Towards Usable Attribute Scaling for Latency Compensation in Cloud-based Games

Edward Carlson, Tian Fan, Zijian Guan, Xiaokun Xu and Mark Claypool
ekcarlson,tfan,zguan,xxu11,claypool@wpi.edu
Worcester Polytechnic Institute
Worcester, Massachusetts, USA

ABSTRACT

Cloud-based games have advantages in convenience over traditional computer games, but have the disadvantage of added latency from the thin client to the cloud-based server and back. This added latency has been shown to decrease player performance. New latency compensation techniques can help by scaling game attributes to make the game easier, exactly counteracting the difficulty added by the latency. We conduct a user study measuring attribute scaling for two games – a first-person shooter and a rhythm game – each having a different attribute scaling method: spatial and temporal. Data from the study shows a decrease in accuracy with an increase in latency and game difficulty, and an increase in accuracy with an increase in attribute scaling. More importantly, we derive a model from the data whereby a pre-determined accuracy can be chosen – say, by the game designer – and the model then outputs the scaling factor to meet that desired target accuracy.

CCS CONCEPTS

• **Applied computing** → **Computer games**; • **Human-centered computing** → *User studies.*

KEYWORDS

latency compensation, gamer, lag, flow

ACM Reference Format:
Edward Carlson, Tian Fan, Zijian Guan, Xiaokun Xu and Mark Claypool. 2021. Towards Usable Attribute Scaling for Latency Compensation in Cloud-based Games. In *Workshop on Game Systems (GameSys '21) (GameSys '21), September 28-October 1, 2021, Istanbul, Turkey*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3458335.3460964

1 INTRODUCTION

Cloud-based games differ from traditional online games in that game clients are relatively lightweight, only sending user input (e.g., keyboard, mouse and controller actions) and receiving game output (i.e., game images and sounds). The heavyweight game logic – applying physics to game objects, resolving collisions, processing Artificial Intelligence, etc. – and rendering are done at the server, with the game frames streamed to the client to display. A cloud-based game system offers advantages over traditional game systems including: modest client hardware requirements, no required client game installation, easier software piracy prevention, and fewer target platforms for developers.

Unfortunately, such systems have the significant disadvantage of round-trip latency from client to server for all game actions. Prior work has shown even modest amounts of latency can impact player performance and Quality of Experience (QoE) for cloud-based games [2, 5, 6]. Thus, to realize their potential benefits to game accessibility and game quality, cloud-based game systems must overcome their unique latency challenge – cloud-based game clients cannot immediately act upon player input but must instead send the input to the server, have it processed, the result rendered, and image data sent back to the client for display.

Approaches to compensate for network latency [1] have been widely used in traditional multiplayer network games. Such techniques include virtual time manipulation (e.g., temporally buffering player actions), prediction (e.g., dead reckoning) and visual trickery (e.g., showing a local animation before processing input). Unfortunately, many of these established techniques cannot be applied to cloud-based game systems since the client is "thin," not having the processing power and game information needed to compute the game state and render the game world.

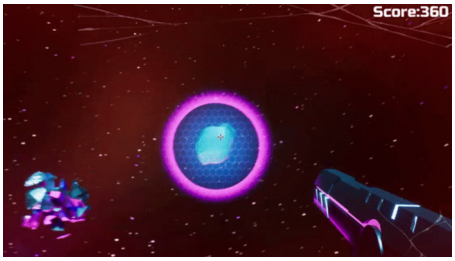
We seek to develop new latency compensation techniques that can be used by game developers with their cloud-based games, techniques that give the developer control over the player experience while providing per-player compensation based on each player's latency to the game server. One promising technique is attribute scaling. Attribute scaling relies upon the concept of flow, keeping a player "in the zone" – a game that is too difficult can lead to frustration, causing the player to quit, while a game that is too easy can lead to boredom, also causing the player to quit. Flow looks for the "sweet spot" in game difficulty where the player has just enough challenge to make the game fun. To compensate for latency, the attributes in a game can be scaled according to a client's latency in order to provide for the "just right" flow that the game designer envisioned.

Such adjustments would not necessarily be appropriate for all game objects and attributes, but instead can be decided by a game developer (e.g., tagging an object attribute). The cloud-based game engine would then make automatic adjustments on the fly based on the round-trip latency experienced by each client. This can provide for a uniform game experience, the game experience envisioned by the game designer, for all players, regardless of their clients' latencies. While some cloud-based game systems may seek to deliver single-player games without source-code changes, others, such as

GameSys '21, September 28-October 1, 2021, Istanbul, Turkey

ACM ISBN 978-1-4503-8437-7/21/09…$15.00
https://doi.org/10.1145/3458335.3460964

(a) Catalyst – a first person, capture the flag shooter game



(b) Nova – a first person, target selection rhythm game

Figure 1: Screen shots of custom-built games used in our study

Google Stadia, provide and encourage development targeted for their platform.

This paper presents some first steps towards attribute scaling for latency compensation in a game engine. We develop two custom games each with a different attribute scaling technique and conduct a 23-person user study to observe the effects of latency on players for conditions of latency, game difficulty and attribute scaling. Analysis of the results confirms decreasing player performance with increasing latency and difficulty and increasing performance with increasing scaling. We derive a model from the data that can allow for game attributes to be automatically scaled by a game engine based on a client's latency, game difficulty and the desired player performance.

The rest of this paper is organized as follows: Section 2 briefly describes related work; Section 3 details our methodology, including our user study; Section 4 analyzes the user study result and derives an attribute scaling model; and Section 5 summarizes our conclusion and presents ongoing and future work.

2 RELATED WORK

Work related to ours includes: latency and cloud-based games, game actions and latency, attribute scaling based on latency, and models of game actions.

Latency and Cloud-based Games: Chen et al. [2] discuss the effects of network latency (and other parameters) on two cloud-based game systems. Jarschel et al. [7] conducted a user study in an emulated cloud-based game system, measuring the quality of experience for games users selected to play. Claypool and Finkel [5] present the results of two user studies that measured the objective and subjective effects of latency on cloud-based games. Sackl et al. [11] analyze the relationships between latency and player experience for cloud gaming. These papers show latency impacts player performance and degrades player quality of experience.

Game Actions and Latency: Claypool and Claypool [3] propose precision and deadline as parameters to predict player performance trends for game actions with latency. Sabet et al. [12] demonstrate that changing precision (in their case, the size of the game objects) and deadline (in their case, the pace of the game) can yield higher gamer scores with latency and an improvement to most aspects of QoE.

Attribute Scaling Based on Latency: Lee et al. [8] demonstrate that geometric compensation – scaling the size of a game object – is effective for counteracting the effects of latency on player performance. Sabet et al. [10] show attribute scaling (game target sizes, spawn rates, and predictability) can improve player performance, albeit without having done so dynamically with per-client latency.

Models of Game Actions: Claypool et al. [4] confirm user studies can provide accurate modeling (adjusted R^2 of 0.99) of individual game actions (in their case, moving target selection with a mouse) with latency. Long et al. [9] show a model of an individual game action (in their case, moving a paddle to intercept a ball) with latency can accurately predict player performance in a more complex game.

3 METHODOLOGY

To assess how latency affects player performance for two attribute scaling methods, the following methodology was deployed: 1) Design and develop two games, each with a different attribute scaling method, and both with tunable scaling, difficulty and latency; 2) Measure base system delay to get the minimum latency values; 3) Design and conduct a user study with the custom games and added latencies to evaluate latency's impact; and 4) Analyze and model the user study data in terms of player performance, game difficulty and scale factor.

3.1 Games and Attribute Scaling

Two custom-made games were designed and developed from scratch using Unreal Engine (UE4 v4.25)[1] for use in our study.

Catalyst is a first-person, team capture-the-flag game where players cast spells (a total of 10 are available) instead of using weapons. Figure 1a shows a screen shot. The player chooses combinations of the three elements in the bottom of the screen to determine the spell cast. The cast spell then launches towards the reticle aimed at an opponent.

The attribute scaling method for Catalyst is shown in Figure 2. The hitboxes are highlighted with the light blue oval and vary in size with the scaling factor. For example, if the player misses 15% more often with 100 ms of latency, the hitboxes can be scaled larger to increase accuracy back to the no latency condition. In the actual game, the hitboxes are invisible.

Nova is a first person rhythm game where players shoot invading asteroids at times aligned with the music. Figure 1b shows a screen shot. The player fires the gun in the bottom right corner,

[1] https://www.unrealengine.com/

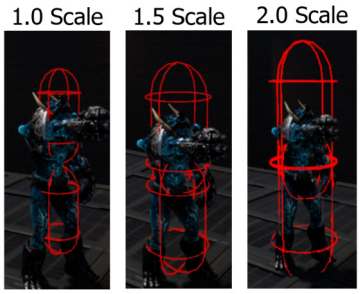


Figure 2: Catalyst – attribute scaling by adjusting hitbox size.

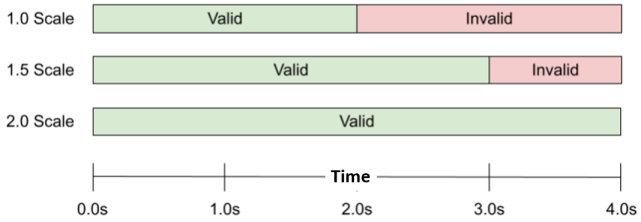


Figure 3: Nova – attribute scaling by adjusting time window.

using a reticle to aim at the asteroid, shown glowing in the center of the screen, when the asteroid is targetable (in time to the music).

The attribute scaling for Nova is depicted in Figure 3. Time is indicated on the horizontal axis. The time an asteroid is targetable is shown by the green box with the word "valid" and, when untargetable, is shown with a red box with the word "invalid". The top bar at 1.0 scale shows a small valid time window, made 50% larger at 1.5 scale and twice as large at 2.0 scale. For example, if the player cannot hit asteroids in time 10% more often with 75 ms of latency, the time windows can be scaled longer to get player performance back to the no latency condition.

3.2 User Study

Our user study was conducted in a dedicated, on-campus computer lab.

The test computer was a PC with an Intel i7-4789k CPU @ 4GHz with 8 cores, 16 GB RAM and an NVIDIA GeForce GTX 960 graphics card. The PC ran Windows 10 Home 20H2. The mouse was a Logitech G203 with 8k DPI running at 1000 Hz. The monitor was an 24" LCD Dell U2412M, 1920x1200 pixels @ 60 Hz.

To provide for accurate assessment of the latencies user experienced in the study described here, the base system latency was measured on the test computer. The measurement method is depicted in Figure 4. A high frame rate camera (a Casio EX-ZR100) filmed a user at 1000 f/s, capturing the moment the mouse button was clicked. By manually examining the video frames, the frame number when the mouse was clicked (finger bent, frame number 327 in Figure 4) is subtracted from the frame number when the output of the action was visible (frame number 377 in Figure 4), giving the base system latency (50 milliseconds in Figure 4).

The measurement method was repeated 10 times on our system, resulting in an average base latency of 80 milliseconds.

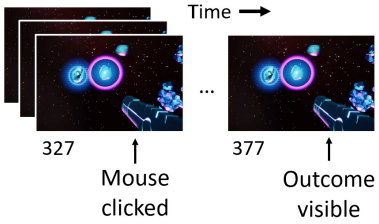


Figure 4: Measuring local system latency

Table 1: Subjective questions per round

	Rate (1-low to 5-high)
Q1	How enjoyable was this round?
Q2	How frustrating was this round?
Q3	How challenging was this round?
Q4	How responsive were the game controls?

In order to test the effects of latencies above the baseline, each game was modified to intercept all keyboard and mouse input. Before the input was applied, it was queued for a fixed amount of time before being dequeued and then applied normally.

The added latencies in this user study were 0, 50, 100 and 125 milliseconds.

In order to facilitate testing the effects of latency over a range of game conditions, the games were adjusted so each focused on just the core gameplay (no start screen, menu lobbies, etc.). Instead, game rounds launched immediately into the action and lasted for 20 seconds. For Nova, the focus was on shooting at individual notes that spawned, then became targetable, then disappeared. Difficulty was controlled by the cooldown time between notes. For Catalyst, the focus was on casting a projectile spell at an AI-controlled opponent that ran back and forth on a platform separated by a moat. Difficulty was controlled by changing the speed of the opponent.

After each round, users were presented with a subjective survey about their experience consisting of four questions on a discrete, 5-point Likert scale about the game experience in the preceding round. The questions are shown in Table 1. After completing the survey, the next round would commence. However, users could take as much time as they wanted before starting the following round.

The computer was configured to start up and launch the user study sessions with minimal instructions so as to reduce user error. Users were randomly chosen to start with either the Catalyst game or the Nova game. The game itself: 1) shuffled the test conditions, then: 2) picked a test condition - latency, scale, and difficulty settings, 3) ran a game round with those settings for 20 seconds, 4) stopped the round and launched the survey questions, 5) gathered and archived all game logs and survey results, and 6) repeated the process for each game condition. Users played all rounds in the first game before proceeding to the second game.

Prior to the start, each user heard a scripted brief about the purpose of the study and signed a consent form. Then, the user sat at the computer, was told to make themself comfortable by adjusting

Table 2: Independent variables

Game	Catalyst, Nova
Latency	0, 50, 100, 125 ms
Scaling	1x, 1.5x, 2x
Difficulty	Catalyst: Easy (200 cm/s), Hard (500 cm/s)
	Nova: Easy (1 s), Hard (0.5 s)

Table 3: Demographics

Users	23
Age (years)	19.8 (1.5)
Gender	15 male, 6 female, 1 non-binary, 1 not specified
Playtime (hrs/wk)	10.4 (8.4)

chair height and monitor angle/tilt and to put on headphones and adjust the volume.[2]

To allow users to get familiar with the games, the first two rounds for each game were practice, where the user played at the easiest settings and did not answer any survey questions. Data from all practice rounds was discarded.

The IRB-approved user study was conducted during the COVID pandemic, so everyone wore masks and respected social distancing requirements. Upon completion of each user's study, we carefully sanitized the keyboard, mouse and earphones.

Before the launch of the formal user study, pilot studies were conducted with volunteers in order to test the viability of the procedure and tune the study settings. The pilot study results helped adjust difficulty, attribute scale factors, latency values, number of rounds, round length and user instructions.

Study participants were solicited via University email lists and the University Social Science research participant pool. Incentives included course credit for students that needed it and a raffle for Google Stadia subscription codes for all participants.

4 ANALYSIS

4.1 Participants

Twenty-three (23) users participated in the user study. Table 3 summarizes the user demographics. For age and playtime, mean values are given with standard deviations in parentheses. Most were computer science or game development majors. The sample is mostly young, skewed toward male and plays games considerably each week – typical of the undergraduate students at the university.

Game rounds that had more shots fired than 1.5x the inter-quartile range below the first quartile or 1.5x above the third quartile were considered outliers and removed – 7.5% for Nova and 2.4% for Catalyst.

4.2 Performance

For Catalyst, the main performance metric is accuracy:

$$\text{Catalyst accuracy} = \frac{\text{shots hit}}{\text{shots fired}} \tag{1}$$

[2]Headphones were provided for use, but participants were also welcome to use their own.

For Nova, the main performance metric is the number of notes that are hit before their time windows expire. We also call this accuracy, but the equation is slightly different:

$$\text{Nova accuracy} = \frac{\text{notes hit}}{\text{total notes}} \tag{2}$$

Figure 5 depicts boxplot distributions for the accuracy scores versus latency. The boxes depicts quartiles and medians for the distributions. Points higher or lower than 1.4 × the inter-quartile range are outliers, shown by the hollow circles. The whiskers span from the minimum non-outlier to the maximum non-outlier. The solid dots show the mean values. For these graphs and all others, the green depicts Catalyst and the blue depicts Nova.

From the graphs for both games, based on the median and mean values accuracy generally decreases with latency. There is considerable variation in accuracy at each latency value as evidenced by the height of the boxes (larger for Catalyst than Nova). This is to be expected, however, as players' performances vary from game to game. In general, however, player performance decreases with latency, with accuracies for both games dropping about 10% from latency 0 ms to latency 125 ms.

Figure 6 depicts boxplot distributions for the accuracy scores versus difficulty, similar to Figure 5 but here broken down by difficulty on the x axis. For both games, the hard rounds result in lower accuracy than the easy rounds.

Figure 7 depicts boxplot distributions for the accuracy scores versus attribute scale value on the x axis. From the graphs, for both games higher scale values generally yield higher accuracies, suggesting scaling hitboxes in Catalyst and time windows in Nova can be used to adjust overall player accuracies.

We conducted a one-way, within-subjects ANOVA for each parameter: latency, difficulty and scale. The results are shown in Table 4. There are significant effects at the $p < .05$ level for all parameters for both games.

Table 4: ANOVA results for accuracy

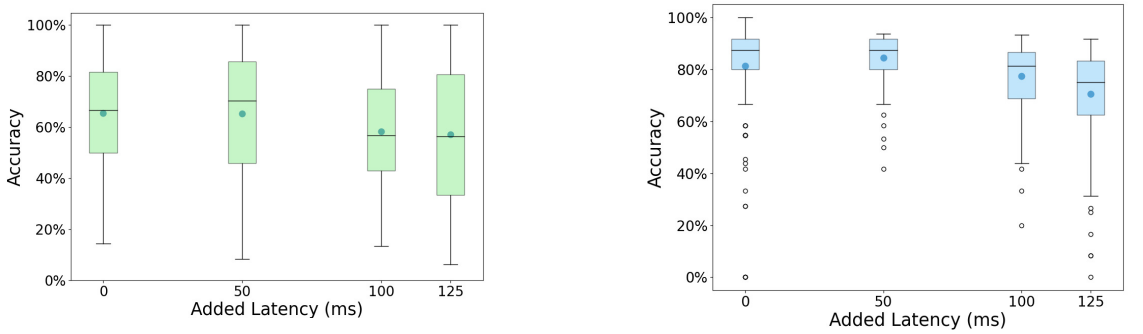
Game	Parameter	F	p
Catalyst	Latency	F(3,592) = 5.2	.001
	Scale	F(2,593) = 33.1	<.001
	Difficulty	F(1,594) = 316.8	<.001
Nova	Latency	F(3,530) = 16.7	<.001
	Scale	F(2,531) = 16.9	<.001
	Difficulty	F(1,532) = 7.1	.008

4.3 Model

In order to provide a latency compensation mechanism that can be used by a game engine to automatically adjust game attributes to accommodate for latency, we first derive a model of accuracy for each game. For the range of game conditions tested, the degradation to player performance appears linear. Thus, we fit a linear regression to the means for each game condition, building a model for each game based on latency, difficulty and scaling.

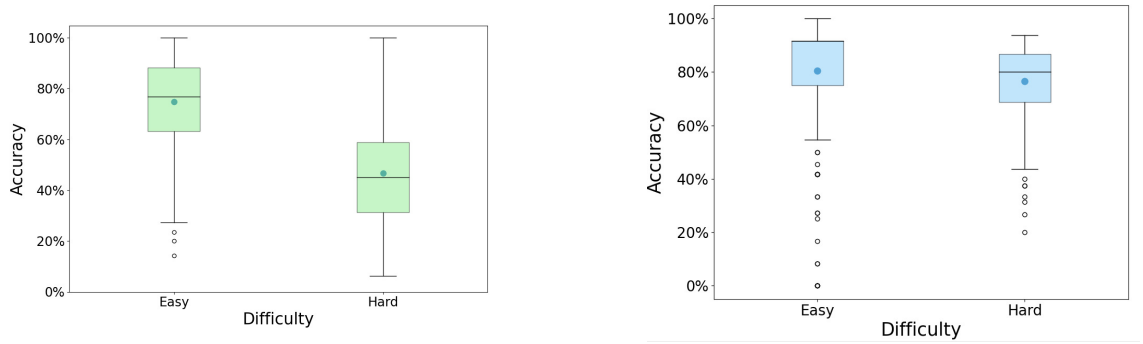
For Catalyst, the model for accuracy (percent) is:

$$a = -0.08l + 20s - 0.1d + 72 \tag{3}$$



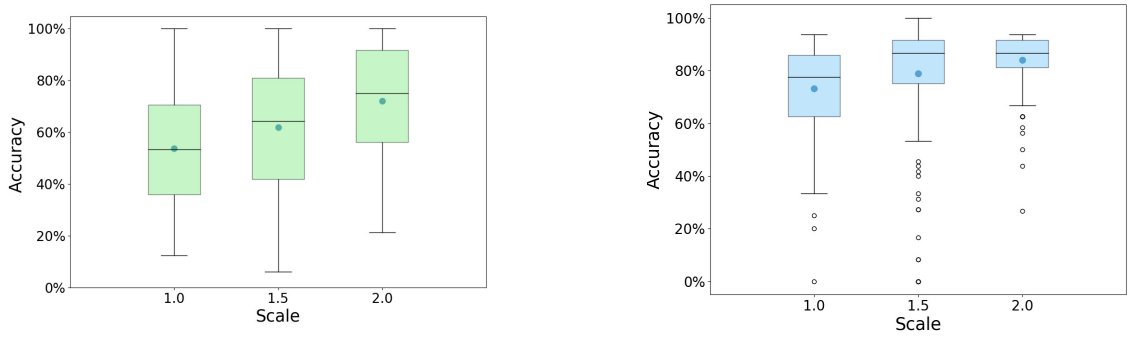
(a) Catalyst **(b) Nova**

Figure 5: Accuracy versus Latency



(a) Catalyst **(b) Nova**

Figure 6: Accuracy versus Difficulty



(a) Catalyst **(b) Nova**

Figure 7: Accuracy versus Scale

where l is the latency (in milliseconds), s is the time window scale factor, and d is the difficulty (the speed of the opponent avatar, in cm/s). From the equation, accuracy decreases by 0.08% for each millisecond of latency, but can be increased 20% for each scale factor increase and 0.1% for each difficulty decrease. The R^2 is 0.95.

For Nova, the model for accuracy (percent) is:

$$a = -0.1l + 11s + 10d + 62 \quad (4)$$

where l is the latency (in milliseconds), s is the time window scale factor, and d is the difficulty (the cooldown between notes spawning, in seconds). From the equation, accuracy decreases by 0.1% for each millisecond of latency, but can be increased 11% for each scale factor increase and 10% for each difficulty increase. The R^2 is 0.69.

While a more complex model could be applied, more data should be obtained prior, and over a wider range of conditions, to justify such fitting. Until then, a linear model appears a reasonable fit.

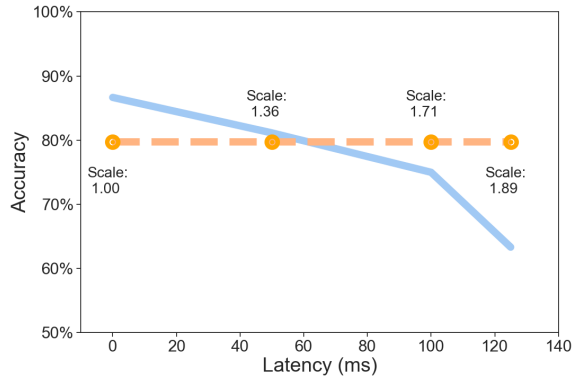


Figure 8: Concept of adaptive attribute scaling for Nova

4.4 Attribute Scaling

While Equation 4 and Equation 3 provide predictions for the average accuracy given a latency, scaling and difficulty, they can also be used to determine the appropriate scaling to yield an intended accuracy in the presence of latency by solving for the scale factor.

For Catalyst, the scaling factor can be set to:

$$s = \frac{a + 0.09l + 0.1d - 73}{20} \tag{5}$$

For Nova, the scaling factor can be set to:

$$s = \frac{a + 0.1l - 12d + 63}{10} \tag{6}$$

For each game, for a given game level, the game designer would choose the intended accuracy based on the challenge they envision for the player (not too easy, not too hard). The game system could then adjust the attribute scale factor using Equation 5 for Catalyst and Equation 6 for Nova.

Figure 8 illustrates this functionality for Nova. The x axis is the client latency and the y axis is the accuracy. The blue line depicts the measured latency values for a user playing Nova in our study with an easy difficulty and a 1.0 scale factor. The orange line shows the intended "just right" accuracy set by the game designer whereby the labels on the orange line are the time window scale factors that the game engine would set based on the measured thin-client latency and Equation 6. At 50 ms of latency, the scale value is set to 1.36, at 100 ms scale is 1.71 and at 125 ms scale is 1.89. In all cases, the expected average player accuracy remains constant, similar to that experienced with no network latency.

5 CONCLUSION

Cloud-based games face challenges from latencies from the thin-clients to the server, latencies that degrade player performance. Game attribute scaling, where aspects of a game are scaled based on the thin-client's latency, has the potential to mitigate the performance hit players take when playing with latency by adjusting the game difficulty to keep the challenge "just right", as intended by the game designer.

We present results from a user study that takes a step towards general attribute scaling for games by establishing use: in two games: Catalyst, a first-person shooter game that scales opponent hitbox sizes with latency, and Nova, a first-person rhythm game that scales the time windows for selecting asteroid-notes with latency. A twenty-three (23) person user study covering a range of latency, difficulty and scaling conditions shows player accuracy degrades with latency but also indicates how this degradation can be overcome with scaling. We derive a mathematical model that can be used to determine the exact scaling value needed for a given latency and game condition to achieve the accuracy a game designer intends.

Our ongoing work is to evaluate the use of the latency compensation methods in the full games. Specifically, using Equation 6 to scale the temporal windows in Nova and Equation 5 to scale the hitboxes in Catalyst. We are deploying the games in the Stadia cloud-game system, assessing performance (accuracy) as well as quality of experience.

Future work is to see how well the attribute scaling equations work for games with similar genres (e.g., first person shooter games and rhythm games) as well as to derive alternatives for other game actions and game genres. This will be accompanied by deployment in a game engine (e.g., UE4) and establishing an application programmer interface (API) for use by designers. Models could developed for quality of experience, as well. Future work should also look to determine game scenarios and conditions where attribute scaling is and is not appropriate, and what game attributes should be scaled. Multi-player games where each player may have a different latency may need particular attention. Other future work may explore the needed frequency for gathering round-trip time measurements and then adjusting attributes, particularly in network settings where variance in latency is high.

Acknowledgments. Thanks to Google Stadia for supporting this work, Alejandra Garza, Joseph Swetz, Cameron Person, Adam Desveaux and James Plante for developing Catalyst, and Michael Bosik, Nina Taurich, and Alex Hunt for developing Nova.

REFERENCES

[1] Y. Bernier. 2001. Latency Compensating Methods in Client/Server In-game Protocol Design and Optimization. In *Proceedings of GDC*. San Francisco, CA, USA.
[2] K. Chen, Y. Chang, H. Hsu, D. Chen, C. Huang, and C. Hsu. 2014. On the Quality of Service of Cloud Gaming Systems. *IEEE Trans. on Multimedia* 26, 2 (Feb. 2014).
[3] M. Claypool and K. Claypool. 2010. Latency Can Kill: Precision and Deadline in Online Games. In *Proceedings of ACM MMSys*. Scottsdale, AZ, USA.
[4] M. Claypool, R. Eg, and K. Raaen. 2017. Modeling User Performance for Moving Target Selection with a Delayed Mouse. In *Proc. of MMM*. Reykjavik, Iceland.
[5] M. Claypool and D. Finkel. 2014. The Effects of Latency on Player Performance in Cloud-based Games. In *Proceedings of NetGames*. Nagoya, Japan.
[6] V. Clincy and B. Wilgor. 2013. Subjective Evaluation of Latency and Packet Loss in a Cloud-based Game. In *Proceedings of the International Conference on Information Technology: New Generations (ITNG)*. Las Vegas, NV, USA.
[7] M. Jarschel, D. Schlosser, S. Scheuring, and T. Hoßfeld. 2011. An Evaluation of QoE in Cloud Gaming Based on Subjective Tests. In *Proccedings of the Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*. Seoul, Korea.
[8] I. Lee, S. Kim, and B. Lee. 2019. Geometrically Compensating Effect of End-to-End Latency in Moving-Target Selection Games. In *Proceedings of ACM CHI*.
[9] M. Long and C. Gutwin. 2018. Characterizing and Modeling the Effects of Local Latency on Game Performance and Experience. In *Proceedings of ACM CHI Play*. Melbourne, VC, Australia.
[10] S. Sabet, S. Schmidt, S. Zadtootaghaj, B. Naderi, C. Griwodz, and S. Moller. 2020. A Latency Compensation Technique Based on Game Characteristics to Mitigate the Influence of Delay on Cloud Gaming Quality Of Experience. In *Proceedings of ACM MMSys*. Istanbul, Turkey.
[11] A. Sackl, R. Schatz, T. Hoßfeld, F. Metzger, D. Lister, and R. Irmer. 2016. QoE Management Made Uneasy: The Case of Cloud Gaming. In *IEEE Conference on Communications Workshops (ICC)*. Kuala Lumpur, Malaysia.
[12] S. Saeed, S. Schmidt, S. Zadtootaghaj, C. Griwodz, and S. Moller. 2018. Towards Applying Game Adaptation to Decrease the Impact of Delay on Quality of Experience. In *Proceedings of IEEE ISM*.