Physics Engine Benchmarking in Three-Dimensional Virtual World Simulation

Sean C. Mondesire, Douglas B. Maxwell
U.S. Army Research Laboratory
Orlando, FL
sean@cs.ucf.edu, douglas.maxwell3.civ@mail.mil

Jonathan Stevens, Steven Zielinski, Glenn A. Martin
Institute for Simulation and Training
University of Central Florida
jonathan.stevens@knights.ucf.edu, szielins@ist.ucf.edu, martin@ist.ucf.edu

ABSTRACT

The U.S. Army is continuously looking for novel and innovative methods and technology to enhance operational and tactical simulation-based training (SBT) for its soldiers. Through the Military OpenSimulator Enterprise Strategy (MOSES) project, the U.S. Army Research Laboratory is identifying critical software features that will improve the next generation of military simulation's training effectiveness. At present, MOSES is investigating methods to increase the number of simultaneous soldiers that can train in a three-dimensional (3D), virtual world simulator. The current limitation of simultaneous trainees in a high-fidelity environment is approximately 16-44 soldiers, representing the platoon echelon in the U.S. Army. This limitation is unfortunate as the Army requires the next generation of SBTs to support training at the company echelon of (100-200) soldiers.

The presented work investigated a common limiting factor in virtual world simulations, the physics engine. We compared the physical load limits of three physics engines; Open Dynamics Engine (ODE), Bullet, and PhysX. These engines were studied because they are common in video games, special effects for television and film, and 3D simulations. More specifically, ODE was the first physics engine implemented for the virtual world simulator, OpenSimulator (OpenSim). BulletSim is an award-winning and industry-recognized engine that is the current default OpenSim physics engine using the BulletSim wrapper. PhysX is the latest real-time physics engine from Nvidia that has been recently integrated into OpenSim. With this study, other simulation users, developers, and administrators gain a quantifiable method to compare physics capabilities among computer-based systems.

ABOUT THE AUTHORS

Dr. Sean C. Mondesire is a Postdoctoral Research Fellow at the U.S. Army Research Laboratory (ARL) and holds a doctoral degree in computer science from the University of Central Florida (UCF). His research focus is on the expansion of virtual world technologies to be used in tactical, military training.

Dr. Douglas B. Maxwell is a Science and Technology Manager at the U.S. Army Research Laboratory-Human Research and Engineering Directorate (ARL-HRED). He is responsible for the Military Open Simulator Enterprise Strategy (MOSES) simulator. Dr. Maxwell earned his Ph.D. in Modeling & Simulation from UCF.

Dr. Jonathan Stevens is a Research Scientist with the Institute for Simulation and Training (IST) at UCF and a Mathematics Professor at Valencia College, specializing in training simulation research. Dr. Stevens is a retired Lieutenant Colonel of the United States Army with over 22 years of military experience. He received his Ph.D. in Modeling & Simulation from UCF**.**

Mr. Steven Zielinski is an Assistant in Simulation at UCF's Institute for Simulation and Training. He has worked on numerous projects, including augmented reality, intelligent training tools, and real-time virtual environments. He has a B.S. in Computer Science from the University of Central Florida.

Dr. Glenn A. Martin is a Research Assistant Professor at UCF's Institute for Simulation and Training. He earned a Ph.D. in Modeling & Simulation in 2012 from the University of Central Florida. He pursues research in adaptive and intelligent training, game-based learning, multi-modal simulation, and interactive high performance computing.

Physics Engine Benchmarking in Three-Dimensional Virtual World Simulation

Sean C. Mondesire, Douglas B. Maxwell
U.S. Army Research Laboratory
Orlando, FL
sean@cs.ucf.edu, douglas.maxwell3.civ@mail.mil

Jonathan Stevens, Steven Zielinski, Glenn A. Martin
Institute for Simulation and Training
University of Central Florida
jonathan.stevens@knights.ucf.edu, szielins@ist.ucf.edu, martin@ist.ucf.edu

INTRODUCTION

The U.S. Army Research Laboratory's (ARL) Military OpenSimulator Enterprise Strategy (MOSES) project utilizes its domain expertise from industry, academic, and the military to identify key technological features for future Army simulation-based trainers (SBTs). In the presented study, the MOSES project analyzed the performance impact of a core component of three-dimensional (3D) virtual world simulators, the physics engine. Physics engines are responsible for handling the calculations required for any physical interaction of objects; these include but are not limited to gravity, forces, collisions, and joints. During simulation execution, the physics engine runs in real-time to manage up to thousands of objects at once within the simulation. Because of this demand, the physics engine affects the simulator's overall performance and can limit the amount of simultaneous users, the number of supported physical objects, and stability of the system. This study compared the performance of three popular physics engines on the military MOSES simulator: Open Dynamics Engine, Bullet, and PhysX. With the study, we identified physics engine features that have had the most significant impact on the tested virtual world simulator.

The MOSES simulator is a derivative of the open-sourced OpenSimulator (OpenSim). OpenSim's core functionality in MOSES has been viable for military collective echelon training and simulator training effectiveness research. OpenSim originally used the Open Dynamics Engine (ODE) and now includes (and defaults to) the Bullet physics engine. Recently, support for Nvidia's PhysX library was added to the MOSES simulator for the new engine's advanced features, such as multi-threaded capabilities. An analysis of these three physics engines were performed and is presented here. Due to OpenSim's initial use of ODE, the engine has received a large amount of testing and supports collisions with all primitives (prims) and any combination of distortions or static mesh objects (Mondesire, Stevens, Leis, & Maxwell, 2015). We included ODE as part of our analysis as it was the original, default physics engine inside of OpenSim. ODE is also used in a number of games and robotic simulators throughout the virtual world industry (Open Dynamics Engine, 2016).

Bullet is an open-source physics engine that has been popular in the entertainment, videogame, and simulator domains (Bullet Physics Library, 2016). BulletSim is a module for OpenSim that uses the Bullet physics engine. The module added physical vehicle compatibility and is now the default physics module in OpenSim (OpenSim BulletSim, 2016).

PhysX is the latest physics engine made available to MOSES and OpenSim (Nvidia Corporation, 2016). The recent release of PhysX as open-sourced software, as well as the recent development of a PhysX plugin for MOSES and OpenSim, had made the engine a new option that may address the needs of the MOSES project. Similar to Bullet, PhysX is also used extensively in video games and 3D simulation software.

OpenSim provided this study with a distributed virtual world platform with an object-rich environment suitable for physics engine testing. The study quantifiably compared ODE, BulletSim, and PhysX in relation to the simulator's overall performance. To do so, we compared simulation frame times affected by each of the different physics engines as the number of physical objects increased in the virtual world. From the produced results, an accurate model of limitations was generated and assessed. The study resulted in the identification of the physics engine and key features that significantly affected simulator performance.

BACKGROUND

The primary objective of this study involving the MOSES project was to find a physics engine that would be capable of supporting a company formation of (62-190) soldiers in a single, virtual training environment. As such, accurate measurements of the current capabilities of the physics engines available in OpenSim were required to identify a suitable physics engine. By utilizing the known capabilities of each physics engine, an appropriate scaling method and physics engine selection could be developed to support complex virtual world training scenarios.

The ODE physics engine has limitations, most notably the lack of multithreading in the physics calculations. Previous work involved ODE and built a parallelized version of ODE's collision detection using the OpenMP framework, and found that in a world with a large number of objects, the central processing unit (CPU) showed a substantial linear increase (Goodstein, Ashley-Rollman, & Zagieboylo, 2016). ODE also lacks the ability to create a convex mesh for use in physics calculations, which will limit the types of objects present inside of the virtual world (Boeing & Bräunl, 2007). Specifically, any object not created through the linking of primitive objects will be unable to have forces of any kind applied to them, although other objects will still collide with a non-moving object.

The same physics engine comparisons showed that Bullet is missing separate values for static and kinetic friction. Bullet is also unable to directly create a cone primitive or directly create a corkscrew, distance, fixed, prismatic, or universal joint (Boeing & Bräunl, 2007). While Bullet supports the application of friction to objects its lack of friction customization may cause problems for vastly complex scenarios and environments. BulletSim (separate from Bullet) currently lacks the support for multithreaded physics calculations. BulletSim does support running the physics calculations on a separate thread from the OpenSim main update loop; however, OpenSim does not support a separate thread for the physics engine and immediately requests the physics calculation results when BulletSim attempts to use a separate thread.

PhysX was built to support parallel physics calculations, allowing for the physics processing to be handled by multiple threads (Goodstein, Ashley-Rollman, & Zagieboylo, 2016). Unlike Bullet, the multithreading abilities of PhysX were implemented by the OpenSim plug-in. In a comparison study, it was observed that a previous version of PhysX made the most drastic compromises when it came to physical accuracy (Erez, Tassa, & Todorov, 2015). However, the focus in this presented study was on how well PhysX 3.33 supported large scalability of virtual world simulation with its improved stability and performance.

Previous works comparing ODE, Bullet, and a pre-3.33 version of PhysX showed that none of the physics engines performed the best at all of the tasks they performed (Boeing & Bräunl, 2007). For example, PhysX outperformed Bullet and ODE in the integrator test. The integrator test evaluated performance by applying forces to an object and compared the physics simulation to expected results. Bullet was more accurate when calculating penetration during collisions while ODE was the most accurate at tracking constraints (although this required an accurate integrator). Overall, none of the physics engines were dramatically better than the others, leading developers to make a selection based on the most important property of a specific virtual world environment (Boeing & Bräunl, 2007). In another study, PhysX was found to outperform Bullet for non-complex geometries in a haptic assembly task, but not for complex shapes (Gonzalez-Badillo, et al.).

In another study, ODE, Bullet and PhysX all performed equally well at collision computation (Hummel, Wolff, Stein, Gerndt, & Kuhlen, 2012). Bullet was less than 1 millisecond (ms) slower than PhysX with 2,500 spheres colliding, and PhysX was slower than ODE by less than 3 ms. A study of the accuracy of the physics model showed that each engine had its strengths and weaknesses. PhysX was stable with little inter-penetrations, but also had unrealistic behavior when it came to restitution and advanced friction. Bullet also had troubles with restitution, but was able to handle the advanced collisions and friction well. ODE was fast and accurate until constraint and collision handling were stressed (Hummel, Wolff, Stein, Gerndt, & Kuhlen, 2012).

With the MOSES project's ultimate goal of supporting a collective training event at the battalion echelon (~1000 users) training simultaneously, the current 2,500 spheres will not be large enough to test the scale of the system (Mondesire, Stevens, Leis, & Maxwell, 2015). Based on these previous results, deciding upon a physics engine for the task can lead to changes in speed and accuracy of the system. For now, this speed and accuracy analysis of the physics engine has helped determine the overall scalability of the OpenSim virtual world environment. Therefore,

the MOSES project used the three physics engines to create an environment whereby quantifiable metrics could be collected to evaluate the necessary performance measures for future use and development.

METHODOLOGY

An experiment was conducted to capture quantifiable data to compare performance between BulletSim, ODE, and PhysX. The experiment measured how well each engine processed varying amounts of simulator physics load and identified the engines' object scalability limitations. This section defines the experiment's physics load scenario, step-by-step testing procedures, gathered statistics, and hardware and simulation specifications.

Physics Load Scenario

A 3D ball pit environment (Figure 1) was used as the physics load scenario to evaluate the three physics engines. This ball pit environment was a 256 x 256 square metered flat, virtual island, referred to as a *region*. The region contained a series of 10, 5-meter tall concentric circle walls with the inner-most circle residing at the center of the region. Each wall served as a boundary that was 10 meters away from its adjacent circle. All of the walls had the setting of being *physical* and *persistent* in the simulation. Physical objects required the simulator's physics engine to calculate gravitational forces, collisions, and other physics-based calculations. Therefore, the simulator experienced additional processing load with each physical object present in a region. Persistent objects resided in the simulation until they are explicitly deleted. Hence, once created, persistent objects automatically are initialized in a region after the simulator's instance is terminated and restarted.

A ball *spawner* also resided in the region. The spawner generated increased physics load on the simulator by continuously creating and dropping balls in the region when activated. The spawner was a persistent scripted tool that was suspended 20 meters off of the ground, centered to the region's x-y coordinates. The spawner created and dropped waves of 50 balls from the spawner's location when activated, at a rate of one ball per millisecond. Each ball was one meter in diameter and set to be a non-persistent, physical object. Because the balls were non-persistent, they were deleted with each termination of the simulator and did not carry over to separate simulator runs.

Testing Procedure

The same testing procedure was executed 30 independent times for each of the three physics engines, totaling 90 separate simulation trials. Each engine used its default settings, without any extra optimized configurations. The following instructions were performed for each trial:

1. The MOSES simulator was launched and initialized.
2. The spawner was activated to spawn 50 balls.
3. Simulator statistics were gathered.
4. Steps 2 and 3 were repeated until a total of 5,000 balls were spawned.
5. All 5,000 balls were deleted from the region.
6. The MOSES instance was terminated.

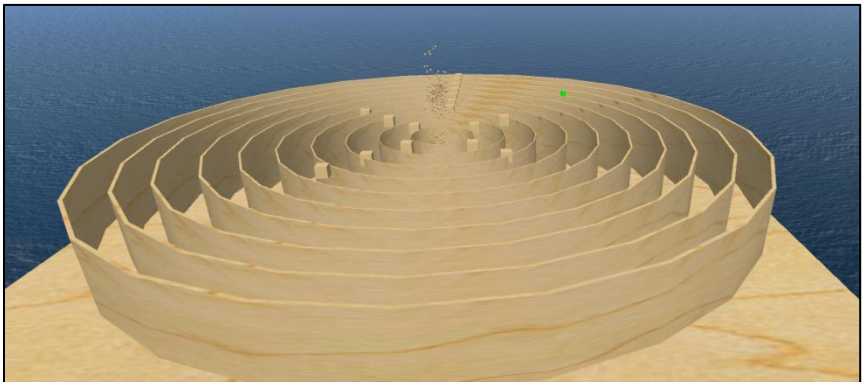

Figure 1: A screen capture of the ball pit scenario.

Statistics Gathered

More than 30 different simulation metrics were gathered throughout these experiments, including memory usage, CPU utilization, and network bandwidth. The number of active primitive objects (AtvPrm), simulation frames per second (SimFPS), and physics frames per seconds (PhyFPS) were isolated to evaluate overall physics engine performance.

AtvPrm measured the number of physical objects (active primitives) that were present in the simulated environment at any given moment. In our experiments, AtvPrm represented how many balls were generated by the spawner at any point of a trial and measured the amount of load the simulator was processing. The SimFPS was the amount of simulation frames the simulator processed per second. A simulation frame consisted of all instructions the simulator had to perform in a single game-loop, including object and avatar state changes, message relaying, object scripting, and a single physics frame for physics-based calculations. The physics frames per second (PhysFPS) metric was the amount of physics frames the engine could process per second. In a physics frame, the engine calculated object and avatar positional updates, collisions, and gravitational forces, among other physics-based calculations. Because a single physics frame was processed within a simulation frame, both frame rates were identical and bounded by the SimFPS rate. Due to a restriction of the OpenSim design, the simulator was limited to a maximum frame rate of 11.333. The experiments followed a stability threshold of 9 frames per second; a frame rate lower than 9 resulted in degraded simulator performance and control unresponsiveness between users and the simulator.

Hardware and Simulation Specification

The experiments were sequentially performed using the same hardware and software for all 90 trials. The MOSES version of OpenSimulator v. 0.8.2 was used, which enabled more robust simulator performance data collection. MOSES also integrated NVIDIA PhysX ver. 3.33 into the simulator, allowing BulletSim, ODE, or PhysX to be physics engine options on the same version of OpenSim. Each trial of the simulator was executed on an Intel i7 6-core processor that supported 12 concurrent threads, 32 GB of RAM, and used the Ubuntu Desktop 14.04 64-bit operating system. The PhysX engine was configured to run on one thread to match the single-thread capabilities of BulletSim and ODE.

RESULTS

The experiment used the simulator's employed physics engine and the number of spawned balls (AtvPrm) as the independent variables. The SimFPS served as the response variable. With these three variables and the collected statistics, each physics engine's performance with various amounts of physical object load was measured, compared, and ranked. Specifically, the maximum amount of load to assure simulator stability was first measured, followed by a comparison of each physics engine's performance at different load levels. This section details the experimental results and the statistical analyses that were performed on each engine.

The first analysis determined the minimum amount of physical object load each physics engine supported that cause the simulator's performance to degrade. Here, the ball count that served as the first occurrence of when the SimFPS averaged less than 9 frames per second for each engine was identified. Table 1 displays the minimum degradation load of each engine. ODE averaged 200 AtvPrm (SimFPS: $M = 8.33$, $SD = 0.46$), BulletSim averaged 1800 AtvPrm ($M = 8.79$, $SD = 0.51$), and PhysX averaged 3300 AtvPrm ($M = 8.83$, $SD = 0.29$) as the minimum degradation load. Next, the maximum amount of load each engine could support before performance started to degrade was identified; this was defined as the highest AtvPrm where the SimFPS was at least 9 FPS. ODE supported 150 AtvPrm (SimFPS: $M= 11.07$, $SD = 0.31$), BulletSim 1750 AtvPrm ($M = 9.35$, $SD = 0.46$), and PhysX 3250 AtvPrm ($M = 9.11$, $SD = 0.39$). When ranked, PhysX supported the highest amount of physical objects, followed by BulletSim, and then ODE. Figure 2 displays each engine's performance based on AtvPrm load.

The second analysis evaluated each engine's performance under pre-determined levels of physical object load. At the lowest object load level of 500 AtvPrm, PhysX averaged a SimFPS of 11.31 ($SD = 0.08$), BulletSim 11.28 ($SD = 0.12$), and ODE 2.47 ($SD = 0.19$). ANOVA found significant difference between the engine SimFPS values, $F(2, 90) = 549.69$, $p < 0.001$. Post-hoc Tukey HSD determined that there was no significant difference between the

Table 1: Minimum degradation load per physics engine.

Physics Engine	AvtPrm	SimFPS	
		AVG	STD
PhysX	3300	8.83	0.29
BulletSim	1800	8.79	0.51
ODE	200	8.33	0.46

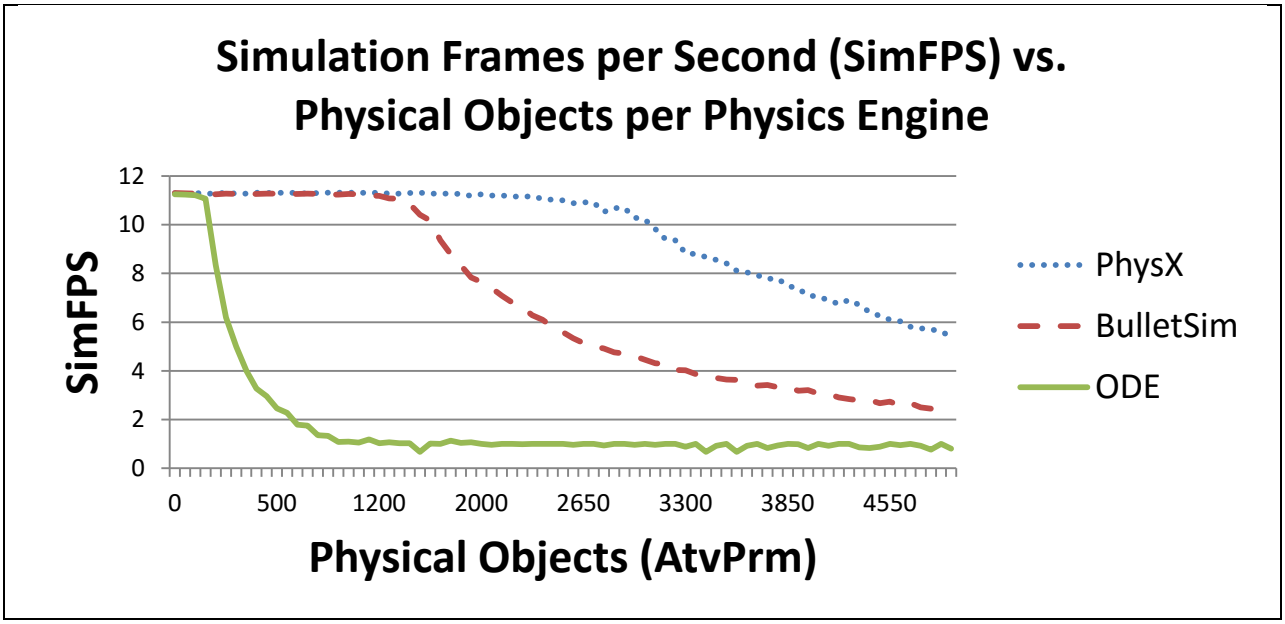

Figure 2: Physics engine performance with active object load.

PhysX and BulletSim averaged SimFPSs, $p = 0.61$. There were significant differences between PhysX and ODE ($p = 0.001$) and BulletSim and ODE ($p = 0.001$). ODE was determined to significantly underperform compared to PhysX and Bullet from these statistical tests.

At 2,500 physical objects, PhysX produced an average SimFPS of 11 (SD = 0.17), BulletSim 5.59 (SD = 0.32), and ODE 1.00 (SD = 0.00). Again, ANOVA found that at least one of the engines performed significantly different from the others, $F(2, 72) = 9{,}538.64$, $p < 0$. Post-hoc Tukey HSD found significant differences between all of the engines, $p < 0.001$. Specifically, at 2,500 physical objects (half the maximum tested load) PhysX produced a significantly higher SimFPS than BulletSim and ODE, while BulletSim averaged a significantly higher frame rate than ODE.

Lastly, engine performance was evaluated at a load of 5,000 AtvPrm. PhysX averaged 5.4 SimFPS (SD = 0.24), BulletSim 2.37 (SD = 0.21), and ODE 0.81 (SD = 0.16). ANOVA found at least one engine to be significantly different from the others, $F(2, 64) = 2{,}728.72$, $p < 0.001$. The Post-hoc Tukey HSD test determined that there were significant differences between all three engine performances, $p < 0.001$. Again, PhysX significantly outperformed both BulletSim and ODE and BulletSim produced a higher frame rate than the worst performing ODE engine.

DISCUSSION

The results showed that all three physics engines had different physical object thresholds and responded differently to various amounts of load placed on the simulator. As expected, the engines maintained the maximum simulation frame rate of 11 frames per second (fps) with minimal physical object load but simulator frame rate decreased as the number of active primitives rose. Our analysis found that ODE was the worst performing engine; its performance decreased dramatically with the frame rate falling below 9 fps with just 200 balls. As the number of physical objects increased, the degradation appeared negative exponential. Additionally, the physics engine's sub-3 SimFPS frame

Table 2: Average SimFPS of each engine at various AtvPrm intervals.

	AVG SimFPS			STD SimFPS		
AtvPrm	**PhysX**	**BulletSim**	**ODE**	**PhysX**	**BulletSim**	**ODE**
0	11.29	11.30	11.24	0.18	0.13	0.15
500	11.31	11.28	2.47	0.08	0.12	0.19
1250	11.27	11.08	1.07	0.13	0.22	0.13
2500	11.00	5.59	1.00	0.17	0.32	0.00
5000	5.40	2.37	0.81	0.24	0.21	0.16

rate beyond 500 physical objects indicated how poorly ODE supported larger loads. ODE's underwhelming performance was due to its technological maturity and lack of development in recent years; however, ODE provided a baseline for comparison.

BulletSim was the second best performing engine. The engine maintained a simulation frame rate of 9 fps up until approximately 1,800 prims, nine times the amount of physical objects than ODE. Furthermore, BulletSim produced a SimFPS of 2.37 when the simulation contained 5,000 physical objects, a comparative frame rate of ODE's SimFPS of 2.47 with just 200 objects. Similar to ODE, BulletSim's simulation performance dropped off sharply after it met its minimum degradation load threshold and appeared to be an exponential reduction on the frames per second as number of active primitives increased.

PhysX was the best performing physics engine. It supported the most physical objects before reaching the stability threshold and produced the highest SimFPS frame rates at each of the tested load intervals. In fact, while averaging PhysX's SimFPS over the physical object load generation (Figure 2), the results showed a linear degradation. Overall, PhysX outperformed BulletSim by 83% and ODE by 1550% in physical object increased scalability. This result was consistent with reported informal results (Terdiman, 2016).

PhysX's was an extensive source-code/software library that contained many advanced capabilities and was built with design and implementation variations from ODE and BulletSim; thus, it was difficult to isolate exact key features from the engine that resulted in its superior performance over the other engines. Therefore, it was concluded that under default, standard conditions, PhysX, as a whole, outperformed ODE and BulletSim in physical object scalability due to the engine's updated and optimized source code.

This work was the first in a series of studies to extract key physics engine-related technological lessons in virtual world simulation. The next two studies will focus on multi-threadedness and optimizing the configuration options of the physics engines. For instance, in this presented study, PhysX was tested only with a single CPU thread, although the implementation supported multiple threads; this was a user-configurable setting. Future work will explore varying the number of threads in PhysX and its effects on the number of supported physical objects. The hypothesis is that multi-threaded enabled physics will allow for additional active primitives at 9 SimFPS or higher. However, the number of threads to be used is an open question, and additional threads may not always improve performance due to overhead. Consequently, some improvements should be achievable even after just adding a second thread. Some informal results have shown improvements with both 2 and 3 threads (PhysXInfo.com, 2016).

Finally, PhysX supports offloading some physics calculations onto Graphics Processing Units (GPUs). While these are limited within PhysX (collision detection and resolution are not supported directly), an extension of this work would determine if the addition of GPUs would improve physical object scalability in the simulator. The use of additional hardware is an intriguing research question as it suggests the notion of creating a “remote physics server” where potentially thousands of objects could be simulated and fed back to client simulations. As needed, this hardware can be enhanced to potentially and even larger number of objects.

CONCLUSION

The focus of this work was to quantifiably compare and rank the ODE, BulletSim, and PhysX physics engines on physical object scalability inside the military simulation-based trainer, MOSES. This process examined how well each engine, under default and standard configurations, responded to various amounts of physical object load in the form of processing thousands of rigid bodied objects. Through experimentation, BulletSim (the MOSES default physics engine) convincingly outperformed ODE, while PhysX significantly outperformed BulletSim.

The goal for evaluating ODE, BulletSim and PhysX was to find an improved engine to support the needs of a larger military simulation-based training environment. Via the default engine of BulletSim, MOSES can stably support 16-44 training soldiers at the platoon echelon; however, the Army's need is to accommodate virtual, collective training at the company echelon, approximately 100-200 soldiers. The increased capability discovered with PhysX was a positive result for the MOSES project. Given the goal of supporting 100-200 soldiers and their interactions with each other, opposition forces, and virtual objects, the improved performance was required. PhysX has shown to improve MOSES and OpenSim's ability to handle higher quantities of physical objects, which is a crucial component for improving user scalability in the virtual world.

REFERENCES

Boeing, A., & Bräunl, T. (2007). Evaluation of Real-time Physics Simulation Systems. *GRAPHITE '07 Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia* (pp. 281-288). ACM.

Bullet Physics Library. (2016, January 29). Retrieved from Bullet Physics: http://bulletphysics.org/

Erez, T., Tassa, Y., & Todorov, E. (2015). *Simulation Tools for Model-Based Robotics: Comparison of Bullet, Havok, MuJoCo, ODE and PhysX*.

Gonzalez-Badillo, G., Medellin-Castillo, H. I., Fletcher, C., Lim, T., Ritchie, J., & Garbaya, S. (n.d.). An Evaluation of Physics Engines and Their Application in Haptic Virtual Assembly Environments. *Proceedings of the 37th International MATADOR Conference*, (pp. 227-230).

Goodstein, M., Ashley-Rollman, M., & Zagieboylo, P. (2016, January 22). *Parallelizing the Open Dynamics Engine*. Retrieved from http://www.cs.cmu.edu/~mpa/ode/final_report.html

Hummel, J., Wolff, R., Stein, T., Gerndt, A., & Kuhlen, T. (2012). An Evaluation of Open Source Physics Engines for Use in Virtual Reality Assembly Simulations. In *Advances in Visual Computing* (pp. 346-357). Springer Berlin Heidelberg.

Mondesire, S. C., Stevens, J., Leis, R., & Maxwell, D. B. (2015). Resource Allocation Predictive Modeling to Optimize Virtual World. *Proceedings of the IEEE ICMLA'15 Workshop on Machine Learning for Predictive Models in Engineering Applications (MLPMEA).* Miami.

Nvidia Corporation. (2016, January 29). Retrieved from PhysX: http://www.geforce.com/hardware/technology/physx

Nvidia Corporation. (2016, January 22). Retrieved from PhysX Source on Github: https://developer.nvidia.com/physx-source-github

Open Dynamics Engine. (2016, January 22). Retrieved from Open Dynamics Engine: http://ode.org/

OpenSim BulletSim. (2016, January 22). Retrieved from BulletSim: http://opensimulator.org/wiki/BulletSim

OpenSimulator. (2016, January 29). Retrieved from OpenSimulator: http://opensimulator.org/

PhysXInfo.com. (2016, January 29). Retrieved from http://physxinfo.com/news/11327/multithreaded-performance-scaling-in-physx-sdk/

Terdiman, P. (2016, January 29). *The evolution of PhysX (2/12)*. Retrieved from Coder Corner: http://www.codercorner.com/blog/?p=761