Do the physics engines in modern games platforms simulate physics accurately enough for future uses of virtual reality?

Daniel Chipping

June 21st, 2020

Introduction

Figure 1: Advanced VR Fluid Physics using Valve's Rubikon Physics Engine in Half Life: Alyx [1]

A game platform is often described as some combination of compute-based hardware tied to some software which runs a video game, the personal computer game (PC game) being the cornerstone platform. Modern game platforms are both diverse in approach and resilient to the compute-intensive demands of today's games. We see this in the form of more ambitious platform form-factors such as home and handheld games consoles to the emergence of action-intensive mobile games, and now virtual reality (VR) through the use of Head Mounted Displays (HMDs). Much of this progress is owed to advances in hardware, more efficient use of compute resources and the sophistication of modern game engines [2].

Game engines have been particularly crucial, enabling developers to pursue more ambitious ideas on increasingly resource-constrained platforms. A game engine is a software package with a set of well-integrated functionalities for game development, a typical modern game engine is likely to provide: a rendering engine for 2D/3D graphics, sound, scripting, animation, artificial intelligence (AI), networking, streaming, memory management and notably a physics engine for simulating the physical interactions within the generated virtual environment.

Physics engines are primarily used to abstract away the complexity of codifying the laws of mechanics

in a virtual world from scratch. Those used in game engines often go a step further, providing developers a highly interoperable solution with other functionalities such as a graphics and sound modules [3]. Originating from a need to process highly specialised scientific simulations, for example ballistics profiling for the US Army being an early use-case [4], modern game-oriented physics engines instead use real-time processing [5]. While unlocking the ability to create highly immersive games that appear perceptually correct, the physical interactions generated are not always true to life. Modern engines struggle with the common computer science adage of a time complexity trade-off, simplifying and approximating physics in favour of a responsive real-time experience.

Virtual reality, a platform still very much in its infancy, has shown great promise at providing an experience unique from its predecessors. Current applications have demonstrated the potential of the medium, from improving the efficacy of remote surgery [6] to highly immersive gaming experiences such as Half Life: Alyx [1]. Much of this progress has leveraged existing technology from modern game stacks to do so, for example, Unity a popular general-purpose game engine with over 230,000 developers is a popular choice for building VR games [7]. Unity, like many other game engines uses a general-purpose physics engine (often NVIDIA PhysX) behind the scenes [8], providing a robust and generalised physical simulation which has proved sufficient for a variety of video games and platforms. However, much of the future of VR is still being defined and it is not certain whether these general-purpose physics engines will be accurate enough for these future uses of VR.

Will current fluid dynamics simulation be sufficient for true immersion in a high-fidelity sandbox building game? Are simulated soft-body dynamics of sufficient realism to train surgeons for real surgery using a VR operating theatre? Can modern physics engines update quickly enough to produce accurate next-generation haptic feedback?

Although there are many lenses through which to analyse the issue, this work seeks to narrow scope. Five key areas of VR which show promise in the future are identified, and through these areas this work analyses whether current game-oriented physics engines are accurate enough. These areas of future use are: 1) Haptics rendering, 2) Simulation to real world applications (Sim2Real), 3) Mixed and augmented reality, 4) Virtual reality on low-compute devices, and 5) Highly immersive experiences. From this analysis we then formulate a conclusion about the current state of game-oriented physics engines for future use in VR.

How does a physics engine work?

Before exploring whether current physics engines suffice for future applications, it is prudent to understand some fundamentals of their inner workings. At the most basic level a real-time physics engine works on a cyclical read/update loop, typically: 1) Assess current forces on all entities in the scenegraph 2) Calculate kinematics for the next timestep 3) Update entities' positions and velocities [9]. Doing so at a high enough frequency simulates the application of force on an entity in real-time and when done in parallel with multiple entities, provides the illusion of global physics (see Figure 2). Under the same force entities kinematic behaviours would then vary depending on their material properties and entity type (e.g. rigid-body, soft-body, cloth, fluid).

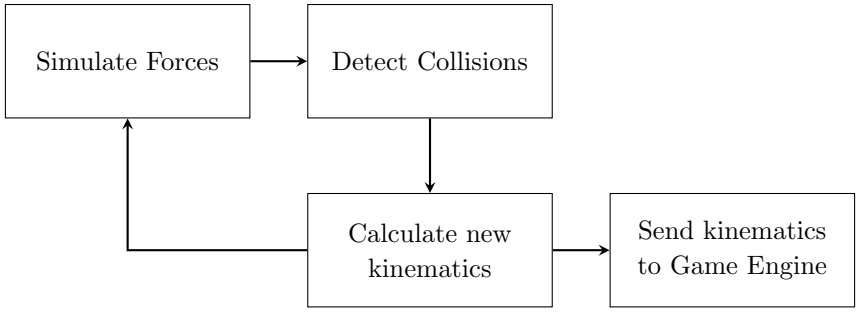

Figure 2: Simplified update loop of a physics engine [10]

Although the exact architecture and physical laws implemented vary between engines the equations of motion are the most important and almost universally relied on. The key equation being Newton's Second Law:

$$\vec{F} = m \times \vec{a}$$

Where $\vec{F}$ is force in the x, y, z axes, m the mass of the entity and $\vec{a}$ the acceleration in each respective axis. When extended to rotational motion i.e. angle, angular velocity and angular acceleration:

$$\vec{M} = I \times \vec{\alpha}$$

Where $\vec{M}$ is the moment about the x, y, z directions, I the mass moment of inertia and $\vec{\alpha}$ the angular acceleration in each respective direction [9]. On each loop of the physics engine the instantaneous translational and rotational kinematics of an entity is updated by:

1. Identifying all moments and forces acting on each entity in the scenegraph.
2. Take the vector sum of all moments and forces for each entity.
3. Use the described equations of motion for translational and angular acceleration.
4. Integrate the acceleration with respect to time to find the translational and angular velocity.
5. Integrate the velocity with respect to time to find the translational and angular displacement.

Where engines differ is by how they complete the above steps and although there exist many subtleties between engines, time is spent to highlight two of the most common variances for the reader's benefit: 1) The numerical solvers used to integrate an entity's kinematic properties 2) The way collisions between entities are handled.

Numerical Solver

Once the sum of moments and forces for an entity is calculated and equations of motion applied, the engine must integrate the equations with respect to time, thus generating the entity's kinematics. As a computer cannot approach these integrals analytically they must be solved with computational methods. Some of the most common include [11]:

- Euler Method
- Verlet Method
- Runge-Kutta Method

Each varying in time complexity, auxiliary memory complexity and convergence rate upon an optimal solution, meaning it is crucial the solver use is calibrated to its use-case. Much of the work around improving solvers leverages the discipline of computational methods.

Collision Handling

Collision handling encompasses collision detection algorithms, entity bounding volumes and the mathematics of momentum transfer between colliding entities. Similarly, the approach to each component will vary between engines, for example, the type of bounding volumes used can vary depending on the accuracy expected and compute the engine can leverage. Typical volumes in order of increasing accuracy and complexity might include [12]:

- Spheres
- Axis-Aligned Bounding Boxes (AABB)
- Oriented Bounding Boxes (OBB)
- Convex Hull

High accuracy collision handling is often highly desired for VR and therefore an important focus for physics engines. A rich tactile experience (generated through momentum transfer from collisions) and perceptually realistic simulation of non-body collisions creates a sensory experience often enough to warner a strong plausibility illusion.

These collisions are registered by the engine's collision detection algorithm which is run during the physics engine loop. The Gilbert-Johnson-Keerthi (GJK) algorithm is one of the commonly used algorithms for achieving precise low false-positive collisions, relying on an iterative method of computing the Euclidean distance between two convex sets in an n-dimensional space [13].

Physics engines found in modern games

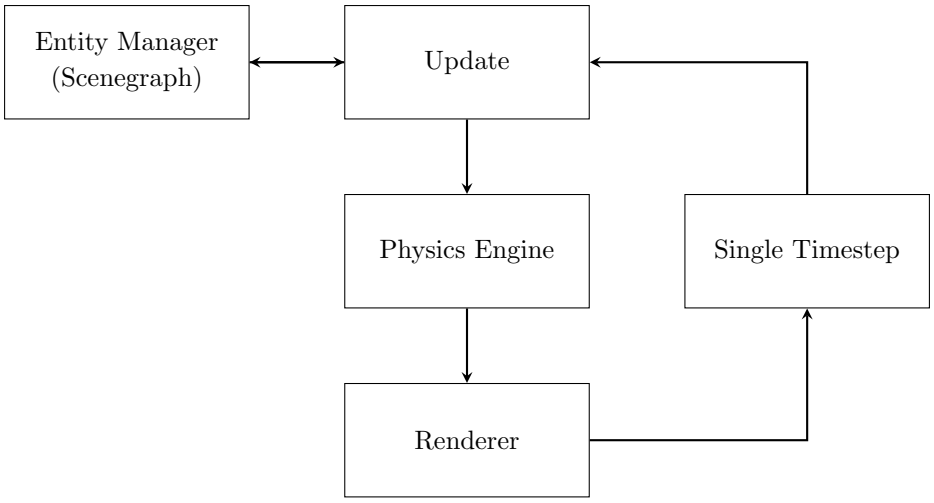

Figure 3: Simplified 'game loop' of a game engine [14]

Of the many general-purpose physics engines that exist, many do so in the context of a modern game engines. This is typically because the the physics engine passes calculated kinematics information of entities in the scenegraph directly to the renderer to visualise, creating the illusion of real-time physics (see Figure 3). This means although many physics engines need not be extensible from their parent game engine or even recognised as a separate software module. Consequently in this section we outline notable 3D real-time physics engines used in modern game engines for reference in later discussion. These range from open source to proprietary as well as some which are non-separable from their parent game engine [15].

NVIDIA PhysX [16]

Colloquially shortened to PhysX, NVIDIA PhysX is considered the most prolific physics engine within the game development community. Numerous games engine's utilise PhysX such as Unity, Unreal Engine and Open 3D Engine. This traction has come from its rich feature set, high accuracy physics, strong CPU/GPU optimisations, large community and widely-permissible licensing. The engine itself is lightweight and flexible and is therefore seeing increasing use in tangential domains such as high-fidelity robotics simulation, medical simulation and scientific visualisation applications.

Havok Physics [17]

Havok Physics is a proprietary physics engine used across many proprietary game engines such as Valve's Source Engine, Ubisoft's AnvilNext Engine, Electronic Art's Frostbite Engine, Activision's IW Engine and recently Unity as a result of Unity's undergoing data-oriented rewrite. Havok Physics provides many of the same features as PhysX such as accurate rigid-body mechanics and hardware optimisations but is sometimes opted for over PhysX due to its easy integration with other Havok middleware.

Bullet [18]

Bullet is a less well known physics engine that is both free and fully open-source, originally developed by Erwin Coumans whilst at Sony Computer Entertainment [19]. As a general-purpose engine it has seen use in many games and is increasingly being used as a alternative to the less actively maintained Open Dynamics Engine [20]. Now Bullet is very much at the for-front of reinforcement learning and robotics simulation, used by countless researchers particularly in the field of quadrupedal robotics.

CryPhysics [21]

CryPhysics is the physics engine used in Crytek's popular game engine CryEngine as well as Amazon's only recently deprecated Amazon Lumberyard engine. CryPhyics has already seen use in many VR experiences, for example, The Climb game series developed alongside Oculus Studios where players attempt to scale massive structures like skyscrapers with movable handholds in VR, received praise in its most recent installment for its "real physical reactivity" [22].

Notable Proprietary Physics Subsystems

Unreal Engine's Chaos Physics [23]

Still in development, Unreal Engine's Chaos Physics is a new lightweight physics engine slated to replace PhysX in future Unreal Engine releases. Chaos Physics has seen its debut in the popular online multi-player game Fortnite, where it is being used as a testbed for high accuracy physics with many networked clients. Advancements over existing solutions include a faster physics solver, increased joint stability for ragdolls and integrated cloth mechanics.

Valve's Rubikon

As part of Valve Corporation's Source 2 game engine, Valve have been developing their in-house Rubikon physics engine [24]. Currently unavailable to the public, more implementation specifics are expected with the general release of Source 2. This engine saw its debut in the milestone VR game Half Life: Alyx, providing quick update loops and high accuracy rigid-body physics with additional support for fluid and body-coupling mechanics.

1: Haptics rendering

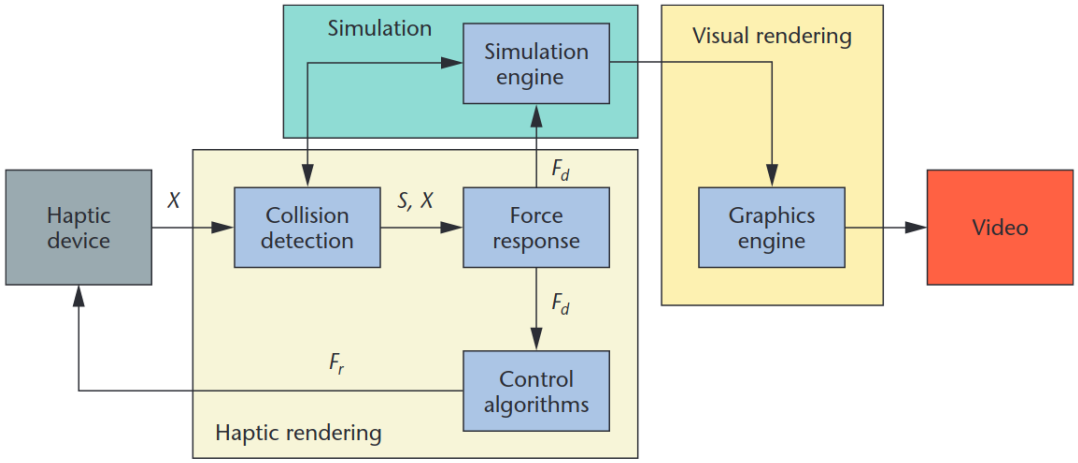

Figure 4: Haptics rendering when integrating with modules of a typical game engine. The *simulation* module is representative of a what has so far been defined as a physics engine. [25]

Haptic rendering is the process of computing and generating forces in response to user and robot interactions with virtual objects [26]. Unlike other forms of rendering such as visual and audio, haptic rendering is uniquely bi-directional due to the position-force feedback loop created from using a virtual environment and a haptic device in tandem. This technology is still very much in its infancy with respect to both the haptic devices being used and the way in which haptics are rendered on them in parallel with VR. Truly immersive VR, a north-star goal for the VR community, will need a strong foundation in haptics. For example in 2021 Meta outlined a 10-year vision for contextually-aware, AI-powered interface for augmented reality (AR) core to which was creating ultra-low-friction input through improved haptics [27]. While haptic devices fall out of scope for this work, high-fidelity haptic rendering is, as Figure 4 shows, reliant on a accurate physics engine.

An important consideration for any rendering system is the Nyquist theorem, which states any kind of sampling needs to happen at twice the frequency we desire for there to be no information loss. For example, the maximum perceivable frequency human eyes can detect is around 20Hz hence we need a sampling frequency and hence a refresh rate 40Hz+ [28]. Similarly this principle applies to haptic rendering where human tactile feedback is vastly more sensitive, thereby demanding a refresh rate of about 1kHz [26]. This requires that haptic rendering operates at a much higher frequency than other rendering components, in particular higher than is expected of a commodity physics engine, for example, PhsyX typically runs at a simulation rate of 60Hz [29].

(a) (b)

Figure 5: Examples of haptic devices. (a) Meta's early stage haptic glove. (b) TouchX haptic device by 3D Systems. [30, 31]

Additionally, physics engines often do not provide the raw force data used in the calculation of physics and instead only provide final entity kinematics for a given simulation step. Although some open source systems like Bullet provide developers access to this information, this is not the case for more popular engines like PhysX and Havok. Without force data from collisions it is hard to generate the necessary force response required in the haptic rendering loop as seen in Figure 4.

Attempts have been made to resolve these issues, for example, Choi et al. implement an intermediary Force Simulation Layer which acts as a data buffer between the physics engine and haptics renderer in the context of surgery simulation [32]. This intermediate layer manages the discrepancy in update rate between PhysX at 60Hz and the haptics renderer (OpenHaptics) at 1kHz as well as reverse engineers historic kinematics data to determine the underlying forces calculated by PhysX for force response. Maciel et al. adopt a similar approach and successfully orchestrate PhsyX and OpenHaptics together using a Model View Controller (MVC) design pattern, as commonly found in the web-development architecture [33].

Although these work-arounds allow for the use of modern general-purpose physics engines such as PhysX they do so with some information loss due to the discrepancy in update rate. This ultimately limits the accuracy with which haptics can be rendered, relying on approximations with stale force data to generate less than ideal tactile response. Future adaptations to physics engines will likely need to consider haptics rendering as a core part of their architecture, making force data more extensible and optimise existing solvers, collision detection algorithms and caching techniques to achieve native update rates of up to 1kHz.

2: Simulation to real world applications (Sim2Real)

Increasingly VR is used as a medium to interact with a virtual environment that is a digital twin of a real environment. The most commonly cited example is surgery simulation, where surgical procedures are simulated in the virtual environment with as accurate as possible physics to the real procedure, and through which an operator uses a HMD to perceive the environment. In the reinforcement learning and robotics community this kind of digital to real coupling is referred to as Sim2Real: simulation to real-world, and this terminology is borrowed for the remainder of this work to refer to the simulation to real-world coupling more generally.

We first analyse the use of sandbox VR training simulations that emulate real-world tasking. These simulations are usually designed for participants to learn and refine their craft or skill in a low-risk environment without repetition limit. Applied contexts range from surgical operations to niche industrial manual-labour based tasks such as system inspection and complex part assembly [34].

As these simulations are designed with the expectation that experience garnered in simulation would transfer to the real-world and yield better operator performance, the simulation physics must therefore be accurate enough to be representative of the real world. Although there exists a strong body of research around Sim2Real in psychological and social settings such as exposure therapy, these contexts benefit from greater visual realism than higher fidelity physics simulation and are hence excluded from this work's consideration.

(a) (b)

Figure 6: VR-based simulation for training demolition robot operators. (a) Demolition robot in simulated construction environment. (b) Real demolition robot being operated. [35]

Adamit et al. create a VR based training simulation of a demolition robot for construction workers that mimics an actual demolition robot which is later used as a benchmark for operator performance [35]. This virtual environment was created using only the Unity game engine and leveraging PhsyX to simulate physics. The authors show an improvement in operator safety protocol when using the simulation prior to operating the robot (see Figure 6) but limited their experiments to the demolition of simple brittle objects such as breeze blocks. More physically challenging objects to simulate such as glass, road-surface and softer-body materials like metal are excluded leaving the simulation not fully representative of all scenarios the robot and operator might face.

Surgical procedures, another common Sim2Real domain, are notoriously hard to simulate due to the very high precision physics simulation needed. Consequently, many surgical training applications

avoid the commodity physics engine found in game engines and use specialised simulators like dV-Trainer [36] and da Vinci Skills Simulator (dVSS) [37]. These simulators instead provide very high precision solvers with strong soft-body and fluid dynamics support [38].

The second area of Sim2Real that attention is drawn to is the increasing use of VR as a proxy for experimental validation of research intended for real world, which has only accelerated as a result of recent work-from-home restrictions. This has varied from testing safety improvements on heavy equipment to the efficacy of co-bot setups in manufacturing all using the physics on offer in modern game engines [39]. Such practises have shown promise in providing a sufficient proxy for real world experimental validation but there has yet to be comprehensive work highlighting the pitfalls in physics simulation for such scenarios.

It is evident that from the work done to simulate real world environments in VR, many applications have been successful in relying on the physics simulation produced from modern game engines. However, more challenging domains like surgery and those requiring advanced non-rigid-body dynamics are still out reach for such engines. For example, there exist many VR training simulations for electricians but few for plumbers which is in part due to the added complexity of simulating accurate fluid dynamics within real-world piping setups.

3: Mixed and augmented reality

Unlike VR, mixed reality (MR) using pass-through and augmented reality both overlay rendered virtual environments onto the real world. This creates a unique opportunity but also adds the challenge of syncing a simulation's behaviour with the behaviour perceived in real world. One of the most challenging aspects of to this is the physics component. When two objects, say tennis balls, one real and one virtual are dropped from the same height and perspective to an AR user, will they respond the same? Current physics engines tell us they are likely not too as there still exists a notable discrepancy between the real and virtual physics simulation which will quickly break a user's immersion.

Figure 7: Niantic's Lightship ARDK provides mobile developers with an easy to implement AR-physics solution using PhysX as its backbone, here virtual die are thrown producing basic elastic collisions with surfaces in the real world [40]

This discrepancy can arise for a multitude of reasons: additional forces on a real world object that are not accounted for in the simulation, insufficient simulation rate providing stuttered trajectory profiles versus a fully continuous world's physics or even poor material property mapping between a real and virtual entity.

Work has been done to reconcile the discrepancy creep in simultaneous virtual and real physics, Kim et al. attempt to modify the physics update loop with a collision prediction module making physics more anticipatory of changes in the real world [41]. Beaney et al. use a landmark based method quickly reconcile creeping differences in collisions between the real and virtual world using the Havok physics [42]. More generally there exists a body of research around object classification in scenes for the purposes of material property inference of the real world, making collisions more informed. These areas of research show promise at producing non-immersion-breaking physics but require significant modifications to exiting physics engines to do so.

Nevertheless it is clear that there exists many applications, particularly in AR games such as Niantic's offerings like Pokemon Go, that prove game-oriented physics engines can be used successfully. However, many of the simulations produced are very basic and use highly elastic rigid-body collision approximations making the generated physics seem cartoon-like. Current physics engines will need to build out native MR and AR modules to allow for more realistic real and virtual physics interoperability. This has been a recent focus for mobile based AR software development kits such as ARKit (for iOS) and ARCore (for Android) [43].

4: VR on low-compute devices

As with many devices there has also been a consistent trend for HMD-based VR devices to become lighter and more ergonomic while still being able to run a high-quality virtual environment. This of course results in a trade-off between compute power and device form-factor. Increasingly lightweight game engines are needed to accompany the shift to low-compute devices enforced by more ergonomic designs and similarly physics engines must also follow suit.

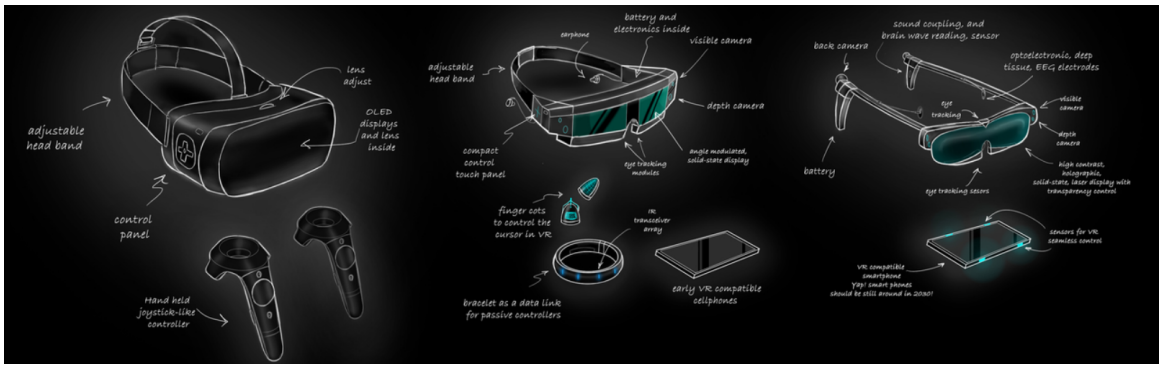

Figure 8: Conceptualised reduction in HMD form-factor. [44]

With less compute on offer lightweight physic engines like those created by Park et al. [45] manage to achieve real-time physics with less compute by approximating away complexity, thereby expensing the reduced compute on lower accuracy physics simulation. Here work around creating computational methods optimised for specific physics calculations provide an avenue to improve the physics engine accuracy-compute trade-off. Additionally some engines like CryPhysics perform minifica-

tion routines at build-time in order to shed unnecessary functionality and reduce the computational demands of the physics engine [21].

But increasingly this constraint is being solved through the use of low-latency, high-bandwith wireless connections which allow high quality multi-platform render streaming. Already there exist many production ready services like Google's Stadia, Amazon's Luna and Microsoft's xCloud, and bar from making processors more efficient provides a possible solution to the low-compute problem. In Friston et al. the edge-computing paradigm is examined in the context of VR simulations and 'edge-physics' proposed, which performs scenegraph streaming from a local server running a Unity application and offloads CPU/GPU intensive workloads in real-time when client device power limits are reached [46].

As a result of these proven alternatives it is likely that in order to solve future challenges concerning accurate physics on low-compute VR devices, the effectiveness of methods like render streaming will be improved over the reduction in physics engine complexity. From this we can conclude that game-oriented physics engines have already proven sufficient for such streaming applications and are therefore well prepared for future on-board compute-power reductions.

5: Highly immersive experiences

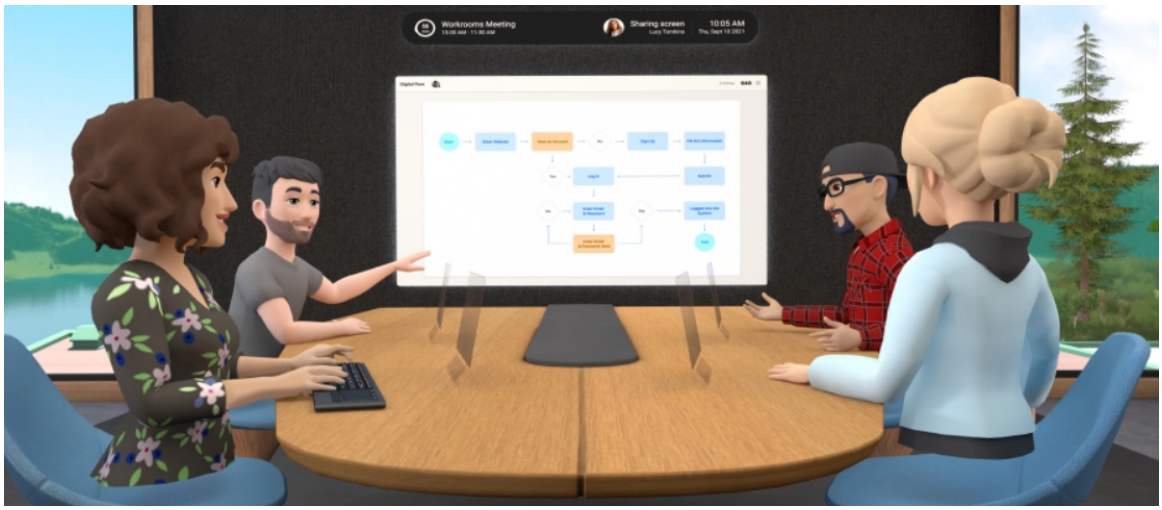

Figure 9: Meta's Horizon Workrooms, an immersive virtual reality solution to virtual meetings [47]

The final future use-case we analyse is perhaps the most obvious, highly immersive experiences, which encompasses any VR experience where embodiment and immersion are the key criteria through which the experience is assessed and unlike Sim2Real, not necessarily how objectively true-to-life the experience is. This can present in a variety of forms, from a non-euclidean sandbox experience to a highly-immersive VR single player game. Consequently, mirror physics of the real world are not strictly necessary but should be 'perceptually' correct enough to create strong cognitive illusion for users.

One key aspect of achieving an immersive experience is the quality of human computer interaction (HCI) with the virtual environment. When framing HCI, the context can fall into two categories: natural interactions and 'magical' interactions. Natural interactions are those analogous to what

would be expected in the real world and therefore set a high initial expectation on the perceived familiarity of them. 'Magicial' interactions are those where the user is given superpowers, for example the ability to fly, and therefore take time for the user to learn but also providing tolerance for experimental interfacing [48]. In both cases HCI provides a powerful opportunity for the experience designers to elicit feelings of embodiment for the user. Importantly, the strength of these feelings are closely correlated to the degree to which the physical response generated in the virtual environment matches a user's expectations.

A recent successful example where both facets of HCI were leveraged to create strong immersion was Valve's survival horror VR game, Half Life: Alyx. Here the player is character Alyx Vance who is a human trapped in a dystopian world where technology such as "gravity gloves" have been made allowing the user to supernaturally snatch objects from a distance, but otherwise following the mechanics of the real world [1]. Through analysis of basic HCI tasks in-game such as locomotion, selection and manipulation it is evident that accurate physics played a key role in cementing the fidelity of each interaction.

Figure 10: Series of keyframes demonstrating a physics-intensive manipulation interaction in Half Life: Alyx. (Top Left) The obstacle in the player's way is a door with a bar jammed between the handles. (Top Right) The player can reach through the window to inspect the obstacle. (Bottom Keyframes) The player reaches through the window then grasps the pipe and removes it to slide open the door [49].

For example, in Figure 10 we analyse a manipulation task of removing a pipe that has jammed a door in the way of the user's path. Without accurate rigid-body physics and precise constraint enforcement, as simulated by Valve's Rubikon physics engine, the process to dislodge the pipe from door handles would be trivial and unnatural thereby breaking immersion. Instead the player has to approach the pipe parallel to its resting position, first slide it out and then rotate and navigate it through the small opening provided by the window before being able to slide open the door.

Additionally, VR games like Boneworks by Stress Level Zero have proven successful at achieving immersive VR experiences involving physics puzzles simulated by commodity physics engines. In

Boneworks users must escape and battle their way out of a virtual city, revolving around realistic physics interactions [50]. Boneworks is built ontop of Unity with custom tooling around PhysX to support some of the novel physics puzzles implemented in the game [51].

Although, we demonstrate novel and promising immersive experiences that use game-oriented physics engines these simulations rely solely on rigid-body physics. These experiences make no attempt at simulating soft-body physics which is a crucial aspect to accurately simulating the way objects deform, for example, squeezing a tennis ball.

Apart from immersive gaming experiences, there is an increasing push around the immersive VR workplace experience. Meta's Horizon workplace (see Figure 9) is a prime example, laying out a vision where people use HMD to join virtual meetings in virtual settings in place of existing video-conferencing solutions [47].

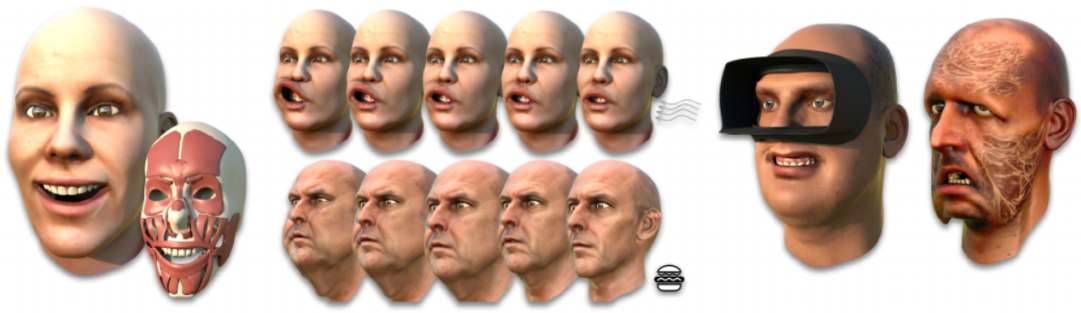

Figure 11: Phace: Physics-based simulation facilitates a number of advanced effects for facial animation, such as applying wind forces, fattening and slimming of the face, wearing a VR headset, and even turning into a zombie. [52]

One barrier to realising this vision has been rendering accurate facial expressions between conversing parties wearing a HMD device. With upwards of 30 muscles in the face [53], accurately modeling the physical changes of a persons facial expression in a photo-realistic way is still out of reach for many physics engines. Should we want photo-realistic rendering game-oriented physics engine will need to add increased support or sub-modules specifically for accurate facial expression modelling. However, a lot of the work around cloth dynamics could prove a useful foundation for enabling this in the future.

Conclusion

The objective of this work was to formulate a conclusion about the current state of game-oriented physics engines for future use in VR. To do this the fundamentals of how a physics engine works are discussed and a brief survey is undertaken of the most popular physics engines found in modern games. After understanding the current state of game-oriented physics engines they are assessed in the context of five key emerging areas of VR: 1) Haptics rendering, 2) Simulation to real world applications (Sim2Real), 3) Mixed and augmented reality, 4) Virtual reality on low-compute devices, and 5) Highly immersive experiences.

Through this technical analysis we begin to understand there exist a number of pitfalls that need addressing if the physics engines found in modern games are to be used in future VR applications. With regard to haptics rendering, the difference in simulation speed of a game-oriented physics

engine is an order of 10 away from the necessary speeds haptics rendering requires. Sim2Real learning experiences powered by commodity physics engines teach basic skills but still are sparsely found in more technically challenging domains to simulate like plumbing and surgery. Although there exist many AR and MR applications that use physics simulation from game engines overlayed on the real world, most of these simulation use very elastic cartoon-like collisions in place of expected real world behaviour. However, we do show techniques to simulate physics accurately on resource constrained devices through networked solutions such as edge-computing and current engines provide physics simulations enough to for highly immersive gaming and social experiences, but with provide some ideas for improvement as well.

Overall, the physics simulations found in modern games platforms do simulate physics accurately enough, but for future applications there is more work to be done. From this analysis we identify three suggestions:

- Future physics engines will need modules that simulate soft-body dynamics and fluid dynamics as well as current engines simulate rigid-body dynamics.
- Future physics engines will need to work faster, operating at a higher refresh rate to improve fidelity and syncing with haptic devices.
- Future physics engines could benefit from the ability to learn the physical laws of the environment it is simulating, potentially through techniques such as reinforcement learning and evolutionary programming.

Basic ideas to achieve such improvements may include making differential calculus easier to perform in these simulators or leaning on more scientific physics engines as the groundwork for future VR-focused physics engines.

References

[1] Valve Corporation. (2022) Half-life: Alyx. [Online]. Available: https://store.steampowered.com/app/546560/HalfLife_Alyx

[2] L. Kugler, "The state of virtual reality hardware," *Commun. ACM*, vol. 64, no. 2, p. 15–16, jan 2021. [Online]. Available: https://doi.org/10.1145/3441290

[3] M. Toftedahl and H. Engström, "A taxonomy of game engines and the tools that drive the industry," in *DiGRA 2019, The 12th Digital Games Research Association Conference, Kyoto, Japan, August, 6-10, 2019*. Digital Games Research Association (DiGRA), 2019.

[4] H. H. Goldstine and A. Goldstine, "The electronic numerical integrator and computer (eniac)," *IEEE Annals of the History of Computing*, vol. 18, no. 1, pp. 10–16, 1996.

[5] B. Laurell, "The inner workings of real-time physics simulation engines," in *IRCSE'08 IDT workshop on interesting results in computer science and engineering*. Citeseer, 2008.

[6] M. R. Desselle, R. A. Brown, A. R. James, M. J. Midwinter, S. K. Powell, and M. A. Woodruff, "Augmented and virtual reality in surgery," *Computing in Science & Engineering*, vol. 22, no. 3, pp. 18–26, 2020.

[7] Unity. (2022) Unity game report 2022. [Online]. Available: https://create.unity.com/gaming-report-2022

[8] Unity Technologies. (2022) Physics in unity 5.0. [Online]. Available: https://docs.unity3d.com/Manual/UpgradeGuide5-Physics.html

[9] C. Ericson, *Real-time collision detection*. Crc Press, 2004.

[10] H. Serrano. (2016) How a physics engine works an overview. [Online]. Available: https://www.haroldserrano.com/blog/how-a-physics-engine-works-an-overview

[11] P. J. Davis and P. Rabinowitz, *Methods of numerical integration*. Courier Corporation, 2007.

[12] P. Terdiman, "Memory-optimized bounding-volume hierarchies," *URL www. codercorner. com/Opcode. pdf*, 2001.

[13] C. J. Ong and E. G. Gilbert, "The gilbert-johnson-keerthi distance algorithm: A fast version for incremental motions," in *Proceedings of International Conference on Robotics and Automation*, vol. 2. IEEE, 1997, pp. 1183–1189.

[14] H. Serrano. (2015) How to build a game engine? [Online]. Available: https://www.haroldserrano.com/blog/how-do-i-build-a-game-engine

[15] T. Erez, Y. Tassa, and E. Todorov, "Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ode and physx," in *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2015, pp. 4397–4404.

[16] NVIDIA Corporation. (2022) Nvidia physx. [Online]. Available: https://developer.nvidia.com/physx-sdk

[17] Havok. (2022) Havok physics. [Online]. Available: https://www.havok.com/havok-physics/

[18] Bullet Physics. (2022) Bullet physics. [Online]. Available: https://bulletphysics.org

[19] J. Thacker. (2015) Bullet and naiad creators win academy awards. [Online]. Available: http://www.cgchannel.com/2015/01/bullet-naiad-and-speedtree-creators-win-academy-awards

[20] R. Smith. (2022) Open dynamics engine. [Online]. Available: https://www.ode.org/

[21] Crytek. (2022) Cryphysics. [Online]. Available: https://docs.cryengine.com/display/CEPROG/CryPhysics

[22] IGN. (2022) The climb 2 review. [Online]. Available: https://www.ign.com/articles/the-climb-2-review

[23] Epic Games. (2022) Chaos physics overview. [Online]. Available: https://docs.unrealengine.com/4.26/en-US/InteractiveExperiences/Physics/ChaosPhysics/Overview/

[24] S. Migdalskiy, "Physics for game programmers debugging (game developers conference)," 2014. [Online]. Available: http://media.steampowered.com/apps/valve/2014/Sergiy_Migdalskiy_Debugging_Techniques.pdf

[25] K. Salisbury, F. Conti, and F. Barbagli, "Haptics rendering: Introductory concepts," *IEEE computer graphics and applications*, vol. 24, no. ARTICLE, pp. 24–32, 2004.

[26] C. Basdogan and M. A. Srinivasan, "Haptic rendering in virtual environments," *Handbook of virtual environments*, vol. 1, pp. 117–134, 2002.

[27] Meta. (2021) Inside facebook reality labs: The next era of human-computer interaction. [Online]. Available: https://tech.fb.com/ar-vr/2021/03/inside-facebook-reality-labs-the-next-era-of-human-computer-interaction/

[28] M. F. Deering, "The limits of human vision," in *2nd International Immersive Projection Technology Workshop*, vol. 2, 1998, p. 1.

[29] NVIDIA Corporation. (2022) Nvidia physx best practices. [Online]. Available: https://docs.nvidia.com/gameworks/content/gameworkslibrary/physx/guide/Manual/BestPractices.html

[30] Meta. (2021) Inside reality labs research: Bringing touch to the virtual world. [Online]. Available: https://about.fb.com/news/2021/11/reality-labs-haptic-gloves-research/

[31] 3D Systems. (2021) Touch x haptic device. [Online]. Available: https://www.3dsystems.com/haptics-devices/touch-x

[32] K.-S. Choi, L. S. Chan, J. Qin, and W.-M. Pang, "Haptic rendering in interactive applications developed with commodity physics engine," *Journal of Multimedia*, vol. 6, no. 2, pp. 147–155, 04 2011, copyright - Copyright Academy Publisher Apr 2011; Last updated - 2011-10-04. [Online]. Available: https://www.proquest.com/scholarly-journals/haptic-rendering-interactive-applications/docview/863204668/se-2?accountid=14511

[33] A. Maciel, T. Halic, Z. Lu, L. P. Nedel, and S. De, "Using the physx engine for physics-based virtual surgery with force feedback," *The International Journal of Medical Robotics and Computer Assisted Surgery*, vol. 5, no. 3, pp. 341–353, 2009. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rcs.266

[34] M. Murcia-Lopez and A. Steed, "A comparison of virtual and physical training transfer of bimanual assembly tasks," *IEEE transactions on visualization and computer graphics*, vol. 24, no. 4, pp. 1574–1583, 2018.

[35] P. Adami, P. B. Rodrigues, P. J. Woods, B. Becerik-Gerber, L. Soibelman, Y. Copur-Gencturk, and G. Lucas, "Effectiveness of vr-based training on improving construction workers' knowledge, skills, and safety behavior in robotic teleoperation," *Advanced Engineering Informatics*, vol. 50, p. 101431, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S147403462100183X

[36] Mimic Simulation. (2014) dv-trainer. [Online]. Available: http://www.mimicsimulation.com/products/dv-trainer/

[37] Intuitive Surgical. (2022) da vinci skills simulator. [Online]. Available: https://www.intuitivesurgical.com/products/skills_simulator/index.php

[38] A. Moglia, V. Ferrari, L. Morelli, M. Ferrari, F. Mosca, and A. Cuschieri, "A systematic review of virtual reality simulators for robot-assisted surgery," *European Urology*, vol. 69, no. 6, pp. 1065–1080, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S030228381500929X

[39] E. Matsas and G.-C. Vosniakos, "Design of a virtual reality training system for human–robot collaboration in manufacturing tasks," *International Journal on Interactive Design and Manufacturing (IJIDeM)*, vol. 11, no. 2, pp. 139–153, 2017.

[40] Niantic. (2022) Niantic lightship ardk. [Online]. Available: https://lightship.dev/

[41] S. Kim, Y. Kim, and S.-H. Lee, "On visual artifacts of physics simulation in augmented reality environment," in *2011 International Symposium on Ubiquitous Virtual Reality*. IEEE, 2011, pp. 25–28.

[42] D. Beaney and B. Mac Namee, "Forked! a demonstration of physics realism in augmented reality," in *2009 8th IEEE International Symposium on Mixed and Augmented Reality*. IEEE, 2009, pp. 171–172.

[43] Android. (2022) Whats new in arcore. [Online]. Available: https://developers.google.com/ar/whatsnew-arcore

[44] Future Avenues. (2022) Vr future headsets concept. [Online]. Available: https://mit2.cgsociety.org/

[45] H. C. Park and N. Baek, "Design of selfengine: A lightweight game engine," in *Information Science and Applications*. Springer, 2020, pp. 223–227.

[46] S. Friston, E. Griffith, D. Swapp, C. Lrondi, F. Jjunju, R. Ward, A. Marshall, and A. Steed, "Quality of service impact on edge physics simulations for vr," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 5, pp. 2691–2701, 2021.

[47] Meta. (2021) Introducing horizon workrooms: Remote collaboration reimagined. [Online]. Available: https://about.fb.com/news/2021/08/introducing-horizon-workrooms-remote-collaboration-reimagined/

[48] A. Dix, J. Finlay, G. D. Abowd, and R. Beale, *Human-computer interaction*. Pearson Education, 2004.

[49] Valve Corporation, "Half-life: Alyx gameplay video 1," 2020. [Online]. Available: https://www.youtube.com/watch?v=LTLotwKpLgk

[50] Stress Level Zero. (2022) Boneworks. [Online]. Available: https://store.steampowered.com/app/823500/BONEWORKS/

[51] UploadVR. (2019) Boneworks review: A stunning showcase of physical interaction that tests vr's limits. [Online]. Available: https://uploadvr.com/boneworks-review/

[52] A.-E. Ichim, P. Kadleček, L. Kavan, and M. Pauly, "Phace: Physics-based face modeling and animation," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, pp. 1–14, 2017.

[53] K. E. Westbrook, T. A. Nessel, M. H. Hohman, and M. Varacallo, "Anatomy, head and neck, facial muscles," in *StatPearls [Internet]*. StatPearls Publishing, 2021.