Analyzing Strengths and Weaknesses of Modern Game Engines

Tauheed Khan Mohd*, Fernando Bravo-Garcia, Landen Love, Mansi Gujadhur, and Jason Nyadu

***Abstract*—The growing relevance of gaming engines in the gaming industry is making it gradually harder for a prospective developer to choose a game engine. In this analytical paper, the authors analyzed the following game engines: Cocos2D-X, CryEngine, GDevelop, Godot, Panda3D, Unity, and Unreal Engine. In every section, the authors elaborated on the advantages and disadvantages of each engine. Some users claim that older gaming engines like CryEngine are irreplaceable and that their engine structure should be exemplary for other engines. The new modern wave of game engines including Unity and Unreal Engine end up dominating the market with compatibility at every gaming platform and the ability to create both 2D and 3D games. Along with that having a strong community and developer support results in a very content user base.**

***Index Terms*—Gaming technologies, video game engine, unreal engine, unity, Godot**

I. Introduction

Andy Serkis, English actor, producer, and director once said, "Every age has its storytelling form, and video gaming is a huge part of our culture. You can ignore or embrace video games and imbue them with the best artistic quality. People are enthralled with video games" [1]. Something that we cannot deny is that video games and video game culture have been permanently ingrained in our society. As we near the halfway point of a century of video game history, society now has access to millions of video games and hundreds of tools to create and distribute them. Fifty years of video game history have given us classic titles like Tetris, Super Mario Bros, or Pac-Man. Although we love the classics, video games and video game development look vastly different than when they did in the past. Computing power has grown exponentially since the beginning of video games; thus, we have gained the ability to create more immersive and complex games. Behind every game is a video game engine, and these different engines are responsible for the different types of games that we play. Someone familiar with video games could easily differentiate that "playing a game such as World of Warcraft on a personal computer is very different to playing Call of Duty on a games console, which is vastly different to playing Candy Crush Saga on a mobile phone" [2]. The purpose of this technical report is to investigate why video games are so different by analyzing the game engines that power them. In this report, we will analyze eight different game engines including Cocos2D- X, CryEngine, GDevelop, Godot, Panda3D, Unity, and Unreal Engine. In the analysis of every game engine, we highlighted features of each game engine but also bring attention to possible flaws that should deter a user from using the engine.

My team and I did a little research on game engines and which one we should include in our research paper. The seven game engines that we chose are the most up-to-date and latest game engines. We also included the best game engines for example, Unity from the forthcoming comparison table in the conclusion section, a fact which proves that Unity is popular for its features. We included some game engines that have been existing for many years and are still known for their simplicity to use, for example, Godot.

II. CryEngine

Most known for the award-winning title 'Far Cry,' CryEngine was considered the "next generation" when it first entered the gaming scene. Developed by CryTek Studios, 'Far Cry' was ahead of its time by using complex algorithms to give the user access to graphics never seen in gaming [3]. Not content with the early success that 'Far Cry' had, the studio decided to develop a new game engine and a game, hand-in-hand; the engine was CryEngine 2 and the game was Crysis. Once again, an immediate success, Crysis added to the pattern of photorealistic shooters that had gaming audiences in a trance [3]. These two engines set the precedent for CryEngine 3; they would carry over their pattern of giving the user a high-definition picture that mirrors reality.

The most up-to-date version of CryEngine is CryEngine 3. Currently, CryEngine 3 supports development for games on PlayStation, Xbox, Windows, Linux, Android, Oculus Rift, OSVR, PSVR, and HTC Vive. The games in which the engine excels are first-person shooters [4]. Users of CryEngine have access to both the engine and the editor code, and because of this, many games are created using a heavily modified version of CryEngine [4]. The languages supported by CryEngine are C++, VisualScript, and Lua [5]. Users should appreciate this level of accessibility provided by CryTek Studios. Cry-Engine is a freeware but they do have a royalty system. First-time users have their first 5000 dollars of income royalty-free, but afterwards they will pay a 5 percent royalty in their game sales [4].

As mentioned before, this engine is most effective for creating First Person Shooter (FPS) games. So, users looking to create a realistic FPS game should start with CryEngine. Unlike other engines CryEngine gives the user a very strong template to start, you don't have to start from scratch to create a game. Users are given access to "Networking, game modes, characters, animation setups, weapons, vehicles, UI – it

Manuscript received July 24, 2022; revised October 14, 2022; accepted November 11, 2022.

Tauheed Khan Mohd is with Augustana College, RockIsland, Illinois, 61201, USA.

Fernando Bravo-Garcia, Landen Love, Mansi Gujadhur, and Jason Nyadu are with the Math and Computer Department, Augustana College, Rock Island, Illinois, 61201, USA.

*Correspondence: tauheedkhanmohd@augustana.edu

basically [comes] with everything required in a game" [4]. CryEngine's advanced rendering tools give its users the ability to create "high-fidelity visuals." One of the major advantages of CryEngine over other engines is the ability to create very complex landscapes [4]. Lastly, CryEngine is a good choice for beginner game developers, it's easy to learn and very powerful. Along with that, new users can get support from CryTek, the community is small, and it makes it very plausible to get in contact with CryTek employees if necessary.

Having a small community also comes with disadvantages. It takes longer for CryEngine to find bugs in their engine due to the smaller user-base and new users may struggle to find support from other users. Considered third to Unreal and Unity, it is very hard to find experienced users in CryEngine [4]. A major disadvantage of CryEngine is that it is very poor when it comes to creating RPG and fast-paced titles, the way the engine is created doesn't lend itself to those games [4]. It'd be better to use a different engine if a user hopes to make those kinds of games. Lastly, even though users are given a strong template they still must create most network systems and development tools [4]. CryEngine is a strong engine responsible for amazing games but depending on your needs it may or may not be the best choice for a user.

III. Cocos2D-X

Cocos2D-X is a mature open-source cross-platform game development framework that supports 2D and 3D game creation. The engine provides rich functions such as graphics rendering, Graphical User Interface (GUI), audio, network, physics, user input, etc... It is widely used in game development and interactive application construction. Its core is written in C++ and supports development in C++, Lua, or JavaScript. Cocos2D-X deploys to iOS, Android, HTML5, Windows, and Mac systems with features focused on native mobile platforms. It is a branch of Cocos2D.

At the heart of Cocos2d-x, you find the Sprite class and what that class does, in simple terms, is keep a reference to two very important rectangles. One is the image (or texture) rectangle, also called the source rectangle, and the other is the destination rectangle. If you want an image to appear in the center of the screen, you will use Sprite. You will pass it the information of what and where that image source is and where on the screen you want it to appear. There is not much that needs to be done to the first rectangle, the source one; but there is a lot that can be changed in the destination rectangle, including its position on the screen, its size, opacity, rotation, and so on Cocos2D-D will then take care of all the OpenGL drawing necessary to display your image where you want it and how you want it, and it will do so inside a render loop. Your code will most likely tap into that same loop to update its own logic. Cocos2D-X has the capability to create any 2D game that a programmer could imagine. Most 2D games can be built with Cocos2D-X with a few sprites and a loop [6]. Also important in Cocos2d-x is the notion of containers (or nodes). These are all the objects that can have sprites inside them (or other nodes.) This is extremely useful at times because by changing aspects of the container, you automatically change aspects of its children. Move the container and all its aspects will move with it.

The containers are Scene, Layer, and Sprite. They all inherit from a base container class called a node. Each container will have its peculiarities, but basically, you will arrange them as follows: Scene: This will contain one or more Node, usually Layer types. It is common to break applications into multiple scenes; for instance, one for the main menu, one for settings, and one for the actual game. Technically, each scene will behave as a separate entity in your application, almost as sub-applications themselves, and you can run a series of transition effects when changing between scenes. Layer: This will most likely contain Sprite. There are several specialized Layer objects aimed at saving you, the developer, some time in creating things such as menus for instance (Menu), or a colored background (Layer Color). "You can have more than one Layer per scene, but good planning makes this usually unnecessary. Sprite: This will contain your images and be added as children to Layer derived containers. To my mind, this is the most important class in all Cocos2D- X, so much so, that after your application initializes, when both a Scene and a Layer object are created, you could build your entire game only with sprites and never use another container class in Cocos2D-X node: This superclass to all containers blurs the line between itself and Layer, and even Sprite at times. It has its own set of specialized subclasses (besides the ones mentioned earlier), such as Motion Streak, Parallax Node, and SpriteBatchNode, to name a few. It can, with a few adjustments, behave just as Layer. But most of the time you will use it to create your own specialized nodes or as a general reference in polymorphism" [7].

Game engines usually support multiple platforms thus making it easy to develop your game and then deploy it to multiple platforms without much overhead at all. Since Cocos2D-X is a game engine, it provides a simplified API for developing cross-platform mobile and desktop games. By encapsulating the power inside an easy-to-use API, you can focus on developing your games and worry less about the implementation of the technical underpinnings. Cocos2D-X will take care of as much or as little of the heavy lifting as you want.

IV. GDevelop

GDevelop was created by Florian Rival who is a software engineer at Google. It is a 2D cross-platform, free and open-source game engine, which mainly focuses on creating PC and Mobile games, as well as HTML5 games playable in the browser. GDevelop is mainly aimed at non-programmers and game developers of all skill sets, employing event-based visual programming like such engines as Construct, Stencyl, and Tynker. It is widely used in games education, primary schools, and university courses because it is simple to use. It has also been used by educators and researchers to create learning and serious games [8]. It is available for the creation of all types of 2D gaming, which can be exported for web platforms (HTML5), and for native platforms (Windows, GNU/Linux, and Mac OS). This software does not require its users to have knowledge of a specific programming language. In GDevelop all the game logic is constructed through an intuitive and powerful graphical interface that is based on the control of events [8]. It uses an event-sheet style of making games, dragging and dropping instead of programming line after line. They're a great entry point for people interested in making games and other software [9]; at the time of writing this paper, GDevelop has just recently published version 5.0.0-beta93 which includes a much-needed pass over the user interface. The interface size has shrunk and no longer takes most of the users' attention, with extra padding around the main interface. Not only that, the User Interface (UI) is more persistent with their changes being saved and restored when opening a scene, extensions, or the debugger. There is no need for coding using this system which is clear and powerful: events are composed of conditions and actions. Actions are launched when conditions are fulfilled.

This is a very user-friendly way of making games and is still efficient for advanced usage, contrary to most other "block"/ "dragonroot" systems. It allows prebuilt behaviors to be added to objects. It is a very efficient way to add a physics engine or make a platformer game. GDevelop includes many behaviors, from the most advanced (Physics, platformer, top-down movement) to simple behaviors (like the behavior to destroy objects when outside the screen or the one to drag objects with a mouse or touch). The programmer still has Fulton rolls over his game as behaviors can be modified using the events [10]. GDevelop doesn't compile the games - it just adds wrappers so each OS can run the HTML5 game it creates. That means it runs much slower than other engines that do compile games. While the engine is free and open source as stated on the main website, it does not mention that some optional features and services are activated through a paid subscription (two tiers: 2 C and 7 C). Those features are no nag screen shown when debugging, additional metrics available on the games dashboard, access to more than two cloud exports per day (unlimited local export can be done without a subscription, provided the right packaging tools are installed and configured), easy removal of GDevelop splash screen (can be done manually without a subscription). Since it has a fully GUI editor, the objects they are allowed to add in their game are pretty generalized (Physics Object, Tiled Sprite, Platformer Object, etc.). This limits the freedom of a game developer while making a game, as the object must follow the preset behaviors imposed on it. This platform is easy to use as it comes with the necessary documentation and tutorials that will ensure that you learn how to use it rather quickly. The software owners of the platform collaborate with their Q&A portal to ensure that there are adequate tutorials and documents concerning the use of the software, which everybody can access. GDevelop's features make it possible for users to embed what they just need to use making it one of the most user-friendly software.

V. GODOT

Godot is a cross-platform, free, and open-source game engine released under the MIT license. It is designed to create both 2D and 3D games targeting PC, mobile, and web platforms. It is a program that allows the user to create games or even applications that can be released on desktop and mobile platforms. They can also make console games using Godot; although, they need strong programming or to hire a developer to port the game. Godot has its own built-in scripting language, GDScript a high-level, dynamically typed programming language that is syntactically like Python [11]. The Godot Game Engine is an open-source tool for developing 2D and 3D games. Currently, Godot is the most popular open-source game engine available and can export to MacOS, Linux, Windows, Android, iOS, HTML, and web assembly. Godot supports multiple programming languages including Python, C#, C++ and GDScript (a Python-like scripting language). Godot provides an interactive editor for the design and creation of video game environments, characters, animations, and menus, which enables fast iteration for prototyping new ideas. Godot has all the features of a modern game engine such as physics simulation, custom animations, plugins, and physically based rendering.

Godot is a strong engine for 2D games, and it is completely free. The only cost for the programmer is the time to develop a game or app. Godot is in demand because there are no royalty fees to pay in order to use it, compared to other engines. Other users prefer Godot for its download size, interface, and GD Script. It is very closely modeled on the Python language. If the user is already familiar with Python, they will find GDScript very familiar. If they are comfortable with another dynamic language, such as JavaScript, they should find it relatively easy to learn. Python is often considered a good beginner programming language, and GDScript shares that user-friendliness [12]. GDScript is a dynamically typed language, meaning they do not need to declare a variable's type when creating it and it uses whitespace (indentation) to denote code blocks. Overall, the result of using GDScript for their game's logic is that they write less code, which means faster development and fewer mistakes to fix.

However, Godot is known for being inferior in producing 3D games. The common failure when using Godot for 3D comes from scaling, so the user might investigate scaling while creating their game. It's also open source and development is only funded through Patreon or by generous contributors, which means slower development than other top titles like Unity [12]. The developers at Godot have been focused on completing their documentation; unfortunately, the documentation is somewhat directed to more experienced developers. In comparison, Unity Engine has more solutions outside of the documentation and they also have their special guides geared towards beginners. Godot Engine has a simpler GUI, which will be easier at first, but as the tasks became more difficult this became more of a disadvantage and some solutions were not found easily among the documentation. In comparison, the Unity Engine has more menus and can seem a lot to a beginner, but it becomes more helpful the harder the tasks become. Even though most solutions cannot be found among the documentation, there was enough information from other sources to solve all problems with an easy search. Godot has a much smaller community and less support than other game engines, but their easy-to-use interface and ability to create astounding 2D games will keep them in the conversation for top game engines.

VI. PANDA3D

Panda3d is a gaming engine that was developed by Disney's VR studio in Python, C++, and C#. The engine includes graphics, audio, I/O, collision detection, and other abilities relevant to the creation of 3D games [13]. The Panda3D game engine was initially a closed-source project of Disney Interactive but was later opened to the community, allowing anyone to use the engine or contribute code. Development of Panda3D is now driven and coordinated in a joint effort by Disney Interactive and the Entertainment Technology Center of Carnegie Mellon University. Together, they are adding new features, fixing bugs, and preparing new releases of the engine. Panda benefits from having an open-source local area for documentation, testing, and advancement of new elements. Its exhibition qualities and free accessibility as an open-source programming project position Panda as a potential graphics engine standard for simulation and game production. It currently uses a revised Berkeley Software Distribution license [14]. This engine was programmed to be flexible enough to support everything from real-time graphics applications to the development of high-end virtual reality theme park attractions or video games. The acronym itself lists Panda's primary features as platform-agnostic, networked, and display architecture [15]. Panda being platform agnostic means that it can run on any platform such as Windows, iOS, and Linux. Panda3D makes extensive use of abstraction layers to facilitate portability. The system represents services such as rendering (the process of generating an image from a model), audio, peripheral devices, graphical user interfaces, and scripting languages in an abstract form so that it can use different platforms and external software packages interchangeably. Panda currently runs on a wide array of graphics cards for all versions of Windows PCs, Linux PCs, and Silicon Graphics workstations running Irix. The engine offers rendering support for DirectX, OpenGL, and Pixar's Renderman [15]. Panda was designed to support a range of networked applications including multiuser design scenarios, multiplayer games, and multiple synchronized displays powered by multiple PCs. In addition, developers can use Panda to deliver downloadable experiences over the Internet. Designers thoroughly leverage Panda's networked features in the development of massively multiplayer online games. In Toontown Online, for example, Panda's small memory footprint makes it possible to support the online delivery of large content. In addition, the in-game requirements for a massively multiplayer game include built-in concepts of distributed objects, visibility, and efficient network communications [15]. Panda can display a scope of presentations from low-end PCs to very good quality multi-screen vivid theater conditions. Arranging applications that require various windows, regardless of whether they require synchronization across equipment stages, is moderately simple. The ability to specify the state of nodes internal to the scene graph means that an object's state depends on an ordered accumulation of all the states in that object's ancestor nodes, up to the scene graph's root. To avoid having to aggregate this state in every rendering frame, Panda includes an efficient caching mechanism. The system initially determines an object's properties by doing a full traversal of the scene graph, caching the results, and storing the collective results at each node. After doing this, the engine only recomputes a node's properties when an intervening state changes. Panda supports several modes of dynamically grading content to take advantage of different hardware capabilities. The system implements several visibility algorithms to eliminate unnecessary rendering, including cell portal visibility.

VII. Unity

Unity was created by a group of three developers: Nicholas Francis, Joachim Ante, and David Helgason. The three were responsible for "creating one of the most useful pieces of software in video game history" [16]. Unity wanted to support independent developers who could not afford licensing fees and become the standard for 3D modeling in video games [16]. A trend in Unity's history is its willingness to adapt to different technologies. Unity was first developed for Mac but as popularity rose there is now a version for Mac OS, Windows, and iPhone with each having strong developer support. Currently, Unity is considered a top-tier cross-platform game engine used to "develop 2D and 3D video games, simulations for computers, virtual reality, consoles, and mobile devices platform" [17]. As aforementioned Unity was first developed for Mac OS, and the developers of Unity, like Mac OS, tried to create an interface that was simple and easy to use. Users of unity cherish Unity for that reason. Unity can be broken up into five main sub-windows: 1) Project Browser, Inspector, Game View, Scene View, and Hierarchy [16]. The project browser contains all assets that the user has imported into the engine [16]; 2) The inspector lets the user fine-tune the objects that the user placed [16]; 3) The game view renders exactly what your game would look like at that moment with whatever objects, backgrounds, or music was added [16]; 4) Scene View allows users to place objects anywhere in your scene, with supports like guiding lines for the user [16]; 5) Lastly, the hierarchy shows all the objects the user placed in one scene. A huge benefit of using Unity is that it was designed so the file system feels like Mac's 'Finder' or Windows' file explorer. Users flock to Unity because of its ease of use for beginners [16]. An advantage that Unity has over every engine is its asset pipeline and ease of use. When a user adds an asset, he simply drags and drops it into the project file. Unity will open the file you select and if it is not in the correct file type will open the necessary program to convert the file into FBX which Unity can use. Editing the asset automatically updates all other assets [16]. Lastly, Unity users have access to three main scripting languages Unity Script, Boo, and C#. Unity Script is a JavaScript-like scripting language that is great for beginners. If users use Unity Script or they can expect to receive online support for queries about Unity [16]. Some game developers believe that Unity is the standard that all other game engines should be reaching for. Unity is an iconic game engine but is not perfect. A complaint that Unity has received is that no templates exist; "any game or project needs to be implemented from scratch to the detailed functions with GUI" [17]. From a graphics, perspective Unity has performed slower than some of its competitors. Lastly, from a programming perspective, users complain about a poor coordinate projection system and little support to connect with databases [17]. Unity should be the first option for users who are first entering the game development world. The founders of Unity created the application with the goal of a smooth application that can be accessed by independent developers, and they have achieved that goal. Games produced by Unity include but are not limited to Cuphead, Hearthstone, or Escape from Tarkov [16]. The flaws of the application are more pertinent to experienced users who may be suited for other game engines.

VIII. Unreal Engine

Created by Epic Games in 1998 for the first-person arena shooter Unreal, the Unreal Engine has since grown into a multi-purpose software development suite for hyperrealistic 3D rendering. While like Unity in terms of video game development, Unreal Engine has given more support to smaller creators in recent years. Epic Games announced in 2020 that game studios that have earned less than $1 million in lifetime revenue do not have to pay 5% royalties for game sales [18, 19]. Unity, by comparison, requires developers to purchase a $1,800 yearly "pro license" if a company is making more than $200,000 annually [18]. The source code for the Unreal Engine is available on a private Github repository that can be accessed once a user syncs their GitHub account with their Epic Games account. The Unreal Engine is primarily a three-dimensional game engine but is technically able to create two-dimensional games as mentioned in Table I.

One of the main drawbacks of the Unreal Engine is that it is primarily programmed in C++. This can be intimidating for newer developers. This causes some developers to prefer Unity as it supports C# and Javascript as primary scripting languages [20]. Although Unreal Engine uses C++ for some of its lower-level functionality, it also supports a visual scripting language called Blueprint. Blueprint is a simple node-based scripting language that interfaces easily with existing C++ code. It is possible to program entire gameplay systems using only Blueprint with minimal performance loss. The Unreal Engine also has multiplatform support allowing for easy exporting of projects to both game consoles and various operating systems such as Windows, Linux, and MacOS. Support for both virtual and augmented reality applications is also available.

Microsoft's augmented reality headset called the Hololens is another platform that the Unreal Engine has recently added support for. With on-premises work and support roles transitioning to remote positions, augmented reality headsets are starting to fulfill a vital role in problem management and support. One of these problem areas is population protection and crisis management [21]. A recent article posted by DAAAM international demonstrates the use of an app created with the Unreal Engine running on the Hololens [21]. This app uses modified map data in the 3D software Blender that has then been exported to the Unreal Engine [21]. A top-down view of an area is visible on the Hololens. The user can then interact with the map in real-time [21]. The user can then highlight certain areas of interest such as hospitals, police stations, the fire brigade, and permanent pressure shelters for the civilian population [21]. The user can make these selections by pressing virtual buttons with their real hands. The Hololens uses multiple cameras to track both the users' position and hand movements. The use of Blueprint in the Unreal Engine allows for this type of rapid prototyping and is easy to export to the Hololens with Unreal Engine's built-in compilation tools.

Besides video game development, Unreal Engine is starting to find uses in the film industry. Disney's critically acclaimed television series "The Mandalorian" heavily relied on the Unreal Engine as part of their filming pipeline. In a minidocumentary published by Industrial Lights & Magic on YouTube, the Unreal Engine is shown being used to display vast landscapes behind actors on a cylindrical stage [22]. This use of the Unreal Engine over traditional Visual effects (VFX) has three major advantages. 1) Background props can be repositioned or replaced in real-time. Because the Unreal Engine is essentially a video game engine, the director can have technicians move any background prop, large or small, to better fix the look and feel that the director desires for a particular shot; 2) The Liquid Crystal Display (LCD) screens project light onto the actors and props in a realistic manner. In The Mandalorian, the main character wears very reflective armor. If the actor was on a traditional greenscreen stage, their armor would reflect the green background; 3) Decreased cost. With a virtual set, almost any scene can be shot within the cylindrical stage. You can transition from a virtual indoor set with large vehicles and crates to an outdoor set with mountain ranges with the click of a button. The only additional changes necessary are the physical props that accompany the actors on set. The director of The Mandalorian, Jon Favreau, said that one major benefit of the digital background was that it gave actors better direction in what the area around them is supposed to look like. "For the actors, it was great because you could walk on the set, and even if it's just for interactive light, you are walking into an environment where you see what's around you. Even though [the Light Emitting Diode (LED)s walls] might not hold up to the scrutiny if you're starting right at it from close, you're still getting a peripheral vision. You know where the horizon is, you feel the light on you. You're also not setting up a lot of lights. You're getting a lot of your interactive light off those LED walls. To me, this is a huge breakthrough" [23].

TABLE I: Comparison Table of Gaming Engines

Engine Name	Platforms Supported	Language Support	Pricing/ Royalties	Types of Game
CryEngine	Win, XBOX, PS, Wii, and others	C++, VS, Lua	Free/ 5%	3D, Realism
Cocos2D-X	Android, iOS, Mac, Win	C++, JS, Lua	Free/ No	2D, Mobile
GDevelop	Android, iOS, Win, Mac, Lin	C++, JS	Free/ No	2D, Mobile, PC
Godot	Android, iOS, Win, Mac, Lin	GDScript, VS	Free/ No	2D/3D, Mobile, PC
Panda3D	Win, Mac, Lin	C++, Python	Free/ No	3D/ PC
Unity	Win, Mac, XBOX, PS, Lin	UnityScript, Boo, C#	Starting Free/ None	All
Unreal Engine	PS, XBOX, Nintendo, Win, Mac, Lin, Android, iOS	C++	Free/ 5%	All

Abbreviations for Table: [Windows (Win), PlayStation (PS), Linux (Lin), JavaScript (JS), VisualScript (VS)].

IX. Conclusion

As stated in the introduction, the purpose of our research was to highlight features and point out flaws of various game engines as shown in Table II. It is nearly impossible to choose the "best" game engine because every engine was created for a different purpose and each user would have to weigh the advantages and disadvantages themselves. A prospective game programmer would have to decide the type of game they wanted to create, the programming language they want to write in, and the platform they want to create a game for. Still, we will select some game engines that we believe provide the user with the best experience. We particularly selected these eight engines as it is the most prominent and still in use nowadays. There are other game engines which we did not even hear the name of such as Game Make, J Monkey, Ogre3D and many more. The game engines we listed above are not so in use nowadays as better and more advanced game engines were launched with more features and it was a perfect fit for us to go deeper in our research.

When analyzing every game engine, two game engines always seemed to finish on top. Both the Unreal Engine and Unity are considered our two best choices for "best game engine." Unreal and Unity can export their games to any gaming platform that is currently in the market. Users have an easy experience because their UI and UX feel natural and lead to an easy game-making progress. Documentation for both programs is vast and they both receive strong community support. Lastly, both technologies are top-class, and still, the game licenses are completely free, and any user can download them. There are paid subscriptions, but no payment is mandatory until royalties activate after a certain amount of revenue. Some other honorable mentions go to Godot and CryEngine who have strong technology but are lacking community support that could take the technology to the next level.

If a user is looking to become a game developer, they should not stray too far from Unreal and Unity. The level of free access that users have used these two technologies is admirable. A user could design for free and still own complete rights to every piece of the game. Other options are possible, but those two engines are the best.

TABLE II: STRENGTHS AND WEAKNESSES OF GAMING ENGINES

Engine Name	Strengths	Weaknesses
CryEngine	Easy to learn and offering the full source code, CryEngine is equipped with quality-oriented audio transaction layers.	Does not have an online support community for developers.
Cocos2D-X	A best-in-class platform that involves many of mobile platforms, free and open-source under the MIT license, includes different kinds of extensions and tools, better performance of the graphic output Quality-oriented online support community.	No dedicated support system for fixing any kind of bugs that arise and other different issues, It does not include a better coding structure.
Godot	Works for 2D and 3D games, completely free and open-source – even commercially, Passionate community, Unique architecture for game development.	Experienced game developers may not like GDScript, as not many resources are available.
GDevelop	Free and open-source tool, create games and then export your game to Windows, Linux, Mac, Facebook Instant Games, Android, and even iOS.	Cannot do complex games, incomplete JavaScript integration.
Panda3D	Panda3D has a documentation includes Python reference and C++ reference, it is open source and free to use even commercially, and easy to start	The community is not as large as some of the popular game engine communities out there.
Unity	Free for beginners, Great for 2D games, Strong game support, VR and AR SDK availability, and Asset Store with several free assets.	Costly licenses for professionals, Higher-end tech demos require better computers, and Many UI changes.
Unreal Engine	A top choice for VR, Visual blueprinting for non-programmers, and a sizeable marketplace with free assets.	Not the best for simple or solo projects, High- end graphics require more powerful computers, better for 3D than 2D games.

CONFLICT OF INTEREST

The work was carried out without any conflict of interest.

AUTHOR CONTRIBUTIONS

Tauheed Khan Mohd has proposed the idea of performing research on the Gaming Engine and collaborated with the team. Landen Love surveyed the three Gaming Engines, which are Godot, Unity, and Unreal Engine. Jason Nyadu has analyzed Cocos2D-X, CryEngine, and Panda3D. Mansi Gujadhur has analyzed the pros and cons of all the Gaming Engines. Fernando Bravo-Garcia researched the platforms-supported pricing, and language support by these Gaming Engines.

REFERENCES

[1] L. Gilbert, "Assassin's creed reminds us that history is human experience: Students' senses of empathy while playing a narrative video game," *Theory & Research in Social Education*, vol. 47, no. 1, pp. 108-137, 2019.

[2] M. Daniel and G. Crawford, *Video Games as Culture: Considering the Role and Importance of Video Games in Contemporary Society*, 2018, Routledge.

[3] M. Mittring, "Findingnextgen: Cryengine2," in *ACMSIGGRAPH2007 Courses*, 2007, pp. 97–121.

[4] M. Dealessandri. (2021). What is the best game engine: Is CryEngine right for you. Saatavilla Osoitteesta: [Online]. Available: https://www.gamesindustry.biz/articles/2020-01-16-what-is-the-bestgame-engine-is-cryengine-the-right-game-engine-for-you

[5] P. Mishra and U. Shrawankar, "Comparison between famous game engines and eminent games," *International Journal of Interactive Mul-timedia & Artificial Intelligence*, vol. 4, no. 1,2016.

[6] R. Engelbert, *Cocos2d-x by Example: Beginner's Guide*, Packt Publishing Ltd., 2015.

[7] S. Shekar, *Learning Cocos2d-x Game Development*, Packt Publishing Ltd., 2014.

[8] J. D. C. Correa, *Digitopolis II: Creation of Video Games GDevelop*, 2015.

[9] A. Chwastek. (2021). Platform Game "Twierdza" Created with GDevelop. [Online]. Available: https://ruj.uj.edu.pl/xmlui/handle/item/275239

[10] J. G. R. Souza and R. O. Prates, "Games by end-users: Analyzing development environments," in *Proc. 2021 20th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, 2021, pp. 69–78.

[11] C. Bradfield, *Godot Engine Game Development Projects: Build Five Cross-Platform 2D and 3D Games with Godot 3.0*, Packt Publishing Ltd., 2018.

[12] R. Flomén and M. Gustafsson. (2020). Game developer experience: A cognitive task analysis with different game engines. [Online]. Available: https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1437636&dswid=4779

[13] C. Lang, *Panda3D 1.7 Game Developer's Cookbook*, Packt Publishing Ltd., 2011.

[14] D. B. Mathews, *Panda3D 1.6 Game Engine Beginner's Guide*, Packt Publishing Ltd., 2011.

[15] M. Goslin and M. Mine, "The panda3d graphics engine," *Computer*, vol. 37, no. 10, pp. 112–114, 2004.

[16] J. Haas, "A history of the unity game engine," Ph.D. dissertation, Worcester Polytechnic Institute, 2014.

[17] I. Buyuksalih, S. Bayburt, G. Buyuksalih, A. Baskaraca, H. Karim, and A. A. Rahman, "3D modeling and visualization based on the unity game engine–advantages and challenges," in *Proc. Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, vol. 4, 2017.

[18] T. Wilde. (May 2020). Unreal engine games no longer owe royalties on their first $1m in revenue. [Online]. Available: https://www.pcgamer.com/unreal-engine-games-no-longer-owe-royalties-on-their-first-dollar1m-in-revenue/

[19] A. Jungherr and B. S. Damien, "The extended reach of game engine companies: How companies like epic games and Unity technologies provide platforms for extended reality applications and the metaverse," *Social Media+ Society*, vol. 8, no. 2, 2022

[20] A. Šmíd, "Comparison of unity and unreal engine," Czech Technical University in Prague, pp. 41-61, 2017.

[21] M. Dzermansky, L. Snopek, K. Vichova, M. Ficek, and J. Rak, "Use of augmented reality technology in population protection and crisis management.," in *Proc. Annals of DAAAM*, vol. 10, no. 2, pp. 408-414, 2021.

[22] The Virtual Production of the Mandalorian, Season One. (Feb. 2020). [Online]. Available: https://www.youtube.com/watch?v=aBVSnny3Bk4

[23] S. Axon. (Feb. 2020). The Mandalorian was shot on a holodeck-esqueset with unreal engine, video shows. [Online]. Available: https://arstechnica.com/gaming/2020/02/the-mandalorian-was-shot-on-a-holodeck-esque-set-with-unreal-engine-video-shows/

T**auheed Khan Mohd** received his B. Tech in computer engineering from Jamia Millia Islamia, New Delhi, India in 2006. He received his M.S degree from the University of Toledo, Ohio in 2015, and finished his Ph.D. in Human-Computer Interaction (HCI) during the Summer of 2019 at The University of Toledo. Previously, Tauheed worked for three years as a software engineer in HCL Technologies, India followed by four years at a French multinational company, SOPRA. He worked onsite for three months at AIRBUS in Toulouse, France, and managed their onboard application called Network Server System (NSS). Tauheed worked as a research assistant on an NSF Funded Project called INITIATE which enables High School Students to get attracted towards STEM subjects. His areas of research are human-computer interaction, multimodal input, autonomous vehicles, and micro-controller devices including Arduino and raspberry Pi.