to study the effect of perspective on cloud gaming frame rates. Although Uniquitous was developed on a desktop, Unity can build to both iOS and Android with full networking support, allowing Uniquitous to provide interactive streaming game video to mobile devices.

This article provides a brief introduction of the design and evaluation of Uniquitous. For details see other publications [8, 9] with source code and system documentation available online [5].

2. Architecture

Uniquitous' architecture is shown in Figure 2. It is composed of three entities: Unity Project, Uniquitous Server and Uniquitous Thin Client. The Uniquitous Server and the Uniquitous Thin Client run on two separate computers connected by an Internet connection while the Unity Project runs on the same computer as the Uniquitous Server. Figure 2 shows three types of data flows in Uniquitous, illustrated with different shades/colors: the red image flow carries data for the game frames; the green audio flow carries data for the game audio; and the blue input flow carries data for user input. Flows within components on the same machine are represented with dashed lines while flows across the network are shown with solid lines.

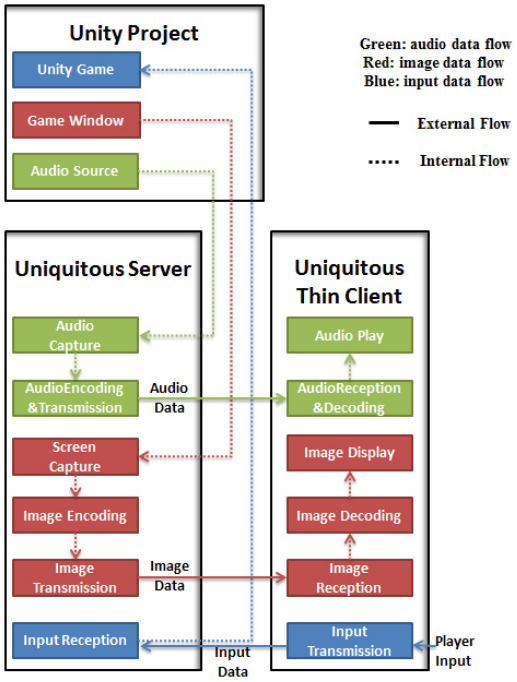

Figure 2. Uniquitous architecture

3. Experiments

We conducted micro experiments to evaluate the performance of the Uniquitous server components, focusing on processing time for bottleneck analysis, and macro experiments to evaluate the Uniquitous system as perceived by the player, focusing on game image quality and frame rate since they are among the most important to the player. Select results are presented in this section, with full results available in [9].

All experiments were run on PCs with Intel 3.4 GHz i7-3770 processors, 12 GB of RAM and AMD Radeon HD 7700 series graphic cards, each running 64-bit Windows 7 Enterprise. The PCs were connected by a 100 Mbps network LAN. The games tested, the Car Tutorial [10] and Angry Bots (version 4.0) [11], are provided by Unity Technologies.

Since Unity by default can do JPEG decoding, the Image Encoding component is implemented using a JPEG encoder [12]. While future work could use inter-frame compression common in video systems (e.g., H.264), JPEG encoding is sufficient to implement and evaluate our Uniquitous prototype.

To evaluate frame rates achievable in Uniquitous, 44 configurations for Car Tutorial and 37 configurations for Angry Bots were tested. Each configuration varied the JPEG encoding quality factor and resolution. To compute the frame rate, the time differences between frames provided the average frame intervals, and the inverse provided the frame rates. Figure 3 shows the frame rate results for Angry Bots. The x axis is the JPEG quality factor, and the y axis is the frame rate. Each point is the framerate average during a predefined period with trendlines grouping the different screen resolutions. Note, the higher resolution images are not tested at higher JPEG quality factors since RPC limits prevent images larger than 64 Kbytes from being transmitted.

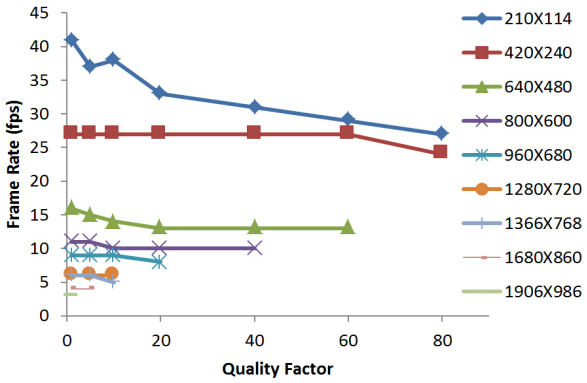

Figure 3. Frame rate versus JPEG quality factor for Angry Bots

From Figure 3, Angry Bots can achieve a maximum frame rate of 41 fps at a 210×114 resolution and 1 JPEG encoding. With the exception of this smallest image resolution, decreasing the JPEG quality factor does little to change the frame rate. However, increasing the frame resolution has a pronounced effect on decreasing the frame rate for both games. Based on previous results [13], frame rate is more important to players than resolution and a game system needs to provide a minimum of 15 fps for reasonable player performance. Both games tested can achieve 15 fps at a resolution of 640×480, hence is the recommended resolution setting for Uniquitous on this hardware setup. Based on player pilot tests, under these settings, both games are quite playable.

4. Frame Rate Predicting Model

With all the data collected from the experiments, we made a model to predict Uniquitous frame rate for configurations not yet tested. In order to build the model, we used a Weka classifier [14] with a 10-fold cross validation to make a linear regression model for both games:

$$F_{CarTutorial} = 1 / (0.1348 \times R + 0.118 \times Q + 21.0)$$
$$F_{AngryBots} = 1 / (0.1361 \times R + 0.1224 \times Q + 22.5)$$

where F is the predicted frame rate, R is the total pixel resolution divided by 1000, and Q is the JPEG quality factor. In order to validate our model, new R and Q values that had not been tested before were chosen, 35 for the Car Tutorial and 30 for Angry Bots, and the actual frame rates measured. The results are show in Figure 4.

The x axis is the predicted frame rate and the y axis is the actual frame rate as measured. Each point is the average frame rate over the experimental run. The diagonal line shows what would be perfect prediction. Generally, most of the data points are near this line,

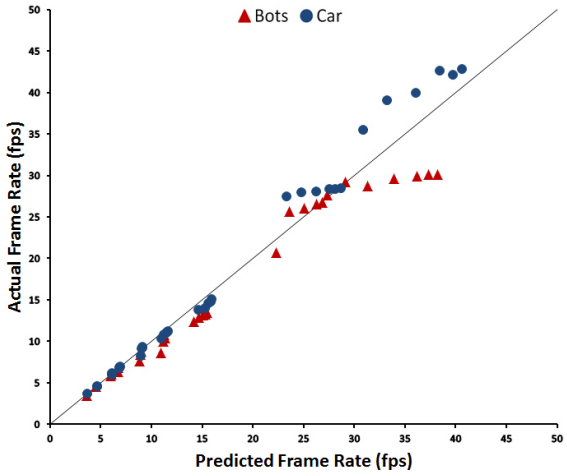

Figure 4. Actual versus predicted frame rate

showing that the model is generally quite accurate. The points are somewhat closer to the line for frame rates under 20 fps than for frame rates over 25 fps, probably due to unaccounted for processing that accumulates more with more frames per second. The actual and predicted frame rates have a correlation of 0.995 for Car Tutorial and 0.981 for Angry Bots.

5. Conclusion

Realizing the potential for cloud gaming requires testbed systems for researchers and developers. This article introduces Uniquitous [5], an open source cloud gaming system in Unity, providing a prototype that can be used for evaluating cloud gaming performance tradeoffs. Uniquitous seamlessly blends with Unity game development, providing control not only over the game system but also over the game content in a cloud-based environment. Micro experiments provide performance evaluation of the Uniquitous components, macro experiments evaluate game quality of the Uniquitous system, and a model predicts Uniquitous frame rates for games and hardware not yet tested. Validation of the model shows effectiveness for predicting frame rate over a range of configuration parameters. The evaluation shows the Unity Project running the game is the most time consuming component on the server when the game image quality and resolution are both low, but Image Encoding becomes the bottleneck for higher resolutions. For our system testbed, JPEG quality factors below 35 and resolutions below 640×480 pixels provide a configuration suitable for game play.

Future work can seek to increase Uniquitous frame rates and/or support higher resolutions and image qualities by addressing the identified bottlenecks. In addition, more game genres can be tested, exploring the relationship between the game genre and cloud

gaming performance. Lastly, since Unity IOS and Android are fully supported by Unity, Uniquitous can be extended and evaluated on mobile devices, helping research and development of cloud-based games on wider range of clients.

References

[1] "Distribution and Monetization Strategies to Increase Revenues from Cloud Gaming," goo.gl/YnlE9V, 2012, accessed 01-May-2014.

[2] OnLive: http://onlive.com/

[3] StreamMyGame: http://streammygame.com/

[4] C. Huang, Y. C. C. Hsu, and K. Chen, "GamingAnywhere: An Open Cloud Gaming System," in Proceedings of ACM MMSys, Oslo, Norway, Feb. 2013.

[5] Uniquitous: http://uniquitous.wpi.edu/

[6] Unity3d: http://unity3d.com/

[7] "Unity Company Facts": http://unity3d.com/public-relations, accessed 09-Jun-2015.

[8] Meng Luo and Mark Claypool. "Uniquitous: Implementation and Evaluation of a Cloud-based Game System in Unity", In *Proceedings of the IEEE Games, Entertainment, Media Conference (GEM)*, Toronto, Canada, October 2015.

[9] M. Luo, "Uniquitous: Implementation and Evaluation of a Cloud-Based Game System in Unity 3D," Master's Thesis, Worcester Polytechnic Institute, 2014, adv: M. Claypool.

[10] Car Tutorial: https://www.assetstore.unity3d.com/en/#!/content/1012

[11] Angry Bots: https://www.assetstore.unity3d.com/en/#!/content/12175

[12] A. Broager, "JPEG Encoder Source for Unity in C#," goo.gl/FwOfOW, accessed 06-May-2014.

[13] M. Claypool and K. Claypool, "Perspectives, Frame Rates and Resolutions: It's all in the Game," in *Proceedings of Foundations of Digital Games (FDG)*, FL, USA, Apr. 2009.

[14] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA Data Mining Software: An Update," *SIGKDD Explorations*, vol. 11, no. 1, 2009.

Meng Luo received a B.Eng. degree in Telecommunications Engineering with Management from Beijing University of Posts and Telecommunications in 2012. He received an M.S. degree in Interactive Media and Game Development from Worcester Polytechnic Institute in 2014. Currently he is working as a software engineer at Advanced Visual Systems, Inc.

Mark Claypool is a Professor in Computer Science and Interactive Media & Game Development at Worcester Polytechnic Institute in Massachusetts, USA. Dr. Claypool earned M.S. and Ph.D. degrees in Computer Science from the University of Minnesota in 1993 and 1997, respectively. His primary research interests include multimedia networking, congestion control, and network games.