Leveraging Game Engines and Data Interchange Formats to Revolutionize Distributed Training

Jerry Huggins, Tim Rodabaugh
Ball Aerospace
UNITED STATES

Jerry.Huggins@ballaerospace.com Timothy.Rodabaugh@ballaerospace.com

Quintin Oliver
AFRL 711 HPW/RHW
UNITED STATES

quintin.oliver@us.af.mil

ABSTRACT

Commercial-off-the-shelf game engines, such as Unreal Engine and Unity, are rapidly expanding their capabilities and becoming an attractive option for modelling and simulation. Game engines have several benefits such as cost and licensing, first-class support for immersive display technologies, development pipelines geared around speed, and rich ecosystems. However, game engines lack physics fidelity, introduce coordinate system disagreements, and do not natively support military standard data interchange formats. Support for data interchange formats such as DIS and HLA is utmost if game engines are to be used in a distributed, networked training enterprise. While commercial data interchange plugins exist, they are expensive and difficult to modify and use. The Gaming Research Integration for Learning Laboratory® (GRILL®) developed a free and open source DIS plugin for both Unreal Engine and Unity which bridges the gap between game engines and the modelling and simulation enterprise. A rich framework was also developed whereby the plugin can be readily expanded to support additional standards such as HLA. Through the use of DIS, one of the primary drawbacks of utilizing game engines for modelling and simulation can be addressed. With help from the community, the plugin could become the industry standard, supporting many data formats.

1.0 INTRODUCTION

Commercial-off-the-shelf game engines, such as Unreal Engine and Unity, are rapidly expanding their capabilities and becoming an attractive option for modelling and simulation (M&S). Game engines are a software framework that supply a suite of tools to ease development of games. Tools provided by game engines typically include a physics engine, media engines for audio and video, developer tools with access to programming languages, character and animation creation tools, lighting and rendering engines, cross-platform and immersive display technology tools, and world and scenario editor tools. In addition to these tools, game engines have other benefits including a low cost of entry, first-class support for immersive display technologies, development pipelines geared around speed, and rich developer ecosystems. However, they are not without their drawbacks as they lack physics fidelity, do not communicate with non-game engines well, operate with flat and often limited coordinate systems, and lack some tools that are desirable inside of a distributed training environment such as out of the box round Earth models and 64-bit location accuracy. The following sections will outline key benefits and deficits of game engines for use within a distributed training environment, how to mitigate drawbacks, and discuss how support for data interchange formats overcome some of the deficits.

2.0 BENEFITS

2.1 Cost/Licensing

M&S software often requires either a one-time purchase cost or a recurring licensing fee. Although game engines are not free of costs, most are free to download and begin using for small teams and projects, hobbyists, and educational use. End-users of game engines will often create projects with no costs incurred. Costs for use of game engines can be incurred when revenue thresholds are exceeded such that royalties must be paid or if additional features require a license. In contrast to Unreal Engine, Unity does not collect royalties. However, Unity does have different license levels available for purchase. For example, Unity's source code can only be accessed with the Unity Enterprise or Unity Industry version of their license [31].

On the other hand, Unreal Engine utilizes royalties. If a product made with Unreal Engine earns more than $1,000,000 USD in its lifetime, a 5% revenue royalty must be paid [13]. Unreal Engine also offers licenses for purchase, but these licenses do not unlock additional engine capabilities like the Unity licenses. Licenses from Epic Games afford private training sessions and access to Epic Games' private Perforce depot [12]. On top of the low cost of entry, game engines become an even more attractive option once the ecosystem surrounding them is considered.

2.2 Ecosystem

Development within game engines is greatly aided by a rich ecosystem of ready-made content, support forums, documentation, and a rich labor pool [21]. The ecosystem is mostly non-proprietary and is accessible from anywhere. This improves the quality of products created and the speed at which products are developed for distributed training. The large labor pool surrounding game engines means developers are readily available for hire while third-part content, plugins, and support forums are pervasive and plentiful. Content can be of varying quality and varying cost but is most often made available by the engine via a sanctioned marketplace or posted to sites like GitHub.

Both Unreal Engine and Unity have their own developer community forums [14][35]. Developers of the game engines and third-party developers collaborate on these sites to troubleshoot issues and share useful information. Documentation includes API references, tutorials, and helpful tips and tricks. By leveraging both the community forums and documentation, most issues encountered during development can be solved. A robust ecosystem comprised of a rich labor pool, ready-made content, plugins, documentation, and support forums form a compelling case to conduct M&S development using game engines.

2.3 Out of Box Capabilities

M&S software is typically characterized by a visual engine and development tools used to customize functionality. Since traditional M&S software is proprietary, ecosystem fragmentation, incompatibilities, and proprietary formats and standards abound. In contrast, mainstream game engines have a rich set of tools that frequently cover industry standard formats and are designed from the start with customization in mind. Game engines have myriad tools and capabilities making them attractive for M&S such as compelling tool chains for physics, audio and video, scripting and software development, animation, immersive display integration and many others [38]. Foremost of those capabilities is visual fidelity and scripting.

Modern video games are geared around visual fidelity so engines have evolved a rich pipeline geared around realistic visuals and optimization of their display at high frame rates. Visuals have become so compelling that the engines themselves are frequently employed for architectural visualization and in movie production [23]. Rendering pipelines and toolchains for 3D content, lighting, animation, and other rendering facets are robust, mature, and well-optimized. Support for common standards, such as 3D model files, allow for almost any third-party 3D modelling tool to be used [34]. Plugins also abound to support direct integration with third-party 3D modelling tools. Support for a wide range of applications and standards enables content developers to freely use their tools of choice. Quickly crafting compelling and realistic visuals is a hallmark of modern game engines and one in which they excel as shown in Figure 1 below.

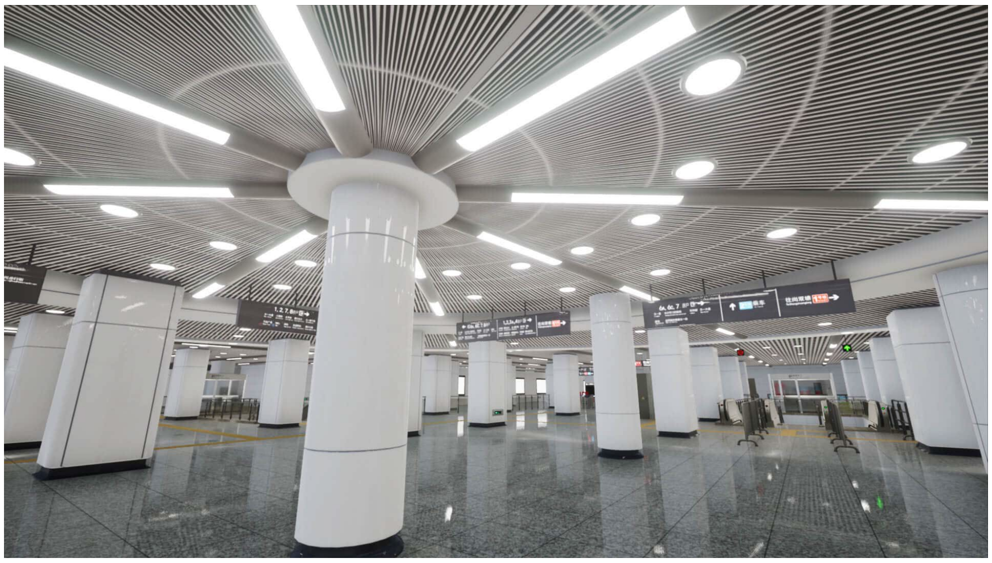

Figure 1: A digital twin of China's Wuyi Square Station made in Unreal Engine [11]

In addition to compelling visuals, mainstream game engines have scripting support that permits functionality to be readily customized, plugins to be developed, and integration of existing software libraries. Unity, for example, supports the C# programming language while Unreal Engine utilizes C++. Both engines have rich mechanisms for developing plugins and custom functionality as evidenced by the wealth of content and plugins available in their respective online marketplaces.

Key to M&S is the integration of existing software systems and libraries. Both Unity and Unreal Engine readily support integration of software libraries either directly through the engine or by way of plugins. Interfacing with a learning management system, for example, is a common use case. Plugins and direct support for standards such as SCORM, xAPI, and web service calls are pervasive for both Unity and Unreal Engine [37]. Since a significant portion of M&S efforts revolves around development and integration of software, game engines are well-positioned as an efficient, flexible, and adaptable M&S software environment.

2.4 Immersive Display Technology Support

Extended reality (XR) is commonly used in training and educational fields [5] as the increased immersion can lead to better muscle memory retention compared to desktop training [3]. Research has also revealed on-the-job applications, especially in maintenance and medical fields, increase employee performance and decrease workplace mishaps [2][24]. In terms of distributed training, XR devices can be used to make simulations more immersive. For example, the virtual and real world can be blended to allow a trainee to interact physical input devices while still being immersed in a virtual environment [17]. Direct support for XR devices and associated standards is included with most mainstream game engines. Both Unity and

Unreal Engine supply virtual reality (VR) and augmented reality (AR) project templates with which new XR projects can be started. Alternatively, existing, non-XR, projects can be updated to utilize XR by enabling built-in XR plugins or settings [16][36].

2.5 Cross-Platform Support

Training on the edge and on demand is becoming increasingly important. User's often want their training available on-demand, whether on their mobile devices or on a local computer, and on the newest of devices. Game engines help to meet this need as software created inside of them can be built to run on a multitude of different platforms from Windows, macOS, and Linux to gaming consoles, handheld devices, and wearable devices [7][30]. Due to the number of supported platforms within game engines, they are attractive if cross platform support is important. The game engines' embedded tools allow the fast generation of multiple executables for the same project. These tools also make cross-platform development quick and easy. For example, User Interfaces (UI) inside of Unreal Engine and Unity come with features that largely automate scaling and display of UI elements across platforms and devices. [15][29]. Different input modalities are also supported within game engines allowing input from devices such as a joystick, keyboard, mouse, and touchscreen to be captured regardless of the device being used.

3.0 DEFICITS

3.1 Distribution

Traditional training based around video or web content can be easily centralized, updated, and integrated with learning management systems. Executables and packages, however, produced by game engines, must be distributed to and installed on target devices. Given that most military organizations constrain their information technology systems, distribution, installation, and maintenance of the application and systems in which it resides are of vital consideration. The following are presented as considerations:

- How does a user discover and download the simulation?
- How are updates distributed?
- Who maintains the devices and ensures they have adequate processing capabilities?
- How does the simulation interface with external systems such as learning management systems?
- Finally, how is all of the above accomplished in a remote environment?

While commercial services such as Xbox, Steam, and application stores largely solve the above considerations [26], it remains to be seen how they will solved within a military context. There are several systems that perform similar functions to these commercial application stores in the United States government, but access to these systems is commonly restricted behind an access wall, owned by different government departments, and have different content restrictions. For example, the Army has a MilGaming website that is public facing, but requires an access certificate to access the website. Special authorization is required in order to place content on the website. Additionally, authorization to download content requires submitting usage and intended audience information. Once content is placed on one of these sites, the military machines the software is to be run on have to be navigated. These devices often have their own local restrictions and may not be able to use all elements a simulation has to offer.

3.2 Approval/Clearance

Since many training exercises within distributed training environments occur within sensitive spaces, obtaining approval to run software is an important prerequisite. Since game engines update rapidly and the origins and security of the code, plugins, and various software can be rather nebulous, military organizations can be understandably hesitant to readily approve the software in a secure environment. Additionally, the nearly limitless combinations of plugins and associated software make it difficult to characterize the security of an executable developed with a game engine. While some engines offer offline installers upon request, game engines are designed to operate optimally with an Internet connection whereby updates and patches are easily installed. Plugins such as Cesium Ion for Unreal Engine that expect or require an Internet connection must be considered and mitigated where possible with offline databases [1]. Careful consideration should be given to the approval process and started as early as feasible should one wish to use a game engine in a sensitive environment requiring strict approvals.

3.3 Geospatial and Round Earth Support

One important aspect to have in distributed training simulations is support for real-world geospatial environments or round Earth. Since game engines employ flat, Cartesian coordinate systems, translations or trade-offs must be made to reconcile coordinate systems [25]. Further, virtual representations of real-world locations can be constructed by hand in a tedious, manual process involving importation, synchronization, and stitching of elevation data and satellite imagery [4]. Automated tools to import geospatial data are becoming more ubiquitous and user friendly. Geospatial products such as ArcGIS and Cesium offer complete geospatial solutions for both Unity and Unreal Engine. These plugins supply 3D tile streaming for satellite imagery with elevation data capabilities to game engines. They also provide tools for anchoring entities in the world, converting between various coordinate systems, and ways to move a user around the round Earth model. With the release of these plugins, making realistic digital twins is becoming even easier to perform inside of game engines due to availability of precise geographic data [18]. While there are still deficits to overcome, game engines are well on their way to supporting and offering solutions for round earth models.

3.4 32 vs 64 bit

Most game engines have 64-bit support through their scripting languages, but lack 64-bit coordinate support to position engine-specific entities, favoring 32-bit support instead, since graphics pipelines are traditionally geared around 32-bit precision [39]. For example, Unreal Engine 4 utilizes C++ as its backend programming language which supports 64-bit precision, but when updating the location of an Unreal Engine entity it only supports 32-bit precision. For most commercial uses of game engines, 32-bit precision is sufficient. However, for a distributed training exercise in a geospatial environment, 32-bit precision is not sufficient to precisely represent locations far from the earth's center.

Thus, 64-bit precision is preferable as geospatial assets need the extra coordinate accuracy [27]. Game engines are starting to trend toward supporting 64-bit precision. Unreal Engine 5 offers 64-bit support for its visualization, simulation, and rendering systems [8] and Unity has a High Precision Framework plugin which supplies 64-bit precision for geographic coordinate support on a global scale [32]. Game engines are increasingly seeing the need for large coordinate support, but there is still progress to be made.

3.5 Physics Fidelity

Physics fidelity of game engines tends to be of average quality as they try to balance realistic physics with performance. Most physics engines employed by game engines can simulate gravity, simple projectile arcs, and simple collisions along with simple impulses, momentum, and movement [10][33]. It is possible to augment a game engine's physics engine with additional custom physics, but doing so is tedious, error prone, and an intensive endeavor. One mitigation is the separation of the physics model from the rendering. For example, Epic Games created the Antoinette Project [6] which utilizes Unreal Engine visuals, but performs flight physics external to Unreal Engine's physics engine through utilization of a physics and math model made by JSBSim [22]. In this case, the JSBSim library is added as a third-party plugin to Unreal Engine and interfaces with it through the C++ developer tools Unreal Engine provides. Another use case would be a standalone physics engine that does all of the processing for realistic physics calculations and sends the results to the engine via a data interchange format, requiring the game engine to only act as the image generator. While the physics engines employed by game engines are becoming increasingly sophisticated, a trade-off is made for the sake of performance.

3.6 Network Communications

Communication from one game engine to another is a straight forward process, but difficulties can arise when trying to communicate with third-party software. Unreal Engine and Unity have built-in networking capabilities and normally rely upon a server-client architecture [9][28]. These networking capabilities allow a project made in one game engine to communicate with multiple instances of itself, but not for a project to communicate with third-party software. Fragmented networking and standards in a distributed training environment is a major drawback as simulations should be able to readily communicate and interoperate. Custom networking capabilities can be developed but these are low level, typically at the socket level, and do not encompass common data interchange formats. To address the lack of a data interchange format, proprietary software needs to be purchased and installed for the game engine or custom functionality coded via the developer tools.

4.0 DATA INTERCHANGE FORMATS

4.1 GRILL® DIS Plugin

The GRILL® DIS plugin is a Distributed Interactive Simulation (DIS) plugin made for both Unreal Engine and Unity that was developed by the Gaming Research Integration for Learning Lab® (GRILL®). The GRILL®'s mission is to conduct integration efforts and training research using game-based technology to increase the operational efficiency and effectiveness of the people defending the United States of America [19]. Game engines are often used by the GRILL® in order to create serious games and synthetic task environments and develop prototypes and training systems. The GRILL® is located at Wright-Patterson Air Force Base where numerous distributed training exercises are conducted. In order to bridge the gap between game engines and existing M&S capabilities, game engines need to support military data interchange formats such as DIS and High Level Architecture (HLA). Not satisfied with the limitations, cost, and lack of customizability of commercial solutions, the GRILL® created a DIS plugin for both Unity and Unreal Engine that is both easy to use and, critically, open source.

The DIS protocol is an IEEE network communication standard and is widely used in M&S to permit network communication between systems. The GRILL® DIS plugin provides basic User Datagram Protocol (UDP) socket support and handles interpretation of DIS network traffic using the Naval Post Graduate School's Open-DIS libraries. Then, the plugin uses the interpreted DIS data to create and update user specified DIS entities in the game engine. The DIS plugin provides an interface for end-users to add custom functionality when specific DIS data is received. The GRILL®'s plugin also supports sending DIS information through utilizing the Open-DIS libraries allowing DIS data to be sent to other DIS-capable simulations. Furthermore, core components of the GRILL® DIS plugin are decoupled to allow for custom functionality to easily be added. If custom UDP socket support, Protocol Data Unit (PDU) processing, or management of DIS entities is desired, the supplied components can easily be replaced by new components. The GRILL® DIS plugin is not coupled to a specific round Earth model and provides its own geospatial conversion functions whether being custom made or wrappers around functionality provided by the game engine. Figure 2 below shows the GRILL® DIS plugin being used in Unreal Engine with the Cesium for Unreal Engine plugin generating the round Earth terrain.

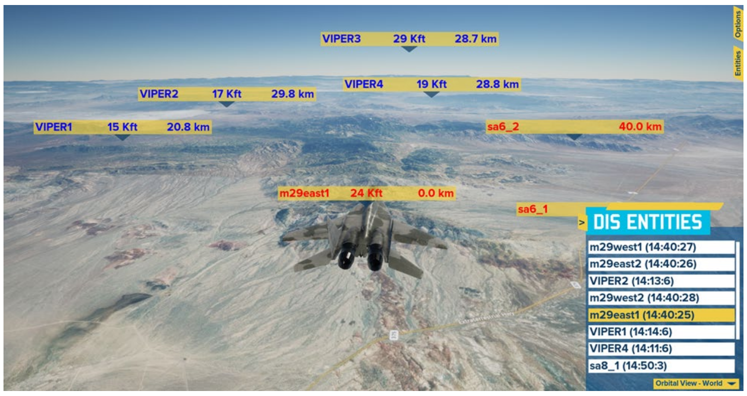

Figure 2: External DIS entities being captured and displayed in Unreal Engine

Through leveraging the DIS data interchange format, some game engine drawbacks were able to be overcome. Network communications with non-game engine software becomes possible as a standardized communication protocol is integrated. Physics fidelity drawbacks are addressed as third-party physics engines can now be utilized for high fidelity physics models and have their information sent to the game engines utilizing DIS. This enables the game engines to act as image generators, employing their state of the art visual fidelity, while keeping their numerous out of the box capabilities, vast ecosystem, and support for immersive displays and a multitude of platforms in the equation.

The GRILL® DIS plugin is still in an early beta phase. It has been made open source on GitHub [20] to allow for the broader community to utilize it and contribute to it. The plugin has also been placed on the respective marketplaces for Unreal Engine and Unity. While core PDUs have been implemented, additional work must be done to implement all PDUs. Additionally, the plugin could be readily expanded to support other standards such as HLA. The GRILL® believes that their plugin could become the new standard for military data interchange within game engines due to the ease of use, flexibility, and expandability afforded by its open source status. We humbly call upon the M&S community to aid us in furthering the development of the plugin.

5.0 CONCLUSION

Through the creation of the DIS plugin, many lessons were learned for how data interchange formats can be used to overcome some of the deficits that game engines face in a distributed training environment. Although data interchange formats do not fix issues with distributing projects or getting projects approved to be in classified environments, they do fix network communication issues with non-game engine software and address some drawback associated with game engine physics fidelity. Data interchange formats provide a standardized way for information to pass between simulations, solving the difficult issue of interoperability with existing M&S systems. Physics issues can be mitigated by using the game engine as an image generator reading DIS data. Game engines already excel at visual fidelity, development pipelines, tool chains, and many other things. By integrating a plugin such as the GRILL DIS Plugin, game engines become a very attractive option even for serious M&S efforts.

Furthermore, tools that support the broader M&S community are constantly being added to game engines to address aforementioned drawbacks. Game engine companies are starting to natively support 64-bit precision or provide tools to do so. Companies like Cesium and ArcGIS have created plugins for game engines that supply support for round Earth models. With these new tools and the use of data interchange formats, a majority of the game engine drawbacks can be overcome, paving the way for game engines to revolutionize distributed training.

6.0 REFERENCES

[1] Cesium. (n.d.). *Cesium On-Premises*. Cesium. https://cesium.com/platform/on-premises-products/

[2] Coupry, C., Noblecourt, S., Richard, P., Baudry, D., & Bigaud, D. (2021, July 24). *BIM-based digital twin and XR devices to improve maintenance procedures in smart buildings: A literature review*. MDPI. https://www.mdpi.com/2076-3417/11/15/6810

[3] Cross, J., Boag-Hodgson, C., Ryley, T., Mavin, T. J., & Potter, L. E. (2022, May 10). *Using Extended Reality in Flight Simulators: A Literature Review*. Using Extended Reality in Flight Simulators: A Literature Review | IEEE Journals & Magazine | IEEE Xplore. https://ieeexplore.ieee.org/document/9772329

[4] Dembski, F., Wössner, U., Letzgus, M., Ruddat, M., & Yamu, C. (2020, March 16). *Urban Digital Twins for Smart Cities and Citizens: The Case Study of Herrenberg, Germany*. MDPI. https://www.mdpi.com/2071-1050/12/6/2307

[5] Doolani, S., Wessels, C., Kanal, V., Sevastopoulos, C., Jaiswal, A., Nambiappan, H., & Makedon, F. (2020, December 10). *A review of Extended reality (XR) technologies for manufacturing training*. MDPI. https://www.mdpi.com/2227-7080/8/4/77

[6] Epic Games. (2022, May 3). *Antoinette Project: tools to create the next generation of flight simulators*. Antoinette Project: tools to create the next generation of flight simulators. https://www.unrealengine.com/en-US/blog/antoinette-project-tools-to-create-the-next-generation-of-flight-simulators

[7] Epic Games. (n.d.). *A world of possibilities*. Game Engine | Build Multi-Platform Video Games – Unreal Engine. https://www.unrealengine.com/en-US/solutions/games

[8] Epic Games. (n.d.). *Large World Coordinates.* Large World Coordinates in Unreal Engine 5 | Unreal Engine Documentation. https://docs.unrealengine.com/5.1/en-US/large-world-coordinates-in-unreal-engine-5/

[9] Epic Games. (n.d.). *Networking Overview*. Networking Overview | Unreal Engine Documentation. https://docs.unrealengine.com/4.27/en-US/InteractiveExperiences/Networking/Overview/

[10] Epic Games. (n.d.). *Physics*. Unreal Engine 4.27 Documentation. https://docs.unrealengine.com/4.27/en-US/InteractiveExperiences/Physics/#:~:text=To%20enhance%20player%20immersion%2C%20Unreal,between%20objects%20within%20the%20world

[11] Epic Games. (2021, September 17). *Remapping China's morning commute with Digital Twins*. Unreal Engine. https://www.unrealengine.com/en-US/spotlights/remapping-china-s-morning-commute-with-digital-twins

[12] Epic Games. (n.d.). *Unreal Engine Licensing Options*. Unreal Engine Licensing Options. https://www.unrealengine.com/en-US/license

[13] Epic Games. (n.d.). *Unreal® Engine End User License Agreement*. EULA - Unreal Engine. https://www.unrealengine.com/en-US/eula/unreal

[14] Epic Games. (n.d.). *Unreal Engine Forum*. Epic Developer Community Forums.

https://forums.unrealengine.com/categories?tag=unreal-engine

[15] Epic Games. (n.d.). *Scaling UI for Different Devices*. Scaling UI for Different Devices in Unreal Engine | Unreal Engine Documentation. https://docs.unrealengine.com/5.1/en-US/scaling-ui-for-different-devices-in-unreal-engine/

[16] Epic Games. (n.d.). *XR Development*. XR Development | Unreal Engine Documentation. https://docs.unrealengine.com/4.27/en-US/SharingAndReleasing/XRDevelopment/

[17] Ernst, J. M., Laudien, T., & Schmerwitz, S. (2023, June 12). *Implementation of a mixed-reality flight simulator: Blending real and virtual with a video-see-through head-mounted display*. SPIE Digital Library. https://www.spiedigitallibrary.org/conference-proceedings-of-spie/12538/125380R/Implementation-of-a-mixed-reality-flight-simulator--blending-real/10.1117/12.2664848.short?SSO=1

[18] Furness, N., Hansen, R. (2022, June 08). *Introducing ArcGIS Maps SDK for Unity*. Introducing ArcGIS Maps SDK for Unity. https://www.esri.com/arcgis-blog/products/unity/announcements/introducing-arcgis-maps-sdk-for-unity/

[19] GRILL®. (n.d.). *Air Force Research Lab's GRILL®*. GRILL®. https://www.af-grill.com/

[20] GRILL®. (n.d.). *Gaming Research Integration for Learning Laboratory (GRILL)*. Gaming Research Integration for Learning Laboratory (GRILL). https://github.com/AF-GRILL/

[21] Jungherr, A., & Schlarb, D. B. (2022, June 24). The Extended Reach of Game Engine Companies: How Companies Like Epic Games and Unity Technologies Provide Platforms for Extended Reality Applications and the Metaverse. https://doi.org/10.1177/20563051221107641

[22] JSBSim-Team. (n.d.). *An open source flight dynamics & control software library*. JSBSim-Team/jsbsim. https://github.com/JSBSim-Team/jsbsim

[23] McGray, M. (2020, June 29). *Using Game Engines For Cinematography and Film Production - Postpace Blog*. Using Game Engines For Cinematography and Film Production. https://postpace.io/blog/using-game-engines-for-cinematography-and-film-production/

[24] Morimoto, T., Kobayashi, T., Hirata, H., Otani, K., Sugimoto, M., Tsukamoto, M., Yoshihara, T., Ueno, M., & Mawatari, M. (2022, January 17). *XR (extended reality: Virtual reality, augmented reality, mixed reality) technology in Spine Medicine: Status Quo and quo vadis*. MDPI. https://www.mdpi.com/2077-0383/11/2/470

[25] Mower, N. (2023, July 8). *A Practical Guide to Unreal Engine's Coordinate System*. techarthub. https://www.techarthub.com/a-practical-guide-to-unreal-engines-coordinate-system/

[26] MozDevNet. (2023, June 28). *Game distribution - game development | MDN*. MDN. https://developer.mozilla.org/en-US/docs/Games/Publishing_games/Game_distribution

[27] Rantanen, T., Julin, A., Virtanen, J.-P., Hyyppä, H., & Vaaja, M. T. (2023a, July 28). *Open Geospatial Data Integration in Game Engine for Urban Digital Twin Applications*. MDPI. https://www.mdpi.com/2220-9964/12/8/310

[28] Unity Technologies. (n.d.-c). *Build Multiplayer Games with Unity Netcode*. Networking & Netcode Software Solution.

https://unity.com/products/netcode?utm_source=youtube&utm_medium=social&utm_campaign=engine_global_generalpromo_2022-06-22_insider-campaign-summer-22-tarodev

[29] Unity Technologies. (n.d.). *Designing UI for Multiple Resolutions*. Designing UI for Multiple Resolutions | Unity UI | 1.0.0. https://docs.unity3d.com/Packages/com.unity.ugui@1.0/manual/HOWTO-UIMultiResolution.html

[30] Unity Technologies. (n.d.-b). *Multiplatform tips and best practices*. Maximize Multiplatform Game Development | Unity. https://unity.com/solutions/multiplatform#multiplatform-unity-2022-lts

[31] Unity Technologies. (n.d.-d). *Plans and pricing*. Powerful 2D, 3D, VR, and AR software for cross-platform development of games and mobile apps. https://unity.com/pricing

[32] Unity Technologies. (n.d.). *High Precision Framework.* Unity-Technologies. https://github.com/Unity-Technologies/com.unity.gis.high-precision-framework

[33] Unity Technologies. (n.d.-e). *Physics*. Unity. https://docs.unity3d.com/Manual/PhysicsSection.html

[34] Unity Technologies. (n.d.-e). *Supported Model File Formats*. Unity - Manual: Supported Model File Formats. https://docs.unity3d.com/2020.1/Documentation/Manual/3D-formats.html

[35] Unity Technologies. (n.d.-e). *Unity Forums*. Unity Forum. (n.d.). https://forum.unity.com/

[36] Unity Technologies. (n.d.-e). *XR*. Unity – Manual: XR. https://docs.unity3d.com/Manual/XR.html

[37] Vidakis, N., Barianos, A. K., Trampas, A. M., Papadakis, S., Kalogiannakis, M., & Vassilakis, K. (n.d.). Generating Education in-Game Data: The Case of an Ancient Theatre Serious Game. https://pdfs.semanticscholar.org/7f41/26cab6f8228e78f6b5d4ce361ee42e707f83.pdf

[38] *What is a Gaming or Game Engine?*. What is a Gaming Engine? - Arm. (n.d.). https://www.arm.com/glossary/gaming-engines#:~:text=A%20gaming%20engine%20may%20include,effects%2C%20an%20animation%20engine%2C%20and

[39] White, S., & Wojciakowski, M. (2022, October 20). *Graphics Pipeline - UWP applications*. UWP applications | Microsoft Learn. https://learn.microsoft.com/en-us/windows/uwp/graphics-concepts/graphics-pipeline