A Taxonomy of Game Engines and the Tools that Drive the Industry

Marcus Toftedahl, Henrik Engström
Division of Game Development
University of Skövde
Skövde, Sweden
marcus.toftedahl@his.se, henrik.engstrom@his.se

ABSTRACT
Game engines are a vital part of a game production pipeline, but there is a vagueness of definitions regarding the boundaries of components in a game engine and the rest of the production tools used in a game development pipeline. The aim of this paper is to nuance the use of the term game engine and to put it into the context of a game development pipeline. Based on data from the current state of game production, a proposed taxonomy for tools in game development is presented. A distinction is made between user facing tools and product facing tools. A defining characteristic of the production pipeline and game engines is their plasticity. One of the conclusions is that a "game engine" as a single entity containing the whole game production pipeline is not desirable due to the large number of competences and needs involved in a game development project.

Keywords
Game development, production pipeline, game engine, development tools, taxonomy, production studies

INTRODUCTION
The origin of this study was an attempt to survey what kind of technical localization support there is in commonly used game engines. This idea was based on our previous research, where it was identified that indie developers tend to realize that localization is necessary rather late in the development process (Toftedahl, Backlund, & Engström, 2018). Our initial hypothesis for this paper was that developers with limited experience of releasing games on a global market were not exposed to localization tools through a game engine in the same way as other parts of game productions. Many of the popular game engines have built-in tools for graphics, animation, networked play, AI etc., and our initial approach was to scrutinize which game engines have built-in tools for localization. During that process, it was soon evident that this initial approach was not particularly interesting to investigate mainly due to the complex area of game production and its related tools. The complexity regarding definitions and interconnections of game engines and development tools made research on a specific function within a game engine difficult. Instead, we chose to alter the focus of this study and propose a taxonomy of game engines and the tools that drives game production.

Game engines are software required for the development of modern games (Anderson, Engel, Comninos, & McLoughlin, 2008). But what is a game engine, who use them – and is it even possible to talk about a "game engine" as a single coherent entity? This paper aims to investigate the current state of the game engine concept in relation to game development and production; who in a game development project is likely to use a game engine, which game engines are common in the game industry of today (i.e.

Proceedings of DiGRA 2019

2019) and what features and functions does a game engine have? The intention with this research is to deepen the understanding of the complexities of game development and its production pipeline from the perspective of the diverse set of competences involved in a game production.

Producing a game includes a multitude of competences. Programmers, graphic artists, sound designers, narrative designers and game designers are all competences with various production based needs and work related tasks in a game development setting (Hagen, 2009). Since games are not only an entertainment product but also a complex technical system with the aim to give the end user a satisfactory, often entertaining, experience, game development is a complex task where system engineering and creative competences in art and design must be handled in the same project infrastructure.

The work processes in game development projects have often been studied from a software engineering perspective, approaching games as a different type of software product; Wang and Nordmark (2015) have through a survey, with game developers as respondents, researched software architecture and creative processes in game development focusing on the software architecture used in a game development project. Their results indicate that the software architecture is important and has impact on the manageability of the complex development situation, where many competences are involved. Further, the findings from this study indicate that the game concept has an impact on the software architecture and subsequently the choice of game engine. A frequent use of “middleware” and other external third-party components was also reported. They state that the technical aspects of game production have become easier during the last 5 years (the study was conducted in 2012) but due to player demands *“...game development itself has not been become easier due to higher player expectations and higher game complexity”* (Wang & Nordmark, 2015, p. 283). Murphy-Hill, Zimmermann, and Nagappan (2014) present a study focusing on the differences between game development and software development, using surveys and interviews with game developers from the triple-A industry (i.e. large projects with big budgets). One of their findings is that game developers are reliant on in-house tools, an area that is underrepresented in research. To systematically investigate these tool sets is identified as an area of further research.

The goal of this paper is to highlight the role of a game engine in a game production and the relation between a game engine and the other tools used in a game production pipeline. To achieve this, a number of steps have been taken to better understand the complex situations and connections found in game development projects. In chronological order as presented in the paper, the steps are: (1) identifying the roles in a game development project to get an understanding of who is likely to use a game engine; (2) a study of which game engines are commonly used in the industry; and (3) an overview of the definition(s) of game engines derived from previous research and game industry sources. The analysis reveals that there is a great deal of misconceptions regarding the tools used in game production and what role a game engine plays. We address these misconceptions by proposing a taxonomy for tools used in game production, including how game engines and other development tools are related to the production pipeline. Our main message is that the most important entity is the production pipeline and that this pipeline, as well as the game engine it contains, is characterized by a very high degree of plasticity.

ROLES IN A GAME DEVELOPMENT PROJECT

As previous research have stated (Hagen, 2009; O'Donnell, 2009; Tschang & Szczypula, 2006; Zackariasson, Styhre, & Wilson, 2006) there is a multitude of competences and roles involved in producing a game. Many of them are involved directly in the production, thus using, or having to relate to the use of, a game engine to make the game. Since the advent of digital distribution, even small scale productions have the possibility to enter the market and reach large groups of consumers. The traditional value chain of the game industry; including developers, publishers, distributors and retailers commonly used in the triple-A big budget industry (Egenfeldt-Nielsen, Smith, & Tosca, 2016; O'Hagan & Mangiron, 2013) is today not the only way of releasing a game. This has paved the way for the “indie” game sector, where smaller teams with smaller budgets can survive (and in some cases thrive) by selling games (Payne & Steirer, 2014; Pereira & Bernardes, 2018). The production scope of triple-A and indie differs though. Regardless of size, both production settings use some kind of game engine.

One often cited source of information regarding the complexities of game development is *Game Development: Harder than you think*, written by Jonathan Blow in 2004. The article focuses on the complexities regarding game development and how game development projects gradually have gotten even more complex over time. As an example, a 3D game circa 1996 contains only a few modules for the base functions (sound, 3D rendering, collision detection etc.) with low level of interconnections, while a 3D game in 2004 contains four times as many modules with interconnections difficult to track (Blow, 2004). (Whitson, 2018) describes how different development tools are used by different developer roles in a project. While the main purpose of such tools are to produce content, the tools used also acts as boundary objects helping the development team to communicate scope and create a common vision for the project.

As previously stated: game development is a complex project to undertake. To get an understanding of the complexities we have chosen to use the credits lists from two games with very different scope and from two traditions of game development: triple-A and indie. The games chosen as representatives from each category are based on both the production tradition, but also on top lists. The triple-A game *Assassin’s Creed: Odyssey* (Ubisoft, 2018) is credited as one of the games in the top selling list of 2018[1]. The indie game chosen, *Undertale* (tobyfox, 2015) has a similar track record on the charts, dubbed as an “indie sensation” and on the top selling lists of 2016[2]. Comparing the projects shows that the number of people involved in creating these products varies massively; from several thousand people to approximately two (one main developer with additional art support from one artist).

Roles in triple-A production

Using statistics from *Assassin’s Creed: Odyssey*, based on its credit sequence shows the complex nature of triple-A game development in sheer numbers. The credit sequence consists of rolling text and is approximately 30 minutes long from start to finish. During that time period, 4388 persons have rolled by where 3355 are developers in one of 692 development roles (Table 1). The development roles in the project are divided on 29 different development studios all over the world.

Assassin's Creed: Odyssey development statistics	
Studios	**Amount**
Development studios	29
Administrative studios	26
Total studios	55
Teams	**Amount**
Development teams	72
Administrative teams	125
Total teams	197
Roles	**Amount**
Development roles	692
Administrative roles	385
Total roles	1077
Staff	**Amount**
Developers	3355
Administrators	1033
Total staff	4388

Table 1: Statistics from the credit sequence of *Assassin's Creed: Odyssey*

The credit list is divided into the respective organizations involved, and we have presented it as such. Studio is the over-arching entity (examples include Ubisoft Québec, Ubisoft Chengdu etc.) which have a number of sub-entities, i.e. teams with more specific roles within each studio (i.e. Development Team Ubisoft Québec: Design, Development Team Ubisoft Chengdu: Narrative etc.). We have also chosen to separate development from administration in the statistics since this is done to a large extent in the credits sequence as well.

Of the 3355 developers credited we found 692 different roles within the development teams. To get an overview of how the roles are weighted in relation to each other, i.e. how many different programmers are there, we used MAXQDA to create a word cloud of roles. We multiplied the occurrence of each role by the amount of people associated with it (i.e. if 3 persons was credited as 3D Artist the data would look like 3D Artist, 3D Artist, 3D Artist) and created the word cloud (Figure 1).

Figure 1: Word cloud regarding the roles credited in *Assassin's Creed: Odyssey*

Tester and programmer were the two largest groups from the data set. There were also a large number of people involved in translation and localization. In Table 2, the 10 most frequent words from the role word cloud are presented.

Word	Frequency	%
tester	415	9.1
programmer	402	8.8
manager	208	4.6
designer	154	3.4
technical	127	2.8
artist	111	2.4
gameplay	106	2.3
level	96	2.1
localization	77	1.7
translation	69	1.5

Table 2: Word count from the *Assassin's Creed: Odyssey* credits data, derived from the word cloud created in MAXQDA.

All words are related to roles in the development. While some seems out of place (i.e. technical, level or gameplay) they have a relation to a specific role like technical programmer, gameplay animator or level designer.

Roles in indie production

To make a similar overview of a popular small scale indie game is easier. While *Undertale* could be seen as an “easy pick” and as an unfair comparison to *Assassins Creed*, it is worth to reiterate that both are examples of top selling games available on multiple platforms. The game *Undertale* is produced in GameMaker and has one developer credited as main developer (“Undertale by: Toby Fox”) and another developer for various tasks, such as cutscenes and logo design. It is interesting to note that the community connection is very apparent in this indie production, with a long list of “Special Thanks” containing 880 names. While the sheer scope and production values vary greatly between *Undertale* and *Assassin’s Creed: Odyssey*, it is also a sign of the range of production resources when producing a popular game.

COMMONLY USED GAME ENGINES – AN OVERVIEW FROM ITCH.IO AND STEAM

To understand which game engines are used to produce and release games we have taken an approach using freely accessible data from two digital game stores: Steam and Itch.io. The choice of digital game stores was based on two main factors: (1) data from the report “The State of the Industry” released in connection to Game Developers Conference 2019 (UBM, 2018) stating that Steam (47%) and Itch.io (18%) are the two most commonly used digital stores for games on PC, and subsequently (2) the difference of content between the two stores. Steam represents a spectra consisting of both big- and small scale production, whereas Itch.io mainly targets indie and hobbyist developers with projects of smaller scope.

In the case of Itch.io, information regarding game engines used in the released games is available on an official statistics page[3]. The data is self-reported by the developers and therefore some games may lack this information and/or information can be faulty, making the data somewhat unreliable.

Steam, the largest digital distribution channel for digital games on the PC platform (UBM, 2018), does not provide information directly like Itch.io does. Regardless of the lack of open data, Steam is still a widely used source of information to understand the state of the game industry of today and several analyst firms and data aggregator sites use available data (i.e. *SteamDB*[4] and *SteamSpy*[5]) to give an overview of the current situation. The statistics available are mostly related to the consumer and user side of the store, with no or little information related to the development aspects.

By using the script *Steam Engines* developed by Github user *limdingwen*[6] information regarding the use of game engines used in games released on Steam has been compiled. The finished list contains 49 281 game titles including expansions and downloadable contents (DLCs). Approximately 15% of the gathered titles have information regarding game engine.

The process to identify engines was conducted according to the following steps:

- A list of the names of all products currently available on Steam is gathered in a comma separated value (CSV) file from the Steam store.
- The script retrieves information about the products from Wikipedia:
    - If there is a Wikipedia page, the script checks if there is information in the “game engine” field on the page. If there is information regarding game engine, this data is stored in the CSV-file in relation to the name of the game.
    - If no “game engine” field exists, the script tags the game with “unknown” in the CSV-file. The same is done if no Wikipedia page exists.
- After the compilation of data from Wikipedia, the CSV-file is manually cleaned from all products not identified as games. Since Steam hosts applications and video, and audio files as well, these are removed using a script in Excel identifying and tagging them for removal. The tagged applications and videos are then removed manually to minimize the risk of removing game titles.

This approach might seem overly complicated related to the results it yields, but reliable information regarding the use of game engines is difficult to get. There is a presumably large margin of error in the data from Steam as the version of the script used does not double check the information with other sources. Upon manually inspecting the results, games with a large number of expansions or downloadable contents can skew the results in favor to a specific engine. We have chosen not to try to separate DLCs from the main games due to the large number of DLC packages.

Unity and Unreal have been in the spotlight in the development community and their market penetration is large. The data derived from Itch.io (Table 3) regarding the most used game engines on that specific game store does support this claim: Unity is the engine used in approximately 47 % of the games published on Itch.io.

Game Engine	Number of projects	% of total games
Unity	24200	47.3 %
Construct	6275	12.3 %
GameMaker	5643	11.0 %
Twine	3184	6.2 %
RPG Maker	1982	3.9 %
Bitsy	1683	3.3 %
PICO-8	1479	2.9 %
Unreal	1458	2.8 %
Godot	1274	2.5 %
Ren'Py	1008	2.0 %
Games with other engines	2993	5.9 %
Total games	**51179**	

Table 3: Game engines used in games released on Itch.io (data collected 2018-12-28)

Other notable game engines used on the Itch.io platform are Construct (12 %) (Scirra, 2018), GameMaker (11 %) (YoYo Games, 2018) and Twine (6 %) (Klimas, 2009) – game engines that mostly are associated with hobbyist or “indie” production. Due to the nature of the Itch.io service, where hobbyists and indie developers can release games with relatively little effort, the indie focus is not surprising. It is interesting to see that Unreal is only used in approximately 3 % of the published games on Itch.io. This share is on par with niche game engines such as PICO-8[7] and RPG Maker[8], game engines that have a very specific target audience and use. PICO-8 is a game engine and game production platform that is based on specifications of a “fantasy console” with harsh limitations to its technical specifications, while RPG Maker is targeted do produce 2D games within the RPG genre.

The statistics from the use of game engines on Steam is presented in Table 4. The number of total identified games including DLC and expansions is consistent with industry reports regarding the matter. In an article on the gaming site PC Gamer (Bolding, 2019) it is described that Steam as of January 2019 have approximately. 30,000 games and 21,000 DLCs, meaning that the data set at least have the total number of game titles fairly correct. Since only approximately 13 % of the games in total have been identified and tagged with a game engine, there is a large margin of error in the data. The number of DLCs related to some titles is also apparent with an overrepresentation of some game engines known to be used only in specific games. Anvil, for example, only used in Ubisoft games such as the *Assassin’s Creed* series, have 166 entries in the data set. The high number of Anvil titles is due to the DLCs related to these games.

Game Engine	Number of projects	% of total games identified
Unreal	1726	25.6 %
Unity	889	13.2 %
Source	270	4.0 %
Cryengine	238	3.5 %
Gamebryo	215	3.2 %
IW	192	2.9 %
Anvil	166	2.5 %
id Tech	113	1.7 %
Essence	73	1.1 %
Clausewitz	68	1.0 %
Identified games with other engines	3266	48.4 %
Total games identified	6743	
Unknown/unidentified games	42538	
Total games in Steam database (including DLC/expansions)	49281	

Table 4: Game engines used in games released on Steam (data from 2018-12-20).

The overlap between the top 10 game engines on Itch.io and the Steam game engine list is small; only Unity and Unreal are present in both lists. None of the other engines from the Steam data set is present in the Itch.io data at all.

GAME ENGINES AND PRODUCTION TOOLS

Previous research on game engines includes perspectives from software engineering and software architecture; Messaoudi, Simon, and Ksentini (2015) investigate the game engine Unity and from a technical standpoint measure its performance using a number of tests straining the CPU and GPU. The authors also discuss a difference between a game engine as a production tool and a game engine as the run-time executable that "drives" the game. They propose to use the term game engine to describe the executable, and the toolset that is used to build the games should be referred to as the "framework" (Messaoudi et al., 2015). Anderson et al. (2008) are also mentioning the misunderstanding in terminology and points to the fact that there is a lack of a "game development language". One of the examples regarding terminology is the "game engine" term. Anderson et al. (2008) reflect on the boundaries of a game engine: (1) the game itself is confused with the game engine and (2) if the toolset is a part of the game engine. Other questions raised includes if it is *"...possible to define a game engine independently of genre"*, i.e. having a game engine that can be so flexible to accommodate for games in all genres – an "überengine" (Anderson et al., 2008, p. 229). O'Donnell (2014) also reflects upon the boundaries between games and engine, with a description of a game engine as *"[t]he underlying software of the game. The engine is not a game; it is more basic than that. It provides a platform, to which more code, data and art assets are produced to make a game"*.

From a software architecture perspective, a game engine is a complex system of intertwined layers relating to hardware and other software. This complexity is also reflected in the production process, as identified by Gregory (2014) who identifies a large number of components integral to a game, handled by a game engine during development. Components include input devices, graphics rendering functions, collision detection systems, physics simulations and animation systems – to name a few. To list all components in a game engine would be a daunting task, and attempts have been made to understand the technicalities and components related to game engines in a more general way. Anderson (2011) presents a conceptual model of a typical game engine (Figure 2) consisting of an Engine Core interfacing with; (1) Application specific code, (2) Engine Modules (input, rendering functions etc.), (3) a Resource Manager which in turn handles (4) external Game Assets.

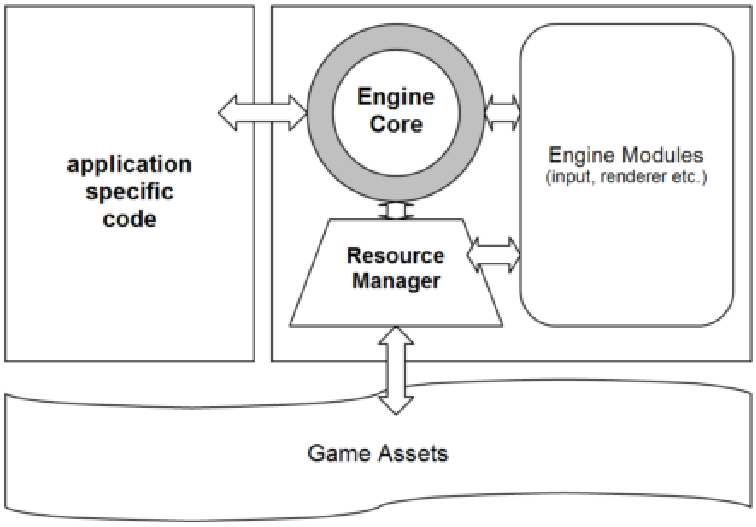

Figure 2: A conceptual model of a game engine (Anderson, 2011, p. 47)

As previously mentioned the term "game engine" has been identified as ambiguous from a research perspective (Anderson, 2011; Messaoudi et al., 2015). To understand the situation deeper, we have made an approach to capture the industry side of the term as well. This approach has also been taken by O'Donnell (2014) who dubs the game production pipeline as one of the most important parts of the game production process. The production pipeline is described by O'Donnell (2014, p. 72) as the *"...set of technologies, standards, and practices through which art assets and design data flow into the underlying game code"*.

For this research, we have chosen to investigate how common current game engines are described and marketed. Focusing on the engines from our game examples (*Assassin's Creed: Odyssey* and *Undertale*), as well as Unity and Unreal (the top results from the market research) we cover a wide range of usage cases and game engine categories.

On Unity's website, a general description of a game engine is provided:

> *A game engine is the software that provides game creators with the necessary set of features to build games quickly and efficiently. A game engine is a framework for game development that supports and brings together several core areas. You can import art and assets, 2D and 3D, from other software, such as Maya or 3s Max or Photoshop; assemble those assets into scenes and environments; add lighting, audio, special effects, physics and animation, interactivity, and gameplay logic; and edit, debug and optimize the content for your target platforms.[9] (Unity Technologies, 2018).*

This general definition of a game engine is consistent with what game engine definitions from the research have shown: a framework for game development where a plethora of functions to support game development is gathered. While this example is describing game engines in general, there is also a description of the Unity game engine itself:

> *The world's leading real-time creation platform. Unity is used to create half of the world's games. Our real time platform, powered by tools and services, offer incredible possibilities for game developers, and creators across industries and applications (Unity Technologies, 2018).*

Unreal Engine, the top result from the Steam data, is described as:

> *Unreal Engine is a complete suite of creation tools designed to meet ambitious artistic visions while being flexible enough to ensure success for teams of all sizes. As an established, industry-leading engine, Unreal delivers powerful, proven performance that you can trust (Epic Games, 2018).*

In the case of *Assassin's Creed: Odyssey*, where an in-house game engine has been used, it is more difficult to get information. The game engine for the *Assassin's Creed* series is called Anvil, and in its current iteration AnvilNext 2.0. Wikipedia has a page with some information, describing its iterations from the initial version developed for the first game *Assassin's Creed* (the engine was then called Scimitar) to its current state:

> *Claude Langlais, technical director of Ubisoft Montreal, says that modeling is done in 3ds Max for environment and ZBrush for characters. The engine uses Autodesk's HumanIK middleware to correctly position the character's hands and feet in climbing and pushing animations at run-time. […]In 2012 an updated version was released called AnvilNext; which was developed for Assassin's Creed III and beyond featuring a number of enhancements. Firstly, AnvilNext adds support for a new weather system, which allows for specific weather settings as well as an automatically cycling mode as seen in Assassin's Creed IV. Secondly, the renderer was rewritten for higher efficiency and support for additional post-processing techniques, enabling up to 3,000 non-playable characters to be rendered in real time (compared to the few 100s in the previous Anvil engine) […] More importantly, AnvilNext starting with Assassin's Creed Unity is capable of generating structures in a flexible and automatic manner while following specific design rules and templates, which reduces the amount of time and manual effort required for artists and designers to create an intricate urban environment. (Wikipedia, 2018)*

GameMaker, the engine used in *Undertale* and in many indie productions (as seen in the Itch.io data) is presented as:

> *Using a single development workflow GameMaker Studio 2 allows you export your game directly to Windows desktop, Mac OS X, Ubuntu, Android, iOS, fireTV, Android TV, Microsoft UWP, HTML5, PlayStation 4, and Xbox One (YoYo Games, 2018).*

The main difference in the GameMaker description lies in its focus on single development workflow. Compared to both Unity and Anvil, GameMaker does not include references to external tools, focusing more on the provided internal set of tools.

Tools, services, licensing models, reusability, modularity, performance/power and *quick iterations* are examples of features that are in focus in the general descriptions of the presented engines. Thus, based on these descriptions provided by game engine creators, it seems there is no (or little) confusion whether game engines are a production tool or a part of the game, as identified by Messaoudi et al. (2015).

MISUNDERSTANDINGS OF THE TERM GAME ENGINE

In this paper, we have presented an overview of roles in a game production, definitions of game engines and a number of used game engines in the game industry today. From this overview, we can see that:

- The scope of a game development project can vary dramatically (i.e. one man indie projects versus large scale triple-A productions)
- There are a multitude of development tools used referred to as game engines
- There is no uniform definition of scope in relation to the term game engine in a production pipeline

While this is problematic from a game production perspective, the communication within a game development project would benefit from having uniform definitions but it also seems that the loose use of the term game engine can be problematic from a publicity perspective as well. In an article on the gaming site Kotaku (Schreier, 2018) it is reported about the backlash from fans regarding the game *Fallout '76* from the game company Bethesda. Upon release, the game was receiving negative feedback from the player community regarding bugs and performance issues, and questions were raised about the game engine used would not be on par with technology used in other games released at the time. The article is describing how a spokesperson from Bethesda stated that they are continuing to use their tools even in forthcoming games and they are happy with their editor. The spokesperson means that their tools are allowing the developers to create content quickly and that they have implemented a new renderer as a part of the tool set used. The term game engine was not mentioned by the spokesperson in the statement. Still, there was a community backlash stating that their game engine was outdated, where Schreier (2018) points out that the critics did not grasp the complexity of a game production with its multitude of tools and competences. In the article, Schreier (2018) is arguing that *"An engine isn't a single program or piece of technology—it's a collection of software and tools that are changing constantly".* This example adds to the need of a uniform "game language" as proposed by Anderson et al. (2008), but in a much wider context – involving the community aspects of communication as well.

In addition, a specific example of how the ambiguous use of the game engine term can be impacting the production is described by gaming site Eurogamer (Yin-Poole, 2019). Eurogamer reports about the problems that the game studio Starbreeze encountered while developing the game *Overkill's The Walking Dead.* The article states that the management of Starbreeze focused a lot of resources to buy and further develop an in-house engine, *Valhalla*, with the intention to use the Valhalla engine in all Starbreeze games. Developers interviewed in the article state that the Valhalla game engine was essentially only a graphics renderer, with no tools attached to it: *"It was just not good. Like most engines, it had good potential, but it wasn't in a good place for people to properly develop a game. That was the problem. It was just way too far behind in the pipeline"* a developer stated (Yin-Poole, 2019). It was later decided that the Valhalla engine should be replaced with Unreal, but the decision came late in the development process and a lot of work had to be redone due to the switch of main production tools.

A PROPOSED TAXONOMY FOR TOOLS IN GAME PRODUCTION

The phrase *game engine* has been used for a very wide range of concepts related to game development. It has been used to represents *all* tools used to produce games. It has also been used to describe the core runtime of a *compiled* game. As discussed above, misconception of the term has led to confusions among both developers and players. To avoid this in the future, we propose a taxonomy for tools in game production that focus on the characteristic functions of tools.

In our proposed taxonomy, the *game production pipeline* is the core production process of a game where professionals from various disciplines create content that is assembled and combined into a game that can be executed on a target platform. This idea is in line with the findings presented by O'Donnell (2014). A game engine will be a part of this pipeline, but as pointed out by O'Donnell (2009) there are many additional components needed for an efficient game production. In the production pipeline, the connections between tools can be automated or manual. Related to Messaoudi et al. (2015) where they propose using the term "framework" as a description of the toolset to produce a game, we would rather call it a production pipeline. The production pipeline can be thought of as a factory assembly line, where many components are assembled by people with different roles and work task with tools relevant to their specific work task. As we identified in the case of *Assassin's Creed: Odyssey*, many roles with different work areas are involved in the production. The flow in the production pipeline is the data generated by the tools. This data can for example be represented as files (e.g. graphical assets) or database entries (e.g. dialog lines). An important part of the data is metadata that can be used to steer the flow and to assemble the game.

In our taxonomy (Figure 3) we make a distinction between tools that are part of the production pipeline (pipeline tools) and those that are not (non-pipeline tools). The non-pipeline tools are used for other processes involved in game production such as project planning, testing, marketing etc. It is outside the scope of this paper to go into any detail of non-pipeline tools. The only exception is that this group includes tools for making tools, something that is central for the plasticity of the pipeline. For the pipeline tools we propose that a distinction is made between three different types:

- **Product facing tools** (in the core engine, typically highly optimized)
- **User facing tools** (typically with GUIs designed for specific tasks)
- **Tool facing tools** (integration tools that connect different parts of the pipeline, middleware that adds certain functionality)

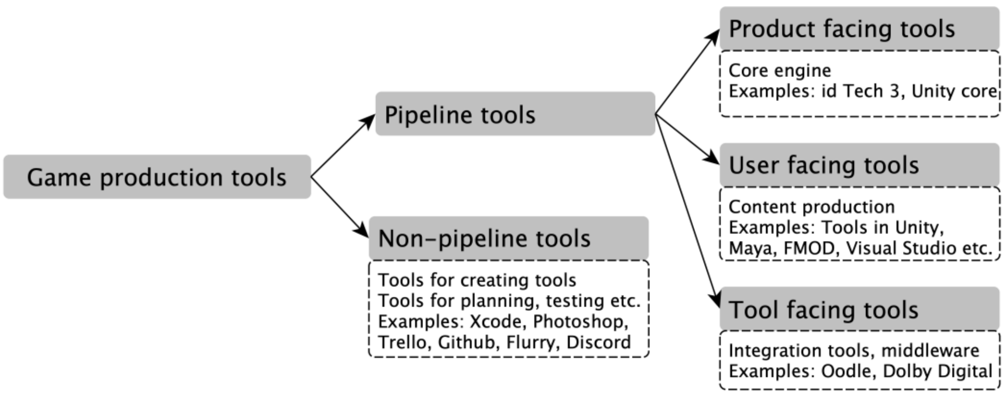

Figure 3: A proposed taxonomy of game production tools.

The product facing tools constitutes the core engine that handles the game simulation and compiles the game for a target platform. This corresponds to the upper, right rectangle in Figure 2. These components have received a lot of attention in the past since they typically handle complex tasks such as rendering, physics and AI. In our taxonomy we do not delve into any such details. The defining characteristic is that these tools are product facing, i.e. the target platform plays a central role. The building to a target platform is a complex process that involves transformations and bundling of both code and assets. A main goal with the product facing tool is performance and quality. These tools will hence have a strong focus on data structures, algorithms and optimization.

The user facing tools are designed to support human developers to create game content. Considering the wide range of competences involved in game production, these tools vary enormously: writing editors, 2D drawing tools, 3D modelling software, Integrated Development Environments (IDE) for programmers, audio mixers, etc. A main goal with user facing tools is to support the creativity and productivity of game developers. These tools will hence have a strong focus on usability.

The role of the tool facing tools is to create bridges between different tools in the production pipeline or to add functionality with a middleware. These tools play a central role for the efficiency in game production. They can be created as extensions, or plug-ins to the other types of tools (e.g. as an export module) or they can be standalone programs (e.g. a daemon) that translate and transfer data from one system to another. These tools hence have a strong focus on interconnectivity and functionality. Based on the defined tools we can define a game engine according to the following:

- A **core engine** is a collection of product facing tools used to compile games to be executed on target platforms. (Examples: id Tech 3, Unity core etc.)
- A **game engine** is a piece of software that contains a core engine and an arbitrary number of user facing tools.
- A **general purpose game engine** is a game engine targeted at a broad range of game genres (Examples: Unity, Unreal).
- A **special purpose game engine** is a game engine targeted at specific game genres. (Examples: GameMaker, Construct, Twine etc.)

Note that with these definitions, a core engine is a game engine without any user facing tools. This was the first type of engines, e.g. the first iterations of id Tech, which were provided as programming libraries. The definition also includes all game engines discussed above (e.g. Unity, Unreal, GameMaker). With our definition, a production pipeline of a game development project (Figure 4) contains a game engine and an arbitrary number of additional pipeline tools.

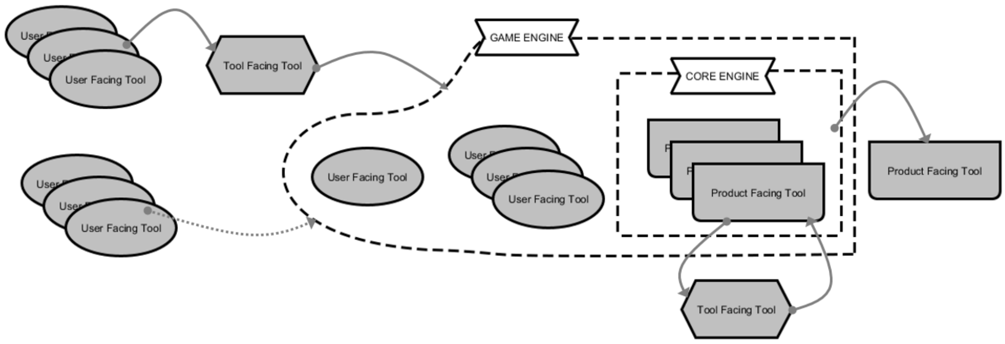

Figure 4: A model of the game production pipeline.

A typical characteristic of a game production pipeline, apparent from the quotes from Unity and Unreal above, is its plasticity – tools can be altered or added depending on the type of game being produced. This can, for example, be to create a design tool that enables modeling of the game levels of a particular game. Some of the tools used have this plasticity included in their native design. In other cases new tools are created. As an example, Unity has inherent functionality to create custom inspectors and panels for the various game components that are created. It also has support for different type of plugin extension from third party software through its asset store. Note that the game engine is drawn with a dashed line in Figure 4 indicating this plasticity. The border between the game engine and other tools in the production pipeline is not of any major importance. As long as the integration of the different tools is handled, they can be located inside the engine or outside.

Depending on the type of game engine, some parts of the production will be conducted with the user facing tools included in the game engine. Other parts will be running as separate applications. As long as the tool facing tools are present, it will not make a fundamental difference. Some connections between the user facing tools and other parts of the pipeline is done via the tool facing tools, while some connections might be done manually (represented by a dotted line in Figure 4). There may however be practical implications of having a user facing tool integrated in the game engine or provided as a standalone application. When many different types of user facing tools are integrated in the game engine, there is a risk that the user interface becomes bloated and the application will be large. With standalone tools the interface can be tailored for the task and the application can be lightweight. Another aspect, not addressed in the literature, is the risk associated with giving all developers access to all elements of the game. There is always a risk for accidental changes or changes from a developer that is unaware of the implications of the change.

Note that with our proposed taxonomy, the defining element of a game engine is the *core engine*. The typical game engine will in addition to the core engine contain a number of user facing tools. With our taxonomy, the general purpose game engine corresponds to the “überengine” proposed by Anderson et al. (2008). We do not put a requirement that the core engine should contain any particular elements. The core engine for Twine, for example, does not have any components for 3D rendering, physics etc. but is a relatively simple parser (Klimas, 2009). For a general purpose engine (such as Unity) the core will have substantially more and complex components.

For complexity reasons, a game engine is not likely to include the whole production pipeline. Only games developed in small teams in special purpose engines can handle the whole pipeline in a single application (for instance GameMaker). Even Unity that includes a lot of user facing tools, e.g. for animation and audio processing, depends on external tools such as Visual Studio for script editing.

It should be noted that this taxonomy is an abstraction and that a real production can include more complex interconnections. The same modules can for example be shared between tools. The tools outside the production pipeline can have dependencies on objects in the production pipeline, e.g. analytics tools have components in game binaries; planning tools relate to production entities, etc. It is also possible that the same tool, e.g. Photoshop, is used in both the production pipeline (to create textures) and in the non-production pipeline (to create promotion material).

DISCUSSION AND CONCLUSIONS

The origin of the study presented in this paper was an attempt to survey what support game engines have for localization. In that process, it was soon evident that this question was the wrong question to address. Instead, the presented study was conducted where we define the tools of a game production pipeline. We argue that everything should not or could not be supported in a general purpose engine. Still, game engine is a term that is used in the industry. While problematic in several cases, it is hard to look away from the fact that it has become a part of the "lingua franca" of game developers. The examples from the industry, reported by Kotaku (Schreier, 2018) and Eurogamer (Yin-Poole, 2019), show that it is important to be able to communicate clearly regarding the production tools. O'Donnell (2014) has addressed the issue and means that the pipeline is an aspect of game production that is least talked about outside game development circles. If this statement is seen in relation to the Kotaku and Eurogamer articles, it seems that there are confusions even within game development projects. The management of Starbreeze, based on the Eurogamer article, seemed to confuse a core engine (as Valhalla was reported to be a renderer only) with a production pipeline. The Kotaku article, where fans reacted to the change of "game engine" reflects the statement of O'Donnell (2014) – the production pipeline seems to be a concept too complicated for the layman, hence the use of the term "game engine" as a substitute.

Our approach to nuance the use of the game engine term and put it into a broader perspective will hopefully help to avoid, or at least minimize the risk of, confusion in the future – at least in a production setting. As we identified 692 separate roles in a triple-A game production setting, it is evident that no single piece of software can accommodate for all these perspectives.

The plasticity of the production pipeline is important and we believe that with our proposed taxonomy, this plasticity is mirrored. The tool types identified are both general and specific enough to be able to describe a large number of production settings. As previous research relating to game engines has pointed out (Anderson et al., 2008), a game engine can be produced to accommodate for a specific game genre or technical foundation of a game. The data flow through the production pipeline is worth to point out as an important factor here; the different tools are handling data in different ways and the interchange between tools can be more or less automated.

The presented study reveals the need for more close-to-industry research and it is a challenge to get reliable data, especially in the case of triple-A game production where legal matters (such as non-disclosure agreements) and the sheer scope of production makes it a daunting task to research. To get information regarding an in-house game engine such as Anvil, we had to use unreliable sources such as Wikipedia. The indie scene seems easier to get access to, due to a more open production environment (partly thanks to the tradition of sharing knowledge). We would like to point out the importance that the game research community takes the opportunity to create industry relevant research to create long-lasting bonds with game developers in all production setting and of all sizes.

Finally, coming back to the original intentions of the presented study; it is interesting to note that localization and translation is well represented among the professions involved in a triple-A production (Figure 1 and Table 2). The inherent support for localization is missing in Unity – one of the most popular general purpose engines. But this does however not mean that all productions using Unity lacks such tools in their production pipeline. Due to the possibility to expand the production pipeline with external tools or plugins, localization tools might not be a big problem. It however becomes a problem for inexperienced developers if they assume that Unity will handle the whole production pipeline. This was apparent in our previous case study with indie studios experiencing challenges when localization was added late in the production process (Toftedahl et al., 2018).

FUTURE RESEARCH

Since the original idea of the paper was to understand how built-in functions affects game production, we have some suggestions for future research. When identifying the commonly used available game engines, we noticed that the origins of them differ. Unreal has its origins as an in-house engine, made available publically at a later stage in its life cycle, while Unity was created as an openly available product from the beginning. How these different origins affected the included tools and pipeline connections is a question for future studies to analyze.

ACKNOWLEDGMENTS

This research is funded by the EU Interreg ÖKS project Game Hub Scandinavia 2.0.

BIBLIOGRAPHY

Anderson, E. F. (2011, 2011). *A Classification of Scripting Systems for Entertainment and Serious Computer Games.* Paper presented at the 2011 Third International Conference on Games and Virtual Worlds for Serious Applications.

Anderson, E. F., Engel, S., Comninos, P., & McLoughlin, L. (2008). *The case for research in game engine architecture.* Paper presented at the Proceedings of the 2008 Conference on Future Play Research, Play, Share - Future Play '08.

Blow, J. (2004). Game Development: Harder than you think. *Queue, 1*(10), 28. doi:10.1145/971564.971590

Bolding, J. (2019). Steam now has 30,000 games. https://www.pcgamer.com/steam-now-has-30000-games/: PC Gamer.

Egenfeldt-Nielsen, S., Smith, J. H., & Tosca, S. P. (2016). *Understanding video games: The essential introduction* (3 ed.): Routledge.

Epic Games. (2018). Unreal Engine [General Purpose Game Engine]. www.unrealengine.com: Epic Games.

Gregory, J. (2014). *Game engine architecture*: AK Peters/CRC Press.

Hagen, U. (2009). Where do Game Design Ideas Come From? Innovation and Recycling in Games Developed in Sweden. *Breaking New Ground: Innovation in Games, Play, Practice and Theory. Proceedings of DiGRA 2009*.

Klimas, C. (2009). Twine [Special Purpose Game Engine]. https://twinery.org.

Messaoudi, F., Simon, G., & Ksentini, A. (2015, 2015). *Dissecting games engines: The case of Unity3D*.

Murphy-Hill, E., Zimmermann, T., & Nagappan, N. (2014). *Cowboys, ankle sprains, and keepers of quality: How is video game development different from software development?* Paper presented at the Proceedings of the 36th International Conference on Software Engineering.

O'Donnell, C. (2009). The everyday lives of video game developers: Experimentally understanding underlying systems/structures. *Transformative Works and Cultures, 2*.

O'Donnell, C. (2014). *Developer's Dilemma: The Secret World of Videogame Creators*: The MIT Press.

O'Hagan, M., & Mangiron, C. (2013). *Game Localization: Translating for the global digital entertainment industry* (Vol. 106): John Benjamins Publishing.

Payne, M. T., & Steirer, G. (2014). Redesigning game industries studies. *Creative Industries Journal*, *7*(1), 67-71. doi:10.1080/17510694.2014.892292

Pereira, L. S., & Bernardes, M. M. S. (2018). Aspects of Independent Game Production: An Exploratory Study. *Comput. Entertain.*, *16*(4), 1-16. doi:10.1145/3276322

Schreier, J. (2018). The Controversy Over Bethesda's 'Game Engine' Is Misguided. https://kotaku.com/the-controversy-over-bethesdas-game-engine-is-misguided-1830435351: Kotaku.

Scirra. (2018). Construct [Special Purpose Game Engine]. www.construct.net: Scirra Ltd.

tobyfox. (2015). Undertale [Computer game]: tobyfox.

Toftedahl, M., Backlund, P., & Engström, H. (2018). *Localization from an Indie Game Production Perspective: Why, When and How?* Paper presented at the DiGRA '18 - Proceedings of the 2018 DiGRA International Conference: The Game is the Message, Torino, Italy.

Tschang, F. T., & Szczypula, J. (2006). Idea creation, constructivism and evolution as key characteristics in the videogame artifact design process. *European management journal*, *24*(4), 270-287.

Ubisoft. (2018). Assassin's Creed: Odyssey [Computer Game]: Ubisoft.

UBM. (2018). *2019 GDC State of the Game Industry*. Retrieved from http://reg.gdconf.com/GDC-State-of-Game-Industry-2019:

Unity Technologies. (2018). Unity [General Purpose Game Engine]. https://unity3d.com/unity.

Wang, A. I., & Nordmark, N. (2015). *Software architectures and the creative processes in game development.* Paper presented at the International Conference on Entertainment Computing.

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*(7), 2315-2332. doi:10.1177/1461444817715020

Wikipedia. (2018). AnvilNext. Retrieved 2019-01-25 https://en.wikipedia.org/wiki/AnvilNext

Yin-Poole, W. (2019). The fall of Starbreeze. https://www.eurogamer.net/articles/2019-01-28-the-fall-of-swedish-game-wonder-starbreeze: Eurogamer.

YoYo Games. (2018). GameMaker [Special Purpose Game Engine]. www.yoyogames.com: YoYo Games.

Zackariasson, P., Styhre, A., & Wilson, T. L. (2006). Phronesis and creativity: Knowledge work in video game development. *Creativity and Innovation Management*, *15*(4), 419-429.

ENDNOTES

[1] https://store.steampowered.com/sale/2018_so_far_top_sellers/, accessed 2019-01-20

[2] https://store.steampowered.com/sale/2016_top_sellers/, accessed 2019-01-20

[3] https://itch.io/game-development/engines/most-projects, accessed 2018-12-20

[4] https://steamdb.info/

[5] https://steamspy.com/

[6] https://github.com/limdingwen/Steam-Engines, accessed 2018-12-20

[7] https://www.lexaloffle.com/pico-8.php, accessed 2018-01-16

[8] http://www.rpgmakerweb.com/, accessed 2018-01-16

[9] https://unity3d.com/what-is-a-game-engine, accessed 2018-01-15