Software Development Kits and Product Innovation: Modularity and Ecosystem Perspectives

Abstract

We observe a growing number of software development kits (SDKs) are externally available. We explore how using externally available SDKs affects the product innovation process. We characterize (1) product development by using externally available SDKs as solving nearly modular problems and (2) product development by building inhouse SDKs as solving an integral problem (i.e., relatively non-modular). We explore the video game industry, in which software development kits are called game engines, and game developers explore theme innovations (i.e., non-technological dimension) as well as technological innovations. Findings suggest that on average, using commercial game engines facilitates module-level innovations but less likely to introduce system-level innovations which require a cross-module coordination. Also, the number of commercial game engine users is an important predictor of product innovation. If when the number of users is not sufficiently large, its weakness in system-level innovations exacerbated, and the strength in module innovation is also weaker, showing that harnessing the power of positive feedback (i.e., ecosystem effect) between the number of users and the quality of engines matters.

Keywords: Software Development Kits, Product Innovation, Modularity, Ecosystem

1. INTRODUCTION

This study explores the impact of using a software development kit (hereafter, SDK) on the direction of product innovation. We observe a growing number of software development kits, which are externally available technological resources. Product innovation is a complex problem which requires close coordination between different types of resources: not only technological resources but also other resources like market resources and knowledge (e.g., Teece 1986, Eggers, Grajek, Kretschmer 2020). This leads to the following question: How does the use of an external technological resource affect the direction of innovations in different dimensions and overall product innovation process? In this study, we attempt to answer this question specifically from the modularity and ecosystem perspectives.

Since Simon (1962) emphasized that nearly decomposable systems could help tackle the challenge in the complexity of product design, the near-decomposability forms the cornerstone of modular designs in complex products (Parnas 1972, Sanchez and Mahoney 1996, Baldwin and Clark 2000). We characterize (1) product development by using externally available SDKs as solving modular problems and (2) product development by building inhouse SDKs as solving an integral problem (i.e., relatively non-modular). The characterization represents an important first step toward exploring the relationship between technological change and firms' product innovation that lies at the heart of the literature on complexity and innovation (e.g., Simon 1962, Levinthal 1997, Kogut and Jain 2013, Posen, Lee, Yi 2013). A modular design is based on a principle of encapsulating interdependencies within self-contained units called modules and minimizing reciprocal interdependencies between modules. In particular, Ethiraj et al. (2008)

emphasize that modular problem structures more easily facilitate incremental and localized innovation within modules and thus help reduce the complexity in product design.

We argue that product development with external SDKs (i.e., solving nearly-modular problems) may facilitate module-level innovations but may weaken system-level innovation which requires cross-module coordination. We explore the video game industry, in which SDKs are called game engines. In our setting, externally available SDKs are commercial game engines. The use of commercial game engines (a technology module) facilitates the separation between technological modules and non-technological modules. This separation helps firms reduce interdependencies between the two modules, leading companies to do a better job in within-module innovations (both within-technological-module innovations and within-theme-module innovations.) However, because of the fact that modularization minimizes the interdependencies between modules, they are less likely to attempt to introduce a system-level innovation which requires coordination between changes in both technological and theme dimensions at the same time.

Prior work notes that unlike inhouse SDKs, commercial SDKs are an ecosystem nature (e.g., Jacobides et al. 2018). The number of commercial game engine adopters and the quality of engines have positive feedback because engine providers could improve their engines by reinvesting their profits. Thus, this positive feedback dynamic leads a small number of companies corner the market. In particular, the top two commercial engines, Unreal and Unity, succeed in attracting a larger number of adopters and the features of their engines cover a broader area of technologies than other engines. Results suggest in 2010s, Unreal and Unity users overcome the commercial engines' weakness in system-level innovation. However, when the number of users is not sufficiently large (the case of other engines), the commercial engines' weakness in system-level innovations exacerbated. Not only that, commercial engines' advantage in module-level innovations is very weak.

This study contributes to the literature on complexity and innovation (e.g., Zhou and Wan, 2017, Ethiraj and Zhou, 2019, Keum 2019). Baumann, Schmidt, and Stieglitz (2019) note that to date, however, theoretical research has been long, but empirical research has been very sparse. This study offers a nuanced characterization of SDKs and evidence on the elusive relationship between SDKs and production innovation.

The rest of the paper is organized as follows. In the next section, we introduce our research setting, the video game industry in some detail. We then review the research literature on modularity and ecosystem and how modularity affects the direction of innovation and develops our hypotheses. In the following section, we describe the data and sample, the variables, empirical strategy, and summary statistics. Finally, we show the results and discuss their implications for research on modularity and ecosystem.

2. SETTING: SOFTWARE DEVELOPMENT KIT AND GAME DEVELOPMENT

2.1. Software development kit in the Video Game Industry: Game Engine

A software development kit (SDK or "devkit") is a collection of software development tools in one installable package. They ease the creation of applications by having compilers, debuggers, application programming interfaces (APIs), and libraries. Nowadays, many third-party companies provide software development kits to other companies to boost sales in their other products and services. For example, Apple's iOS SDKs are for attracting programmers to develop applications for their Apple products. We observe the surge of popularity in providing and using externally available software development kits.

A game engine is a software development kit designed for people to build video games. Figure 1 shows a screenshot of a game engine (Unity engine). It has reusable components for developers to code and plan out a game easily without building one from the ground up. They are technological modules that typically provided by a game engine includes a rendering engine for 2D or 3D graphics, a physics engine or collision detection, sound, scripting, animation, artificial intelligence, networking, streaming, memory management, threading, localization support, scene graph, and may include video support for cinematics. Its system enables beginners to build an entire game without writing a line of code.

Insert Figure 1 about here

There are two types of game engines: inhouse game engines and commercial game engines. Inhouse game engines are game engines that were developed by game developers. First, some video game companies develop their own game engines and reuse them for subsequent video game development. These engines are called in-house game engines. The list of the top 10 popular inhouse game engines are in Table 1 Panel A. For example, EA games has developed Frostbite engine and used it to develop 35 games. The second type is commercial game engines. Some video game companies like Epic Games are specialized in developing game engines and license these engines to other video game developers and receive licensing fees. The list of top 10 popular commercial game engines in Table 1 Panel B. Unreal and Unity engines are the top 2 engines. They have unparalleled popularity compared to other engines. In particular, 586 games were developed with Unreal engines, and 1,535 games were developed with Unity engines. These numbers are much larger than the number of reuses of inhouse engines.

The popularity of commercial game engines has been increasing as shown in Figure 2. In the 2010s, the popularity increased significantly; for example, in 2016, about 25% of video games were developed with commercial game engines.

Insert Table 1 and Figure 2 about here

2.2. Game development and locus of innovation: technological and non-technological dimensions

A video game is a complex product that requires not only technological capabilities but also non-technological capabilities on understanding what type of themes (e.g., storyline, art, design) will appeal to which types of customers (Gregory, 2017). In particular, Kanode and Haddad (2009) note that "Games that make it to the store shelf can still fail from flawed code or a lack of entertainment value. The potentially "fun" game that has a beautiful storyline and art or has an engaging interface will achieve fame or notoriety based on its software foundation." Therefore, while technology is an essential part of product design and development, game development requires capabilities for the non-technical dimension (i.e., theme dimension). Thus, the capabilities related to theme design is related to market knowledge, which helps companies understand what kind of themes will appeal to what types of customers. Thus, the locus of innovation in game development lies theme as well as technology.

The variety of personnel in game development shows that technology is not the only factor determining the success of a video game. In particular, our data analysis of MobyGames

Database, which has extensive information on job titles on game developers, shows that 28.9% of personnel are technology-related (e.g., coding, programmer, developer, engineering, tools, or quality assurance), 32% of personnel are theme-related (e.g., game design, artwork, composers, animation, graphics, or movies). The other 38% of people are doing supporting functions like directors, administration & support, or marketing.

Now, we turn to introduce what types of technological and theme components have been introduced. Table 2 Panel A shows what technological elements have been introduced and used. We list the top 20 technology elements. Most technological elements were introduced in the 1980s and 90s. The reason why many technological elements were introduced in 1988 is that the Gameopedia database covers from 1988. Since the 1990s, the novelty in the technological dimension mainly comes from combining different technological elements. Table 2 Panel B shows the most popular technological element combinations. As Table 2 covers the most popular combinations, it is hard to see whether recent games also introduce new technological element combinations. Nintendo's Super Mario Odyssey (released in 2017) is an exemplary case of the introduction of a new technological element combination. It has a new combination of the four technological elements: 2D/3D, Side on view, Third-person view, and 2 people coop play on the same screen.

Insert Table 2 about here

There have been innovations in theme dimensions too. Like technological element innovation, most theme keywords were introduced in the 1980s and 1990s (Table 3 Panel A), but a recombinant search by having multiple themes in one game has been a major way to differentiate games as shown in Table 3 Panel B. Still, game companies have been introducing new theme element combinations. An example is Batman Arkham City (released in 2011) which introduced a new theme element combinations: Crime, Law Enforcement, Super-hero, and Detective.

Insert Table 3 about here

Finally, a commercial game engine is an externally available technological module. A commercial game engine help game companies separate technological dimension and non-technological dimensions. Especially, when game companies are using commercial game engines, the separation of technological and non-technological dimensions becomes prominent. Gregory (2017) articulates this point with an example of Doom as follows.

> "Doom was architected with a reasonably well-defined separation between its core software components (such as the three-dimensional graphics rendering system, the collision detection system, or the audio system) and the art assets, game worlds, and rules of play that comprised the player's gaming experience. The value of this separation became evident as developers began licensing games and re-tooling them into new products by creating new art, world layouts, weapons, characters, vehicles, and game rules with only minimal changes to the "engine" software." (Gregory, 2017, p. 11).

3. LITERATURE REVIEW

3.1. Modularity in complex production innovation

In his work on the architecture of complexity, Simon (1962) points out that systems that are nearly decomposable reduce the complexity of the challenge in product design. The near-decomposability become the backbone of modular designs in complex products (Parnas 1972, Sanchez and Mahoney 1996, Baldwin and Clark 2000). Ethiraj et al. (2008) emphasize that it is generally accepted that a modular design is based on a principle of encapsulating interdependencies within self-contained units called modules and minimizing reciprocal interdependencies between modules. Encapsulating interdependencies and minimizing reciprocal interdependencies make a system nearly decomposable. As a result of this, modular problem structures more easily facilitate incremental and localized innovation within modules and thus help reduce the complexity in product design.

Conversely, in nonmodular (or integral) structures, the management of interdependencies is not the primary guiding principle of design (Ethiraj and Levinthal 2004, Fang and Kim 2018). As the size of systems grows bigger, the negative effect of interdependencies on product design grows exponentially. Other things held constant, for systems of identical size the complexity of an integral design will be significantly greater than the complexity of a modular one (Ethiraj et al., 2008).

We characterize (1) product innovation by using externally available SDKs as solving a nearly modular problem and (2) game development by building inhouse SDKs as solving an integral problem (i.e., relatively non-modular). Broadly speaking, following the extant research on the role of modularity in product innovation, product development with a modular structure may facilitate innovation with modules but may weaken system-level innovation which requires cross-module coordination. Specifically, in our setting, externally available SDKs are commercial game engines. As the use of commercial game engines (a technology module) facilitates the separation between technological modules and non-technological modules, companies will do a better job in module-level innovations when they develop games. Thus, because of the fact that modularization minimizes the interdependencies between modules, they are less likely to implement a system-level innovation that requires coordination between changes in both technological and theme dimensions at the same time.

3.2. Software development kit as an ecosystem

Unlike in-house game engines, commercial game engines require managing an ecosystem of the innovative platform. An important characteristic of ecosystems is closely related to modularity. Modularity creates the conditions for an ecosystem to emerge (e.g., Iansiti & Levien, 2004, Teece 2014, Adner 2017). In particular, technological modularity allows interdependent components of a system to be produced by different producers, with limited coordination required (Adner, 2012; Adner & Kapoor, 2010; Kapoor & Lee, 2013). Companies have autonomy in how they design and innovate their respective modules (Jacobides et al. 2016). More modularization has been associated with a greater prevalence of ecosystems in a number of sectors, from telecommunications to financial services to mobility. Many of the sectors that have been studied in the context of ecosystems—IT, telecommunications, video games—tend to be more modular, suggesting that ecosystems may well be a distinct solution to the problem of inter-firm coordination, distinct from the use of alliances, supply chains, or market-based interactions.

Another key aspect of the ecosystem is increasing returns to the adoption of new technology. Arthur (1989) notes that modern, complex technologies often display increasing returns to adoption in that the more they are adopted, the more experience is gained with them, and the more they are improved. The software development kits are a good example of a technology with increasing returns to adopters. In particular, if an SDK provider does not attract enough users, it may not have enough revenue or profit to reinvest into the improvement in their software development kits. However, if the SDK provider succeeds in attracting a sufficient number of adopters and create enough profits, they can expand functions and add state-of-the-art functions in their kits so that the kit adopters could introduce new technology combinations by using such new functions.

3.3. Adoption of external SDKs and innovations in technology-module

In our setting, there are two important modules in game development: technological features and theme designs. First, we address the impact of the adoption of commercial game engines on innovations within the technology module. We will look into the impact of using a commercial game engine on each component. The impact of using commercial game engines is more direct in the case of innovation within technological dimensions. Since the 1990s, the introduction of new technological elements was very rare but most game companies differentiate their games from others by combining different technologies. We argue that recombinant innovations in the technological module can be facilitated by using commercial game engines. Commercial game engine adopters do not need to develop technological elements for themselves. If companies use commercial game engines, they do not have to make technology from scratch but can combine existing technological elements provided by commercial game engines.

However, there might be a downside to using a commercial game engine. In particular, companies may experience such a downside if they want to differentiate from other companies in the technological dimension. If many companies are using the SDK, the kit may become a standard SDK. Prior research on standard components has mainly explored the case of physical standard components. It is generally accepted that the use of standard components is efficient due to economies of scale, but tends to limit differentiation possibilities (Ulrich 1995). Would the same logic apply to SDKs which are not physical components? We argue SDKs play differently because unlike physical standard components, companies can utilize different parts of functions within an SDK. We argue that the combinatorial possibilities within the SDK are important boundary conditions on whether the adoption of commercial game engines chokes creating new things. The combinatorial possibilities may be low when the toolkit is not well developed. Especially, SDK providers do not attract enough users, they may not have enough revenue or profit to reinvest into the improvement in their software development kits. However, when SDK providers succeed in attracting a sufficient number of adopters and create enough profits, they can expand functions and add state-of-the-art functions in their SDKs so that the kit adopters could introduce new technology combinations by using such new functions.

As Ulrich (1995) notes, another critical problem of physical standard components is that standardization usually acts as an inertial force preventing firms from adopting a better component technology because of compatibility issues in the installed base of products. Software kits are relatively more flexible in attaining compatibility with the installed base. Therefore, we hypothesize:

H1. (Technology module innovation) Game companies that use commercial game engines (SDKs) will be more likely to introduce new-to-the-world technological feature combinations (innovation within the technological dimension).

3.4. Adoption of external SDKs and innovations in theme-module

Another important dimension in game development is theme design (a non-technological dimension). Here, we address the impact of using commercial game engines on innovations in theme-modules. Even a small change in one component can spread to a broader system through interdependencies (Levinthal 1997). In particular, a change in a particular component may engender a ripple effect on a broader system (Zhou 2011, Zhou and Wan 2017). Here, we address how a change in technological module leads to the change in innovation behavior in another component of product design: innovation in theme-module.

When companies use commercial game engines, they can start with a more prepared toolkit and explore new possibilities in other dimensions more easily. Thus, they do not have to spend money on developing technological toolkits but allocate those spendings to theme designs. Especially, many potential game developers who have a bottleneck with technology can explore broader search space on the theme dimension.

Empirical research shows that bottleneck shapes investment and product innovation. For example, Ethiraj (2007) notes that when companies face a bottleneck component, they increase investment in that component. Thus. if they do not face a bottleneck, they can allocate more resources to other components like theme design. And this positive effect of commercial game engines on theme innovation will be more prominent, as the game engine companies attract more adopters and offer a higher-quality engine to them. Game engines with more and better features in technological modules (even though they are not new-to-the-world elements) will be less likely to become a bottleneck in game production. Thus, search in non-technological dimensions will be facilitated by using externally available kits. Therefore, we hypothesize:

H2. (Theme module innovation) Game companies that use commercial game engines (SDKs) will be more likely to introduce innovations within the theme dimension.

3.5. Adoption of external SDKs and system-level innovation (co-occurrence of innovations in both tech-module and theme-module

Yet, while modularity may be necessary for ecosystems to function, it is clearly not sufficient for product innovation. As Baldwin and Clark (2000), and Jacobides and Winter (2005) argued, modularization and the subsequent separation between technological modules and non-technological modules may have a weakness in introducing system-level innovation which requires changes in different modules and subsequent coordination. To introduce such innovation, the management by hierarchy (i.e., an organization or inhouse development) will be a better way. Even though technological modularity enables the division of innovative labor between technological and theme dimensions. Still, there is room for game developers that have in-house game engines to bring an innovative game that requires coordination between two modules. This system-level innovation would not be easy for game developers which use commercial game engines. Therefore, we hypothesize:

H3. (System-level innovation) Game companies that use commercial game engines (SDKs) will be more likely to introduce system-level innovation which changes both technological and theme dimensions.

4. EMPIRICAL STRATEGY

4.1. Sample and Database

To examine the effect of a start-up's toolkit choice on its differentiation strategy, we collect a dataset of 6,751 newly established start-ups in the video game industry from 2000 to 2017. This dataset was collected from three major sources: MobyGames and Metacritic (firm, genre, platform, and project-specific information) and Gamepedia (technology and theme elements).

4.2. Variables

4.2.1. Independent variable

When we test hypotheses, the unit of analysis is the firm. The independent variable is a dummy, $Dummy\_SDK_{it}$, that is equal to 1 when startup *i* used one of the SDKs in the list in Figure 2 Panel A (i.e., Unreal, Unity, and other commercial engines.) and is otherwise (i.e., developing and using the proprietary game engine) 0. We identify which startups are using which game engines by using multiple sources: Gameopedia, Unreal Wiki, Wikipedia, and manual search.

4.2.2. Dependent variable

We measure the two different types of innovations: (1) technological innovation and (2) theme innovation (i.e., non-technological innovation). First, we measure technological differentiation to measure vertical differentiation by using the Metacritic database. The Gameopedia database offers what technological elements are used for each game. The average number of technological elements used per game is 2.46. We use a dummy for technological innovation, $New\_Tech\_Combination_i$, which is equal to 1 if the game has one or more new-to-the-world technological element combinations.

Second, we measure the theme innovation to measure the introduction of new theme element combinations. The Gameopedia database offers theme elements for each game. Each game in our sample has on average 0.99 keywords. The overall number of theme keywords is 218, and the number of theme element combinations are 2,507. We use a dummy for theme innovation, $New\_Theme\_Combination_i$, which is equal to 1 if the game has one or more new-to-the-world theme element combinations.

4.2.3. Control variables

We control for (1) the number of platforms the game was introduced (Mobygames, Metacritic, Gameopedia), (2) the number of people who participated in the game development (Mobygames), (3) the number of game companies that use the same game engine (Mobygames, Gameopedia, Wikipedia), (4) dummy equals to 1 if the game's theme is licensed from other sources (Mobygames), (5) dummy equals to 1 if the game's main platform is a video console (Mobygames, Metacritic, Gameopedia), (6) the number of games released in the same genre (Mobygames, Metacritic, Gameopedia), (7) publisher age (Mobygames, Metacritic, Gameopedia), (8) publisher size by the number of games published in the same year (Mobygames, Metacritic, Gameopedia), and (9) publisher experience by the number of cumulative games published (Mobygames, Metacritic, Gameopedia). We also include (4) year dummies and (5) genre dummies. We also use these variables to match similar firms before running the main stage regressions.

4.3. Empirical specification

4.3.1. Baseline OLS models and endogeneity issues

We use OLS regressions as the baseline tests of my hypotheses to validate the impact of whether video game company *i* which uses SDKs at year *t* changes its differentiation strategy in theme innovation and tech innovation. We add a vector of control variables that might influence the video game company's decision on adopting commercial SDKs. Thus, our initial specification is

$$Dependent\_Variable_i = \beta_0 + \beta_1 SDK_i + \beta_2 X_i + C_i + G_g + T_t + e_{it}, \quad (1)$$

where *i* indexes firms, and *t* calendar time, X_{it} is a set of observable characteristics of the firm as described above as control variables, C_i is country fixed effects, G_{it} is genre fixed effects, and T_t is the year-fixed effect. Whereas equation (1) controls for correlation between using an SDK and control variables, one may still be concerned about selection based on omitted variables (Hamilton and Nickerson, 2003). In an ideal experimental design, we would randomly assign development kit status and measure the ex-post difference in their differentiation strategy. In practice, we observe changes in both the practice of choosing development kits and differentiation strategies.

4.3.2. Coarsened exact matching (CEM)

We address this endogeneity issue by utilizing CEM (Iacus et al., 2009). CEM matching estimators control for selection bias by creating a matched sample of treatment and control observations that are similar in respect to the observable characteristics. To implement CEM, continuous variables are 'coarsened' into splines for the purposes of creating 'strata', or discrete mutually exclusive bundles of control variables. Treatment and control group observations are then matched exactly within each stratum, which eliminates the need to compare the means of the treatment and control groups after matching. We allow for unbalanced matching within each strata, as recommended by Iacus et al. (2012). Then, we adjust the second stage regressions by weighting so that the results can be interpreted as average treatment effects.

4.4. Summary Statistics

Table 4 reports descriptive statistics on all the variables at the game-year level. First, the descriptive statistics for the independent variable show that the proportion of the game-years that adopt a commercial game engine is 18.9%. Second, we have the three game-level dependent variables: each dummy variable takes the value of one if game i introduces a tech module-level innovation, a theme module-level innovation, and a system-level innovation, accordingly. The proportion of the game that introduces a tech module-level innovation is 1.97% and the standard deviation is relatively large (0.1390). The proportion of the game with a theme module-level innovation is 15.76% and the variance ha a large value (0.3644). Lastly, the proportion of the game with a system-level innovation is 0.58% and the standard deviation is 0.0758. Also, we report descriptive statistics on the commercial engine subsample and inhouse game engine subsample in Table 5. The commercial engine sample shows a higher average on the proportion of the game with a theme module-level innovation than the inhouse engine sample.

Insert Table 4 and 5 about here
----

5. RESULTS

5.1. Does the adoption of commercial game engines lead to innovations within tech-module?

We test the first hypothesis that game companies that use commercial game engines will be more likely to introduce new-to-the-world technological feature combinations. We compare games are developed by commercial engines and games are developed by in-house engines. Table 6 shows the results of tests on the impact of game engines on the tech module-level innovation. We estimate the four different versions of the same question. OLS without control variables, OLS with control variables, logistic regression with control variables, and coarsened exact matching regression with control variables.

Column 1 reports estimates from a simple OLS specification without control variables. We find a strong correlation between the adoption of a commercial game engine and the tech module-level innovation. Specifically, the estimate (0.0397) from OLS with the control variable (which means the 3.97% point increase) suggests that tech module-level innovation increases by 3.97% point in games that adopt commercial game engines compared to those that adopt in-house game engines. Next, Column 2 shows the results of the same model after adding control variables and other effects, which show the coefficient is larger than the coefficient in Column 1, and the coefficient (0.0477) is positive and significant; the *t*-statistic of the coefficient is 3.44.

Column 3 presents estimates from the logistic regression. We calculate the marginal effects of the estimates. The coefficient (0.0447;4.47% increase in the chance of tech module-level innovation) has a similar value with OLS models. Finally, we present estimates from the OLS model after matching to controls for observable differences between games that adopt commercial game engines and games that adopt in-house game engines. The coefficient from the matching model is 0.0667 and its t-statistics is 4.38. Collectively, the findings in Table

6 suggest that when game companies use commercial game engines, they will be more likely to introduce tech module-level innovation than when they use in-house game engines.

Also, we test whether the number of commercial game engine users increases the positive impact of commercial game engines on innovation. We predict the benefit from the adoption of commercial game engines will increase with a larger number of users because of a positive feedback and ecosystem effect. Hence, we divide the full sample into two ways: (1) time periods (2000-2005, 2006-2011, and 2012-2018) and (2) popularity of commercial game engines (Unreal/Unity and other commercial game engines). We use three different regression models to estimate the relationship between the number of commercial game engine users and innovation.

In Table 7, Panel A shows the results from the subsample analysis on the tech module-level innovation by different time periods. While the numbers of commercial engines are not sufficiently large (during 2000-2011), adoption of commercial game engines even decreases tech module-level innovation (Column 1-3) or increases innovation but the coefficients are not strongly significant (Column 4-6). However, when the numbers of commercial engines became sufficiently large (since 2012), we find a very strong correlation between the adoption of a commercial game engine and the tech module-level innovation.

In Table 7, Panel B shows the results from the subsample analysis on the tech module-level innovation by different game engine popularity. While commercial game engines have a larger number of users with higher popularity, the benefits from the adoption of commercial game engines increase. When the numbers of commercial engines are not sufficiently large (during 2000-2011), the adoption of any commercial engines decrease tech module-level innovation. However, since the Unreal/Unity game engines attained more number of users than other commercial game engines (since 2006), only Unreal/Unity game engine increase the tech module-level innovation. The trend is more apparent when the Unreal/Unity game engine achieves a noticeably huge user base compared to other commercial game engines (since 2012). In sum, the number of commercial game engine users enhances the tech module-level innovation of games with commercial game engines.

Insert Table 6 and 7 about here

5.2. Does the adoption of commercial game engines lead to innovations within theme-module?

We turn now to the second hypothesis, which tests the impact of the adoption of commercial game engines on theme module-level innovation. Table 8 shows the results from the test on the impact of commercial game engines on theme model level innovation. We estimate the same four different regression models in the prior subsection. Column 1 reports estimates from a simple OLS specification without control variables. We find a strong correlation between the adoption of commercial game engines and theme model level innovation. Specifically, the number of theme module-level innovations increases by 17.02% point in games that use commercial game engines. Next, Column 2 shows the results of the same model after adding control variables and other effects. The coefficient is 0.145 and is strongly significant (t-statistic:9.71).

Column 3 presents estimates from the logistic regression. The coefficient is 0.1182 and its Z-statistics is 10.65. In Column 4, we present estimates from the matched sample. The coefficient from the matching model is 0.1531, and its *t*-statistic is 9.35. Collectively, the findings in Table 7 suggest that when game companies use commercial game engines, they will be more likely to introduce theme module-level innovation than when they use in-house game engines.

The trends regarding different time periods and the popularity of the game engine in the previous subsection are evident in theme module-level innovation as well. In Table 9, Panel A shows the results from the subsample analysis on the theme module-level innovation by different time periods. While the numbers of commercial engines are not sufficiently large (during 2000-2011), the adoption of commercial game engines increases innovation but the coefficient are not strongly significant (Column 1-6). However, when the numbers of commercial engines became sufficiently large (since 2012), we find a very strong correlation between the adoption of a commercial game engine and the theme module-level innovation. In Table 10, Panel B shows the results from the subsample analysis on the theme module-level innovation by different game engine popularity. Unlike tech module innovation, all the commercial game engines increase the theme module-level innovation (since 2012), but Unreal/Unity engines show slightly a higher theme module-level innovation. To sum up, the number of commercial game engine users also enhances the theme module-level innovation of games with commercial game engines.

Insert Table 8 and 9 about here

5.3. Does the adoption of commercial game engines lead to system-level innovation (co-occurrence of innovations in both tech-module and theme-module)?

In table 10, we test the impact of commercial game engines on system-level innovation. We estimate the same four different regression models in the prior subsections. Columns 1 and 2 report estimates from OLS specifications. We find a negative relationship between the adoption of commercial game engines and system-level innovation. Specifically, the estimate (0.0044) from OLS with control variables (which means the 0.44% point increase) suggests that system-level innovation decreases by 0.44% point in games that use commercial game engines.

In Columns 3, we present estimates from the matched sample logistic regressions. The marginal effect is larger than estimates from the OLS models (-0.0093) but it is not statistically significant. Finally, we present estimates from the matched sample. The coefficient from the matching model is -0.0045, and its *t*-statistic is -2.18. In sum, the findings in Table 8 suggest that when game companies use commercial game engines, unlike tech module-level or theme module-level innovation, they will be less likely to introduce system-level innovation than when they use in-house game engines.

Table 11, Panel A shows the results from the subsample analysis on the system-level innovation by different time periods. While the numbers of commercial engines are not sufficiently large (during 2000-2011), the adoption of commercial game engines decrease system-level innovation (Column 1-6). However, when the numbers of commercial engines became sufficiently large (since 2012), the negative correlation between adoption of a commercial game engine and the system-level innovation is no longer significant. In Table 11, Panel B shows the results from the subsample analysis on the system-level innovation by different game engine popularity. When the numbers of commercial engines are not sufficiently large (during 2000-2011), the adoption of any commercial engines decrease system-level innovation. However, since the Unreal/Unity game engines have more number of users than other commercial game engines (since 2012), the negative correlation between adoption of a commercial game engine and the system-level innovation is no longer significant for Unreal/Unity game engine. In sum, the number of commercial game engine users alleviate the commercial game engines' weakness in system-level innovations.

Insert Table 10 and 11 about here
----

6. DISCUSSION AND CONCLUSIONS

We characterize (1) product development by using externally available SDKs as solving modular problems and (2) product development by building inhouse SDKs as solving an integral problem (i.e., relatively non-modular). The characterization represents an important first step toward exploring the relationship between technological change and firms' product innovation that lies at the heart of the literature on complexity and innovation (e.g., Simon 1962, March 1991, Kogut and Kulatilaka 2001, Posen and Levinthal 2012). We test our predictions using data on the use of commercial game engines in game developers. Unreal and Unity successfully create game engines for other developers. We find the correlational relationship between the use of commercial game engines and the direction of product innovation. Video game companies that used commercial game engines introduced module-level innovations (both technology module and theme module) but decrease system-level innovations which require coordination between innovations in technological module and innovations in theme module.

This paper exploits a unique and interesting empirical setting to explore the relationship between the use of SDKs and product innovation. However, the idiosyncrasies of the video game industry should not cloud the general applicability of my conceptual approach to a broad range of firms and industries. Indeed, starting with the seminal work of Simon (1962), scholars have long examined whether modularity facilitates innovation. In our setting, if firms use external SDKs, they can experiment with module-level innovations more easily. Our study particularly highlights that the characterization of SDKs as a technological module is missing from the literature. Although I do not claim that my model is universal to all industries, the findings of Ethiraj (2007), Ethiraj et al. (2008), Fang and Kim (2018) hint at its broad applicability to settings outside the video game industry.

In addition, although we identify the conditions under which using external SDKs leads to increased product innovation, and we show that my particular empirical context fits well with these conditions, I have little to say about the performance implication (either sales or review score) of using different SDKs yet. I assume product innovation attempts will lead to a wider standard deviation of performance. Extending on our work to explore performance implications would be an interesting extension, but that is beyond the scope of this work.

For practitioners, this study suggests an opportunity for forward-looking managers to anticipate the impact of the use of external SDKs on product innovation, one of the cornerstones of complexity and innovation. As we observe a growing number of SDKs and their popularity, managers need to anticipate the impact of using external SDKs on their product innovation outcomes. We offer implications to managers on how to formulate strategies to adapt to this shifted technological regime by depicting the impact of the SDK-related ecosystem has on product innovation.

REFERENCES

Adner, R. 2017. Ecosystem as structure: an actionable construct for strategy. ***Journal of Management***, 43(1): 39–58.

Adner, R. 2012. ***The wide lens: A new strategy for innovation***. Penguin Uk.

Adner, R., & Kapoor, R. 2010. Value creation in innovation ecosystems: How the structure of technological interdependence affects firm performance in new technology generations. ***Strategic Management Journal***, 31(3): 306–333.

Arthur, W. B. 1989. Competing technologies, increasing returns, and lock-in by historical events. ***The Economic Journal***, 99(394): 116–131.

Baldwin, C. Y., Clark, K. B., & Clark, K. B. 2000. ***Design rules: The power of modularity***, vol. 1. MIT press.

Baumann, O., Schmidt, J., & Stieglitz, N. 2019. Effective search in rugged performance landscapes: A review and outlook. ***Journal of Management***, 45(1): 285–318.

Ethiraj, S. K. 2007. Allocation of inventive effort in complex product systems. ***Strategic Management Journal***, 28(6): 563–584.

Ethiraj, S. K., & Levinthal, D. A. 2004. Modularity and Innovation in Complex Systems. ***Management Science***, 50(2): 159–173.

Ethiraj, S. K., Levinthal, D., & Roy, R. R. 2008. The Dual Role of Modularity: Innovation and Imitation. ***Management Science***, 54(5): 939–955.

Ethiraj, S., & Zhou, Y. M. 2019. Fight or flight? Market positions, submarket interdependencies, and strategic responses to entry threats. ***Strategic Management Journal***.

Fang, C., & Kim, J. 2018. The power and limits of modularity: A replication and reconciliation. ***Strategic Management Journal***, 39(9): 2547–2565.

Gregory, J. 2017. ***Game engine architecture***. AK Peters/CRC Press.

Hamilton, B. H., & Nickerson, J. A. 2003. Correcting for endogeneity in strategic management research. ***Strategic Organization***, 1(1): 51–78.

Iacus, S. M., King, G., & Porro, G. 2012. Causal inference without balance checking: Coarsened exact matching. ***Political Analysis***, 20(1): 1–24.

Iacus, S., King, G., & Porro, G. 2009. cem: Software for Coarsened Exact Matching. ***Journal of Statistical Software***, 30(i09).

Iansiti, M., & Levien, R. 2004. Strategy as ecology. ***Harvard Business Review***, 82(3): 68–81.

Jacobides, M. G., MacDuffie, J. P., & Tae, C. J. 2016. Agency, structure, and the dominance of OEMs: Change and stability in the automotive sector. ***Strategic Management Journal***, 37(9): 1942–1967.

Jacobides, M. G., & Winter, S. G. 2005. The co-evolution of capabilities and transaction costs: Explaining the institutional structure of production. ***Strategic Management Journal***, 26(5): 395–413.

Kanode, C. M., & Haddad, H. M. 2009. Software engineering challenges in game development. ***2009 Sixth International Conference on Information Technology: New Generations***, 260–265. IEEE.

Kapoor, R., & Lee, J. M. 2013. Coordinating and competing in ecosystems: How organizational forms shape new technology investments. ***Strategic Management Journal***, 34(3): 274–296.

Keum, D. D. 2019. Cog in the wheel: Resource release and the scope of interdependencies in corporate adjustment activities. ***Strategic Management Journal***.

Kogut, B., & Kulatilaka, N. 2001. Capabilities as real options. ***Organization Science***, 12(6): 744–758.

Levinthal, D. A. 1997. Adaptation on rugged landscapes. ***Management Science***, 43(7): 934–950.

March, J. G. 1991. Exploration and exploitation in organizational learning. ***Organization Science***, 2(1): 71–87.

Parnas, D. L. 1972. On the criteria to be used in decomposing systems into modules. ***Communications of the ACM***, 15(12): 1053–1058.

Posen, H. E., & Levinthal, D. A. 2012. Chasing a Moving Target: Exploitation and Exploration in Dynamic Environments. ***Management Science***, 58(3): 587–601.

Sanchez, R., & Mahoney, J. T. 1996. Modularity, flexibility, and knowledge management in product and organization design. ***Strategic Management Journal***, 17(S2): 63–76.

Simon, H. A. 1962. The architecture of complexity. ***Proceedings of the American Philosophical Society***, 106(6): 467–482.

Teece, D. J. 2014. The foundations of enterprise performance: Dynamic and ordinary capabilities in an (economic) theory of firms. ***Academy of Management Perspectives***, 28(4): 328–352.

Ulrich, K. 1995. The role of product architecture in the manufacturing firm. ***Research Policy***, 24(3): 419–440.

Zhou, Y. M. 2011. Synergy, coordination costs, and diversification choices. ***Strategic Management Journal***, 32(6): 624–639.

Zhou, Y. M., & Wan, X. 2017. Product variety and vertical integration. ***Strategic Management Journal***, 38(5): 1134–1150.

Figure 1: Unity engine editor

Figure 2: Popularity of Commercial Game Engines over Time

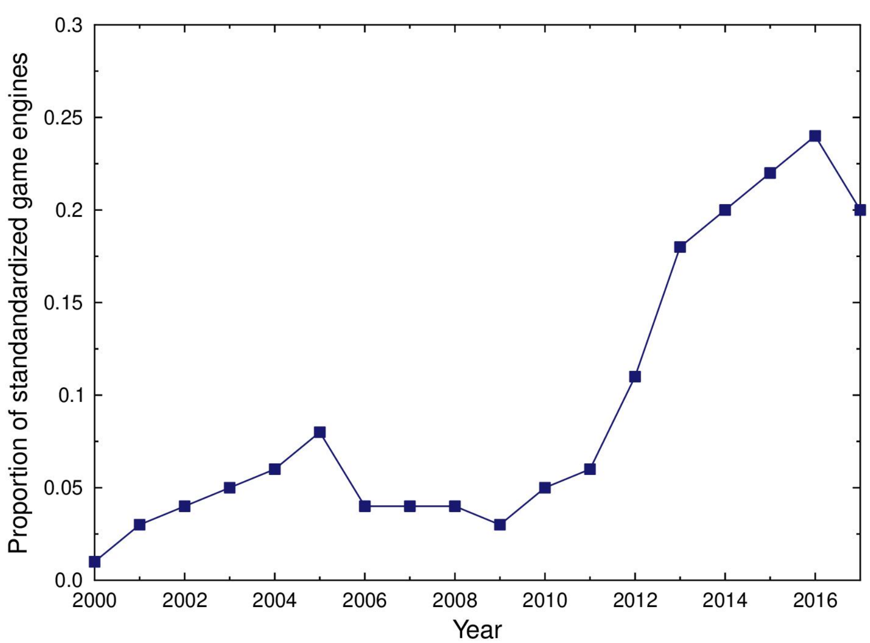

Table 1: Types of Game Engines and Their Popularity over Time

Panel A. Inhouse Game Engines

	Name	Developer	Initial release	No. of games	Notable games
1	Frostbite	Dice (EA)	2008	35	Battlefield series
2	MT Framework	Capcom	2006	28	Resident Evil series
3	Ignite	EA Sports	2013	21	NHL series
4	Anvil	Ubisoft	2007	20	Assassin's Creed series
5	Fox engine	Konami	2013	14	Metal gear Solid V
6	Rage	RAGE Technology Group	2006	10	Grand Theft Auto series
7	Decima	Guerrilla Games	2013	6	Death Stranding
8	Creation engine	Bethesda Game Studios	2011	5	Fallout series

Panel B. Commercial Game Engines

	Name	Developer	Initial release	No. of games	Notable games
1	Unreal Engine	Epic Games	1998	586	Gears of War series
2	Unity	Unity Technologies	2005	1,535	Cities: Skylines
3	Source	Valve Corporation	2004	45	Half-Life series
4	Renderware	Criterion Software	1993	161	Burnout series
5	PhyreEngine	Sony Interactive Entertainment	2008	64	Journey
6	Gamebryo	Gamebase	1997	33	Civilization IV
7	CryEngine	Crytek	2002	43	Far Cry series
8	Gamemaker Studio	Yoyo Games	1999	137	Gunpoint

Table 2: Summary of Technological elements and Their Combinations

Panel A. List of the top 20 Technological elements

	Tech keywords	Since	Obs.	Proportion	Category	Main genre
1	3D	1988	8166	0.1490	Graphic Tech	Action Shooter
2	1st-Person	1988	5413	0.0988	Perspective	Adventure
3	2D	1988	5155	0.0941	Graphic Tech	Puzzle
4	3rd-Person	1988	4699	0.0857	Perspective	Adventure
5	Side View	1988	3922	0.0716	Perspective	Action Platformer
6	2D Scrolling	1988	3328	0.0607	Visual	Action Platformer
7	Bird's-Eye View	1988	3039	0.0555	Perspective	Strategy
8	Top-Down	1988	2756	0.0503	Perspective	Strategy
9	Direct Control	1988	2563	0.0468	Interface	Action Shooter
10	Behind View	1988	2498	0.0456	Perspective	Racing
11	Point and Select	1988	1877	0.0342	Interface	Adventure
12	Fixed / Flip-Screen	1988	1623	0.0296	Visual	Puzzle
13	Isometric	1988	993	0.0181	Visual	Strategy
14	Online Versus (up to 8)	1990	841	0.0153	Network	Strategy
15	Online Versus (up to 2)	1988	678	0.0124	Network	Strategy
16	Versus on split screen (up to 4)	1989	616	0.0112	Versus	Sports
17	Versus on split screen (up to 2)	1988	607	0.0111	Versus	Sports
18	Free-Roaming Camera	1989	554	0.0101	Visual	Strategy
19	Online Versus (8+)	1988	552	0.0101	Network	Action Shooter
20	Cop-op on split screen (up to 2)	1988	539	0.0098	Co-op	Action Shooter

Panel B. List of top 20 Technology Combinations

	Tech combination	Since	Obs.	Proportion	Main genre
1	3rd-Person, 3D	1996	1,094	0.0658	Action Adventure
2	1st-Person, 3D	1988	738	0.0444	Action Shooter
3	Side View, 2D Scrolling, 2D	1988	585	0.0352	Action Platformer
4	Behind View, 3D	1996	512	0.0308	Action Adventure
5	1st-Person, 2D	1988	370	0.0223	Adventure
6	Top-Down, 2D	1988	350	0.0210	Strategy
7	Bird's-Eye View, 3D	1997	282	0.0170	Strategy
8	1st-Person, 3rd-Person, 3D	1992	265	0.0159	Simulation
9	1st-Person, Behind View, 3D	1995	247	0.0149	Racing
10	1st-Person, 3rd-Person	1990	244	0.0147	Simulation
11	Side View, 2D	1988	207	0.0124	Action Platformer
12	1st-Person, 3D, Direct Control	1995	201	0.0121	Action Shooter
13	Side View, 2D Scrolling, 2D, Direct Control	1992	161	0.0097	Action Platformer
14	1st-Person, Behind View	1990	155	0.0093	Racing
15	3rd-Person, 2D	1997	149	0.0090	Adventure
16	Side View, 2D Scrolling, 3D	1988	143	0.0086	Action Platformer
17	1st-Person, Direct Control	1992	139	0.0084	Action Shooter
18	Behind View, 3D, Direct Control	1996	139	0.0084	Action Adventure
19	Bird's-Eye View, 2D	1992	130	0.0078	Role-Playing
20	Side View, Fixed / Flip-Screen	1988	120	0.0072	Puzzle

Table 3: Summary of Themes and Their Combinations

Panel A. List of the top 20 Popular Themes

	Theme keywords	Since	Obs.	Proportion	Main genre
1	Fantasy	1988	3,179	0.1465	Role-playing
2	Sci-Fi	1988	2,474	0.1140	Action Shooter
3	War	1988	1,051	0.0484	Strategy
4	Animal	1992	854	0.0393	Action Platformer
5	Magic	1988	773	0.0356	Role-playing
6	Mystery	1988	699	0.0322	Adventure
7	Military	1988	616	0.0283	Strategy
8	Space	1989	479	0.0220	Action Shooter
9	Historic	1988	476	0.0219	Strategy
10	Supernatural	1990	407	0.0187	Adventure
11	Alien	1988	401	0.0184	Action Shooter
12	Horror	1990	398	0.0183	Action Adventure
13	Cars	1988	377	0.0173	Racing
14	Urban	1988	329	0.0151	Action Adventure
15	Post-apocalyptic	1988	310	0.0142	Action Shooter
16	Comedy	1988	305	0.0140	Adventure
17	Robot	1989	294	0.0135	Action Platformer
18	Martial arts	1988	292	0.0134	Action Fighting
19	Crime	1989	287	0.0132	Adventure
20	Business	1994	283	0.0130	Strategy

Panel B. List of Top 20 Theme Combinations

	Theme combinations	Since	Obs.	Proportion	Main genre
1	Fantasy, Magic	1989	408	0.0589	Role-Playing
2	Sci-Fi, Space	1989	219	0.0316	Action Shooter
3	Military, War	1992	143	0.0206	Strategy
4	Military, Historic, War	1991	127	0.0183	Strategy
5	Sci-Fi, Robot	1989	126	0.0182	Action Shooter
6	Sci-Fi, Alien	1988	119	0.0172	Action Shooter
7	Sci-Fi, Fantasy	1995	116	0.0167	Role-Playing
8	Motorsport, Cars	1997	106	0.0153	Racing
9	Animal, Fantasy	1995	100	0.0144	Action Platformer
10	Fantasy, Creature(s) / Monster(s)	1988	87	0.0126	Role-Playing
11	Detective, Mystery	1996	64	0.0092	Adventure
12	Sci-Fi, Alien Invasion	1988	63	0.0091	Action Shooter
13	Sci-Fi, Alien, Space	1992	56	0.0081	Action Shooter
14	War, Fantasy	2002	49	0.0071	Strategy
15	Historic, War	1988	46	0.0066	Strategy
16	Fantasy, Comedy	1996	45	0.0065	Role-Playing
17	Post-Apocalyptic, Sci-Fi	1989	43	0.0062	Role-Playing
18	Mythology, Fantasy	1989	41	0.0059	Role-Playing
19	War, Sci-Fi	1998	39	0.0056	Strategy
20	Fantasy, Demon	1999	36	0.0052	Action Adventure

Table 4: Summary statistics: full sample

Variable name	Level of observation	Obs.	Mean	Std. Dev.	Min.	Max.
Dependent variables:						
(Dummy) Tech module-level innovation	Game-year	6751	0.0197	0.1390	0	1
(Dummy) Theme module-level innovation	Game-year	6751	0.1576	0.3644	0	1
(Dummy) system-level innovation	Game-year	6751	0.0058	0.0758	0	1
Independent variable:						
(Dummy) Adoption of a commercial game engine	Game-year	6751	0.1895	0.3919	0	1
Control variables:						
Licensed game	Game-year	6751	0.0698	0.2548	0	1
Console exclusive game	Game-year	6751	0.1949	0.3962	0	1
No. of platforms in which the game was introduced	Game-year	6751	1.3551	0.8107	1	8
No. of staffs who participated in game development	Game-year	6751	93.6978	107.1696	1	2854
Publisher age	Game-year	6751	2.5144	4.5166	0	29
Publisher size	Game-year	6751	4.6595	7.3082	1	48
Publisher experience	Game-year	6751	15.2579	48.6286	0	569
No. of games in the same genre	Game-year	6751	132.8729	86.4374	12	393
Genre: Action Adventure	Game-year	6751	0.0672	0.2505	0	1
Genre: Action Fight	Game-year	6751	0.0150	0.1214	0	1
Genre: Action General	Game-year	6751	0.1037	0.3049	0	1
Genre: Action Platformer	Game-year	6751	0.0816	0.2738	0	1
Genre: Action Shooter	Game-year	6751	0.1135	0.3172	0	1
Genre: Adventure	Game-year	6751	0.1235	0.3291	0	1
Genre: Puzzle	Game-year	6751	0.1029	0.3039	0	1
Genre: Role playing	Game-year	6751	0.0855	0.2796	0	1
Genre: Simulation	Game-year	6751	0.0502	0.2184	0	1
Genre: Sports	Game-year	6751	0.0425	0.2018	0	1
Genre: Strategy	Game-year	6751	0.1274	0.3334	0	1
Genre: Racing	Game-year	6751	0.0384	0.1921	0	1
Year	Game-year	6751	2011.4460	5.2761	2000	2017

Table 5: Summary statistics: subsammple by types of game engine

Variable name	Commercial engine			Inhouse engine		
	Obs.	Mean	Std. Dev.	Obs.	Mean	Std. Dev.
Dependent variables:						
(Dummy) Tech module-level innovation	1279	0.0133	0.1146	5472	0.0212	0.1441
(Dummy) Theme module-level innovation	1279	0.2854	0.4518	5472	0.1277	0.3338
(Dummy) system-level innovation	1279	0.0016	0.0395	5472	0.0068	0.0820
Control variables:						
Licensed game	1279	0.0352	0.1843	5472	0.0779	0.2680
Console exclusive game	1279	0.0680	0.2519	5472	0.2246	0.4174
No. of platforms in which the game was introduced	1279	1.7240	1.0307	5472	1.2688	0.7235
No. of staffs who participated in game development	1279	97.3984	100.3962	5472	92.8328	108.6825
Publisher age	1279	1.6536	3.7135	5472	2.7156	4.6620
Publisher size	1279	2.7404	4.5569	5472	5.1080	7.7449
Publisher experience	1279	7.1931	31.9062	5472	17.1429	51.5849
No. of games in the same genre	1279	158.3206	89.3795	5472	126.9249	84.6474
Genre: Action Adventure	1279	0.1024	0.3033	5472	0.0590	0.2357
Genre: Action Fight	1279	0.0141	0.1178	5472	0.0152	0.1222
Genre: Action General	1279	0.1079	0.3104	5472	0.1027	0.3036
Genre: Action Platformer	1279	0.1087	0.3114	5472	0.0753	0.2639
Genre: Action Shooter	1279	0.1478	0.3550	5472	0.1054	0.3072
Genre: Adventure	1279	0.1400	0.3471	5472	0.1197	0.3246
Genre: Puzzle	1279	0.0633	0.2437	5472	0.1122	0.3157
Genre: Role playing	1279	0.0797	0.2710	5472	0.0868	0.2816
Genre: Simulation	1279	0.0414	0.1994	5472	0.0523	0.2226
Genre: Sports	1279	0.0250	0.1562	5472	0.0466	0.2108
Genre: Strategy	1279	0.1016	0.3023	5472	0.1334	0.3400
Genre: Racing	1279	0.0313	0.1741	5472	0.0400	0.1960
Year	1279	2014.4600	3.1518	5472	2010.7420	5.4227

Table 6: The impact of game engine on tech module-level innovation

	DV: (Dummy) Tech module-level innovation			
	(1)	(2)	(3)	(4)
	OLS	OLS	Logit (Marginal effects)	CEM Matching
(Dummy) Adoption of a commercial game engine	0.0397**	0.0477***	0.0447***	0.0667***
	(0.0136)	(0.0139)	(0.0125)	(0.0153)
Licensed game		-0.0214	-0.0235	-0.1118***
		(0.0199)	(0.0219)	(0.0274)
Console exclusive game		0.0146	0.0158	-0.0655***
		(0.0141)	(0.0139)	(0.0190)
No. of platforms in which the game was introduced		-0.0014	-0.0008	-0.0291***
		(0.0064)	(0.0065)	(0.0080)
No. of staffs who participated in game development		0.0001*	0.0001+	0.0001
		(0.0001)	(0.0000)	(0.0001)
Publisher age		0.0011	0.0012	-0.0123**
		(0.0017)	(0.0015)	(0.0041)
Publisher size		-0.0006	-0.0005	-0.0073***
		(0.0008)	(0.0009)	(0.0013)
Publisher experience		0.0002	0.0001	0.0016
		(0.0002)	(0.0001)	(0.0012)
No. of games in the same genre		-0.0003**	-0.0003*	-0.0002+
		(0.0001)	(0.0001)	(0.0001)
Constant	0.1522***	0.2191***		0.2539***
	(0.0261)	(0.0382)		(0.0511)
Genre fixed effects	*no*	*yes*	*yes*	*yes*
Year fixed effects	*yes*	*yes*	*yes*	*yes*
R-Square	0.008	0.033	0.039	0.044
N	6751	6751	6751	4606

Note: Standard errors are in parentheses, $^{+}p < 0.1, ^{*}p < 0.05, ^{**}p < 0.01, ^{***}p < 0.001$. All specifications include year fixed effects.

Table 7: The impact of game engine on tech module-level innovation (overtime)

Panel A. The impact of all commercial game engines

	DV: (Dummy) Tech module-level innovation								
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
			CEM			CEM			CEM
	OLS	OLS	Matching	OLS	OLS	Matching	OLS	OLS	Matching
		2000-2005			2006-2011			2012-2017	
(Dummy) Adoption of a commercial game engine	-0.1073***	-0.1120***	-0.0261	0.1033*	0.0891+	0.0997+	0.0417**	0.0553***	0.0718***
	(0.0307)	(0.0337)	(0.0414)	(0.0482)	(0.0507)	(0.0582)	(0.0148)	(0.0154)	(0.0165)
Licensed game		0.0020	-0.0069		-0.0337	-0.0976+		-0.0622	-0.2233***
		(0.0286)	(0.0495)		(0.0362)	(0.0504)		(0.0452)	(0.0484)
Console exclusive game		0.0225	-0.0471		0.0443+	-0.0240		-0.0159	-0.1092***
		(0.0245)	(0.0346)		(0.0240)	(0.0337)		(0.0255)	(0.0290)
No. of platforms in which the game was introduced		-0.0182	0.0040		0.0327+	-0.0136		-0.0089	-0.0354***
		(0.0137)	(0.0263)		(0.0171)	(0.0279)		(0.0081)	(0.0088)
No. of staffs who participated in game development		0.0003	-0.0002		-0.0000	-0.0000		0.0002*	0.0001
		(0.0002)	(0.0003)		(0.0001)	(0.0002)		(0.0001)	(0.0001)
Publisher age		-0.0013	-0.0136*		0.0011	-0.0114		0.0018	-0.0141*
		(0.0033)	(0.0061)		(0.0030)	(0.0086)		(0.0025)	(0.0061)
Publisher size		-0.0005	0.0001		-0.0014	0.0198*		-0.0011	-0.0104***
		(0.0019)	(0.0033)		(0.0014)	(0.0095)		(0.0013)	(0.0016)
Publisher experience		0.0008	-0.0023		0.0002	-0.0013		0.0000	0.0033
		(0.0005)	(0.0018)		(0.0003)	(0.0011)		(0.0003)	(0.0026)
No. of games in the same genre		-0.0007	-0.0002		-0.0002	-0.0003		-0.0002	-0.0001
		(0.0009)	(0.0011)		(0.0005)	(0.0006)		(0.0002)	(0.0002)
Constant	0.1538***	0.2894***	0.0888	0.1288***	0.0382	0.0887	0.2090***	0.3526***	0.3876***
	(0.0262)	(0.0714)	(0.0880)	(0.0232)	(0.0566)	(0.0801)	(0.0263)	(0.0461)	(0.0507)
Genre fixed effects	*no*	*yes*	*yes*	*no*	*yes*	*yes*	*no*	*yes*	*yes*
Year fixed effects	*yes*	*yes*	*yes*	*yes*	*yes*	*yes*	*yes*	*yes*	*yes*
R-Squared	0.007	0.025	0.010	0.010	0.028	0.037	0.004	0.040	0.051
N	1215	1215	579	1598	1598	916	3938	3938	3111

Panel B. The impact of Unreal/Unity and other commercial engines

	DV: (Dummy) Tech module-level innovation								
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
			CEM			CEM			CEM
	OLS	OLS	Matching	OLS	OLS	Matching	OLS	OLS	Matching
		2000-2005			2006-2011			2012-2017	
(Dummy) Adoption of Unreal/Unity game engine	-0.1482***	-0.1288***	-0.0783*	0.1690*	0.1558*	0.1699+	0.0419**	0.0546***	0.0719***
	(0.0142)	(0.0301)	(0.0377)	(0.0696)	(0.0719)	(0.0904)	(0.0153)	(0.0159)	(0.0170)
(Dummy) Adoption of other commercial game engines	-0.0932*	-0.1063*	-0.0070	0.0321	0.0174	0.0397	0.0400	0.0618	0.0713+
	(0.0397)	(0.0425)	(0.0523)	(0.0631)	(0.0647)	(0.0679)	(0.0420)	(0.0397)	(0.0429)
Controls	*no*	*yes*	*yes*	*no*	*yes*	*yes*	*no*	*yes*	*yes*
Genre fixed effects	*no*	*yes*	*yes*	*no*	*yes*	*yes*	*no*	*yes*	*yes*
Year fixed effects	*yes*	*yes*	*yes*	*yes*	*yes*	*yes*	*yes*	*yes*	*yes*
R-Squared	0.007	0.024	0.008	0.011	0.030	0.038	0.004	0.040	0.050
N	1215	1215	579	1598	1598	916	3938	3938	3111

Note: Standard errors are in parentheses, $^{+}p < 0.1, ^{*}p < 0.05, ^{**}p < 0.01, ^{***}p < 0.001$. All specifications include year fixed effects.

Table 8: The impact of game engine on theme module-level innovation

	DV: (Dummy) Theme module-level innovation			
	(1)	(2)	(3)	(4)
	OLS	OLS	Logit (Marginal effects)	CEM Matching
(Dummy) Adoption of a commercial game engine	0.1702***	0.1450***	0.1182***	0.1531***
	(0.0145)	(0.0149)	(0.0111)	(0.0164)
Licensed game		-0.0430*	-0.0481*	-0.0507
		(0.0185)	(0.0235)	(0.0406)
Console exclusive game		0.0230+	0.0250+	-0.0639***
		(0.0134)	(0.0136)	(0.0162)
No. of platforms in which the game was introduced		0.0585***	0.0484***	0.0638***
		(0.0074)	(0.0054)	(0.0107)
No. of staffs who participated in game development		0.0002**	0.0001**	-0.0001
		(0.0001)	(0.0000)	(0.0001)
Publisher age		0.0044**	0.0041**	0.0054
		(0.0016)	(0.0013)	(0.0049)
Publisher size		0.0007	0.0009	-0.0088***
		(0.0008)	(0.0008)	(0.0012)
Publisher experience		-0.0003+	-0.0003*	0.0025**
		(0.0002)	(0.0001)	(0.0009)
No. of games in the same genre		-0.0003*	-0.0002*	-0.0002
		(0.0001)	(0.0001)	(0.0001)
Constant	0.0929***	0.0441		0.1211*
	(0.0211)	(0.0328)		(0.0489)
Genre fixed effects	*no*	*yes*	*yes*	*yes*
Year fixed effects	*yes*	*yes*	*yes*	*yes*
R-Squared	0.050	0.081	0.087	0.086
N	6751	6751	6751	4583

Note: Standard errors are in parentheses, $^{+}p < 0.1, ^{*}p < 0.05, ^{**}p < 0.01, ^{***}p < 0.001$. All specifications include year fixed effects.

Table 9: The impact of game engine on theme module-level innovation

Panel A. The impact of all commercial game engines

	DV: (Dummy) Theme module-level innovation								
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
	OLS	OLS	CEM Matching	OLS	OLS	CEM Matching	OLS	OLS	CEM Matching
		2000-2005			2006-2011			2012-2017	
(Dummy) Adoption of a commercial game engine	0.1349*	0.0836	0.0907	0.1247*	0.0801+	0.0901+	0.1771***	0.1493***	0.1600***
	(0.0608)	(0.0623)	(0.0771)	(0.0496)	(0.0480)	(0.0545)	(0.0156)	(0.0165)	(0.0177)
Licensed game		-0.0861***	-0.1187***		-0.0103	0.0255		0.0341	-0.0302
		(0.0191)	(0.0333)		(0.0355)	(0.0938)		(0.0546)	(0.0927)
Console exclusive game		-0.0030	-0.0178		0.0154	-0.0438		0.0535*	-0.1039***
		(0.0212)	(0.0343)		(0.0221)	(0.0292)		(0.0265)	(0.0234)
No. of platforms in which the game was introduced		0.0303+	0.0518		0.0575**	0.0595+		0.0639***	0.0630***
		(0.0155)	(0.0365)		(0.0176)	(0.0310)		(0.0096)	(0.0120)
No. of staffs who participated in game development		0.0005*	0.0003		0.0003***	-0.0001		0.0001	-0.0002
		(0.0002)	(0.0003)		(0.0001)	(0.0002)		(0.0001)	(0.0001)
Publisher age		0.0025	0.0017		0.0059*	0.0192*		0.0046+	0.0077
		(0.0030)	(0.0093)		(0.0028)	(0.0096)		(0.0024)	(0.0079)
Publisher size		0.0011	-0.0075+		0.0007	-0.0163+		0.0003	-0.0072***
		(0.0018)	(0.0045)		(0.0016)	(0.0093)		(0.0012)	(0.0012)
Publisher experience		-0.0001	0.0036		-0.0003	0.0033***		-0.0005+	-0.0025
		(0.0005)	(0.0039)		(0.0003)	(0.0009)		(0.0003)	(0.0021)
No. of games in the same genre		0.0002	-0.0004		-0.0002	-0.0008		0.0001	0.0003
		(0.0007)	(0.0008)		(0.0004)	(0.0005)		(0.0002)	(0.0002)
Constant	0.0933***	0.0174	0.0624	0.0774***	0.0048	0.0747	0.2014***	0.1747***	0.2121***
	(0.0212)	(0.0475)	(0.1057)	(0.0184)	(0.0568)	(0.0724)	(0.0266)	(0.0446)	(0.0500)
Genre fixed effects	*no*	*yes*	*yes*	*no*	*yes*	*yes*	*no*	*yes*	*yes*
Year fixed effects	*yes*	*yes*	*yes*	*yes*	*yes*	*yes*	*yes*	*yes*	*yes*
R-Squared	0.005	0.059	0.052	0.032	0.077	0.070	0.050	0.077	0.087
N	1215	1215	574	1598	1598	892	3938	3938	3117

Panel B. The impact of Unreal/Unity and other commercial engines

	DV: (Dummy) Theme module-level innovation								
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
	OLS	OLS	CEM Matching	OLS	OLS	CEM Matching	OLS	OLS	CEM Matching
		2000-2005			2006-2011			2012-2017	
(Dummy) Adoption of Unreal/Unity game engine	0.3613*	0.3028*	0.3516*	0.1324+	0.0770	0.0329	0.1794***	0.1495***	0.1604***
	(0.1403)	(0.1374)	(0.1580)	(0.0683)	(0.0634)	(0.0773)	(0.0163)	(0.0172)	(0.0185)
(Dummy) Adoption of other commercial game engines	0.0567	0.0089	-0.0143	0.1164+	0.0835	0.1371+	0.1556***	0.1477***	0.1565***
	(0.0602)	(0.0630)	(0.0685)	(0.0701)	(0.0697)	(0.0736)	(0.0454)	(0.0447)	(0.0474)
Controls	*no*	*yes*	*yes*	*no*	*yes*	*yes*	*no*	*yes*	*yes*
Genre fixed effects	*no*	*yes*	*yes*	*no*	*yes*	*yes*	*no*	*yes*	*yes*
Year fixed effects	*yes*	*yes*	*yes*	*yes*	*yes*	*yes*	*yes*	*yes*	*yes*
R-Squared	0.012	0.066	0.070	0.031	0.077	0.071	0.049	0.077	0.087
N	1215	1215	574	1598	1598	892	3938	3938	3117

Note: Standard errors are in parentheses, $^{+}p < 0.1, ^{*}p < 0.05, ^{**}p < 0.01, ^{***}p < 0.001$. All specifications include year fixed effects.

Table 10: The impact of game engine on system-level innovation

	DV: (Dummy) system-level innovation			
	(1)	(2)	(3)	(4)
	OLS	OLS	Logit (Marginal effects)	CEM Matching
(Dummy) Adoption of a commercial game engine	-0.0029*	-0.0044*	-0.0093	-0.0045*
	(0.0014)	(0.0019)	(0.0065)	(0.0021)
Licensed game		-0.0065	-0.0075	-0.0063
		(0.0041)	(0.0076)	(0.0043)
Console exclusive game		0.0013	0.0022	0.0010
		(0.0034)	(0.0036)	(0.0038)
No. of platforms in which the game was introduced		0.0045*	0.0046***	0.0052*
		(0.0019)	(0.0014)	(0.0021)
No. of staffs who participated in game development		-0.0000	-0.0000	0.0000
		(0.0000)	(0.0000)	(0.0000)
Publisher age		-0.0000	0.0003	-0.0001
		(0.0004)	(0.0004)	(0.0004)
Publisher size		0.0003	0.0003	0.0004
		(0.0003)	(0.0002)	(0.0003)
Publisher experience		0.0000	-0.0000	0.0000
		(0.0001)	(0.0000)	(0.0001)
No. of games in the same genre		0.0000	-0.0001	0.0000
		(0.0000)	(0.0001)	(0.0000)
Constant	0.0106	0.0081		0.0075
	(0.0074)	(0.0099)		(0.0106)
Genre fixed effects	*no*	*yes*	*yes*	*yes*
Year fixed effects	*yes*	*yes*	*yes*	*yes*
R-Squared	0.005	0.010	0.130	0.011
N	6751	6751	4428	4858

Note: Standard errors are in parentheses, $^{+}p < 0.1,^{*} p < 0.05,^{**} p < 0.01,^{***} p < 0.001$. All specifications include year fixed effects.

Table 11: The impact of game engine on system-level innovation

Panel A. The impact of all commercial game engines

	DV: (Dummy) system-level innovation								
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
			CEM			CEM			CEM
	OLS	OLS	Matching	OLS	OLS	Matching	OLS	OLS	Matching
		2000-2005			2006-2011			2012-2017	
(Dummy) Adoption of a commercial game engine	-0.0058*	-0.0097+	-0.0092	-0.0115***	-0.0099*	-0.0144*	-0.0019	-0.0029	-0.0030
	(0.0025)	(0.0058)	(0.0059)	(0.0029)	(0.0043)	(0.0058)	(0.0016)	(0.0021)	(0.0022)
Licensed game		-0.0098*	-0.0067*		0.0017	-0.0002		-0.0077**	-0.0081*
		(0.0047)	(0.0034)		(0.0113)	(0.0128)		(0.0030)	(0.0034)
Console exclusive game		0.0021	0.0043		0.0000	-0.0020		0.0053	0.0069
		(0.0050)	(0.0054)		(0.0067)	(0.0073)		(0.0058)	(0.0065)
No. of platforms in which the game was introduced		0.0105	0.0113		0.0050	0.0127		0.0024	0.0027
		(0.0073)	(0.0080)		(0.0054)	(0.0089)		(0.0018)	(0.0020)
No. of staffs who participated in game development		0.0000	0.0000		-0.0000	0.0000		-0.0000	0.0000
		(0.0001)	(0.0001)		(0.0000)	(0.0000)		(0.0000)	(0.0000)
Publisher age		-0.0006	-0.0005		0.0006	0.0002		0.0001	0.0002
		(0.0005)	(0.0005)		(0.0008)	(0.0009)		(0.0005)	(0.0005)
Publisher size		0.0004	0.0001		0.0004	0.0008		0.0001	0.0002
		(0.0005)	(0.0004)		(0.0007)	(0.0007)		(0.0002)	(0.0002)
Publisher experience		0.0000	0.0001		0.0000	0.0000		-0.0000	-0.0001+
		(0.0001)	(0.0001)		(0.0001)	(0.0002)		(0.0000)	(0.0000)
No. of games in the same genre		-0.0001	-0.0001		-0.0001	-0.0001		0.0001	0.0001
		(0.0002)	(0.0002)		(0.0001)	(0.0001)		(0.0000)	(0.0000)
Constant	0.0106	-0.0043	-0.0066	0.0005**	0.0101	0.0009	0.0201*	0.0190*	0.0192+
	(0.0074)	(0.0121)	(0.0128)	(0.0002)	(0.0194)	(0.0209)	(0.0089)	(0.0096)	(0.0100)
Genre fixed effects	*no*	*yes*	*yes*	*no*	*yes*	*yes*	*no*	*yes*	*yes*
Year fixed effects	*yes*	*yes*	*yes*	*yes*	*yes*	*yes*	*yes*	*yes*	*yes*
R-Squared	0.002	0.001	0.005	0.001	0.004	0.008	0.007	0.016	0.018
N	1215	1215	646	1598	1598	971	3938	3938	3241

Panel B. The impact of Unreal/Unity and other commercial engines

	DV: (Dummy) system-level innovation								
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
			CEM			CEM			CEM
	OLS	OLS	Matching	OLS	OLS	Matching	OLS	OLS	Matching
		2000-2005			2006-2011			2012-2017	
(Dummy) Adoption of Unreal/Unity game engine	-0.0061*	-0.0087	-0.0063	-0.0121***	-0.0107*	-0.0138*	-0.0015	-0.0026	-0.0026
	(0.0027)	(0.0067)	(0.0055)	(0.0032)	(0.0047)	(0.0058)	(0.0017)	(0.0022)	(0.0023)
(Dummy) Adoption of other commercial game engine	-0.0057*	-0.0101	-0.0102	-0.0110***	-0.0091+	-0.0150*	-0.0053**	-0.0059**	-0.0063**
	(0.0027)	(0.0062)	(0.0067)	(0.0028)	(0.0049)	(0.0068)	(0.0016)	(0.0021)	(0.0023)
Controls	*no*	*yes*	*yes*	*no*	*yes*	*yes*	*no*	*yes*	*yes*
Genre fixed effects	*no*	*yes*	*yes*	*no*	*yes*	*yes*	*no*	*yes*	*yes*
Year fixed effects	*yes*	*yes*	*yes*	*yes*	*yes*	*yes*	*yes*	*yes*	*yes*
R-Squared	0.003	0.001	0.006	0.001	0.004	0.007	0.007	0.015	0.018
N	1215	1215	646	1598	1598	971	3938	3938	3241

Note: Standard errors are in parentheses, $^{+}p < 0.1, ^{*}p < 0.05, ^{**}p < 0.01, ^{***}p < 0.001$. All specifications include year fixed effects.

Appendix 1: Correlation table

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	(Dummy) Tech module-level innovation	1																							
2	(Dummy) Theme module-level innovation	0.0528	1																						
3	(Dummy) system-level innovation	0.538	0.176	1																					
4	(Dummy) Adoption of a commercial game engine	-0.0223	0.170	-0.0269	1																				
5	Licensed game	0.0114	-0.0371	-0.00553	-0.0656	1																			
6	Console exclusive game	0.0675	-0.0292	0.0316	-0.155	0.178	1																		
7	No. of platforms in which the game was introduced	0.0326	0.138	0.0365	0.220	0.106	-0.0716	1																	
8	No. of staffs who participated in game development	-0.00432	0.0844	0.00461	0.0167	0.0583	0.0129	0.170	1																
9	Publisher age	0.0518	0.0399	0.0424	-0.0921	0.188	0.302	0.109	0.198	1															
10	Publisher size	0.0469	-0.00349	0.0536	-0.127	0.147	0.254	0.0210	0.103	0.501	1														
11	Publisher expeirence	0.0485	0.0349	0.0496	-0.0802	0.121	0.261	0.107	0.310	0.725	0.586	1													
12	No. of games in the same genre	-0.0737	0.0175	-0.0457	0.142	-0.168	-0.239	-0.0335	0.0752	-0.184	-0.107	-0.139	1												
13	Genre: Action Adventure	0.0215	0.0721	0.00294	0.0679	0.0100	-0.0246	0.0896	0.0460	0.0122	-0.0121	-0.00159	-0.0940	1											
14	Genre: Action Fight	0.0264	-0.00307	0.0389	-0.00353	0.0573	0.112	0.0198	0.0114	0.0257	0.0293	0.0151	-0.154	-0.0331	1										
15	Genre: Action General	-0.00276	0.00223	-0.00669	0.00667	0.00603	0.0338	-0.00272	-0.0122	-0.0422	-0.0467	-0.0189	0.102	-0.0913	-0.0419	1									
16	Genre: Action Platformer	-0.0228	-0.0235	-0.00845	0.0478	-0.0179	-0.0128	0.104	-0.0183	-0.0468	-0.0290	-0.0284	-0.0530	-0.0800	-0.0367	-0.101	1								
17	Genre: Action Shooter	-0.00703	0.00420	-0.0149	0.0523	-0.0668	-0.00863	0.0213	0.0291	-0.0402	-0.0155	-0.0210	0.0375	-0.0961	-0.0441	-0.122	-0.107	1							
18	Genre: Adventure	-0.0273	0.0402	-0.0227	0.0241	-0.0321	-0.137	-0.0650	-0.0269	-0.0713	-0.0249	-0.0591	0.396	-0.101	-0.0463	-0.128	-0.112	-0.134	1						
19	Genre: Puzzle	-0.00944	-0.0596	-0.0000962	-0.0630	-0.0564	0.0105	-0.0263	-0.0590	-0.00794	0.0403	-0.00238	0.0343	-0.0910	-0.0417	-0.115	-0.101	-0.121	-0.127	1					
20	Genre: Role playing	0.0215	-0.0130	-0.00233	-0.00989	-0.0317	-0.0314	-0.0332	0.0583	0.0125	-0.0173	-0.00405	-0.0587	-0.0821	-0.0377	-0.104	-0.0911	-0.109	-0.115	-0.104	1				
21	Genre: Simulation	0.0113	0.0290	0.0362	-0.0194	-0.0310	-0.00870	-0.0463	-0.00106	0.0132	0.0143	0.0151	-0.156	-0.0617	-0.0283	-0.0782	-0.0685	-0.0823	-0.0863	-0.0779	-0.0703	1			
22	Genre: Sports	-0.0140	-0.0468	-0.0161	-0.0419	0.202	0.154	0.0680	0.0144	0.104	0.0821	0.0915	-0.139	-0.0566	-0.0260	-0.0717	-0.0628	-0.0754	-0.0791	-0.0714	-0.0644	-0.0484	1		
23	Genre: Strategy	-0.00301	0.00422	0.00605	-0.0373	-0.0366	-0.0815	-0.0868	-0.0317	0.0244	-0.0312	-0.0192	0.0684	-0.103	-0.0471	-0.130	-0.114	-0.137	-0.143	-0.129	-0.117	-0.0879	-0.0805	1	
24	Genre: Racing	-0.000569	-0.0377	0.00513	-0.0178	0.130	0.0944	0.0381	0.00766	0.0440	0.0537	0.0458	-0.179	-0.0536	-0.0246	-0.0679	-0.0595	-0.0715	-0.0750	-0.0677	-0.0611	-0.0459	-0.0421	-0.0763	1