SM+S
social media + society

United We Stand: Platforms, Tools and Innovation With the Unity Game Engine

Social Media + Society
October-December 2019: 1–10

DOI: 10.1177/2056305119880177
journals.sagepub.com/home/sms
SAGE

Maxwell Foxman

Abstract

The skirmish between game engines Unity and Unreal presents a new front in the platformization of cultural production. This article argues that such programs are "platform tools." They enable amateurs and professionals to not only build content for platforms but also "lock-in" industry ideologies in the ideation, production, implementation, and distribution of digital creative work, resulting in a homogeneity of developers, practices, and products. The Unity engine's history, features, and place in the game production pipeline makes it a paradigmatic "platform tool." Findings from 90 interviews with VR enthusiasts show that Unity set the boundaries or "rules" for developers' everyday activities and, despite enthusiasm about the medium's potential, compelled them to create content which conformed to popular gaming genres and standards.

Keywords

virtual reality, immersive media, diffusion of innovations, video games, game engine, lock-in

The evolution of digital platforms has been beset by "wars." These clashes between Windows/Mac, Google/Yahoo, Facebook/Myspace, and iOS/Android established battlegrounds for platform makers to retain control over "the conditions under which creative content is produced" (Gillespie, 2010, p. 358). While such feuds are well documented, another rivalry rages under the radar between Epic Games and Unity Technologies, whose game engines "are in a battle for the hearts and minds" (Takahashi, 2015) of programmers. Beyond games, the two are also vying for control over Virtual Reality (VR) and other immersive media. Ultimately, through their software's technical specifications, interoperability with other platforms and business models, the winner of the war will possess the de facto "tool" for making interactive content.

This article investigates how the Unity game engine sets standards for games and emerging technology (like VR). To do this, first I review the literature surrounding "plaformization," which reveals a persistent logic embedded in the economic, social, and technical makeup of platforms. One underlying component of this logic is "lock-in," where platforms are made dependent on and interoperable with each other to assure market viability. However, "lock-in" also constricts creative and ideological alternatives. These restraints are emblematic of what I call a "platform tool," or productivity software that simultaneously enables and locks-in how amateurs and professionals build digital content for platforms.

Unity offers a glimpse into the power platform tools have over creators. It both gives access to the necessary elements for virtual world building and compels developers to adopt norms derived from the game and tech industries in which it is enmeshed. To uncover the software's influence on creative practices, semi-structured interviews were conducted with VR enthusiasts. Their perspective elucidates the role of the engine in current and future modes of production. Unity opened an easy point of entry to play with VR, while it locked-in genres, user identity, professional criteria, and even the potential future of the medium around digital games' culture and industry.

The Problem With Platforms

There is "no consensus definition" (Martens, 2016) for "platforms." The term describes a variety of phenomena: from social networks (Helmond, 2015) to music distribution services, game consoles (Montfort & Bogost, 2009), and broadly any multisided market where an exchange occurs (Martens, 2016).[1] As a result, Nieborg and Poell (2018) characterize platforms by their "contingency" (p. 4276), in that they are at once dependent on each other and yet change constantly. A platform can therefore be understood as an app,

University of Oregon, USA

Corresponding Author:
Maxwell Foxman, School of Journalism and Communication, University of Oregon, Eugene, OR 97403, USA.
Email: mfoxman@uoregon.edu

software environment, or service contingent: (a) on other platforms both economically (via multisided marketplaces) and technically (through interoperable code), (b) in their mutability and modularity, and (c) on user content and feedback for functionality.

Platformization

Given platforms' broad definition and reach, scholars have also sought to distill their commonalities. Schwarz (2017) states that there is a complex "platform logic" that underlies all levels of use, from local code-based control to global networks (p. 94). "Platformization," as Nieborg and Poell (2018) name this logic, refers to the "penetration of *economic, governmental, and infrastructural extensions* of digital platforms" (p. 4276). This represents three fundamental shifts in the philosophy and politics of cultural production: first, in the role platforms play in shaping multisided markets; second, in the control exerted by platform providers on users; and third, in the infrastructure of the software itself (p. 4281). The result is that platforms impose limitations for the production, consumption, and even ideation of digital content: producers modify their products to fit platformized business models, and developers are incentivized to manufacture content that is "contingent, modularized, constantly altered, and optimized for platform monetization" (p. 4282).

Finally, platformization consolidates power and equity into a small number of global "platform behemoths," who can further entrench themselves because of the interconnectivity and interoperability inherent to the platform economy. Bechmann (2013) labels this phenomenon as "*intra*operability" to highlight the asymmetry in power relationships between end-users and providers who are "dominant in terms of market share, attitude, or acquiescence" (p. 75). Ultimately, retaining users, or keeping them "locked into" (p. 84) a platformized ecosystem, is an ideological prerogative of platform leaders.

Lock-In

Although rarely the explicit focus of platform studies, lock-in is a useful concept for further interpreting the implications of platformization. "Lock-in" is a negotiation between companies, products, and consumers in which the norms and uses surrounding a product are set and adopted. In doing so, different standards and products are obscured as switching to or introducing an alternative becomes more difficult and costly.

The concept is rooted in economics and studies of path dependence, in which "there are a finite number of perfect stable alternative states, one of which will arise based on the particular initial conditions" (Margolis & Liebowitz, 1998). Lock-in explains how history, social factors, and business strategies may cause the adoption of subpar consumer goods and services. Classic examples include the QWERTY keyboard's triumph over the more optimal Dvorak, and VHS tapes over Betamax.[2] In both cases, inferior versions dominated the market because they were "locked-in" by users before the competing standard. Besides removing competitive alternatives (Cantner & Vannuccini, 2017, p. 11), lock-in's value is that once established in consumers' lives it preserves profitable as well as failing aspects of a product or service.

In critiques of platforms, lock-in is referenced as a source of oligopolistic control of markets (Bodle, 2011), users, developers, and vendors (Nieborg & Poell, 2018; Plantin, Lagoze, Edwards, & Sandvig, 2016). However, these texts do not put the concept at the forefront of their studies.

This may be because its application is so widespread. In business studies alone, vendors, innovations, and technology can all be subject to lock-in. Shapiro and Varian (1998) also describe benefits of customer lock-in; since it is costly to switch from locked-in products, users can be enticed to hold onto them for years (such as Microsoft Office). Lock-in is at once a technical parameter, a business strategy, and even a simple explanation for the adoption of innovations.

I argue that lock-in also functions as an ideology to impede competing options, reflecting popular technology philosopher Lanier's (2011) concern that

> [l]ock-in removes ideas that do not fit into the winning digital representation scheme . . . [and] reduces or narrows the ideas it immortalizes, by cutting away the unfathomable penumbra of meaning that distinguishes a word in natural language from a command in a computer program. (p. 10)

It constrains the creative potentiality of cultural producers as they become inculcated into specific software. Lock-in, therefore, serves three purposes for cultural production: it sets consumer expectations of use, establishes a technical and social pipeline for common practices, and, perhaps most importantly, cements and institutionalizes corporations' imperatives and philosophies.

As such, lock-in represents an important avenue for assaying the impact of platformization and contingency. It establishes the technical and economic criteria by which companies, consumers, and producers interact with platforms, simultaneously constricting their creative possibilities while affording them normalized parameters by which they create. In other words, platforms can lock-in the tools with which users have to work, play, and produce.

Platform Tools

Lock-in takes on particular significance when considering what I call "platform tools." This software enables both amateurs and professionals to build content from and for platforms, thus serving an explicitly utilitarian purpose, as opposed to social platforms like Facebook, distribution platforms like Spotify, or even online marketplaces like Etsy. As tools, they are integral to the entire production process—from the ideation of a project through creation, production, and eventually

distribution. But as a *platform* tool, the software "locks-in" specific practices at each of these stages, setting rules and guidelines based on the platformized digital media ecosystem.

Like platforms, these tools are contingent on other platforms for economic viability, are constantly changing to meet the needs of existing and new platforms, and ultimately depend on consumers/producers to build content with them. But they are distinct in a number of key ways: they generate applications and creative work that may not be explicitly tied to or hosted on a platform (though they often are), and they are not billed as marketplaces.[3] Instead, a platform tool acts as an *intermediary* between industries and platforms to aid in the construction of a stand-alone application. However, a byproduct of this function is that it locks-in specific ideologies connected to those industries.

As a consequence, platform tools resemble "middlebroware" (Lesage, 2015) or a set of "commoditised media software and its related practices of design" (p. 90) used as a "glue" for "simultaneously enabling and constraining the production, circulation, and appreciation of cultural content" (p. 92). Lesage's example is Adobe Photoshop, software also for professionals and amateurs, which, he notes, constantly adds formats, features, and third-party plug-ins to sustain it as a vital force in shaping who and how digital photos are reproduced. Platform tools, as I conceive them, do not share the "symbolic order" of content that preoccupies Lesage. Furthermore, Photoshop does not require inter- or *intra*operable platforms to successfully create or distribute photographs.[4]

Nor does it make as extensive use of prefabricated software packages, kits, and application programming interfaces (APIs)[5] for content creation. These promote path dependence because it is less tedious for developers to work with the compatible software and premade code than to program from scratch. However, packages further define how applications interact with each other by providing a roadmap for what can be designed. These technical restrictions are central to software studies' critiques of platforms. Bogost (2008) states how they "both facilitate and limit discursive production, just as the rules of natural language bound poetry and the rules of optics bound photography" (p. 66). However, he adds that such restrictions can be empowering for users and even useful creatively. Similarly, Plantin et al. (2016) describe how platforms' "affordances simultaneously allow and constrain expression" (p. 298) through APIs, which act as "gateway[s], permitting other systems to interact" (p. 303) with dominant platforms that "*locks* [emphasis added] . . . groups into a landscape defined and controlled" by the leading company. This assertion underscores the notion that platform tools like the Unity Game Engine are a nexus by which to view the impact of platformization and lock-in both on platforms and producers.

Unity as a Platform Tool

Unity's features and market choices make it a quintessential "platform tool." As an "engine," it is a utility with the explicit purpose of building games and applications for broader distribution. Furthermore, it is connected to all aspects of game production and interfaces with existing platforms through its own multisided marketplace and compatible software packages.

Despite its ubiquity in game development—Unity is used by 45% of independent developers compared with Unreal, which has captured only 2% of the market according to popular reporting (Beschizza, 2018)—critical literature on the engine is surprisingly sparse. Schmalz (2015) briefly mentions it in his dissertation on the history of video game innovation in the 2000s. Panourgias, Nandhakumar, and Scarbrough (2014) write about game design through engines like Unity (without mentioning it specifically); they contend that development requires an interplay between creative intentions and practical restrictions coded into a program. Whitson (2018) goes one step further and argues for the vital role Unity plays in game studios. Scholars have not addressed issues of lock-in, platforms and cultural production even while path dependence is fixed as Unity becomes a popular tool for development.

Why Unity?

Unity, like other "game engines," is a "software framework" or a set of tools that facilitates rendering, physics, and input by developers, freeing them from having to construct virtual spaces from the ground up (Ward, 2008). The engine provides the building blocks for both three-dimensional (3D) and two-dimensional (2D) virtual worlds, which is no small task; consider the innumerable laws of nature that we take for granted—from gravity to the reactions of others when we touch them. These fundamental aspects of life must be taken into account to make virtual spaces playable.

Prior to 2009, most game engines were proprietary and closely guarded by companies. From its inception, Unity moved away from that model and offered tools directly to hobbyists. The engine's growth gained momentum from the "modding" movement, where enthusiasts modified games based on just enough code made available to them by publishers. Kücklich (2005) has defined the activities of such "modders" as "playbor," which capitalizes on the loyalty of these customers, adds life and value to existing products by having modders generate new content, and acts as a testing ground for ideas, all driven by users' passion and excess efforts.

Such playbor allowed Unity to pivot from a paid application to one that billed developers only after they earned a certain amount of revenue: US$100,000 during my period of study (Downie, 2016). In this way, their business model capitalized on the success of independent and mobile developers' enthusiasm and work (Haas, 2014, p. 10). Furthermore, as it grew in popularity, the engine expanded into new markets, including enterprise software and digital animation.

These strategies situated Unity as a go-to tool for making games and other interactive material. Whitson (2018) describes how developers treated the application as the "lowest common

denominator" (p. 2319) for production because it interfaced with different teams and software across studios. This underlines Unity's contingency as a "platform tool," along with the role the engine served in "apprenticing" and "socializing" (p. 2321) users; Unity imparted basic skills, and even "common purpose" (p. 2322). As attested in my interviews, it viscerally shaped the entire production experience.

This matches the image the company wants to convey: to give "developers around the world the tools to create rich, interactive, 2D, 3D, VR and AR experiences" ("Unity Public Relations Fact Page," n.d.). In fact, one of the reasons for immersive technologies' recent wave of popularity, along with the reduction in hardware cost, can be attributed to the software's interoperability and ease-of-use. Developers can make content across devices and distribute it through low-cost marketplaces like Apple's App Store and the Steam game library, which has contributed to a profusion of software for commercial VR. Meanwhile, platform behemoths including Google, Facebook-owned Oculus, and Microsoft have joined forces with Unity in the manufacture of their headsets to facilitate content creation.

Unity's popularity has led to a somewhat paradoxical view of the engine. On one hand, it has been credited with democratizing game production. A common trope is expressed in an *Ars Technica* post: Unity is "really letting anyone make a game" and "is partially responsible for the boom of independent and artistic games over the past half-decade" (Axon, 2016). On the other hand, the engine makes it just as possible for inexperienced developers to proliferate poor content. Democratization, the article continues, has an "unanticipated side-effect—it lowers quality standards for gamers. And worse . . . the new glut of Unity-bred games makes earning a profit harder in an already difficult market" (Axon, 2016). In sum, the engine, beyond playing a role in game development and 3D production, also is integral to who, how, and in what ways content is produced for a variety of platforms.

Unity's Platform Features

Unity contains a number of features commensurate with other platforms. For one, it is interoperable. To allow publishing on many platforms and consoles, Unity developed a "build and run" protocol, which only requires the tap of a button to load and start playing content on different devices. The code is continuously updated to accommodate builds for emerging formats.

At the same time, business expansion included opening its own multisided marketplace. Their "Asset Store" permits amateurs and professionals alike to upload homemade scenes, code, add-ons, and avatars for other users to download (for free or a fee) and populate virtual spaces.

Finally, the engine makes heavy use of *intra*operable code. Unity "packages" can be employed not only by developers to easily export their own material to other computers but also to import code written by hardware manufacturers. Packages usually include custom "scenes" or virtual world setups where the capabilities of hardware and software are modeled and assessed. If a new device comes out, such as the hand-tracking "Leap Motion" (n.d.) controller, scenes are made available to download through the engine so that developers can test the hardware and use them to build games and applications.

Thus, Unity positions itself as an indispensable go-between. It interacts with a universe of *intra*operable platforms and markets at every stage of production, from pulling together content and reconstructing it in virtual space, to publishing and distribution, making it an emblematic platform tool.

Unity in the Production Pipeline

To gain a better understanding of Unity's place within platform and video game ecology, the production pipeline of game development requires brief explanation. Derived from computer scientists Labschütz, Krösl, Aquino, Grashäftl, and Kohl (2011), game construction comprises four steps:[6] *ideation* or "concept phase" (p. 3), *production* or "3D content creation pipeline" (p. 3), *implementation* (p. 6), and *distribution* or "release" (p. 7).

Game development is iterative and often recursive. Users will go back and forth between phases as they progress. It also involves interfacing with other programs like "modeling" software (which is used to design 3D content) and software packages.[7] As Figure 1 and the following sections illustrate, each phase solidifies Unity's role as a key intermediary in game production while simultaneously locking-in existing platforms' standards into the development process.

Ideation. In the ideation phase, initial concepts, aesthetics, mechanics, and levels are brainstormed, then brought into the engine. Unity locks itself in as a platform tool even as a user envisions the game. In part, it is the ease with which content can advance from conception to distribution that makes it attractive to users. One reason Labschütz et al. (2011) chose the engine was because of its simple drag and drop features, which meant art could be imported with little or no coding (p. 2).

Production. Production involves creating the "assets" (characters, objects and other materials) that will be used in the virtual space of the game, as well as level design. Some features (such as lighting, animation, and "primitive" objects) are built into the engine as defaults, but Unity also offers integration with more sophisticated 3D graphics applications, such as Maya and Blender. For those proficient in either game development or modeling, this locks them into the compatible pieces of software. For novices, Unity provides another path to lock-in: its preloaded Asset Store markets content, plug-ins, packages, and other utilities that simplify production.

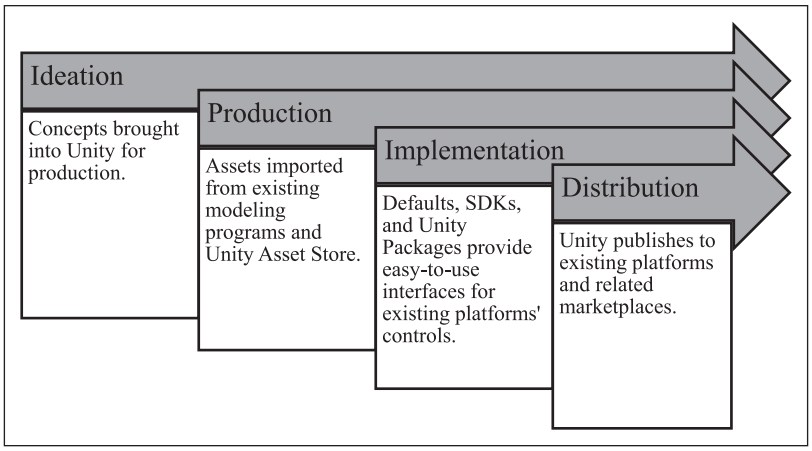

Figure 1. The game development production pipeline including Unity's platform tool features.

Implementation. At implementation, player controls are coded into the game, as are interactions with assets and objects. Software development kits (SDKs),[8] APIs, and packages bring the project to life by incorporating game controls and other forms of interactivity. This process often involves coding for the player, other objects, and characters in relation to the desired hardware on which the game will be played. As an example, navigation and interaction in VR requires the use of controllers and headsets. To assist programming for both, assets with built-in code are available in the Asset Store, such as the Vive Input Utility ("VIVE Input Utility—Asset Store," n.d.), made by the VR hardware manufacturer HTC. These also lock users into making projects for specific platforms. To switch to another piece of hardware, they must refit the content, often using other software packages. Unity increasingly attempts to standardize such code in their engine, making it even more essential for hardware developers.

Distribution. In this phase, the game's environment, code, and assets are compiled for certain platforms including mobile phones, computers, consoles, and so on. In Unity's case, distribution occurs through its "build and run" feature. Since excluded formats are significantly more difficult for designers to manage and the engine periodically removes underused platforms such as Tizen mobile and the Samsung TV formats (Akshay, 2017), Unity has become the arbiter of desirable arenas for publishing games, VR and other 3D content.

Unity and Lock-In

At each stage of this workflow, Unity locks-in existing platforms at a functional level. It achieves this through *intra*operability: its Asset Store and other platforms affect each stage of the production pipeline. Consequently, Unity reifies and further entrenches current platforms and facilitates their use to its benefit.

As a result, the game engine exacerbates distrust about platforms already voiced by scholars. By vying for market dominance and using its own packages and kits, Unity is "crowding out . . . exceptions and alternatives" (Nieborg & Poell, 2018, p. 4289) through platform dependence. Similarly, its move to infiltrate other markets, as well as launch an asset store for 3D objects and code, ultimately conforms to the "underlying logic of the platform" (Helmond, 2015, p. 8), which serves to make data not only expansive and accessible but also commodifiable and restrictive.

Dependence on platforms seems immutable as a result of the productive nature of the program and its explicit marketing as a tool for creating applications. It exploits the inherent power of platforms, which hold "undeniable benefits" for users who can easily produce work with them, but, at the same time, allow major corporations "to gain footholds as the modern-day equivalents of the railroad, telephone, and electric utility monopolies of the 19th and 20th centuries" (Plantin et al., 2016, p. 295). Unity ultimately establishes a production pipeline where the use of platforms seems like the best (if not only) option to conceive of, make, and distribute games.

The consequences for users, as Lanier asserts, is that the multitude of creative possibilities are increasingly locked-in, which affects all stages of game development and manifests in countless ways. The engine itself locks gaming conventions into any 3D process—for example, navigation through the software includes the W, A, S, D key configuration common to computer games. Core assets have names such as the "FPSController," which is labeled and patterned after first-person shooter games.

However, Lanier also suggests that such lock-in has a profound, but complicated relationship with users, not only limiting what they can do but also providing the rules by which they create. Whitson (2018) similarly states that programmers call Unity "voodoo software," which exhibits "a mind of its own . . . in a manner counter to users' input and goals" (p. 2324). This led to "unplanned 'features'" (p. 2327) and

framed how developers conceived of projects and interacted with one another.

What then is the influence of tools such as Unity on users as they adopt and interact—even playbor—with it? What is the effect of locked-in ideologies on the process of creation, particularly with new innovations? Insights into these questions surfaced through interviews with VR developers and enthusiasts.

Methods

Semi-structured interviews were conducted with content creators, enthusiasts, and developers of VR and Augmented Reality (AR) hardware as part of a larger project studying immersive technologies (Foxman, 2018). Interviewees were garnered from enthusiast meetups in the United States; some centered explicitly on game development, while others focused on business, education, and technical aspects of emerging media. As a consequence, a wide swath of users at various levels of expertise in Unity was represented. In addition, interviews were held with members of a private VR lab, which explicitly utilized the game engine.

Interviews occurred over a year and a half, starting in 2016. A total of 76% interviewed were developers or content creators, and 34% worked in development-oriented businesses. A total of 77% were male, reflecting the general attendance of the meetups.

In total, I conducted 90 interviews, which lasted between 15 min and 2.5 hr and occurred over phone, in-person, and email. All respondents were given the option to be anonymous, and as a consequence, have not been identified.

While similar topics were covered in each interview, I took a grounded approach by consistently re-theorizing follow-ups from primary questions. This directly affected the investigation of Unity. My original inquiry concerned workflow, the experience of VR development, use of devices/development kits, and overall perceptions of the medium, but the engine came up with such frequency in answers that I re-theorized the direction of my queries.

The result is not a complete description of using Unity, but rather an analysis of perceptions of the engine in the daily work of developers and enthusiasts, its relationship to the game industry and its significance in VR diffusion.

Unity, Virtual Reality, and the Limits of Platform Tools

Many exposed to Unity through meetups and the lab were bullish about VR's potential; it realized long-held childhood dreams and a ground floor opportunity to enter the games and tech industries. They used expressions like "revolutionary," "transformative," and "all-encompassing," and made bold claims about Unity and its impact on the world. "In terms of unlocking your potential, everybody talks about that—unlock a kid's potential. It's all talk, but Unity actually does it. And they give it to you for free . . ."

Respondents lauded Unity for its ease-of-use. Content could be made in merely 30 min and published as a rudimentary project. One interviewee found the immediate results "gratifying," and thought it would inspire those unfamiliar with programming to venture further into VR development. It was also efficient, especially its "drag and drop" feature which allowed assets to be placed into virtual environments with little difficulty. Furthermore, the interoperability of the platform was appreciated. Developers began projects by experimenting with Unity packages; one described how she downloaded demos, then modified them "because I couldn't code" until she got a new piece of hardware working. Citing the Unity Asset Store, another stated that the many premade assets and scripts which one could conveniently purchase, download, and "drag and drop" would familiarize any user with the engine who could then "patch things together" to make an application. The range of compatible packages was also praised. Another interviewee touted the "cool" even "beautiful" discussions that occurred online with every SDK release.

The benefits of Unity's interconnectivity and interoperability also extended to publishing. The low fee for doing so on popular marketplaces, such as the App Store, Google Play, or Steam, was extolled as a means to achieve success and increase output. Because of Unity's build and run features, the same project could be circulated across all of them, allowing for mass exposure: rather than taking an unpaid internship, an enthusiast recommended beginners spend US$25 on a one-time developers' license and release "a bunch of stuff" on the stores.

The unbridled optimism over Unity was tempered in some individuals, however, because specific forms of social and technical lock-in circumscribed the genre associated with VR, the identity of developers, professional interest, and ultimately its future direction.

Lock-in of Genre

In my interviews, many respondents had backgrounds in content creation, but not necessarily in game development. Yet, as one interview put it, "VR and AR is almost synonymous with gaming." Another called gaming and entertainment media the obvious "portals" for VR. This assumption extended to Unity. "I always wanted to make games" one respondent said. Because he wanted to make 3D content, he started using Unity, where he "then eventually found virtual reality." The engine and the medium were linked in their gaming orientation.

For those who enjoyed gaming, this was a delight. One interviewee described the entire process as "building a sandbox where people could play and create memories . . . for me, that's what life is all about." He compared Unity to Adobe Photoshop, which he tinkered with as a teenager, and enjoyed looking "at all the tools and then you see what's possible," saying there was

"a large aspect of play to this." He even enjoyed showing off each minor accomplishment to family, friends, and enthusiasts alike as he started building games and VR projects.

But for those who were disinterested in games, Unity only bolstered the overall perception that VR was dominated by the genre, as one interviewee put it: "If people are going to study VR in Unity I really think they should take some . . . classes in video game design and maybe even coding." The implication is a lock-in of games, game design, and even the game industry within the diffusion of VR. The engine and the medium can be used for a multitude of purposes such as architecture, art, and film, but, for early enthusiasts, Unity was primarily intended for gaming. This is best illustrated by a respondent who was designing an application for retail services. When looking to employ Unity developers, he was concerned that the engine "defined [a] set of rules" and mechanics around gaming, which comprised "the vocabulary of how we communicate" about VR and immersive media. Conflicted, he did not know if those rules would be what he wanted in his app.[9]

In a somewhat ironic twist, those who knew about the diversity of options for which Unity could be applied tended to come from a gaming background and often felt empowered when it came to working with the tool.[10]

Lock-in of Identity

Those "in the know" also tended to fall into age, gender, and socio-economic stereotypes: "I would also *broadly* characterize the white young males as 'dudes', and that they are coders or digital artists who are also avid gamers," wrote one interviewee via email about the VR community, adding "I would also guess that their income is above average. I've only met one expert level female developer in [Unity] who codes VR applications as her full-time job." Another interviewee summed it up as "more guys, more tech people . . ."

By contrast, one older female lab participant feared attending weekly meetings because she lacked a coding background. Another said it was "dis-enabling" and "shocking" that she had to learn both Unity and its interoperable code, the C# programming language, simultaneously. The engine and VR were viewed as part of a boys' club that was antithetical to the interests of more marginalized groups.

Respondents also sensed the absence of people of color within the enthusiast community. Two female interviewees noted the high proportion of males, followed by saying that African Americans were specifically underrepresented. One Korean student said she did not have anyone to "connect with." An African American developer stated she had to learn to "accommodate" herself to the meetups to "feel comfortable." Thus, just as the engine locked-in a specific genre (games) and with it a specific type of enthusiast (a gamer or game developer), it also locked-in the norms of the surrounding gamer culture, which has been critiqued for its focus on, marketing to and support of affluent White males at the exclusion of other social groups (Shaw, 2011). It was this faction that had the cultural knowledge, income, and ability to successfully use and deploy VR content.

Lock-in of Professionals

Because Unity was locked-into a wider set of platforms in game production, professionals knowledgeable in these toolsets were most successful with the emerging medium. Competence in VR production required not only familiarity with Unity but also learning 3D graphics programs like Maya to make VR. A respondent described the "steep learning curve" for anyone without a 3D modeling background. Another novice said it was hard to "ramp up" in and recounted many hours spent online researching and testing the engine.

As a consequence, Unity tended to lock-in developers, while locking out almost any other profession. Film and video editors had little patience to learn new software when they already had expertise in established design applications such as those in the Adobe Creative Suite. A hobbyist with a background in film critiqued the engine for not being cooperative enough. She could not do "simple" things easily like creating a video or audio loop. Ultimately, she found it "convoluted and messy," a "kind of *Lord of the Flies* of computer programming." In my interviews, there was almost a palpable desire for a tool better suited to the needs and practices of non-game makers.

These remarks betray an awareness of the restrictions by which developers had to work even while they were pushing Unity's creative envelope for VR. The engine principally afforded a platform for game creation but could be repurposed for other ends only with significant effort.

This also extended to distribution, where once again success with the engine was contingent upon platforms and tools that the professionals already understood or owned. Gamers and developers possessed the hardware required to easily build VR projects with Unity and were primed to produce and experiment with content. By contrast, average users would have to spend thousands of dollars and many hours to gain access to necessary technology to produce and distribute VR.

Lock-in of VR's Future

The previous observations anticipate Unity's role in molding VR's future. Those without the professional backgrounds or who did not identify with "gamers" found using the engine and associated platforms bewildering, or, at very least, impeding their creativity. Interviewees highlighted the implicit and explicit restrictions that will ultimately thwart VR's progress. One enthusiast stated, "I look at certain things about AR and VR in the future, and I'm like, 'We should be careful to think about what we're building . . .'" He went on to compare VR's development to the game industry's missteps when it came to promoting violence and then concluded, "We should think a

little bit about what we're building and why, before moving ahead at full pace." Thus, for all its revolutionary potential, there was a persistent concern about how the long-standing conventions of games and gaming, supported by Unity itself, might ultimately shape VR.

The popularity of Unity for VR production indicates path dependence accompanied by pervasive lock-in. Who can use the software (game developers), how they can use it (primarily for game development), what they can publish (game-related content), and where they publish it (app stores already connected to both Unity and games) are well-established, despite commercial VR's infancy.

Rules of the Game: Discussion

These interviews paint a contradictory picture about Unity: it is simultaneously a user-friendly program essential to realizing the potential of a budding medium, while also befitting a specific set of users acculturated to gaming.

Lock-in is present at all stages of production, where it set technical and ideological parameters. The engine, on both an intellectual and visceral level, affected each decision developers made, and required an awareness of, if not an acquiescence to, the conditions under which they worked. Developers needed Unity to successfully navigate VR production. Furthermore, since many early users were both producers and consumers (or players) of content, this caused a kind of self-selection—VR became fun for those eager to produce with Unity.

Under these circumstances, I argue that compliance to the standards supported by Unity is sustained through the coping mechanism of "playbor." As Kücklich (2005) describes, playbor's basic tenets are mirrored in developers' activities to the benefit of the corporation: their homemade assets and scripts end up in the Unity Asset Store and serve as de facto marketing; users' efforts add capabilities and consequently "shelf life" to the engine, and they test novel features when new hardware such as VR headsets are released. They even aspire to be hired by Unity, or related industries with which the engine is compatible.

More broadly, playbor manifests "play" as it functions within platformized labor practices. Adopters of Unity are "playing within rules," which entail the "lock-in" of the engine's norms and technical requirements. This was alluded to in interviews: "It feels like we're already locked in for shoot-em-up things and enterprise tools." For some (particularly those versed in the "rules" of cultural production surrounding Unity), playbor is rewarding and fun to master, reinforced by positive feedback loops (another key characteristic of play and games). This pleasure explains the passion of certain VR adherents; for them, the synchronicity and low barriers of entry are powerful incentives to continue work within the engine. But for those who neither have the desire nor the wherewithal to play, the environment is not as enjoyable.

Such rules reach beyond Unity to its business and market partners. The interoperability of platforms expands the "playing field" where Unity can function, but then requires developers to further invest time and effort to understand the guidelines of compatible companies and applications. At the same time, these companies must also interface and work within the engine's boundaries. Thus, as a platform tool, Unity sets the conditions by locking-in specific ways of working within the production environment. Furthermore, it does this at all levels of content creation: from inception to distribution. These rules ultimately support playbor and playborers who are amenable to work within its economic, technical, and social restraints.

From a business-to-business perspective, platform tools are indispensable. Interoperability has enabled experimentation with VR by almost every industry—there is no need to go back to the drawing board when utilizing the medium for architecture, games, or medicine (as examples). Yet, this creates a counterintuitive situation: the engine opens up new horizons for, while simultaneously dictating the conditions of, production.

A final consideration is the effect on industry when platform tools like Unity are standardized, locked-in, and played with. Respondents' perspectives insinuate that Unity is an important agent for reification. VR hardware manufacturers release headsets with APIs, software packages, and code, which are compatible with Unity. Then developers make use of them to create content that proliferates within the existing platform ecosystem, since it is a necessary part of Unity's pipeline for both innovating and publishing. Consequently, the platform "behemoths" already invested in VR and immersive media further solidify their status as technology leaders. Unity also reinforces its place as a vital emissary by locking-in industries, associated marketplaces and its own position within them. Practically, this means that VR will be associated with gaming for the foreseeable future, which is the foundation upon which Unity was built.

Furthermore, Whitson (2018) asserts that such tools are the first interface by which many hobbyists approach interactive content. As familiarity with these "agents . . . mentors, producers and community organizers" increases, the gap between the "developer and the tools-developer" (p. 2329) widens, endowing the latter with more power. The diffusion of software like Unity into a wide variety of emerging fields perpetuates this power imbalance as future innovations enter the marketplace. In short, lock-in, play/playbor, and the reification of compatible industries are becoming conventional in digital creation.

Reaching the Limit: Conclusions

Using Unity as a quintessential example of a platform tool, this article showed how platforms can provide the technical, social, and economic conditions to lock-in specific industries, design practices, and inequalities. Through interviews with enthusiasts of VR and other immersive media, a visceral apprehension of such lock-in was uncovered, where the potentially limitless possibilities of the emerging medium and

who can work with it are constrained by the engine. This does not mean that creativity is hindered, but rather is shaped by norms and rules embedded within the tool.

For users, this perception is implicit as they attempt to learn the engine and fashion content beyond games. However, these are only users' impressions. Future study should probe how the views expressed by respondents affect their daily practice, to further elucidate the influence of lock-in on creative processes.

However, these findings highlight three major issues in the examination of platforms and platformization. First, platform tools remain seriously understudied. Unity is not the only highly platformized and, to a degree, "de-professionalized" piece of sophisticated productivity software that has become routine to developers. New users feel emboldened through such applications and assume their integrity without critically assessing their ideological underpinnings. Thus, it is imperative for there to be continued scholarly analysis and rigorous critique of them.

A second issue is the intimate relationship between "platform behemoths" and consumer lock-in. Unity is just one example of the myriad ways that platform tools can sustain and entrench those in power. Developers need to work with industry-provided packages, kits, and assets—not to mention to commit extra effort, time, and money in the case of VR—to generate successful content. They, therefore, must succumb to the bounds of the existing industry within which companies like Unity are deeply embedded. This may explain why titans of tech are investing in similar software; Amazon's "Sumerian" game engine and Google's "Stadia" streaming service are just two examples. Such acquisitions strengthen these behemoths by broadening their reach beyond the simple capture of user data to devising ways to capitalize upon enthusiasts' labor.

Third, platform tools manifest the extent to which *intra*operability has disrupted traditional cultural production and the logic of platformization has informed the contemporary economy. Interviews reveal how a single platform tool impacts users' lives by prescribing the qualifications of developers, their production methods, and agendas. With this article's emphasis on emerging media, questions arise as to how novel technologies—from robotics to artificial intelligence—may be shaped by platforms, but more importantly, how platforms will shape the people and ideas that are instrumental in building these innovations.

In sum, cultural producers are encountering an increasingly rule-bound set of tools with which they must construct content. Those rules flow from the top down, rather than the bottom up, creating a path dependence for creativity. Still, the path dependence outlined may not inevitably lead to purely negative ends. Lock-in does obscure forms of creativity and development, but limits can also be powerful. Playing within rules allows for both enjoyment and the opportunity for prowess in production, and ultimately commands adherence to obtain mastery. After all, a game can only be well-played when its rules are known to all. Such knowledge also instigates subversion and modification of the rules, which are ordinary (even necessary) elements of play. When rules are imposed rather than mutually agreed upon, more egregious problems persist. Only by understanding the rules and standards with which cultural producers are playing can all parties find ways to win.

Declaration of Conflicting Interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) received no financial support for the research, authorship, and/or publication of this article.

ORCID iD

Maxwell Foxman https://orcid.org/0000-0001-6499-4372

Notes

1. This is in part because platforms are studied by many fields including management studies, media studies, cultural studies, and political economy. See Plantin, Lagoze, Edwards, and Sandvig (2016) for further review.
2. The examples have been critiqued for inflating differences between the competing products (Cantner & Vannuccini, 2017; Liebowitz & Margolis, 1995).
3. In the case of Unity, a marketplace is featured, but is not a core component of the engine.
4. At the same time, platform tools may be considered an evolution of middlebroware and codify Lesage's (2015) findings that users indulge in "playful, less disciplinarily sophisticated, experiments" (p. 106).
5. Application programming interfaces (APIs) or sets of underlying rules for using existing code in a software program. See Helmond (2015) for analysis in terms of platformization.
6. The same production pipeline is present for virtual reality (VR), augmented reality (AR), and other forms of virtual world building.
7. Whitson (2018) also describes this process with Unity and three-dimensional (3D) Studio Max.
8. A "Software Development Kit," or suite of code and examples that can be imported into Unity.
9. The full quote can be found in Foxman (2018, p. 243).
10. One genre absent from the analysis is pornography, which is credited as a reason for VHS locking-in over Betamax (Johnson, 1996). Interviewees rarely mentioned pornography, which may reflect how they were recruited—through semi-professional labs and meetups.

References

Akshay. (2017, November 6). Unity drops support for Tizen mobile and Samsung TVs in the latest Beta release. *IoT Gadgets*. Retrieved from https://www.iotgadgets.com/2017/11/unity-drops-support-tizen-mobile-samsung-tvs-latest-beta-release/

Axon, S. (2016, September 27). *Unity at 10: For better—or worse—game development has never been easier*. Retrieved

from https://arstechnica.com/gaming/2016/09/unity-at-10-for-better-or-worse-game-development-has-never-been-easier/

Bechmann, A. (2013). Internet profiling: The economy of data intraoperability on Facebook and Google. *MedieKultur: Journal of Media and Communication Research*, *29*, 72–91.

Beschizza, R. (2018, July 17). *The most popular engines for indie games*. Retrieved from https://boingboing.net/2018/07/17/the-most-popular-engines-for-i.html

Bodle, R. (2011). Regimes of sharing: Open APIs, interoperability, and Facebook. *Information, Communication and Society*, *14*, 320–337.

Bogost, I. (2008). *Unit operations: An approach to videogame criticism*. Cambridge, MA: MIT Press.

Cantner, U., & Vannuccini, S. (2017). Innovation and lock-in. In H. Bathelt, P. Cohendet, S. Henn, & L. Simon (Eds.), *The Elgar companion to innovation and knowledge creation* (pp. 165–181). Cheltenham, UK: Edward Elgar.

Downie, C. (2016, June 16). Evolution of our products and pricing. *Unity Blog*. Retrieved from https://blogs.unity3d.com/2016/06/16/evolution-of-our-products-and-pricing/

Foxman, M. H. (2018). *Playing with virtual reality: Early adopters of commercial immersive technology*. New York, NY: Columbia University. Retrieved from http://academiccommons.columbia.edu/download/fedora_content/download/ac:6q573n5tc8/content/Foxman_columbia_0054D_14522.pdf

Gillespie, T. (2010). The politics of "platforms." *New Media & Society*, *12*, 347–364.

Haas, J. (2014). *A history of the unity game engine*. Worcester, UK: Worcester Polytechnic Institute. Retrieved from http://web.wpi.edu/Pubs/E-project/Available/E-project-030614-143124/unrestricted/Haas_IQP_Final.pdf

Helmond, A. (2015). The platformization of the web: Making web data platform ready. *Social Media + Society*, *1*(2). doi:10.1177/2056305115603080

Johnson, P. (1996). Pornography drives technology: Why not to censor the Internet. *Federal Communications Law Journal*, *49*, 217–226.

Kücklich, J. (2005). Precarious playbour: Modders and the digital games industry. *Fibreculture*, *5*(1). Retrieved from http://five.fibreculturejournal.org/fcj-025-precarious-playbour-modders-and-the-digital-games-industry/

Labschütz, M., Krösl, K., Aquino, M., Grashäftl, F., & Kohl, S. (2011). Content creation for a 3D game with Maya and Unity 3D. Institute of Computer Graphics and Algorithms, Vienna University of Technology. Retrieved from https://www.researchgate.net/profile/Reinhold_Preiner/publication/267417785_Content_Creation_for_a_3D_Game_with_Maya_and_Unity_3D/links/554788c70cf26a7bf4d93df6.pdf

Lanier, J. (2011). *You Are Not a Gadget: A Manifesto*. New York, NY: Vintage.

Leap Motion. (n.d.). Available from https://www.leapmotion.com/

Lesage, F. (2015). Middlebroware. *Fibreculture Journal*, *25*, 89–114.

Liebowitz, S. J., & Margolis, S. E. (1995). Path dependence, lock-in, and history. *Journal of Law, Economics, & Organization*, *11*, 205–226.

Margolis, S., & Liebowitz, S. J. (1998). *Path dependence: New Palgrave dictionary of economics and law*. London, England: Palgrave MacMillan.

Martens, B. (2016). *An economic policy perspective on online platforms*. Retrieved from https://doi.org/10.2139/ssrn.2783656

Montfort, N., & Bogost, I. (2009). *Racing the beam: The Atari video computer system*. Cambridge, MA: MIT Press.

Nieborg, D. B., & Poell, T. (2018). The platformization of cultural production: Theorizing the contingent cultural commodity. *New Media & Society*, *20*, 4275–4292.

Panourgias, N. S., Nandhakumar, J., & Scarbrough, H. (2014). Entanglements of creative agency and digital technology: A sociomaterial study of computer game development. *Technological Forecasting and Social Change*, *83*, 111–126.

Plantin, J.-C., Lagoze, C., Edwards, P. N., & Sandvig, C. (2016). Infrastructure studies meet platform studies in the age of Google and Facebook. *New Media & Society*, *20*, 293–310.

Schmalz, M. (2015). *Limitation to innovation in the North American console video game industry 2001-2013: A critical analysis*. London, Ontario, Canada: Western University. Retrieved from https://ir.lib.uwo.ca/etd/3393/

Schwarz, J. (2017). Platform logic: An interdisciplinary approach to the platform-based economy. *Policy & Internet*, *9*, 374–394.

Shapiro, C., & Varian, H. R. (1998). *Information rules: A strategic guide to the network economy*. Cambridge, MA: Harvard Business Press.

Shaw, A. (2011). Do you identify as a gamer? Gender, race, sexuality, and gamer identity. *New Media & Society*, *14*, 28–44.

Takahashi, D. (2015, August 16). *In the game engine wars, Epic and Unity aim at enabling VR*. Retrieved from https://venturebeat.com/2015/08/16/in-the-game-engine-wars-epic-and-unity-aim-at-enabling-vr/

Unity public relations fact page. (n.d.). Retrieved from https://unity3d.com/public-relations

VIVE input utility—Asset store. (n.d.). Retrieved from https://assetstore.unity.com/packages/tools/integration/vive-input-utility-64219

Ward, J. (2008, April 29). What is a game engine? *GameCareerGuide.com*. Retrieved from https://www.gamecareerguide.com/features/529/what_is_a_game_.php

Whitson, J. R. (2018). Voodoo software and boundary objects in game development: How developers collaborate and conflict with game engines and art tools. *New Media & Society*, *20*, 2315–2332.

Author Biography

Maxwell Foxman (PhD Columbia University) is an assistant professor of Media Studies at the University of Oregon. His research interests include Virtual and Augmented Reality, the game industry, and the gamification of journalism and other professions. His work has appeared in journals such as *Games and Culture, Computational Culture*, and *First Monday*.